DESIGN OF ARITHMETIC AND LOGIC UNIT FOR THE MICROPROCESSOR EMU544 BY USING XILINX SOFTWARE

Yaseer Arafat DURRANI

Undergraduate Project Report
Submitted in partial fulfillment of
The requirements for the
Degree of Bachelor of Science (B.S.)

in

Electrical and Electronic Engineering Department
Eastern Mediterrenean University

June 1999

Approval of the Electrical and Electronic Engineering Department

Assoc. Prof. Dr. Dervis Deniz
Chairman

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in cope and quality, as an Undergraduate Project.

Assist. Prof. Dr. Stephane Le Goff

……………………………………..
Supervisor

Members of the examining committee.

Name	Signature
Assist. Prof. Dr. Stephane Le Goff	……………………………………..
Prof. Dr. Suha Bayindir	……………………………………..
Assoc. Prof. Dr. Osman Kukrer	……………………………………..
Assist. Prof. Dr. Huseyin Ozkaramanli	……………………………………..

Date: …………………………….

Abstract

DESIGN OF ARITHMETIC AND LOGIC UNIT FOR THE MICPROCESSOR EMU544 BY USING XILINX SOFTWARE

by
Yaseer Arafat DURRANI

Electrical and Electronic Engineering Department
Eastern Mediterrenean University

Supervisor: Assist. Prof Dr. Stephane Le Goff

Keywords: VLSI, ALU, Xilinx, Microprocessor, Adders, Subtracters.

The arithmetic logic unit (ALUs) are combinational logic circuits that can perform basic arithmetic (addition or subtraction) or logical (AND, OR, NOT, etc.) operations on two *m*-bit operands. ALUs can be constructed from standard integrated, circuits or programmable logic devices. We can perform the design of ALU in three stages. First, we design the arithmetic section, then logic and finally, we can combine the two sections to form the ALU.

All digital computers contain logic circuits to implement selected arithmetic operations in the particular number systems selected for use in those computers. The most commonly used number system for representing integers is the two's complement number system, because it simplifies the representation of both positive and negative values and the implementation of addition and subtraction circuits.

One of the microprocessor's major logic function is the ALU. It contains the microprocessor's data processing logic.

June 1999

To my Allah

AKNOWLEDGEMENT

I would like to acknowledge and express my deep gratitude to my supervisor Assist. Prof. Dr. Stephane Le Goff for his numerous and valuable comments and suggestions in my study. I would like also to express my appreciation to all my teachers, who helped me to complete my B.S study in Electrical and Electronic Engineering.

I would like to acknowledge with my God and with my family, especially my parents for encouragement and enthusiasm to proceed and complete my BS study.

Table of Contents

Chapter 1
INTRODUCTION

The arithmetic and logic units (ALUs) are combinational circuits that perform various operations on data. The type of data that it can manipulate and the set of operations that it can perform on those data characterize each ALU.

Most ALUs support operations on integers of various sizes and may also include operations to manipulate fixed-point and floating numbers and various nonnumeric data. Typical ALU operations include the following:

- Arithmetic: add, subtract, multiply, divide.
- Logic: AND, OR, exclusive-OR, complement.
- Shift and rotate data.
- Convert data from one type to another.

Under these operations in the microprocessor RISC EMU544 ALU is designed and simulated in Xilinx foundation software, which can be implemented into a chip.

The theory about ALU and the basic functions that how it performs, design and simulation results are shown in the report.

Chapter 2

IC TECHNOLOGY AND SYSTEM DESIGN CONSIDERATIONS

2.1 Overview of IC Technology

The transistor was invented in 1947 at Bell Laboratory. In the 1950s, transistors replace vacuum tubes in many electronics systems, including computers. It was until 1959 that the first integrated circuit was successfully fabricated and tested by Jack Kilby of Texas Instruments. Prior to the invention of the IC, the use of transistors, along with other discrete components such as capacitors and resistors, was common in computer design. Early transistors were made of germanium, which was later abandoned in favor of silicon. This was due to the fact that the slightest rise in temperature resulted in massive current flows in germanium-based transistors. In semiconductor terms, it is a massive flow of electrons from the valence band to the conduction band when the temperature rises even slightly. By the late 1960s and early 1970s, the use of the silicon-based IC was widespread in mainframe and in minicomputers. Transistors and ICs were based on P-type materials. Due to the fact that the speed of electrons is much higher (about two and a half times) than the speed of the hole, N-type devices replaced P-type devices.

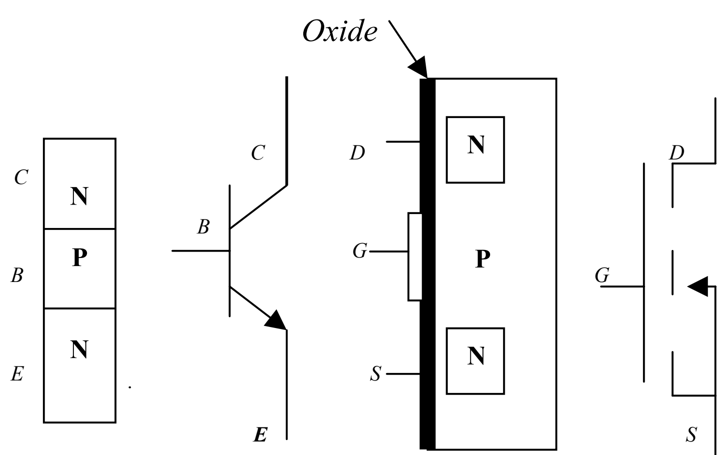

Fig .2.1. Bipolar NPN Transistor NMOS Transistor

By the mid-1970s, NPN and NMOS transistors had replaced the slower PNP and PMOS transistors in every sector of the electronics industry, including in the design of microprocessors and computers. Since the early 1980s, CMOS has become the dominant method of IC design.

2.2 MOS vs Bipolar Transistors

There are two type of transistors: bipolar and MOS (metal-oxide semiconductor). Both have three leads. In bipolar transistors, the three leads are referred to as the emitter, base, and collector, while in MOS transistors they are named source, gate, and drain. In bipolar the carrier flows from the emitter to the collector and the base is used as a flow controller. In MOS, the carrier flows from the source to the drain and the gate is used as a flow controller. In NPN-type bipolar transistors, the electron carrier leaving the emitter must overcome two voltage barriers before it reaches the collector as shown in Fig.2.1.

One is the N-P junction of the emitter-base and the other is the P-N junction of the base-collector. The voltage barrier of the base collector is the most difficult one for the electron to overcome (since it is reversed biased) and it causes the most power dissipation. This led to the design of the unipolar type transistor called MOS. In N-channel MOS transistors, the electrons leaves the source reaching the drain without going through any voltage barrier.

The low power dissipation of MOS allows putting millions of transistors on a single IC chip. In today's million-transistor microprocessors and DRAM memory chips, the use of MOS technology is indispensable. Without the MOS transistor, the advent of desktop personal computers would not have been possible, at least not so soon. The use of bipolar transistors in both the mainframe and minicomputer of the 1960s and 1970s required expensive cooling systems and large rooms due to their bulkiness. MOS transistors do have one major drawback: They are slower than bipolar transistors. This is due partly to the gate capacitance of the MOS transistor. For MOS to be turned on, the input capacitor of the gate takes time to charge up to turn-on (threshold) voltage, leading to a longer propagation delay.

2.3 Overview of Logic Families

Logic families are judged according to

1). Speed

2). Power dissipation

3). Noise immunity

4). Input/Output interface compatibility

5). Cost

Desirable qualities are high speed, low power dissipation, and high noise immunity (since it prevents the occurrence of false logic signals during switching transition). This means that high-driving-capability outputs are desired. This plus the fact that the input and output voltage levels of MOS and bipolar transistors are not compatible mean that one must be concerned with the ability of one logic family in driving the other one. In terms of the cost of a given logic family, it is high during the early years of its introduction and prices decline as production and use rise.

2.4 The Case of Inverters

In a one-transistor inverter, the transistor plays the role of a switch, R is the pull-up resistor. We can see in Fig.2.2,

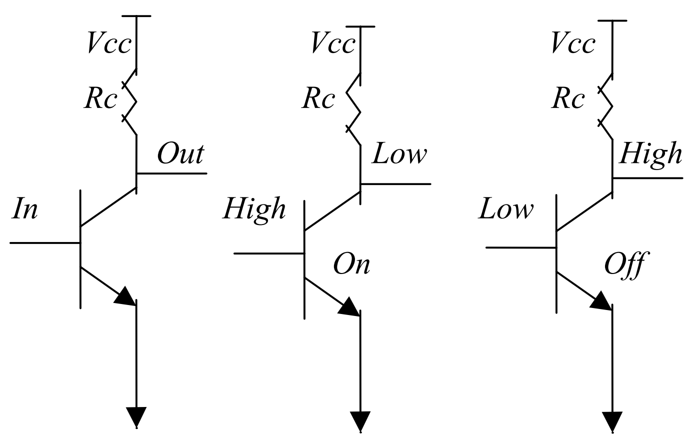

Fig.2.2.One-Transistor Inverter with Pull-up Resistor.

However, for this inverter to work effectively in digital circuits, the R value must be high when the transistor is "on" to limit the current flow from Vcc to ground in order to have low power dissipation ($P=VI$, where $V=5V$). In other words, the lower the I, the lower the power dissipation. One the other hand, when the transistor is "off", R must be a small value to limit the voltage drop across R, thereby making sure that Vout is close to Vcc. This is a contradictory demand on R. This is one reason that logic gate designers use active components (transistors) instead of passive components (resistors) to implement the pull-up resistor R.

2.5 CMOS Inverter

In the case of CMOS-based logic gates, PMOS and NMOS are used to construct a CMOS inverter as shown in Fig.2.3,

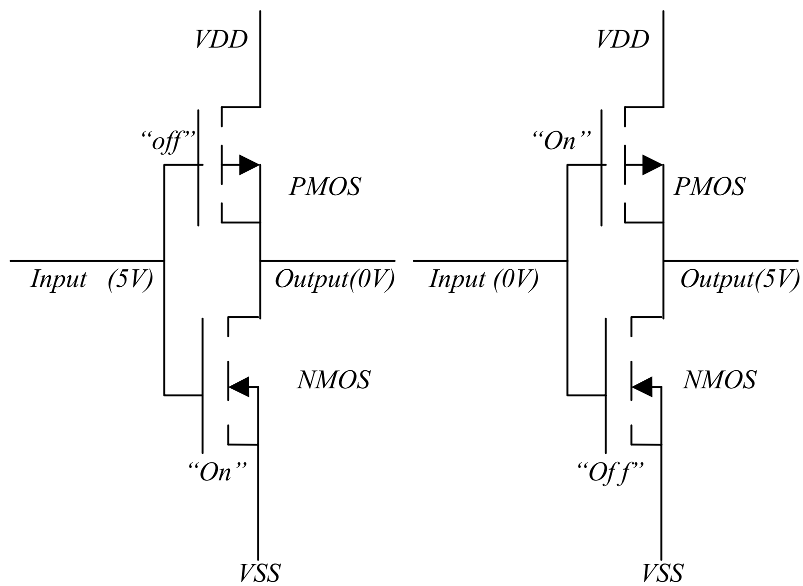

Figure.2.3. CMOS Inverter

In CMOS inverters, when the PMOS transistor is off, it provides a very high impedance path, making leakage current almost zero (about 10*n*A); when the PMOS is on, it provides a low resistance on the path of VDD to load. Since the speed of the hole is slower than the electron, the PMOS transistor is wider to compensate for this disparity.

2.6 Input, Output Characteristics of Some Logic Families

In 1968 the first logic family made of bipolar transistors was marketed. It was commonly referred to as the standard TTL family. The MOS based logic family, the CD 4000/74C series, was marketed in 1970. Schottky diode to the base-collector of bipolar transistors in the early 1970s gave rise to the S-family. Schottky diode shortens the propagation delay of the TTL family by preventing the collector from going into what is called *deep saturation.*

Table.2.1 Characteristic of some logic families

Characteristic	Std TTL	LSTTL	ALSTTL	HCMOS
Vcc	5 V	5 V	5 V	5 V
VIH	2.0 V	2.0 V	2.0 V	3.15 V
VIL	0.8 V	0.8 V	0.8 V	1.1 V
VOH	2.4 V	2.7 V	2.7 V	3.7 V
VOL	0.4 V	0.5 V	0.4 V	0.4 V
IIL	-1.6 mA	-0.36 mA	-0.2 mA	-1 μA
IIH	40 μA	20 μA	20 μA	1 μA
IOL	16 mA	8 mA	4 mA	4 mA
IOH	400 μA	-400 μA	-400 μA	4 mA
Propagation delay	10 ns	9.5 ns	4 ns	9 ns
Static power dissipation (F=0)	10 mW	2 mW	1 mW	0.0025 nW
Dynamic power dissipation at F=100KHz	10 mW	2 mW	1 mW	0.17 mW

Table.2.1, lists major characteristics of some logic families. In Table.2.1, it is noted that as the CMOS circuit's operating frequency rises the power dissipation also increases. This is not the case for bipolar-based TTL .

2.7 History of Logic Families

Early logic families and microprocessors required both positive and negative power voltages. In the mid-1970s, 5V Vcc became standard. For example, Intel's 4004, 8008, & 8080 all used negative and positive voltages for the power supply. In the late 1970s, advances in IC technology allowed combining the speed and derive of the S-family with the lower power of LS to form a new logic family called FAST (Fairchild advanced schottky TTL). In 1985, AC/ACT (advanced CMOS technology), a much higher speed version of HCMOS was introduced. With the introduction of FCT (fast

CMOS technology) in 1986, at last the speed gap between CMOS and TTL was closed. Since FCT is the CMOS version of FAST, it has a low power consumption of CMOS but the speed is comparable with TTL. Table.2.2, provides an overview of logic families up to FCT.

Table.2.2 Logic family overview

Product	Year Introduced	Speed (ns)	Static Supply Current (mA)	High/Low Family Drive (mA)
Std/TTL	1968	40	30	-2/32
CD4K/74C	1970	70	0.3	-0.48/6.4
LS/S	1971	18	54	-15/24
HC/HCT	1977	25	0.08	-6/-6
FAST	1978	6.5	90	-15/64
AS	1980	6.2	90	-15/64
ALS	1980	10	27	-15/64
AC/ACT	1985	10	0.08	-24/2
FCT	1986	6.5	1.5	-15/64

2.8 Recent Advances in Logic Families

As the speed of high-performance microprocessors such as the 386 and 486 reached 25MHz, it shortened the CPU's cycle time, leaving less time for the path delay. There must be a corresponding decline in the propagation delay of logic families used in the address and data path as the system frequency is increased. In recent years, many semiconductor manufacturers have responded to this need by providing logic families that have high speed, low noise, and high drive.

Table.2.3 Advanced logic general characteristics

Family	Year	Number Supplies	Tech Base	I/O LEVEL	Speed	Static Current	IOH/IOL
ACQ	1989	2	CMOS	CMOS/CMOS	6.0 ns	80 μA	-24/24 mA
ACQ	1989	2	CMOS	TTL/CMOS	7.5 ns	80 μA	-24/24 mA
FCTx	1987	3	CMOS	TTL/CMOS	4.1-4.8 ns	1.5 mA	-15/64 mA
TCTxT	1990	2	CMOS	TTL/TTL	4.1-4.8 ns	1.5mA	-15/64 mA
FASTr	1990	1	Bipolar	TTL/TTL	3.9 ns	50 mA	-15/64 mA
BCT	1987	2	BICMOS	TTL/TTL	5.5 ns	10 mA	-15/64 mA

Table.2.3, provides the characteristics of high-performance logic families introduced in recent years. ACQ/ACTQ are the second-generation advanced CMOS (ACMOS) with much lower noise. While ACQ has the CMOS input level, ACQT is equipped with TTL-

level input. The FCTx and FCTx-T refers to various speed grades, such as A,B, and C where A designation means low speed and C means high speed. For designers who are well versed in using the FAST logic family, the use of FASTr is an ideal choice since it is faster than FAST, has higher driving capability (IOL , IOH), and produces much lower noise than FAST. At the time of this writing, next to ECL and gallium arsenate logic gates, FASTr is the fastest logic family in the market (with the 5Vcc), but the power consumption is high related to other logic families

Table.2.4 Importance of speed

System Clock Speed (MHz)	Clock Period (ns)	Predominant Logic for Path
2--10	100—500	HC,LS
10--30	33—100	ALS,AS,FAST,FACT
30--66	15—33	FASTr,BCT,FCTA

as shown in Table2.3. Recently, a 3.3V Vcc with higher speed and lower power consumption is starting to appear. The combing of high-speed bipolar TTL and low power consumption of CMOS has given birth to what is called BICMOS. Although BICMOS seems to be the future trend in IC design, at this time it is expensive due to extra steps required in BICMOS IC fabrication but in some cases there is no other choice. For example, Intel's Pentium microprocessor, a BICMOS product, had to use high-speed bipolar transistor to speed up some of the internal functions in order to keep up with RISC processor performance. Table.2.3, provides advanced logic charateristics. Table.2.4, shows logic families are used in systems with different speeds. The x is for different speed where A,B and C are used for designation. A is the slower one while C is the fastest one. The above data is for the '244 buffer.

2.9 Evolution of IC Technology in Intel's 80x86 Microprocessors

Since 1971, when Intel introduced the first microprocessor, the 4004, until the introduction of the Pentium microprocessor, IC technology has gone through massive changes. The early processors (4004 and 8008) used PMOS. The 8080, 8085, 8088,and 80286 all used NMOS when first introduced. In recent years, CMOS versions of the 8088,8086, and 286 have been introduced for power-efficient systems. Currently, CMOS is the universal technology in the design of microprocessors. Only CMOS could allow designers to put over 3 million transistors on a single chip, make it work at 100 MHz, and consume around 10 watts of power. There has been a steady decline in the transistor's dimension throughout the 1970s, 1980s and 1990s. The design rule, the thickness of the lines inside the IC, has come down from a few microns to fraction of a microns during this time as shown in Table.2.5.

Table.2.5 Intel microprocessor evolution

Microprocessor	Year	Line Thickness(μm)	Power Supply (V)	Number of Transistors	IC Tech
8086	1978	3	5	29,000	NMOS
80286	1982	1.5	5	130,000	NMOS
80386	1985	1.5	5	275,000	CMOS
80486	1989	1	5	1.2 million	CMOS
Pentium	1992	0.8	5	3.1 million	BICMOS
Pentium	1993	0.6	3.3	3.1 million	BICMOS

The early microprocessors used power supplies with negative and positive voltages. For example, the 4004 used –10 and +50V. The 8008 used –9 and +5V, and the 8080 used –5,+5,+12V. Since the introduction of the 8085, the use of a +5V power supply has become standard in all microprocessors. To reduce power consumption, 3.3V Vcc is being embraced by many designers. The lowering of Vcc to 3.3V has two major advantages:

1. It lowers the power consumption, resulting in prolonging the life of the battery in systems such as a laptop PC or hand-held personal digital assistant.
2. It allows a further reduction of line size (design rule) to submicron dimensions. This reduction results in putting more transistors in a given die size. The decline in the size is expected to reach 0.1 micrometer by the year 2000 and transistor density per chip will reach 100 million transistors.

2.10 Power Dissipation Considerations

Power dissipation of a system is a major concern of system designers, especially for laptop and hand-held systems such as personal digital assistants (PDA). Although power dissipation is a function of the total current consumption of all components of a system, the impact of Vcc is much more pronounced.

A pin of an IC has an input capacitance of 5 to 7pF. This means that a single output that drives many inputs sees a large capacitance load since the inputs are parallel and therefore added together.

$$P=CFV^2 \tag{2.1}$$

In equation (2.1), the effects of frequency and Vcc voltage should be noted. While the power dissipation goes up linearly with frequency, the impact of the power supply voltage is much more pronounced (squared).

2.11 Dynamic and Static Currents

There are two major types of currents flowing through an IC: dynamic and static. A dynamic current is a function of the frequency under which the component is working. This means that a frequency goes up, the dynamic current and power dissipation go up. The static current, also called dc, is the current consumption of the component when it is inactive (not selected).

2.12 ECL and Gallium Arsenide (GaAs) Chips

The use of EDC (error detection and correction) in systems with speeds of 66MHz and higher is adding to the data and address path delay. This is forcing designers to resort to using ECL and GaAs chips. Due to the fact that ECL chips have a very high power dissipation, they are not used in PC/Workstation design. However, GaAs chips are showing up in high-speed Pentium and RISC-based computers. This is especially the case for the GaAs EDC and cache controller chips. The mass of electrons in GaAs is lighter than silicon, due to its quantum mechanics structure. As a result, the electrons in GaAs have a much higher speed. This power dissipation of the GaAs transistor is compatible to silicon-based MOS transistor. Therefore, GaAs technology might appear to provide the ideal chip since it has the speed of ECL (it is even faster than ECL) and the power dissipation of CMOS. However it has the following disadvantages.

1. Unlike silicon, of which there is plentiful supply in nature in the form of sand, GaAs is a rare commodity, and therefore more expensive.

2. GaAs is a compound made of two materials, Ga and As, and therefore is unstable at high temperatures.
3. It is very brittle, making it impossible to have large wafers. As a consequence, at this time no more than 100,000 transistors can be placed on a single chip. Contrast this to the millions of transistors for silicon-based chips.
4. The GaAs yields are much lower than silicon, making the cost per chip much more expensive than silicon chips.

These problems make the building of an entire computer based on GaAs a visionary product, if not an impossible one. This is now the case for the CRAY III supercomputer, which is based on GaAs and runs at speeds of 1GHz .

Chapter 3

BASIC COMPUTER SYSTEM ORGANIZATION

Every computer contains five elements or units:

1. The arithmetic-logic unit (ALU)
2. The memory unit
3. The input unit
4. The control unit
5. The output unit

The basic interconnection of these units is shown Fig.3.1.

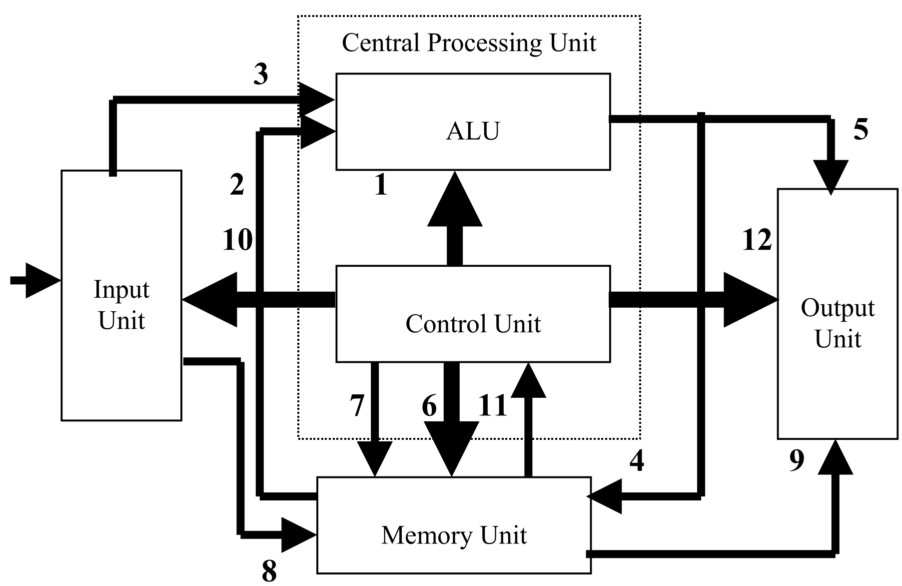

Fig.3.1. Basic Computer Organization

The arrows in this diagram indicate the direction in which data, information, or control signals are flowing. Two different-size arrows are used; the larger arrows represent data or information that actually consists of a relatively large number of parallel lines, and the smaller arrows represent control signals that are normally only one or a few lines. The various arrows are also numbered to allow easy reference to them in the following descriptions.

3.1 Arithmetic Logic Unit

The ALU is the area of the computer in which arithmetic and logic operations are performed on data. The type of operation that is to be performed is determined by signals from the control unit (arrow 1). The data that are to be operated on by the ALU can some from either the memory unit (arrow 2) or the input unit (arrow 4). Results of operations performed in the ALU can be transferred to either the memory unit for storage (arrow 4) or the output unit (arrow 5).

3.2 Memory Unit

The memory stores groups of binary digits (words) that can represent instructions (program) that the computer is to perform and the data that are to be operated on by the program. The memory also serves as storage for intermediate and final results of arithmetic operations (arrow 4). Operation of the memory is controlled by the control unit (arrow 6) , which signals for either a read or a write operation. A given location in memory is accessed by the control unit that provides the appropriate address code (arrow 7) . Information can be written in to memory from the ALU or the input unit (arrow 8), again under control of the control unit. Information can be read from memory into the ALU (arrow 2) or into the output unit (arrow 9).

3.3 Input Unit

The input unit consists of all of the devices used to take information and data that are external to the computer and put them into the memory unit (arrow 8) or the ALU (arrow 3). The control unit determines where the input information is sent (arrow 10). The input unit is used to enter the program and data into the memory unit prior to starting the computer. This unit is also used to enter data into the ALU from an external device during the execution of a program. Some of the common input devices are keyboards, toggle switches, teletypewriters, punched-card readers, magnetic disk units, magnetic tap units, and analog-to-digital converters (ADCs).

3.4 Output Unit

The output unit consists of the devices used to transfer data and information from the computer to the “outside world”. The output devices are directed by the control unit (arrow 12) and can receive data from memory (arrow 9) or the ALU (arrow 5); the data are then put into appropriate form for external use. Examples of common output devices are LED readouts, indicator lights, printers, disk or tape units, video monitors, and digital-to-analog converters (DACs).

As the computer executes in program, it usually has results or control signals that it must present to the external world. For example, a computer system might have a line printer as an output devices. Here the computer sends out signals to print out the results on paper. A small microcomputer might display its results on indicator lights or on LED displays.

3.5 Interfacing

The most important aspect of the I/O units involves interfacing, which can be defined as the joining of dissimilar devices in such a way that they are able to function in a compatible and coordinated manner. Computer interfacing is more specifically defined as the synchronization of digital information transmission between the computer and external input/output devices.

Many I/O devices are not directly compatible with the computer because of differences in such characteristics as operation speed, data format (e.g., hex, ASCII, binary), data transmission mode (e.g., serial, parallel), and logic signal level. Such I/O devices require special interface circuits, which allow them to computer system. A common example is the video display terminal (VDT), which can operate both as input and an output device. The VDT transmits and receives data serially (one bit at a time) while most computers handle data in parallel form. Thus, a VDT requires interface circuitry in order to send data to or receive data from a computer.

3.6 Control Unit

The function of the control unit should now be obvious. It directs the operation of all the other units by providing timing and control signals. In a sense, the control unit is like the conductor of an orchestra, who is responsible for keeping each of the orchestra members in proper signals necessary to execute each instruction in a program.

The control unit fetches an instruction from memory by sending an address (arrow 7) and a read command (arrow 6) to the memory unit. The instruction word stored at the memory location is then transferred to the control unit (arrow 11). This instruction word, which is in some form of binary code, is then decoded by logic circuitry in the control unit to determine which instruction is being called for. The control unit used this information to generate the necessary signals for executing the instruction.

3.7 Central Processing Unit (CPU)

From the Fig.3.1, the ALU and control units are shown combined into one unit called the central processing unit (CPU). This is commonly done to separate the actual “brains” of the computer from the other units.

Chapter 4

INTRODUCTION TO MICROPROCESSOR

4.1 The Evolution of the Microprocessors

History shows us that the ancient Babylonians first began using the *abacus* (a primitive calculator made of beads) in bout 500 BC This simple calculating machine eventually sparked humankind into the development of calculating machinery that used gears and wheels (Blaise Pascal in 1642). Refinements continued with the giant computing machines of the 1940s and 1950s constructed with relays and vacuum tube. Nextthe transistor and solid-state electronics were used to build the mighty computers of the 1960s. Finally, the advent of the integrated circuit led to the development of the microprocessor and microprocessor-based computer system.

4.1.1 The 4-bit microprocessor

The 1971, Intel corporation and creative talents of Marcian E. Hoff released the world's first microprocessor-the 4004, a 4-bit microprocessor. This integrated, programmable controller on a chip was meager by today's standards, because it only addressed 4,096 4-bit memory locations. The 4004 contained an instruction set that offered only 45 different instructions. As a result, the 4004 could be used only in limited applications such as early video games and small microprocessor-based controllers. When more sophisticated applications emerged for the microprocessor, the 4004 proved inadequate.

4.1.2 The 8-bit microprocessor

Later in 1971, realizing that the microprocessor was commercially viable product, Intel Corporation released the 8008-the 8-bit microprocessor. The expanded memory size (16K x 8) and additional instructions (a total of 48) in this new microprocessor provided by opportunity for many more advanced applications. (1K is equal to 1,024 and a byte is an 8-bit number).

As engineers developed more demanding uses for the microprocessor, the still relatively small memory and instruction set of the 8008 soon limited its usefulness. Thus, in 1973, the Intel Corporation introduced the 8080-the first of the modern 8-bit microprocessor. Soon other companies began releasing their own versions of the 4 and 8-bit microprocessors. Table .4.1, shows the lists of many early microprocessors.

Table.4.1 Early 8-bit microprocessors

Manufacturer	Part Number
Fairchild	F-8
Intel	8080
MOS Technology	6502
Motorola	MC6800
National Semiconductor	IMP-8
Rockwell International	PPS-8

But what was special about the 8080? Not only did it address more memory and execute more instructions, but also it executed instructions ten times faster than the 8008. An addition that took 20 microseconds on an 8008-based system took only 2.0 microseconds on an 8080-based system. Also, the 8080 were transistor-transistor logic (TTL) compatible, which meant it could be easily interfaced to standard TTL logic components. All these advantages ushered in the era of the 8080, and the ever-expanding era of the *microprocessor.*

The 8085, was introduced by Intel Corporation in 1977. Only slightly more advanced than the 8080, the 8085 addresses the same amount of memory, executes about the same number of instructions, and adds in 1.3 micro-seconds instead of 2.0 micro-seconds. The main advantages of the 8085 are its built-in clock generator and system controller, which were external components in the 8080-based system. Intel alone has sold well over 100 million copies of the 8085 microprocessor. Other companies, such as NEC, AMD, Toshiba and Hitachi, also manufacture a licensed version of the 8085 microprocessor. These features have made the 8085 one of Intel's most popular microprocessors.

4.1.3 The 16-bit microprocessor

In 1978, Intel Corporation released the 8086 microprocessor and about a year later, the 8088. Both devices are 16-bit microprocessors that execute instructions in as little as 400 nano-seconds, a vast improvement over the execution speed of the 8085. The 8086 and 8088 are also capable of addressing a 1M byte (8-bit wide) or a 512K word (16-bit wide) memory. The higher execution speeds and larger memory sizes allow the 8086 and 8088 to replace smaller minicomputers in many applications.

One important need that spurred the evolution of the 16-bit microprocessor is hardware multiplication and division. These functions are not available on most of the 8-bit microprocessors, with the exception of the Motorola MC6809, which can multiply, but no t divide. But the 16-bit microprocessor evolved for other reasons as well. It provides a larger addressable memory space than the 8-bitmicroprocessor, allowing it to perform some very sophisticated operations that just didn't fit into 64K bytes of memory. The 8086 and 8088 have a large number of internal registers, which are accessible in 200 nano-seconds compared to the 800 nano-seconds it takes to reach a register on an 8-bit microprocessor.

These additional registers allow software to be written far more efficiently. Finally, software application programs (databased management systems, spreadsheets, word processors and spelling checkers) began to require more than the 64K bytes of memory available on the 8-bit microprocessor. The time was ripe for the 16-bit microprocessor.

Evolution of the 16-bit microprocessor did not end with the 8086 and 8088; it continued with the introduction of the 80186, a highly integrated version of the 8086. The 80186 is today one of Intel's more popular 16-bit microprocessors. The 80186 are used in many control system applications, but not as the main microprocessor in personal computer systems. If the 80186 is found in a personal computer, it is on a plug-in daughter board that may control a hard disk memory system or a communications interface.

The last 16-bit microprocessor developed by Intel is the 80286, an improved version of the 8086 that contains a memory-management unit and addresses a 16M byte memory instead of a 1M byte memory. The clock speed of the 80286 also increased to

16MHz on the latest version produced by Intel. The basic version of the 8086 and 8088 executed up to 2.5 MIPs (millions of instructions per second), while the basic version of the 80286 executes up to 8 MIPs.

4.1.4 The 32-bit microprocessor

The most recent version of the microprocessor is the 32-bit microprocessor. Table.4.2 shows the list of all Intel microprocessors.

Intel currently produces two main versions: the 80386 and 80486. The 80386 were the first 32-bit microprocessor produced by Intel. The main advantages of this device are a much higher clock frequency (33MHz for the 80386 and 66MHz for the double-clocked version of the 80486) and a much larger memory spaces (4G bytes).

Table.4.2 Intel microprocessor

Part Number	**Data Bus Width**	**Memory Size**
8048	8	2K internal
8051	8	8K internal
8085A	8	64K
8086	16	1M
8088	8	1M
8096	16	8K internal
80186	16	1M
80188	8	1M
80286	16	16M
80386DX	32	4G
80386SL	16	32M
80386SX	16	16M
80486DX	32	4G
80486SX	32	4G
Pentium	32/64	4G

The 80486 microprocessor basically contains an improved 80386, an arithmetic coprocessor (for the DX version of the 80486), and an 8K byte internal cache memory. The 80386 execute many instructions in 2 clock, while the 80486 execute many instructions in 1 clock. These improvements combined with a 66MHz clock (80486DX2) allow instructions to execute at 54 MIPs according to Intel Corporation. This compares to the 8085, released 12 years prior to the 80486, that executes instructions at the rate of about 0.5 MIPs. These speed improvements will continue with newer versions of the 32-bit microprocessor as they become available. The next generation (Pentium) promised speeds of 1000 MIPs.

4.2 Microprocessor RISC EMU544

The general architecture of Microprocessor RISC EMU544 that has been designed in our department which is shown in Fig.4.1. This microprocessor has got 26 instructions. We are going now to describe in details all these instructions.

4.2.1 First class of instructions (contains 8 instructions)

(# register) op (Accu) $\longrightarrow$ *(Accu)*

The operation of the ALU has 8 possible operations, which are used in microprocessor EMU544, as shown below.

The 4 Arithmetic operations are

1. F = A + B
2. F = A +1
3. F = A – B
4. F = A – 1

The 4-Logic operations are

1. F = A and B
2. F = A or B
3. F = A
4. F = $\overline{A}$

The layout diagram of the environment of the ALU is shown in Fig.4.2.

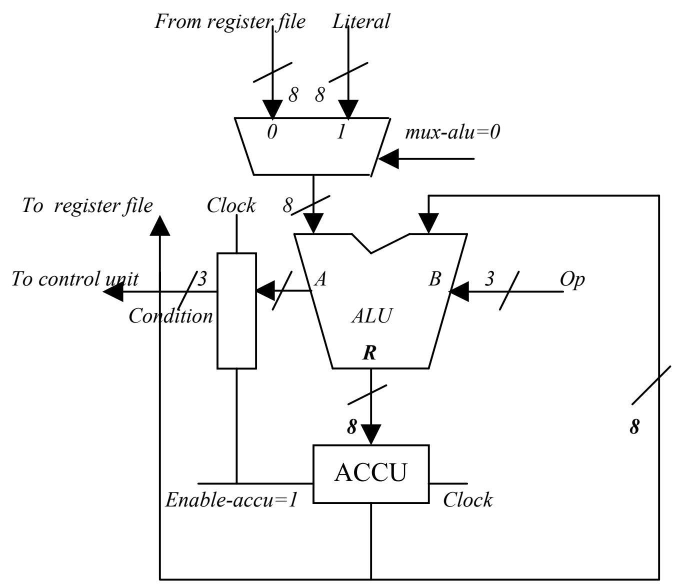

Fig.4.2 The ALU

The number of registers represents the content of a register chosen among 8 possible registers. These 8 registers form what is called the “register file”, which is shown in Fig.4.3.

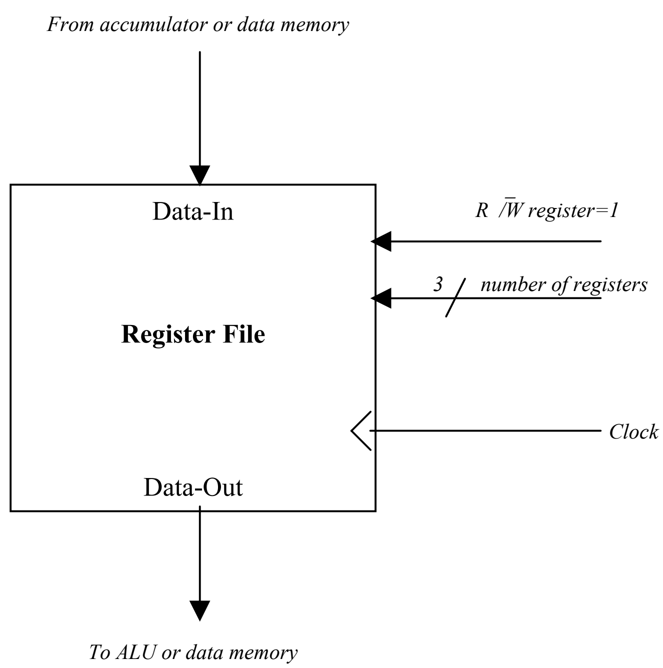

Fig.4.3. Register file

4.2.2 Second class of instructions (contains 8 instructions)

Literal op (Accu) $\longrightarrow$ *(Accu)*

The literal represents a number of 8-bits. For this instruction, we use the same hardware as previously with:

Enable accumulator = 1

MUX ALU = 1

R / $\overline{\text{W}}$ = 1

4.2.3 Third class of instructions (contains 1 instruction)

(Accu) $\longrightarrow$ *(# register)*

The content of the accumulator goes into the register, which is selected as shown in Fig.4.4.

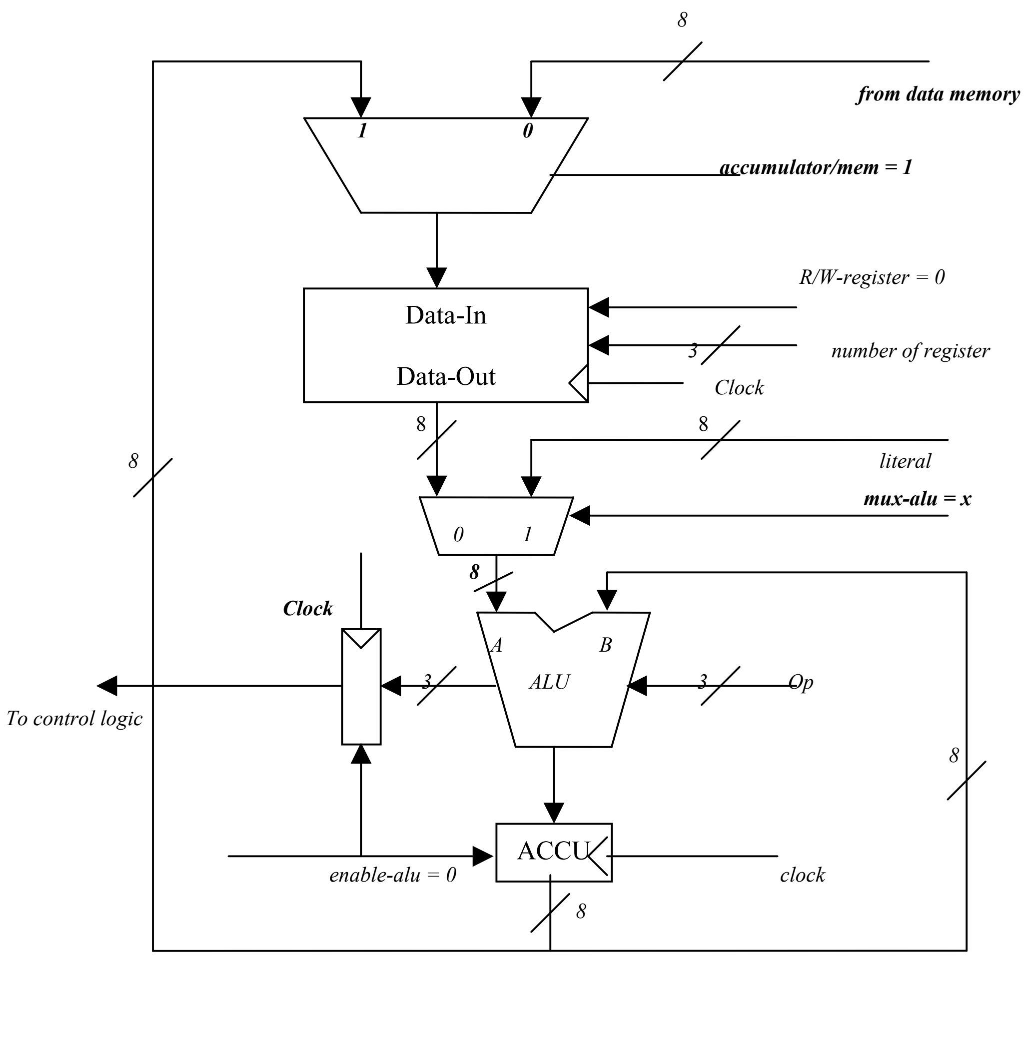

Fig.4.4. The accumulator

4.2.4. Fourth class of instructions (contains 1 instruction)

(@ ADR) $\longrightarrow$ *(# register)*

The content of address of data memory goes into the register, which is selected as shown in Fig.4.5.

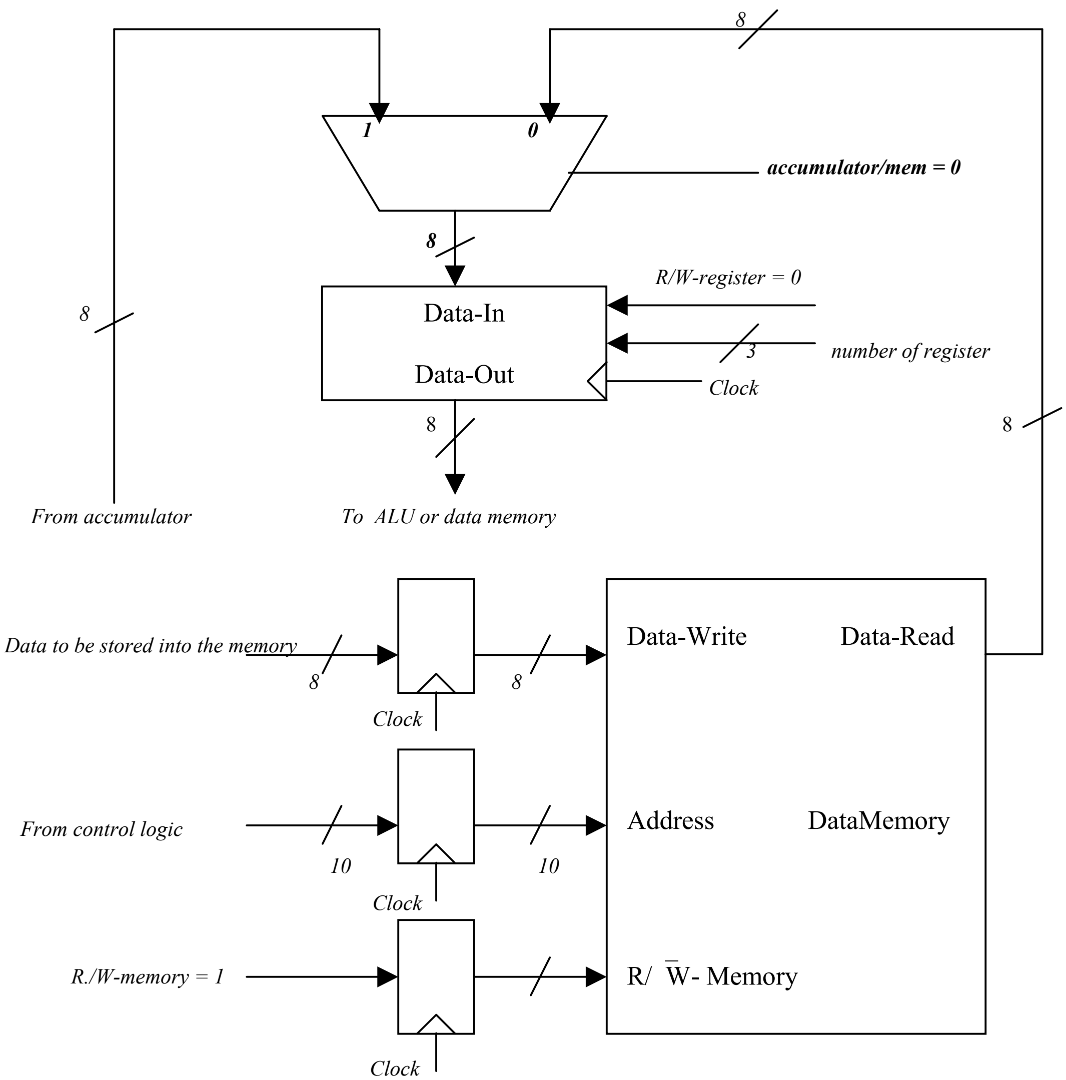

Fig.4.5.Adder address of data memory

4.2.5 Fifth class of instructions (contains 1 instruction)

(# register) $\longrightarrow$ *@ ADR*

The content of the register, which is selected, goes into the address of the data memory.

We need definitely 2-clock cycles to execute this instruction in order to be sure that the ‘write’ operation into the data memory will be made properly.

For this instruction , R/ $\overline{W}$ –memory = 0 .

4.2.6 Sixth class of instructions (contains 7 instructions)

Interrupt (INT), Jump, Call, Return, Condition for the jump, Jump value or Call address

Fig.4.6, shows clearly these 7 instruction, that how to process.

INTERRUPT (INT)

Interrupts are used to get the attention of the microprocessor. In the 8086/88 there are a total interrupts: INT 00, INT 01, INT 02, …, INT FF. The address that an interrupt jumps to is always four times the value of the interrupt number.

Every interrupt has a program associated with it called interrupt service routine (ISR).

Hardware Interrupts

The 8086/88 microprocessors have two pins set aside for inputting hardware interrupts. They are INTR (interrupt request) and NMI (nonmaskable interrupt). Although INTR can be ignored through the use of software masking, NMI cannot be masked using software. These interrupts are activated externally by putting 5 volts on the hardware pins of NMI or INTR.

Software Interrupts

These interrupts are called software interrupts since they are invoked as a result of the execution of an instruction and no external hardware is involved.

JUMP(JMP)

“Jump” means that the program counter has to be incremented, if the condition which is specified is true.

The instruction is used to transfer control unconditionally to a new address. The difference between CALL and JMP is that the CALL instruction will return and continue execution with the instruction following the CALL, Whereas JMP will not return. The target address could be within the current code segment, which is near jump, outside the current code segment, which is called far jump.

There are four possible conditions:

1. No condition
2. Accumulator (ACCU) is negative (result of subtraction)
3. ACCU = 0
4. ACCU has got a carry (result of an addition)

CALL

Means that the program counter has to be loaded with a new address, since we call a subroutine. The current address, at the same time is save in a LIFO (Last In First Out) stack.

The main function of the CALL is to control to a procedure. RET is used to return control to the instruction after the call. There are two types of CALLs: NEAR and FAR. If the target address is within the same code segment, it is a NEAR call. If the target address is outside the current code segment, it is a FAR CALL.

RETURN

Return means that the program counter has to be leaded with address, which was in this program counter just before the last “CALL”. So we must read the content of the LIFO stack.

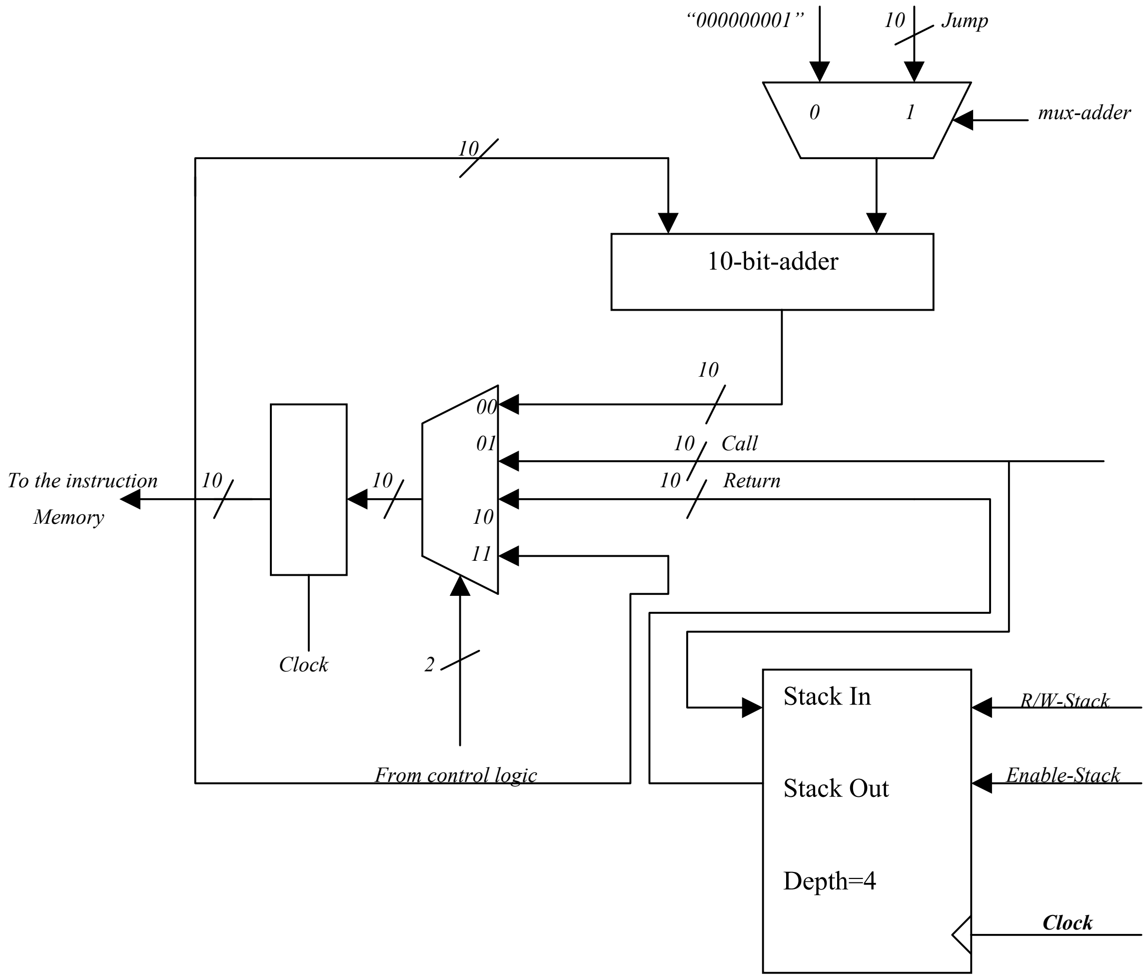

Fig. 4.6. Process of 7-instruction

In order to execute this instruction, we need 2-clock cycle, and not only 1 cycle as was the case for previous instructions.

During the first clock cycle, we get the instruction then we decode it by using the control circuit. At the end if this cycle, the address and the data are on the D inputs of registers used in association with the data memory.

During the second clock cycle, we send to the data memory the address and the input R/ $\overline{W}$ –memory equal to 1. At the end of this cycle , the data to be loaded into the selected register is generated by the port DATA-READ of the data memory and is present on the input DATAIN of the register file.

The fact we need 2 clock cycles to execute this instruction (and the following one) properly complicates a bit the structure of the control logic. Indeed the control logic circuit cannot be purely combinational. We can include the circuit in Fig.4.7, in the control logic to deal with this problem.

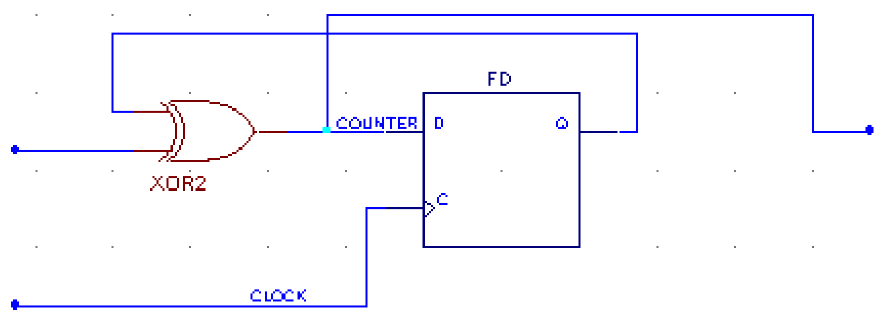

Fig.4.7. Clock cycle

Chapter 5

THE ARITHEMATIC/ LOGIC UNIT

The ALU is a combinational circuits that performs a set of basic arithmetic and logic microoperations on two n-bit operands. Its may be constructed from standard integrated circuits or programmable logic devices and are available as single-chip medium-scale integrated circuits. Integrated ALUs may be cascaded to form word lengths than are available in a single device. The ALU has a number of selection lines use to determine the operation to be performed. The selection lines are decoded with in the ALU, so the k selection lines can specify upto 2κ distinct operations. Fig.5.1, shows the block diagram of a typical n-bit ALU.

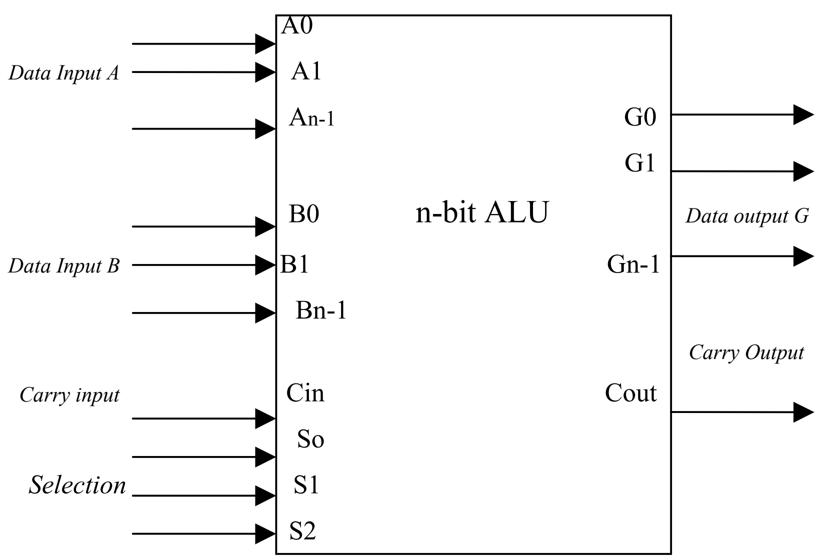

Fig.5.1. Block Diagram of *n*-bit ALU

5.1 The Arithmetic Circuit

The basic component of an arithmetic circuit is a parallel adder, which can be constructed with a number of full-adder circuits connected in cascaded, as in Fig.5.2.

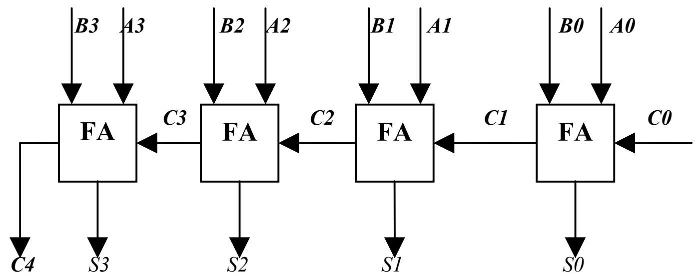

Fig.5.2. 4-Bit Ripple Carry Adder

By controlling the data inputs to the parallel adder, it is possible to obtain different types of arithmetic operations. The block diagram is in Fig.5.3,

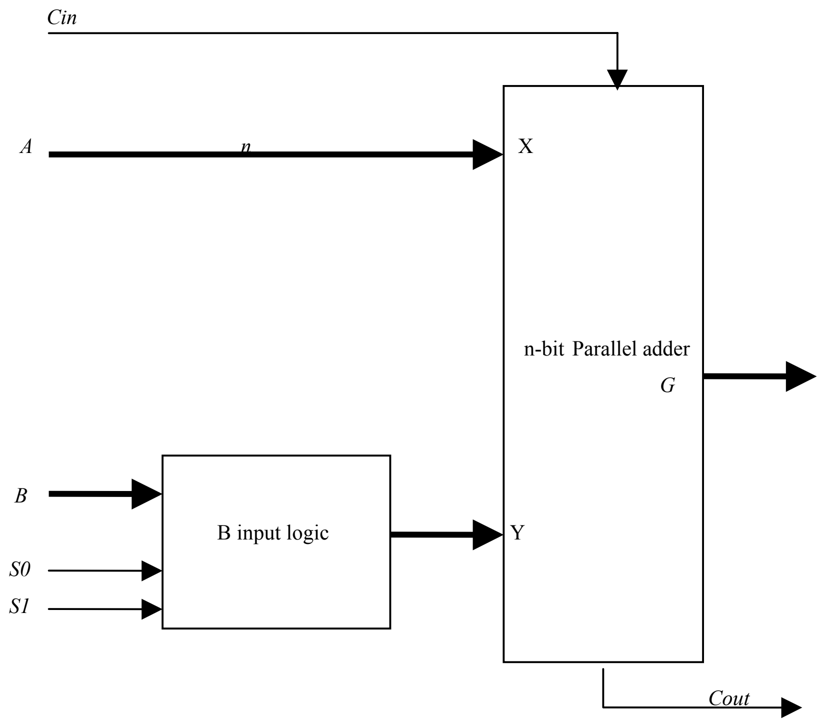

Fig.5.3.Block diagram of an arithmetic circuit

demonstrates a configuration in which one set of inputs to the parallel adder is controlled by the select lines S_1 and S_0. There are *n* bits in the arithmetic circuit, with two inputs A and B and output G. The *n* inputs from B go through the B input logic to the Y inputs of the parallel adder. The input carry C_{in} goes in the carry input of the full adder in the least-significant-bit position. The output carry C_{out} is forms the full adder in the most-significant-bit position. The output of the parallel adder is calculated from the arithmetic sum in (5.1)

$$G = X + Y + C_{in} \tag{5.1}$$

Where X is the *n*-bit binary number from the inputs and Y is the *n*-bit binary number from the B input logic. C_{in} is the input carry, which equals 0 and 1. Note that the symbol + in the equation denotes arithmetic addition.

The ALU module has eight functions:

the four standard arithmetic operations are

1. $F = A + B$
2. $F = A + 1$
3. $F = A - B$
4. $F = A - 1$

The four standard logic operations are

1. F = A and B (or) $F = A \cap B$
2. F = A or B (or) $F = A \cup B$
3. $F = A$
4. F = not A (or) $F = \overline{A}$

Since there are a total of eight operations, the selection code must contain 3 bits; that is, $S = S_2S_1S_0$. The selection codes as given in Table.5.1, for the eight ALU functions.

Table.5.1 The ALU function table

Selection Code			**ALU Function**	**Description**
S2	**S1**	**S0**		
0	0	0	$F = A + 1$	Increment
0	0	1	$F = A - 1$	Decrement
0	1	0	$F = A + B$	Add
0	1	1	$F = A - B$	Subtract
1	0	0	$F = A \cap B$	AND
1	0	1	$F = A \cup B$	OR
1	1	0	$F = A$	Transfer A
1	1	1	$F = \overline{A}$	NOT

The design in a *hierarchical, top-down* fashion. This means that top-level ALU design should initially be decomposed into a small number of modules. These modules are subsequently decomposed further until the entire design finally is represented by an interconnected hierarch of small, well-defined functional modules. Then the logic circuits for these modules are interconnected to form the complete ALU circuit.

The desired range of numbers to be manipulated by ALU for an intended application determines the number of bit, n, in the binary numbers A, B, and F. To facilitate the development of an ALU circuit design that can be used for arbitrary values of n. The top-down design by decomposing the ALU into one-bit slices, where slice I performs the desired functions on bits a_i and b_i of the operands and produces result f_i, as illustrate in Fig.5.4.

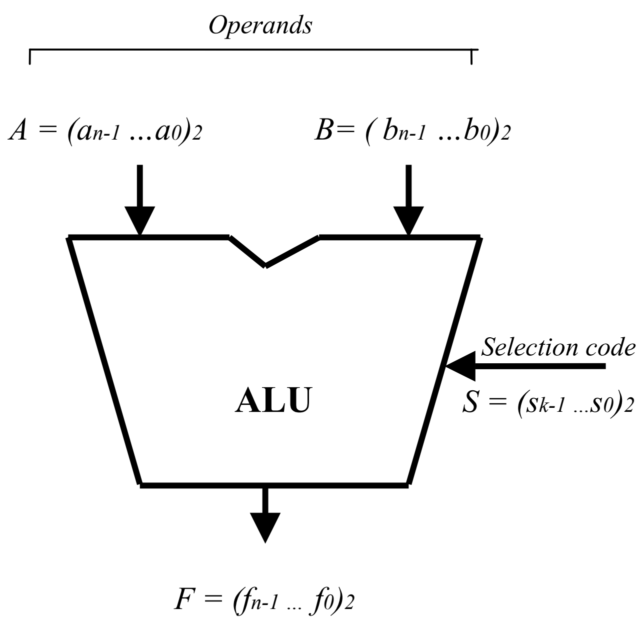

Fig.5.4. ALU logic symbol

For arithmetic functions, noted that each slice has a carry C_{i-1} and a carry output C_i. Once we have designed the circuit for the basic 1-bit slice, we then create an n-bit ALU (that is, an ALU for which A, B, and F are n-bit numbers) by simply cascaded n of the 1-bit slices as illustrated in Fig.5.5, with a special circuit to generate the initial carry input C_{-1}.

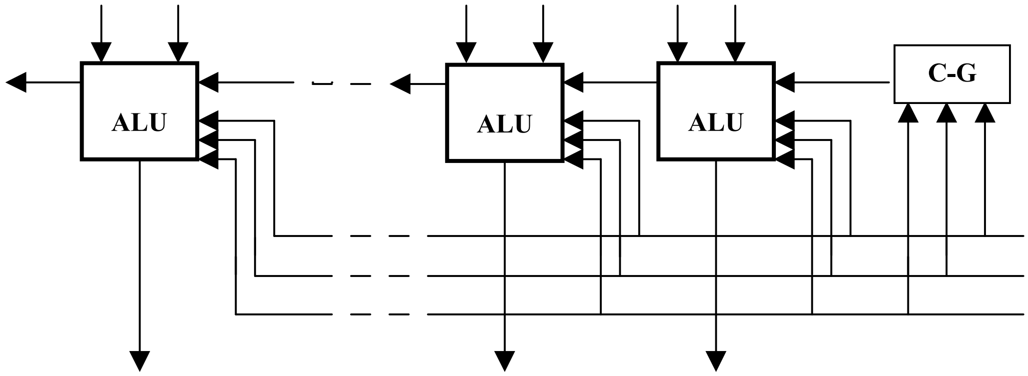

Fig.5.5. *n*-bit ALU as a cascade on *n* one-bit slices

If we consider the design of the basic one-bit ALU slice, the four arithmetic operations are somewhat related, as are the four logical operations, we can partition the ALU slice into three separate modules: an arithmetic unit (AU), a logic unit (LU), and an output multiplexer. This is illustrated in the block diagram of Fig.5.6.

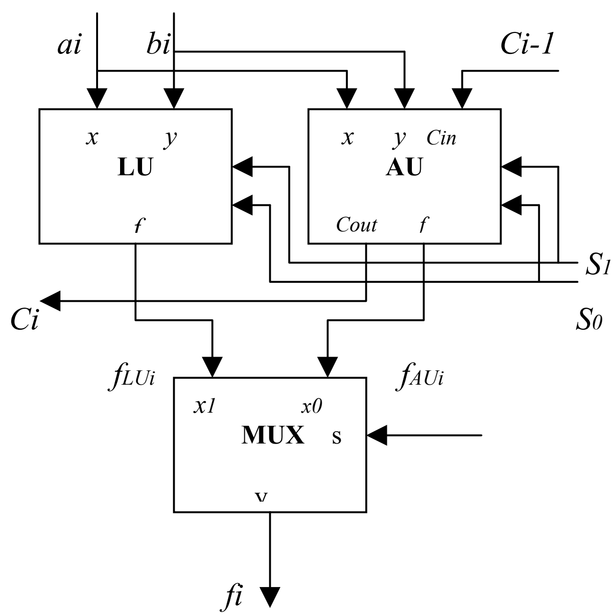

Fig.5.6. One bit ALU partitioned into separate arithmetic and logic units

The selection codes in Table.5.1, were defined so that bit S_2 determines whether the output f_i is to be an arithmetic or logic result. Therefore, the output multiplexer selects the AU output ($f_i = f_{AUi}$) for $S_2 = 0$, and the LU output ($f_i = f_{LUi}$) for $S_2 = 1$.

If we develop the design of each of the three modules of Fig.5.6, the output multiplexer is a standard 2-to-1 multiplexer module. Since this is a straightforward design, it is not decomposed further. A two-level AND gate circuit for the 2-to-1 multiplexer is given in Fig.5.7.

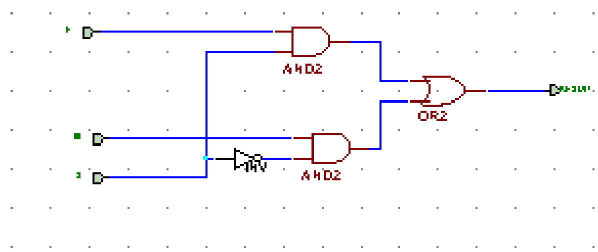

Fig.5.7.The 2-to-1 multiplexer

5.2 Logic Unit

The logic microoperations manipulate the bits of the operands by treating each bit in a register as a binary variable, giving bit wise operations. There are four commonly used logic operations-AND, OR, XOR (exclusive-OR), and NOT-from which others can be conveniently derived.

The logic functions of a digital computer system are parallel, bit-wise operations. This means that bit *i* of the result, f_{LUi} , is a logic function of input a_i and b_i , as summarized in Table.5.2.

Table.5.2. Logic unit function

Function	S1	S0	*FLUi*
AND: F = A AND B	0	0	Aibi
OR: F = A OR B	0	1	ai + bi
NOT: F = NOT A	1	0	Not ai
XOR: F = A XOR B	1	1	ai XOR bi

Once approach to implementing the LU module is to use a single primitive logic gate to realize each of four logic functions, with the output of the desired gate selected using a 4-

to-1 multiplxer, according to the selection code S_1S_0. This circuit is in Fig.5.8, where LU inputs x and y are connected to ALU inputs a_i and b_i, respectively,

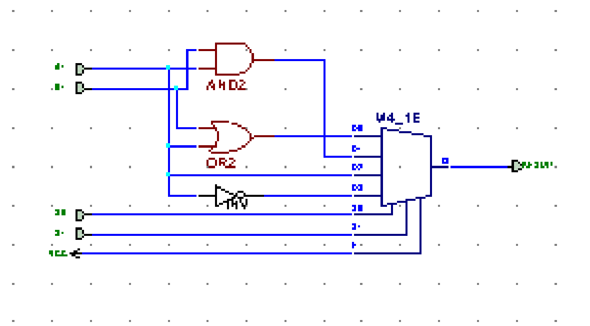

Fig.5.8. The logic unit

and LU output f is connected to f_{LUi}. The 4-to-1 multiplexer module can be realized by the circuit of Fig.5.9.

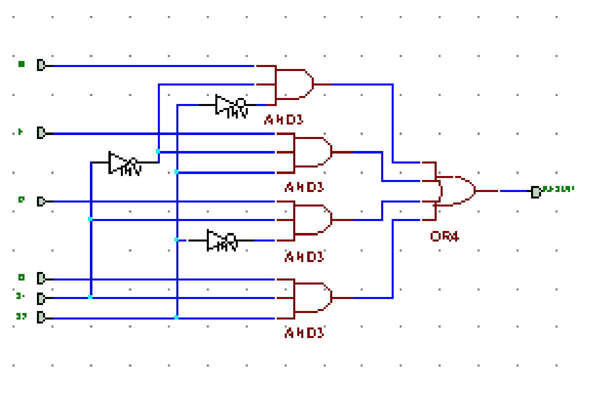

Fig.5.9. The 4-to-1 multiplexer

If minimization of the number of gates in the LU module is important, then we can use K-map.

5.3 Arithmetic Unit

5.3.1 Basic Adders and Subtracters

The basic building block for most arithmetic circuits is the **full adder.** A full adder is a logic circuit that produces the two-bit sum (S and C) of three one-bit binary numbers (X, Y, and Z). Table 5.3, shows the truth table of full adder. A logic symbol and a gate-level realization of a full adder are shown in Fig.5.10.

Table.5.3. Full adder truth table

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

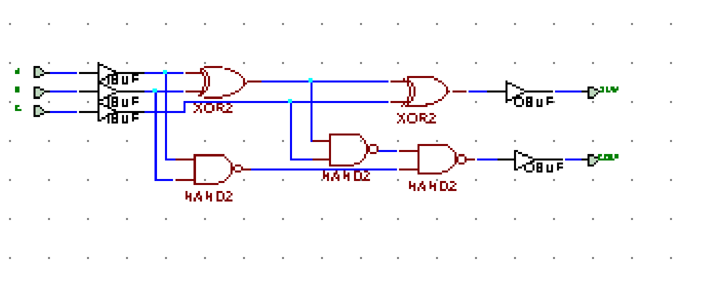

Fig.5.10. The gate diagram of full adder

The addition of two *n*-bit binary numbers (A = a_{n-1} ……a_1a_0 and B = b_{n-1} ……b_1b_0) can be a accomplished with *n* full adders cascaded as shown in Fig.5.2, such a circuit is called a **ripple-carry adder** since carries produced by lower-order stages must propagate or ripple through the higher order stages before the addition operation is complete.

Ripple-carry adders are simple in both operation and structure but are slow since in the worst case (A= 1…11 and B = 0 …01) a carry produced in the least significant full adder must propagate through all the more significant ones. The worst-case add time, T_{add}, is given in (5.1) where T_{pd} is the propagation delay introduced at each stage.

$$T_{add} = nT_{pd} \tag{5.1}$$

This assumes that all addend bits are presented to the adder simultaneously. It is important to note that in the least significant full adder, T_{pd} represents the time to compute c_1 from a_0 and b_0 and in the most significant full adder the time to compute s_{n-1} after c_{n-1} is received. In the intermediate stages, T_{pd} is the time needed to compute c_{i+1} from ci. The propagation delay is approximately equal to that of a three-level logic circuit which is consistent with the realization of a full adder given in Fig.5.10.

Subtraction can easily be performed by adding the minuend to the negative of the subtrahend. In a two's complement number system, X – Y can thus be obtained by computing $X + \overline{Y} + 1$. The ripple-carry adder described above can be easily modified to perform this computation by placing inverters on the Y inputs of each full adder and by making the carry-in (co) equal to 1.

A device that can perform either addition or subtraction can be built by replacing the inverters in the subtracter with exclusive-OR gates and using the carry-in (co) as a control signal. The device will function as ripple-carry adder when co = 0 and as a two's complement subtracter when co = 1.

5.3.2 High-Speed Adders

Several different adder designs have been developed for performing high-speed addition. These include **carry lookahead adders (CLAs)**, carry-completion adders, conditional-sum adders, and carry-select adders. Carry lookahead adders have gained wide acceptance in the design of ALUs due to the speed obtained and because they can be conveniently implemented in integrated circuit form.

Addition is a combinational process so it is theoretically possible to construct a 2n-bit "full-adder" that can be realized by a three level combinational logic circuit and that can perform addition of two *n*-nit numbers in the time equal to the delay of the circuit. However, such circuits are too costly in terms of gate fan-in to be implemented for reasonable values of *n*. Carry lookahead is a practical and effective compromise between fully parallel adders and ripple-carry adders. The block diagram of a four-bit CLA is shown in Fig.5.11.

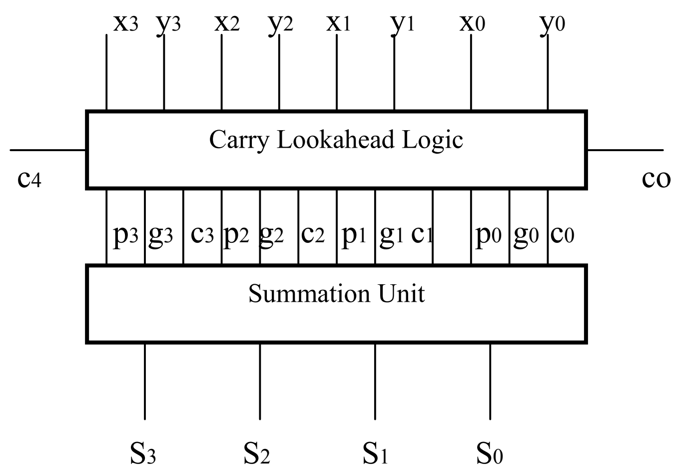

Fig.5.11. Carry lookahead adder

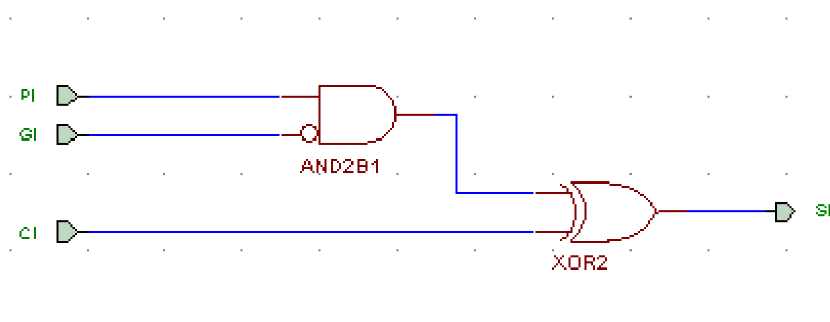

Fig.5.12. Summation stage i

CLAs are based on the observation that carryout (ci) of the *i*th stage of a stage full adder is produced by either the propagation of the carry-in (c_{i-1}) through the *i*th stage or the generation of a carry-in the *i*th stage. This can be seen in the following logic equations (5.2), (5.3), and (5.4) for ci:

$$c_i = x_{i-1}y_{i-1} + x_{i-1}c_{i-1} + 1c_{i-1} \quad (5.2)$$

$$c_i = x_{i-1}y_{i-1} + (x_{i-1} + y_{i-1})c_{i-1} \quad (5.3)$$

$$c_i = g_{i-1} + p_{i-1}c_{i-1} \quad (5.\ 4)$$

Where $g_i = x_iy_i$ and $p_i = x_i + y_i$ are the generate and propagate terms, respectively, for stage I for I = 0 to n – 1.

The carry equations for an n-bit adder can be derived by repeatedly applying the above equation. The following set of equations (5.5), (5.6), (5.7), and (5.8) in results for the n = 4 case:

$$c_1 = g_o + p_oc_o \quad (5.5)$$

$$c_2 = g_1 + p_1g_o + p_1p_oc_o \quad (5.6)$$

$$c_3 = g_2 + p_2g_1 + p_2p_1g_o + p_2p_1p_oc_o \quad (5.7)$$

$$c_4 = g_3 + p_3g_2 + p_3p_2g1 + p_3p_2p_1g_0 + p_3p_2p_1c_o \quad (5.8)$$

The carry equations can be realized by three-level combinational logic circuits to form the carry lookahead logic block shown in Fig.5.11. The sum (si) bits for the ithe stage of an adder can be written in terms of gi, pi, and ci and generated by the logic circuit given in Fig.5.12. This completes the description of the four-bit CLA.

The add-time, Tadd, for a CLA. Assume that both addends are applied to the CLA simultaneously and that $c_o=0$. Also Tpd represents the propagation delay of a three-level logic circuit. There are two components that contribute to the add time. First, the three-level carry lookahead logic must produce the carries. This takes Tpd. Then, the summation unit must produce the final values of the sum bits. This step takes a time equal to the propagation delay of the exclusive-OR gate in the summation unit, which is Tpd since an exclusive-OR gate can be realized as a three-level combinational logic circuit. Hence, the add time for a CLA is

$$T_{add} = 2T_{pd} \quad (5.9)$$

The above result indicates that the add time of a CLA is not only much faster than a ripple-carry adder but is also independent of the length (n) of the addends. Hence, one might conclude that CLAs are the final answer to the high-speed adder problem. However, a closer look at the set of carry equations given above quickly reveals that the equations become progressively more complex in the number of product terms and literal. Therefore, fan-in constraints will eventually limit the particularly of realizing the equations in three-level-logic. The actual limit in technology is dependent standard single-chip medium-scale ALUs typically handle four-bit operands, although longer lenghts are certainly feasible with today's technology.

CLAs may be cascaded to produce an adder for longer operands. Fig.5.2, shows a cascade of four 4-bit CLAs to produce a 16-bit adder. Carries are produced using carry ahead logic within each CLAs stage but must propagate between stages in a manner reminiscent of a ripple-carry adder. Hence the add time of cascaded CLAs is dependent on the number of stages in the cascade. The four-stage adder shown in Fig.5.2, has a worst-case add time of $5T_{pd.}$ In general, the add time of an m-stage cascade is $(m + 1)T_{pd.}$

5.3.3 Multifunction Arithmetic Logic Units

Devices that can provide a variety of addition, and logical operations can be easily designed around the adders/Subtracters. The logic diagram of the first two stages of an *n*-bit multifunction ALU is given in Fig.5.19. Operand inputs for the device are $X = x_{n-1} \ldots x_1 x_o$ and $Y = y_{n-1} \ldots y_1 y_0$ and the output is
$S = s_{n-1} \ldots s_1 s_0.$ The function performed on the operand(s) is determined by the values of the control inputs S_2, S_1, S_0, as shown in Table.5.1. The given realization is based on a ripple-carry adder for simplicity of presentation.

5.3.4. The ALU Slice

In the arithmetic unit of our ALU, the addition and subtraction will be performed by a single full-adder circuit, using two'complement arithmetic. For this design, we will use the full-adder (FA) circuit as shown in Fig.5.10. When the ALU slices are cascaded, the FA stages will be connected in the ripple-carry adder configuration as also shown in Fig.5.2 . If we recall that an *n*-bit full adder realizes the expression.

$$F = X + Y + C_{-1} \qquad (5.10)$$

Where F, X, and Y are *n*-bit binary numbers and C_{-1} is the carry input. The four desired arithmetic operations can be implement by manipulating the values of Y and C_{-1} in the Equation (5.10). Therefore, we will design a circuit that will produce the y_i input for each

FA module, according to the selection code bit S_1 and S_0, and another circuit to derive C_{-1} .The overall 1-bit AU slice configuration is shown in Fig.5.13. Note that FA input Xi is simply connected to ALU input a_i.

If we consider each of the four arithmetic operations separately.

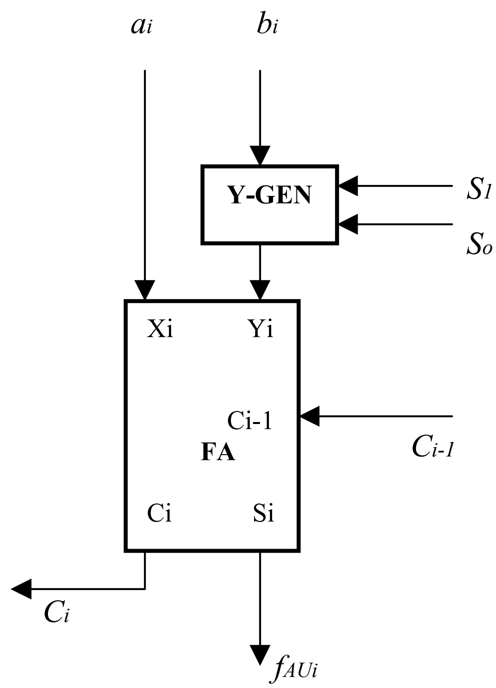

Fig.5.13. Block diagram of the ALU slice

Add: <u>F = A + B</u>

Here , for the FA module, we simply set X = A, Y = B, and $C_{-1} = 0$. Therefore, the Y-GEN module should connect input b_i to the Yi input of the FA.

Subtract: <u>F = A – B</u>

The two's complement

F = A – B

$F = A + [B]_2$

$F = A + (\overline{b}_{n-1} \ldots \ \overline{b}_1 \ \overline{b}_0) + 1$

Therefore, the subtract function is implemented by setting $y_i = \overline{b_i}$ and $C_{-1} = 1$. Consequently, the Y-GEN module should connect the complement of b_i to the y_i input of the FA.

Increment: <u>F = A + 1</u>

In this case we will set Y = 0 and $C_{-1} = 1$ in equation (5.10). Therefore, the Y-GEN module should supply 0 to the Yi input of the FA.

Decrement: <u>F = A – 1</u>

Again we will make the two's complement as follows:

$$
\begin{aligned}
F &= A - 1 \\
&= A + (-1) \\
&= A + [00 \ldots 01]_2 \\
&= A + (11 \ldots 11) \\
&= A + (11 \ldots 11) + 0
\end{aligned}
$$

Therefore, we realize the decrement function by setting the FA inputs $y_i = 1$ and $C_{-1} = 0$.

Table.5.4, summarizes the preceding discussion by listing the required values of FA inputs y_i and C_{-1} for each of the four arithmetic operations. From this table, we can derive circuits for the Y-GEN module of Fig.5.13. and C-GEN module of Fig.5.5.

Table.5.4. Values of yi and C_{-1} for the arithmetic function

Function	S_1	S_0	y_i	C_{-1}
Increment	0	0	b_i	0
Decrement	0	1	$\overline{b_i}$	1
Add	1	0	0	1
Subtract	1	1	1	0

So after all specifications , we can design the 1-bit AU slice as shown in Fig.5.14.

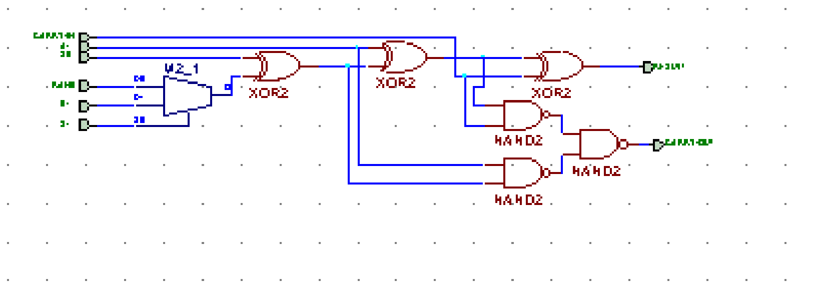

Fig.5.14. Arithmetic unit (AU) slice

The 1-bit ALU slice is now formed by interconnecting the individual modules developed previously (LU, FA, YGEN and MUX). The complete circuit for the 1-bit ALU slice is shown in Fig.5.15.

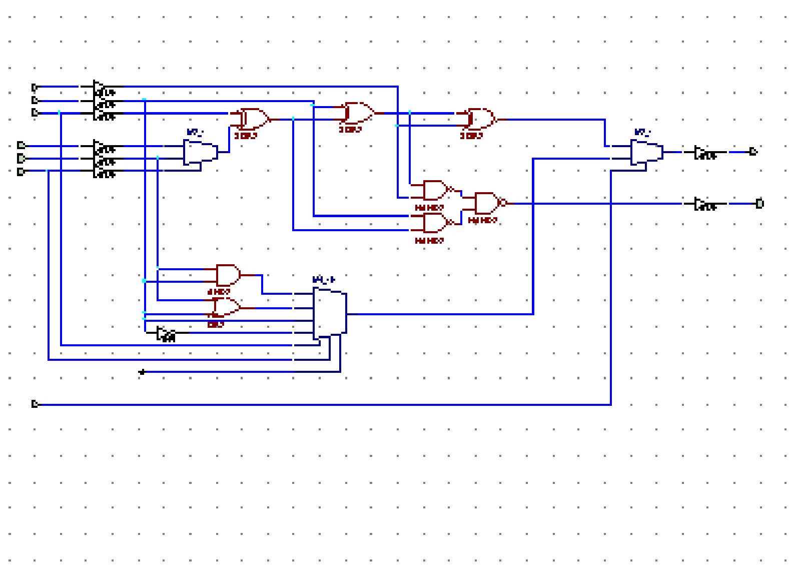

Fig.5.15. Complete 1-bit ALU unit

If we show inside the gate diagram of 2-to-1 and 4-to-1 multiplexer in an 1-bit bit ALU slice then the full gate diagram is shown in Fig5.16.

Fig.5.16.The complete gate diagram of 1-bit ALU

The final step of the design is to create our *n*-bit ALU by cascading *n* of the 1-bit ALU slices and connecting the C-GEN module, was shown in Fig.5.5. can implement. For this we have to put 1-bit ALU slice in one "hierarchy" as shown in Fig.5.17. called 1-bit ALU cell.

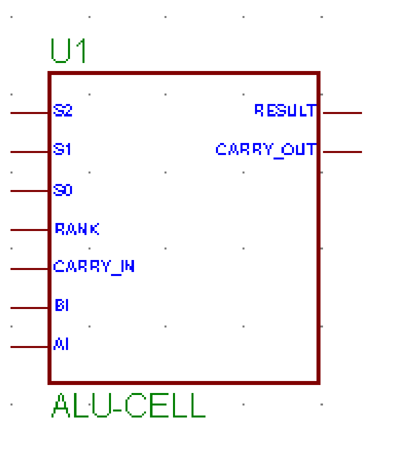

Fig.5.17. 1-ALU cell

Similarly we can design 64-bit ALU in similar manners. For this we have to use another hierarchy which has 8-bit ALU as shown in Fig.5.18.

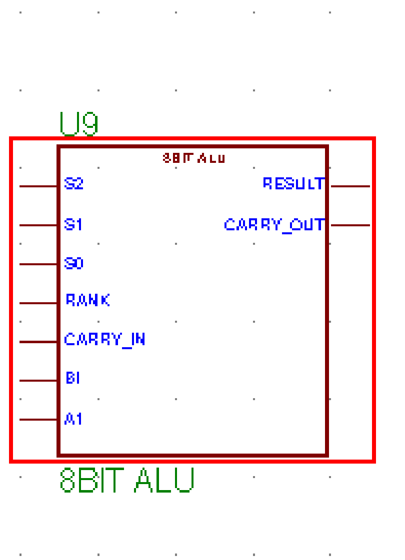

Fig. 5.18. 8-bit ALU cell

From Fig.5.18, we can connect 8-bit ALU cell 8 times properly in cascaded, then we can design 64-bit ALU cell as shown in Fig.5.19

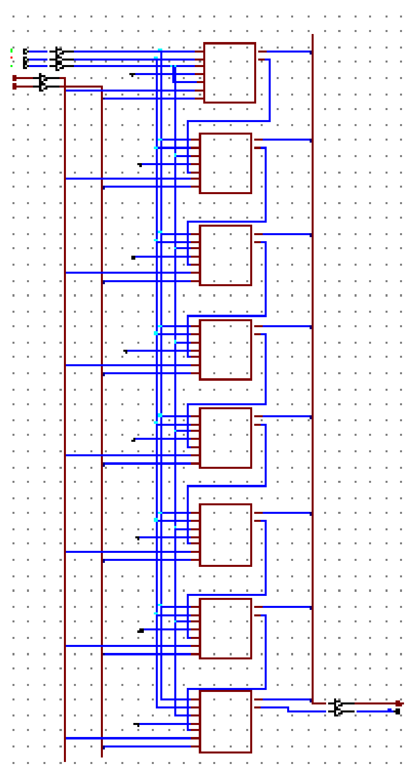

Fig.5.19. 64-bit ALU

5.4 Standard Integrated Circuit ALUs

The device described above generic in nature but are similar in function and realization to many commercially available integrated circuit products. Representative products are summarized in Table.5.5.

Table.5.5. Typical integrated circuit ALU devices

Part Number	Function	Features
74LS181	4-bit multifunction (16) ALU	BCL outputs
74LS182	Carry lookahead generator	Use with 74LS181 for BCL
74LS183	Full adder	Two per package
74LS183	4-bit binary adder	Internal CL
74LS181	4-bit multifunction (8) ALU	BCL outputs
74LS182	4-bit multifunction (8) ALU	Ripple-carry output

Chapter 6

COMPUTER AIDED DESIGN OF LOGIC CIRCUITS

Many of today's digital logic circuits contain the equivalent of thousands to hundreds of thousands of logic gates. Most of these circuits are fabricated on single integrated circuit chips (ICs). Designing and fabrication a VLSI (Very large-scale Integrated) circuit chips is a complex and expensive process. Therefore, it is necessary to verify the correctness of the logic circuit design before beginning the design of the actual chip to ensure a high probability of correct operation the first time the chip is fabricated. The same is true when designing digital systems out of multiple ICs and circuit boards. Circuits and systems of this complexity are virtually impossible to develop and verify without the assistance of computer-aided design (CAD) tools.

The design cycle for a digital logic comprises a number of steps between concept and physical implementation, including design synthesis, simulation, realization, and testing. This processes is depicted in Fig.6.1. From a statement of the problem, the designer begins by developing an abstract solution that is systematically transformed into a digital logic circuit. This transformation is aided by modeling and evaluating the circuit at each level of design abstraction. A design is evaluated by using its model to simulate its operation, allowing the circuit's response to various input stimuli to be verified. The model is revised and resimulated as needed until the correct responses are obtained. In addition to verifying the correct operation, the effects of different design options on circuit performance can be evaluated to assist in making cost-effective design decisions. Once the modeled behavior of the design is acceptable, the physical design is developed and implemented. Finally, the finished circuit is tested with the test results compared to the modeled behavior to detect faulty devices.

Modeling a digital logic circuit or system serves several purposes. First, the process of developing a model helps the designer formulize a solution. Second, a circuit model can be processed by a digital computer to check for design errors, verify correctness, and predict timing characteristics. In addition, a number of CAD tools are available that automatically perform all or some of the synthesis steps, including design optimization and transformation of the design from an abstract form to a physical realization.

A model can represent a digital system at different levels of design abstraction, ranging from behavior to structural.

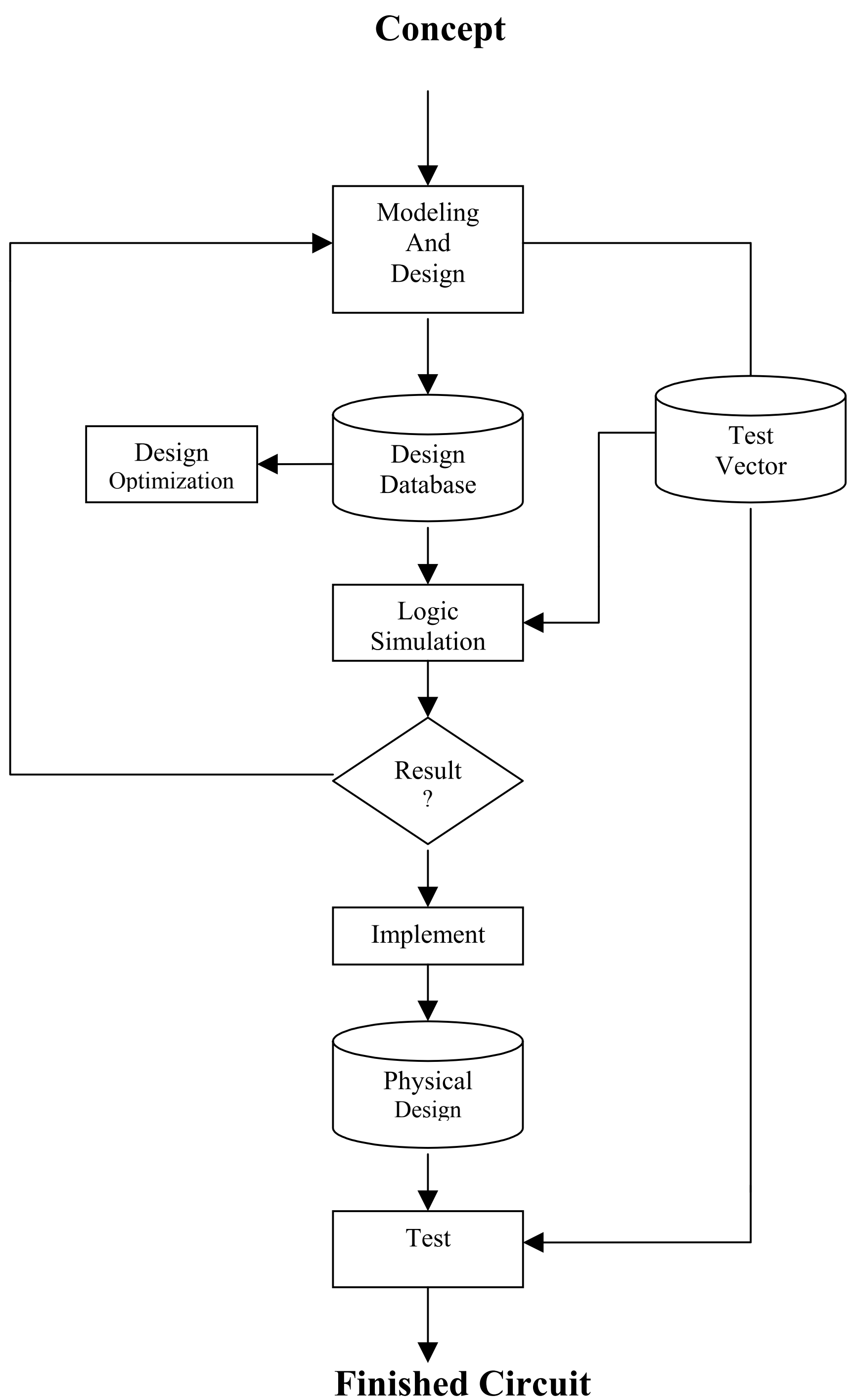

Fig.6.1. The Layout of design procedure

Chapter 7

CONCLUSIONS

The design and simulation is not an easy job. For this we should have enough background to understand the actual concept and then design and simulate. For me it was difficult to start my project, because before starting I had no idea about how we can design and simulate on the computer.

For this I learn Xilinx software by myself and then tried to understand the actual theory and concept about an ALU. According to the given specifications, I used my previous knowledge of "Logic Circuit Design and VLSI". First I designed the arithmetic part which was the most difficult part, while the second part was logic part which was much easier. When I designed both parts of ALUs, I combined them with logic gates on the Xilinx software. Fortunately the results were 100% correct that what I expected. This thing gives me more encouragement and instead of 8-bit ALU I tried to design 64-bit.

Due to lack of time, I tried to modify it more but I could not success. I mean we can modify our ALU like instead of by using unsigned number we can use signed numbers, as similarly we can handle with the overflow problems.

Chapter 8

SIMULATION RESULTS

REFRENCES

[1] M. A. Mazidi, J. G. Mazidi. *The 80x86 IBM PC and Compatible Computers (Volumes I & II), Second Edition,* Prentice Hall, USA 1998.

[2] V. P. Nelson, H. T. Nagle, B. D. Carroll, J. D. Irwin, *Digital Logic Circuit Analysis & Design,* Prentice Hall, USA 1995.

[3] M. M. Mano, C. R. Kime, *Logic and Computer Design Fundamentals,* Prentice Hall, USA 1997.

[4] B. B. Brey, *The Intel Microprocessors 8086/8088, 80186, 80286,80386, and 80486 Architecture, Programming, and Interfacing, Third Edition,* Prentice Hall, USA 1991.

[5] R. C. Draf (chief editor), *The Electrical Engineering handbook,* CRC Press, USA 1993.

[6] R. J. Tocci, *Digital Systems Principles and Applications, Fifth Edition,* Prentice Hall, USA 1991.

[7] S. Le Goff, *lecture notes on microprocessor RISC EMU 544.*