COMPUTER UNIVERSITY (MANDALAY)

FINAL YEAR PROJECT REPORT

ON

DESIGN OF 2-BIT ARITHMETIC LOGIC UNIT

Bachelor of Computer Technology

(B.C.Tech.)

Presented by Group (1)

2014-2015

ACKNOWLEDGEMENTS

I would like to express my appreciation and thanks to the following persons who guided and aided directly toward the success of this project.

First of all, I respectfully thank to **Dr. Win Aye**, Rector of the Computer University (Mandalay) for her kind permission to develop this project.

I am deeply thankful to Dr. Zarni Sann, Associate Professor, and my supervisor **Daw Thazin Nwe**, Hardware Department, Computer University (Mandalay) for her valuable guidance, supervision and encouragement that have strengthened my focus, presentation and correctness of the project.

I am also deeply thankful to **Daw Kyi Kyi Mon**, Lecture, English Department, Computer University (Mandalay), for her editing this project from the English language.

Last but not the least, I am very much grateful to all my teachers, friends and colleagues of the Computer University (Mandalay) for their cooperation and help to fulfill this project.

Group Member List

Sr.No	Name	Roll No.
1	Ma Khwar Nyo Pwint	4CT-7
2	Ma Kyawt Kay Thwal	4CT-15

Supervisor

Name: **Daw Thazin Nwe** 28.9.15

Rank: **Tutor**

Department: **Hardware**

Computer University (Mandalay)

Project Schedule

Project Proposal : : March, 2015

First Seminar : :4.6.2015

Second Seminar : :9.7.2015

Third Seminar : :30.7.2015

Book Submission : :September, 2015

Time Schedule	March 2015	April 2015	July 2015	July 2015	September 2015
Project Proposal					
First Seminar					
Second Seminar					
Third Seminar					
Book Submission					

Abstract

This project is intended to implement the 2-bit ALU. The 3-bit decoder is used to select between various operations: AND, OR, NAND, NOR, XOR, Addition and Subtraction. Half-adder and full-adder are operated for addition and subtraction. AND, OR, NAND, NOR and XOR are used for other operation. When addition and subtraction are not done, the carry flag is ignored. The carry signal, which is used for addition and subtraction, is generated and passed out of the ALU for every operation. An overflow is only maths related; this will be implemented in the ADD/SUB unit. When adding, if two input bits are "1", the result bit will be different and then have an overflow.

List of Figures

List of Tables

CONTENTS

CHAPTER 3 DESIGN AND IMPLEMENTATION

CHAPTER 4 CONCLUSION

CHAPTER (1)
INTRODUCTION

1.1 Introduction

Arithmetic Logic Unit (ALU) can perform multi-operation, combinational-logic digital functions. ALU is a critical component of a microprocessor. 2-bit ALU can perform 8 operations such as AND, OR, NOT, NOR, NAND, XOR, addition and subtraction. In this project,2-bit ALU will perform AND, OR, NAND, NOR, XOR, Addition and Subtraction.

1.2 Objectives of the Project

- To know how to create the design of 2-bit ALU with logic gates,
- To know how to implement the design of full-adder and half-adder with logic gates,
- To study the existing method for arithmetic and logic operations,to implement the using of 3bit decoder,
- To be used for the person who learn with the aid of this practical device.

1.3 Project Requirements

1.3.1 AND Gate(7408)

A Boolean operator which gives the value one if and only if all the operands are one, and otherwise has a value of zero.

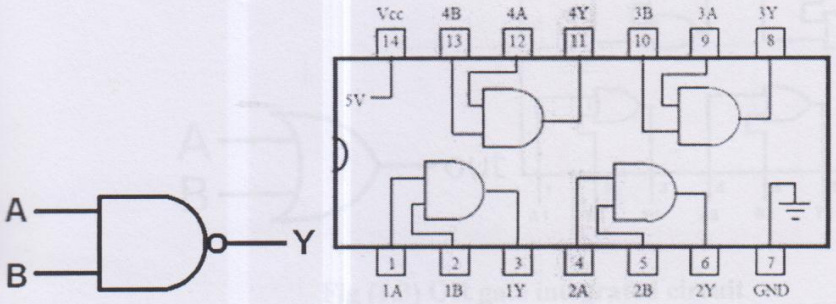


Fig (1.1) *AND* gate integrated circuit

1.3.2 NOT Gate(7404)

In digital logic, an inverter or NOT gate is a logic gate which implements logical negation. The truth table is shown on the right.

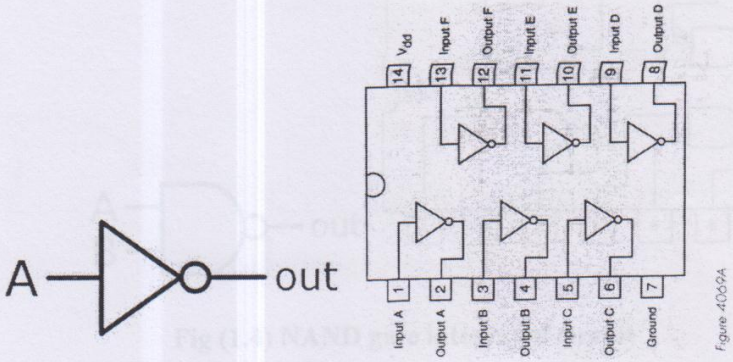


Fig (1.2) *NOT* gate integrated circuit

1.3.3 OR Gate(7432)

The OR gate is an electronic circuit that gives a high output(1)if *one or more* of its inputs are high.

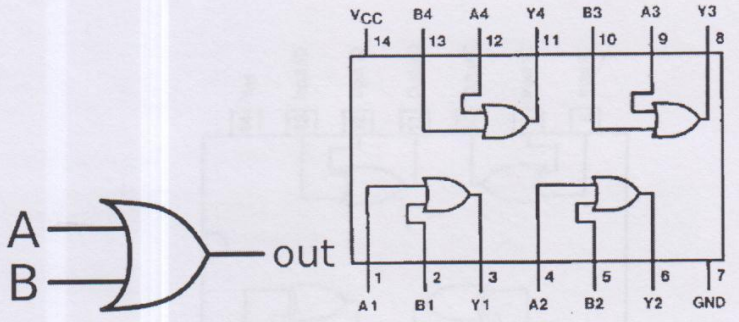


Fig (1.3) OR gate integrated circuit

1.3.4 NAND Gate(7400)

The outputs of all NAND gates are high if *any* of the inputs are low.

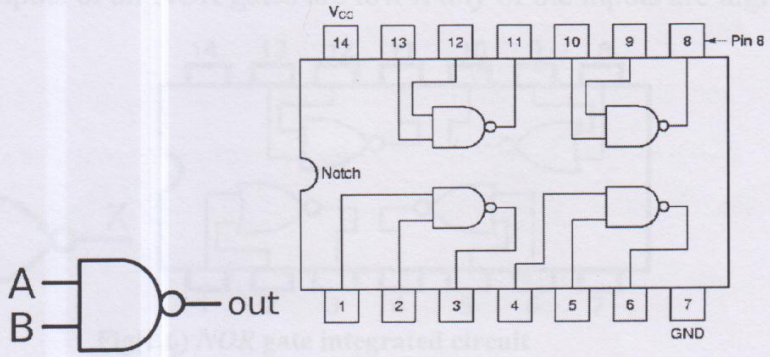


Fig (1.4) NAND gate integrated circuit

1.3.5 XOR Gate(7486)

The Exclusive-OR gate is a circuit which will give a high output if *either* of two inputs is high.

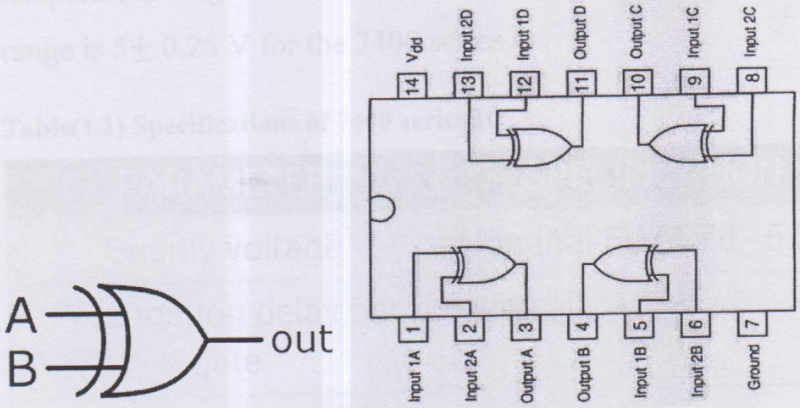
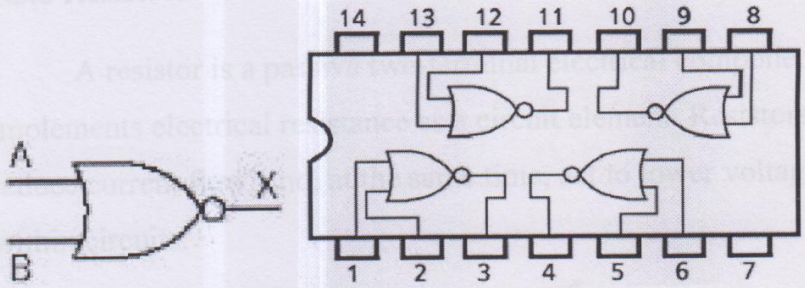


Fig (1.5) X*OR* gate integrated circuit

1.3.6NOR Gate (7402)

The outputs of all NOR gates are low if *any* of the inputs are high.



Fig(1.6) *NOR* gate integrated circuit

1.3.7 Specifications of 7400 series IC

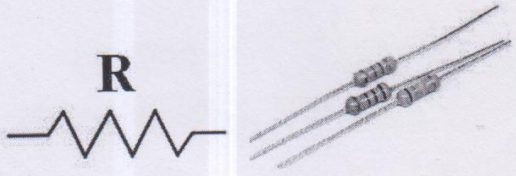
TTL 7400 series is the most popular and commonly used series of digital ICs. 74oo devices are used for commercial applications. The temperature range is 0° C to 70° C for the 7400 series.The supply voltage range is 5± 0.25 V for the 7400 series.

Table(1.1) Specifications of 7400 series IC

PARAMETER	SPECIFICATION
Supply voltage	Nominal 5V (4.75 - 5.25)
Propagation delay per gate	Typically 10 ns
Max toggle speed	25 MHz
Power consumption per gate	10 mW

1.3.8 Resistors

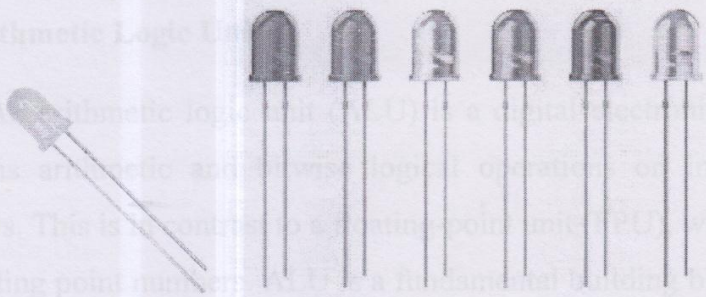
A resistor is a passive two-terminal electrical component that implements electrical resistance as a circuit element. Resistors act to reduce current flow, and, at the same time, act to lower voltage levels within circuits.



Fig(1.7) Resistor symbol and component

1.3.9 LEDs

Light emitting diode(LED) is a two-lead semiconductor light source . LED is a junction diode which emits light when activated.



Fig(1.8) Light Emitting Diode(LED)

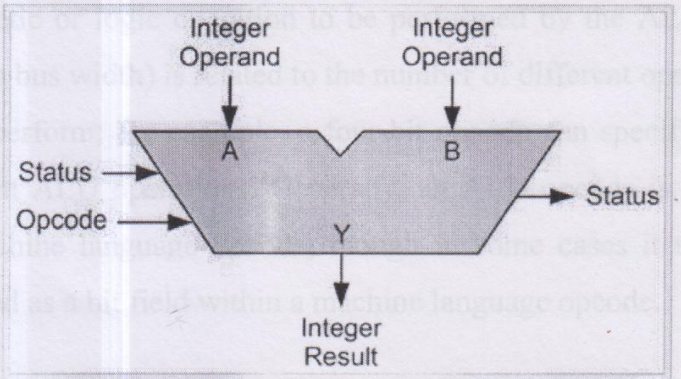
CHAPTER (2)

THEORY BACKGROUND

2.1 Arithmetic Logic Unit

An arithmetic logic unit (ALU) is a digital electronic circuit that performs arithmetic and bitwise logical operations on integer binary numbers. This is in contrast to a floating-point unit (FPU), which operates on floating point numbers. ALU is a fundamental building block of many types of computing circuits, including the central processing unit (CPU) of computers, FPUs, and graphics processing units (GPUs). A single CPU, FPU or GPU may contain multiple ALUs.

The inputs to an ALU are the data to be operated on, called operands, and a code indicating the operation to be performed; the ALU's output is the result of the performed operation. In many designs, the ALU also exchanges additional information with a status register, which relates to the result of the current or previous operations.



A symbolic representation of an ALU and its input and output signals, indicated by arrows pointing into or out of the ALU, respectively. Each arrow represents one or more signals.

Fig(2.1) A symbolic representation of an ALU

2.1.1 Signals

An ALU has a variety of input and output nets, which are the shared electrical connections used to convey digital signals between the ALU and external circuitry. When an ALU is operating, external circuits apply signals to the ALU inputs and, in response, the ALU produces and conveys signals to external circuitry via its outputs.

2.1.2 Data Bus

A basic ALU has three parallel data buses consisting of two input operands (A and B) and a result output (Y). Each data bus is a group of signals that conveys one binary integer number. Typically, the A, B and Y bus widths (the number of signals comprising each bus) are identical and match the native word size of the encapsulating CPU (or other processor).

2.1.3 Opcode

The opcode input is a parallel bus that conveys to the ALU an operation selection code, which is an enumerated value that specifies the desired arithmetic or logic operation to be performed by the ALU. The opcode size (its bus width) is related to the number of different operations the ALU can perform; for example, a four-bit opcode can specify up to sixteen different ALU operations. Generally, an ALU opcode is not the same as a machine language opcode, though in some cases it may be directly encoded as a bit field within a machine language opcode.

2.1.4 Status

The status outputs are various individual signals that convey supplemental information about the result of an ALU operation. These

outputs are usually stored in registers so they can be used in future ALU operations or for controlling conditional branching. The collection of bit registers that store the status outputs are often treated as a single, multi-bit register, which is referred to as the "status register" or "condition code register". General-purpose ALUs commonly have status signals such as:

- Carry-out, which conveys the carry resulting from an addition operation, the borrow resulting from a subtraction operation, or the overflow bit resulting from a binary shift operation.
- Zero, which indicates all bits of the Y busis logic zero.
- Negative, which indicates the result of an arithmetic operation is negative.
- Overflow, which indicates the result of an arithmetic operation has exceeded the numeric range of the Y bus.
- Parity, which indicates whether an even or odd number of bits on the Y bus is logic one.

The status input allows additional information to be made available to the ALU when performing an operation. Typically, this is a "carry-in" bit that is the stored carry-out from a previous ALU operation.

2.1.5 Circuit Operation

An ALU is a combinational logic circuit, meaning that its outputs will change asynchronously in response to input changes. In normal operation, stable signals are applied to all of the ALU inputs and, when enough time (known as the "propagation delay") has passed for the signals to propagate through the ALU circuitry, the result of the ALU operation appears at the ALU outputs. The external circuitry connected to the ALU is responsible for ensuring the stability of ALU input signals throughout the operation, and for allowing sufficient time for the signals to propagate through the ALU before sampling the ALU result.

In general, external circuitry controls an ALU by applying signals to its inputs. Typically, the external circuitry employs sequential logic to control the ALU operation, which is paced by a clock signal of a sufficiently low frequency to ensure enough time for the ALU outputs to settle under worst-case conditions.

For example, a CPU begins an ALU addition operation by routing operands from their sources (which are usually registers) to the ALU's operand inputs, while the control unit simultaneously applies a value to the ALU's opcode input, configuring it to perform addition. At the same time, the CPU also routes the ALU result output to a destination register that will receive the sum. The ALU's input signals, which are held stable until the next clock, are allowed to propagate through the ALU and to the destination register while the CPU waits for the next clock. When the next clock arrives, the destination register stores the ALU result and, since the ALU operation has completed, the ALU inputs may be set up for the next ALU operation.

2.1.6 Functions

A number of basic arithmetic and bitwise logic functions are commonly supported by ALUs. Basic, general purpose ALUs typically include these operations in their repertoires.

2.1.7 Arithmetic Operations

- *Add*: A and B are summed and the sum appears at Y and carry-out.
- *Add with carry*: A, B and carry-in are summed and the sum appears at Y and carry-out.
- *Subtract*: B is subtracted from A (or vice-versa) and the difference appears at Y and carry-out. For this function, carry-out

is effectively a "borrow" indicator. This operation may also be used to compare the magnitudes of A and B; in such cases the Y output may be ignored by the processor, which is only interested in the status bits (particularly zero and negative) that result from the operation.

- *Subtract with borrow*: B is subtracted from A (or vice-versa) with borrow (carry-in) and the difference appears at Y and carry-out (borrow out).
- *Two's complement (negate):* A (or B) is subtracted from zero and the difference appears at Y.
- *Increment*: A (or B) is increased by one and the resulting value appears at Y.
- *Decrement*: A (or B) is decreased by one and the resulting value appears at Y.
- *Pass through*: all bits of A (or B) appear unmodified at Y. This operation is typically used to determine the parity of the operand or whether it is zero or negative.

2.1.8 Bitwise Logical Operations

- AND: the bitwise AND of A and B appears at Y.
- OR: the bitwise OR of A and B appears at Y.
- Exclusive-OR: the bitwise XOR of A and B appears at Y.
- One's complement: all bits of A (or B) are inverted and then appear at Y.

2.1.9 Bit Shift Operations

ALU shift operations cause operand A (or B) to shift left or right (depending on the opcode) and the shifted operand appears at Y. Simple ALUs typically can shift the operand by only one bit position, whereas

more complex ALUs employ barrel shifters that allow them to shift the operand by an arbitrary number of bits in one operation. In all single-bit shift operations, the bit shifted out of the operand appears on carry-out; the value of the bit shifted into the operand depends on the type of shift.

- Arithmetic shift: the operand is treated as a two's complement integer, meaning that the most significant bit is a "sign" bit and is preserved.
- Logical shift: a logic zero is shifted into the operand. This is used to shift unsigned integers.
- Rotate: the operand is treated as a circular buffer of bits so its least and most significant bits are effectively adjacent.
- Rotate through carry: the carry bit and operand are collectively treated as a circular buffer of bits.

2.1.10 History

Mathematician John von Neumann proposed the ALU concept in 1945 in a report on the foundations for a new computer called the EDVAC.

The cost, size, and power consumption of electronic circuitry was relatively high throughout the infancy of the information age. Consequently, all serial computers and many early computers, such as the PDP-8, had a simple ALU that operated on one data bit at a time, although they often presented a wider word size to programmers. One of the earliest computers to have multiple discrete single-bit ALU circuits was the 1948 Whirlwind I, which employed sixteen of such "math units" to enable it to operate on 16-bit words.

In 1967, Fairchild introduced the first ALU implemented as an integrated circuit, the Fairchild 3800, consisting of an eight-bit ALU with accumulator. Other integrated-circuit ALUs soon emerged, including

four-bit ALUs such as the Am2901 and 74181. These devices were typically "bit slice" capable, meaning they had "carry look ahead" signals that facilitated the use of multiple interconnected ALU chips to create an ALU with a wider word size. These devices quickly became popular and were widely used in bit-slice minicomputers.

Microprocessors began to appear in the early 1970s. Even though transistors had become smaller, there was often insufficient die space for a full-word-width ALU and, as a result, some early microprocessors employed a narrow ALU that required multiple cycles per machine language instruction. Examples of this include the original Motorola 68000, which performed a 32-bit "add" instruction in two cycles with a 16-bit ALU, and the popular Zilog Z80, which performed eight-bit additions with a four-bit ALU.[3] Over time, transistor geometries shrank further, following the Moore's law, and it became feasible to build wider ALUs on microprocessors. Modern integrated circuit (IC) transistors are orders of magnitude smaller than those of the early microprocessors, making it possible to fit highly complex ALUs on ICs. Today, many modern ALUs have wide word widths, and architectural enhancements such as barrel shifters and binary multipliers that allow them to perform, in a single clock cycle, operations that would have required multiple operations on earlier ALUs.

2.2 Two's Complement

Two's complement is a mathematical operation on binary numbers, as well as a binary signed number representation based on this operation. Its wide use in computing makes it the most important example of a radix complement.

The two's complement of an N-bit number is defined as the complement with respect to 2N; in other words, it is the result of

subtracting the number from 2N, which in binary is one followed by N zeroes. This is also equivalent to taking the ones' complement and then adding one, since the sum of a number and its ones' complement is all 1 bits. The two's complement of a number behaves like the negative of the original number in most arithmetic, and positive and negative numbers can coexist in a natural way.

In two's-complement representation, positive numbers are simply represented as themselves, and negative numbers are represented by the two's complement of their absolute value; two tables on the right provide examples for N = 3 and N = 8. In general, negation (reversing the sign) is performed by taking the two's complement. This system is the most common method of representing signed integers on computers. An N-bit two's-complement numeral system can represent every integer in the range-(2^{N-1}) to + $(2^{N-1} - 1)$ while ones' complement can only represent integers in the range-$(2^{N-1} - 1)$ to + $(2^{N-1} - 1)$.

Table(2.1) Some special numbers to note

Some special numbers to note

Decimal	Two's complement
127	0111 1111
64	0100 0000
1	0000 0001
0	0000 0000
−1	1111 1111
−64	1100 0000
−127	1000 0001
−128	1000 0000

2.2.1 History

The method of complements had been used to perform subtraction in decimal adding machines and mechanical calculators. John von Neumann suggested use of two's complement binary representation in his 1945 First Draft of a Report on the EDVAC proposal for an electronic stored-program digital computer.[3] The 1949 EDSAC, which was inspired by the First Draft, used two's complement representation of binary numbers.

Many early computers, including the CDC 6600, use ones' complement notation. The IBM 700/7000 series scientific machines use sign/magnitude notation, except for the index registers which are two's complement. Early commercial two's complement computers include the Digital Equipment Corporation PDP-5 and the 1963 PDP-6. The System/360, introduced in 1964 by IBM, then the dominant player in the computer industry, made two's complement the most widely used binary representation in the computer industry. The first minicomputer, the PDP-8 introduced in 1965, uses two's complement arithmetic as do the 1969 Data General Nova, the 1970 PDP-11, and almost all subsequent minicomputers and microcomputers.

2.2.2 Converting to 2's complement representation

In two's complement notation, a non-negative number is represented by its ordinary binary representation; in this case, the most significant bit is 0. Though, the range of numbers represented is not the same as with unsigned binary numbers. For example, an 8-bit unsigned number can represent the values 0 to 255 (11111111). However a two's complement 8-bit number can only represent positive integers from 0 to

127 (01111111), because the rest of the bit combinations with the most significant bit as '1' represent the negative integers 1 to 128.

The two's complement operation is the additive inverse operation, so negative numbers are represented by the two's complement of the absolute value.

2.3 Full Adder

A logic circuit that accept three digit binary numbers and produces two outputs; a sum output(S) and a carry out(C out) .

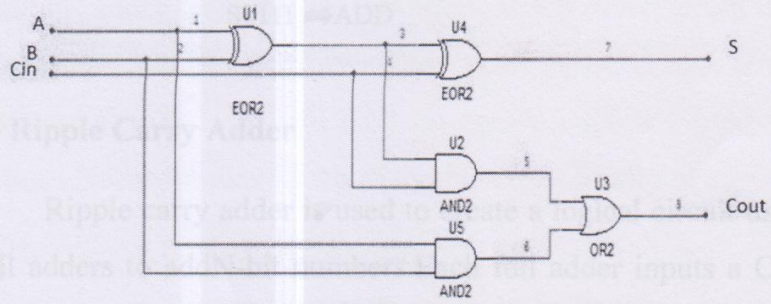


Fig (2.2) Full adder circuit diagram

Table (2.2) Truth table of Full-Adder

Input			Output	
A	B	Cin	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

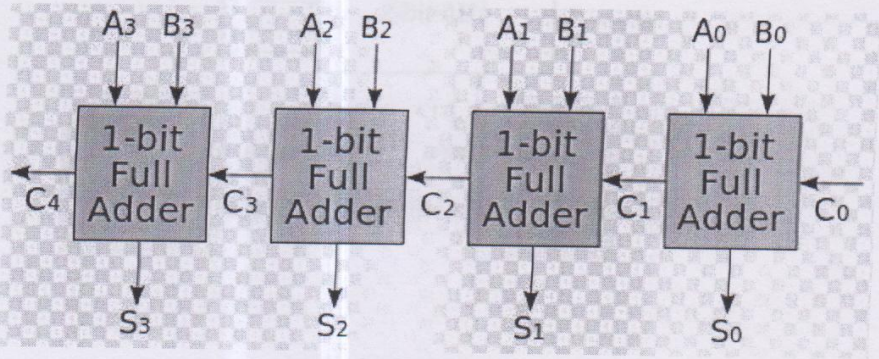
2.4 Decoder

A decoder is a circuit that changes a code into a set of signals. In this project, 3 bit decoder is used for select 8 functions.

S=000 ➡ NOT
S=001 ➡ AND
S=010 ➡ OR
S=011 ➡ NOR
S=100 ➡ NAND
S=101 ➡ XOR
S=110 ➡ SUB
S=111 ➡ ADD

2.5 Ripple Carry Adder

Ripple carry adder is used to create a logical circuit using multiple full adders to addN-bit numbers.Each full adder inputs a Cin, which is the Cout of the previous adder. This kind of adder is called a ripple-carry adder, since each carry bit "ripples" to the next full adder.



Fig(2.3) Ripple Carry Adder

CHAPTER (3)

DESIGN AND IMPLEMENTATION

3.1 Design of 2-bit ALU

This project will specify eight operations such as NOT, AND,OR, NOR,NAND, XOR , Addition and Subtraction.This ALU is a 2-bit ALU with three inputs(operands) named Cin (carry in),A and B:A[0] and B[0]are the least-significant bits and A[1] and B[1]are most-significant bits.Full adder is used for addition and subtraction.In addition and subtraction: C0, C1 and overflow (Cin XOR Cout) are outputs. In subtraction, initial Cin will be 1, to compute the 2's complement.The 3 bit decoder determines which of the functions is output.

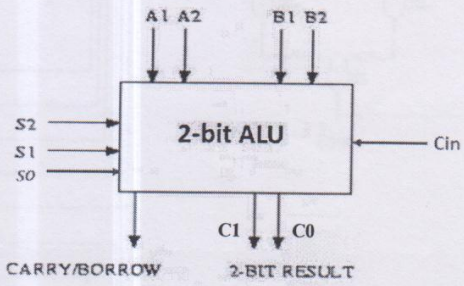
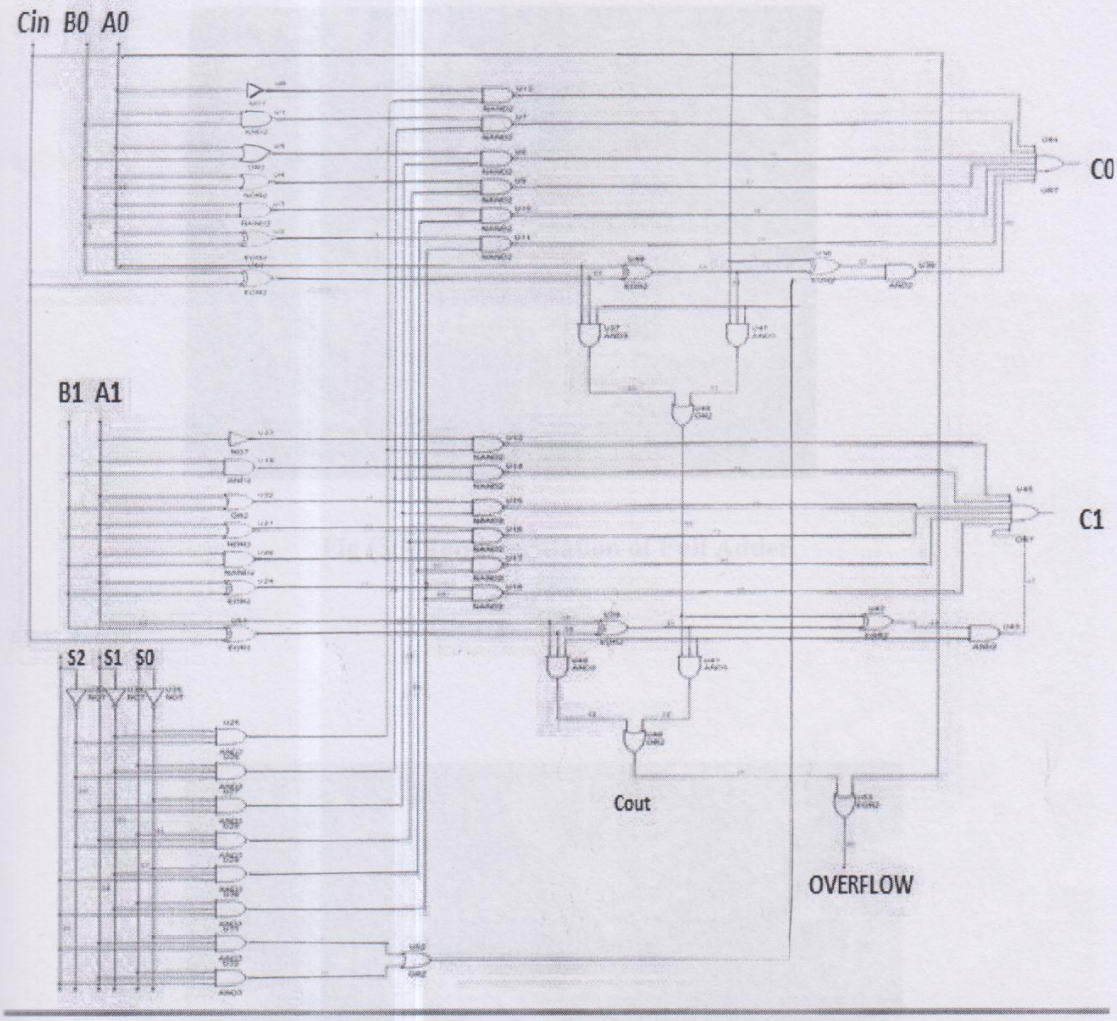


Fig (3.1) Block diagram of 2-Bit ALU

3.2 Circuit Diagram



Fig(3.2) Circuit diagram of 2-bit ALU

3.3 Implementation of 2-bit ALU

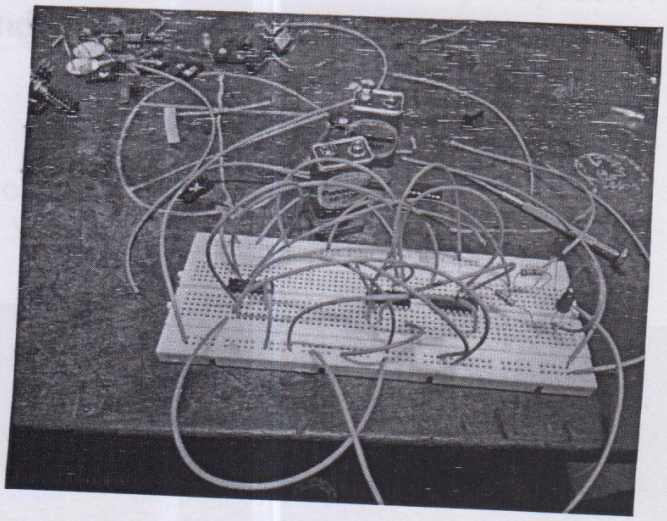


Fig (3.3) Implementation of Full Adder

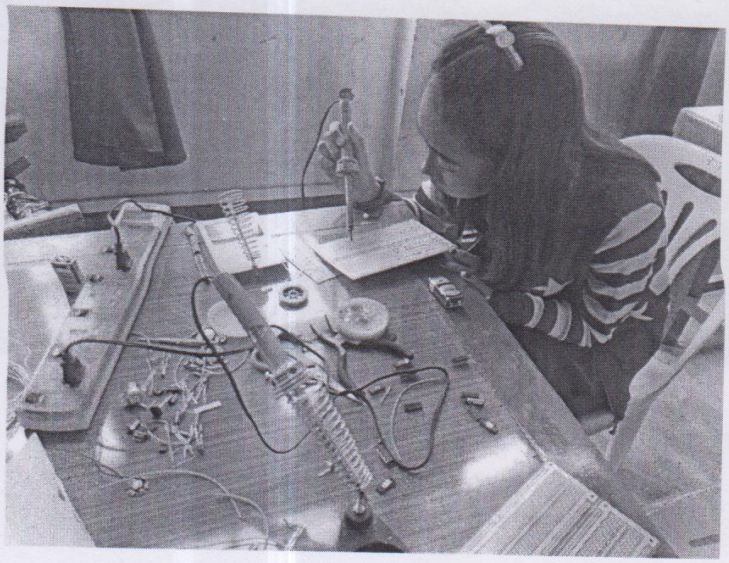


Fig (3.4) Implementation of 2-bit ALU

3.4 Hardware Implementation

There are 3 inputs for decoder, 5 input for A0, A1, B0, B1, Cin and 3 outputs for C0, C1 and overflow bit.

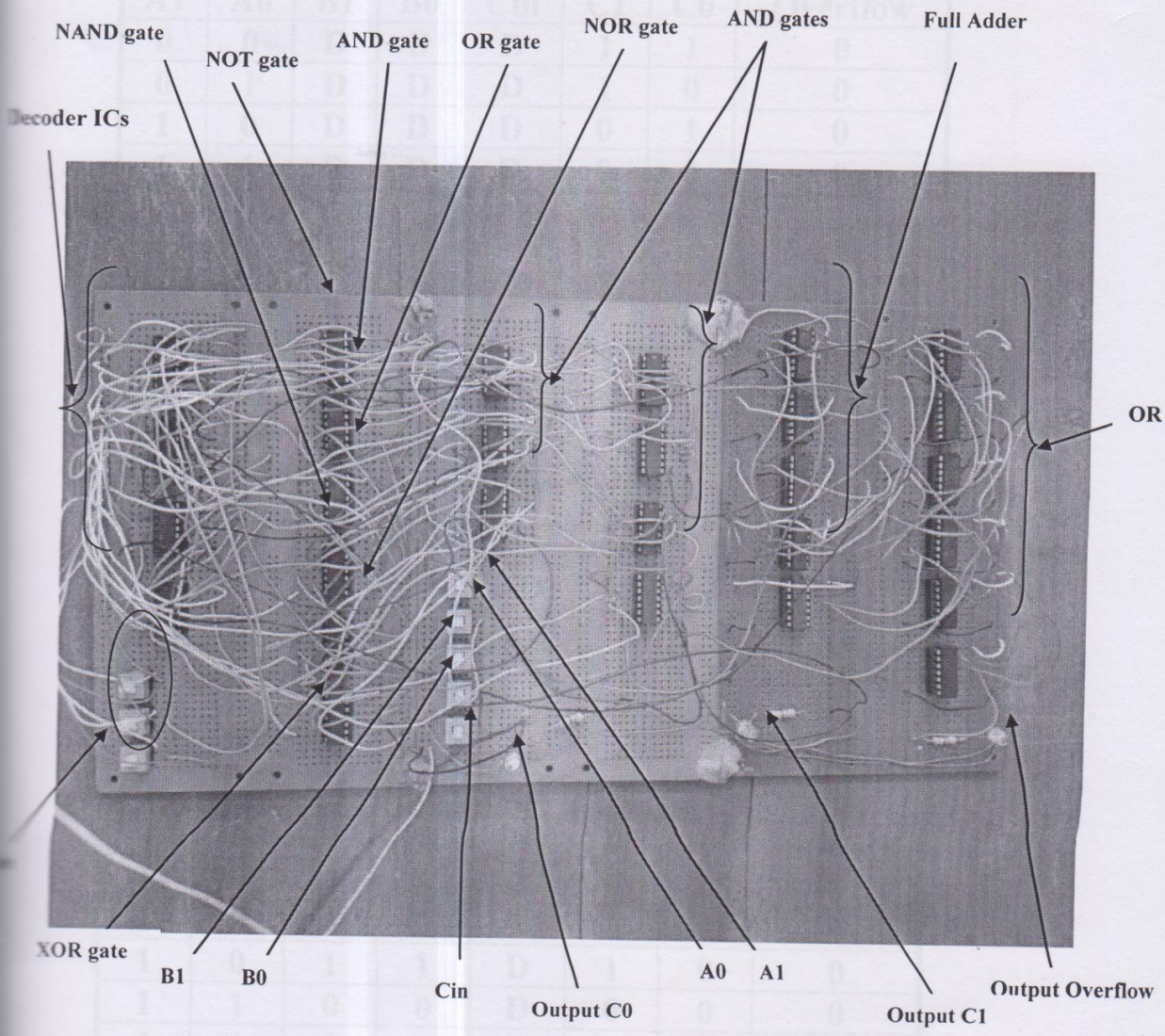


Figure (3.5) Hardware Implementation

Table (3.1) Truth table of NOT operation

NOT- A							
INPUT					OUTPUT		
A1	A0	B1	B0	Cin	C1	C0	Overflow
0	0	D	D	D	1	1	0
0	1	D	D	D	1	0	0
1	0	D	D	D	0	1	0
1	1	D	D	D	0	0	0

Table (3.2) Truth table of AND operation

AND							
INPUT					OUTPUT		
A1	A0	B1	B0	Cin	C1	C0	Overflow
0	0	0	0	D	0	0	0
0	0	0	1	D	0	0	0
0	0	1	0	D	0	0	0
0	0	1	1	D	0	0	0
0	1	0	0	D	0	0	0
0	1	0	1	D	0	1	0
0	1	1	0	D	0	0	0
0	1	1	1	D	0	1	0
1	0	0	0	D	0	0	0
1	0	0	1	D	0	0	0
1	0	1	0	D	1	0	0
1	0	1	1	D	1	0	0
1	1	0	0	D	0	0	0
1	1	0	1	D	0	1	0
1	1	1	0	D	1	0	0
1	1	1	1	D	1	1	0

Table (3.3) Truth table of OR operation

OR							
INPUT					OUTPUT		
A1	A0	B1	B0	Cin	C1	C0	Overflow
0	0	0	0	D	0	0	0
0	0	0	1	D	0	1	0
0	0	1	0	D	1	0	0
0	0	1	1	D	1	1	0
0	1	0	0	D	0	1	0
0	1	0	1	D	0	1	0
0	1	1	0	D	1	1	0
0	1	1	1	D	1	1	0
1	0	0	0	D	1	0	0
1	0	0	1	D	1	1	0
1	0	1	0	D	1	0	0
1	0	1	1	D	1	1	0
1	1	0	0	D	1	1	0
1	1	0	1	D	1	1	0
1	1	1	0	D	1	1	0
1	1	1	1	D	1	1	0

Table (3.4) Truth table of NAND operation

NAND							
INPUT					OUTPUT		
A1	A0	B1	B0	Cin	C1	C0	Overflow
0	0	0	0	D	1	1	0
0	0	0	1	D	1	1	0
0	0	1	0	D	1	1	0
0	0	1	1	D	1	1	0
0	1	0	0	D	1	1	0
0	1	0	1	D	1	0	0
0	1	1	0	D	1	1	0
0	1	1	1	D	1	0	0
1	0	0	0	D	1	1	0
1	0	0	1	D	1	1	0
1	0	1	0	D	0	1	0
1	0	1	1	D	0	1	0
1	1	0	0	D	1	1	0
1	1	0	1	D	1	0	0
1	1	1	0	D	0	1	0
1	1	1	1	D	0	0	0

Table (3.5) Truth table of XOR operation

XOR							
INPUT					OUTPUT		
A1	A0	B1	B0	Cin	C1	C0	Overflow
0	0	0	0	D	0	0	0
0	0	0	1	D	0	1	0
0	0	1	0	D	1	0	0
0	0	1	1	D	1	1	0
0	1	0	0	D	0	1	0
0	1	0	1	D	0	0	0
0	1	1	0	D	1	1	0
0	1	1	1	D	1	0	0
1	0	0	0	D	1	0	0
1	0	0	1	D	1	1	0
1	0	1	0	D	0	0	0
1	0	1	1	D	0	1	0
1	1	0	0	D	1	1	0
1	1	0	1	D	1	0	0
1	1	1	0	D	0	1	0
1	1	1	1	D	0	0	0

Table (3.6) Truth table of NOR operation

NOR							
INPUT					OUTPUT		
A1	A0	B1	B0	Cin	C1	C0	Overflow
0	0	0	0	D	1	1	0
0	0	0	1	D	1	0	0
0	0	1	0	D	0	1	0
0	0	1	1	D	0	0	0
0	1	0	0	D	1	0	0
0	1	0	1	D	1	0	0
0	1	1	0	D	0	0	0
0	1	1	1	D	0	0	0
1	0	0	0	D	0	1	0
1	0	0	1	D	0	0	0
1	0	1	0	D	0	1	0
1	0	1	1	D	0	0	0
1	1	0	0	D	0	0	0
1	1	0	1	D	0	0	0
1	1	1	0	D	0	0	0
1	1	1	1	D	0	0	0

Table (3.7) Truth table of ADD operation

ADD							
INPUT					OUTPUT		
A1	A0	B1	B0	Cin	C1	C0	Overflow
0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0
0	0	1	0	0	1	0	0
0	0	1	1	0	1	1	0
0	1	0	0	0	0	1	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	1	0
0	1	1	1	0	0	0	1
1	0	0	0	0	1	0	0
1	0	0	1	0	1	1	0
1	0	1	0	0	0	0	1
1	0	1	1	0	0	1	1
1	1	0	0	0	1	1	0
1	1	0	1	0	0	0	1
1	1	1	0	0	0	1	1
1	1	1	1	0	1	0	1

Table (3.8) Truth table of SUB operation

SUB {(A+(NOT B)+1)=A-B}							
INPUT					OUTPUT		
A1	A0	B1	B0	Cin	C1	C0	Overflow
0	0	0	0	1	0	0	0
0	0	0	1	1	0	1	1
0	0	1	0	1	1	0	1
0	0	1	1	1	1	1	1
0	1	0	0	1	0	1	0
0	1	0	1	1	0	0	0
0	1	1	0	1	0	1	1
0	1	1	1	1	1	0	1
1	0	0	0	1	1	0	0
1	0	0	1	1	0	1	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	1	1
1	1	0	0	1	1	1	0
1	1	0	1	1	1	0	0
1	1	1	0	1	0	1	0
1	1	1	1	1	0	0	0

CHAPTER (4)
CONCLUSION

4.1 Conclusion

In this project, full-adder and half-adder are used for addition. This project is intended to study the concept of ALU. This project is designed to aid student practically.

4.2 Advantages of the Project

It is less expensive due to using minimum gates.In this project, 2bit arithmetic operations can be calculated.It ispossible to minimize the logic gates.

4.3 Limitations and Further Extension

Operating temperature range of AND, OR, NOR, NAND, XOR, NOT gates are -40 C to +125 C.Voltage ranges of AND, OR, NOR, AND, XOR, NOT gates are 3V to 6V. By using 2-bit ALU basically, we can produce 8-bit, 32-bit or 64-bit.

References

References Order

[1] Computer Architecture and Organization/John P. Hayes

[2] Electronic Devices/Floyd

[3] Digital fundamental

[4] Website (www.wikipedia.org/arithmetic-logic-unit)

[5] Website (www.wikipedia.org/carry-lookahead-adder)

[6] Website (www.mhhe.com)