ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

Academic Year: 2020-2021 **Class/Branch**: SE-A & B/COMP

Subject: CSL302: DLCOA **Semester**: III

Course Outcomes:	
CSL302.1	To analyze different number systems, their conversions and basics of various digital components.
CSL302.2	To analyze and minimize Boolean expressions.
CSL302.3	To design and analyze combinational circuits.
CSL302.4	To demonstrate arithmetic algorithms using ALU.
CSL302.5	To demonstrate the working of ALU and various control units.
CSL302.6	To describe multicore architecture, pipelining and parallel processing.

Sr. No.	**List of Experiments**	**Lab Outcome**
1	To study and verify the truth table of various logic gates and realize Boolean expressions using gates.	**CSL302.1**
2	To realize basic gates using universal gates.	**CSL302.2**
3	To implement code conversion(binary to gray and gray to binary)	**CSL302.3**
4	To realize arithmetic circuits	**CSL302.3**

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

	Half adder ii) Full adder	
5	To simulate the working of ALU	**CSL302.4**
6	To simulate the working of Booth's algorithm	**CSL302.4**
7	To simulate the working of Restoring division algorithm	**CSL302.5**
8	To simulate the working of Non restoring division algorithm	**CSL302.5**
9	Study on multicore processors.	**CSL302.6**
10	To simulate the memory design	**CSL302.5**

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

Experiment – 1: To Study and Verify the Truth Table of Various Logic Gates and Realize Boolean Expressions Using Gates.

Date: 30th July 2020

1. Aim:

1. To study logic gates & verify the truth tables.
2. Verify the logical expression using truth table.

2. Requirements

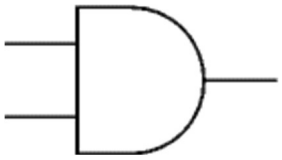
Java, Simulator to use Virtual lab

3. Pre-Experiment Exercise
Brief Theory

A. Basic Gates

1. AND gate

The output Q is true if input A AND input B are both true: **Q = A AND B** An AND gate can have two or more inputs, its output is true if all inputs are true.



Input A	Input B	Output Q
0	0	0
0	1	0
1	0	0
1	1	1

Traditional symbol Truth Table

2. OR gate

The output Q is true if input A OR input B is true (or both of them are true): **Q = A OR B**. An OR gate can have two or more inputs, its output is true if at least one input is true.

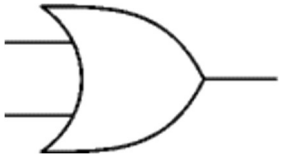
ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

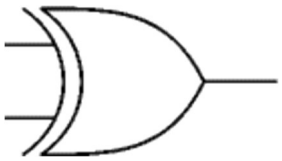


Input A	Input B	Output Q
0	0	0
0	1	1
1	0	1
1	1	1

Traditional symbol Truth Table

3. EX-OR (EXclusive-OR) gate

The output Q is true if either input A is true OR input B is true, **but not when both of them are true**: **Q = (A AND NOT B) OR (B AND NOT A)**.This is like an OR gate but excluding both inputs being true. The output is true if inputs A and B are **DIFFERENT**. EX-OR gates can only have 2 inputs.



Input A	Input B	Output Q
0	0	0
0	1	1
1	0	1
1	1	0

Traditional symbol Truth Table

4. EX-NOR (EXclusive-NOR) gate

This is an EX-OR gate with the output inverted, as shown by the 'o' on the output. The output Q is true if inputs A and B are the **SAME** (both true or both false): **Q = (A AND B) OR (NOT A AND NOT B)**. EX-NOR gates can only have 2 inputs.

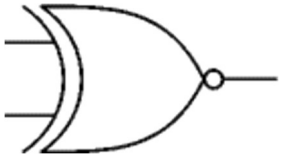
ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

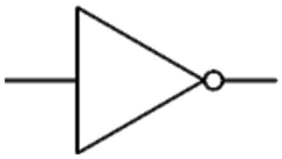


Input A	Input B	Output Q
0	0	1
0	1	0
1	0	0
1	1	1

Traditional symbol Truth Table

5. NOT gate (inverter)

The output Q is true when the input A is NOT true, the output is the inverse of the input: **Q = NOT A**. A NOT gate can only have one input. A NOT gate is also called an inverter.



Input A	Output Q
0	1
1	0

Traditional symbol Truth Table

Universal Gates

1. NOR gate (NOR = Not OR)

ST. FRANCIS INSTITUTE OF TECHNOLOGY

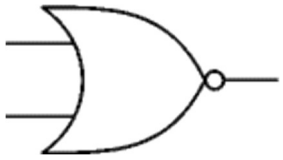
Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

This is an OR gate with the output inverted, as shown by the 'o' on the output. The output Q is true if NOT inputs A OR B are true: **Q = NOT (A OR B)** A NOR gate can have two or more inputs, its output is true if no inputs are true.

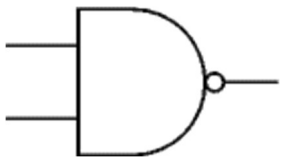


Input A	Input B	Output Q
0	0	1
0	1	0
1	0	0
1	1	0

Traditional symbol Truth Table

2. NAND gate (NAND = Not AND)

This is an AND gate with the output inverted, as shown by the 'o' on the output. The output is true if input A AND input B are NOT both true: **Q = NOT (A AND B)** A NAND gate can have two or more inputs, its output is true if NOT all inputs are true.



Input A	Input B	Output Q
0	0	1
0	1	1
1	0	1
1	1	0

Traditional symbol Truth Table

B. **De Morgan's Theorem: following are the expressions for De Morgan's Law.**

$$\overline{A + B} = \overline{A}\,\overline{B} \quad \text{and} \quad \overline{AB} = \overline{A} + \overline{B}$$

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

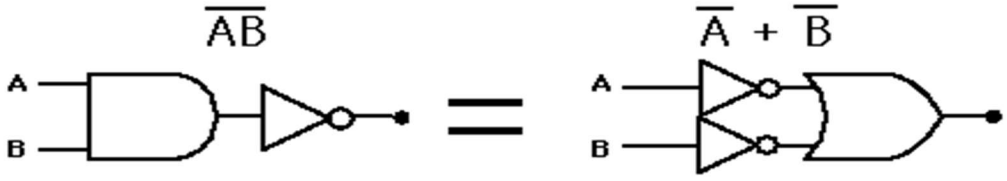
Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

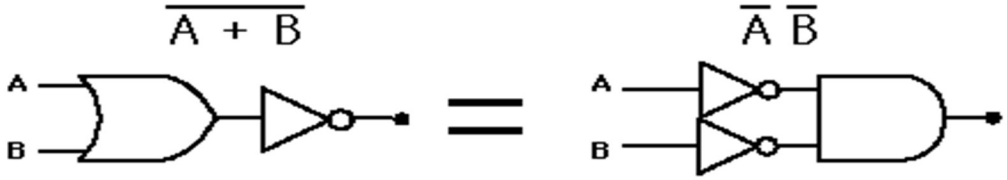
The most important logic theorem for digital electronics, this theorem says that any logical binary expression remains unchanged if we

1. Change all variables to their complements.
2. Change all AND operations to ORs.
3. Change all OR operations to ANDs.
4. Take the complement of the entire expression.

A practical operational way to look at De Morgan's Theorem is that the inversion bar of an expression may be broken at any point and the operation at that point replaced by its opposite (i.e., AND replaced by OR or vice versa). Two forms of De Morgan's Theorem implemented with basic gates.



A NAND gate is equivalent to an inversion followed by an OR



A NOR gate is equivalent to an inversion followed by an AND

Truth Table for De Morgan's Theorem(NAND = Inversion followed by an OR)

A	B				
0	0	1	1	1	1
0	1	1	0	1	1
1	0	0	1	1	1
1	1	0	0	0	0

Truth Table for De Morgan's Theorem(NOR = inversion followed by an AND)

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

A	B				
0	0	1	1	1	1
0	1	1	0	0	0
1	0	0	1	0	0
1	1	0	0	0	0

Questions

1. What are Universal gates? Why they are called as universal gates?

2. Explain laws of Boolean algebra.

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

DLCOA Exp 1

Name : Keegan Vaz
Sec: SE CMPN B
Roll No: 242

1) What are universal gates? Why are they called as universal gates?

→ A ~~universal~~ Universal gate is a gate which can implement any Boolean function without need to use any other gate type. NAND and NOR gates are called universal gates because they can perform all the three basic logic functions OR, AND and NOT

2) Explain laws of Boolean algebra

→ A set of rules or Laws of Boolean Algebra expressions have been invented to help reduce the number of logic gates needed to perform by a particular logic operation resulting in a list of functions or theorems known commonly as the laws of Boolean Algebra

$A \wedge B = B \wedge A$
$A \vee B = B \vee A$ → Commutative

$(A \wedge B) \wedge C = A \wedge (B \wedge C)$
$(A \vee B) \vee C = A \vee (B \vee C)$ → Associative

$\sim(A \vee B) = \sim A \wedge \sim B$
$\sim(A \wedge B) = \sim A \vee \sim B$ → De Morgan's Law

$A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$
$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$ → Distributive

$A \vee A = A$
$A \wedge A = A$ → Idempotent

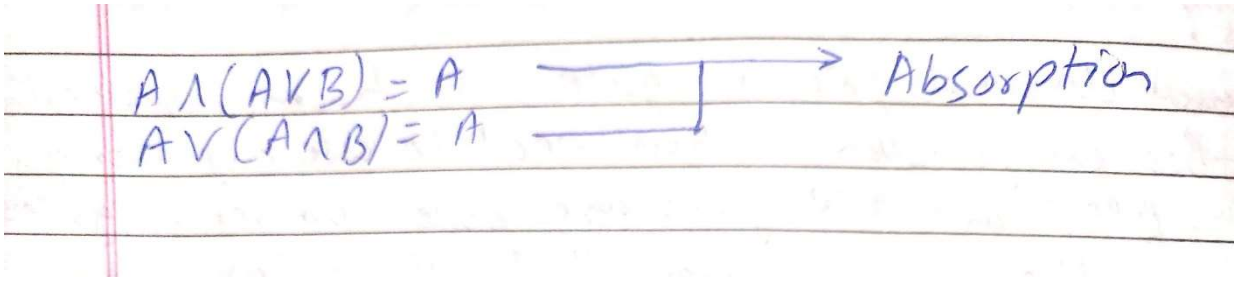
ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

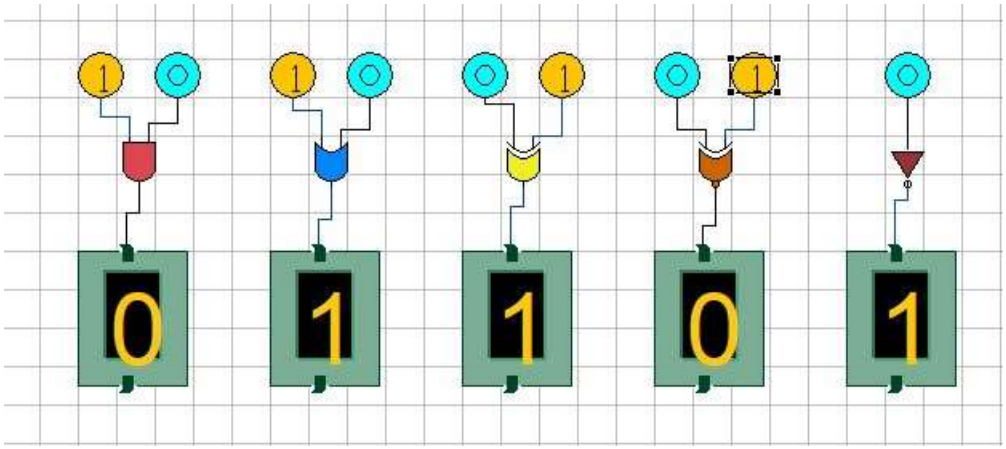
Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**



Outputs:

Logic Gates:



ST. FRANCIS INSTITUTE OF TECHNOLOGY

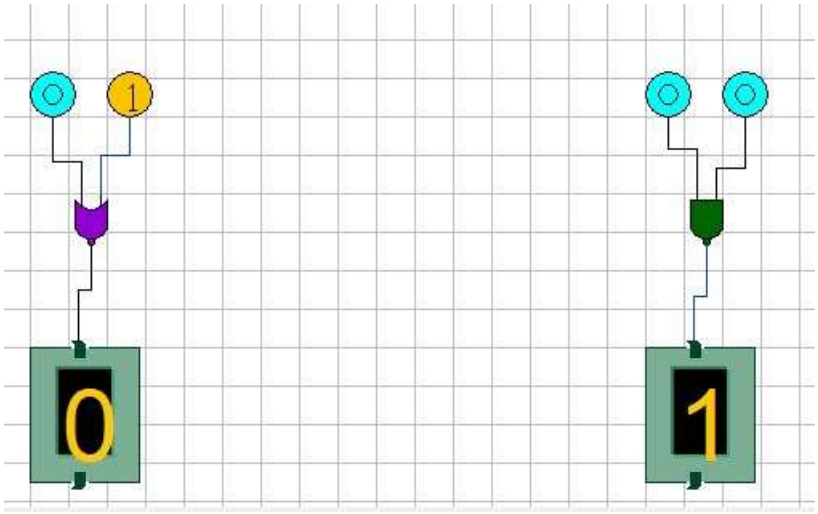
Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

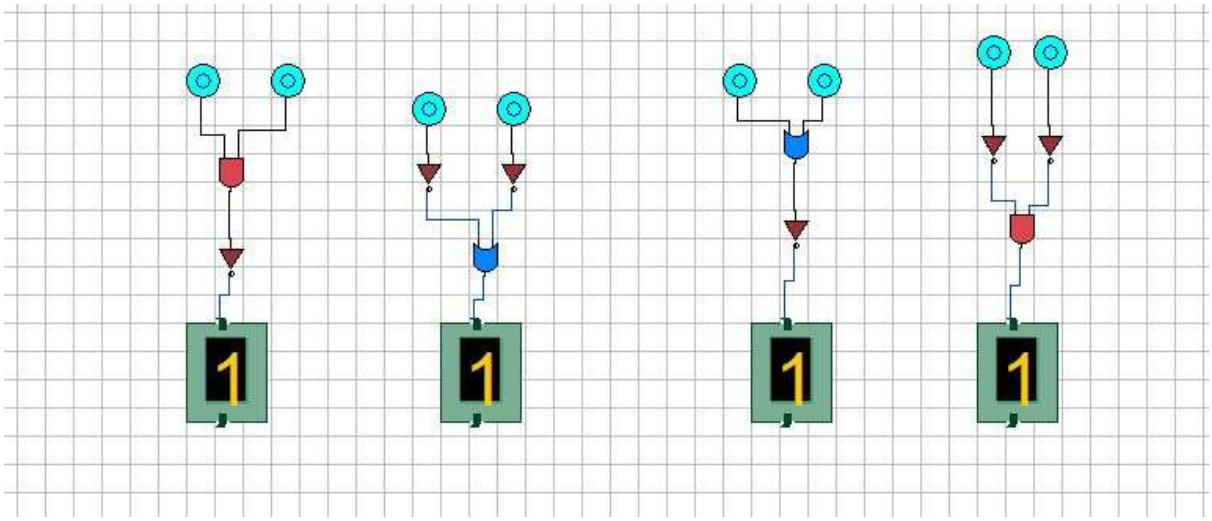
Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

Universal Gates:



De Morgan's Law:



ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

Experiment – 2: Realization of Basic Gates using Universal Gates

Date: 6th August 2020

1. **Aim:** Verify the truth tables using Universal gates and Basic gates.

2. **Requirements**

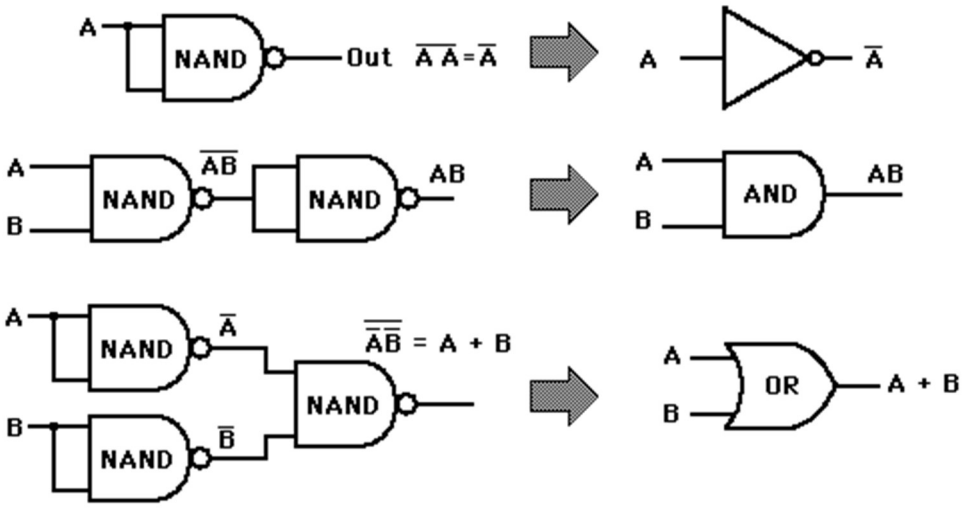
   JAVA, Simulator

3. **Pre-Experiment Exercise**
   Brief Theory

   1. **Using Universal Gates:** All the basic gates (e.g. AND, OR, Not etc.) can be implemented using universal gates (NAND, NOR).

A) NAND Gate Operations

The NAND gate is called a universal gate because combinations of it can be used to accomplish all the basic functions.



Truth Table for NOT Using NAND

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

A		
0	1	1
1	0	0

Truth Table for AND Using NAND

A	B		AB
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

Truth Table for OR Using NAND

A	B				A+B
0	0	1	1	0	0
0	1	1	0	1	1
1	0	0	1	1	1
1	1	0	0	1	1

B) NOR Gate Operations

The NOR gate is called a universal gate because combinations of it can be used to accomplish all the basic functions.

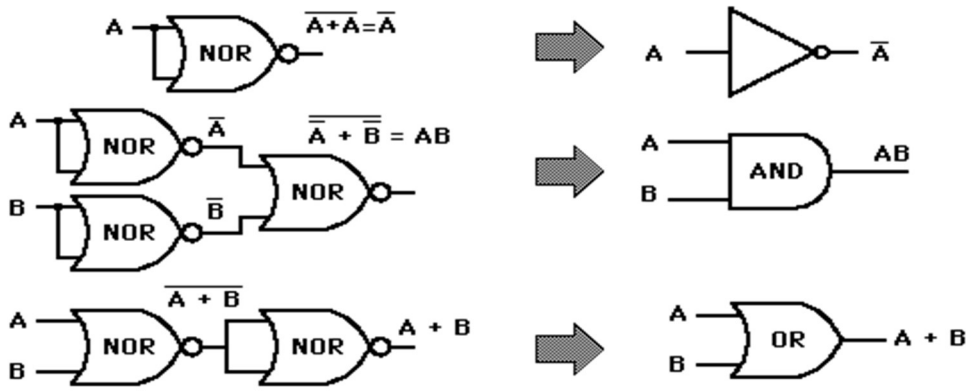
ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**



Truth Table for NOT Using NOR

A	A+A	
0	1	1
1	0	0

Truth Table for AND Using NOR

A	B				AB
0	0	1	1	0	0
0	1	1	0	0	0
1	0	0	1	0	0
1	1	0	0	1	1

Truth Table for OR Using NOR

A	B	A+B	A+B
0	0	0	0
0	1	1	1

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

1	0	1	1
1	1	1	1

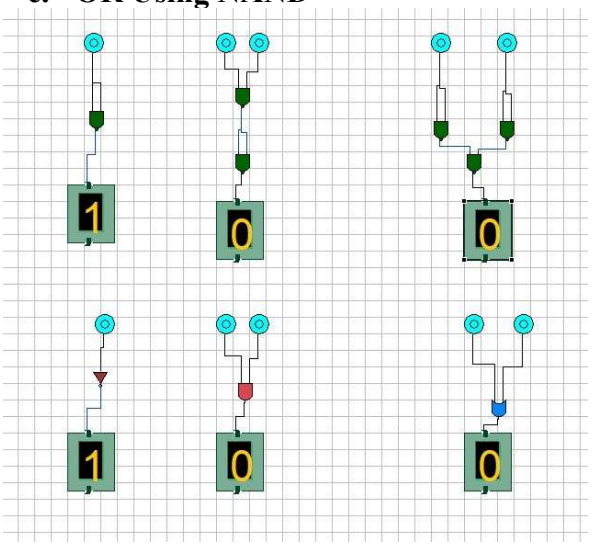
4. Laboratory Exercise

a. Observations Table

Universal Gates Operations:

1. NAND Gate Operations

a. Not Using NAND
b. AND Using NAND
c. OR Using NAND



ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

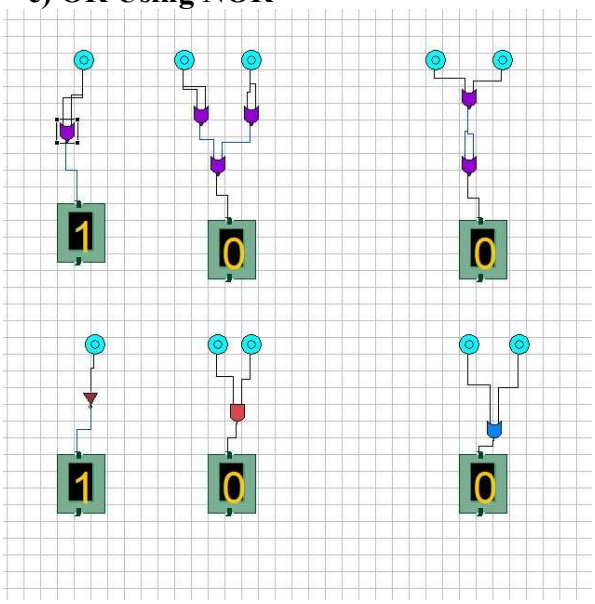
Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

2. NOR Gate Operations:

a) Not Using NOR
b) AND Using NOR
c) OR Using NOR



ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

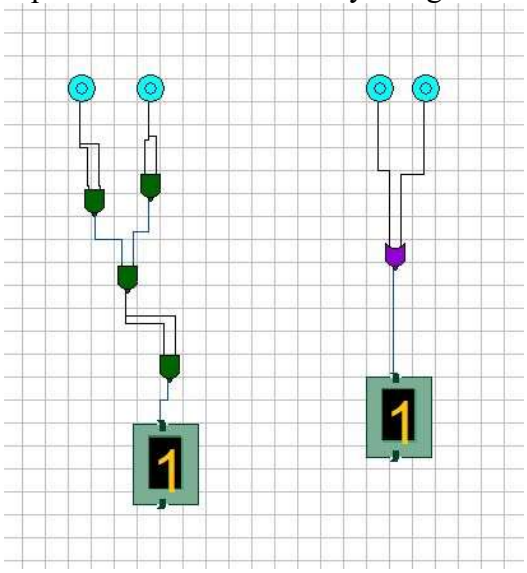
Roll No: 242 **PID: 192120**

5. Post-Experiment Exercise

A) Conclusion/Comments

B) Questions

1. Implement NOR function by using NAND gates.



ST. FRANCIS INSTITUTE OF TECHNOLOGY

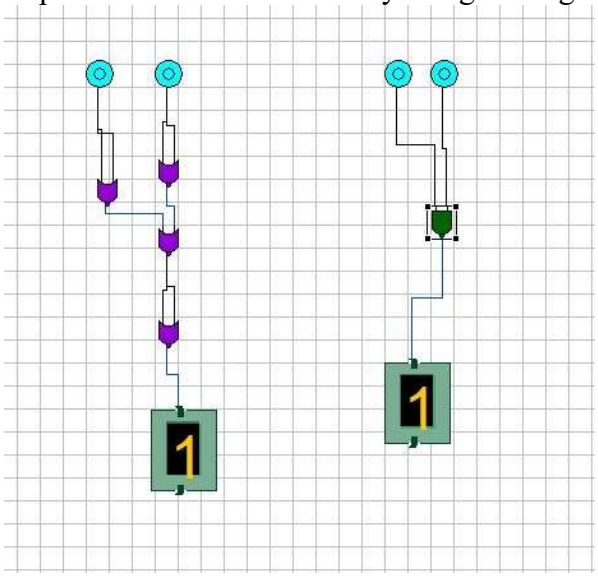
Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

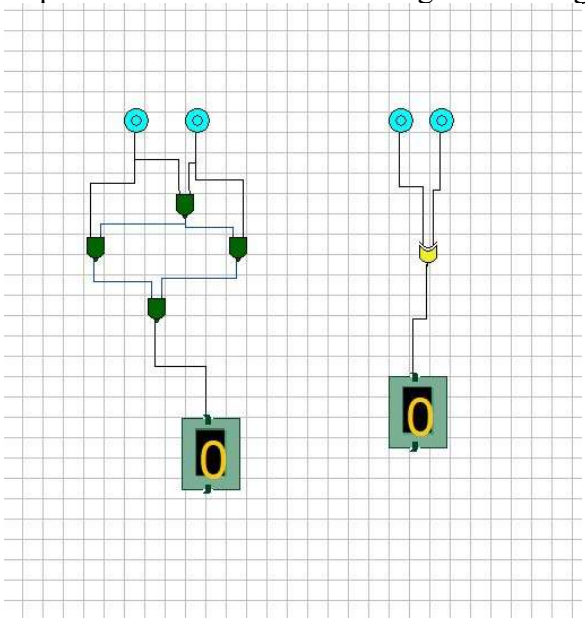
Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

2. Implement NAND function by using NOR gates.



3. Implement Ex-OR function using Universal gates.



ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

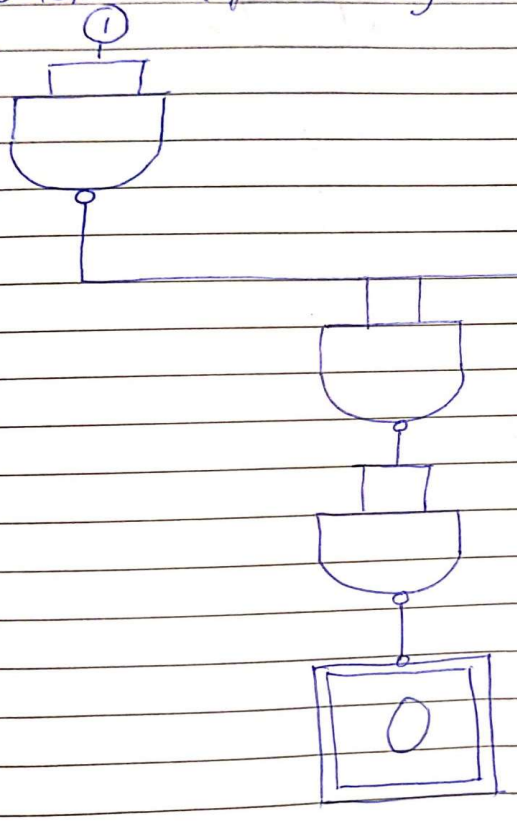
DLCOA Exp 2

Name : Keegan Vaz

Sec: SE CMPN B

Roll No: 242

Q1) Implement NOR function by using NAND gates



ST. FRANCIS INSTITUTE OF TECHNOLOGY

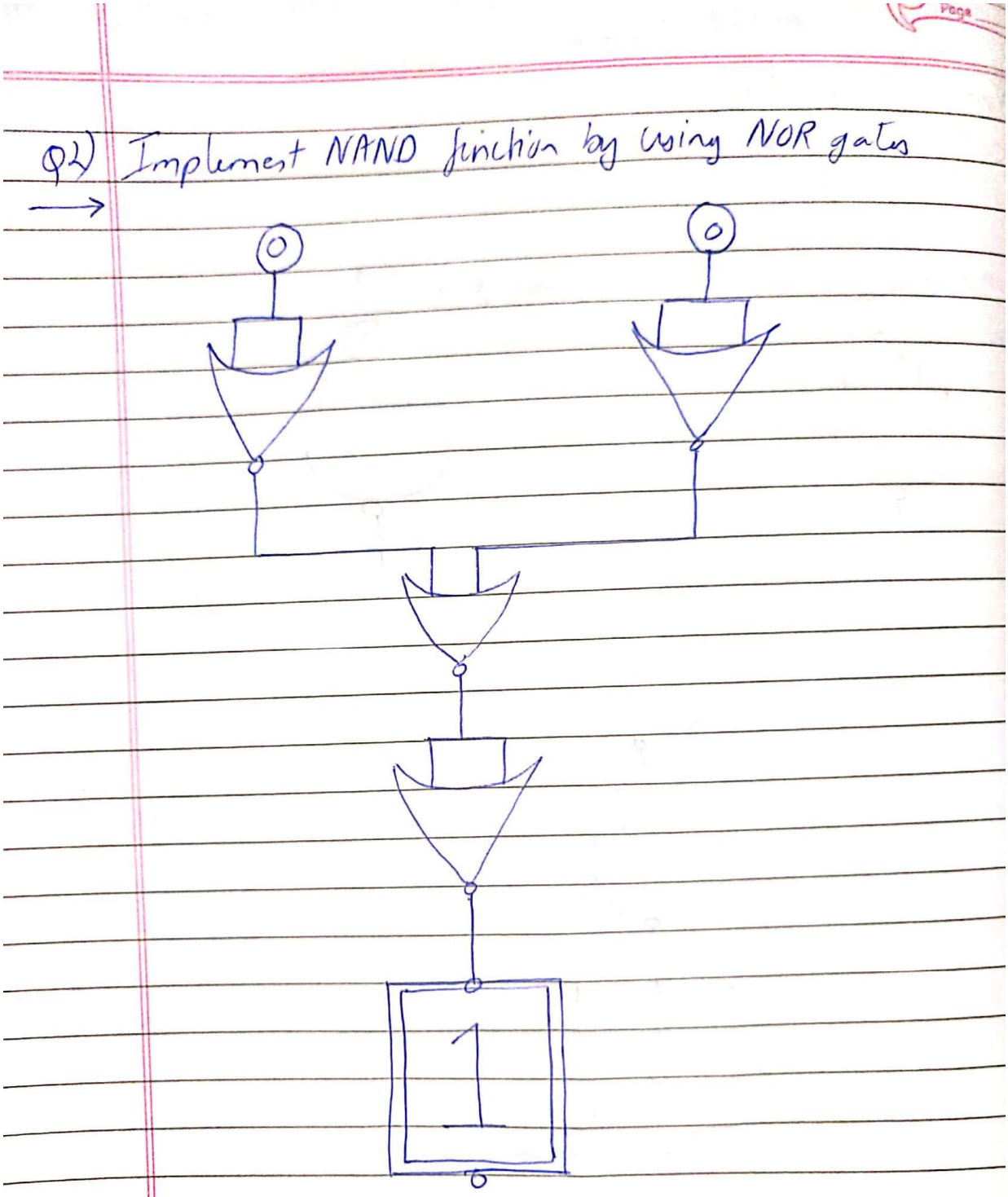
Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

Q2) Implement NAND function by using NOR gates



ST. FRANCIS INSTITUTE OF TECHNOLOGY

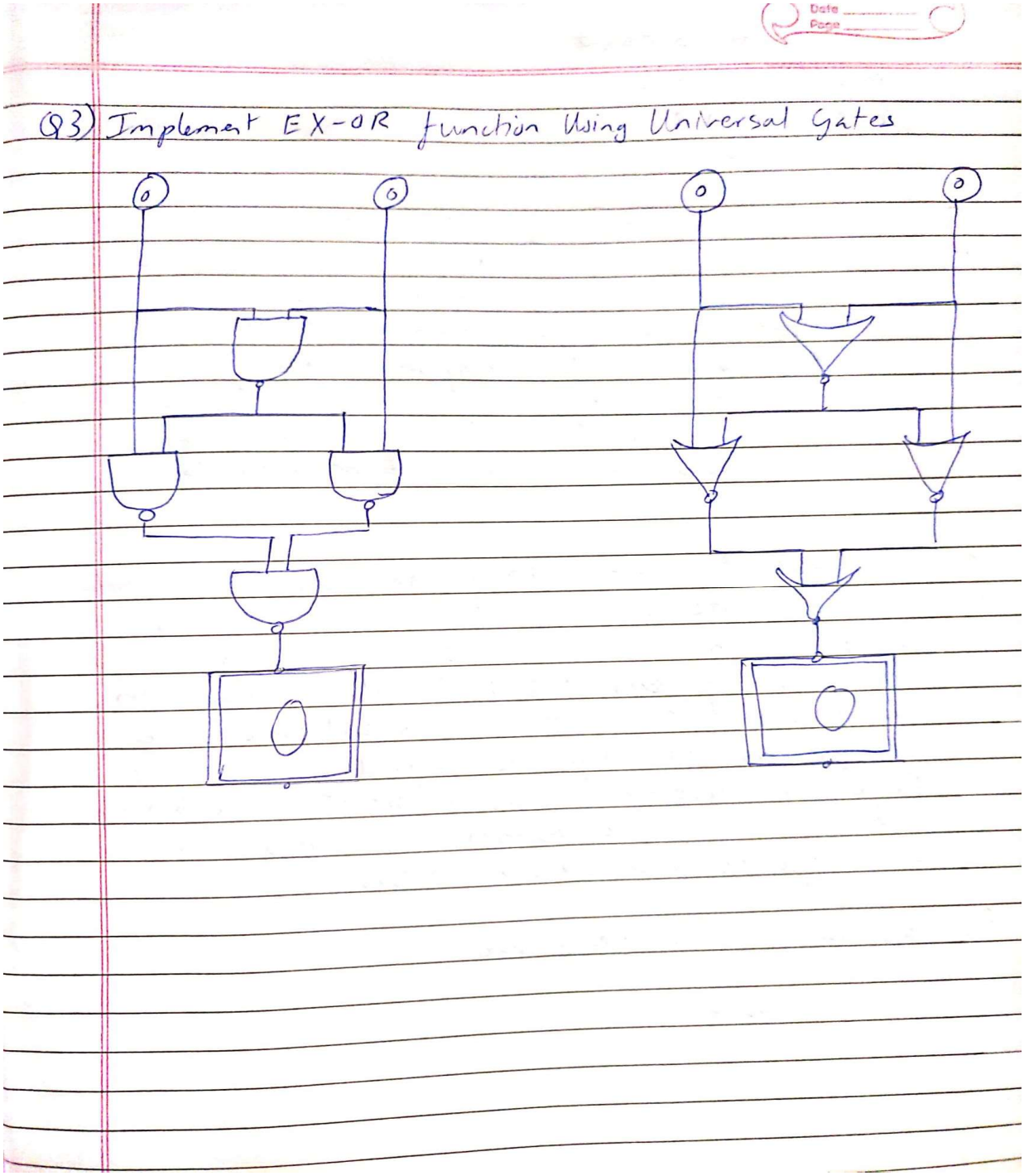
Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

(Q3) Implement EX-OR function Using Universal Gates



ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

Experiment – 3: To Realize Binary to Gray Code and Gray Code to Binary Converter.

Date: 16th October 2020

1. **Aim:** Design and implement code conversion circuit
   i) Binary to gray code converter
   ii) Gray to binary converter

1. **Requirements**
   JAVA, Simulator
3. **Pre-Experiment Exercise**
   Brief Theory

Gray Code is one of the most important codes. It is a non-weighted code which belongs to a class of codes called minimum change codes. In this codes while traversing from one step to another step only one bit in the code group changes.

Conversion from Binary to Gray code: Following are the steps for converting binary into gray code.

1. The left most significant bit of given binary code number is same as most left significant bit of gray number
2. To obtain the successive gray bits for given binary code, add the first bit of binary number to second one and write down the result next to the first bit, add second bit and repeat the same operation until last bit

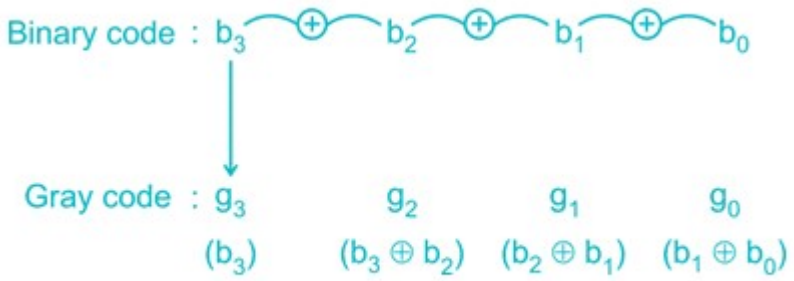
ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

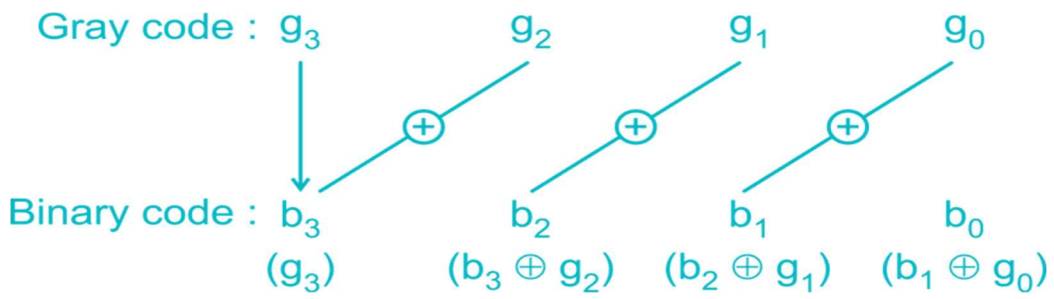
Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**



Conversion from Gray to Binary code: Following are the steps for converting gray to binary code.

1. The left most significant bit of given gray code number is same as most left significant bit of binary number
2. To obtain the successive binary bits for given gray code, add the second bit of gray code number to first bit of binary and write down the result next to the first bit and repeat the same operation until last bit



Truth Table (Binary to Gray Code)

Binary code input				**Gray code output**			
B3	B2	B1	B1	G3	G2	G1	G0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

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

Truth Table (Gray to Binary Code)

Gray code input				**Binary code output**			
G3	G2	G1	G0	B3	B2	B1	B1
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	0	0	1	1	1	1	0

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

1	0	0	0	1	1	1	1

4. Laboratory Exercise

A) Procedure

B) Observation Table

Binary to Gray Code

Binary Code Input				**Gray Code Output(Logic State)**			
B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

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

Gray to Binary Code

Gray Code Input				**Binary Code Output(Logic State)**			
G_3	G_2	G_1	G_0	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

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

C) Code

Grey To Binary:

```
#include<stdio.h>
#include<conio.h>
void main()
{
	int a[10],i=0,c=0,n;
	printf("\n enter the gray code");
	scanf("%d",&n);
	while(n!=0)
	{
		a[i]=n%10;
		n/=10;
		i++;
		c++;
	}
	for(i=c-1;i>=0;i--)
	{
		if(a[i]==1)
		{
			if(a[i-1]==1)
			a[i-1]=0;
			else
			a[i-1]=1;
		}
	}
	printf("\n the binary code is");
	for(i=c-1;i>=0;i--)
	printf("%d",a[i]);
	getch();
}
```

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

Output:

```
enter the gray code1001
the binary code is1110
```

Binary to Gray Code:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a[10],i=0,c=0,n,b[10];
        printf("\n enter the binary code");
        scanf("%d",&n);
        while(n!=0)
        {
                a[i]=n%10;
                n/=10;
                i++;
                c++;
        }
        for(i=c-1;i>=0;i--)
        {
                b[i]=a[i];
```

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

```
        }
        for(i=c-1;i>=0;i--)
        {
                if(b[i]==1)
                {
                        if(a[i-1]==1)
                        a[i-1]=0;
                        else
                        a[i-1]=1;
                }
        }
        printf("\n the gray code is");
        for(i=c-1;i>=0;i--)
        printf("%d",a[i]);
        getch();
  }
```

Output:

```
enter the binary code10001
the gray code is11001_
```

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

5. Post-Experiment Exercise

A) Conclusion/Comments

From the above experiment we are able to convert binary code to grey code and vice versa. We also learn that the EX-OR condition is necessary for these conversions.

B) Questions

1. What is gray code? Why is it called as unit distance code?
2. Write short note on application of gray code (Shaft encoder).
3. Represent $(35)_{10}$ in gray code.

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

DLCOA Exp 3

Name: Keegan Vaz
Sec: SE CMPN B
Roll No: 242

Q1) What is gray code? Why is it called as unit distance co

→ Gray code is a non-weighted code which belongs to a class of codes called minimum change codes. In this codes while travessing from one step to another step only one bit in the code group changes. Adjacent codes differ only in one bit. Therefore they are known as grey codes. It is called as unit distance as there is only one bit change between the consecutive numbers.

Q2) Write short note on application of gray code (shaft encoder

→ Gray codes are widely used to prevent spurious output from electrochemical switches and to facilitat error correction in digital communication such as digital terrestrial television and some cable TV systems. The Gray Code is used to eliminate the error problem which is inherient in the binary code. The gray code assures that only one bit will change between adjacent sectors.

Q3) Represent $(35)_{10}$ in gray code

$(35)_{10} = (110010)$

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

Experiment4: To design and verify the circuit of i) Half Adder ii) Full Adder

Date: 16th October 2020

1. **Aim** 1.Design and verify the circuit for following:
   i) Half Adder and Full Adder.
2. **Requirements**
   Java, Simulator
3. **Pre-Experiment Exercise**
   Brief Theory

I.Types of adders: For single bit adders, there are two general types.

A **half adder** has two inputs, generally labelled *A* and B, and two outputs, the sum S and carry C. S is the two-bit XOR of A and B, and C is the AND of A and B. Essentially the output of a half adder is the sum of two one-bit numbers, with C being the most significant of these two outputs.

A **full adder** has three inputs - *A*, *B*, and a carry in *C*, such that multiple adders can be used to add larger numbers. To remove ambiguity between the input and output carry lines, the carry in is labelled C_i or C_{in} while the carry out is labelled C_o or C_{out}.

a. Half adder: A half adder is a logical circuit that performs an addition operation on two binary digits. The half adder produces a sum and a carry value which are both binary digits. The drawback of this circuit is that in case of a multibit addition, it cannot include a carry.A half adder takes in two inputs A and B, adds, and returns the resulting sum S and a carry out C. Its logic table and Truth table is shown below.

Input		**Output**	
A	**B**	**CARRY(S)**	**SUM(S)**
0	0	0	0
0	1	0	1

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

1	0	0	1
1	1	1	0

b) Full adder: Inputs: {A, B, CarryIn} → Outputs: {Sum, CarryOut}

A **full adder** is a logical circuit that performs an addition operation on three binary digits. The full adder produces a sum and carry value, which are both binary digits. A full adder is more powerful than a half adder. Unlike a half adder, it can also take in a carry-in bit, and uses that in a calculation. A full adder takes in three inputs, Cin, A, and B, adds, and outputs the resulting sum S, and a carry out Cout. Its logic table and Truth table is shown below.

Input			**Output**	
A	**B**	**CarryIn(Cin)**	**CarryOut(Cout)**	**Sum (S)**
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

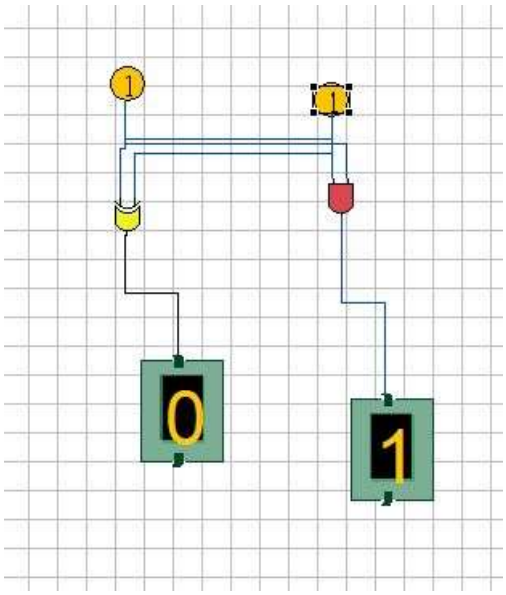
1	1	0	1	0
1	1	1	1	1

A full adder can be constructed from two half adders by connecting *A* and *B* to the input of one half adder, connecting the sum from that to an input to the second adder, connecting C_i to the other input and OR the two carry outputs. Equivalently, *S* could be made the three-bit xor of *A*, *B*, and C_i and C_o could be made the three-bit majority function of *A*, *B*, and C_i. The output of the full adder is the two-bit arithmetic sum of three one-bit numbers.

A) Procedure

B) Implementation

Half Adder



ST. FRANCIS INSTITUTE OF TECHNOLOGY

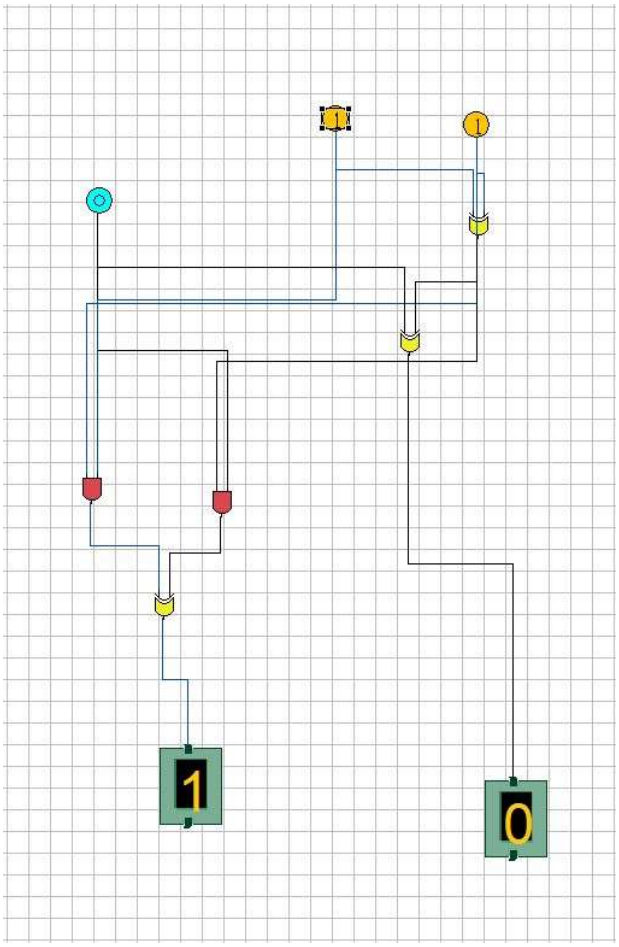
Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

Full Adder



Post-Experiment Exercise

A) Conclusion/Comments

B) Questions

ST. FRANCIS INSTITUTE OF TECHNOLOGY

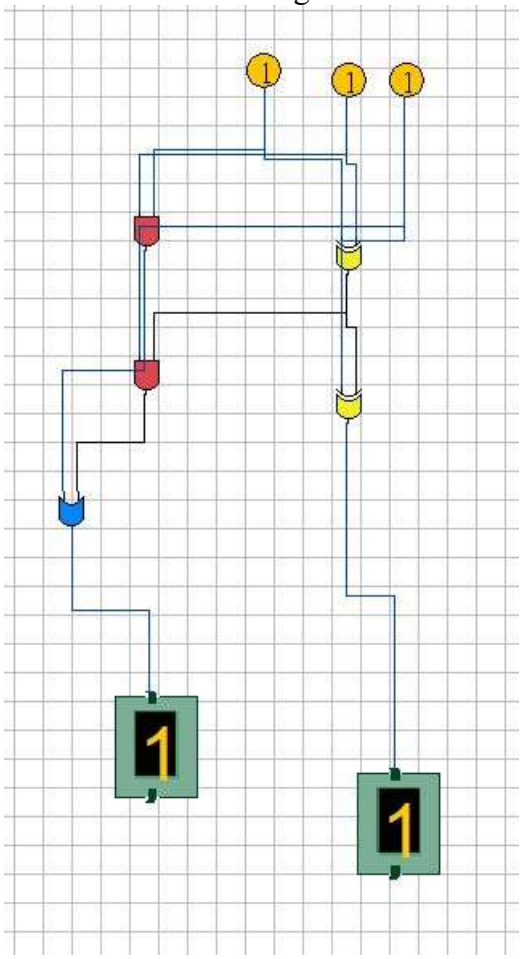
Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

1)Realize a full adder using two HALF adders and one OR gate.



2)Perform division of decimal 9 and 3 using binary arithmetic

3) Subtract using 1's and 2's complement method

a) $(56)_{10} - (76)_{10}$

b) $(15)_{10} - (21)_{10}$

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

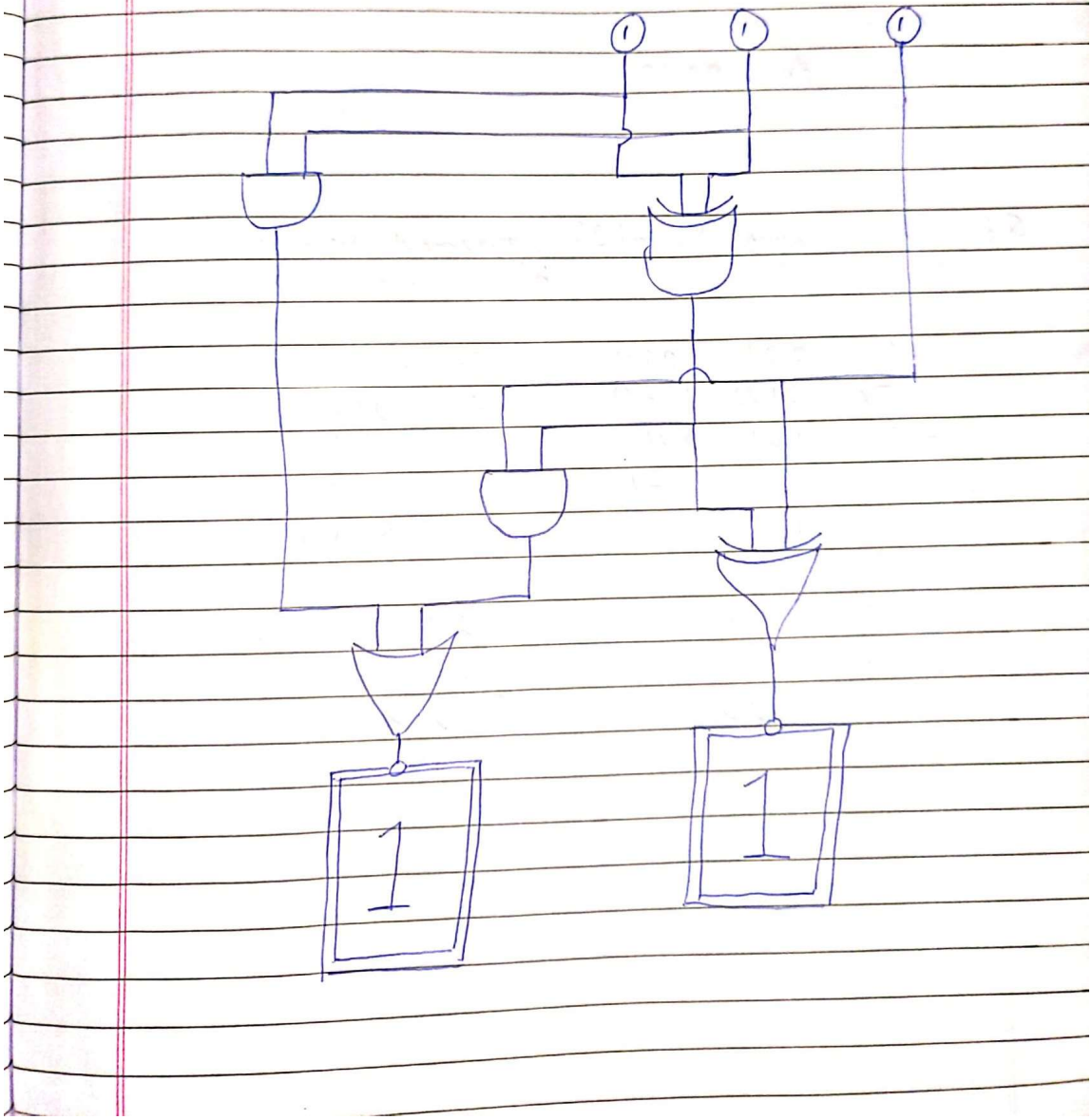
Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

DLCOA Exp 4

Name: KEEGAN VAZ
Sec: SE CMPN B
Roll No: 242

Q1] Realize a full adder using two HALF adder and one OR gate



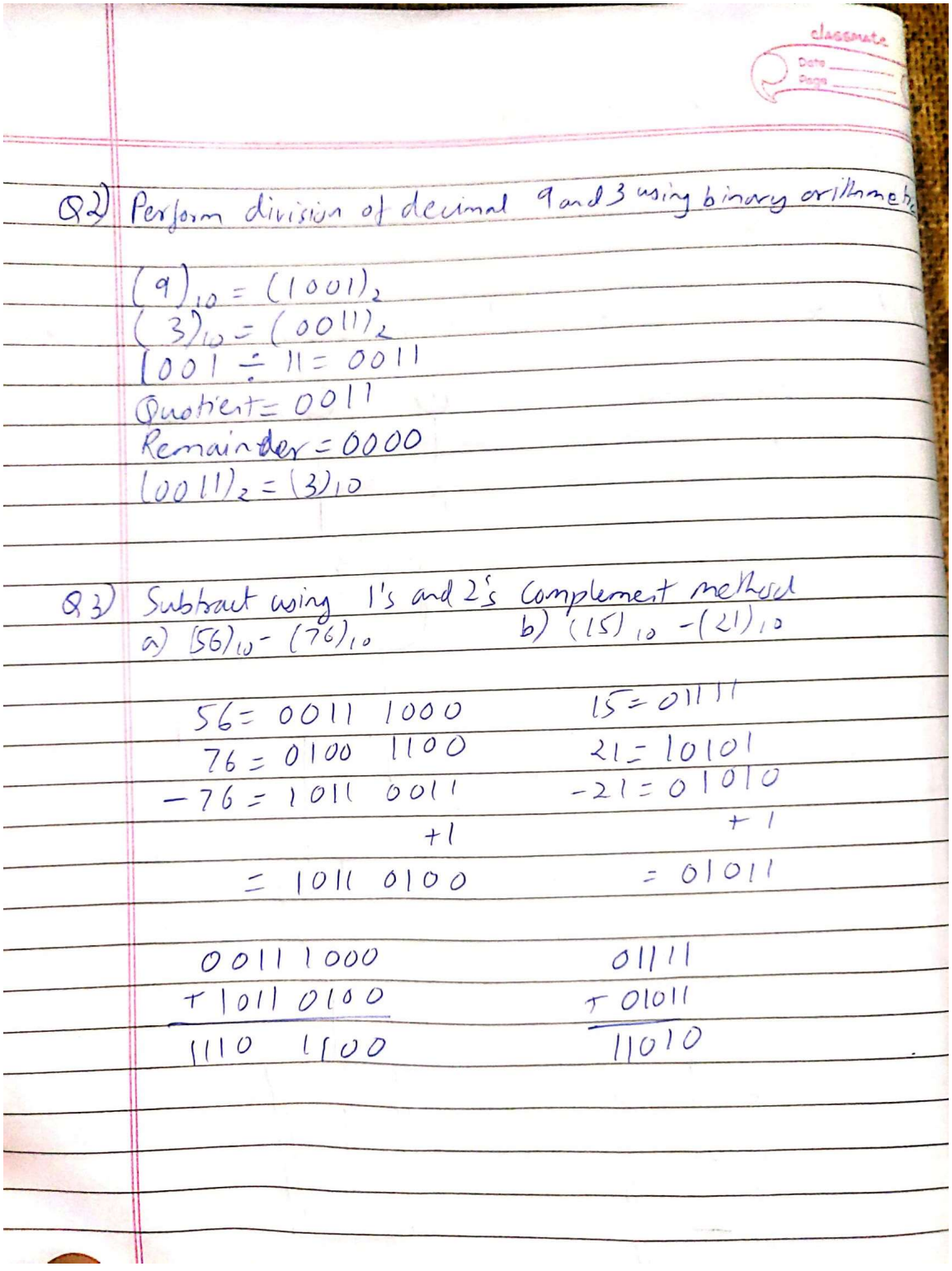
ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**



Q2) Perform division of decimal 9 and 3 using binary arithmetic

$(9)_{10} = (1001)_2$
$(3)_{10} = (0011)_2$
$1001 \div 11 = 0011$
Quotient = 0011
Remainder = 0000
$(0011)_2 = (3)_{10}$

Q3) Subtract using 1's and 2's Complement method
a) $(56)_{10} - (76)_{10}$ b) $(15)_{10} - (21)_{10}$

56 = 0011 1000
76 = 0100 1100
−76 = 1011 0011
+1
= 1011 0100

0011 1000
+ 1011 0100
1110 1100

15 = 01111
21 = 10101
−21 = 01010
+1
= 01011

01111
+ 01011
11010

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

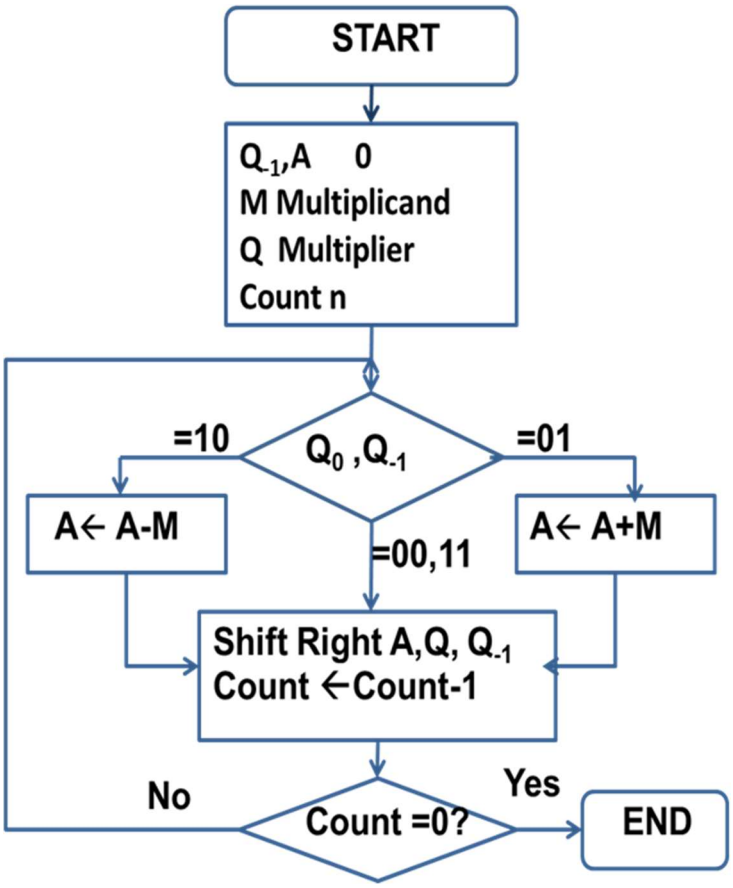
Experiment-5

Aim:- To implement Booth's Multiplication

Algorithm.

Theory:

Booth's multiplication algorithm is a multiplication algorithm that multiplies two signed binary numbers in two's complement notation. The flowchart for the algorithm is as follows:



ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

Code:

```
#include <stdio.h>
#include <math.h>

int a = 0,b = 0, c = 0, a1 = 0, b1 = 0, com[5] = { 1, 0, 0, 0, 0};
int anum[5] = {0}, anumcp[5] = {0}, bnum[5] = {0};
int acomp[5] = {0}, bcomp[5] = {0}, pro[5] = {0}, res[5] = {0};

void binary()
{
 int r, r2, i, temp;
    a1 = fabs(a);
    b1 = fabs(b);
    for (i = 0; i < 5; i++)
          {
        r = a1 % 2;
        a1 = a1 / 2;
        r2 = b1 % 2;
        b1 = b1 / 2;
        anum[i] = r;
        anumcp[i] = r;
        bnum[i] = r2;
        if(r2 == 0)
```

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

```
                {
        bcomp[i] = 1;
    }
    if(r == 0)
                {
        acomp[i] =1;
    }
}
c = 0;
for ( i = 0; i < 5; i++)
     {
    res[i] = com[i]+ bcomp[i] + c;
    if(res[i] >= 2)
                {
        c = 1;
    }
    else
        c = 0;
    res[i] = res[i] % 2;
}
for (i = 4; i >= 0; i--)
     {
bcomp[i] = res[i];
```

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

```
}
if (a < 0)
      {
 c = 0;
 for (i = 4; i >= 0; i--)
       {
   res[i] = 0;
 }
 for ( i = 0; i < 5; i++)
       {
   res[i] = com[i] + acomp[i] + c;
   if (res[i] >= 2){
       c = 1;
   }
   else
       c = 0;
   res[i] = res[i]%2;
 }
 for (i = 4; i >= 0; i--)
       {
   anum[i] = res[i];
   anumcp[i] = res[i];
 }
```

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

```
    }
    if(b < 0){
    for (i = 0; i < 5; i++){
        temp = bnum[i];
        bnum[i] = bcomp[i];
        bcomp[i] = temp;
    }
    }
}

void add(int num[]){
   int i;
   c = 0;
   for ( i = 0; i < 5; i++){
        res[i] = pro[i] + num[i] + c;
        if (res[i] >= 2){
            c = 1;
        }
        else{
            c = 0;
        }
        res[i] = res[i]%2;
    }
```

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

```
    for (i = 4; i >= 0; i--){
        pro[i] = res[i];
        printf("%d",pro[i]);
    }
  printf(":");
  for (i = 4; i >= 0; i--){
        printf("%d", anumcp[i]);
    }
}

void arshift()
{
   int temp = pro[4], temp2 = pro[0], i;
   for (i = 1; i < 5  ; i++)
   {
     pro[i-1] = pro[i];
   }
   pro[4] = temp;
   for (i = 1; i < 5  ; i++)
   {
      anumcp[i-1] = anumcp[i];
   }
   anumcp[4] = temp2;
```

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

```
    printf("\nAR-SHIFT: ");
    for (i = 4; i >= 0; i--){
        printf("%d",pro[i]);
    }
    printf(":");
    for(i = 4; i >= 0; i--){
        printf("%d", anumcp[i]);
    }
}

int main(){
  int i, q = 0;
  printf("Multiplying two numbers using booth's multiplication algorithm");
  printf("\nEnter two numbers to multiply which are less then 16");
     do{
       printf("\nEnter A: ");
       scanf("%d",&a);
       printf("Enter B: ");
       scanf("%d", &b);
    }while(a >=16 || b >=16);
    printf("\nExpected product = %d", a * b);
    binary();
    printf("\n\nBinary Equivalents are: ");
```

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

```
printf("\nA = ");
for (i = 4; i >= 0; i--){
   printf("%d", anum[i]);
}
printf("\nB = ");
for (i = 4; i >= 0; i--){
   printf("%d", bnum[i]);
}
printf("\nB'+ 1 = ");
for (i = 4; i >= 0; i--){
   printf("%d", bcomp[i]);
}
printf("\n\n");
for (i = 0;i < 5; i++){
      if (anum[i] == q){
         printf("\n-->");
         arshift();
         q = anum[i];
      }
      else if(anum[i] == 1 && q == 0)
{
         printf("\n-->");
         printf("\nSUB B: ");
```

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**


```
            add(bcomp);
            arshift();
            q = anum[i];
          }
          else
    {
            printf("\n-->");
            printf("\nADD B: ");
            add(bnum);
            arshift();
            q = anum[i];
          }
      }
      printf("\nProduct is = ");
      for (i = 4; i >= 0; i--){
          printf("%d", pro[i]);
      }
      for (i = 4; i >= 0; i--){
          printf("%d", anumcp[i]);
      }
  }
```

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

Output:

```
Multiplying two numbers using booth's multiplication algorithm
Enter two numbers to multiply which are less then 16
Enter A: 13
Enter B: 14

Expected product = 182

Binary Equivalents are:
A = 01101
B = 01110
B'+ 1 = 10010


-->
SUB B: 10010:01101
AR-SHIFT: 11001:00110
-->
ADD B: 00111:00110
AR-SHIFT: 00011:10011
-->
SUB B: 10101:10011
AR-SHIFT: 11010:11001
-->
AR-SHIFT: 11101:01100
-->
ADD B: 01011:01100
AR-SHIFT: 00101:10110
Product is = 0010110110
```

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

Viva Questions:

1. Explain the steps of Booth's Algorithm.
2. Find 3 by (-4), using Booth's Multiplication Algorithm

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

DLCOA Exp 5

Name : KEEGAN VAZ
Sec: SE CMPN B
Roll No: 242

Q1) Explain The Steps Of Booth's Algorithm

→ a. Multiplier and Multiplicand are placed in the Q and M register respectively.
b. Results for this will be stored in AC and Q registers
c. Initially, AC and Q, register will be 0
d. Multiplication of a number is done in a cycle
e. A 1-bit register Q_{-1} is placed right of the least significant bit Q_0 of the register Q
f. In each of the cycles, Q_0 and Q_{-1} bits will be checked.
1. If Q_0 and Q_1 are 11 or 00 then the bits of AC, Q and Q_{-1} are shifted to the right by 1 bit
2. If the values is shown 01 then multiplicand is added to AC. After addition AC, Q_0, Q_{-1} registered register are shifted to the right by 1 bit.
3 If the value is shown as 10 then multiplicand is subtracted from AC. After Subtraction AC, Q_0, Q_{-1} register is shifted to the right by 1 bit

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

```
Multiplying two numbers using booth's multiplication algorithm
Enter two numbers to multiply which are less then 16
Enter A: 3
Enter B: -4

Expected product = -12

Binary Equivalents are:
A = 00011
B = 11100
B'+ 1 = 00100



-->
SUB B: 00100:00011
AR-SHIFT: 00010:00001
-->
AR-SHIFT: 00001:00000
-->
ADD B: 11101:00000
AR-SHIFT: 11110:10000
-->
AR-SHIFT: 11111:01000
-->
AR-SHIFT: 11111:10100
Product is = 1111110100
```

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

Conclusion

From the above experiment we are able to understand about Booth's Multiplication algorithm. We can easily multiply two signed binary numbers using two's complement.

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

Experiment No.: 6

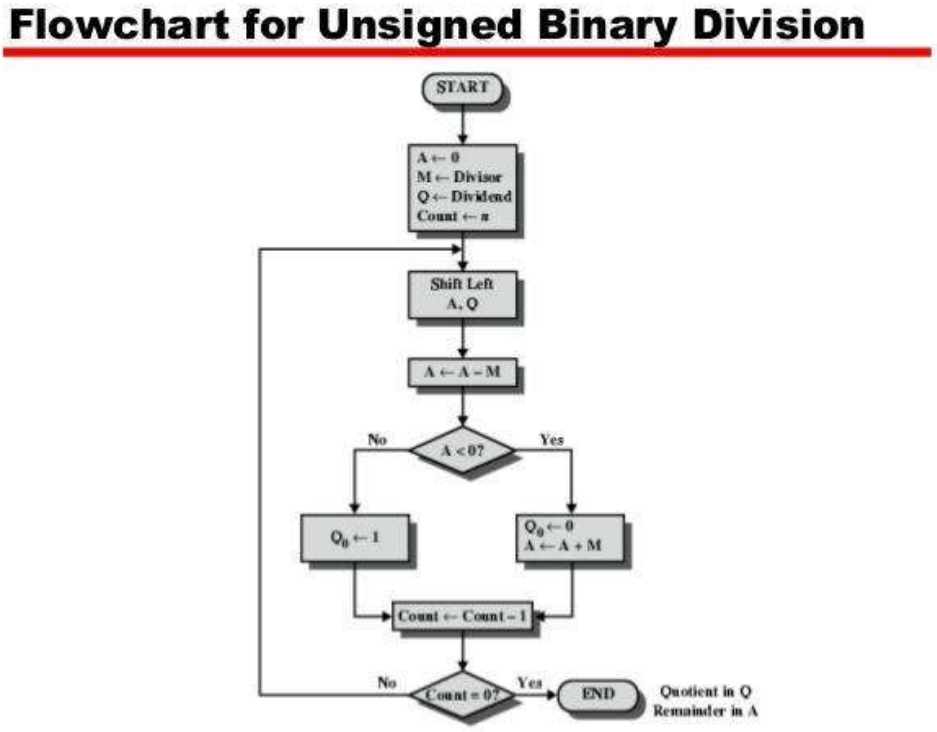
Aim:- To implement Restoring division algorithm.

Theory:-

There are number of binary division algorithm. One of them is restoring division algorithm. The number of steps required in the algorithm is equal to the number of bits in the dividend.

On completion of the algorithm, Q will get the quotient and A will get the remainder..

Flowchart of Restoring Division Algorithm



ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

Code:

```
#include <stdio.h>
#include <conio.h>
#include <math.h>

int a=0,b=0,c=0,com[5]={1,0,0,0,0},s=0;
int anum[5]={0},anumcp[5] ={0},bnum[5]={0};
int acomp[5]={0},bcomp[5]={0},rem[5]={0},quo[5]={0},res[5]={0};

void binary()
{
        int r, r2, i, temp;
    a = fabs(a);
    b = fabs(b);
    for(i = 0; i < 5; i++){
        r = a % 2;
        a = a / 2;
        r2 = b % 2;
        b = b / 2;
        anum[i] = r;
        anumcp[i] = r;
        bnum[i] = r2;
        if(r2 == 0){
            bcomp[i] = 1;
        }
```

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

```
        if(r == 0){
            acomp[i] =1;
        }
    }
  c = 0;
  for( i = 0; i < 5; i++){
        res[i] = com[i]+ bcomp[i] + c;
        if(res[i]>=2){
            c = 1;
        }
        else
            c = 0;
        res[i] = res[i]%2;
    }
  for(i = 4; i>= 0; i--){
    bcomp[i] = res[i];
  }
}

void add(int num[]){
    int i;
    c = 0;
    for( i = 0; i < 5; i++){
        res[i] = rem[i]+ num[i] + c;
        if(res[i]>=2)
```

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**


```
        {
            c = 1;
        }
        else
            c = 0;
        res[i] = res[i]%2;
    }
    for(i = 4; i>= 0; i--){
        rem[i] = res[i];
        printf("%d",rem[i]);
    }
    printf(":");
    for(i = 4; i>= 0; i--){
        printf("%d",anumcp[i]);
    }
}

void shl(){
    int i;
    for(i = 4; i > 0  ; i--){
        rem[i] = rem[i-1];
    }
    rem[0] = anumcp[4];
    for(i = 4; i > 0  ; i--){
        anumcp[i] = anumcp[i-1];
```

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**


```
    }
    anumcp[0] = 0;
    printf("\nSHIFT LEFT: ");
    for(i = 4; i>= 0; i--){
        printf("%d",rem[i]);
    }
    printf(":");
    for(i = 4; i>= 0; i--){
        printf("%d",anumcp[i]);
    }
}

int main()
{
    int i;
    printf("RESTORING DIVISION ALGORITHM");
    printf("\nEnter two numbers to divide and both the numbers should be less than 16");
    do{
        printf("\nEnter A: ");
        scanf("%d",&a);
        printf("Enter B: ");
        scanf("%d",&b);
    }
 while(a>=16 || b>=16);
    printf("\nExpected Quotient = %d", a/b);
```

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

```
printf("\nExpected Remainder = %d", a%b);
if(a*b <0){
    s = 1;
}
binary();
printf("\n\nUnsigned Binary Equivalents are: ");
printf("\nA = ");
for(i = 4; i>= 0; i--){
    printf("%d",anum[i]);
}
printf("\nB = ");
for(i = 4; i>= 0; i--){
    printf("%d",bnum[i]);
}
printf("\nB'+ 1 = ");
for(i = 4; i>= 0; i--){
    printf("%d",bcomp[i]);
}
printf("\n\n-->");
shl();
for(i=0;i<5;i++){
    printf("\n-->");
    printf("\nSUB B: ");
    add(bcomp);
    if(rem[4]==1){
```

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

```
            printf("\n-->RESTORE");
            printf("\nADD B: ");
            anumcp[0] = 0;
            add(bnum);
        }
        else{
            anumcp[0] = 1;
        }
        if(i<4)
            shl();
    }
    printf("\n");
    printf("\nSign of the result = %d",s);
    printf("\nRemainder is = ");
    for(i = 4; i>= 0; i--){
        printf("%d",rem[i]);
    }
    printf("\nQuotient is = ");
    for(i = 4; i>= 0; i--){
        printf("%d",anumcp[i]);
    }
return 0;
}
```

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

Output:

```
RESTORING DIVISION ALGORITHM
Enter two numbers to divide and both the numbers should be less than 16
Enter A: 15
Enter B: 5

Expected Quotient = 3
Expected Remainder = 0

Unsigned Binary Equivalents are:
A = 01111
B = 00101
B'+ 1 = 11011

-->
SHIFT LEFT: 00000:11110
-->
SUB B: 11011:11110
-->RESTORE
ADD B: 00000:11110
SHIFT LEFT: 00001:11100
-->
SUB B: 11100:11100
-->RESTORE
ADD B: 00001:11100
SHIFT LEFT: 00011:11000
-->
SUB B: 11110:11000
-->RESTORE
ADD B: 00011:11000
SHIFT LEFT: 00111:10000
-->
SUB B: 00010:10000
SHIFT LEFT: 00101:00010
-->
SUB B: 00000:00010

Sign of the result = 0
Remainder is = 00000
Quotient is = 00011
```

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

Conclusion:-In this experiment we have learnt the theory concept behind the Restoring Division Algorithm and how to implement it in C language

Viva Questions:-

1. Find 8/4, using Restoring Division Algorithm
2. Explain the steps of Restoring Division Algorithm

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

Answers

```
RESTORING DIVISION ALGORITHM
Enter two numbers to divide and both the numbers should be less than 16
Enter A: 8
Enter B: 4

Expected Quotient = 2
Expected Remainder = 0

Unsigned Binary Equivalents are:
A = 01000
B = 00100
B'+ 1 = 11100

-->
SHIFT LEFT: 00000:10000
-->
SUB B: 11100:10000
-->RESTORE
ADD B: 00000:10000
SHIFT LEFT: 00001:00000
-->
SUB B: 11101:00000
-->RESTORE
ADD B: 00001:00000
SHIFT LEFT: 00010:00000
-->
SUB B: 11110:00000
-->RESTORE
ADD B: 00010:00000
SHIFT LEFT: 00100:00000
-->
SUB B: 00000:00000
SHIFT LEFT: 00000:00010
-->
SUB B: 11100:00010
-->RESTORE
ADD B: 00000:00010

Sign of the result = 0
Remainder is = 00000
Quotient is = 00010
```

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

DLCOA Exp 6

Name: Keegan Vaz
Sec: SE CMPN B
Roll No: 242

Q2) Explain the steps of Restoring Division Algorithm.

→ 1. Load the divisor into the M register and the dividend into Q register and A register is 0.
2. At each step left shift A and Q one binary position.
3. Subtract M from A, and place the answer back in A.
4. If the sign of A is 1, set Q_0 to 0 and add M back to A (that is, restore A); otherwise, set Q_0 is 1
5. Repeat steps 2 through 4 as many times as there are bit positions in Q (Divided)
6. The remainder is in A and the quotient is in Q

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

Experiment No.: 7

Aim:-A program for Non-Restoring Division

Theory:

1. There are number of binary division algorithm. One of them is non- restoring division algorithm. Let Q register hold the dividend , M register holds the divisor and A register is 0
2. On completion of the algorithm, Q will get the quotient and A will get the remainder.

The flowchart for the algorithm is as follows:

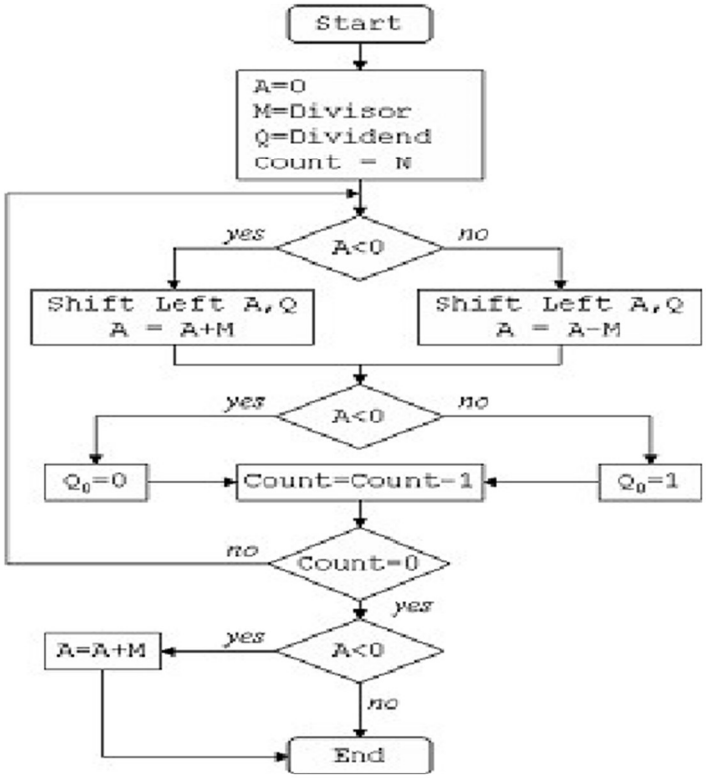


Fig.1 Flowchart for Non-Restoring Division

Conclusion :-

Viva Questions:-

1. Find 11/2,using Non-Restoring Division Algorithm.
2. Explain the steps of Non-Restoring Division Algorithm.

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

Code:

```
#include<stdio.h>
#include<conio.h>

int main()
{
  int a[10],m[10],q[10],i,j,c=0,b[10],mc[10],z=0,s[10],x[10];
  printf("Non-Restoring Division");
  printf("\nEnter Divisor [M] : ");
  for(i=4;i>=1;i--)
  {
    scanf("%d",&m[i]);
    mc[i]=!m[i];
    a[i]=0, x[i]=0;
  }
  x[1]=1, x[5]=0, m[5]=0, mc[5]=1;
  for(i=1;i<=5;i++)
  {
    s[i]=x[i]^mc[i]^z;
                        c=(x[i]&&mc[i])||(x[i]&&z)||(mc[i]&&z);
                        z=c;
                        mc[i]=s[i];
  }
  printf("\n\nEnter Divident [Q] : ");
  for(i=4;i>=1;i--)
```

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

```
{
                scanf("%d",&q[i]);
}
printf("\n\n\n Step \t\t Action Performed \t\t A \t\tQ\n");
printf("\n\n 0\t\t Initialization\t 0 0 0 0 0 \t ");
for(i=4;i>=1;i--)
{
                printf(" %d",q[i]);
}
for(j=1;j<=4;j++)
{
                printf("\n\n\n %d",j);
                printf("\t\t Left Shift \t\t ");
                for(i=5;i>=2;i--)
                {
                  a[i]=a[i-1];
                }
                a[1]=q[4];
                for(i=4;i>=2;i--)
                {
                  q[i]=q[i-1];
                }
                for(i=5;i>=1;i--)
                {
                  printf(" %d",a[i]);
```

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

```
                }
                printf("\t ");
                for(i=4;i>=2;i--)
                {
                  printf(" %d",q[i]);
}
                if(a[5]==0)
                {
                  z=0;
                  for(i=1;i<=5;i++)
                  {
                        s[i]=a[i]^mc[i]^z;
                        c=(a[i]&&mc[i])||(a[i]&&z)||(mc[i]&&z);
                        z=c;
                        a[i]=s[i];
                  }
                  if(a[5]==1)
 {
                        q[1]=0;
                  }
                  else
                  {
                        q[1]=1;
                  }
                  printf("\n\n\t\t a = a-m\t\t ");
```

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

```
  for(i=5;i>=1;i--)
  {
        printf(" %d",a[i]);
  }
  printf("\t ");
  for(i=4;i>=1;i--)
  {
        printf(" %d",q[i]);
  }
}
else
{
  z=0;
  for(i=1;i<=5;i++)
  {
        s[i]=a[i]^m[i]^z;
        c=(a[i]&&m[i])||(a[i]&&z)||(m[i]&&z);
        z=c;
        a[i]=s[i];
  }
  if(a[5]==1)
  {
        q[1]=0;
  }
  else
```

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

```
                    {
                        q[1]=1;
                    }
                    printf("\n\n\t\t a = a+m\t\t ");
                    for(i=5;i>=1;i--)
                    {
                        printf(" %d",a[i]);
                    }
                    printf("\t ");
                    for(i=4;i>=1;i--)
                    {
                        printf(" %d",q[i]);
   }
                }
}
if(a[5]==1)
{
                printf("\n\n\n 5");
                for(i=1;i<=5;i++)
                {
                  s[i]=a[i]^m[i]^z;
                  c=(a[i]&&m[i])||(a[i]&&z)||(m[i]&&z);
                  z=c;
                  a[i]=s[i];
                }
```

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

```
                printf("\t\t a = a+m\t\t ");
                for(i=5;i>=1;i--)
                {
                  printf(" %d",a[i]);
                }
                printf("\t ");
                for(i=4;i>=1;i--)
                {
                  printf(" %d",q[i]);
                }
  }
  printf("\n\n\n\n\nQuotient [Q] :");
  for(i=4;i>=1;i--)
  {
                printf(" %d",q[i]);
  }
  printf("\n\n\n\nRemainder [A] :");
  for(i=4;i>=1;i--)
  {
                printf(" %d",a[i]);
  }
  return 0;
}
```

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

Output:

```
Non-Restoring Division
Enter Divisor [M] : 1 0 1 1

Enter Divident [Q] : 1 0 0 1


 Step            Action Performed                A             Q

 0               Initialization  0 0 0 0 0       1 0 0 1

 1               Left Shift              0 0 0 0 1       0 0 1
                 a = a-m                 1 0 1 1 0       0 0 1 0

 2               Left Shift              0 1 1 0 0       0 1 0
                 a = a-m                 0 0 0 0 1       0 1 0 1

 3               Left Shift              0 0 0 1 0       1 0 1
                 a = a-m                 1 0 1 1 1       1 0 1 0

 4               Left Shift              0 1 1 1 1       0 1 0
                 a = a-m                 0 0 1 0 0       0 1 0 1


Quotient [Q] : 0 1 0 1


Remainder [A] : 0 1 0 0
```

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

Conclusion:

By this experiment we are able to study about non- restoring division where Q holds the dividend and M holds the divisor.

We are able to get the Quotient and remainder easily.

Answers:

```
Non-Restoring Division
Enter Divisor [M] : 0 0 1 0

Enter Divident [Q] : 1 0 1 1


 Step            Action Performed               A            Q

 0               Initialization  0 0 0 0 0       1 0 1 1

 1               Left Shift              0 0 0 0 1      0 1 1
                 a = a-m                 1 1 1 1 1      0 1 1 0

 2               Left Shift              1 1 1 1 0      1 1 0
                 a = a+m                 0 0 0 0 0      1 1 0 1

 3               Left Shift              0 0 0 0 1      1 0 1
                 a = a-m                 1 1 1 1 1      1 0 1 0

 4               Left Shift              1 1 1 1 1      0 1 0
                 a = a+m                 0 0 0 0 1      0 1 0 1


Quotient [Q] : 0 1 0 1


Remainder [A] : 0 0 0 1
```

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

DLCOA EXP 7

Name : Keegan Vaz
Sec: SE CMPN B
Roll No: 242

Q2) Explain the steps of Non-Restoring Division Algorithm

→ 1. At each steps left shift the divided by 1 position
2. Subtract the divisor from A (perform A-M)
3. If the result is positive then the step is said to be successful. In this case quotient will be 1 and no restoration is required. The next will also be subtraction.
4. If the result is negative then the step is said to be unsuccessful. In this case the quotient bit will be 0. But no restoration is performed. Instead next step will be addition in place of subtraction.
5. Repeat steps 1 to 4 for all bits of the dividend
6. If the last step is unsuccessful then restoring is performed

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

Experiment 8

Aim: Case Study on multi-core Processors.

Theory:

Multiple Instruction, Multiple Data (MIMD)

Multiple Instructions, Multiple Data (MIMD) refers to a parallel architecture, which is probably the most basic, but most familiar type of parallel processor. Its key objective is to achieve parallelism. MIMD architecture includes a set of N-individual, tightly-coupled processors. Each processor includes memory that can be common to all processors, and cannot be directly accessed by the other processors. MIMD architecture includes processors that operate independently and asynchronously. Various processors may be carrying out various instructions at any time on various pieces of data.

Points to be included:

- **Why multicore?**
- **Features**
- **Instruction level parallelism**
- **Thread level parallelism**
- **Memory types**
- **Applications**
- **Example**

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

DLCOA Exp 8

Name : Keegan Vaz
Sec : SE CMPN B
Roll No : 242

- Why Multicore?

→ The more CPU cores you have the more tasks you can run. Multicore processors also allow multiple databases to be consolidated onto a single server.

- Features

→ A multicore processor has two or more independent computing/processing units on the same chip. Multiple cores have the advantage that they run on lower frequency as compared to the single processing unit, which reduces the power dissipation or temperature

- Instruction level parallelism:

→ It allows the compiler and processor to overlap the execution of multiple instructions. It can change the order in which instructions are executed.

- Thread level parallelism:

→ A multicore processor incorporates multiple processors on the same chip. Every processor will have its own level 1 cache. Different processors execute independently thus allowing for embedded task- or thread-level parallelism

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

- Memory types

→ Closet to the processor is level 1 (L1) cache; this is very fast memory used to store data frequently used by the processor. Level 2 (L2) cache is just off-chip, slower than L1 cache, but still much faster than main memory; L2 cache is larger than L1 cache and used for the same purpose.

- Application

→ There are various fields where multi core processor are used such as VLIW, superscalar, vector or multithreading

- Example

→ 1. A-series, dual-, triple-, and quad-core of Accelerator Processor units (APU).

2. FX-series, quad-, 6-, 8-core desktop processor

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

Experiment No. 9

Aim: To simulate the working of Ripple Carry Adder.

Requirement: Virtual simulator

THEORY:

To design a ripple carry adder using full adders to mimic the behavior of the working module. The adder will add two 4 bit numbers

Arithmetic operations like addition, subtraction, multiplication, division are basic operations to be implemented in digital computers using basic gates like AND, OR, NOR, NAND etc. Among all the arithmetic operations if we can implement addition then it is easy to perform multiplication (by repeated addition), subtraction (by negating one operand) or division (repeated subtraction).

Half Adders can be used to add two one bit binary numbers. It is also possible to create a logical circuit using multiple full adders to add N-bit binary numbers.Each full adder inputs a **Cin**, which is the **Cout** of the previous adder. This kind of adder is a **Ripple Carry Adder**, since each carry bit "ripples" to the next full adder. The first (and only the first) full adder may be replaced by a half adder.The block diagram of 4-bit Ripple Carry Adder is shown here below –

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

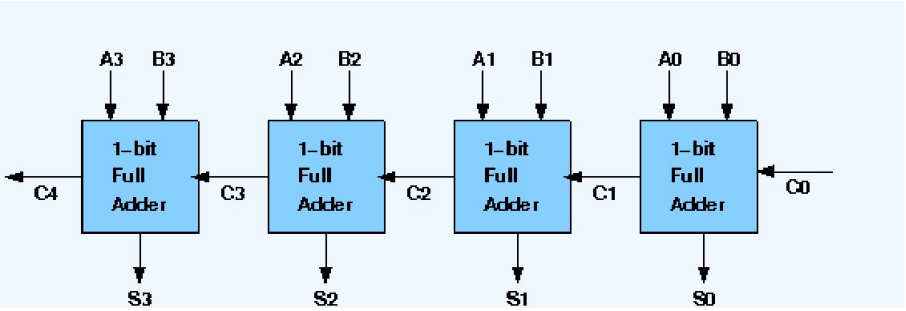


Fig.1 Block Diagram 4-bit Ripple Carry Adder

Implementation:

1. Simulate a half adder and full adder. Design a 4-bit ripple carry adder. Take snapshot.
2. Set one input to zero (0) and check the output. Take snapshot of 4 instances.
3. Set one input to all one and another as 0001 in ripple carry adder. Check the output.
4. Check the ripple carry adder with input carry with arbitrary

   inputs. Take snapshot of 2 instances.

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

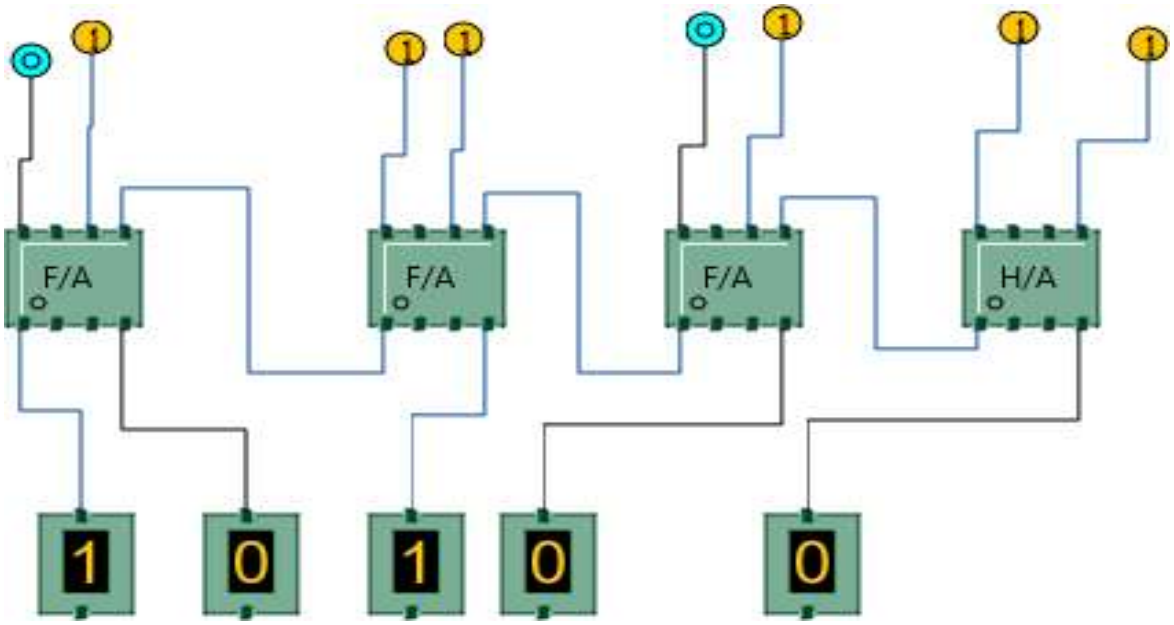


Fig. (2) 4 bit Ripple Carry Adder

ST. FRANCIS INSTITUTE OF TECHNOLOGY

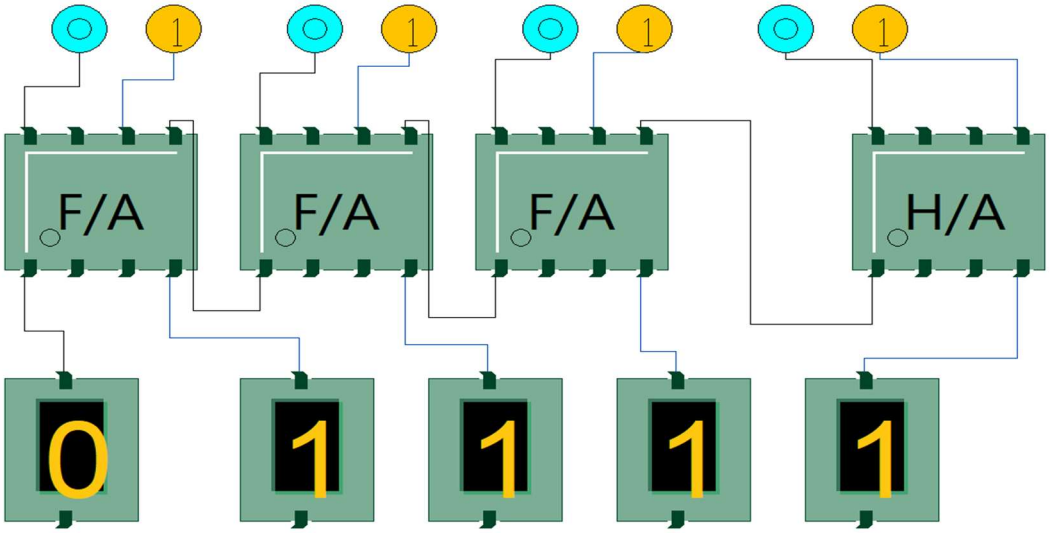
Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

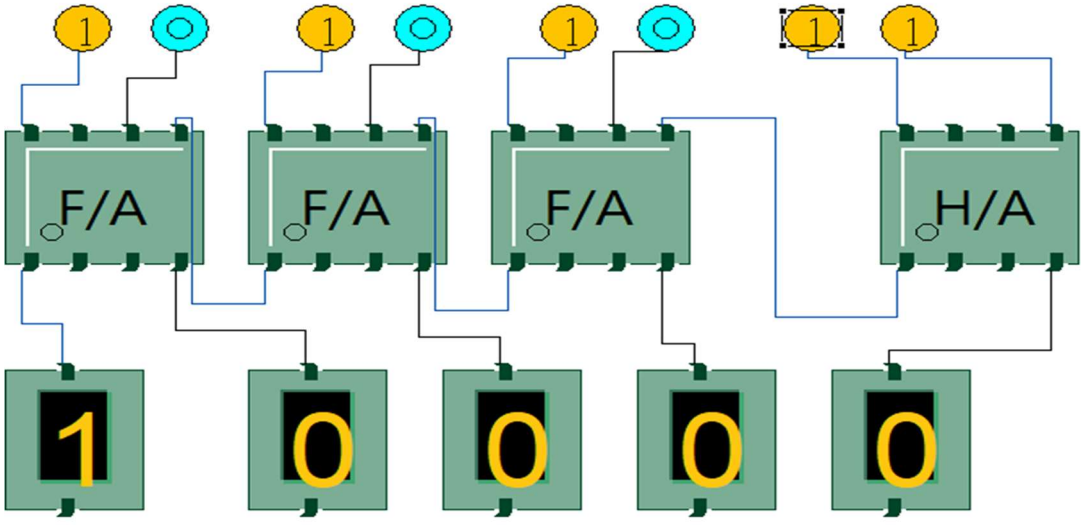
Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

1. **When one input is zero:**



2. **When one input is one and other format is in 0001:**



ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

Conclusion: The students will be able to understand the working of ripple carry adder.

Viva Questions:

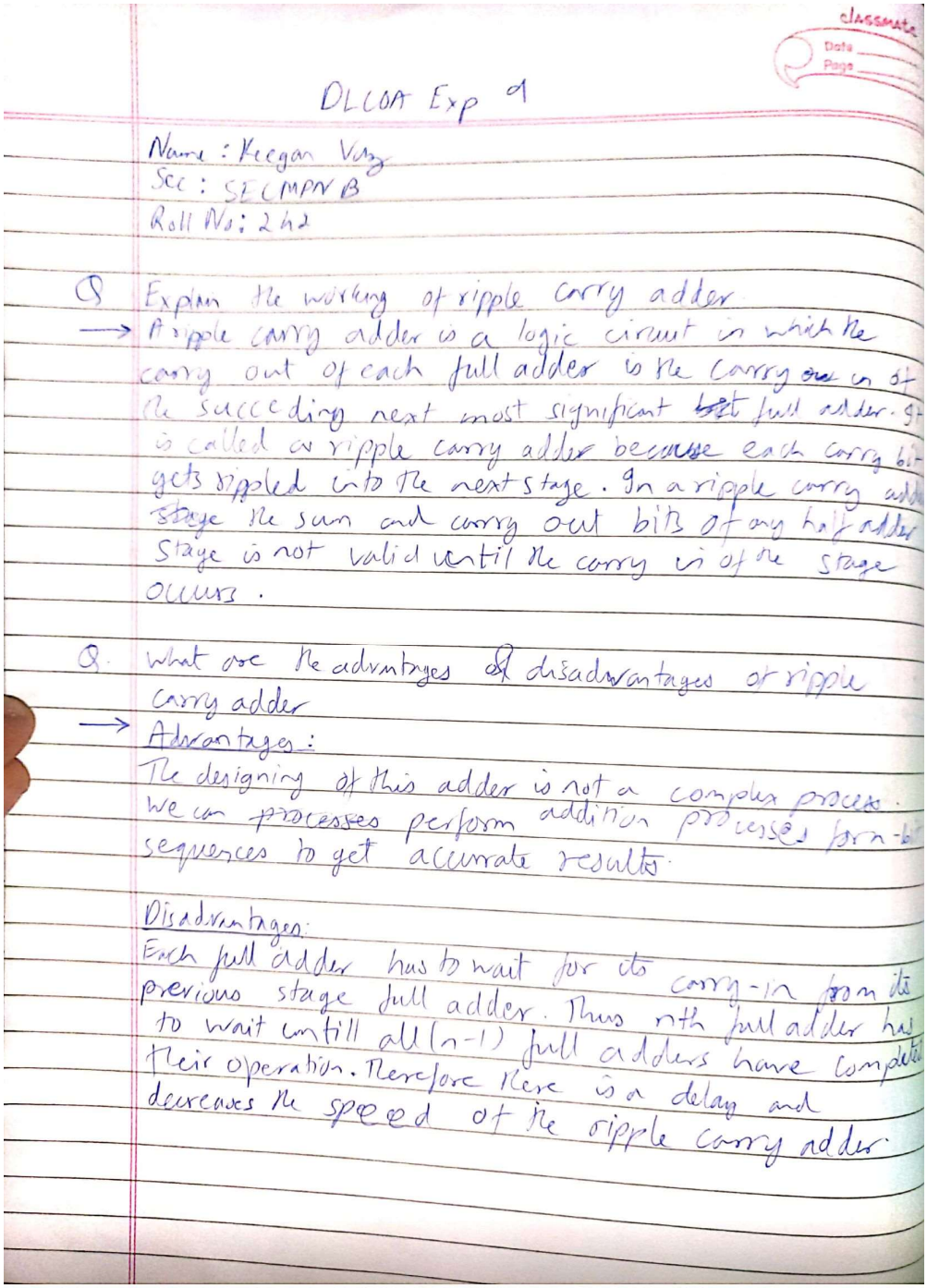
ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**



DLCOA Exp 01

Name : Keegan Vaz
Sec : SE CMPN B
Roll No: 242

Q Explain the working of ripple carry adder

→ A ripple carry adder is a logic circuit in which the carry out of each full adder is the carry in of the succeeding next most significant bit full adder. It is called as ripple carry adder because each carry bit gets rippled into the next stage. In a ripple carry adder stage the sum and carry out bits of any half adder stage is not valid until the carry in of the stage occurs.

Q. What are the advantages & disadvantages of ripple carry adder

→ Advantages:
The designing of this adder is not a complex process. We can processes perform addition processes for n-bit sequences to get accurate results.

Disadvantages:
Each full adder has to wait for its carry-in from its previous stage full adder. Thus nth full adder has to wait until all (n-1) full adders have completed their operation. Therefore there is a delay and decreases the speed of the ripple carry adder.

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

Experiment no. 10

Aim: Case Study on various buses like PCI , ISA , IDE etc.

THEORY:

For connecting different devices to a computer different uses are used. Each bus typically has a different data transfer speed.

Prepare a word document of two pages including bit of history, features and applications of following:

(Use diagrams)

ISA (Industry Standard Architecture) bus:

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

Industry Standard Architecture (ISA) is the 16-bit internal bus of IBM PC/AT and similar computers based on the Intel 80286 and its immediate successors during the 1980s. The bus was (largely) backward compatible with the 8-bit bus of the 8088-based IBM PC, including the IBM PC/XT as well as IBM PC compatibles. Originally referred to as the PC bus (8-bit) or AT bus (16-bit), it was also termed I/O Channel by IBM. It was introduced in the year 1981 by IBM(International Business Machines Corporation). It has 8 or 16 bits in width. It can be used in upto 6 devices and the speed is found to be 8 Mb/s. The style of ISA is parallel.

The PC/XT-bus is an eight-bit ISA bus used by Intel 8086 and Intel 8088 systems in the IBM PC and IBM PC XT in the 1980s. Among its 62 pins were demultiplexed and electrically buffered versions of the 8 data and 20 address lines of the 8088 processor, along with power lines, clocks, read/write strobes, interrupt lines, etc. Power lines included −5 V and ±12 V in order to directly support pMOS and enhancement mode nMOS circuits such as dynamic RAMs among other things. The XT bus architecture uses a single Intel 8259 PIC, giving eight vectorized and prioritized interrupt lines.

ST. FRANCIS INSTITUTE OF TECHNOLOGY

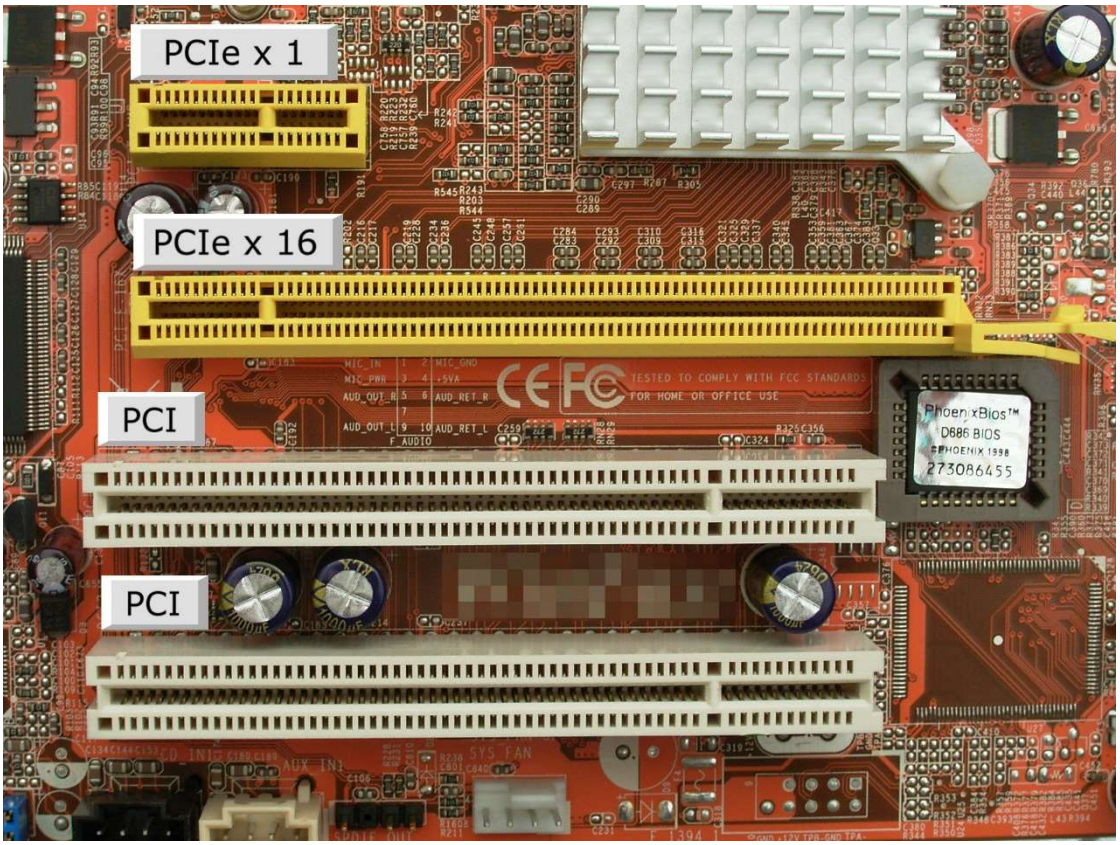
Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

PCI (Peripheral Component Interconnect) bus:



Peripheral Component Interconnect (PCI) is a local computer bus for attaching hardware devices in a computer and is part of the PCI Local Bus standard. The PCI bus supports the functions found on a processor bus but in a standardized format that is independent of any particular processor's native bus. Devices connected to the PCI bus appear to a bus master to be connected directly to its own bus and are assigned addresses in the processor's address space. It is a parallel bus, synchronous to a single bus clock. Attached devices can take either the form of an integrated circuit fitted onto the motherboard

itself (called a planar device in the PCI specification) or an expansion card that fits into a slot. The PCI Local Bus was first implemented in IBM PC compatibles, where it displaced the combination of several slow Industry Standard Architecture (ISA) slots and one fast VESA Local Bus slot as the bus configuration. It has subsequently been adopted for other computer types. Typical PCI cards used in PCs include: network cards, sound cards, modems, extra ports such as USB or serial, TV tuner cards and hard disk drive host adapters. PCI video cards replaced ISA and VESA cards until growing bandwidth requirements outgrew the capabilities of PCI. The preferred interface for video cards then became AGP, itself a superset of PCI, before giving way to PCI Express. Work on PCI began at Intel's Architecture Development Lab (IAL) c. 1990. PCI was immediately put to use in servers, replacing MCA and EISA as the server expansion bus of choice. In mainstream PCs, PCI was slower to replace VESA Local Bus (VLB), and did not gain significant market penetration until late 1994 in second-generation Pentium PCs. By 1996, VLB was all but extinct, and manufacturers had adopted PCI even for 486 computers. EISA continued to be used alongside PCI through 2000. Apple Computer adopted PCI for professional Power Macintosh computers (replacing NuBus) in mid-1995, and the consumer Performa product line (replacing LC PDS) in mid-1996. The 64-bit version of plain PCI remained rare in practice though, although it was used for example by all (post-iMac) G3 and G4 Power Macintosh computers.

SCSI (Small Computer System Interface) bus:

Small Computer System Interface (SCSI) is a set of standards for physically connecting and transferring data between computers and peripheral devices. The SCSI standards define commands, protocols, electrical, optical and logical interfaces. SCSI is most commonly used for hard disk drives and tape drives, but it can connect a wide range of other devices, including scanners and CD drives, although not all controllers can handle all devices. The SCSI standard defines command sets for specific peripheral device types; the presence of "unknown" as one of these types means that in theory it can be used as an interface to almost any device, but the standard is highly pragmatic and addressed toward commercial requirements.

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

SCSI is derived from "SASI", the "Shugart Associates System Interface", developed circa 1978 and publicly disclosed in 1981. Larry Boucher is considered to be the "father" of SASI and ultimately SCSI due to his pioneering work first at Shugart Associates and then at Adaptec. A SASI controller provided a bridge between a hard disk drive's low-level interface and a host computer, which needed to read blocks of data. Recent physical versions of SCSI—Serial Attached SCSI (SAS), SCSI-over-Fibre Channel Protocol (FCP), and USB Attached SCSI (UAS)—break from the traditional parallel SCSI bus and perform data transfer via serial communications using point-to-point links. Although much of the SCSI documentation talks about the parallel interface, all modern development efforts use serial interfaces. Serial interfaces have a number of advantages over parallel SCSI, including higher data rates, simplified cabling, longer reach, improved fault isolation and full-duplex capability. The primary reason for the shift to serial interfaces is the clock skew issue of high-speed parallel interfaces, which makes the faster variants of parallel SCSI susceptible to problems caused by cabling and termination.

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

IDE (Integrated Drive Electronics) bus:

Integrated Drive Electronics (IDE) is a standard interface for connecting a motherboard to storage devices such as hard drives and CD-ROM/DVD drives. The original IDE had a 16-bit interface that connected two devices to a single-ribbon cable. This cost-effective IDE device carried its own circuitry and included an integrated disk drive controller. Prior to IDE, controllers were separate external devices. IDE's development increased data transfer rate (DTR) speed and reduced storage device and controller issues. It is also known as Advanced Technology Attachment (ATA) or intelligent drive electronics (IDE).

The IDE interface contains two IDE device connections and two motherboard connectors for two data cables. An IDE-integrated controller sends an array of 512-byte blocks between the drive and motherboard, which houses up to four chipset-controlled IDE devices within one system. Most personal computers (PC) contain hard drive and CD-ROM connections. The hard drive uses one cable and connects to the motherboard via the primary IDE connector. The CD-ROM drive and other storage devices

ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (West), Mumbai - 400103

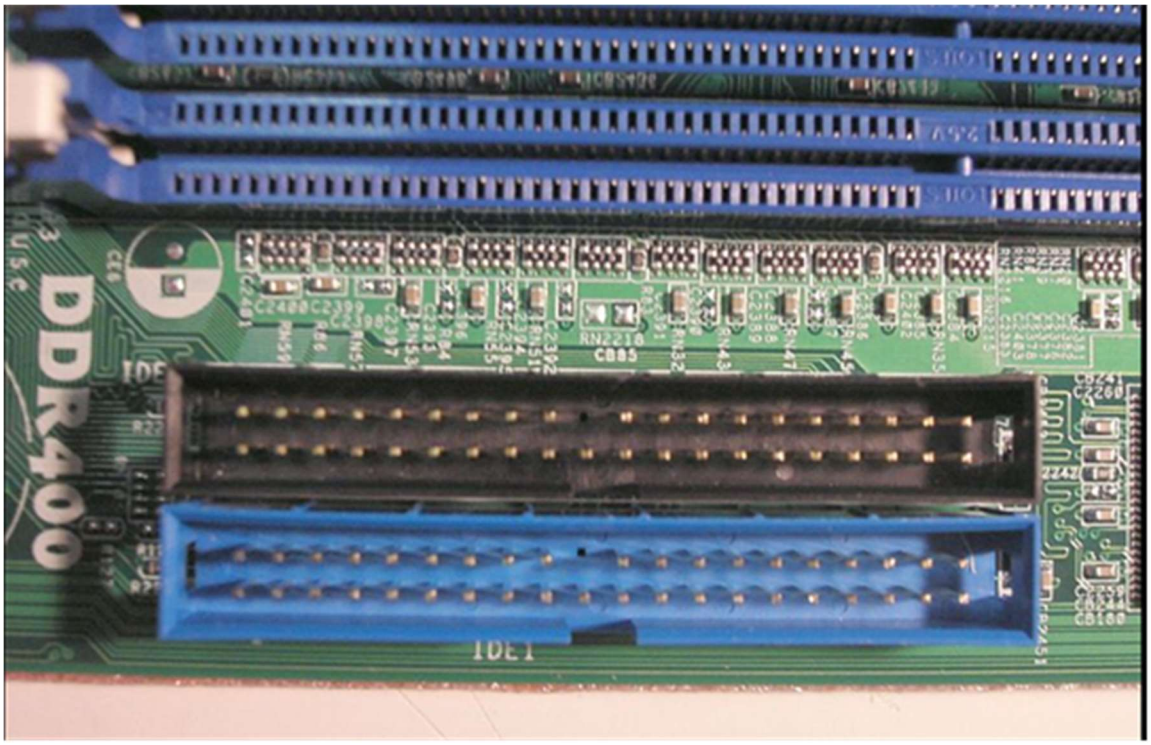
Computer Engineering Department

Name: Keegan Vaz **Sec: SE CMPN B**

Roll No: 242 **PID: 192120**

share one IDE cable. The standard IDE (ATA/ATAPI) uses two different connectors. The IDE/ATA cable attaches to the data connector, and the standardized power connector provides power. IDE reduced problems associated with storage devices and integrated controllers. Prior to IDE, controllers were separate external devices. Several hardware manufacturers, including Compaq Computer Corporation (which was sold to Hewlett-Packard), and Control Data Corporation (CDC), industrialized the ST-506 hard drive controller and signalling protocols.

In today's computers, the IDE controller is often built into the motherboard. Prior to the IDE drive, controllers were separate external devices so IDE reduced problems associated with storage devices and integrated controllers.



Indicating the primary IDE in blue and the secondary IDE in black which is connected in the motherboard.