

LABORATORY MANUAL
for
DIGITAL CIRCUITS (ECE181313)

B.Tech 3rd Semester
Assam Science and Technology University



Prepared by: Niranjana Jyoti Borah
Asst. Prof., E&TE Dept.

Smita Sarma
Asst. Prof., E&TE Dept.

DEPT. OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING
ASSAM ENGINEERING COLLEGE
JALUKBARI-781013

Course Outcome

At the end of the course, the students will be able to

CO1:

Simplify, design and implement Boolean expression/half and full adders using basic/universal gates.

CO2:

Design and implement the various combinational circuits using MSI components.

CO3:

Design and implement the various sequential circuits

CO4:

Analyze the results, and prepare a formal laboratory report.

Table of Contents

Experiment no.	Objective	Page no.
1	To familiarize with logic gate IC packages and to verify the truth tables of logic gates	1-5
2	To verify De-morgan's theorem for 2 variables.	6-7
3	To design and set up Half Adder and Half Subtractor using a. EXOR gates and gates b. NAND gates	8-10
4	To design and set up Full Adder and Full Subtractor using a. EXOR gates and gates b. NAND gates	11-13
5	Design and implementation of parallel adder/ Subtractor using IC7483	14
6	Design and implementation of a. BCD-to- excess-3code converter and vice versa. b. Gray-to- binary and vice-versa.	15-17
7	Design and implementation of one bit, two-bit magnitude comparators.	18-20
8	Design and set up a Multiplexer (MUX) using gates and ICs.	21-22
9	Implementation and verification of truth table for J-K flip-flop, Master-slave J-K flip-flop, D flip-flop and T flip-flop.	23-26
10	Design and implementation of Mod-N synchronous counter using J-K flip-flops.	27-29
11	Design and implementation of shift register to function as a) SISO b) SIPO c) PISO d) PIPO e) shift left and f) shift right operation	30-32
12	Design and implementation of a) Ring counter and b) Johnson counter	33-34

EXPERIMENT NO. 1

AIM: To familiarize with logic gate IC packages and to verify the truth tables of logic gates

HARDWARE REQUIRED: IC (NOT-7404, OR-7432, AND-7408, NOR-7402, NAND-7400, EXOR-7486, EXNOR-74266), Multimeter, Breadboard, Wires, LED, Resistor, Power source.

THEORY:

The **Digital Logic Gate** is the basic building block from which all digital electronic circuits and microprocessor based systems are constructed. Basic digital logic gates perform logical operations of AND, OR and NOT on binary numbers.

In digital logic design only two voltage levels or states are allowed and these states are generally referred to as Logic “1” and Logic “0”, or HIGH and LOW, or TRUE and FALSE. These two states are represented in Boolean Algebra and standard truth tables by the binary digits of “1” and “0” respectively.

A good example of a digital state is a simple light **switch**. The switch can be either “ON” or “OFF”, one state or the other, but not both at the same time.

Digital logic gates can have more than one input, for example, inputs A, B, C, D etc., but generally only have one digital output, (Q). Individual logic gates can be connected or cascaded together to form a logic gate function with any desired number of inputs, or to form combinational and sequential type circuits, or to produce different logic gate functions from standard gates.

Standard commercially available digital logic gates are

- DTL, Diode-Transistor logic gates
- RTL, Resistor-Transistor logic gates
- ECL, Emitter-Coupled logic gates
- TTL which stands for *Transistor-Transistor Logic* such as the 7400 series
- MOS, Metal oxide semiconductor
- CMOS which stands for *Complementary Metal-Oxide-Semiconductor* which is the 4000 series of chips.

This notation of TTL or CMOS refers to the logic technology used to manufacture the integrated circuit, (IC) or a “chip” as it is more commonly called.

TTL and ECL are based upon bipolar transistors. TTL has a well established popularity among logic families. ECL is used only in systems requiring high-speed operation. MOS and CMOS, are based on field effect transistors. They are widely used in large scale integrated circuits because of their high component density and relatively low power consumption. CMOS logic consumes far less power than MOS logic.

Integrated Circuits or IC's as they are more commonly called, can be grouped together into families according to the number of transistors or "gates" that they contain.

- Small Scale Integration or (SSI)
- Medium Scale Integration or (MSI)
- Large Scale Integration or (LSI)
- Very-Large Scale Integration or (VLSI)
- Super-Large Scale Integration or (SLSI)
- Ultra-Large Scale Integration or (ULSI).

Another level of integration which represents the complexity of the Integrated Circuit is known as the *System-on-Chip* or (SOC) for short. Here the individual components such as the microprocessor, memory, peripherals, I/O logic etc, are all produced on a single piece of silicon and which represents a whole electronic system within one single chip, literally putting the word "integrated" into integrated circuit.

TTL (Transistor-Transistor Logic)

In standard TTL (transistor-transistor logic) IC's there is a pre-defined voltage range for the input and output voltage levels which define exactly what is a logic "1" level and what is a logic "0" level and these are shown below.

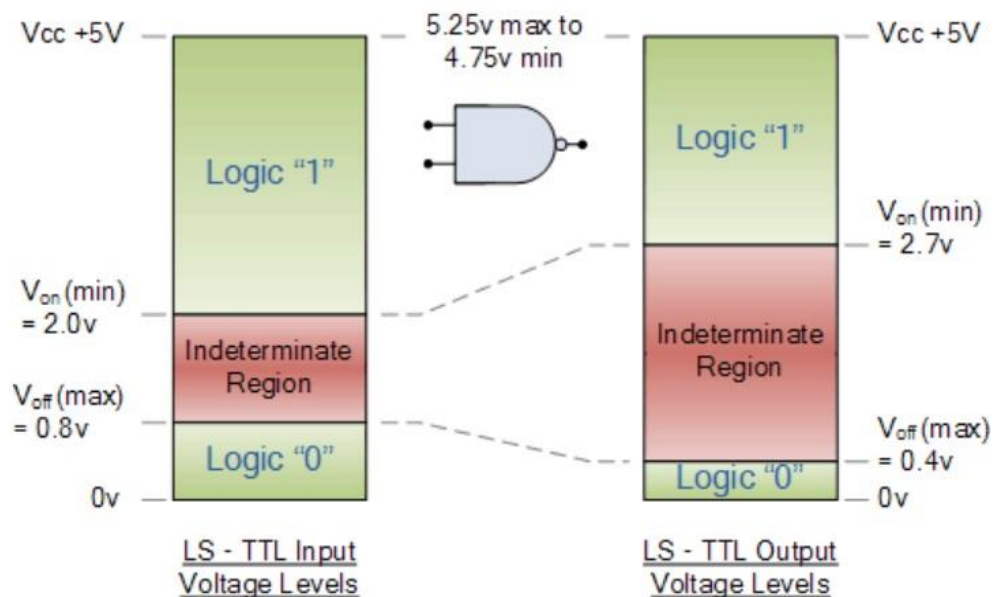


Fig 1.1: TTL input and output voltage levels

There are a large variety of logic gate types in the bipolar 7400 family of digital logic gates such as 74Lxx, 74LSxx, 74ALSxx, 74HCxx, 74HCTxx, 74ACTxx etc, with each one having its own distinct advantages and disadvantages compared to the other. The exact switching voltage

required to produce either a logic “0” or a logic “1” depends upon the specific logic group or family.

However, when using a standard +5 volt supply any TTL voltage input between 2.0v and 5v is considered to be a logic “1” or “HIGH” while any voltage input below 0.8v is recognised as a logic “0” or “LOW”. The voltage region in between these two voltage levels either as an input or as an output is called the *Indeterminate Region* and operating within this region may cause the logic gate to produce a false output.

The “74” Sub-families of Integrated Circuits


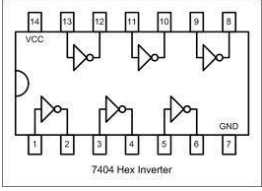
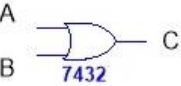
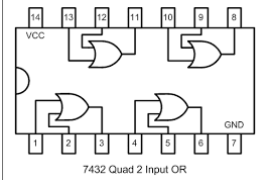
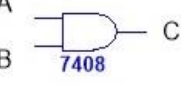
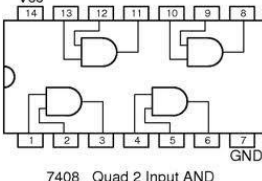
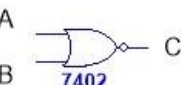
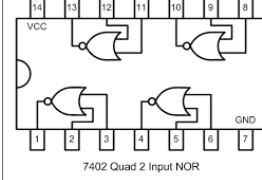
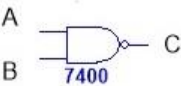
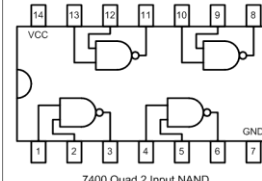
With improvements in the circuit design to take account of propagation delays, current consumption, fan-in and fan-out requirements etc, this type of TTL bipolar transistor technology forms the basis of the prefixed “74” family of digital logic IC’s, such as the “7400” Quad 2-input NAND gate, or the “7402” Quad 2-input NOR gate, etc.


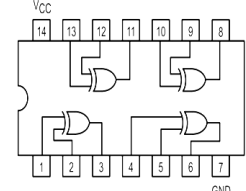

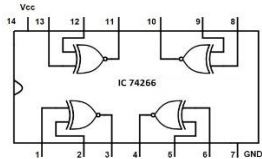
Sub-families of the 74xxx series IC’s are available relating to the different technologies used to fabricate the gates and they are denoted by the letters in between the 74 designation and the device number. There are a number of TTL sub-families available that provide a wide range of switching speeds and power consumption such as the 74L00 or 74ALS00 NAND gate, where the “L” stands for “Low-power TTL” and the “ALS” stands for “Advanced Low-power Schottky TTL” and these are listed below.

- 74xx or 74Nxx: Standard TTL – These devices are the original TTL family of logic gates introduced in the early 70’s. They have a propagation delay of about 10ns and a power consumption of about 10mW. Supply voltage range: 4.75 to 5.25 volts.
- 74Lxx: Low Power TTL – Power consumption was improved over standard types by increasing the number of internal resistances but at the cost of a reduction in switching speed. Supply voltage range: 4.75 to 5.25 volts.
- 74Hxx: High Speed TTL – Switching speed was improved by reducing the number of internal resistances. This also increased the power consumption. Supply voltage range: 4.75 to 5.25 volts.
- 74Sxx: Schottky TTL – Schottky technology is used to improve input impedance, switching speed and power consumption (2mW) compared to the 74Lxx and 74Hxx types. Supply voltage range: 4.75 to 5.25 volts.
- 74LSxx: Low Power Schottky TTL – Same as 74Sxx types but with increased internal resistances to improve power consumption. Supply voltage range: 4.75 to 5.25 volts.
- 74ASxx: Advanced Schottky TTL – Improved design over 74Sxx Schottky types optimised to increase switching speed at the expense of power consumption of about 22mW. Supply voltage range: 4.5 to 5.5 volts.

- 74ALSxx: Advanced Low Power Schottky TTL – Lower power consumption of about 1mW and higher switching speed of 4nS compared to 74LSxx types. Supply voltage range: 4.5 to 5.5 volts.
- 74HCxx: High Speed CMOS – CMOS technology and transistors to reduce power consumption of less than 1uA with CMOS compatible inputs. Supply voltage range: 4.5 to 5.5 volts.
- 74HCTxx: High Speed CMOS – CMOS technology and transistors to reduce power consumption of less than 1uA but has increased propagation delay of about 16nS due to the TTL compatible inputs. Supply voltage range: 4.5 to 5.5 volts.

Table 1.1: Logic Gates and their Properties

Gate	Description	Truth Table			Logic Symbol	Pin Diagram
		I/P's		O/P		
		A	B	C		
NOT	In this gate the output is opposite to the input state. Mathematically, $C = \bar{A}$	0	-	1		
		1	-	0		
OR	The output is active high only if both the inputs are in active high state. Mathematically, $C = A + B$	0	0	0		
		0	1	1		
		1	0	1		
		1	1	1		
AND	The output is active high only if both the inputs are in active high state. Mathematically, $C = A.B$	0	0	0		
		0	1	0		
		1	0	0		
		1	1	1		
NOR	The output is active high only if both the inputs are in active low state. Mathematically, $C = \overline{(A + B)}$	0	0	1		
		0	1	0		
		1	0	0		
		1	1	0		
NAND	The output is active high only if any one of the input is in active low state. Mathematically, $C = \overline{(A.B)}$	0	0	1		
		0	1	1		
		1	0	1		
		1	1	0		

EXOR	The output is active high only if any one of the input is in active high state. Mathematically, $C = \bar{A} B + A \bar{B}$	0	0	0		
		0	1	1		
		1	0	1		
		1	1	0		
EXNOR	The output is active high only if both the inputs are alike. Mathematically, $C = \bar{A} \bar{B} + A B$	0	0	1		
		0	1	0		
		1	0	0		
		1	1	1		

Result: Truth Table for logic Gates are verified.

EXPERIMENT NO. 2

AIM: To verify De-morgan's theorem for 2 variables.

HARDWARE REQUIRED: IC (NOT-7404, OR-7432, AND-7408), Multimeter, Breadboard, Wires, LED, Resistor, Power source.

THEORY:

DeMorgan's Theorem and Laws can be used to find the equivalency of the NAND and NOR gates. While George Boole's set of laws and rules allows us to analyse and simplify a digital circuit, there are two laws within his set that are attributed to **Augustus DeMorgan** (a nineteenth century English mathematician) which views the logical NAND and NOR operations as separate NOT AND and NOT OR functions respectively.

DeMorgan's Theory

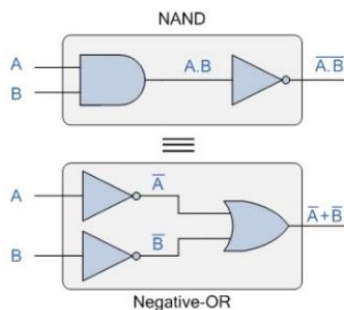
DeMorgan's Theorems are basically two sets of rules or laws developed from the Boolean expressions for AND, OR and NOT using two input variables, A and B. These two rules or theorems allow the input variables to be negated and converted from one form of a Boolean function into an opposite form.

DeMorgan's First Theorem

DeMorgan's First theorem proves that when two (or more) input variables are AND'ed and negated, they are equivalent to the OR of the complements of the individual variables. Thus the equivalent of the NAND function and is a negative-OR function proving that $A \cdot B = \overline{\overline{A} + \overline{B}}$ and we can show this using the following table and logic gates as depicted below.

Table2.1: Verifying DeMorgan's First Theorem using Truth Table

Inputs		Truth Table Outputs for Each Term				
A	B	$A \cdot B$	$\overline{A \cdot B}$	\overline{A}	\overline{B}	$\overline{A} + \overline{B}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0



Thus an OR gate with inverters (NOT gates) on each of its inputs is equivalent to a NAND gate function, and an individual NAND gate can be represented in this way as the equivalency of a NAND gate is a negative-OR.

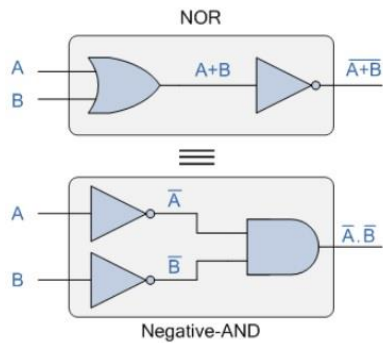
Fig 2.1: De Morgan's First Law Implementation using Logic Gates

✚ DeMorgan's Second Theorem

DeMorgan's Second theorem proves that when two (or more) input variables are OR'ed and negated, they are equivalent to the AND of the complements of the individual variables. Thus the equivalent of the NOR function and is a negative-AND function proving that $A+B = \overline{\overline{A} \cdot \overline{B}}$ and again we can show this using the following truth table and logic gates as depicted below.

Table2.2: Verifying DeMorgan's Second Theorem using Truth Table

Inputs		Truth Table Outputs for Each Term				
A	B	$A+B$	$\overline{A+B}$	\overline{A}	\overline{B}	$\overline{A} \cdot \overline{B}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0



Thus an AND gate with inverters (NOT gates) on each of its inputs is equivalent to a NOR gate function, and an individual NOR gate can be represented in this way as the equivalency of a NOR gate is a negative-AND.

Fig 2.2: De Morgan's Second Law Implementation using Logic Gates

Result: Hence De-morgan's theorem for 2 variables is verified.

EXPERIMENT NO. 3

AIM: To design and set up Half Adder and Half Subtractor using

- a. EXOR gates and gates b. NAND gates

HARDWARE REQUIRED: IC (NOT-7404, AND-7408, NAND-7400, EXOR-7486), Multimeter, Breadboard, Wires, LED, Resistor, Power source.

THEORY:

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.

If A and B are binary inputs to the half adder, then the logic function to calculate sum S is Ex – OR of A and B and logic function to calculate carry C is AND of A and B. Combining these two, the logical circuit to implement the combinational circuit of Half Adder is shown below.

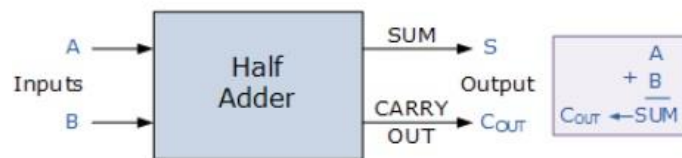


Fig 3.1: Block diagram of HA

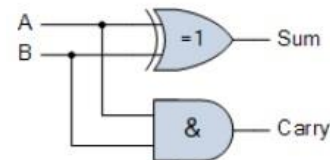


Fig 3.2: Logic diagram of HA

Table 3.1: Truth Table of HA

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

From the truth table of the half adder we can see that the SUM (S) output is the result of the Exclusive-OR gate and the Carry-out (Cout) is the result of the AND gate. Then the Boolean expression for a half adder is as follows.

For the **SUM** bit:

$$\text{SUM} = A \text{ XOR } B = A \oplus B$$

For the **CARRY** bit:

$$\text{CARRY} = A \text{ AND } B = A.B$$

As we know that NAND and NOR are called universal gates as any logic system can be implemented using these two, the half adder circuit can also be implemented using them as shown below.

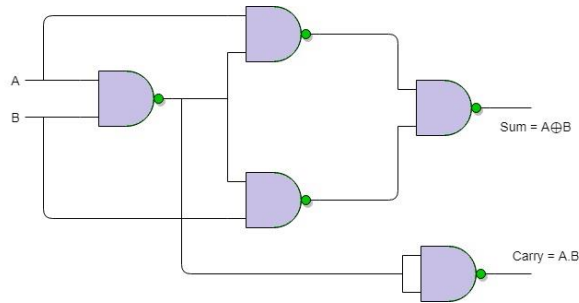


Fig 3.3: Implementation of HA using NAND Gates

Half Subtractor

A half subtractor is a logical circuit that performs a subtraction operation on two binary digits. The half subtractor produces a sum and a borrow bit for the next stage.

If X and Y are binary inputs to the half subtractor, then the logic function to calculate difference D is $X \oplus Y$ – OR of X and Y and logic function to calculate borrow B_{OUT} is AND of \bar{X} and Y. Combining these two, the logical circuit to implement the combinational circuit of Half Subtractor is shown below.

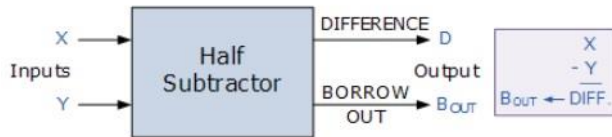


Fig 3.4: Block diagram of HS

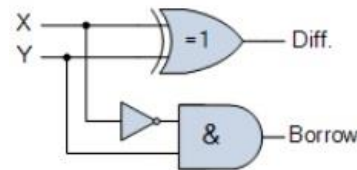


Fig 3.5: Logic diagram of HS

Table 3.2: Truth Table of HS

X	Y	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

From the truth table of the half subtractor we can see that the DIFFERENCE (D) output is the result of the Exclusive-OR gate and the Borrow-out (Bout) is the result of the NOT-AND combination. Then the Boolean expression for a half subtractor is as follows.

For the **DIFFERENCE** bit:

$$D = X \text{ XOR } Y = X \oplus Y$$

For the **BORROW** bit:

$$B = \text{not-}X \text{ AND } Y = \bar{X}.Y$$

The half subtractor circuit can also be implemented using NAND gate as shown below.

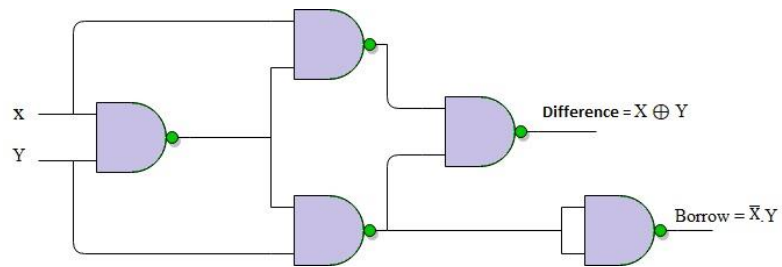


Fig 3.6: Implementation of HA using NAND Gates

Result: Hence HA and HS circuit is designed and studied.

EXPERIMENT NO. 4

AIM: To design and set up Full Adder and Full Subtractor using

- a. EXOR gates and gates
- b. NAND gates

HARDWARE REQUIRED: IC (NOT-7404, AND-7408, NAND-7400, EXOR-7486, OR-7432), Multimeter, Breadboard, Wires, LED, Resistor, Power source.

THEORY:

Full adder

Full adder is a digital circuit used to calculate the sum of three binary bits which is the main difference between the Full adder and Half adder. The same two single bit data inputs A and B as before plus an additional *Carry-in* (C_{in}) input to receive the carry from a previous stage as shown below.

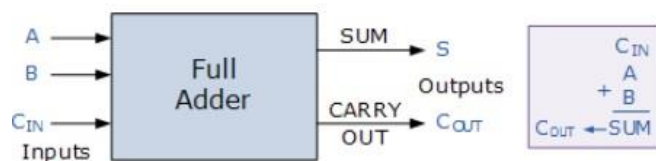


Fig 4.1: Block diagram of FA

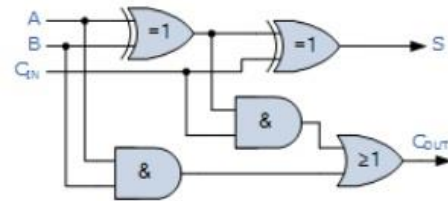


Fig 4.2: Logic diagram of FA

Then the **full adder** is a logical circuit that performs an addition operation on three binary digits and just like the half adder, it also generates a carry out to the next addition column. Then a *Carry-in* is a possible carry from a less significant digit, while a *Carry-out* represents a carry to a more significant digit.

In many ways, the full adder can be thought of as two half adders connected together, with the first half adder passing its carry to the second half adder as shown.

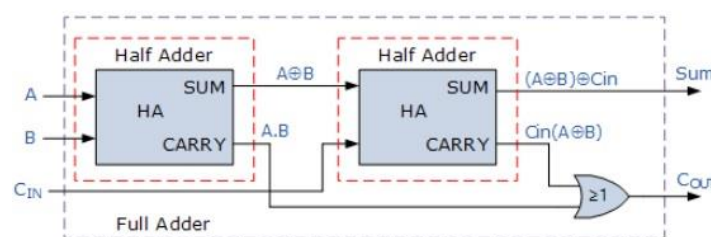


Fig 4.3: FA as a combination of two HA and OR gate

As the full adder circuit is basically two half adders connected together, the truth table for the full adder includes an additional column to take into account the *Carry-in*, C_{in} input as well as the summed output, S and the Carry-out, C_{out} bit.

Table 4.1: Truth Table of FA

C _{in}	A	B	Sum	C _{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

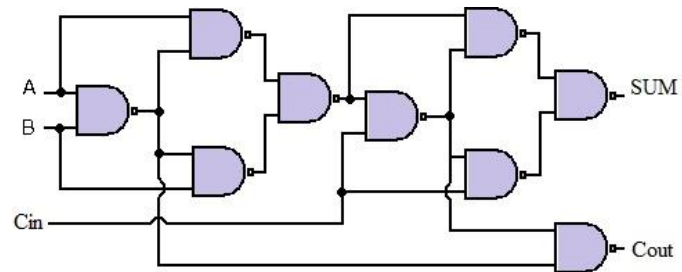


Fig 4.4: FA using NAND

Then the Boolean expression for a full adder is as follows.

For the **SUM** (S) bit:

$$\text{SUM} = (A \text{ XOR } B) \text{ XOR } C_{in} = (A \oplus B) \oplus C_{in}$$

For the **CARRY-OUT** (C_{out}) bit:

$$\text{CARRY-OUT} = A \text{ AND } B \text{ OR } C_{in}(A \text{ XOR } B) = A.B + C_{in}(A \oplus B)$$

Full Subtractor

The main difference between the Full Subtractor and the previous Half Subtractor circuit is that a full subtractor has three inputs. The two single bit data inputs X (minuend) and Y (subtrahend) the same as before plus an additional *Borrow-in* (B-in) input to receive the borrow generated by the subtraction process from a previous stage as shown below.

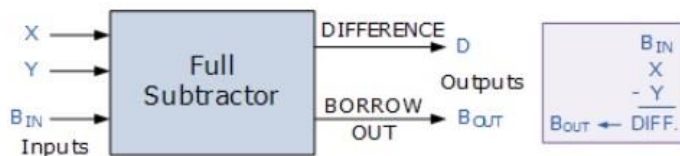


Fig 4.5: Block diagram of FS

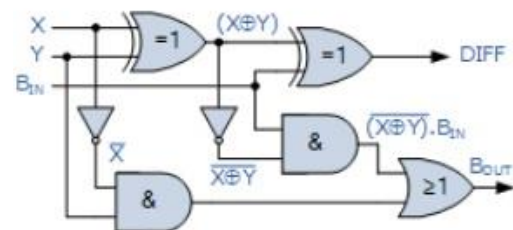
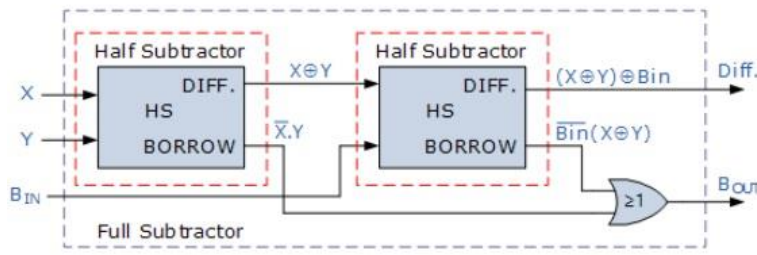


Fig 4.6: Logic diagram of FS

Then the combinational circuit of a “full subtractor” performs the operation of subtraction on three binary bits producing outputs for the difference D and borrow B-out. Just like the binary

adder circuit, the full subtractor can also be thought of as two half subtractors connected together, with the first half subtractor passing its borrow to the second half subtractor as follows.



As the full subtractor circuit represents two half subtractors cascaded together, the truth table for the full subtractor will have, the data bits, the *Borrow-in*, B_{IN} input, the difference output, D and the Borrow-out, B_{OUT} bit.

Fig 4.7: FS as a combination of two HS and OR gate

Table 4.2: Truth Table of FS

B_{in}	X	Y	Diff.	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

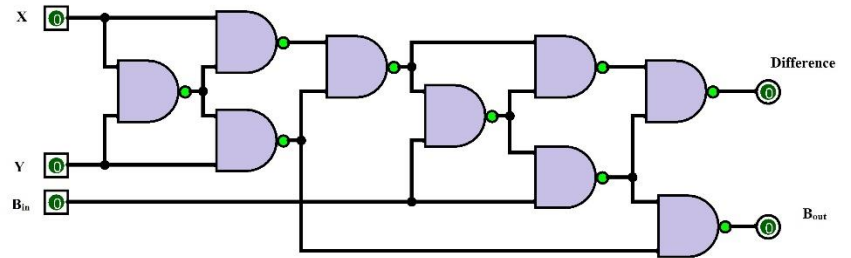


Fig 4.8: FS using NAND

Then the Boolean expression for a full subtractor is as follows.

For the **DIFFERENCE** (D) bit:

$$D = (X.Y.B_{IN}) + (X.Y.B_{IN}) + (X.Y.B_{IN}) + (X.Y.B_{IN})$$

which can be simplified too:

$$D = (X \text{ XOR } Y) \text{ XOR } B_{IN} = (X \oplus Y) \oplus B_{IN}$$

For the **BORROW OUT** (B_{OUT}) bit:

$$B_{OUT} = (X.Y.B_{IN}) + (X.Y.B_{IN}) + (X.Y.B_{IN}) + (X.Y.B_{IN})$$

which will also simplify too:

$$B_{OUT} = X \text{ AND } Y \text{ OR } (X \text{ XOR } Y)B_{IN} = X.Y + (X \oplus Y)B_{IN}$$

Result: Hence FA and FS circuit is designed and studied.

EXPERIMENT NO. 5

AIM: Design and implementation of parallel adder/ Subtractor using IC7483

HARDWARE REQUIRED: IC (7483, EXOR-7486), Multimeter, Breadboard, Wires, LED, Resistor, Power source.

THEORY:

IC type 7483 is a four-bit parallel adder. The pin assignment is shown in Fig 5.1. The 2 four-bit input binary numbers are A1 through A4 and B1 through B4. The four bit sum is obtained from S1 through S4. C0 is the input carry and C4 the output carry. This IC can be used as an adder-subtractor and as a magnitude comparator.

For adding the two numbers $A_4A_3A_2A_1$ and $B_4B_3B_2B_1$, the two numbers are given to input terminals 1,3,8,10 and 16,4,7,11 of the IC 7483 and carry in the terminal 13 is set to zero. To subtract two numbers by two's complement method, 2's complement of the subtrahend is added to the minuend. The 2's complement can be obtained by taking the 1's complement and adding 1. The final carry is neglected and the difference is taken from $S_4S_3S_2S_1$. So to perform $A - B$, we complement the four bits of B, add them to the four bits of A and add 1 through the input carry. This is done as shown in Fig 5.2.

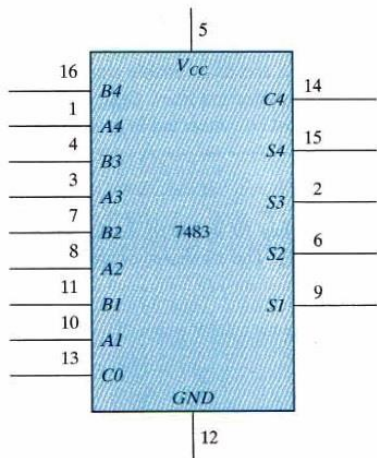


Fig 5.1: IC type 7483 four-bit binary adder

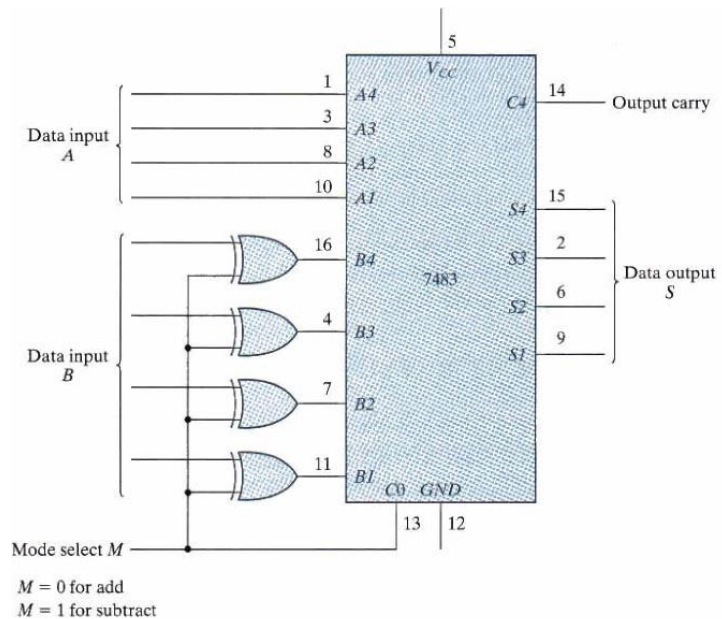


Fig 5.2: Four bit adder-subtractor

In the circuit we set mode control such that when the mode control is zero, addition is performed and subtraction is performed when the mode control is one. We use XOR gates to feed the input so that when mode control is one, the complement of each of the four bits are fed and when mode control is zero, the input as such is fed.

Result: Hence parallel adder/ Subtractor using IC7483 is implemented.

EXPERIMENT NO. 6

AIM: Design and implementation of

- BCD-to- excess-3code converter and vice versa.
- Gray-to- binary and vice-versa.

HARDWARE REQUIRED: IC (7483, EXOR-7486), Multimeter, Breadboard, Wires, LED, Resistor, Power source.

THEORY:

The availability of large variety of codes for the same discrete elements of information results in the use of different codes by different systems. A conversion circuit must be inserted between the two systems if each uses different codes for same information. Thus, code converter is a circuit that makes the two systems compatible even though each uses different binary code.

BCD-to- excess-3code converter and vice versa

Code converter is a combinational circuit that translates the input code word into a new corresponding word. The excess-3 code digit is obtained by adding three to the corresponding BCD digit. To Construct a BCD-to-excess-3-code converter with a 4-bit adder feed BCD- code to the 4-bit adder as the first operand and then feed constant 3 as the second operand. The output is the corresponding excess-3 code.

To make it work as a excess-3 to BCD converter, we feed excess-3 code as the first operand and then feed 2's complement of 3 as the second operand. The output is the BCD code.

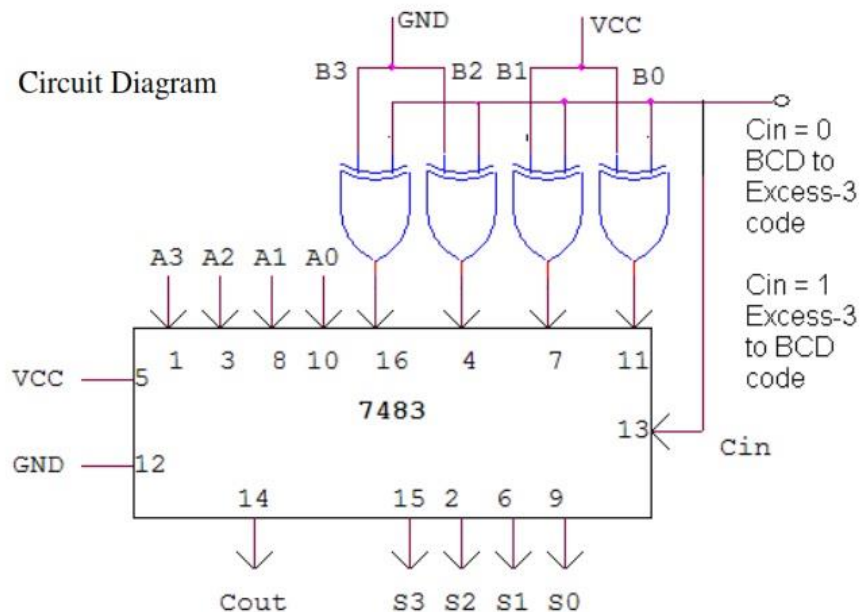


Fig 6.1: BCD to excess3 converter and vice-versa

Table 6.1: Truth table of BCD to Excess 3 converter and vice-versa

BCD				Excess 3			
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

Excess 3				BCD			
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	0
0	1	1	0	0	0	1	1
0	1	1	1	0	1	0	0
1	0	0	0	0	1	0	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	1	1
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	1

Gray-to-binary and vice-versa

It is sometimes convenient to use the Gray code to represent digital data that have been converted from analog data. The advantage of the Gray code over the straight binary number sequence is that only one bit in the code group changes in going from one number to the next.

The Gray code is used in applications in which the normal sequence of binary numbers may produce an error or ambiguity during the transition from one number to the next.

The truth table and logic circuit to implement the conversion is shown below.

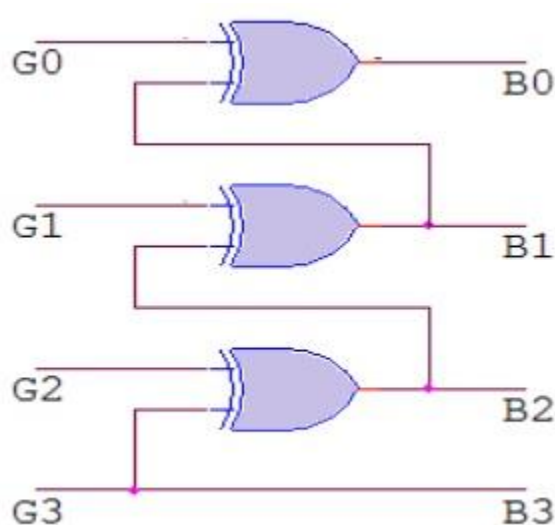


Fig 6.2: Gray to binary converter

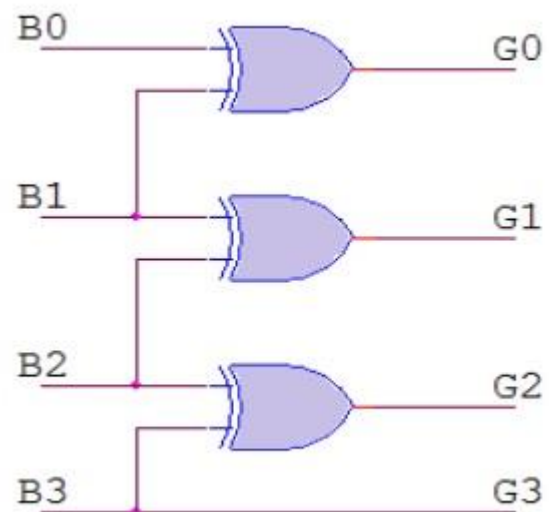


Fig 6.3: Binary to Gray converter

Table 6.2: Truth table of Gray to binary converter

Decimal No.	4bit Gray No.				4bit Binary Code			
	G3	G2	G1	G0	B3	B2	B1	B0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	1	0	0	1	0
3	0	0	1	0	0	0	1	1
4	0	1	1	0	0	1	0	0
5	0	1	1	1	0	1	0	1
6	0	1	0	1	0	1	1	0
7	0	1	0	0	0	1	1	1
8	1	1	0	0	1	0	0	0
9	1	1	0	1	1	0	0	1
10	1	1	1	1	1	0	1	0
11	1	1	1	0	1	0	1	1
12	1	0	1	0	1	1	0	0
13	1	0	1	1	1	1	0	1
14	1	0	0	1	1	1	1	0
15	1	0	0	0	1	1	1	1

$$B3 = G3$$

$$B2 = G3 \oplus G2$$

$$B1 = B2 \oplus G1$$

$$B0 = B1 \oplus G0$$

Table 6.3: Truth table of binary to Gray converter

Decimal No.	4bit Binary No.				4bit Gray Code			
	B3	B2	B1	B0	G3	G2	G1	G0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

$$G3 = B3$$

$$G2 = B3 \oplus B2$$

$$G1 = B2 \oplus B1$$

$$G0 = B1 \oplus B0$$

Result: Hence Code converters are designed and implemented

EXPERIMENT NO. 7

AIM: Design and implementation of one bit, two-bit magnitude comparators.

HARDWARE REQUIRED: IC (NOT-7404, AND-7408, OR-7432), Multimeter, Breadboard, Wires, LED, Resistor, Power source.

THEORY:

A magnitude digital Comparator is a combinational circuit that compares two digital or binary numbers in order to find out whether one binary number is equal, less than or greater than the other binary number. We logically design a circuit for which we will have two inputs one for A and other for B and have three output terminals, one for $A > B$ condition, one for $A = B$ condition and one for $A < B$ condition.

ONE BIT MAGNITUDE COMPARATOR:

A comparator used to compare two bits is called a single bit comparator. It consists of two inputs for two single bit numbers and three outputs to generate less than, equal to and greater than between two binary numbers.

Table 7.1: Truth Table

A	B	$A < B$	$A = B$	$A > B$
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

From the above truth table logical expressions for each output can be expressed as follows:

$$(A \text{ greater than } B) = AB'$$

$$(A \text{ less than } B) = A'B$$

$$(A \text{ equal to } B) = A'B' + AB$$

BLOCK DIAGRAM:

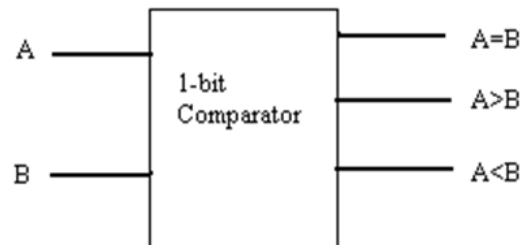


Fig 7.1: Block diagram of 1-bit comparat

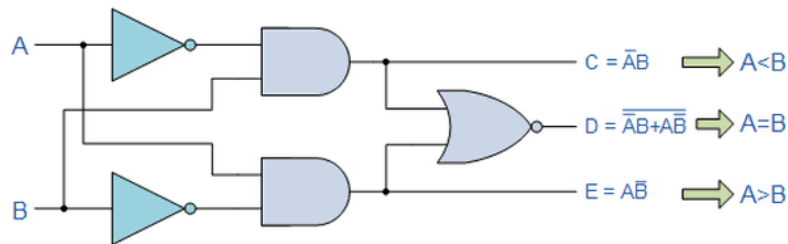


Fig 7.2: Logic diagram of 1 bit comparator

2-BIT MAGNITUDE COMPARATOR:

A comparator used to compare two binary numbers each of two bits is called a 2-bit magnitude comparator. It consists of four inputs and three outputs to generate less than, equal to and greater than between two binary numbers.

BLOCK DIAGRAM:

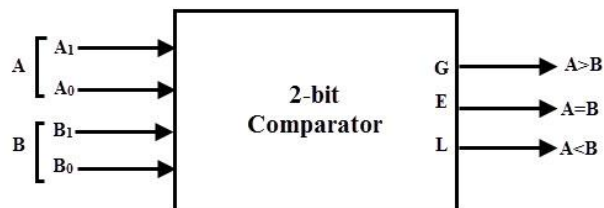
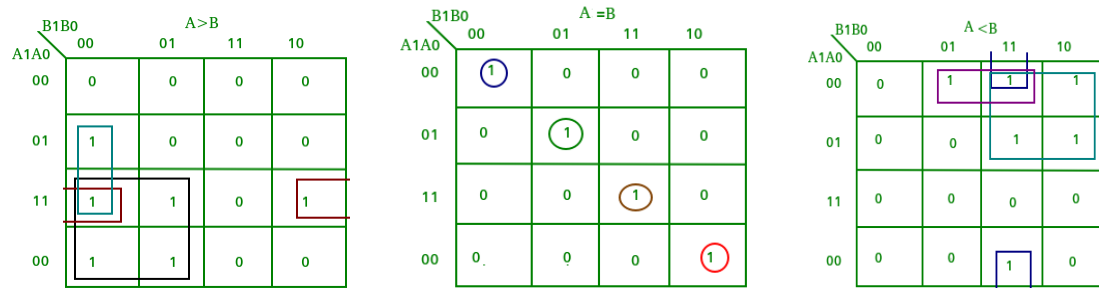


Fig 7.3: Block diagram of 2bit magnitude comparator

Table 7.2: Truth Table

Inputs				Outputs		
A ₁	A ₀	B ₁	B ₀	A > B	A = B	A < B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

K-MAP AND FORMULA: From the above truth table K-map for each output can be drawn as follows:



We get the following equations:

$$A \text{ greater than } B = A_1B_1' + B_0'(A_0B_1' + A_0A_1)$$

$$A \text{ less than } B = B_1A_1' + B_0B_1A_0' + A_1'A_0B_0$$

$$A \text{ equal to } B = \overline{A_0 \oplus B_0} \cdot \overline{A_1 \oplus B_1}$$

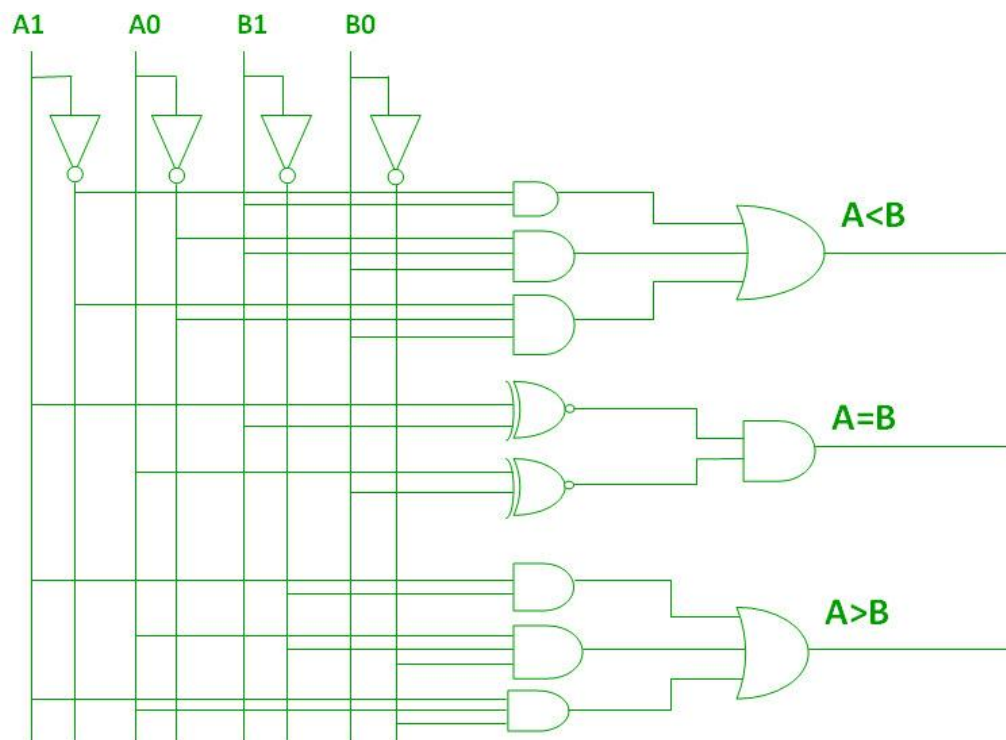


Fig 7.4: Logic diagram of 2 bit magnitude comparator

Result: Hence 1-bit and 2-bit magnitude comparator is designed and implemented.

EXPERIMENT NO. 8

AIM: Design and set up a Multiplexer (MUX) using gates and ICs.

HARDWARE REQUIRED: IC (AND-7408, OR-7432), Multimeter, Breadboard, Wires, LED, Resistor, Power source.

THEORY:

The multiplexer or MUX is a digital switch, also called as data selector. It is a combinational circuit with more than one input line, one output line and more than one select line. It allows the binary information from several input lines or sources and depending on the set of select lines, particular input line, is routed onto a single output line.

These multiplexers are available in IC forms with different input and select line configurations. Some of the available multiplexer ICs include 74157 (2-to-1 MUX), 78158 (2-to-1 MUX), 74352 (4-to-1 MUX), 74153 (4-to-1 MUX), 74152 (8-to-1 MUX) and 74150 (16-to-1 MUX).

2:1 MULTIPLEXER:

A 2-to-1 multiplexer consists of two inputs D_0 and D_1 , one select input S and one output Y . Depending on the select signal, the output is connected to either of the inputs. Since there are two input signals only two ways are possible to connect the inputs to the outputs, so one select is needed to do these operations. If the select line is low, then the output will be switched to D_0 input, whereas if select line is high, then the output will be switched to D_1 input.

Table 8.1 Truth table of 2:1 MUX

Select	Inputs		Output
0	0	0	0
0	0	1	1
1	1	0	1
1	1	1	1

From the above truth table logical expressions for each output can be expressed as follows:

$$Y = D_0 \bar{S} + D_1 S$$

BLOCK DIAGRAM:

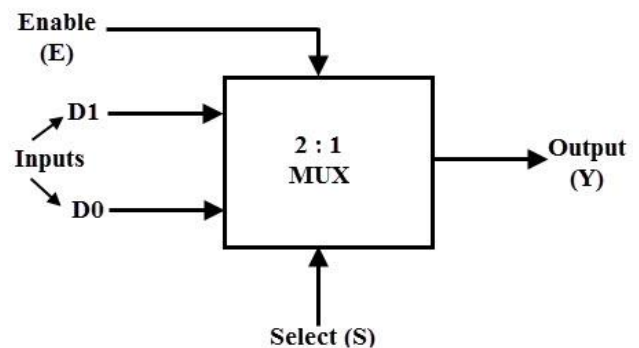


Fig 8.1: Block diagram of 2:1 Mux

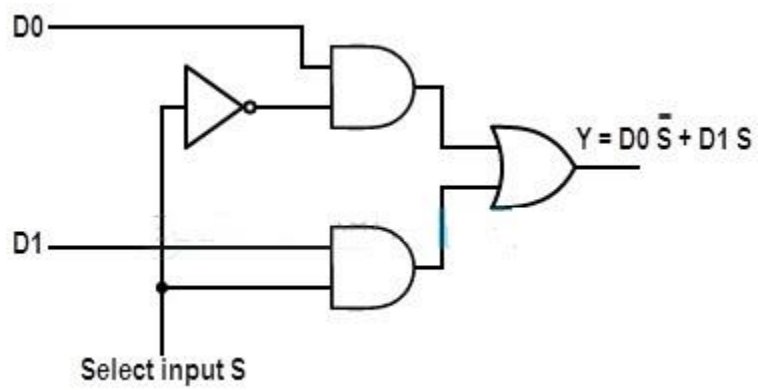


Fig 8.2: Logic diagram of 2:1 MUX

Result: Hence MUX is designed and studied.

EXPERIMENT NO. 9

AIM: Implementation and verification of truth table for J-K flip-flop, Master-slave J-K flip-flop, D flip-flop and T flip-flop.

HARDWARE REQUIRED: IC (NOT-7404, AND-7408, NAND-7400), Multimeter, Breadboard, Wires, LED, Resistor, Power source.

THEORY:

A flip-flop is an electronic circuit with two stable states that can be used to store binary data. The stored data can be changed by applying varying inputs. Flip-flops and latches are fundamental building blocks of digital electronics systems used in computers, communications, and many other types of systems. Flip-flops and latches are used as data storage elements. It is the basic storage element in sequential logic.

JK FLIP-FLOP:

Due to the undefined state in the SR flip flop, another flip flop is required in electronics. The JK flip flop is an improvement on the SR flip flop where $S=R=1$ is not a problem. The input condition of $J=K=1$, gives an output inverting the output state. However, the outputs are the same when one tests the circuit practically.

In simple words, If J and K data input are different (i.e. high and low) then the output Q takes the value of J at the next clock edge. If J and K are both low then no change occurs. If J and K are both high at the clock edge then the output will toggle from one state to the other. JK Flip Flop can function as Set or Reset Flip flop.

Table 9.1 : Truth table of JK FF

J	K	Q	Q'
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	1
0	0	1	1
0	1	1	0
1	0	1	1
1	1	1	0

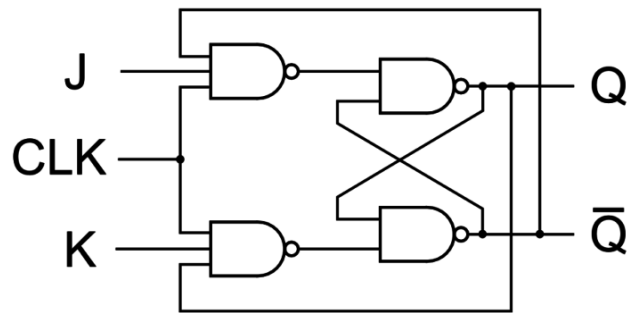


Fig 9.1: Logic diagram of JK FF

D FLIP-FLOP:

D flip-flop is a better alternative that is very popular with digital electronics. They are commonly used for counters and shift-registers and input synchronization. In D flip-flop, the output can be only changed at the clock edge, and if the input changes at other times, the output will be unaffected.

Table 9.2: Truth table of D FF

Clock	D	Q	Q'
↓ » 0	0	0	1
↑ » 1	0	0	1
↓ » 0	1	0	1
↑ » 1	1	1	0

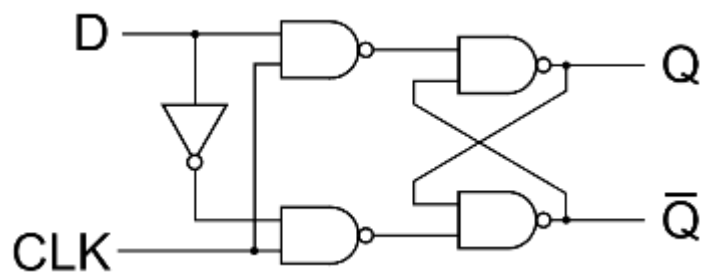


Fig 9.2: Logic diagram of D FF

T FLIP-FLOP:

T flip-flop is like JK flip-flop. These are basically a single input version of JK flip flop. This modified form of JK flip-flop is obtained by connecting both inputs J and K together. This flip

flop has only one input along with the clock input. These flip-flops are called T flip-flops because of their ability to complement its state (i.e.) Toggle, hence the name Toggle flip-flop.

Table 9.3: Truth table of T FF

T	Q	Q (t+1)
0	0	0
1	0	1
0	1	1
1	1	0

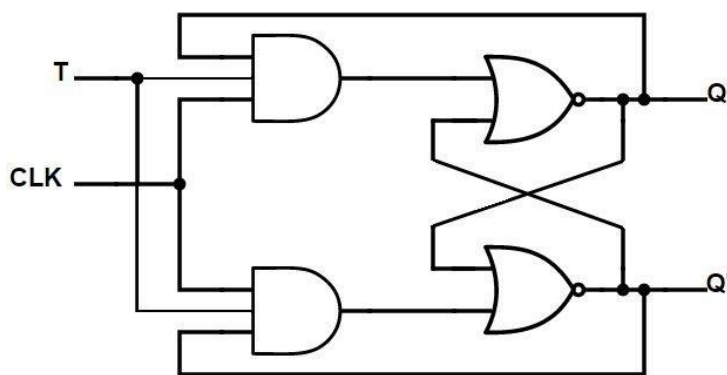


Fig 9.3: Logic diagram of T FF

MASTER SLAVE JK FLIP-FLOP:

The Master-Slave Flip-Flop is basically two gated SR flip-flops connected together in a series configuration with the slave having an inverted clock pulse. The outputs from Q and Q from the “Slave” flip-flop are fed back to the inputs of the “Master” with the outputs of the “Master” flip flop being connected to the two inputs of the “Slave” flip flop. This feedback configuration from the slave’s output to the master’s input gives the characteristic toggle of the JK flip flop as shown below.

The input signals J and K are connected to the gated “master” SR flip flop which “locks” the input condition while the clock (Clk) input is “HIGH” at logic level “1”. As the clock input of the “slave” flip flop is the inverse (complement) of the “master” clock input, the “slave” SR flip flop does not toggle. The outputs from the “master” flip flop are only “seen” by the gated “slave” flip flop when the clock input goes “LOW” to logic level “0”. When the clock is “LOW”, the outputs from the “master” flip flop are latched and any additional changes to its inputs are ignored. The gated “slave” flip flop now responds to the state of its inputs passed over by the “master” section. Then on the “Low-to-High” transition of the clock pulse the inputs of the “master” flip flop are fed through to the gated inputs of the “slave” flip flop and on the “High-to-

Low” transition the same inputs are reflected on the output of the “slave” making this type of flip flop edge or pulse-triggered.

Then, the circuit accepts input data when the clock signal is “HIGH”, and passes the data to the output on the falling-edge of the clock signal. In other words, the Master-Slave JK Flip flop is a “Synchronous” device as it only passes data with the timing of the clock signal.

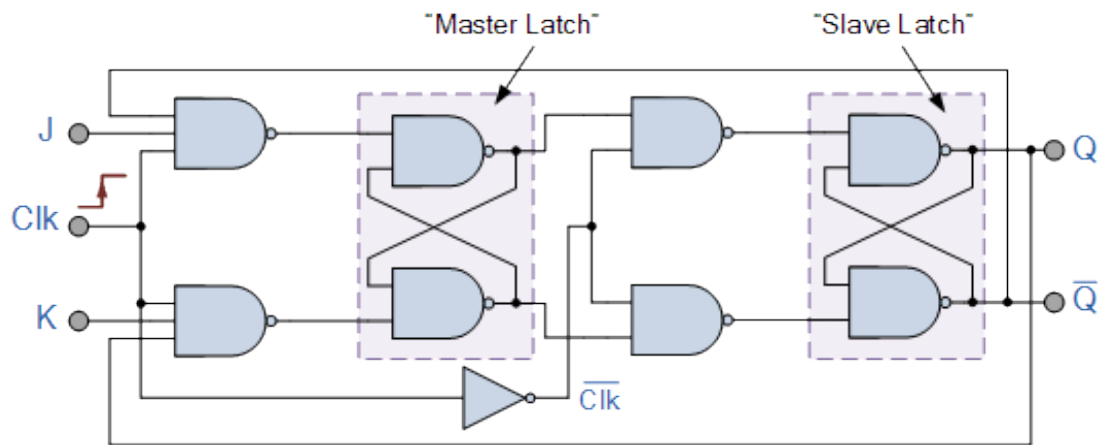


Fig 9.4: Logic diagram of Master Slave FF

Result: Hence various types of FF has been implemented and verified its truth table

EXPERIMENT NO. 10

AIM: To design and implement Mod-N synchronous counter using JK flip-flops.

HARDWARE REQUIRED: Breadboard, Wires, Resistors, LEDs, Power Supply, Multimeter, IC: 7410 (3 i/p NAND gate), 7400 (2 i/p NAND gate)

THEORY:

COUNTER:

In digital logic and computing, a counter is a device which stores the number of times a particular event or process has occurred, often in relationship to a clock. The job of a counter is to count by advancing the contents of the counter by one count with each clock pulse. Counters which advance their sequence of numbers or states when activated by a clock input are said to operate in a “count-up” mode. Likewise, counters which decrease their sequence of numbers or states when activated by a clock input are said to operate in a “count-down” mode. Counters that operate in both the “up” and “down” modes, are called bidirectional counters. The most common type is a sequential digital logic circuit with an input line called the clock and multiple output lines. Unlike Combinational Logic Circuits that change state depending upon the actual signals being applied to their inputs at that time, Sequential Digital Circuits use flip-flops as memory elements and are able to take into account their previous input state as well as those actually present.

The modulus of a counter is the number of different states it is allowed to have. Counter modulus is normally 2^N unless controlled by a feedback circuit which limits the number of possible states (an example being the decimal counter). A “Mod-N” counter will require “N” number of flip-flops connected together to count a single data bit while providing 2^n different output states, (n is the number of bits). Counters are very widely used in almost all computers and other digital electronic systems.

There are two major categories of counters:

1. Asynchronous Counter
2. Synchronous Counter

ASYNCHRONOUS COUNTER: In asynchronous counter, there is no use of universal clock. Only first flip flop is driven by main clock and the clock input of the rest of the following counters is driven by output of previous flip flops. Due to this chain system propagation delay appears during counting stage and create counting delays.

SYNCHRONOUS COUNTER: In synchronous counter, the clock input across all the flip flops use the same source and create the same clock signal at the same time. So, a counter which is using the same clock signal from the same source at the same time is called synchronous counter. Unlike the asynchronous counter, synchronous counter has one global clock which drives each flip flop, so output changes in parallel.

One advantage of synchronous counter over the other is that it can operate on higher frequency as it eliminates the cumulative flip-flop delay seen in ripple counter. J-K flip-flops are normally used in the synchronous counters due to the enabling (controlling) feature of the J and K inputs.

There are two types of synchronous counter:

1. Synchronous up counter
2. Synchronous down counter

MOD – 5 SYNCHRONOUS COUNTER:

Technically as well as being a 1-bit storage device, a single flip-flop on its own could be thought of as a MOD-2 counter, as it has a single output resulting in a count of two, either a 0 or 1, on the application of the clock signal. But a single flip-flop on its own produces a limited counting sequence, so by connecting together more flip-flops to form a chain, the counting capacity can be increased and a MOD counter of any value can be constructed.

For MOD-5 counter, $2^n \geq 5 \Rightarrow n=3$ (as $2^3 = 8 > 5$)

If a single flip-flop can be considered as a modulo-2 or MOD-2 counter, then adding two more flip-flops would give us a MOD-5 counter giving a natural count of 000 to 111 in binary (0-7 in decimal).

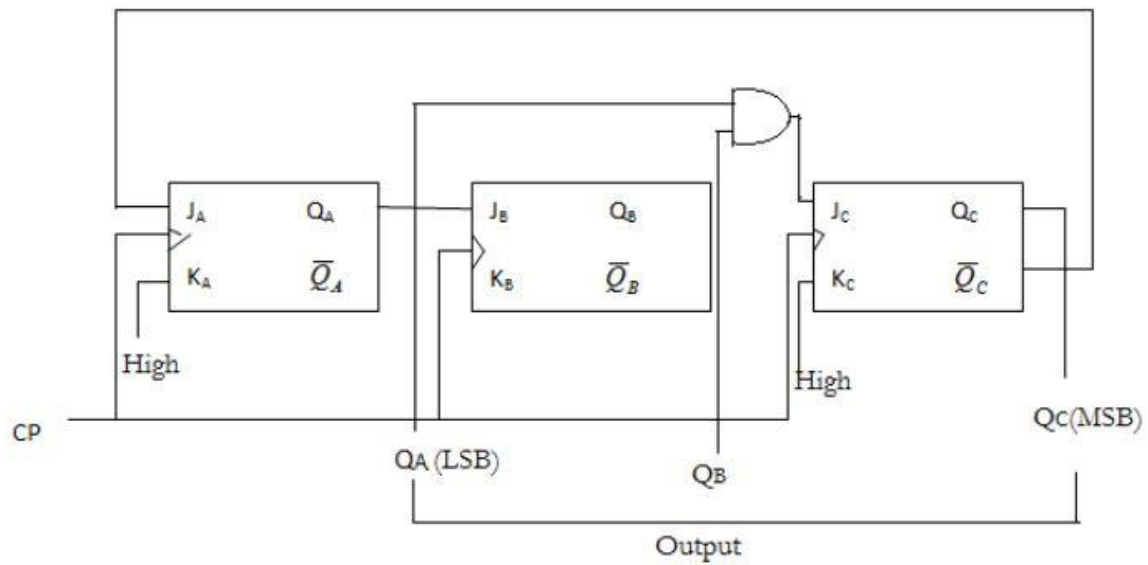


Fig 10.1: Logic diagram of Mod 5 Synchronous counter using JK FF

Result: Hence Mod 5 Synchronous counter has been designed and implemented.

EXPERIMENT NO. 11

AIM: Design and implementation of shift register to function as

- | | |
|-------------------|--------------------------|
| a) SISO | b) SIPO |
| c) PISO | d) PIPO |
| e) Shift left and | f) Shift right operation |

HARDWARE REQUIRED: Breadboard, wires, LEDs, Power source, Resistor, Multimeter, 7400 IC (NAND).

THEORY:

Flip flops can be used to store a single bit of binary data (1 or 0). However, in order to store multiple bits of data, we need multiple flip flops. N flip flops are to be connected in an order to store n bits of data. A **Register** is a device which is used to store such information. It is a group of flip flops connected in series used to store multiple bits of data. The information stored within these registers can be transferred with the help of **shift registers**. Shift Register is a group of flip flops used to store multiple bits of data. This sequential device loads the data present on its inputs and then moves or “shifts” it to its output once every clock cycle, hence the name Shift Register.

A shift register generally consists of several single bit “D-Type Data Latches”, one for each data bit, either a logic “0” or a “1”, connected together in a serial type daisy-chain arrangement so that the output from one data latch becomes the input of the next latch and so on.

- a) **Serial-in to Serial-out (SISO)** - the data is shifted serially “IN” and “OUT” of the register, one bit at a time in either a left or right direction under clock control.
- b) **Serial-in to Parallel-out (SIPO)** - the register is loaded with serial data, one bit at a time, with the stored data being available at the output in parallel form.
- c) **Parallel-in to Serial-out (PISO)** - the parallel data is loaded into the register simultaneously and is shifted out of the register serially one bit at a time under clock control.
- d) **Parallel-in to Parallel-out (PIPO)** - the parallel data is loaded simultaneously into the register, and transferred together to their respective outputs by the same clock pulse.
- e) **Shift left-** An n-bit shift register can be formed by connecting n flip-flops where each flip flop stores a single bit of data. The registers which will shift the bits to left are called “Shift left registers”.
- f) **Shift right** - An n-bit shift register can be formed by connecting n flip-flops where each flip flop stores a single bit of data. The registers which will shift the bits to right are called “Shift right registers”.

Logic Diagrams:

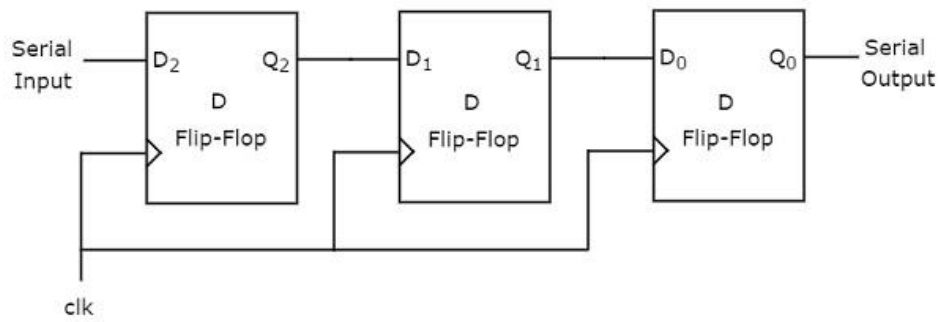


Fig 11.1: SISO shift register(3-bit)

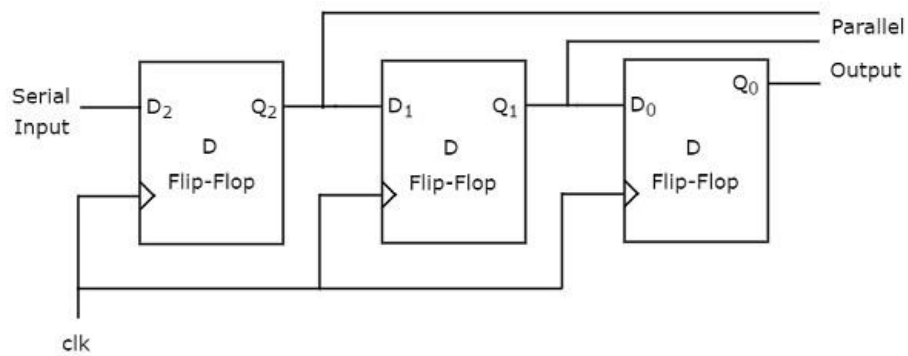


Fig 11.2: SIPO shift register(3 bit)

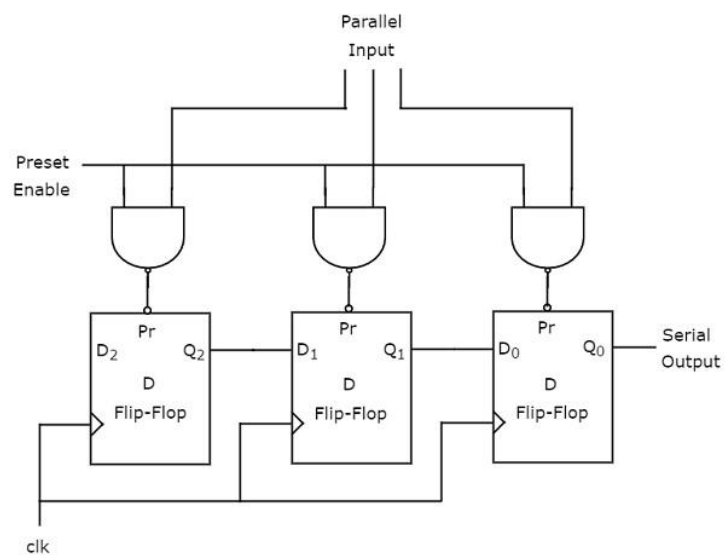


Fig 11.3: PISO shift registers(3 bit)

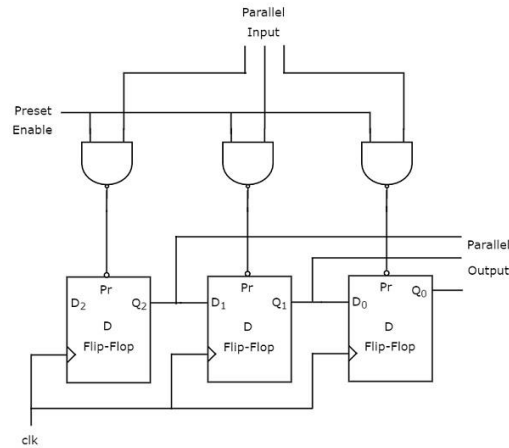


Fig 11.4: PIPO shift register(3 bit)

BIDIRECTIONAL SHIFT REGISTER:

If we shift a binary number to the left by one position, it is equivalent to multiplying the number by 2 and if we shift a binary number to the right by one position, it is equivalent to dividing the number by 2. To perform these operations we need a register which can shift the data in either direction. Bidirectional shift registers are the registers which are capable of shifting the data either right or left depending on the mode selected. If the mode selected is 1(high), the data will be shifted towards the right direction and if the mode selected is 0(low), the data will be shifted towards the left direction. The logic circuit given below shows a Bidirectional shift register. The circuit consists of four D flip-flops which are connected. The input data is connected at two ends of the circuit and depending on the mode selected only one and gate is in the active state.

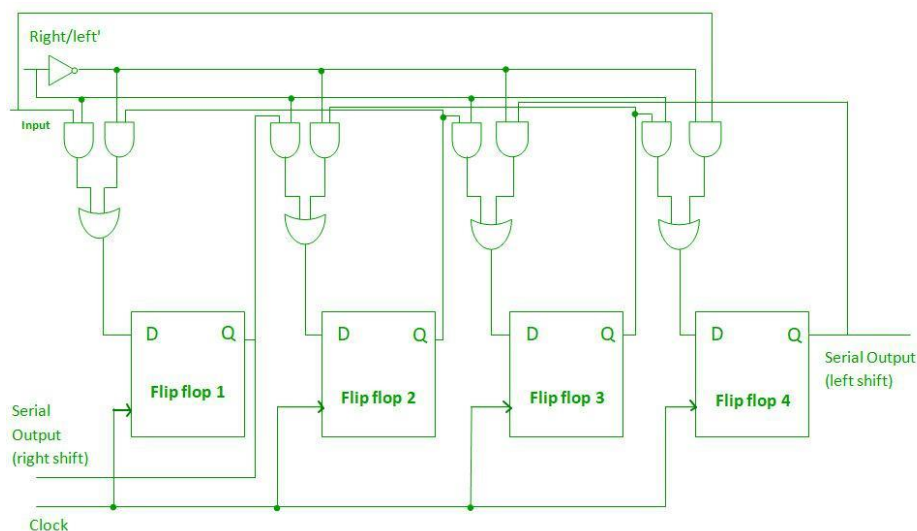


Fig 11.5: Logic diagram of bi-directional shift register

Result: Hence various shift registers are designed and implemented.

EXPERIMENT NO. 12

AIM: Design and implementation of

- a) Ring counter and
- b) Johnson counter using 4-bit shift register

HARDWARE REQUIRED: IC nos. 7474 (D flip flop) and 7404 (NOT gate), connecting wires, Breadboard, Multimeter, Power supply.

THEORY:

Counters are the shift registers in which the outputs are connected back to the inputs in order to produce particular sequences. These are basically of two types:

RING COUNTER:

A ring counter is basically a shift register counter in which the output of the first flip flop is connected to the next flip flop and so on and the output of the last flip flop is again fed back to the input of the first flip flop, thus the name ring counter. The data pattern within the shift register will circulate as long as clock pulses are applied. A Ring counter is generally used because it is self-decoding. No extra decoding circuit is needed to determine what state the counter is in.

The logic circuit given below shows a Ring Counter. The circuit consists of four D flip-flops which are connected. Since the circuit consists of four flip flops the data pattern will repeat after every four clock pulses as shown in the truth table below:

K-map:

Clock Pulse	Q1	Q2	Q3	Q4
0	1	0	0	1
1	1	1	0	0
2	0	1	1	0
3	0	0	1	1

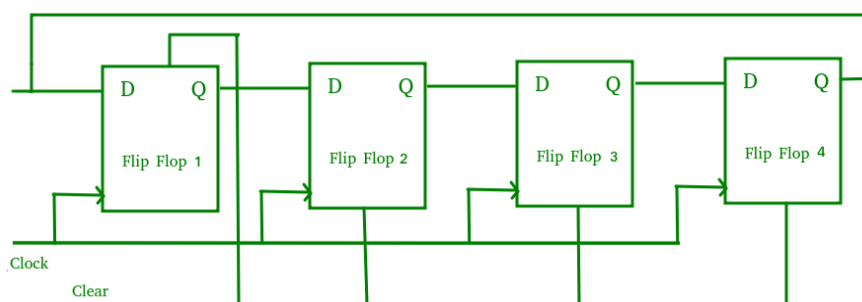


Fig 12.1: Logic diagram of Ring Counter

JOHNSON COUNTER:

A Johnson counter is basically a shift register counter in which the output of the first flip flop is connected to the next flip flop and so on and the inverted output of the last flip flop is again fed back to the input of the first flip flop. They are also known as twisted ring counters. The main advantage of Johnson counter is that it only needs n number of flip-flops compared to the ring counter to circulate a given data to generate a sequence of $2n$ states.

The logic circuit given below shows a Johnson Counter. The circuit consists of four D flip-flops which are connected. An n -stage Johnson counter yields a count sequence of $2n$ different states, thus also known as a mod- $2n$ counter. Since the circuit consists of four flip flops the data pattern will repeat every eight clock pulses as shown in the truth table below:

K-map:

Clock Pulse	Q1	Q2	Q3	Q4
0	0	0	0	1
1	0	0	0	0
2	1	0	0	0
3	1	1	0	0
4	1	1	1	0
5	1	1	1	1
6	0	1	1	1
7	0	0	1	1

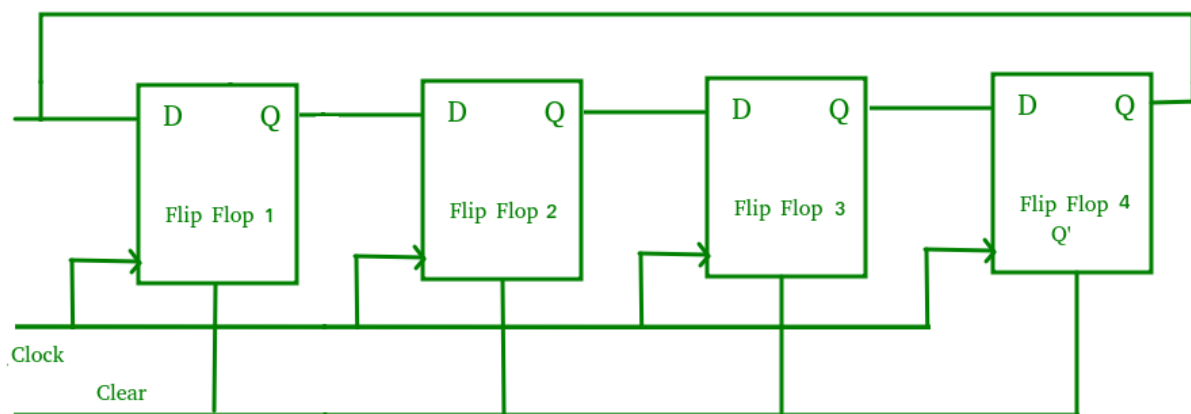


Fig 12.2: Logic diagram of Jhonson Counter

Result: Hence Ring counter and Jhonson counter has been designed and implemented.