Lab 1: Study of Gates & Flip-flops

1.1 Aim

To familiarize with circuit implementations using ICs and test the behavior of different logic gates and Flip-flops.

1.2 Hardware Requirement

a. Equipments - Digital IC Trainer Kit

b. Discrete Components - 74LS00Quad 2-Input NAND gate
74LS02 Quad 2-Input NOR gate
74LS04 Hex 1-Input NOT gate
74LS08 Quad 2-Input AND gate
74LS10 Triple 3-Input NAND gate
74LS11 Triple 3-Input AND gate
74LS32 Quad 2-Input OR gate
74LS86 Quad 2-Input XOR
74LS73 JK-Flip flop
74LS74 D Flip flop

1.3 Background

Digital logic devices are the circuits that electronically perform logic operations on binary variables. The binary information is represented by high and low voltage levels, which the device processes electronically. The devices that perform the simplest of the logic operations (such as AND, OR, NAND, etc.) are called gates. For example, an AND gate electronically computes the AND of the voltage encoded binary signals appearing at its inputs and presents the voltage encoded result at its output.

The digital logic circuits used in this laboratory are contained in integrated circuit (IC) packages, with generally 14 or 16 pins for electrical connections. Each IC is labeled (usually with an 74LSxx number) to identify the logic it performs. The logic diagrams and pin connections for these IC's are described in the TTL Data Book by Texas Instruments1.

The transistor-transistor logic(TTL) IC's used in this laboratory require a 5.0 volt power supply for operation. TTL inputs require a voltage greater than 2 volts to represent a binary 1 and a voltage less than 0.8 volts to represent a binary 0.

Pin numbering is standard on IC's. Figure 1-1 illustrates the pin numbering for a 14-pin dual in-line package (DIP). With the IC oriented as shown, the numbering starts at the top left and proceeds counter-clockwise around the chip:

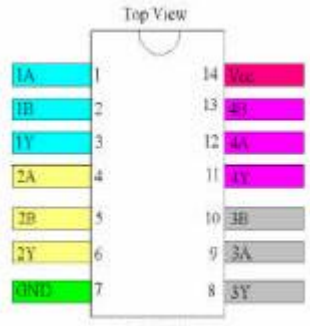
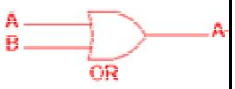
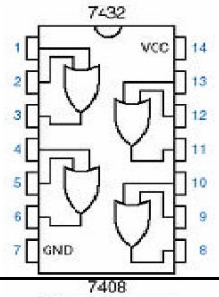
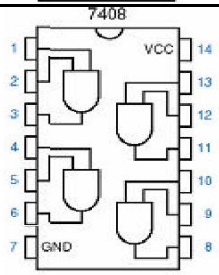
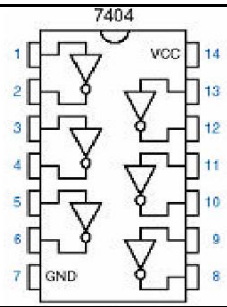
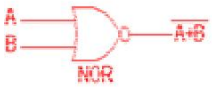
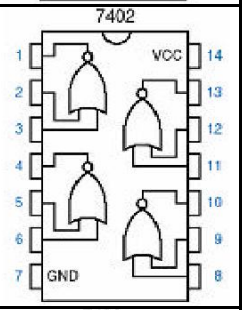
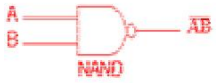
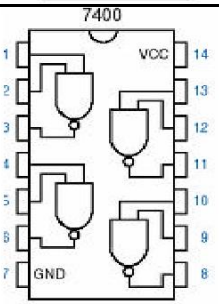
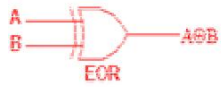
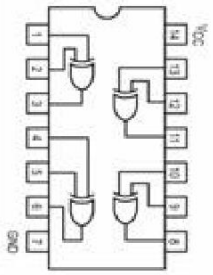


FIGURE 1-1

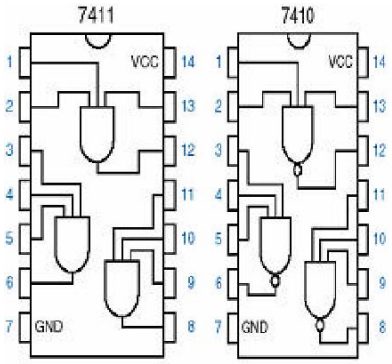
To construct circuits with IC's, a circuit board that allows easy connections to IC pins should be used. The circuit board contains rows of solder less tie points, a 5-volt power supply, a common circuit point (ground), toggle switches for input, and LEDs (light emitting diodes) for output.

1.3.1 Logic Gates and their Properties

Gate	Description	Truth Table			Logic Symbol	Pin Diagram
OR	The output is active high if any one of the input is in active high state, Mathematically, **Q = A+B**	**A** 0 0 1 1	**B** 0 1 0 1	**Output** 0 1 1 1		
AND	The output is active high only if both the inputs are in active high state, Mathematically, **Q = A.B**	**A** 0 0 1 1	**B** 0 1 0 1	**Output** 0 0 0 1		
NOT	In this gate the output is opposite to the input state, Mathematically, **Q = (A)'**	**A** 0 1		**Output** 1 0		
NOR	The output is active high only if both the inputs are in active low state, Mathematically, **Q = (A+B)'**	**A** 0 0 1 1	**B** 0 1 0 1	**Output** 1 0 0 0		
NAND	The output is active high only if any one of the input is in active low state, Mathematically, **Q = (A.B)'**	**A** 0 0 1 1	**B** 0 1 0 1	**Output** 1 1 1 0		

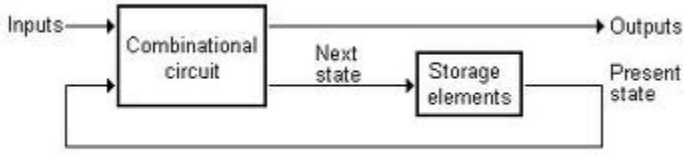
EXOR	The output is active high **only if** any one of the input is in active high state, Mathematically, **Q = A'B+AB '**	**A** 0 0 1 1	**B** 0 1 0 1	**Output** 0 1 1 0		**7486** 

1.3.2 Pin Diagram of 3-Input Gates



1.4 Flip- Flops-Theory

Digital electronic circuit is classified into combinational logic and sequential logic. Combinational logic output depends on the inputs levels, whereas sequential logic output Depends on stored levels and also the input levels.



The storage elements (Flip -flops) are devices capable of storing 1-bit binary info. The binary info stored in the memory elements at any given time defines the state of the Sequential circuit. The input and the present state of the memory element determines the output. Storage elements next state is also a function of external inputs and present state.

1.4.1. Flip-Flops and their properties

Flip-flops are synchronous bistable devices. The term synchronous means the output changes state only when the clock input is triggered. That is, changes in the output occur in synchronization with the clock. A flip-flop circuit has two outputs, one for the normal value and one for the complement value of the stored bit. Since memory elements in sequential circuits are usually flip-flops, it is worth summarizing the behavior of various flip-flop types before proceeding further. All flip -flops can be divided into four basic types: **SR**, **JK**, **D** and **T**. They differ in the number of inputs and in the response invoked by different value of input signals. The four types of flip -flops are defined in the Table 1.1.

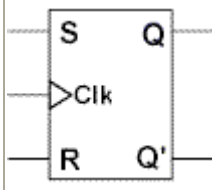
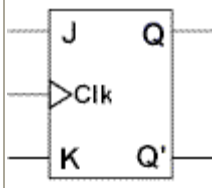
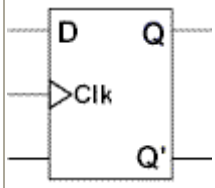
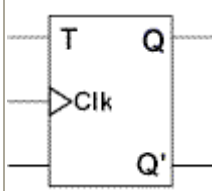
<table>
<tr><th>Flip-Flop Name</th><th>Flip-Flop Symbol</th><th>Characteristic Table</th><th>Characteristic Equation</th><th>Excitation Table</th></tr>
<tr>
<td>SR</td>
<td>
</td>
<td>
<table>
<tr><th>S</th><th>R</th><th>Q(next)</th></tr>
<tr><td>0</td><td>0</td><td>Q</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>?</td></tr>
</table>
</td>
<td>Q(next) = S + R’Q
SR = 0</td>
<td>
<table>
<tr><th>Q</th><th>Q(next)</th><th>S</th><th>R</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>X</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>X</td><td>0</td></tr>
</table>
</td>
</tr>
<tr>
<td>JK</td>
<td>
</td>
<td>
<table>
<tr><th>J</th><th>K</th><th>Q(next)</th></tr>
<tr><td>0</td><td>0</td><td>Q</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>Q’</td></tr>
</table>
</td>
<td>Q(next) = JQ’ + K’Q</td>
<td>
<table>
<tr><th>Q</th><th>Q(next)</th><th>J</th><th>K</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>X</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>X</td></tr>
<tr><td>1</td><td>0</td><td>X</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>X</td><td>0</td></tr>
</table>
</td>
</tr>
<tr>
<td>D</td>
<td>
</td>
<td>
<table>
<tr><th>D</th><th>Q(next)</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
</table>
</td>
<td>Q(next) = D</td>
<td>
<table>
<tr><th>Q</th><th>Q(next)</th><th>D</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
</td>
</tr>
<tr>
<td>T</td>
<td>
</td>
<td>
<table>
<tr><th>T</th><th>Q(next)</th></tr>
<tr><td>0</td><td>Q</td></tr>
<tr><td>1</td><td>Q’</td></tr>
</table>
</td>
<td>Q(next) = TQ’ + T’Q</td>
<td>
<table>
<tr><th>Q</th><th>Q(next)</th><th>T</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td>
</tr>
</table>

Table 1.1 Flip-flops and their properties

The ***characteristic table*** in the third column of Table 1.1 defines the state of each flip-flop as a function of its inputs and previous state. **Q** refers to the present state and **Q(next)** refers to the next state after the occurrence of the clock pulse **Pin Diagram of JK-Flip flop**

IC7474-D - Flip flop Connection Diagram with Function Table

INPUTS				OUTPUTS	
PR	CLR	CLK	D	Q	$\overline{Q}$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H	H
H	H	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	Q_0	$\overline{Q0}$

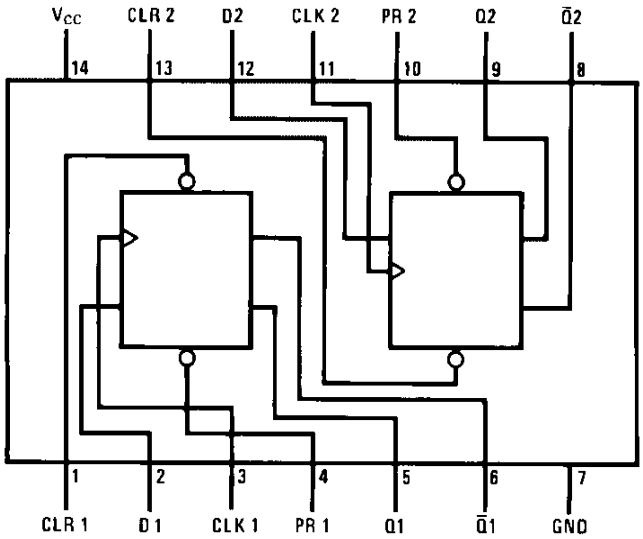
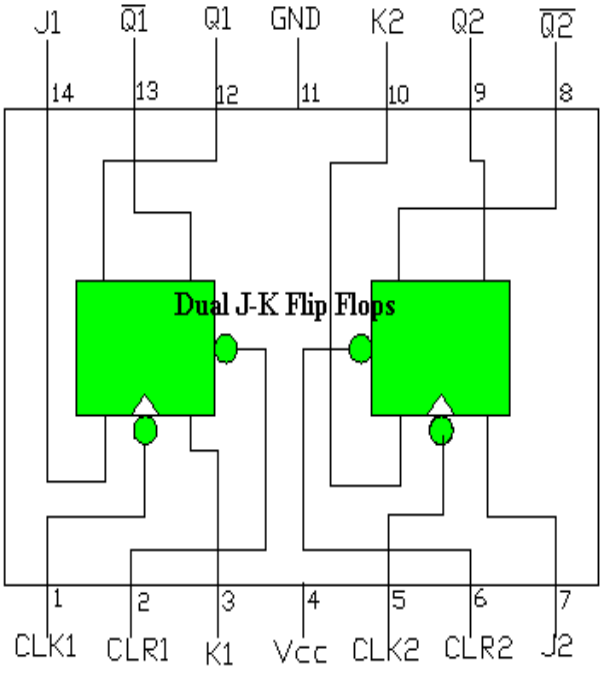


Figure 1.1 D Flip-flop connection diagram

H HIGH Logic Level
X Either LOW or HIGH Logic Level
L LOW Logic Level
↑ Positive-going transition of the clock.
Q0 The output logic level of Q before the indicated input conditions were established.

IC7473-JK- Flip flop Connection Diagram with Function Table

Connection Diagram



Function Table

Inputs				Outputs	
CLR	CLK	J	K	Q	$\overline{Q}$
L	X	X	X	L	H
H	⎍	L	L	Q_0	$\overline{Q}_0$
H	⎍	H	L	H	L
H	⎍	L	H	L	H
H	⎍	H	H	Toggle	

H = HIGH Logic Level
L = LOW Logic Level
X = Either LOW or HIGH Logic Level
⎍ = Positive pulse data. the J and K inputs must be held constant while the clock is HIGH. Data is transferred to the outputs on the falling edge of the clock pulse.
Q_0 = The output logic level before the indicated input conditions were established.
Toggle = Each output changes to the complement of its previous level on each HIGH level clock pulse.

1.4 Pre -lab Questions

1. A basic 2-input logic circuit has a HIGH on one input and a LOW on the other input, and the output is HIGH. What type of logic circuit is it?
2. A logic circuit requires HIGH on all its inputs to make the output HIGH. What type of logic circuit is it?
3. Develop the truth table for a 3-input AND gate and also determine the total number of possible combinations for a 4-input AND gate.
4. Which logic gate is used as a two-bit adder?
5. What is Flip flop?

1.5 Lab Procedure

- Refer datasheet for the input and output pin numbers of the IC. (For example in a NAND gate IC, pin numbers 1, 2, 4, 5, 9 10, 12, 13 are inputs and 3, 6, 8 and 11 are outputs).
- Connect the particular input pins to the logic input section using a connecting wire.
- Similarly connect the output pin to the logic output section of the trainer kit.
- Verify the functionality of NOT gate and 2-input AND, OR, NAND, NOR and EXOR gates,JK Flip flop and D Flip flop.
- Write the truth-table for each.

1.7 Result

Truth table for Logic Gates and Flip-flops are verified.

1.8 Post-lab Questions

a. If the two waveforms A and B shown in figure 1.2 are applied to the circuit, draw the timing diagram for the circuit, showing the outputs of G1, G2 and G3 with the inputs A and B.

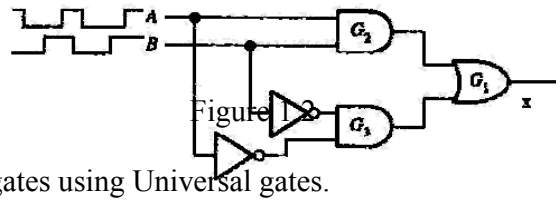
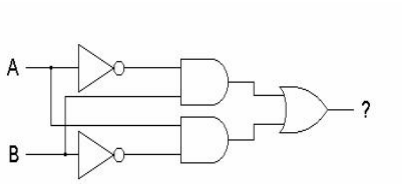


Figure 1.2

2. Implement the basic gates using Universal gates.
3. Implement NOR using NAND gates and NAND gate using NOR gates.
4. What type of logic gate does this logic circuit configuration represent?



NAND Gate EXOR Gate

NOR Gate EXNOR Gate

5. What is the Difference between Combinational circuits and Flip-flop?

Lab 2: Design of Half Adder and Full Adder

2.1 Aim

To design and verify the truth table for half adder & full adder.

2.2 Hardware Requirement

Equipment : Digital IC Trainer Kit

Discrete Components : 74LS08 Quad 2 input AND gate
74LS32 Quad 2 input OR gate
74LS86 Quad 2 input XOR gate

2.3 Theory

A Binary adder is a circuit which is able to add together two binary numbers. The half adder adds two binary digits an addend and an augend to produce a sum and carry. The sum can be implemented by using an Exclusive OR gate and an AND gate can be used for carry generation.

The Boolean expression for the sum and carry are

Sum $S = A \oplus B$

Carry $C = A.B$

The full adder adds an addend, an augend and carry input generated by the previous stage addition. It has two outputs, sum and carry. Full adder circuit can be implemented using AND,OR and EX-OR gates. Full adder circuit can also be implemented with the help of two half adder circuits. The first half adder is used to add two inputs and generate sum and carry output. Then second half adder combines the sum and carry input and generate final sum and carry out.

The sum and carry can be expressed as

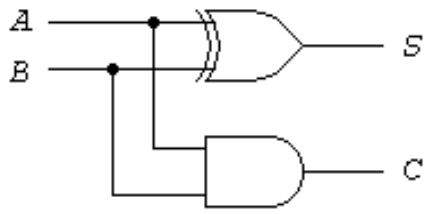
Sum $S = A \oplus B \oplus C_{in}$

Carry $C = (A \oplus B)C_{in} + AB = AB + BC_{in} + AC_{in}$

2.3.1 Experimental Procedure:

HALF ADDER:

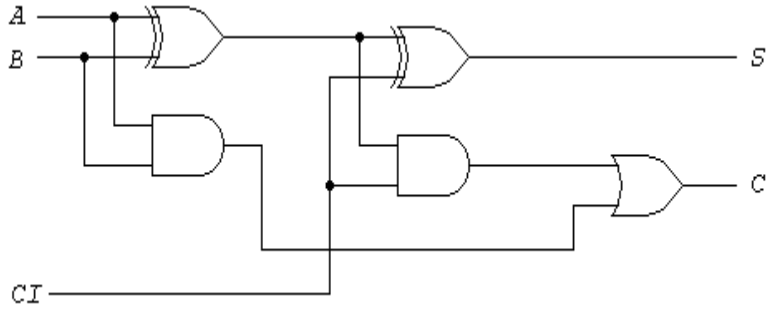
Circuit Diagram



Truth table

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

FULL ADDER
Circuit Diagram



Truth Table

A	B	C_{in}	S	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

2.4 Pre lab Questions

1. What is meant by combinational circuit?
2. List the combinational circuits?
3. What is a Half adder?
4. What is a Full adder?
5. What is a Parallel adder?
6. State the limitations of Karnaugh map.

2.5 Lab Procedure

1. Connect the circuit as shown in the figure.
2. Connect the power supply and ground to the respective pin numbers to all the IC's.
3. Give the inputs and verify the output in the truth table.

2.6 Result:

The truth tables of half adder and full adder circuits are verified.

2.7 Post lab Questions

1. Two numbers (1101 and 1011) are applied to a 4-bit parallel adder. The input Carry is '1'. Determine the sum and output carry.
2. Draw the circuit of a 4 bit Adder.

Lab 3: Design of Magnitude Comparator

3.1 Introduction

The purpose of this experiment is to introduce the design of 2-Bit Magnitude Comparator.

3.2 Hardware Requirement

a. Equipments - Digital IC Trainer Kit

b. Discrete Components -

74LS02 Quad 2-Input NOR gate
74LS04 Hex 1-Input NOT gate
74LS08 Quad 2-Input AND gate
74LS00 Quad 2-Input NAND gate
74LS266 Quad 2-Input XNOR gate
74LS86 Quad 2-Input XOR
74LS10 Triple 3-Input NAND

3.3 Background

Digital or Binary Comparators are made up from standard AND, NOR and NOT gates that compare the digital signals at their input terminals and produces an output depending upon the condition of the inputs. For example, whether input A is greater than, smaller than or equal to input B etc.

Digital Comparators can compare a variable or unknown number for example A (A1, A2, A3, An, etc) against that of a constant or known value such as B (B1, B2, B3, ... Bn, etc) and produce an output depending upon the result. For example, a comparator of 2-bit, (A and B) would produce the following three output conditions. **A > B, A = B, A < B** This is useful if we want to compare two values and produce an output when the condition is achieved. For example, produce an output from a counter when a certain count number is reached. Consider the simple 2-bit comparator below.

3.3.1 2 – Bit Magnitude Comparator

Implementation
$A = A_1A_0$
$B = B_1B_0$
Here each subscript represents one of the digits in the numbers.

Equality:

The binary numbers A and B will be equal if all the pairs of significant digits of both numbers are equal, i.e.,

$A_1 = B_1$ and $A_0 = B_0$

Since the numbers are binary, the digits are either 0 or 1 and the Boolean function for equality of any two digits Ai and Bi can be expressed as

$$x_i = A_i \cdot B_i + \overline{A_i} \cdot \overline{B_i}$$

x_i is 1 only if A_i and B_i are equal.

For the equality of A and B, all xi variables (for i=0,1) must be 1. So the equality condition of A and B can be implemented using the AND operation as

$(A = B) = x_1 x_0$

The binary variable (A=B) is 1 only if all pairs of digits of the two numbers are equal.

Inequality:
In order to manually determine the greater of two binary numbers, we inspect the relative magnitudes of pairs of significant digits, starting from the most significant bit, gradually proceeding towards lower significant bits until an inequality is found. When an inequality is found, if the corresponding bit of A is 1 and that of B is 0 then we conclude that A>B. (A>B) and (A < B) are output binary variables, which are equal to 1 when A>B or A<B respectively.

3.3.2 Experimental Procedure:

Circuit diagram:

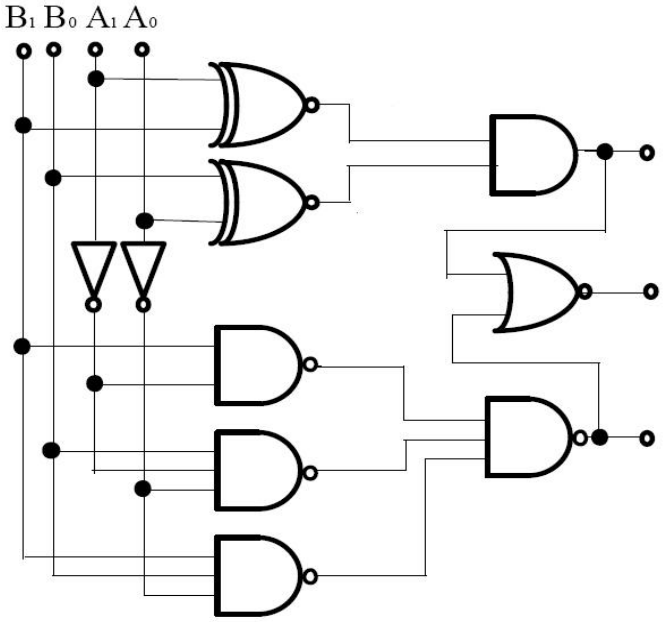


Figure 3.1 2 bit Magnitude Comparator

2 Bit Magnitude Comparator Truth table:

INPUTS	OUTPUTS

A1	A0	B1	B0	A < B	A = B	A > B
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

3.4 Pre – Lab questions

1. Define magnitude comparator?
2. Write the output expressions for A<B and A>B.

3. Find out and label the output terminals (A<B,A=B and A>B) for the given 2 bit comparator circuit. (experimental circuit diagram)
4. List out the applications of comparators?
5. Which Logic gate is used to find A=B?

3.5 Lab Procedure

1. Construct the logic circuit of the 2-bit magnitude comparator shown in Figure 3.1.
2. Use different sets of inputs for A and B to check each of the outputs A<B, A=B and A>B.

3.6 Result:

Thus the 2-bit magnitude comparator is designed and verified.

3.7 Post Lab questions:

1. Design 1 bit comparator using logic gates.
2. Design 8 bit comparator using IC 7485.
3. How many 4-bit comparators are needed to construct 12-bit comparator?
4. How many IC 7485 required to design 24 bit comparator?
5. Give a summary of the points that you have learned from this experiment.

Lab. 4: Design of Encoder & Decoder

4.1 Aim

To design and verify the truth table for 8-3 Encoder & 3-8 Decoder logic circuit.

4.2 Hardware Requirement

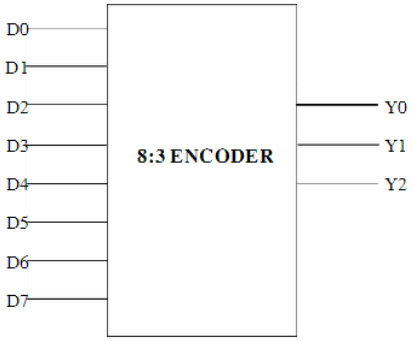
Equipment : Digital IC Trainer Kit
Discrete Components : 74LS08 Quad 2 input AND gate
74LS32Quad 2 input OR gate
74LS04 Hex 1 input NOT gate

4.3 Theory:

(i) Encoder: Encoder takes all the data inputs one at a time and converts them to a single encoded output, it is a multi-input data line, combinational logic circuit that converts the logic level 1 data at its input to an equivalent binary code at its output. Encoder has 2^n input lines with common types that include 4 to 2,8 to 3 & 16 to 4 line configuration. Encoders are available to encode either a decimal or hexadecimal input pattern to typically binary or BCD output code.

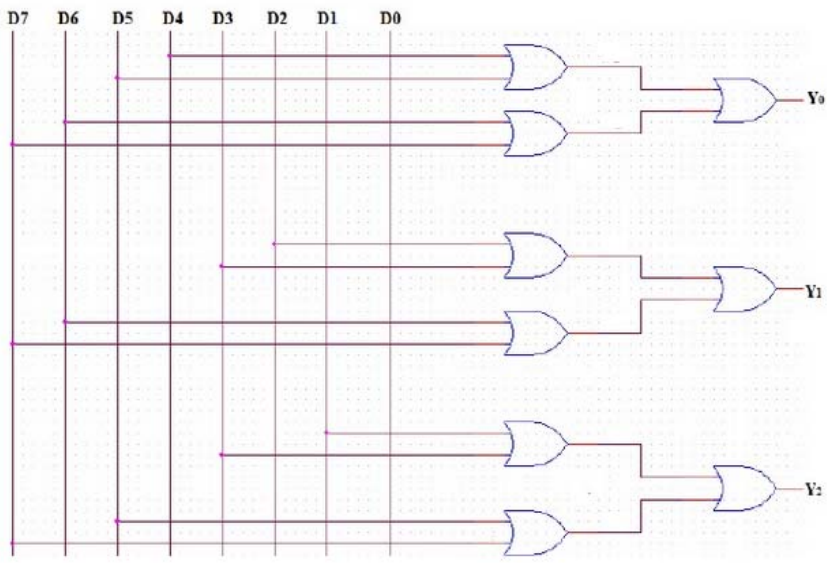
(ii)Decoder: A decoder is a multiple-input, multiple-output logic circuit that converts coded inputs into coded outputs, where the input and output codes are different; e.g. n-to-2n, BCD decoders. Decoding is necessary in applications such as data multiplexing, 7 segment display and memory address decoding. Any n-variable logic function, in canonical sum-of-minterms form can be implemented using a single n-to-2^n decoder to generate the minterms, and an OR gate to form the sum. The output lines of the decoder corresponding to the minterms of the function are used as inputs to the or gate. Any combinational circuit with n inputs and m outputs can be implemented with an n-to-2^n decoder with m OR gates. Suitable when a circuit has many outputs, and each output function is expressed with few minterms.

Block Diagram

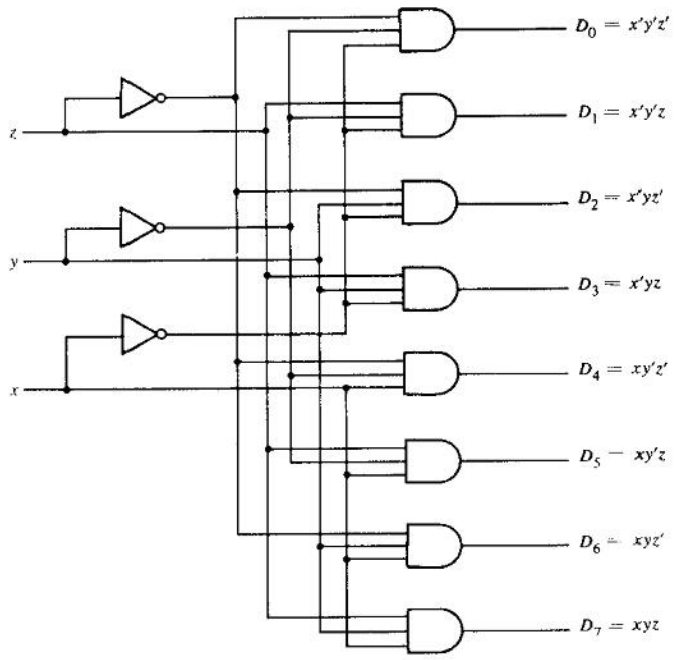


Input								Output		
D0	D1	D2	D3	D4	D5	D6	D7	Y0	Y1	Y2
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

Logic circuit for 8-3 encoder:



3-8 decoder logic circuit:



Truth Table for 3-8 Decoder:

Inputs			Outputs							
X	Y	Z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	**1**	0	0	0	0	0	0	0
0	0	1	0	**1**	0	0	0	0	0	0
0	1	0	0	0	**1**	0	0	0	0	0
0	1	1	0	0	0	**1**	0	0	0	0
1	0	0	0	0	0	0	**1**	0	0	0
1	0	1	0	0	0	0	0	**1**	0	0
1	1	0	0	0	0	0	0	0	**1**	0
1	1	1	0	0	0	0	0	0	0	**1**

4.4 Procedure

- The truth table and a design of 8 to 3 Encoder, 3 to 8 decoder are given.
- Realize this circuit on your board by using logic circuit.
- Connect three inputs x,y,z to the switches & eight outputs viceversa.
- Connect the functions outputs to LEDs.
- Verify input/output relation (Truth table) of this converter.

4.5 Pre lab Questions

1. What is an encoder?
2. What is a decoder?
3. List the difference between Mux & encoder?
4. What is the use of Encoder?
5. Draw 4-2 encoder circuit?

4.6 Result

The truth tables of Encoder & Decoder circuits are verified.

4.7 Post lab Questions

1. Draw Full adder with decoder circuit?
2. Draw the logic symbol of 3x8 Decoder using two 2x4 decoder?
3. Implement the following expression using decoder F=XY+YZ.
4. Write the truth table for 3-input priority encoder.

Lab 5: Design of 4:1 Multiplexer and 1:4 De-multiplexer

5.1 Aim

To implement and verify the functional table of 4:1 Multiplexer and 1:4 De-multiplexer.

5.2 Hardware Requirement

a. Equipments - Digital IC Trainer Kit

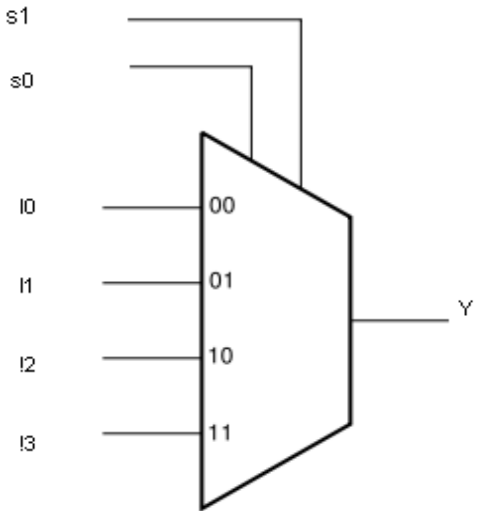
b. Discrete Components - 74LS04 Hex 1-Input NOT gate
74LS08 Quad 2-Input AND gate
74LS11 Triple 3-Input AND gate
74LS32 Quad 2-Input OR gate

5.3 Theory

5.3.1 Multiplexer

Multiplexers which sometimes are simply called "Mux" or "Muxes", are devices that act like a very fast acting rotary switch. They connect multiple input lines 2, 4, 8, 16 etc one at a time to a common output line and are used as one method of reducing the number of logic gates required in a circuit. A multiplexer of 2^n inputs has n select bits, which are used to select which input line to send to the output.

A multiplexer, or data selector, can be also be used to implement combinational logic circuits. A ***multiplexer implementation table*** is used to determine the input connections for the multiplexer.
A 2 x 1 multiplexer can be used to implement a function of 2 variables, such as f(A,B)
A 4 x 1 multiplexer can be used to implement a function of 3 variables, such as f(A,B,C)
A 8 x 1 multiplexer can be used to implement a function of 4 variables, such as f(A,B,C,D)



(a) Block Diagram

s1	s0	Y
0	0	I0
0	1	I1
1	0	I2
1	1	I3

(b) Function Table

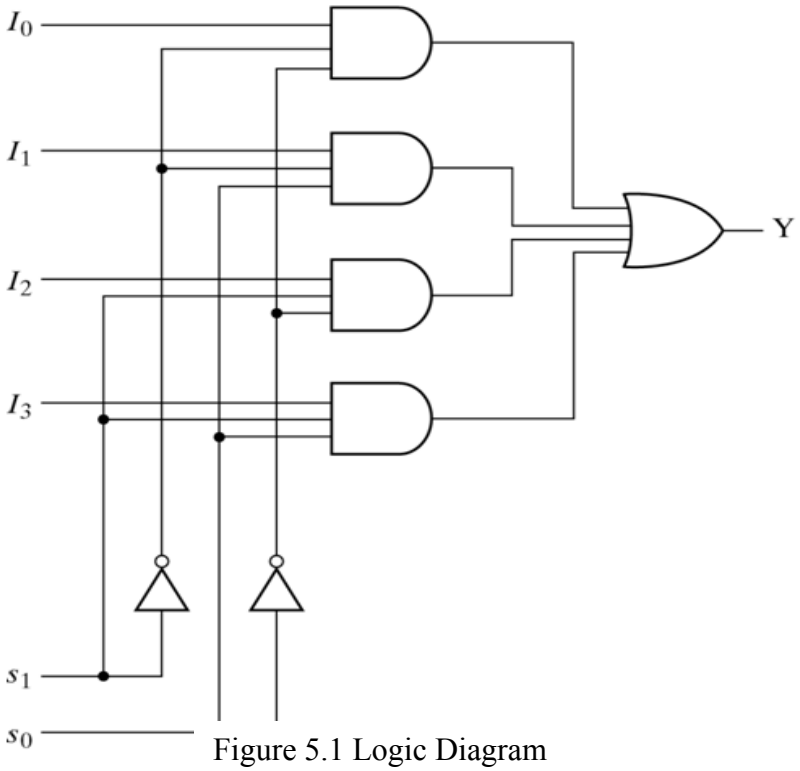


Figure 5.1 Logic Diagram

The Boolean function for a 4x1Multiplexer is: $Y = I_0s_1's_0' + I_1s_1's_0 + I_2s_1s_0' + I_3s_1s_0$

5.3.2 Implement the Function using MUX

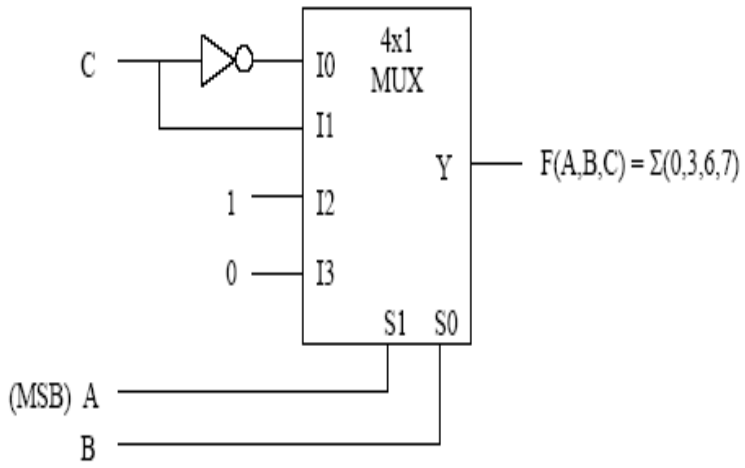
F(A,B,C) = Σ(0, 3, 6, 7) using a 4 x 1 multiplexer.

Connect A and B to select lines

Express F in terms of the other input (C)

	A	B	C	F	
MUX input 0	0	0	0	1	F = C' (so connect C' to MUX input 0)
	0	0	1	0	
	0	1	0	0	F = C
	0	1	1	1	
	1	0	0	0	F = 0
	1	0	1	0	
	1	1	0	1	F = 1
	1	1	1	1	

MUX Implementation Table

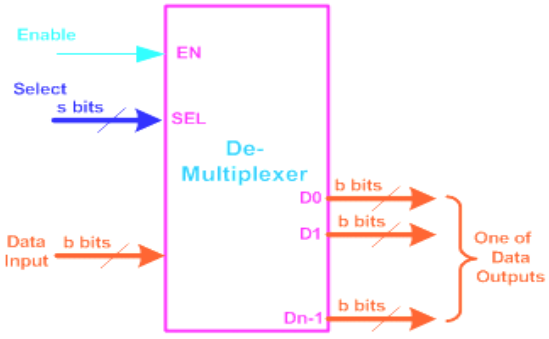


Logic Diagram of F(A,B,C) = Σ(0, 3, 6, 7)

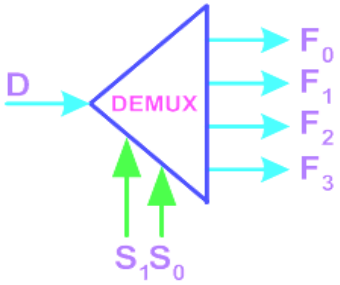
5.3.3 De-multiplexers

De-multiplexers or "De-muxes", are the exact opposite of the **Multiplexers** we saw in the previous tutorial in that they have one single input data line and then switch it to any one of their individual multiple output lines one at a time. **The De-multiplexer** converts the serial data signal at the input to a parallel data at its output lines.

They are digital switches which connect data from one input source to one of n outputs. Usually implemented by using n-to-2n binary decoders where the decoder enable line is used for data input of the de-multiplexer.The figure below shows a de-multiplexer block diagram which has got s-bits-wide select input, one b-bits-wide data input and n b-bits-wide outputs.



De-Multiplexer Block Diagram



1x4 DeMUX Logic Symbol

S1	S0	F0	F1	F2	F3
0	0	D	0	0	0
0	1	0	D	0	0
1	0	0	0	D	0
1	1	0	0	0	D

1x4 DeMUX –Truth Table

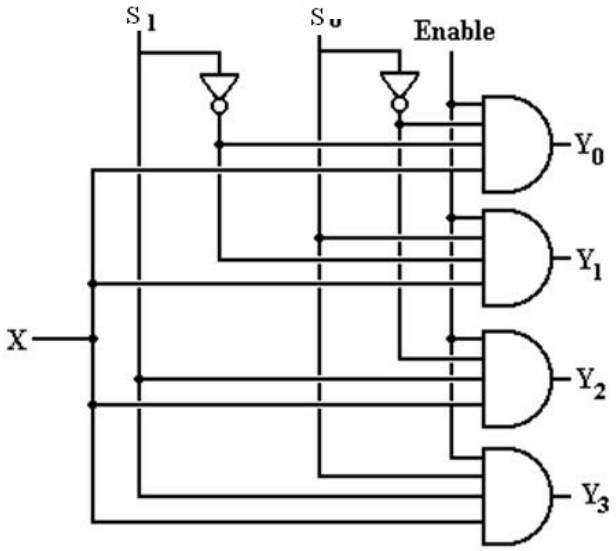


Figure5.2 1:4 Demultiplexer

5.4 Lab Procedure

MUX

a. Write the Functional table for 4: 1 MUX. (as shown in Fig. 5.1)
b. From the functional table, derive the logical expression for the output in terms of the data input and the select inputs.
c. Using the derived expression, implement 4: 1 Mux using logic gates and verify its functional table.

DEMUX

a. Write the functional table for 1: 4 De- MUX (as shown in Fig. 5.2)
b. From the functional table, derive the logical expression for the output in terms of the data input and the select inputs.

c. Using the derived expression, implement 1: 4 De- Mux using logic gates and verify its functional table.

5.5 Pre-lab Questions

1) Multiplexer is also called a data selector. Why?
2) A certain multiplexer can switch one of 32 data inputs to its output. How many select inputs does this MUX have?
3) Implement a 4:1 using 2:1 MUX only?
4) Implement a 1:4 De-Mux using 1:2 De-mux only?
5) Implement a 2-input NAND function using suitable multiplexer?

5.6 Result

Thus the 4:1 MUX and 1:4 De-MUX were constructed and their functional tables were verified.

5.7 Post Lab questions

1. Draw the block diagram of 8x1 MUX using two 4x1 MUX and one 2x1 MUX and draw its function table.
2. Design a Combinational Circuit using Full Adder and Multiplexer as a building blocks to implement
    a. 4-bit Adder/Subtractor, ADD when S=0 and SUB when S=1.
    b. Multiply a 4-bit unsigned no. by 2 when S=0 and transfer zero to output when S=1.
3. Implement the following Boolean Function $F(x,y,z) = \sum m(1,2,5,7)$ using Multiplexer.
4. Draw the block diagram of 1x4 DeMUX using 1x2 DeMUX. Draw its truth table.

Lab 6: DESIGN OF CODE CONVERTERS

6.1 Aim:

1. To design and verify four bit Binary to Gray, Gray to Binary Number converter circuit.

6.2 Hardware Requirement

a. Equipments - Digital IC Trainer Kit
b. Discrete Components – 74LS86 Quad 2-Input EX-OR gate

6.3 Theory

The conversion from one code to another is common in digital systems. Sometimes the output of a system is used as the input to the other system.
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. The bit combination assigned to binary code to gray code. Since each code uses four bits to represent a decimal digit. There are four inputs and outputs. Gray code is a non-weighted code. The input variable are designated as B3,B2,B1,B0 and the output variable are designated as C3,C2,C1,Co.from the truth table, combinational circuit is designed. The Boolean functions are obtained from K-Map for each output variable.

Binary-to-Gray code conversion:

1. MSB Gray code = MSB Binary code
2. From left to right, add each adjacent pair of binary code bits to get the next Gray code bit. Discard carries

Example: Consider the decimal number 68.
(68)10 = (1000100)2
Binary code : 1 0 0 0 1 0 0
Gray code : 1 1 0 0 1 1 0

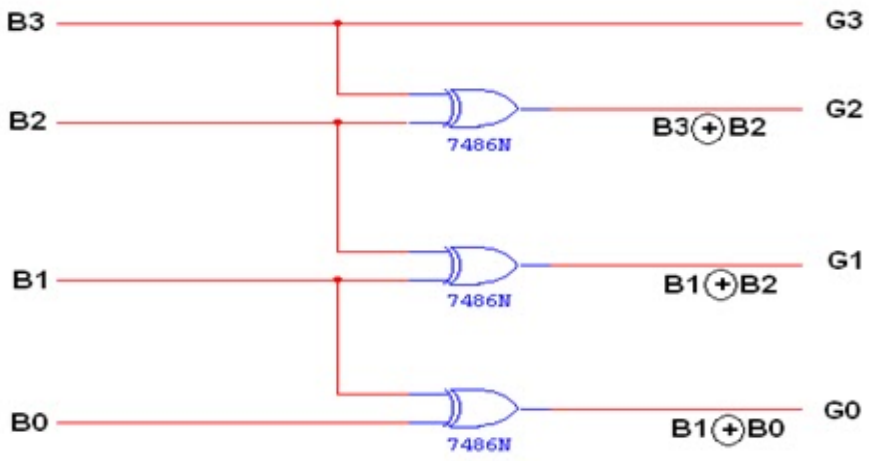
Gray-to-binary code conversion:

1. MSB binary code = MSB Gray code
2. Add each binary code bit generated to the Gray code bit in the next adjacent position. Discard carries

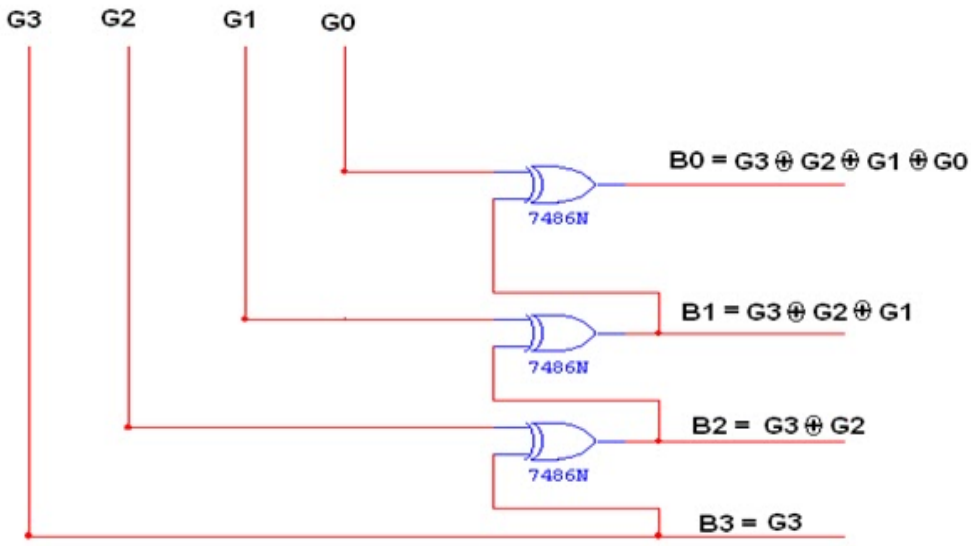
Example: Consider the decimal number 68.
(68)10 = (1000100)2
Gray code : 1 1 0 0 1 1 0
Binary code : 1 0 0 0 1 0 0

The logic equations for Gray to binary code conversion

B0 = G3 ⊕ G2 ⊕ G1 ⊕ G0
B1 = G3 ⊕ G2 ⊕ G1
B2 = G3 ⊕ G2
B3 = G3
Binary to gray code conversion
G0 = B0 ⊕ B1
G1 = B1 ⊕ B2
G2 = B2 ⊕ B3
G3 = B3



Logic diagram for binary to Gray code converter

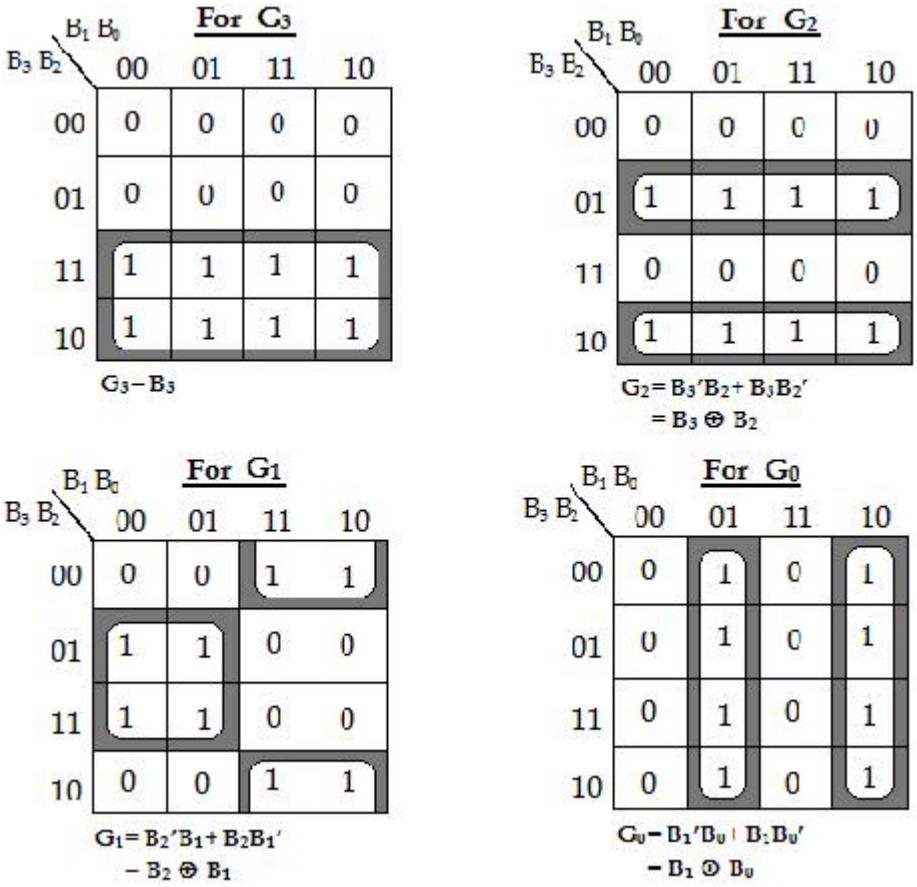


Logic diagram for gray to binary code converter

6.4 Lab Procedure

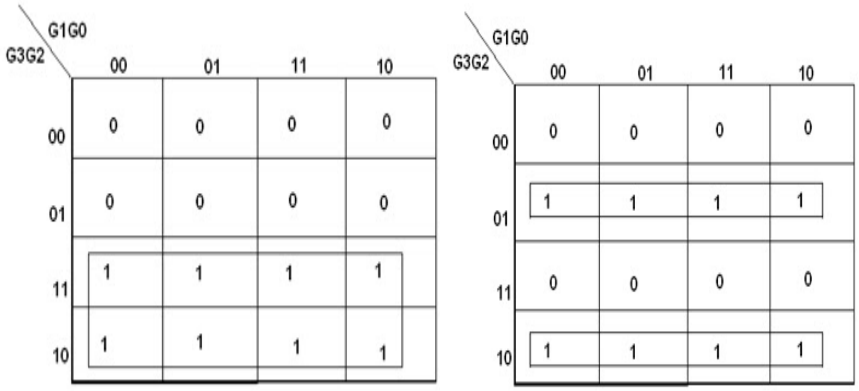
1 Using the derived expression, implement binary to gray and Gray to binary code convertor using logic gates and verify its functional table.
2 In the case of gray to binary conversion, the inputs G0, G1, G2 and G3 are given at respective pins and outputs B0, B1, B2 and B3 are taken for all the 16 combinations of inputs.
3 In the case of binary to gray code conversion, the inputs B0, B1, B2 and B3 are given at respective pins and outputs G0, G1, G2 and G3 are taken for all the 16 combinations of inputs.

6.4.1 K-Map for Gray to binary code convertor

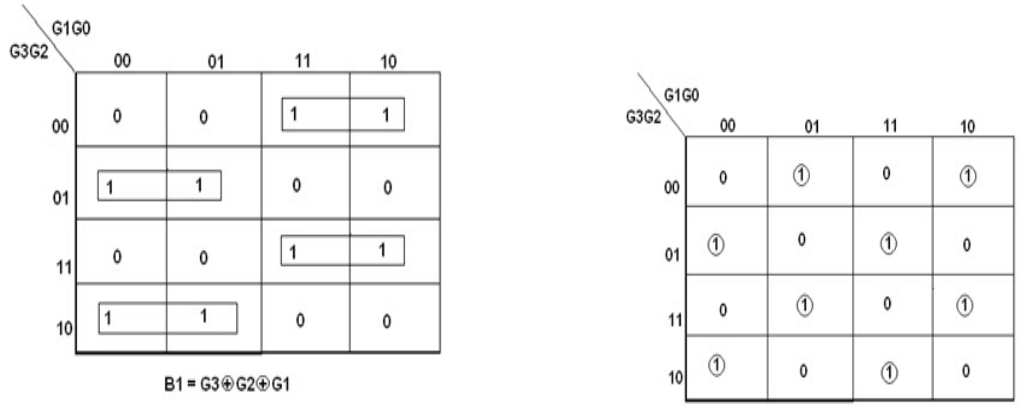


6.4.2 K-Map for Gray to binary code convertor

K-MAP for B3 &B2



K-MAP for B1& B0



Truth table for Binary to Gray code

Decimal	Binary Code (input)	Gray Code (output)
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Truth table for Gray to Binary code

Gray Code Input				Binary Code Output			
G3	G2	G1	G0	B3	B2	B1	B0
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
1	0	0	0	1	1	1	1

6.5 Result: Binary to Gray and Gray to Binary code converter are designed and verified.

6.7 Prelab

1. What do you mean by code conversion? What are the applications of code conversion?
2. What are the advantages of gray code?
3. Determine the Gray code for (a) 37 and (b) 128.
4. Derive the Boolean expression for four bit Excess 3 code to Binary code converter
5. Derive the Boolean expression for four bit BCD to Gray code converter
6. Convert gray code 101011 into its binary equivalent.

6.8 Post lab

1. Design the BCD to Binary code converter circuit

Lab: 7 Combination Circuits using standard ICs

7.1 Aim

To implement the combinational circuits using standard ICs.

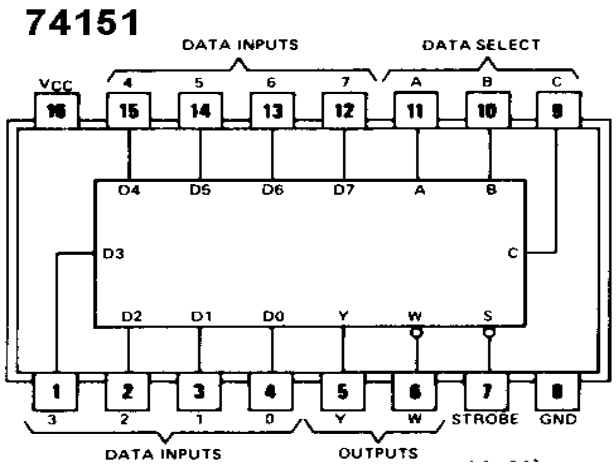
7.2 Apparatus Required

1. IC74151- 8:1 Multiplexer
2. IC74LS138-3:8 Decoder
3. IC trainer kit
4. Connecting wires

7.3 Procedure

7.3.3.1 Verify IC74151 – 8:1 Multiplexer IC.

Pin Diagram:



IC Description:

74151 is a 8 line-to-1 line multiplexer. It has the schematic representation shown in Fig 1. Selection lines S2, S1 and S0 select the particular input to be multiplexed and applied to the output.

Strobe S acts as an enable signal. If strobe =1, the chip 74151 is disabled and output y = 0. If strobe = 0 then the chip 74151 is enabled and functions as a multiplexer. Table 1 shows the multiplex function of 74151 in terms of select lines.

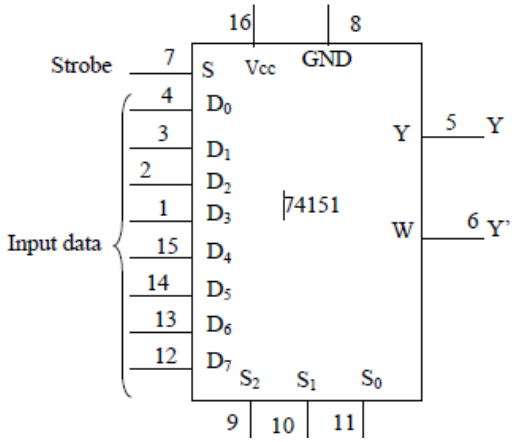


Fig1. IC74151 Pin Description

Strobe	Select Lines			Output
S	S_2	S_1	S_0	Y
1	X	X	X	0
0	0	0	0	D0
0	0	0	1	D1
0	0	1	0	D2
0	0	1	1	D3
0	1	0	0	D4
0	1	0	1	D5
0	1	1	0	D6
0	1	1	1	D7

Table1. Multiplexer Function

1. **Implement F(A,B,C,D) = Σm (0, 1, 4, 5, 8,10,12, 14, 15) using IC74151.**

   i) Take B, C and D as S0, S1 and S2

   ii)

	D0	D1	D2	D3	D4	D5	D6	D7
A’	(0)	(1)	2	3	(4)	(5)	6	7
A	(8)	9	(10)	11	(12)	13	(14)	15
	0	A’	A	0	1	A’	A	0

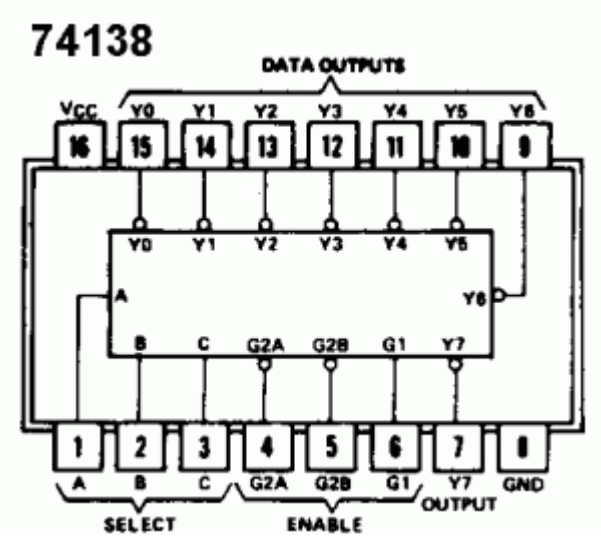
   iii) Apply A input to D2 and D6 and apply A’ input to D1 and D5.
   iv) Give +5V to D4 and Connect D0 and D7 to ground.
   v) Verify the following truth table.

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

7.3.3.2 Verify 3x8 Decoder using IC74LS138.

Decoder is the combinational circuit which contains 'n' input lines to 2^n output lines. The decoder is used for converting the binary code into the octal code. The IC74138 is the 3*8 decoder which contains three inputs and eight outputs and also three enables out of them two are active low and one is active high. Decoders are used in the circuit where required to get more outputs than that of the inputs which also used in the chip designing process for reducing the IC chip area.

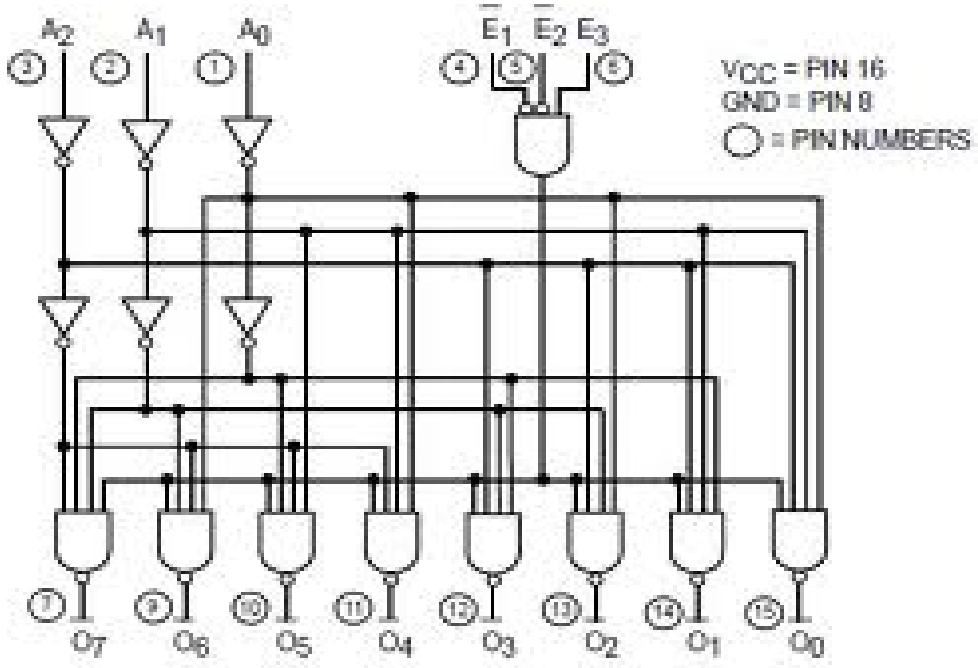
Pin Diagram:



i) Connect the inputs to the corresponding pins.
ii) Verify the following truth table.

Inputs						Outputs							
$\overline{E_1}$	$\overline{E_2}$	E_3	A_0	A_1	A_2	$\overline{O_0}$	$\overline{O_1}$	$\overline{O_2}$	$\overline{O_3}$	$\overline{O_4}$	$\overline{O_5}$	$\overline{O_6}$	$\overline{O_7}$
H	X	X	X	X	X	H	H	H	H	H	H	H	H
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	L	X	X	X	H	H	H	H	H	H	H	H
L	L	H	L	L	L	L	H	H	H	H	H	H	H
L	L	H	H	L	L	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	L	H	H	H	H	H
L	L	H	H	H	L	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	L	H	H
L	L	H	L	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	L

Function Diagram



7.4 Prelab Questions

1. What is meant by combinational circuits?
2. What are the applications of Multiplexer?
3. What are the difference between Decoder and Demultiplexer?
4. Give some examples for standard combinational circuit ICs.

7.5 Result

Thus the 8:1Multiplexer and 3:8 decoder are verified using standard combinational ICs.

7.6 Post Lab Questions

1. Implement $F(W,X,Y,Z) = \Sigma m(1,2,5,7,9,12,15)$ using suitable IC74151.
2. Design 4-bit parity generator using Multiplexer IC.
3. What is the role of an encoder in communication?
4. Implement the function $F(A,B,C,D) = \sum m(1,3,4,11,12,13,14,15)$ using MUX. Draw its truth table and its logic symbol.

Lab 8: Design of 3-bit Synchronous Counters

8.1 Aim

To design and verify the truth table for 3-bit synchronous up/down counter.

8.2 Hardware Requirement

Equipment : Digital IC Trainer Kit

Discrete Components :

IC 7473 Dual JK Flip Flop
74LS08 Quad 2 input AND gate
74LS32Quad 2 input OR gate
74LS04 Hex 1 input NOT gate

8.3 Theory

Circuits for counting events are frequently used in computers and other digital systems. Since a counter circuit must remember its past states, it has to possess memory. The number of flip flops used and how they are connected determine the number of states and the sequence of the states that the counter goes through in each complete cycle.

Counters can be classified into two broad categories according to the way they are clocked:

a. Asynchronous (Ripple) Counters - the first flip-flop is clocked by the external clock pulse, and then each successive flip -flop is clocked by the Q or Q' output of the previous flip -flop.
b. Synchronous Counters - all memory elements are simultaneously triggered by the same clock.

Synchronous Counters

In *synchronous counters*, the clock inputs of all the flip-flops are connected together and are triggered by the input pulses. Thus, all the flip-flops change state simultaneously (in parallel). The circuit below is a 3-bit synchronous counter. The J and K inputs of FF0 are connected to HIGH. FF1 has its J and K inputs connected to the output of FF0, and the J and K inputs of FF2 are connected to the output of an AND gate that is fed by the outputs of FF0 and FF1. After the 3rd clock pulse both outputs of FF0 and FF1 are HIGH. The positive edge of the 4th clock pulse will cause FF2 to change its state due to the AND gate.

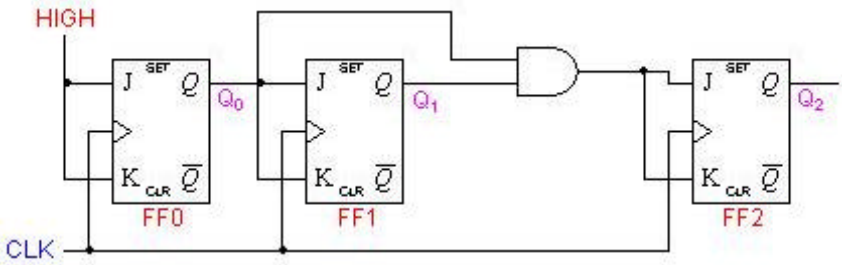


Figure 8.1 Logic diagram of 3-bit Synchronous counter

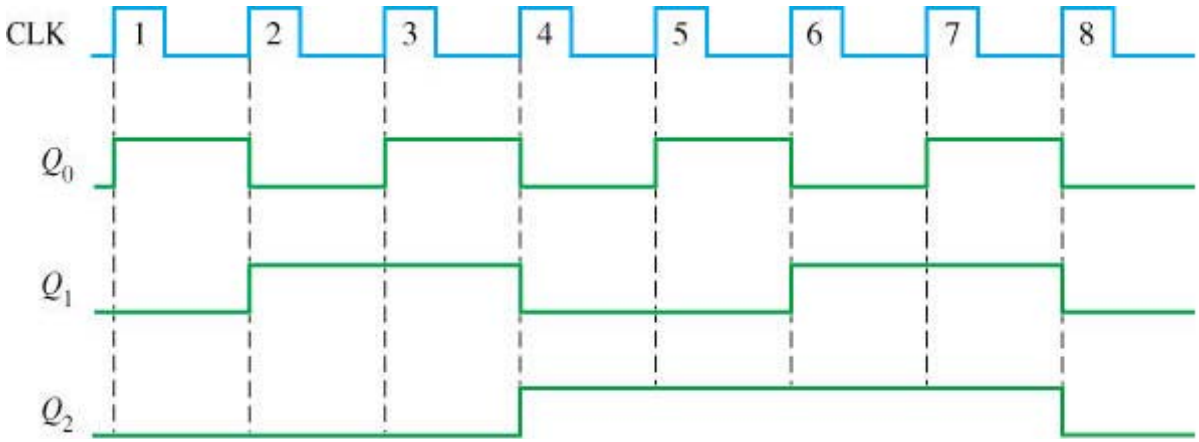


Figure8.2 Timing Diagram of 3-bit Counter

The most important advantage of synchronous counters is that there is no cumulative time delay because all flip -flops are triggered in parallel. Thus, the maximum operating frequency for this counter will be significantly higher than for the corresponding ripple counter.

8.4 Lab Procedure

1. Construct the logic circuit as shown in figure 8.1.
2. Use the up/(down)' input to choose up counter or down counter.
3. Verify the count sequence as given in figure 8.4.

Logic diagram:

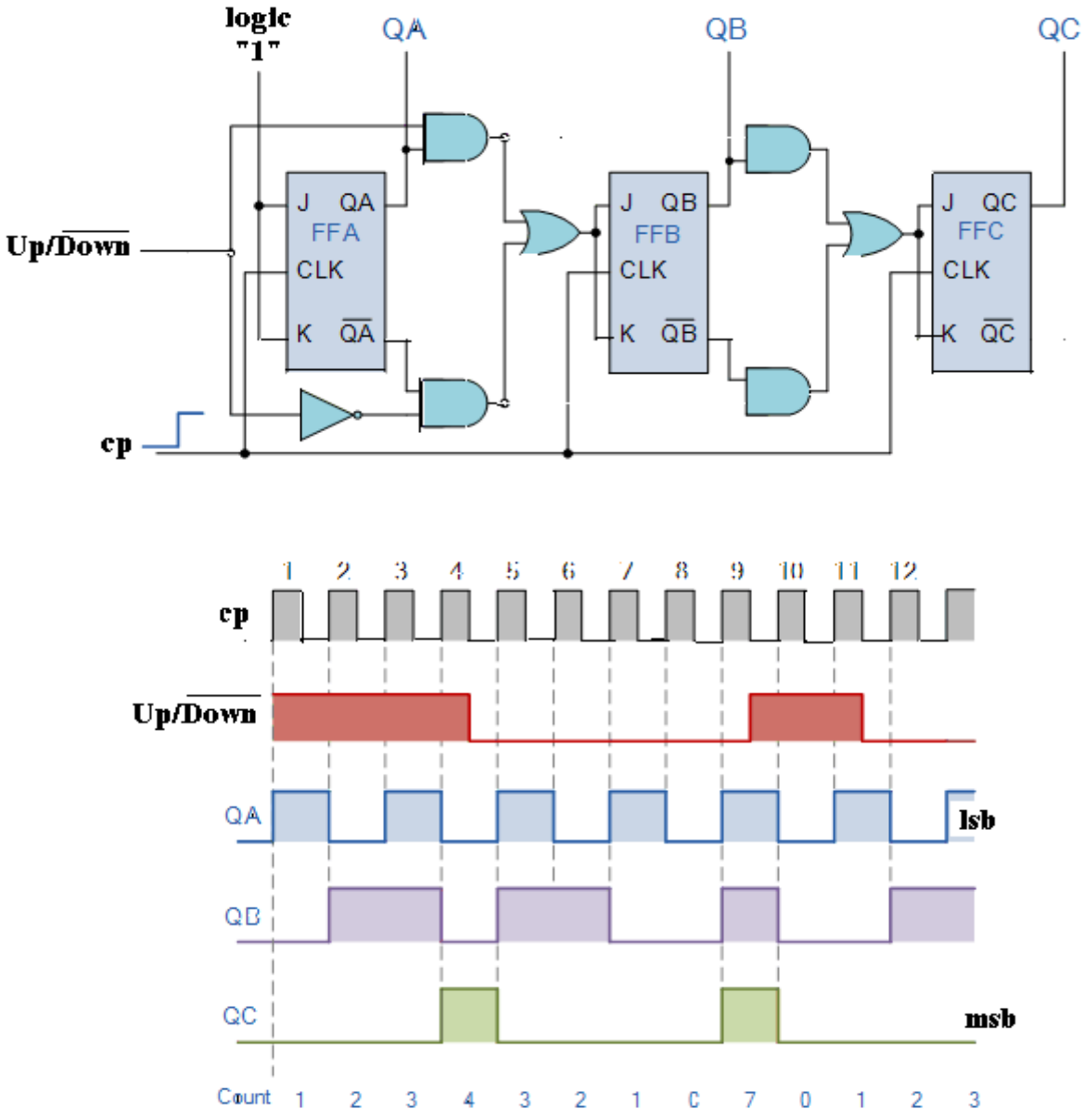
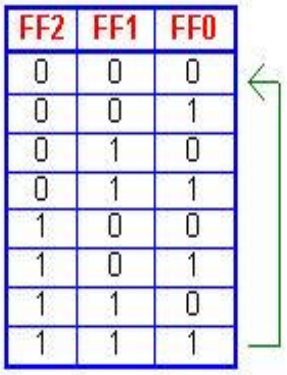


Figure 8.3 3-bit Synchronous Up/Down Counter and Output of Each Flip Flop



FF2	FF1	FF0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Figure 8.4 Count Sequence

8.5 PreLab questions

1. How does synchronous counter differ from asynchronous counter?
2. A 4-bit up/down binary counter is in the DOWN mode and in the 1010 state. On the next clock pulse, to what state does the counter go?
3. How many flip-flops do you require to design Mod-7 counter.
4. Give the Transition table and excitation table of JK Flip flop.

8.6 Result

Thus the 3-bit synchronous up/down counter is designed and verified.

8.7 PostLab questions

1. Draw the state Diagram, state table and Timing Diagram of a 2-bit synchronous counter.
2. Deign a 3-bit Up/Down Gray Code Counter using D Flip-flop
3. Design a 11011 sequence detector using JK flip-flops. Allow overlap.
4. What is decade Counter?

Lab 9: Design of 3-bit Ripple Counters

9.1 Aim

To design and verify the timing diagram of 3 bit Ripple Counter

9.2 Apparatus Required

a. Equipments - Digital IC Trainer Kit
b. Discrete Components - IC7473 Dual JK Flip-flop

9.3 Theory

Asynchronous Counter is sequential circuit that is used to count the number of clock input signal. The output of one flip flop is given as a clock input to another flip-flop, so it is called as Serial Counter.

A ripple counter is an asynchronous counter where only the first flip-flop is clocked by an external clock. All subsequent flip-flops are clocked by the output of the preceding flip-flop. Asynchronous counters are also called ripple-counters because of the way the clock pulse ripples it way through the flip-flops.

The MOD of the ripple counter or asynchronous counter is 2^n if n flip-flops are used. A three-bit asynchronous counter is shown on the below figure . The external clock is connected to the clock input of the first flip-flop (FF0) only. So, FF0 changes state at the falling edge of each clock pulse, but FF1 changes only when triggered by the falling edge of the Q output of FF0 similarly FF2 changes only when triggered by the falling edge of the Q output of FF1. Because of the inherent propagation delay through a flip-flop, the transition of the input clock pulse and a transition of the Q output of FF0 can never occur at exactly the same time. Therefore, the flip-flops cannot be triggered simultaneously, producing an asynchronous operation.

Usually, all the CLEAR inputs are connected together, so that a single pulse can clear all the flip-flops before counting starts. The clock pulse fed into FF0 is rippled through the other counters after propagation delays, like a ripple on water, hence the name Ripple Counter.

.

Logic diagram

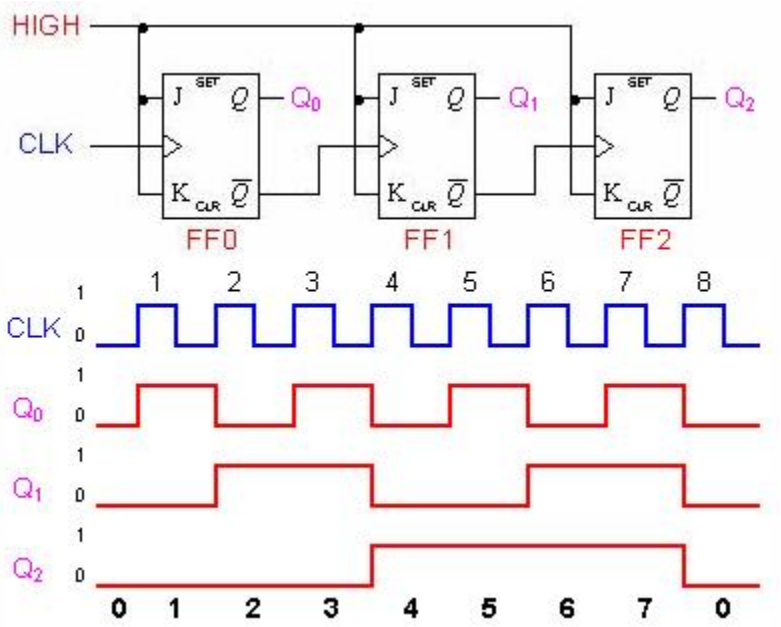


Figure 9.1 3 bit Ripple counter with Timing Diagram

Truth Table

FF2	FF1	FF0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Figure 9.2 Count Sequence

9.4 Prelab questions

1. What do you mean by Glitch?
2. How many flip-flops are required to produce a divide-by-64 device?
3. Why Asynchronous counter is called as Ripple Counter?
4. What do you mean by synchronous reset and asynchronous reset?
5. What is the use of Preset input?
6. What is use of Ring and Johnson's Counter?

9.5 Lab Procedure

1. Construct the logic circuit as shown in Figure 9.1.
2. Verify the count sequence as given in figure 9.2.

9.6 Result:

Thus the timing diagram and state diagram of 3 bit asynchronous Ripple counter was verified.

9.7 Postlab questions

1. Draw the logic diagram of Mod 12 Asynchronous Counter and its timing diagram.
2. Design a 4-bit frequency divider.
3. Design a sequential circuit that is used to generate the timing signals with a combination of Shift register and a decoder.
4. What is state table?

Lab10: Design of MOD-N counters

10.1 Aim

The purpose of this experiment is to introduce the design of Mod-N Counter and to implement it using suitable Flip-flops.

10.2 Hardware Requirement

Equipment : Digital IC Trainer Kit
Discrete Components : IC 7473 Dual JK Flip Flop
IC 7400 NAND Gate

10.3 Theory:

Circuits for counting events are frequently used in computers and other digital systems. Since a counter circuit must remember its past states, it has to possess memory. The number of flip flops used and how they are connected determine the number of states and the sequence of the states that the counter goes through in each complete cycle.

Counters can be classified into two broad categories according to the way they are clocked:

1. Asynchronous (Ripple) Counters - the first flip-flop is clocked by the external clock pulse, and then each successive flip -flop is clocked by the Q or Q' output of the previous flip -flop.
2. Synchronous Counters - all memory elements are simultaneously triggered by the same clock.

A mod N counter is a counter that has N states. Its output frequency is f/N. A counter which is reset at the fifth clock pulse is called Mod 5 counter or Divide by 5 counter. The circuit diagram of Mod 5 counter is shown in the figure. This counter contains three JKMS flip-flop.

Mod 5 Asynchronous Counter:-

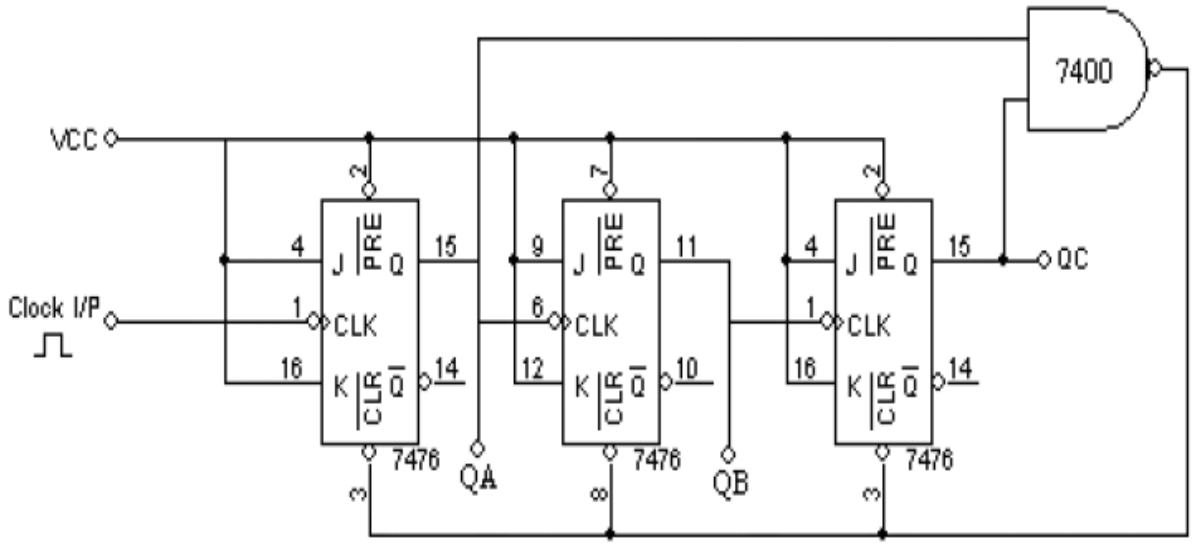


Figure 10.1: Mod 5 Asynchronous Counter

A 3 bit binary counter is normally counting from 000 to 111. The actual output of a 3 bit binary counter at the fifth clock pulse is 101. A two input NAND gate is used to make a Mod 5 counter.

The outputs of the first and third flip flops (QA and QC) are connected to the input of the give NAND gate, and its output is connected to the RESET terminal of the counter,

Hence the counter is reset at the fifth clock pulse, which produces the output QC,QB,QA as 000. It is called divide by 5th counter or mod 5 counter.

Mod 5 Asynchronous counter			
Clock	QC	QB	QA
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	0	0	0

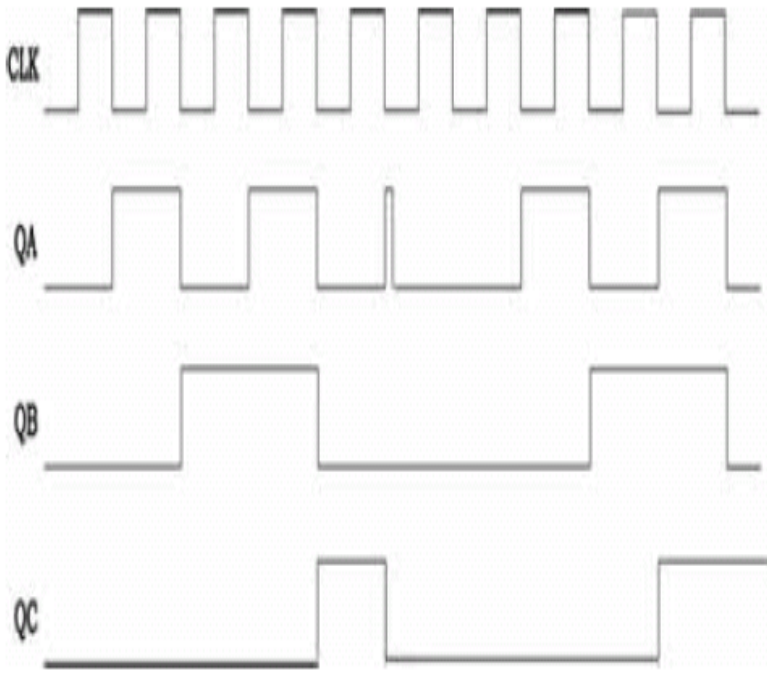


Figure 10.2: Truth Table and Timing Diagram

10.4 Lab Procedure

1. Connections are made as per circuit diagram.
2. Clock pulses are applied one by one at the clock I/P and the O/P is observed at QA, QB & QC for IC 7476.
3. Truth table is verified.

10.5 Prelab questions

1. Which flipflop is suitable for counter? Why?
2. Draw the timing diagrams for mod 6 counter.

.

10.6 Result

Thus the Mod-5 counter is designed and verified.

10.7 PostLab questions

1. Draw the state Diagram, state table and Timing Diagram of a 2-bit synchronous counter.
2. Design a modulus seven synchronous counter that can count 0, 3, 5, 7, 9, 11, and 12 using D flip-flop.

.

Lab 11: Design of Shift Registers and Shift Register Counters

11.1 Aim

The purpose of this experiment is to introduce the design of Shift Registers. The student should also be able to design n-bit shift register.

11.2 Hardware Requirement

a. Equipments - Digital IC Trainer Kit

b. Discrete Components - IC7474 Dual JK Flip-flop

11.3 Theory

Shift registers are a type of sequential logic circuit, mainly for storage of digital data. They are a group of flip-flops connected in a chain so that the output from one flip-flop becomes the input of the next flip-flop. Most of the registers possess no characteristic internal sequence of states. All the flip-flops are driven by a common clock, and all are set or reset simultaneously. The basic types of shift registers are

1. Serial In - Serial Out
2. Serial In - Parallel Out
3. Parallel In - Serial Out
4. Parallel In - Parallel Out
5. Bidirectional shift registers

11.3.1 Serial In - Serial Out Shift Registers

A basic four-bit shift register can be constructed using four D flip-flops, as shown below. The operation of the circuit is as follows. The register is first cleared, forcing all four outputs to zero. The input data is then applied sequentially to the D input of the first flip-flop on the left (FF0). During each clock pulse, one bit is transmitted from left to right. Assume a data word to be 1001. The least significant bit of the data has to be shifted through the register from FF0 to FF3. Assume a data word to be 1001. The least significant bit of the data has to be shifted through the register from FF0 to FF3.

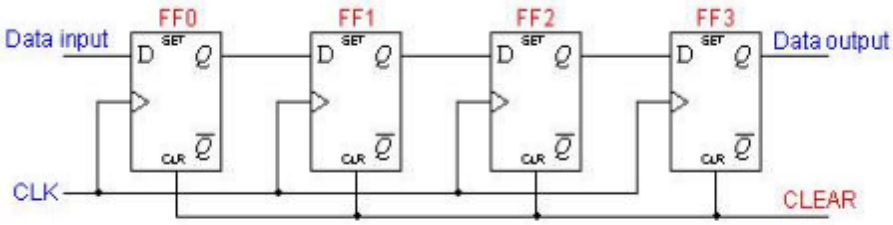


Figure 9.1 Logic Diagram - SISO

11.3.2 Parallel In - Parallel Out Shift Registers

For parallel in - parallel out shift registers, all data bits appear on the parallel outputs immediately following the simultaneous entry of the data bits. The following circuit is a four-bit parallel in - parallel out shift register constructed by D flip-flops.

The D's are the parallel inputs and the Q's are the parallel outputs. Once the register is clocked, all the data at the D inputs appear at the corresponding Q outputs simultaneously.

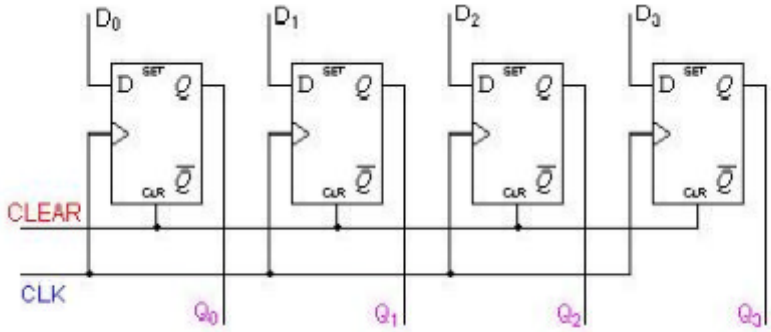


Figure 11.2 Logic Diagram – PIPO

Shift Register Counter

Two of the most common types of shift register counters are Ring counter and the Johnson counter. They are basically shift registers with the serial outputs connected back to the serial inputs in order to produce particular sequences. These registers are classified as counters because they exhibit a specified sequence of states.

Johnson counters are a variation of standard ring counters, with the inverted output of the last stage fed back to the input of the first stage. They are also known as twisted ring counters. An n-stage Johnson counter yields a count sequence of length 2n, so it may be considered to be a mod-2n counter. The circuit above shows a 4-bit Johnson counter. The state sequence for the counter is given in the table as well as the animation on the left.

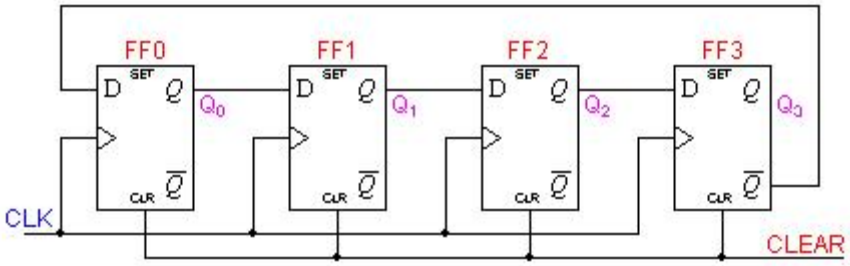


Figure 11.3 Logic Diagram with Truth Table

Clock Pulse	Q3	Q2	Q1	Q0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	1	1	1
4	1	1	1	1
5	1	1	1	0
6	1	1	0	0
7	1	0	0	0

Figure 11.4 Truth Table

11.4 Prelab questions

1. Redraw the above logic diagrams using JK flip-flops.
2. How many flip-flops are required to store the data 1001?
3. How many clock pulses are required to enter a byte of data serially into an 8-bit shift register?
4. Define shift register counters.
5. What is bidirectional and unidirectional shift register?
6. Explain the function of SHIFT/LOAD input in PISO shift register.

11.5 Result

Thus, design and verification of shift registers are done successfully.

11.6 PostLab questions

1. Draw the Logic diagram of Universal shift register.
2. Write the applications of Shift registers.
3. Draw the State Table and Block diagram of a Serial Adder.

Lab 12: Implementation of sequential logic functions using standard IC's

12.1 Aim

To implement and verify the functional table of sequential logic functions using IC.

12.2 Hardware Requirement

a. Equipments - Digital IC Trainer Kit

b. Discrete Components - 74LS194

12.3 Theory

The register capable of shifting both right and left is called a bidirectional shift register.

The SN54/74LS194A is a High Speed 4-Bit Bidirectional Universal Shift Register. As a high speed multifunctional sequential building block, it is useful in a wide variety of applications.

Like a parallel register it can load and transmit data in parallel. Like shift registers it can load and transmit data in serial fashions, through left shifts or right shifts. In addition, the universal shift register can combine the capabilities of both parallel and shift registers to accomplish tasks that neither basic type of register can perform on its own. For instance, on a particular job a universal register can load data in series (e.g. through a sequence of left shifts) and then transmit/output data in parallel.

Universal shift registers, as all other types of registers, are used in computers as memory elements. Although other types of memory devices are used for the efficient storage of very large volume of data, from a digital system perspective when we say computer memory we mean registers. In fact, all the operations in a digital system are performed on registers. Examples of such operations include multiplication, division, and data transfer.

This device can operate in four distinct modes, determined by the values at the control inputs *S*1 and *S*0: hold data (*S*1, *S*0 = 00), shift right *Q*0 toward *Q*3 (*S*1, *S*0 = 01), shift left *Q*3 toward *Q*0 (*S*1, *S*0 = 10), and parallel load from the 0,*1*, 2, 3 inputs (*S*1, *S*0 = 11). In addition, the register has an active low (asynchronous) reset signal CLR and two serial shift inputs DSL and DSR.

The parallel load takes place when *S*1 and *S*0 are both high and a rising edge arrives at the clock input. At the same time, the value at input P0 replaces the contents of the *Q*0 flip-flop, P1 replaces *Q1*, and so on. This is called a synchronous load because it takes place in response to a clock event.
*S*1 low and *S*0 high indicate a right shift. On the rising edge of the clock, the value on the DSR input replaces *Q0*, *Q*0 replaces *Q1*, *Q1* replaces *Q*2, and *Q2* replaces *Q3*. The old value at *Q3* is lost. We can construct a right circular shifter by wiring the *Q*4 output to the DSR input.

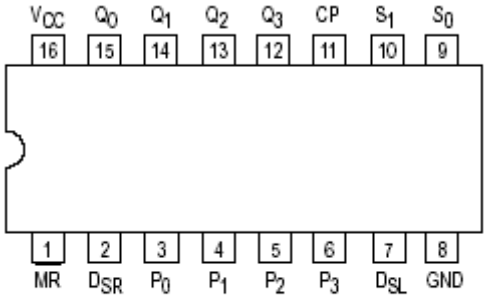
*S*1 high and *S*0 low specify a left shift. In this case, DSL replaces *Q3*, *Q3* replaces *Q*2, *Q*2 replaces *Q1*, and *Q1* replaces *Q0*, all on the rising edge of the clock. We can construct a left circularshifterbywiring*Q*0toDSL.

*S*1 and *S*0 both low tell the shift register to hold its state. The outputs do not change even though the clock signal undergoes a low-to-high transition.

A universal shift register is an integrated logic circuit that can transfer data in three different modes. Like a parallel register it can load and transmit data in parallel. Like shift registers it can load and transmit data in serial fashions, through left shifts or right shifts. In addition, the universal shift register can combine the capabilities of both parallel and shift registers to accomplish tasks that neither basic type of register can perform on its own. For instance, on a particular job a universal register can load data in series (e.g. through a sequence of left shifts) and then transmit/output data in parallel.

Universal shift registers, as all other types of registers, are used in computers as memory elements. Although other types of memory devices are used for the efficient storage of very large volume of data, from a digital system perspective when we say computer memory we mean registers. In fact, all the operations in a digital system are performed on registers. Examples of such operations include multiplication, division, and data transfer.

PIN Diagram



74194

PIN N**ames**:-

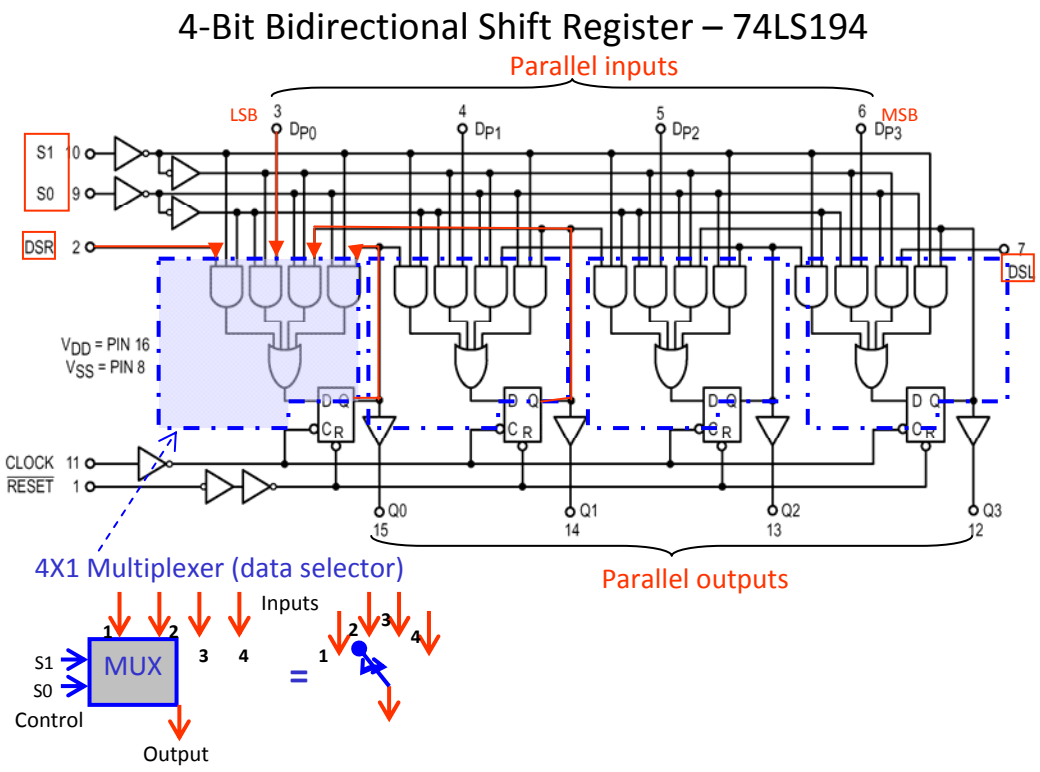
S0, S1 Mode Control Inputs
P0–P3 Parallel Data Inputs
D_{SR} Serial (Shift Right) Data Input
D_{SL} Serial (Shift Left) Data Input
CP Clock (Active HIGH Going Edge) Input
MR Master Reset (Active LOW) Input
Q0–Q3 Parallel Outputs

MODE SELECT — TRUTH TABLE

OPERATING MODE	INPUTS						OUTPUTS			
	MR	S_1	S_0	D_{SR}	D_{SL}	P_n	Q_0	Q_1	Q_2	Q_3
Reset	L	X	X	X	X	X	L	L	L	L
Hold	H	l	l	X	X	X	q_0	q_1	q_2	q_3
Shift Left	H	h	l	X	l	X	q_1	q_2	q_3	L
	H	h	l	X	h	X	q_1	q_2	q_3	H
Shift Right	H	l	h	l	X	X	L	q_0	q_1	q_2
	H	l	h	h	X	X	H	q_0	q_1	q_2
Parallel Load	H	h	h	X	X	P_n	P_0	P_1	P_2	P_3

L = LOW Voltage Level
H = HIGH Voltage Level
X = Don't Care
l = LOW voltage level one set-up time prior to the LOW to HIGH clock transition
h = HIGH voltage level one set-up time prior to the LOW to HIGH clock transition
P_n (q_n) = Lower case letters indicate the state of the referenced input (or output) one set-up time prior to the LOW to HIGH clock transition

Logic diagram



12.4 Lab procedure

- Connect the 74194 to the breadboard with appropriate inputs and outputs.
- Verify the functionality of shift register.
- Verify the Truth Table and observe the outputs

12.5 Prelab questions

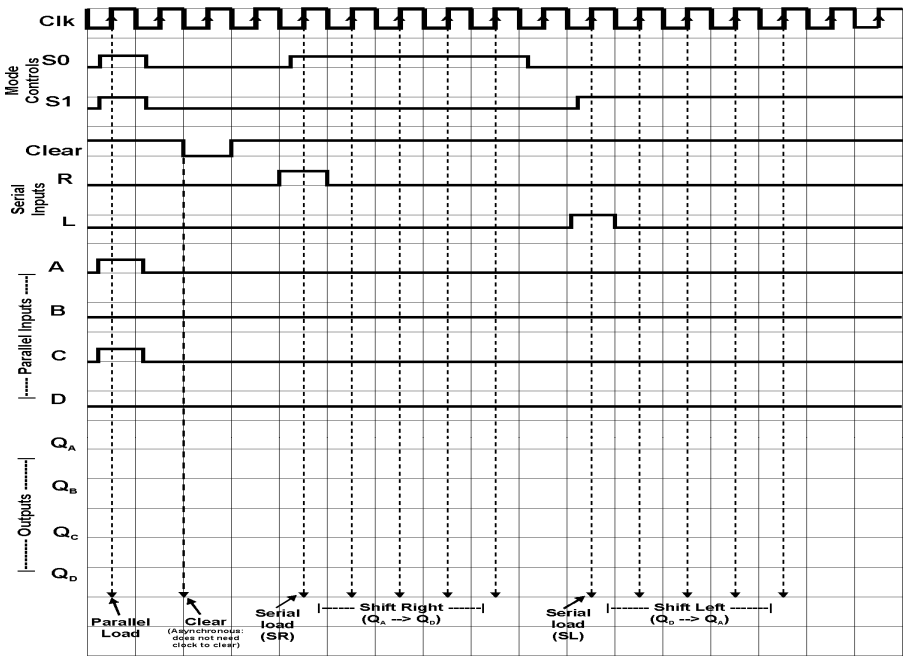
1. What is Universal shift Register?
2. What is the necessity for sequence generation?
3. What are the operations performed by a universal shift register?
4. Explain how to convert serial data to parallel and parallel data to serial.
5. What type of register is needed?
6. What is the Difference b/w shift register and universal shift register**?**

12.6 Result

The function of sequential logic circuit is verified.

12.7 Post lab questions

1. From which of the outputs Q can we get the data serial output from the 74194 register? And how many pulses do we need?
2. Use two ICs of the 74194 to build an 8-bit bidirectional register, and verify the functional table for the resultant register.
3. Connect the 74194 to the breadboard with appropriate input and outputs and Fill the timing diagram given below.



Lab 13: Simulation Experiments using Logisim

Goals

To understand how to design and debug combinatorial circuits using the Logisim logic simulator.

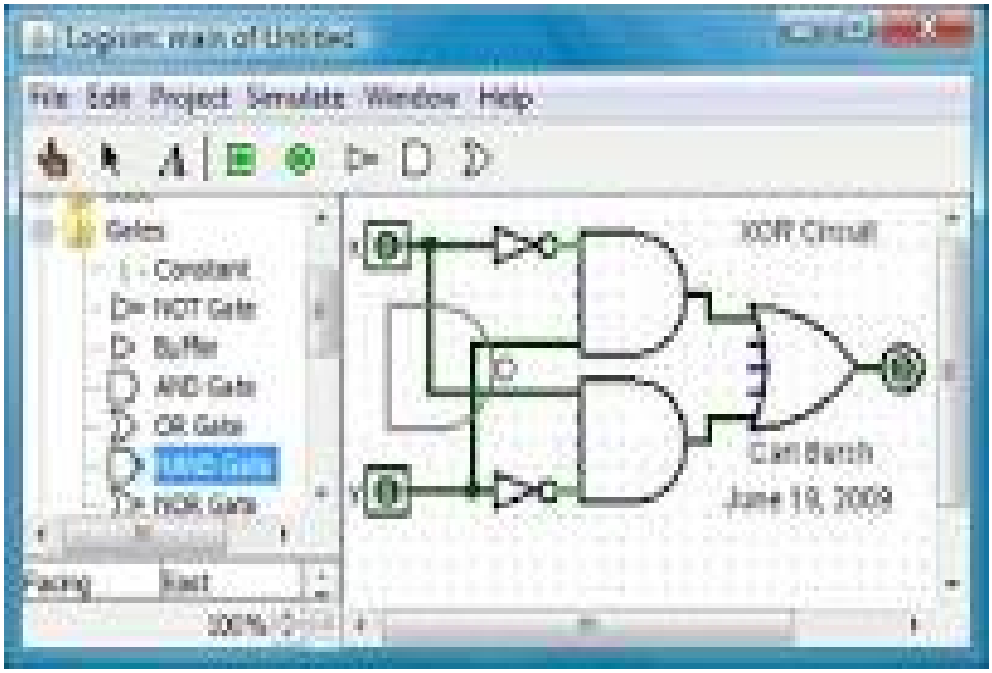
Procedure

- Start up Logisim and create a new circuit. Save this file as "lab13part1.circ". Then, create a circuit given below.
- Square boxes are input pins. Their values are being used to compute a value.
- Circle boxes are output pins, as their values are being computed by the circuit and output to circuitry that wants the result of the circuit.
- Triangles are not-gates, as they invert whatever their input is.
- Green lines are wires. Bright green wires are 'on' (true, 1). Dull green wires are 'off' (false, 0). Blue wires are unconnected. We often have blue lines (e.g., when a 5-input AND gate only has 2 inputs connected, the blue input will be ignored).

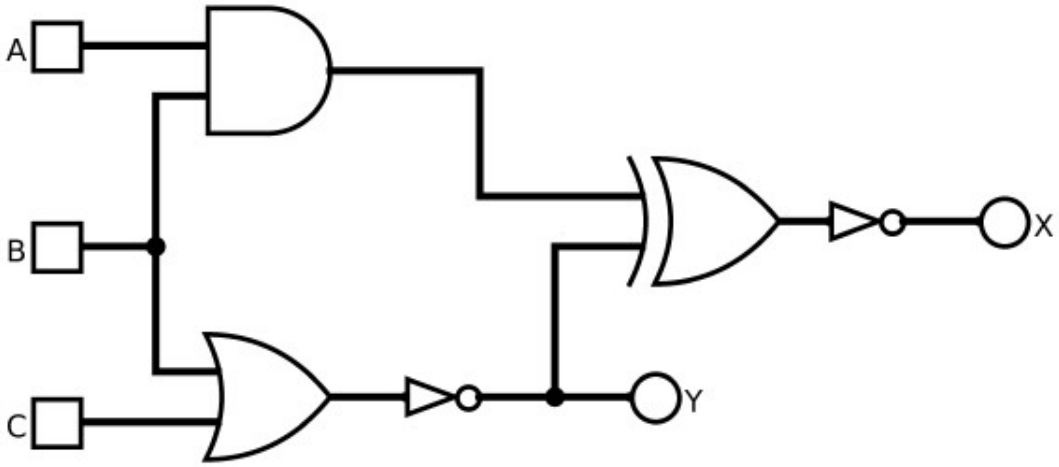
- All components are available from the main tool bar (below the "File" menu).

The poke tool (the hand icon) lets you change the values of input pins, to test different inputs. The arrow tool lets you add, select, and manipulate wires. You can "undraw" wires to shorten them, or use the delete key to remove the selected wire. Be sure to add the labels You should never ever have red wires. These indicate an error, such as two outputs being connected together.

- By changing the values of the input pins, fill out the following truth table.



1) Design the circuit shown below:



2) Find truth table values by manually toggling the input values and examining the output values. This is done in Logisim by selecting the hand icon and then clicking on each input. Fill in the rest of the truth table for the circuit manually:

A	B	C	Y	X
0	0	0	1	
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		0

3) Next, generate the complete truth table with Logisim by selecting Project->Analyze Circuit from the Project menu and then viewing the table tab. Does it match the one you filled in above? If not, check your answers or your circuit.

4) Save your circuit (File->Save) as r4.circ in your R4 directory. Export an image of the circuit (File->Export Image) as a .png file called r4.png and save it to the R4 directory.

5) Build a circuit that implements the truth table below, and verify it by checking against the truth table made with Logisim. What kind of circuit does this represent?

A	B	W	X	Y	Z
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1