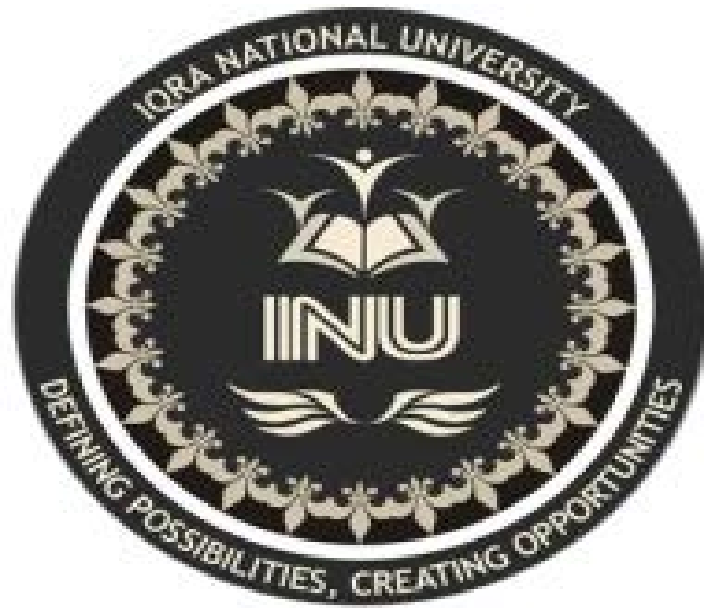


# Digital Logic Design Lab



Department of Computer Science

IQRA NATIONAL UNIVERSITY, PESHAWAR

---

# Contents

---

<b>0</b>	<b>Course Learning Outcomes (CLOs)</b> . . . . .	<b>1</b>
<b>1</b>	<b>Lab 1: Verification of Gates</b>	<b>2</b>
1.1	Aim . . . . .	2
1.2	Objectives: . . . . .	2
1.3	Apparatus/Equipment Required: . . . . .	2
1.3.1	Pin Diagram . . . . .	3
1.4	Theory . . . . .	4
1.4.1	AND Gate . . . . .	4
1.4.2	OR Gate . . . . .	5
1.4.3	NOT Gate . . . . .	6
1.4.4	NAND Gate . . . . .	6
1.4.5	NOR Gate . . . . .	7
1.4.6	Exclusive-OR (XOR Gate) . . . . .	7
1.4.7	Exclusive-NOR (XNOR) . . . . .	7
1.5	Procedure: . . . . .	8
1.6	Observation Table: . . . . .	8
1.7	Results and Analysis: . . . . .	8
1.8	Conclusion: . . . . .	9
<b>2</b>	<b>Lab 2: Half Adder</b>	<b>10</b>
2.1	Aim . . . . .	10
2.2	Objectives . . . . .	10
2.3	Apparatus Required . . . . .	10
2.4	Pin Diagram: . . . . .	10
2.5	Theory . . . . .	10
2.6	Procedure: . . . . .	11
2.7	Observation table . . . . .	12
2.8	Results and Analysis . . . . .	12
2.9	Conclusion: . . . . .	12
<b>3</b>	<b>Lab 3: Full Adder</b>	<b>13</b>
3.1	Aim . . . . .	13
3.2	Objective: . . . . .	13
3.3	Apparatus Required . . . . .	13
3.4	Pin Diagram . . . . .	14
3.5	Theory . . . . .	14

3.6	Procedure: . . . . .	14
3.7	Observation Table . . . . .	15
3.8	Results And Analysis . . . . .	16
3.9	Conclusion: . . . . .	16
<b>4</b>	<b>Lab 4: Half Subtractor</b>	<b>17</b>
4.1	Aim . . . . .	17
4.2	Objective: . . . . .	17
4.3	Apparatus Requirement: . . . . .	17
4.4	Pin Diagram: . . . . .	18
4.5	Theory: . . . . .	18
4.6	Procedure: . . . . .	18
4.7	Observation Table: . . . . .	19
4.8	Results and Analysis: . . . . .	20
4.9	conclusion: . . . . .	20
<b>5</b>	<b>Lab 5: Full Subtractor</b>	<b>21</b>
5.1	Aim . . . . .	21
5.2	Objective: . . . . .	21
5.3	Apparatus Requirement: . . . . .	21
5.4	Pin Diagram: . . . . .	22
5.5	Theory: . . . . .	22
5.6	Procedure: . . . . .	23
5.7	Observation Table: . . . . .	24
5.8	Results and Analysis: . . . . .	24
5.9	Conclusion: . . . . .	24
<b>6</b>	<b>Lab 6: Multiplexer</b>	<b>25</b>
6.1	Aim . . . . .	25
6.2	Objective: . . . . .	25
6.3	Apparatus Required: . . . . .	25
6.4	Apparatus Requirement: . . . . .	25
6.5	Pin Diagram: . . . . .	26
6.6	Theory: . . . . .	26
6.7	Procedure: . . . . .	27
6.8	Observation Table: . . . . .	28
6.9	Results and Analysis: . . . . .	28
6.10	Conclusion: . . . . .	28

<b>7</b>	<b>Lab 7: De-Multiplexer</b>	<b>29</b>
7.1	Aim . . . . .	29
7.2	Objective: . . . . .	29
7.3	Apparatus Required: . . . . .	29
7.4	Pin Diagram: . . . . .	29
7.5	Theory: . . . . .	29
7.6	Procedure: . . . . .	30
7.7	Observation Table: . . . . .	31
7.8	Results and Discussion: . . . . .	31
7.9	Conclusion: . . . . .	32
<b>8</b>	<b>Lab 8: Sequential Circuits</b>	<b>33</b>
8.1	Objective: . . . . .	33
8.2	Description: . . . . .	33
8.3	Flip-flops and Latches: . . . . .	34
8.3.1	Basic Flip-flop circuit . . . . .	34
8.4	Latches Vs. Flip-flops: . . . . .	35
8.4.1	D Latch . . . . .	35
8.4.2	D Flip-flop . . . . .	35
8.5	Lab Work . . . . .	36
8.5.1	Parts List: . . . . .	36
8.5.1.1	D Latch . . . . .	36
8.5.1.2	D Flip-flop . . . . .	36
8.6	Conclusion: . . . . .	38
<b>9</b>	<b>Lab 9: JK Flip Flop</b>	<b>39</b>
9.1	Aim . . . . .	39
9.2	Objective: . . . . .	39
9.3	Apparatus Required: . . . . .	39
9.4	Pin Diagram: . . . . .	39
9.5	Theory: . . . . .	39
9.6	Procedure: . . . . .	40
9.7	Observation Table: . . . . .	41
9.8	Results and Discussion: . . . . .	42
9.9	Conclusion: . . . . .	42
<b>10</b>	<b>Lab 10: Gray to Binay Code</b>	<b>43</b>
10.1	Aim . . . . .	43
10.2	Objective: . . . . .	43
10.3	Apparatus Required: . . . . .	43
10.4	Pin Diagram: . . . . .	44

10.5 Theory: . . . . .	44
10.6 Procedure: . . . . .	44
10.7 Observation Table: . . . . .	45
10.8 Calculation: . . . . .	45
10.8.1 Kmap Simplification: . . . . .	45
10.8.2 Boolean Expression: . . . . .	45
10.9 Results and Discussion: . . . . .	47
10.10 Conclusion: . . . . .	47
<b>11 Lab 11: BCD to Excess-3</b>	<b>48</b>
11.1 Aim . . . . .	48
11.2 Objective: . . . . .	48
11.3 Apparatus Required: . . . . .	48
11.4 Theory: . . . . .	49
11.4.1 Equations: . . . . .	50
11.5 Procedure: . . . . .	50
11.6 Observation Table: . . . . .	51
11.7 Calculation: . . . . .	52
11.7.1 Kmap Simplification: . . . . .	52
11.7.2 Boolean Expression: . . . . .	52
11.8 Results and Discussion: . . . . .	52
11.9 Conclusion: . . . . .	53
<b>12 Lab 12: Shift Register</b>	<b>54</b>
12.1 Aim . . . . .	54
12.2 Objective: . . . . .	54
12.3 Background: . . . . .	54
$\hat{a} \sim A'c$ . . . . .	54
12.4 Lab Work . . . . .	55
12.4.1 Parts List: . . . . .	55
12.4.2 Serial-In Parallel-Out Shift Register . . . . .	56
12.4.3 Parallel-In Series-out/Serial-In Serial-out Register . . . . .	56
12.4.4 Bidirectional Shift Register . . . . .	57
12.5 Results and Discussion: . . . . .	57
12.6 Conclusion: . . . . .	57
<b>13 Lab 13: Counters</b>	<b>60</b>
13.1 Objectives . . . . .	60
13.2 Background . . . . .	60
13.2.1 Classification of Counters . . . . .	60
13.2.2 Binary Ripple Counter . . . . .	62

13.2.3	BCD Ripple Counter . . . . .	62
13.3	Prelab . . . . .	62
13.4	Required Equipment: . . . . .	62
13.5	Procedures: . . . . .	63
13.5.1	4-bit binary Asynchronous Counter . . . . .	63
<b>14</b>	<b>Lab 14: Synchronous Counter</b>	<b>66</b>
14.1	Aim: . . . . .	66
14.2	Apparatus Required: . . . . .	66
14.3	Procedure: . . . . .	66
14.4	Counter Circuit . . . . .	66
14.5	Synchronous BCD Counter . . . . .	66
14.5.1	Synchronous BCD Up/Down Counter . . . . .	67
14.6	Summary . . . . .	68
<b>15</b>	<b>Lab 15: 4 bit ALU</b>	<b>69</b>
15.1	Aim: . . . . .	69
15.2	Apparatus Required: . . . . .	69
15.3	Procedure: . . . . .	69
15.4	Pin Specification . . . . .	69

---

# Course Learning Outcomes (CLOs)

---

**Course Learning Outcomes (CLO):**

**Upon successful completion of the course, the students will be able to:**

**CLO1:**

**Understand fundamental concepts of digital logic design including basic and universal gates, number systems, binary coded systems, basic components of combinational and sequential circuits**

**CLO2:**

**Demonstrate the acquired knowledge to apply techniques related to the design and analysis of digital electronic circuits including Boolean algebra and multi-variable Karnaugh map methods**

**CLO3:**

**Analyze small-scale combinational and sequential digital circuits**

**CLO4:**

**Synthesize small-scale combinational and synchronous sequential digital circuit using Boolean algebra and K-maps**

---

# Lab 1: Verification of Gates

---

## 1.1 Aim

To study and verify the Truth Tables of AND, OR, NOT, NAND, NOR, XOR, XNOR logic gates for positive logic.

## 1.2 Objectives:

- To get familiar with the usage of the available lab equipment.
- To get familiar with Prototyping board (breadboard)
- To describe and verify the operation for the AND, OR, NOT, NAND, NOR, XOR, XNOR gates.
- To study the representation of these functions by truth tables, logic diagrams and Boolean algebra
- To Introduce a basic knowledge in integrated circuit devices operation
- To practice how to build a simple digital circuit using ICs and other digital components .
- Learn how to Wire a circuit

## 1.3 Apparatus/Equipment Required:

- Prototyping board (breadboard)
- DC Power Supply 5V
- Light Emitting Diode (LED)
- Digital ICs:
  - 7404 :Hex Inverter
  - 7408 :Quad 2 input AND



- 7432 :Quad 2 input OR
- 7400: Quad 2 input NAND
- 7402: Quad 2 input NOR
- 7486: Quad 2 input XOR
- 74266:Quad 2 input XNOR

- Connecting Wires

### 1.3.1 Pin Diagram

Figure 1.1 shows the architecture of SAP-1, a bus-organized computer. All register outputs to the 8 bit W-bus are three states; this allows orderly transfer of data. All other register outputs are two state; these outputs continuously drive the boxes they are connected to. The layout of Fig.1.1 emphasizes the registers used in SAP- 1. For this reason, no attempt has been made to keep all control circuits in one block called the control unit, all input-output circuits in another block called the I/O unit, etc

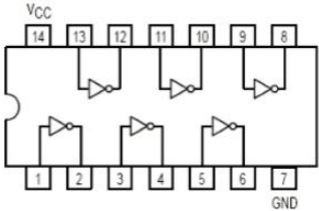
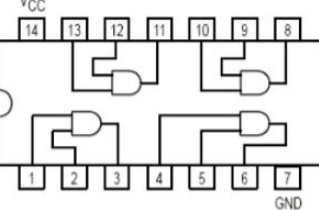
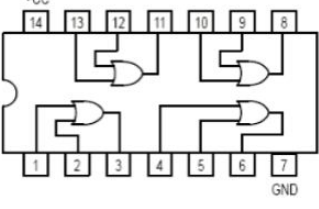
	<p><b>NOT Gate: IC 7404(HEX Inverter)</b>  <b>14 Pin</b>  <b>Supply voltage :5V</b></p>
	<p><b>AND Gate: IC 7408</b>  <b>14 Pin</b>  <b>Quad 2 input AND Gate</b>  <b>Supply voltage :5V</b></p>
	<p><b>OR Gate: IC 7432</b>  <b>14 Pin</b>  <b>Quad 2 input OR Gate</b>  <b>Supply voltage :5V</b></p>

Figure 1.1: Pin Specification of various Digital ICs.

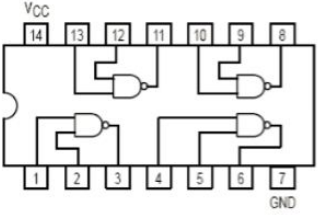
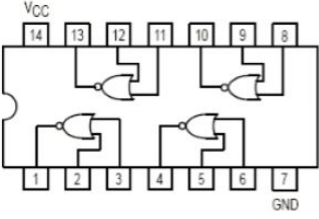
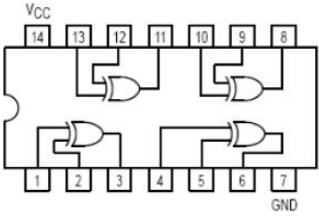
	<p><b>NAND Gate: IC 7400</b></p> <p><b>14 Pin</b></p> <p><b>Quad 2 input NAND Gate</b></p> <p><b>Supply voltage :5V</b></p>
	<p><b>NOR Gate: IC 7402</b></p> <p><b>14 Pin</b></p> <p><b>Quad 2 input NOR Gate</b></p> <p><b>Supply voltage :5V</b></p>
	<p><b>XOR Gate: IC 7486</b></p> <p><b>14 Pin</b></p> <p><b>Quad 2 input EXOR Gate</b></p> <p><b>Supply voltage :5V</b></p>

Figure 1.2: Pin Specification of various Digital ICs.

## 1.4 Theory

A Digital Logic Gate is an electronic device that makes logical decisions based on the different combinations of digital signals present on its inputs. Logic gates are the building blocks of digital circuits. Combinations of logic gates form circuits designed with specific tasks in mind. They are fundamental to the design of computers. Digital logic using transistors is often referred to as Transistor-Transistor Logic or TTL gates. These gates are the AND, OR, NOT, NAND, NOR, XOR and XNOR gates.

### 1.4.1 AND Gate

A multi-input circuit in which the output is 1 only if all inputs are 1. The symbolic representation of the AND gate is:

A dot (.) is used to show the AND operation i.e.  $A \cdot B$  if A and B are the inputs to AND Gate.

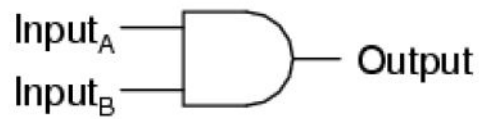


Figure 1.3: Logic Symbol of AND Gate

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

Figure 1.4: Truth Table for AND Gate

#### 1.4.2 OR Gate

A multi-input circuit in which the output is 1 when any input is 1. The symbolic representation of the OR gate is shown:

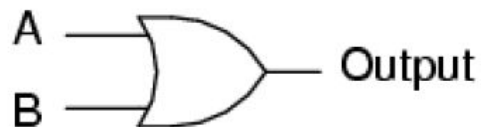


Figure 1.5: Logic Symbol of OR Gate

The OR gate is an electronic circuit that gives a high output (1) if one or more of its inputs are high. A plus (+) is used to show the OR operation.

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

Figure 1.6: Truth Table for OR Gate

### 1.4.3 NOT Gate

The output is 0 when the input is 1, and the output is 1 when the input is 0. The symbolic representation of an inverter is :

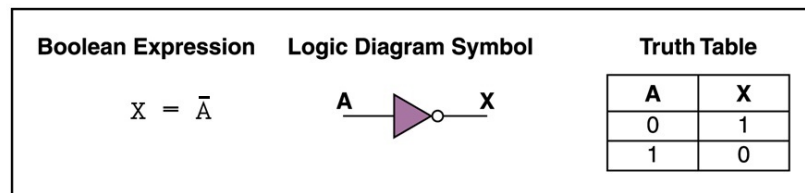


Figure 1.7: Logic Symbol and Truth Table for NOT gate

The NOT gate is an electronic circuit that produces an inverted version of the input at its output. It is also known as an inverter. If the input variable is A, the inverted output is known as NOT A. This is also shown as A', or A with a bar over the top, as shown at the outputs.

### 1.4.4 NAND Gate

AND followed by INVERT. It is also known as universal gate. The symbolic representation of the NAND gate is:

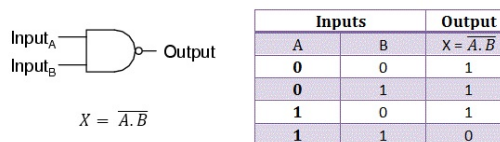


Figure 1.8: Logic Symbol and Truth Table for NAND gate

This is a NOT-AND gate which is equal to an AND gate followed by a NOT gate. The outputs of all NAND gates are high if any of the inputs are low. The symbol is an AND gate with a small circle on the output. The small circle represents inversion.

### 1.4.5 NOR Gate

OR followed by inverter. It is also known as universal gate. The symbolic representation is:

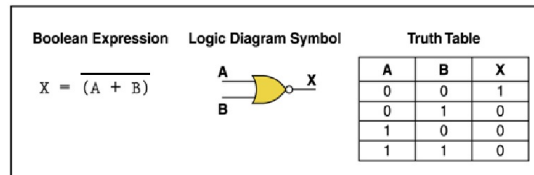


Figure 1.9: Logic Symbol and Truth Table for NOR gate

### 1.4.6 Exclusive-OR (XOR Gate)

The output of the Exclusive-OR (XOR) gate is 0 when its two inputs are the same and its output is 1 when its two inputs are different.

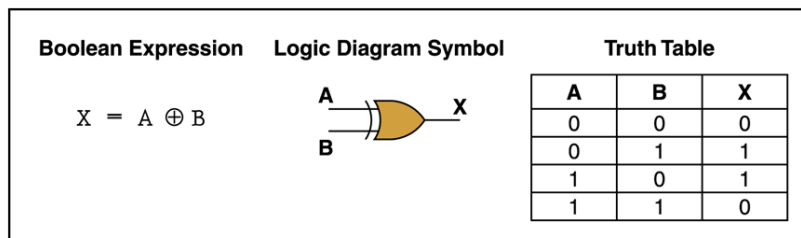


Figure 1.10: Logic Symbol and Truth Table for XOR gate

The 'Exclusive-OR' gate is a circuit which will give a high output if either, but not both, of its two inputs are high. An encircled plus sign (  $\oplus$  ) is used to show the XOR operation.

### 1.4.7 Exclusive-NOR (XNOR)

The XNOR gate is a digital logic gate whose function is the inverse of the operation exclusive OR (XOR) gate. The output of the Exclusive NOR gate, is 0 when its two inputs are the different and its output is 1 when its two inputs are same. An encircled plus sign and bar are used to show the XNOR operation.

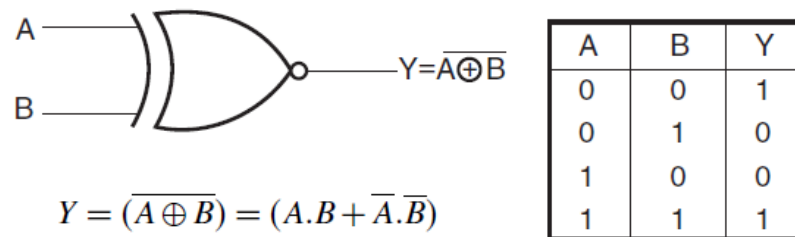


Figure 1.11: Logic Symbol and Truth Table for XNOR gate

### 1.5 Procedure:

1. Collect the components necessary to accomplish this experiment.
2. Plug the IC chip into the breadboard.
3. Connect the supply voltage and ground lines to the chips. PIN7 = Ground and PIN14 = +5V.
4. According to the pin diagram of each IC mentioned above, wire only one gate to verify its truth table.
5. Connect the inputs of the gate to the input switches of the LED.
6. Connect the output of the gate to the output LEDs.
7. Once all connections have been done, turn on the power switch of the breadboard
8. Operate the switches and fill in the truth table ( Write "1" if LED is ON and "0" if LED is OFF Apply the various combination of inputs according to the truth table and observe the condition of Output LEDs.
9. Repeat the above steps 1 to 5 for all the ICs.

### 1.6 Observation Table:

Input variables: A and B

Output variable: Y

LED ON: Logic 1

LED OFF: Logic 0

### 1.7 Results and Analysis:

- NOT Gate: When logic 1 is applied to one of NOT gate of 7404 IC, then output becomes zero. When input LED is ON (RED), the output LED become OFF (Green) vice versa.

Table 1.1: Output Observation Table

S.No	Inputs		Outputs						
	A	B	NOT	AND	OR	NAND	NOR	XOR	XNOR
1									
2									
3									
4									

- OR Gate: The output of an OR gate is a 1 if one or the other or both of the inputs are 1, but a 0 if both inputs are 0. When One or the other or Both of the input LEDs are ON (RED Light), then output LED is ON(RED) otherwise Output LED is OFF(Green Light)
- AND Gate: The output of an AND gate is only 1 if both its inputs are 1. For all other possible inputs the output is 0. When both the LEDs are On, then output LED is ON (RED Light) otherwise Output LED is OFF.
- NOR Gate: The output of the NOR gate is a 1 if both inputs are 0 but a 0 if one or the other or both the inputs are 1. NAND Gate: The output of the NAND gate is a 0 if both inputs are 1 but a 1 if one or the other or both the inputs are 0.
- XOR gate: The output of the XOR gate is a 1 if either but not both inputs are 1 and a 0 if the inputs are both 0 or both 1.
- XNOR gate: The output of the Exclusive-OR gate, is 0 when its two inputs are the different and its output is 1 when its two inputs are same.

## 1.9 Conclusion:

Any Boolean expression can be realized using NOT, AND, OR, NAND, NOR, XOR, XNOR gates.

# Lab 2:Half Adder

---

## 2.1 Aim

Design and verify the logic circuit of Half adder using logic gates.

## 2.2 Objectives

- To understand the principle of binary addition.
- To understand half adder concept.
- Use truth table and Boolean Algebra theorems in simplifying a circuit design.
- To implement half adder circuit using logic gates

## 2.3 Apparatus Required

- Prototyping board (breadboard)
- DC Power Supply +5V
- Light Emitting Diode (LED)
- Digital ICs:
  - 7408 :Quad 2 input AND
  - 7486: Quad 2 input XOR
  - 7432 :Quad 2 input OR

## 2.4 Pin Diagram:

## 2.5 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.



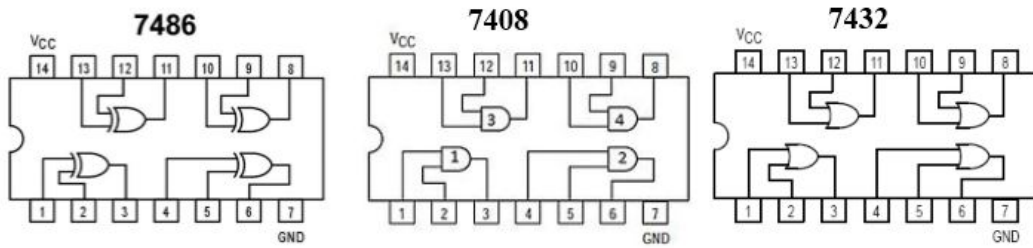


Figure 2.1: Pin Diagram for 7408, 7486 and 7432 ICs

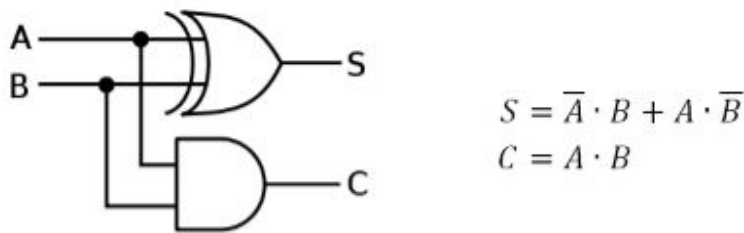


Figure 2.2: Half Adder Logic circuit and Boolean expression for each outputs

## 2.6 Procedure:

1. Collect the components necessary to accomplish this experiment.
2. Plug the IC chip into the breadboard.
3. Connect the supply voltage and ground lines to the chips. PIN7 = Ground and PIN14 = +5V.
4. According to the pin diagram of each IC mentioned above, make the connections according to circuit diagram.
5. Connect the inputs of the gate to the input switches of the LED.
6. Connect the output of the gate to the output LEDs.
7. Once all connections have been done, turn on the power switch of the breadboard
8. Operate the switches and fill in the truth table ( Write "1" if LED is ON and "0" if LED is OFF Apply the various combination of inputs according to the truth table and observe the condition of Output LEDs.

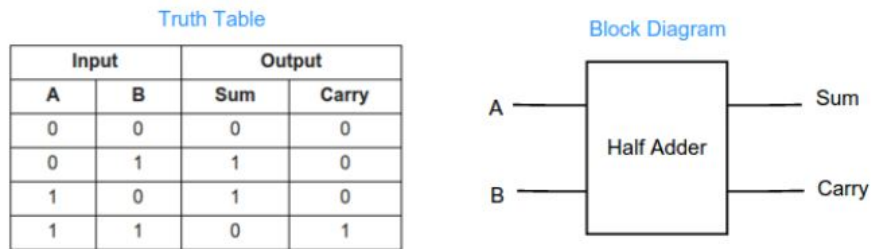


Figure 2.3: Half Adder Block diagram and Truth Table

## 2.7 Observation table

Half Adder Input Variable: A ,B

Output Variable: S, C

LED ON: RED Light: Logic 1

LED OFF: Green Light: Logic 0

Table 2.1: Output Observation Table

Inputs		Outputs	
A	B	Sum (S)	Carry C

## 2.8 Results and Analysis

Half Adder: Verified the truth table of Half Adder as  $S = 1$  i.e. LED which is connected to S terminal glows when inputs are A ,B Verified the truth table of Half Adder as  $C = 1$  i.e LED which is connected to C terminal glows when inputs are A, B.

## 2.9 Conclusion:

- To add two bits we require one XOR gate(IC 7486 ) to generate Sum and one AND (IC 7408) to generate carry.
- To add three bits we require two half adders.

# Lab 3: Full Adder

---

## 3.1 Aim

Design and verify the logic circuit Full adder using Half adder.

## 3.2 Objective:

- To understand the principle of binary addition.
- To understand full adder concept.
- Use truth table and Boolean Algebra theorems in simplifying a circuit design.
- To implement full adder circuit using logic gates.

## 3.3 Apparatus Required

- Prototyping board (breadboard)
- DC Power Supply 5V Battery
- Light Emitting Diode (LED)
- Digital ICs:
  - 7408 :Quad 2 input AND
  - 7486: Quad 2 input XOR
  - 7432 :Quad 2 input OR
- Connecting Wires

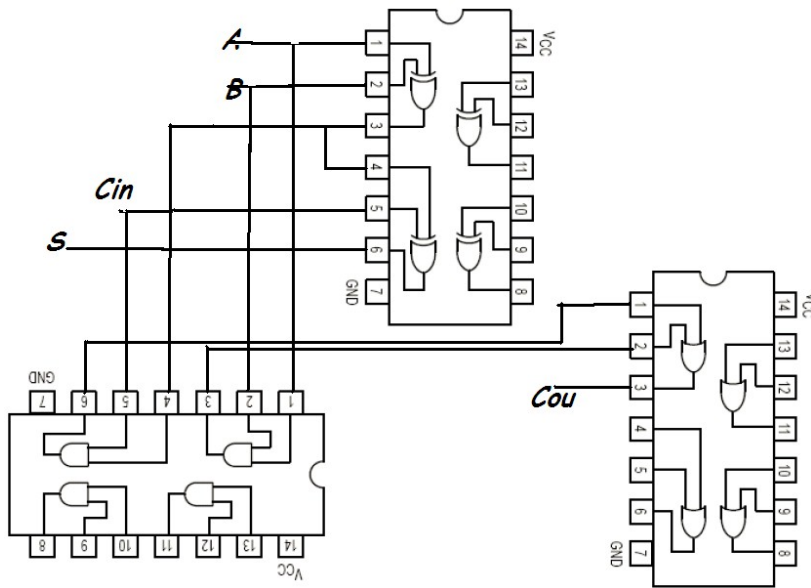


Figure 3.1: Pin diagram of Full adder

### 3.4 Pin Diagram

### 3.5 Theory

Full Adder: Full adder is a logical circuit that performs an addition operation on three binary digits. The full adder produces a sum and carry value, which are both binary digits. It can be combined with other full adders or work on its own.

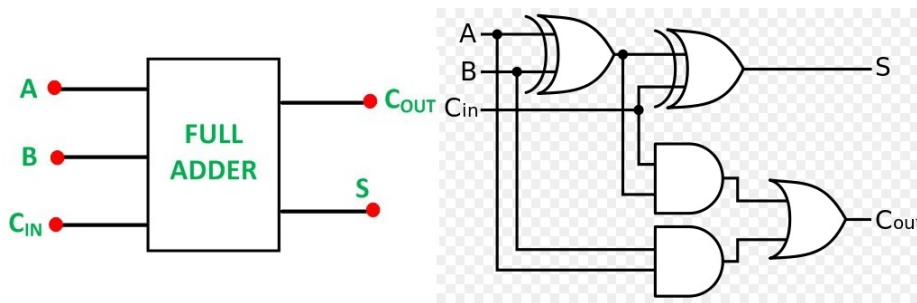


Figure 3.2: Logic circuit and block diagram of Full Adder

### 3.6 Procedure:

1. Collect the components necessary to accomplish this experiment.
2. Plug the IC chip into the breadboard.

### Full adder truth table

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

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

A	B	Carry-In	Sum	Carry-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

Figure 3.3: Boolean Expression and Truth Table for Full Adder

3. Connect the supply voltage and ground lines to the chips. PIN7 = Ground and PIN14 = +5V.
4. According to the pin diagram of each IC mentioned above, make the connections according to circuit diagram.
5. Connect the inputs of the gate to the input switches of the LED.
6. Connect the output of the gate to the output LEDs.
7. Once all connections have been done, turn on the power switch of the breadboard
8. Operate the switches and fill in the truth table ( Write "1" if LED is ON and "0" if LED is OFF Apply the various combination of inputs according to the truth table and observe the condition of Output LEDs.

### 3.7 Observation Table

Full adder: Input Variable: A ,B,Ci Output Variable: SUM(S), Carry(C<sub>0</sub>) LED ON: RED Light: Logic 1 LED OFF: Green Light: Logic 0

Table 3.1: Add caption

Inputs			Outputs	
A	B	Ci	Sum (S)	Carry $\hat{A}$ '

### **3.8 Results And Analysis**

Verified the truth table as follows.

Full Adder: Verified the truth table of Full Adder as  $S = 1$  i.e LED which is connected to S terminal glows when inputs are A ,B ,Ci Verified the truth table of Full Adder as  $Co = 1$  i.e LED which is connected to Co terminal glows when inputs are A ,B, Co

### **3.9 Conclusion:**

1. To add two bits we require one XOR gate(IC 7486 ) to generate Sum and one AND (IC 7408) to generate carry. 2. To add three bits we require two half adders.

# Lab 4: Half Subtractor

---

## 4.1 Aim

Design and verify the logic circuit of Half-subtractor using logic gate.

## 4.2 Objective:

- To understand the principle of binary subtraction.
- To understand half-subtractor concept.
- Use truth table and Boolean Algebra theorems in simplifying a circuit design.
- To implement half-subtractor circuit using logic gates

## 4.3 Apparatus Requirement:

- Prototyping board (breadboard)
- DC Power Supply 5V Battery
- Light Emitting Diode (LED)
- Digital ICs:
  - 7408 :Quad 2 input AND
  - 7486: Quad 2 input XOR
  - 7432 :Quad 2 input OR
  - 7404: Hex inverter (NOT Gate)
- Connecting Wires

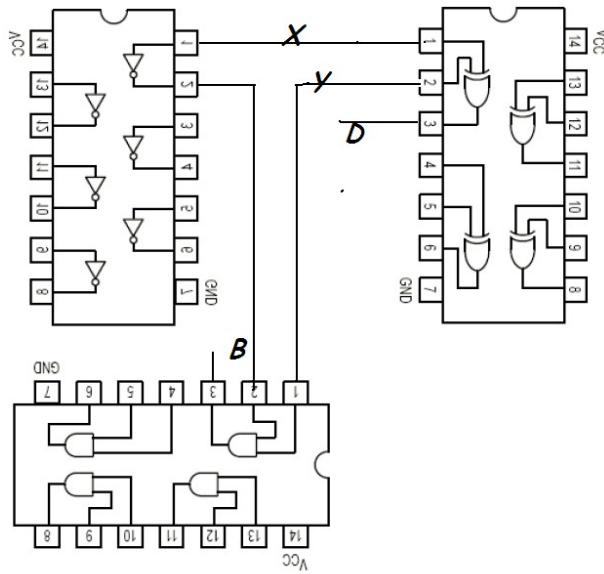


Figure 4.1: Pin Diagram of Half-Subtractor

#### 4.4 Pin Diagram:

#### 4.5 Theory:

Half Subtractor: The half-subtractor is a combinational circuit which is used to perform subtraction of two bits. It has two inputs, X (minuend) and Y (subtrahend) and two outputs D (difference) and B (borrow).

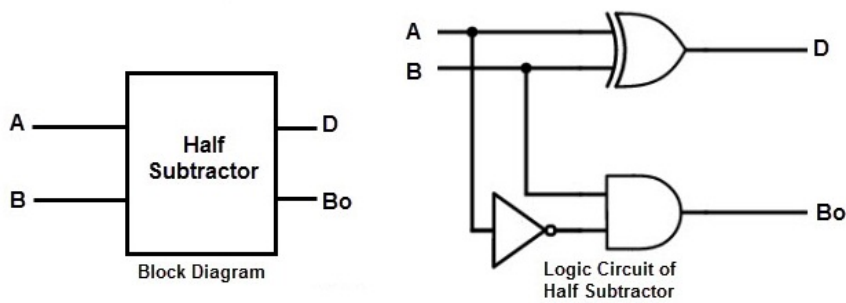


Figure 4.2: Logic Circuit and Block diagram of half-subtractor

#### 4.6 Procedure:

- Collect the components necessary to accomplish this experiment.
- Plug the IC chip into the breadboard.



A	B	D	B <sub>0</sub>
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Truth Table

$$D = A + B$$

$$B = A\bar{B}$$

Boolean Expression

Figure 4.3: Truth Table and Boolean expression of half-subtractor

- Connect the supply voltage and ground lines to the chips. PIN7 = Ground and PIN14 = +5V.
- According to the pin diagram of each IC mentioned above, make the connections according to circuit diagram.
- Connect the inputs of the gate to the input switches of the LED.
- Connect the output of the gate to the output LEDs.
- Once all connections have been done, turn on the power switch of the bread-board
- Operate the switches and fill in the truth table ( Write "1" if LED is ON and "0" if LED is OFF Apply the various combination of inputs according to the truth table and observe the condition of Output LEDs.

#### 4.7 Observation Table:

Full Subtractor: Input Variable: A,B,BIN, D,BOUT Output Variable: D,B LED ON: RED Light: Logic 1 LED OFF: Green Light: Logic 0

Table 4.1: Output Observation Table

Inputs		Outputs	
A	B	D	B <sub>0</sub>

## 4.8 Results and Analysis:

Verified the truth table as follows. Full Subtractor: Verified the truth table of Full Subtractor as  $D = 1$  i.e LED which is connected to D terminal glows when inputs are X, Y, BIN Verified the truth table of Full Subtractor as  $BOUT = 1$  i.e LED which is connected to BOUT terminal glows when inputs are X, Y, BIN

## 4.9 conclusion:

- To add two bits we require one XOR gate(IC 7486 ) to generate Difference and one AND (IC 7408) and NOT Gate(IC 7432) to generate Borrow.
- To add three bits we require two half subtractor.

# Lab 5: Full Subtractor

---

## 5.1 Aim

Design and verify the logic circuit of Full subtractor using of Half subtractor.

## 5.2 Objective:

- To understand the principle of binary subtraction.
- To understand full subtractor concept.
- Use truth table and Boolean Algebra theorems in simplifying a circuit design.
- To implement full subtractor circuit of Half subtractor

## 5.3 Apparatus Requirement:

- Prototyping board (breadboard)
- DC Power Supply 5V
- Light Emitting Diode (LED)
- Digital ICs:
  - 7408 :Quad 2 input AND
  - 7486: Quad 2 input XOR
  - 7432 :Quad 2 input OR
  - 7404: Hex inverter (NOT Gate)
- Connecting Wires

## 5.4 Pin Diagram:

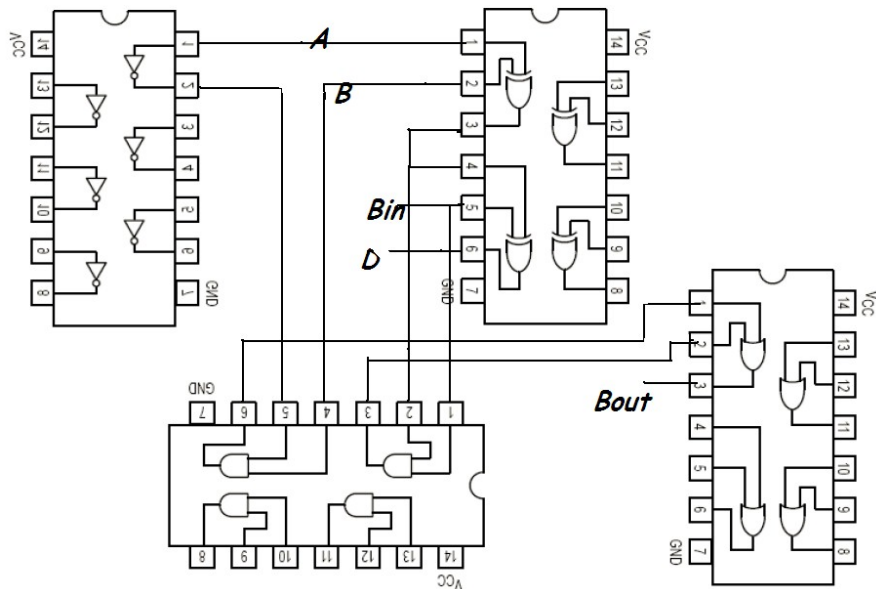


Figure 5.1: Pin Diagram of Full Subtractor

## 5.5 Theory:

Full subtractor: A full Subtractor is combinational circuit that performs a subtraction between three bits, taking into account that a '1' may have been borrowed by a lower significant stage. The 3 inputs denote minuend, subtrahend and previous borrow, respectively. The 2 outputs are difference(D) and borrow(B).

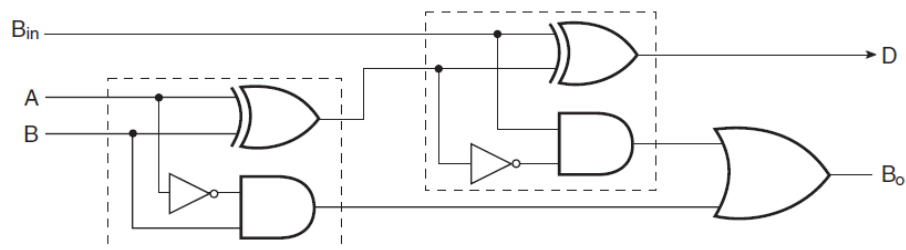


Figure 5.2: Logic Circuit of half subtractor

<b>A</b>	<b>B</b>	<b>B<sub>in</sub></b>	<b>D</b>	<b>B<sub>out</sub></b>
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Figure 5.3: Truth Table of Full subtractor

## 5.6 Procedure:

1. Collect the components necessary to accomplish this experiment.
2. Plug the IC chip into the breadboard.
3. Connect the supply voltage and ground lines to the chips. PIN7 = Ground and PIN14 = +5V.
4. According to the pin diagram of each IC mentioned above, make the connections according to circuit diagram.
5. Connect the inputs of the gate to the input switches of the LED.
6. Connect the output of the gate to the output LEDs.
7. Once all connections have been done, turn on the power switch of the breadboard
8. Operate the switches and fill in the truth table ( Write "1" if LED is ON and "0" if LED is OFF Apply the various combination of inputs according to the truth table and observe the condition of Output LEDs.

## 5.7 Observation Table:

Input Variable:  $x, y$  Output Variable:  $D, B$  LED ON: RED Light: Logic 1 LED OFF: Green Light: Logic 0

Table 5.1: Add caption

Inputs			Outputs	
A	B	$B_{in}$	D	$B_{out}$

## 5.8 Results and Analysis:

Verified the truth table as follows. Full Subtractor: Verified the truth table of Full Subtractor as  $D = 1$  i.e LED which is connected to D terminal glows when inputs are X, Y, BIN Verified the truth table of Full Subtractor as  $BOUT = 1$  i.e LED which is connected to BOUT terminal glows when inputs are X, Y, BIN

## 5.9 conclusion:

- To add two bits we require one XOR gate(IC 7486 ) to generate Difference and one AND (IC 7408) and NOT Gate(IC 7432) to generate Borrow.
- To add three bits we require two half subtractor.

# Lab 6: Multiplexer

---

## 6.1 Aim

To the Truth Table of 4:1 Multiplexer using IC 74153.

## 6.2 Objective:

- To get familiar with the concept of multiplexing
- To get familiar with MSI (medium scale integration) technology.

## 6.3 Apparatus Required:

- Prototyping board (breadboard)
- DC Power Supply 5V
- Light Emitting Diode (LED)
- Digital ICs:74153:Dual 4:1 MUX
- Connecting Wires

## 6.4 Apparatus Requirement:

- Prototyping board (breadboard)
- DC Power Supply 5V
- Light Emitting Diode (LED)
- Digital ICs:
  - 7408 :Quad 2 input AND

- 7486: Quad 2 input XOR
- 7432 :Quad 2 input OR
- 7404: Hex inverter (NOT Gate)
- Connecting Wires

## 6.5 Pin Diagram:

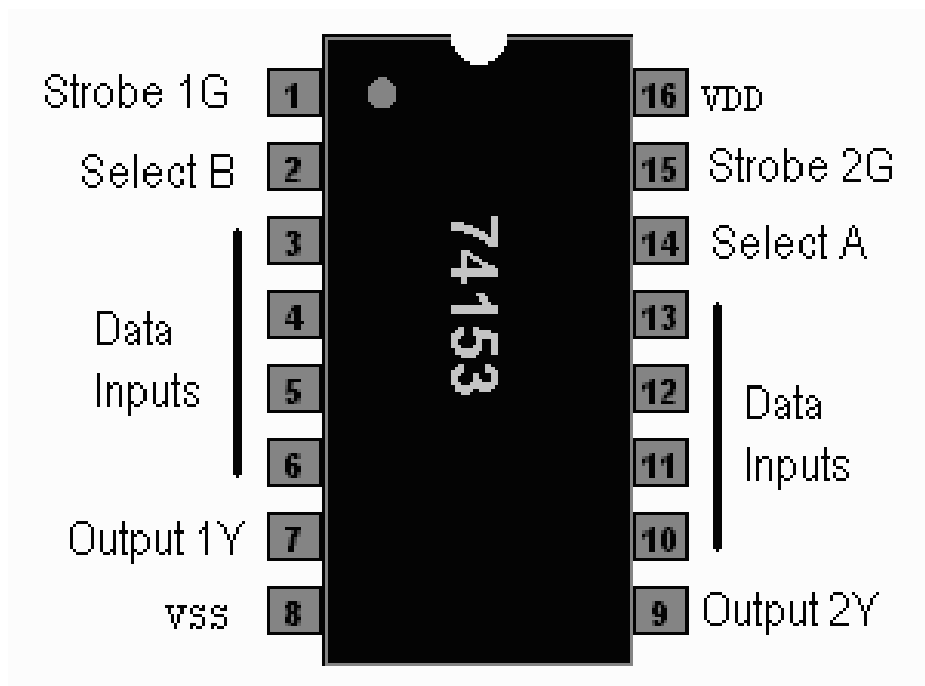


Figure 6.1: Pin Diagram of 4 x 1 Multiplexer

## 6.6 Theory:

Multiplexer: A data selector, more commonly called a Multiplexer, shortened to "MUX" or "MPX", is combinational logic switching devices that operate like a very fast acting multiple position rotary switches. They connect or control, multiple input lines called "channels" consisting of either 2, 4, 8 or 16 individual inputs, one at a time to an output. Then the job of a multiplexer is to allow multiple signals to share a single common output. For example, a single 8-channel multiplexer would connect one of its eight inputs to the single data output.

The Boolean expression for this 4-to-1 Multiplexer above with inputs I0 to I3 and data select lines S0 ,S1 is given as:  $Y = S_0S_1I_0 + S_0S_1I_1 + S_0S_1I_2 + S_0S_1I_3$



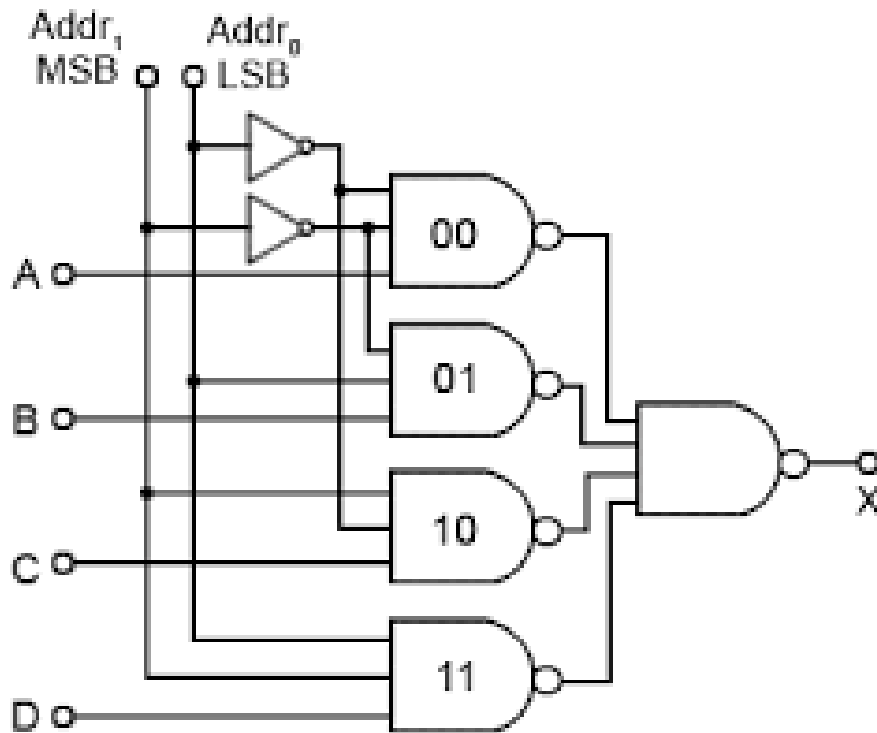


Figure 6.2: Logic Circuit of 4X1 MUX

### 6.7 Procedure:

1. Collect the components necessary to accomplish this experiment.
2. Plug the IC chip into the breadboard.
3. Connect the supply voltage and ground lines to the chips. PIN7 = Ground and PIN14 = +5V.
4. Make connections as shown in the respective circuit diagram.
5. Connect the inputs of the gate to the input switches of the LED.
6. Connect the output of the gate to the output LEDs.
7. Once all connections have been done, turn on the power switch of the breadboard
8. Operate the switches and fill in the truth table ( Write "1" if LED is ON and "0" if L1 is OFF Apply the various combination of inputs according to the truth table and observe the condition of Output LEDs.

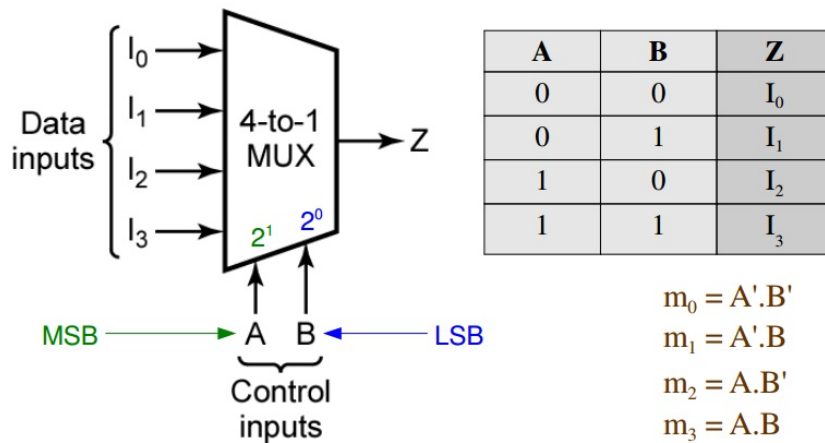


Figure 6.3: Truth Table and Block diagram of 4x1 MUX

### 6.8 Observation Table:

Input Lines: I<sub>3</sub>, I<sub>2</sub>, I<sub>1</sub>, I<sub>0</sub> Select Lines: S<sub>1</sub>, S<sub>0</sub> Output Variable: Z LED ON: RED Light:  
 Logic 1 LED OFF: Green Light: Logic 0

Table 6.1: Add caption

Select Lines		Output
S <sub>1</sub>	S <sub>0</sub>	Z

### 6.9 Results and Analysis:

Verified the truth table as follows. The input data was routed to output by varying the addresses on select lines

### 6.10 Conclusion:

The truth table of 4:1 MUX using IC 74153 has been verified.

# Lab 7: De-Multiplexer

---

## 7.1 Aim

To the Truth Table of 1:4 De-multiplexer using IC 74139

## 7.2 Objective:

- To get familiar with the concept of de-multiplexing

## 7.3 Apparatus Required:

- Prototyping board (breadboard)
- DC Power Supply 5V
- Light Emitting Diode (LED)
- Digital IC:74139:Dual 1:4 DEMUX
- Connecting Wires

## 7.4 Pin Diagram:

## 7.5 Theory:

De-multiplexer: The data distributor, known more commonly as a Demultiplexer or "Demux", is the exact opposite of the Multiplexer. The demultiplexer takes one single input data line and then switches it to any one of a number of individual output lines one at a time. The demultiplexer converts a serial data signal at the input to a parallel data at its output lines.

The Boolean expression for this 1-to-4 DeMultiplexer above with inputs  $I_0$  to  $I_3$  and data select lines  $S_0, S_1$  is given as:  $Y = S_0S_1D_0 + S_0S_1D_1 + S_0S_1D_2 + S_0S_1D_3$

The function of the Demultiplexer is to switch one common data input line to any one of the 4 output data lines. Some standard demultiplexer IC's also have an

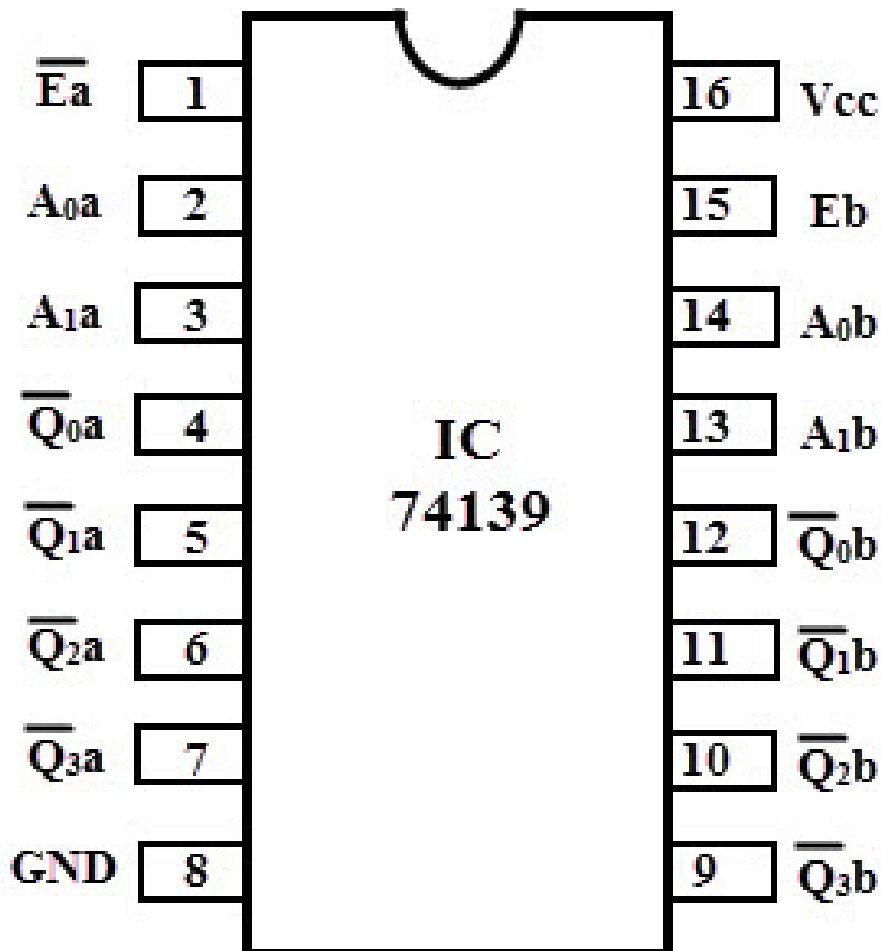


Figure 7.1: Pin Diagram of of dual 1 X 4 Demultiplexer

"enable output" input pin which disables or prevents the input from being passed to the selected output. Also some have latches built into their outputs to maintain the output logic level after the address inputs have been changed. However, in standard decoder type circuits the address input will determine which single data output will have the same value as the data input with all other data outputs having the value of logic "0".

## 7.6 Procedure:

1. Collect the components necessary to accomplish this experiment.
2. Plug the IC chip into the breadboard.
3. Connect the supply voltage and ground lines to the chips. PIN7 = Ground and PIN14 = +5V.

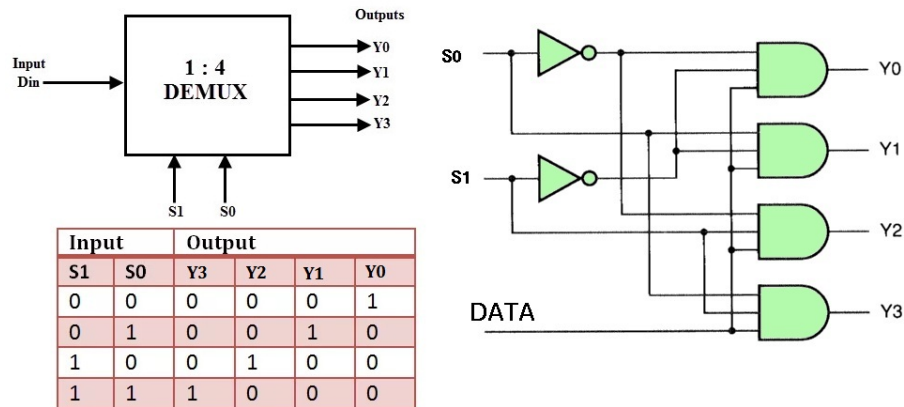


Figure 7.2: Block diagram, logic circuit and truth table for 1:4 Demultiplexer

4. Make connections as shown in the respective circuit diagram.
5. Connect the inputs of the gate to the input switches of the LED.
6. Connect the output of the gate to the output LEDs.
7. Once all connections have been done, turn on the power switch of the bread-board.
8. Operate the switches and fill in the truth table ( Write "1" if LED is ON and "0" if L1 is OFF Apply the various combination of inputs according to the truth table and observe the condition of Output LEDs.

## 7.7 Observation Table:

Input Variable: D, S0, S1 Output Variable: Y0, Y1, Y2, Y3 LED ON: RED Light: Logic 1 LED OFF: Green Light: Logic 0

Table 7.1: Add caption

Select Lines		Data Input	Output			
S1	S0	Data	Y	Y2	Y1	Y0

## 7.8 Results and Discussion:

Multiplexer and demultiplexer help in reducing the cost of transmission of digital signals. Logic design is simple and boolean expression need not be simplified. Mux

and Demux acts as rotary switches. Multiplexers are used as one method of reducing the number of logic gates required in a circuit or when a single data line is required to carry two or more different digital signals i.e it converts parallel data into serial data and also used for data selection. The demultiplexer converts a serial data signal at the input to a parallel data at its output lines and used for data distribution. Both are available as ICs.

## **7.9 Conclusion:**

1:4 DEMUX using IC74139 has been verified.

# Lab 8: Sequential Circuits

---

## 8.1 Objective:

- To be familiar with sequential circuits: SR Latch, D Flip Flop (FF), practically

## 8.2 Description:

Digital circuits can be categorized into combinational logic and sequential logic. Combinational logic refers to a circuit whose outcome is a function of the inputs and does not depend on previous states. Gates, encoders, decoders, multiplexers, demultiplexers, read only memories (ROM), programmable logic arrays (PLA) are all examples of combinational logic.

Sequential logic incorporates time as an input parameter. The outcome of sequential logic depends not only on the present inputs but also on the previous state of the output. Examples of sequential circuits are flip-flops, latches, counters, registers, time-state generators. A block diagram of a sequential circuit is shown in figure 8.1. It consists of a combinational circuit to which memory elements are connected to form feedback path. The memory elements are devices capable of storing binary information within them.

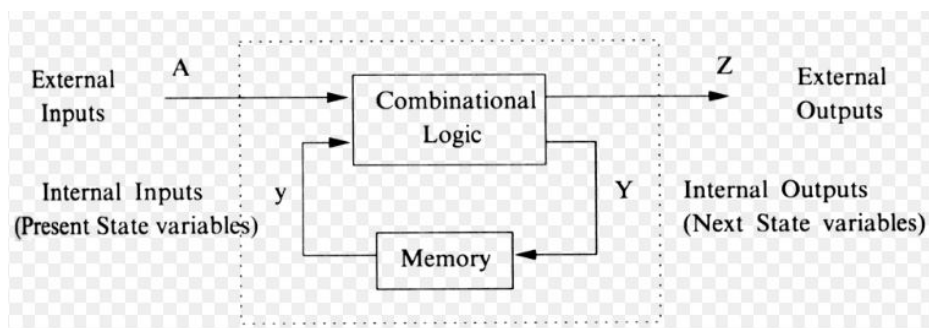


Figure 8.1: Block Diagram of Sequential Circuits

### 8.3 Flip-flops and Latches:

A flip-flop is a bi-stable device (It is a circuit having two stable conditions (states); it can be used to store binary symbols) with inputs, that remains in a given state as long as power is applied and until input signals are applied to cause its output to change. The major differences among various types of flip-flops are in the number of inputs they possess and in the manner in which the inputs affect the binary state. A flip-flop is a bi-stable device (It is a circuit having two stable conditions (states); it can be used to store binary symbols) with inputs, that remains in a given state as long as power is applied and until input signals are applied to cause its output to change. The major differences among various types of flip-flops are in the number of inputs they possess and in the manner in which the inputs affect the binary state.

It is often convenient to think of a sequential system as one clock period. A clock system as just described is called synchronous. The alternative, in which combinational operations trigger other operations as they occur, is called asynchronous operation.

#### 8.3.1 Basic Flip-flop circuit

It is often convenient to think of a sequential system as one clock period. A clock system as just described is called synchronous. The alternative, in which combinational operations trigger other operations as they occur, is called asynchronous operation.

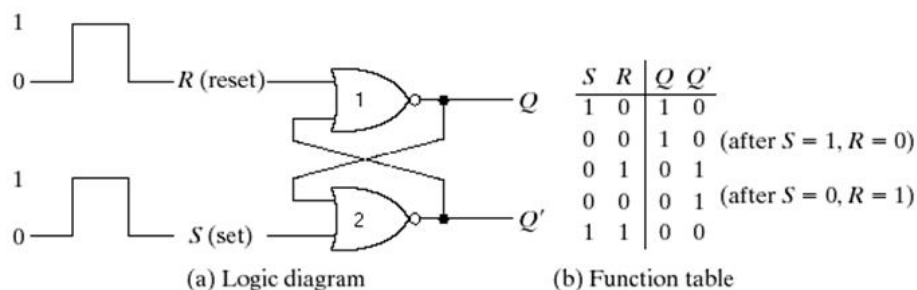


Figure 8.2: S-R Latch with NOR Gates and its Truth Table

We can analyze the operation of the circuit of figure 8.3 as follows:- As a starting point, assume that the set input is 1 and reset input is 0. Since gate 2 has an input of 1, its output  $Q'$  must be 0, which puts both inputs of gate 1 at 0, so that output Q is 1. When the set input is returned to 0, the outputs remain the same, because output Q remains a 1, leaving one input of gate 2 at 1. That causes output  $Q'$  to stay at 0, which leaves both inputs of gate number 1 at 0, so that output Q is a 1. In the same manner, it is possible to show that a 1 in the reset input changes output Q to 0 and  $Q'$  to 1. When the reset input returns to , the outputs do not change. When a 1 is applied to both the set and the reset inputs, both Q and  $Q'$  outputs go to 0. this condition violates the fact that outputs Q and  $Q'$  are the complements of each other. In normal operation, this condition must be avoided



by making sure that 1s are not applied to both inputs simultaneously. A flip-flop has two useful states. When  $Q = 1$  and  $Q' = 0$ , it is in the set state (or 1-state). When  $Q = 0$  and  $Q' = 1$ , it is in the clear state (or 0-state). The outputs  $Q$  and  $Q'$  are complements of each other and are referred to as the normal and complement outputs, respectively. The binary state of the flip-flop is taken to be the value of the normal output. Under normal operation, both inputs remain at 0 unless the state of the flip-flop has to be changed.

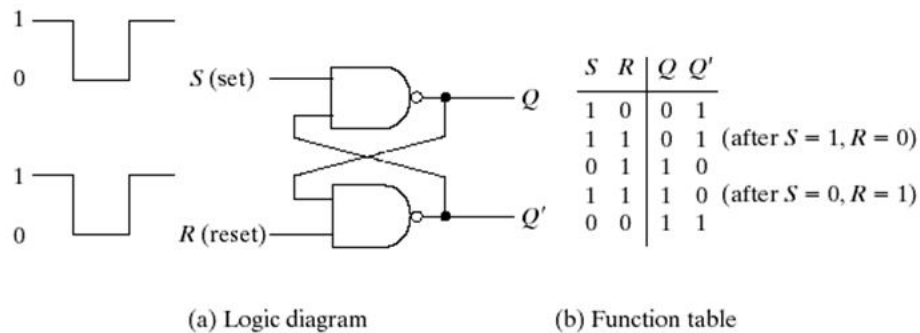


Figure 8.3: S-R Latch with NAND Gates and its Truth Table

The NAND basic flip-flop circuit of Figure 8.4 operates with both inputs normally at 1 unless the state of the flip-flop has to be changed.

## 8.4 Latches Vs. Flip-flops:

1. Latches are flip-flops for which the timing of the output changes is not controlled.
2. For a latch, the output essentially responds immediately to changes on the input lines (and possibly the presence of a clock pulse).
3. A flip-flop is designed to change its output at the edge of a controlling clock signal.

### 8.4.1 D Latch

D-Type Latch is the simplest type of storage device, The D-Type Latch is in fact a special case of the S-R Latch, For both the S-R and D-Type Latches, the stored value responds to any input change for the whole time the clock is high. This can cause problems. For proper synchronization, we build a modified latch which is edge-triggered. This is known as a Flip-flop.

### 8.4.2 D Flip-flop

One way to eliminate the undesirable condition of the indeterminate state in the RS flip-flop is to ensure that inputs  $S$  and  $R$  are never equal to 1 at the same time. This

is done in the D flip-flop shown in figure 8.6(a). The D flip-flop has only two inputs: D and C. The D input goes directly to the S input and its complement is applied to the R input. As long as the pulse input is at 0, the outputs of gates 3 and 4 are at the 1 level and the circuit cannot change state regardless of the value of D. The D input is sampled when C = 1. If D is 1, the Q output goes to 1, placing the circuit in the set state. If D is 0, output Q goes to 0 and the circuit switches to clear state (see figure 8.6(b)).

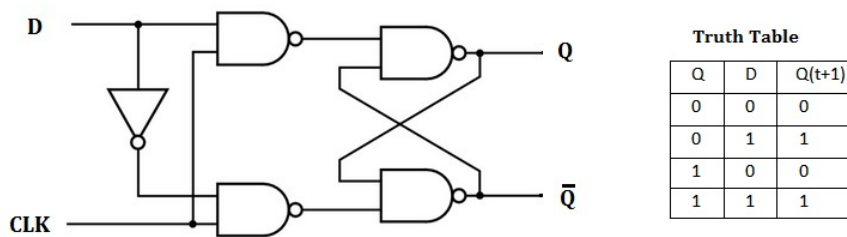


Figure 8.4: D FLip-flop and its truth table

## 8.5 Lab Work

### 8.5.1 Parts List:

- 74LS74 dual D type FF.
- 74LS76 dual JK type FF.
- 74LS75 quad Latch.

#### 8.5.1.1 D Latch

The 74x75 is a quad D type latch circuit in which any data change at the output will be transferred to the output whenever the proper logic level is present in the enable input of the device.

- Install the 74LS75 in the board and quickly verify the proper operation of one latch by connecting the input to switch and the output to LED. Connect also the enable input to switch and investigate its operation.
- Verify that the truth table is consistent with the device.
- Verify that the truth table is consistent with the device by PROTEUS simulator.

#### 8.5.1.2 D Flip-flop

The 74x74 is a dual D type FF sequential circuit in which any data change at the input will be transferred to the output ONLY on a rising edge transition

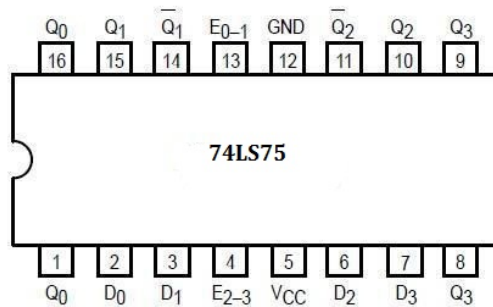


Figure 8.5: D Latch IC (74LS75) Pin specification

of a clock at the clock input signal of the FF. Note that the information will be stored by the FF and will not change as long as no rising edge voltage potential appears at the clock input of that particular FF.

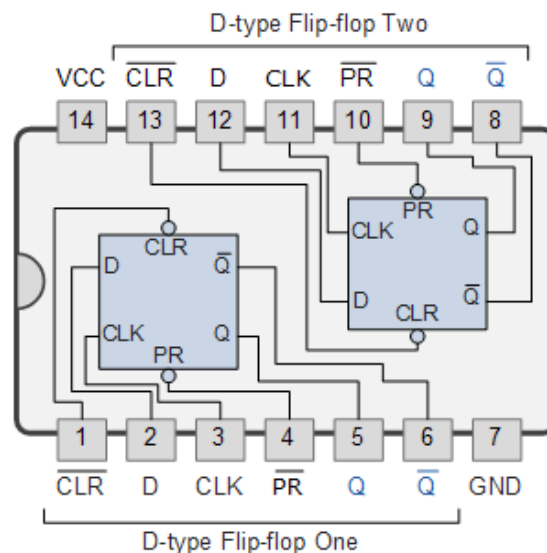


Figure 8.6: D Flip-flop IC (74LS74) Pin specification

1. Install the 74LS74 in the board and quickly verify the proper operation of one FF by connection the D input to switch, the clock input to the clock from the board, and the output to LED. Verify functional table for proper preset and clear inputs conditions.
2. Verify that data is transferred ONLY at the rising edge transition of the switch at the clock input when changing from "low" to "high" level.
3. Verify that regardless of the data changes present at the input of a FF, the data Will Not be transferred if edge transition of the switch at the clock input does not occur.

4. Verify that the truth table is consistent with the device by PROTEUS simulator.  
( Simulate all states).

## **8.6 Conclusion:**

D latch and D Flip-flops have been implemented and their operation has been verified successfully.

# Lab 9: JK Flip Flop

---

## 9.1 Aim

To Design and verify the truth table of J K Flip flop using IC 7473.

## 9.2 Objective:

- To understand the principle of operation of sequential circuit
- To differentiate between combinational circuit and sequential circuit.
- To get familiar with basic Flip flops
- Determine the logic operation of JK flip flops.
- Connect and observe the state transition of JK as connected to the clock generator circuit.

## 9.3 Apparatus Required:

- Prototyping board (breadboard)
- DC Power Supply 5V
- Light Emitting Diode (LED)
- Digital ICs: 7473: Dual master Slave J K Flip flop
- Connecting Wires

## 9.4 Pin Diagram:

## 9.5 Theory:

Sequential Logic circuits: In digital circuit theory, sequential logic is a type of logic circuit whose output depends not only on the present input but also on the history

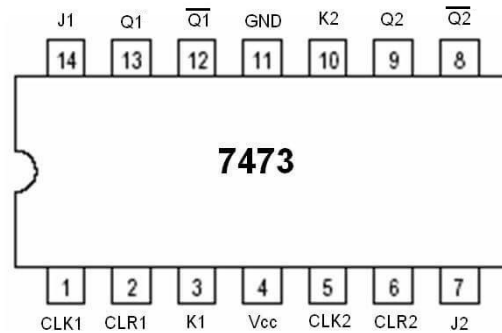


Figure 9.1: Pin Diagram of JK Flip-flop

of the input. This is in contrast to combinational logic, whose output is a function of, and only of, the present input. In other words, sequential logic has state (memory) while combinational logic does not. The memory elements are devices capable of storing 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. Memory elements next state is also a function of external inputs and present state. A sequential circuit is specified by a time sequence of inputs, outputs, and internal states.

**Flip Flop:** In electronics, a flip-flop is a circuit that has two stable states and can be used to store state information. The circuit can be made to change state by signals applied to one or more control inputs and will have one or two outputs. Flip-flops and latches are used as data storage elements. There are four types of flip flops. These are SR, D, JK and T. On this experiment we will explore the operation of JK flip flop. JK flip flop: JK flip flop is considered as the universal flip flop. When configured in various ways, it is capable of operating like most other types of flip flop. A JK flip-flop is a refinement of the SR flip-flop in that the indeterminate state of the SR type is defined in the JK type. Inputs J and K behave like inputs S and R to set and clear the flip-flop. When logic 1 inputs are applied to both J and K simultaneously, the flip-flop switches to its complement state, i.e., if  $Q=1$ , it switches to  $Q=0$  and vice versa. In that way it is like a toggle. A clocked JK flip-flop is shown below. Output Q is ANDed with K and CLK inputs so that the flip-flop is cleared during a clock pulse only if Q was previously 1. Similarly, output  $\bar{Q}$  is ANDed with J and CLK inputs so that the flip-flop is set with a clock pulse only if  $\bar{Q}$  was previously 1.

## 9.6 Procedure:

1. Collect the components necessary to accomplish this experiment.
2. Plug the IC chip into the breadboard.
3. Connect the supply voltage and ground lines to the chips. PIN7 = Ground and

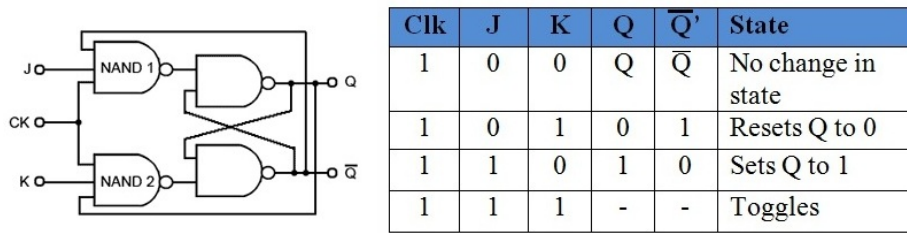


Figure 9.2: Logic circuit and truth table for JK Flip-flop

PIN14 = +5V.

4. According to the pin diagram of each IC mentioned above, make the connections according to circuit diagram.
5. Connect the inputs of the gate to the input switches of the LED.
6. Connect the output of the gate to the output LEDs.
7. Once all connections have been done, turn on the power switch of the breadboard
8. Operate the switches and fill in the truth table ( Write "1" if LED is ON and "0" if LED is OFF Apply the various combination of inputs according to the truth table and observe the condition of Output LEDs.

## 9.7 Observation Table:

JK Flip flop: Input Variable: CLR, CLK, J, K

Output Variable: Q, Q bar

LED ON: RED Light:Logic 1

LED OFF: Green Light:Logic 0

Table 9.1: Add caption

Control inputs		Inputs		Outputs	
CLR BAR	CLK	J	K	Q	Q bar

## **9.8 Results and Discussion:**

Flip-flops (FFs) are devices used in the digital field for a variety of purposes. Flip-flops are a fundamental building block of digital electronics systems used in computers, communications, and many other types of systems. In JK flip-flop, the letter J is for set and the letter K is for clear. When logic 1 inputs are applied to both J and K simultaneously, the flip-flop switches to its complement state, i.e, if  $Q=1$ , it switches to  $Q=0$  and vice versa. Flip-flops and latches are used as data storage elements. Such data storage can be used for storage of state, and such a circuit is described as sequential logic. When used in a finite-state machine, the output and next state depend not only on its current input, but also on its current state (and hence, previous inputs.) It can also be used for counting of pulses, and for synchronizing variably-timed input signals to some reference timing signal.

## **9.9 Conclusion:**

The function table of JK flip flop using IC 7473 has been verified.



# Lab 10: Gray to Binary Code

---

## 10.1 Aim

To Design and verify the truth table of code conversion from gray to binary code (4 bit) using basic Logic Gates

## 10.2 Objective:

- Design of different combinational circuits and their applications using basic logic gates.
- Creation and observation of the four-bit binary code number representation sequence
- Exercising the design of code conversion logic circuits,
- Creating the truth table of conversion functions from Gray to binary code
- Developing skills in simplification of specified logical functions

## 10.3 Apparatus Required:

- Prototyping board (breadboard)
- DC Power Supply 5V Battery
- Light Emitting Diode (LED)
- Digital ICs:7486: Quad 2 input XOR
- Connecting Wires

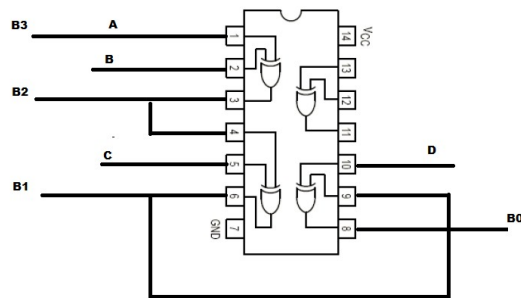


Figure 10.1: Pin Diagram of Gray to Binary code converter using 7486IC

## 10.4 Pin Diagram:

## 10.5 Theory:

Code Converters: A code converter is a circuit that makes two digital systems using different codes for the same information. It means that a code converter is a code translator from one code to the other. The code converter is used since to systems using two different codes but they need to use the same information. So the code converter is the solution. Gray-to Binary Converter: An interesting application for the exclusive-OR gate is a logic gate to change a gray number to its equivalent in binary Code. The logic circuit can be used to convert a 4-bit gray number ABCD into its binary-code equivalent, B3, B2, B1 and B0. Application: Some sensors send information in Gray code. These must be converted to binary in order to do arithmetic with it. Occasionally, it is necessary to convert back. Advantages: Higher speed or smaller code.

## 10.6 Procedure:

1. Collect the components necessary to accomplish this experiment.
2. Plug the IC chip into the breadboard.
3. Connect the supply voltage and ground lines to the chips. PIN7 = Ground and PIN14 = +5V.
4. Make connections as shown in the respective circuit diagram.
5. Connect the inputs of the gate to the input switches of the LED.
6. Connect the output of the gate to the output LEDs.
7. Once all connections have been done, turn on the power switch of the breadboard

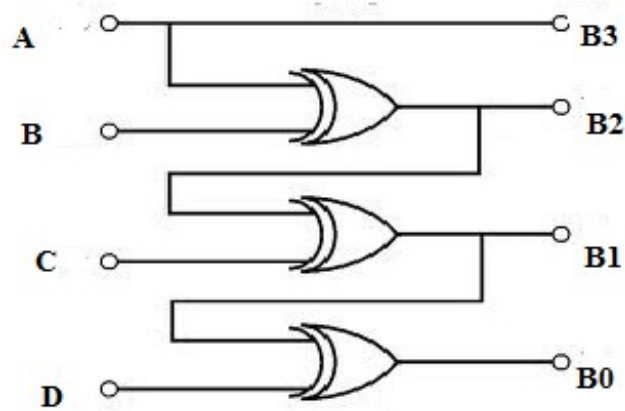


Figure 10.2: Circuit diagram of Binary to Gray Code Converter

8. Operate the switches and fill in the truth table Write "1" if LED is ON and "0" if L1 is OFF Apply the various combination of inputs according to the truth table and observe the condition of Output LEDs.

### 10.7 Observation Table:

: Input Variable: A B C D  
 Output Variable: B3 B2 B1 B0  
 LED ON: RED Light: Logic 1  
 LED OFF: Green Light: Logic 0

### 10.8 Calculation:

#### 10.8.1 Kmap Simplification:

For B3  
 For B2  
 For B1  
 For B0

#### 10.8.2 Boolean Expression:

B3= B2= B1= B0=

Table 10.1: Add caption

INPUTS				OUTPUTS			
A	B	C	D	B3	B2	B1	B0

Table 10.2: Simplification for B3


Table 10.3: Simplification for B2


Table 10.4: Simplification for B1


Table 10.5: Simplification for B0


BCD code				Gray code			
$B_3$	$B_2$	$B_1$	$B_0$	$G_3$	$G_2$	$G_1$	$G_0$
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

Figure 10.3: Truth table of Binary to Gray Code Converter

### 10.9 Results and Discussion:

The gray to binary code converter is used since two systems using two different codes but they need to use the same information. Gray to binary code converter convert correctly gray 0000 to 1111 into binary codes. The circuit diagram is very simple and only uses an 74886 IC i.e XOR gate. Unless the karnaugh map is used many gates may be used. but result of karnaugh map minimization, it can work only using XOR Gate. Gray code is a weighted ,cyclic and reflective code are used in instrumentation and acquisition system where linear or angular displacement is measured, shaft encoders, I/O devices ,A/D converters and outer peripheral devices.

### 10.10 Conclusion:

The function table of JK flip flop using IC 7473 has been verified.

# Lab 12: Shift Register

---

## 12.1 Aim

To Design and verify the function of Truth Table.

## 12.2 Objective:

- To investigate the operation of the shift registers.

## 12.3 Background:

A register capable of shifting its binary information either to right or to the left is called a shift register. The simplest possible shift register is one that uses only flip-flops, as shown in figure 10.1. The Q output of a given flip-flop is connected to the D input of the flip-flop at its right. Each clock pulse shifts the contents of the register one bit position to the right. The serial input determines what goes into the leftmost flip-flop during the shift. The serial output is taken from the output of the rightmost flip-flop prior to the application of a pulse. Although this register shifts its contents to the right, if we turn the page upside down; we find that the register shifts its contents to the left. Thus a unidirectional shift register can function either as a shift-right or as shift-left register.

âĀĀ The register in figure 10.1 shifts its contents with every clock pulse during the positive edge of the pulse transition. If we want to control the shift so that it occurs only with certain pulses but not with others, we must control CLK input of the register. Shift registers can be used for converting serial data to parallel, and vice versa. If we have access to all the flip-flop outputs of a shift register, then information entered serially by shifting can be taken out in parallel from the outputs of the flip-flops. If a parallel-load capability is added to a shift register, then data entered in parallel can be taken out in serial fashion by shifting the data stored in the register.

There are five basic types of shift registers:

- Serial In - Serial Out.

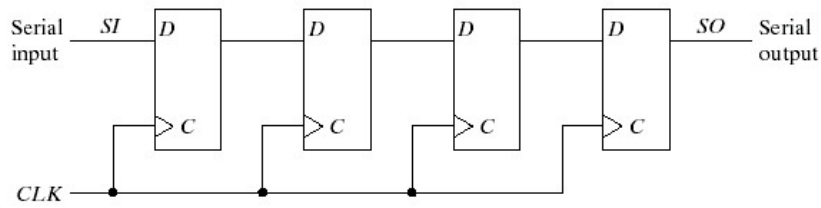


Figure 12.1: 4-bit Shift Register

- Serial In - Parallel Out.
- Parallel In - Serial Out.
- Parallel In - Parallel Out.
- Bidirectional shift registers.

Some shift registers provide the necessary input and output terminals for parallel transfer. They may also have both shift-right and right-shift capabilities. The most general shift register has all the following capabilities:

1. 1. A clear control to clear the register to 0.
2. 2. A CP (or CLK) input for clock pulses to synchronize all operations.
3. 3. A shift-right control to enable the shift-right operation and the serial input and output lines associated with the shift right.
4. 4. A shift-left control to enable the shift-left operation and the serial input and output lines associated with the shift left.
5. 5. A parallel-load control to enable a parallel transfer and the n input lines associated with the parallel transfer.
6. 6. n parallel output lines.
7. 7. A control state that leaves the information in the register unchanged even though clock pulses are continuously applied

## 12.4 Lab Work

### 12.4.1 Parts List:

1. 1- module KL-33008
2. 2- 74LS164 (8-bit shift register:serial In-Parallel Out Register)
3. 3- 74LS165 (8-bit Parallel -to-Serial Shift Register)
4. 4- Two 74LS194 ICs (bidirectional shift Register)

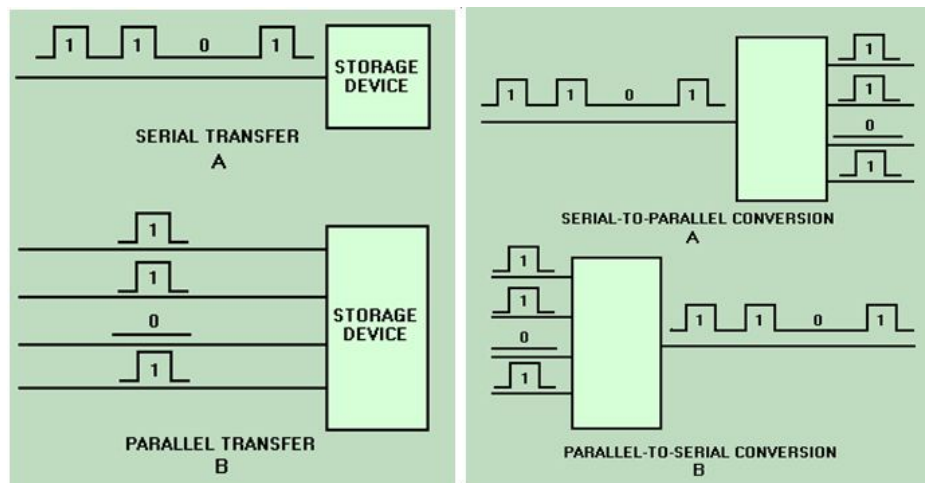


Figure 12.2: Modes of data transfer

### 12.4.2 Serial-In Parallel-Out Shift Register

The 74x164 is an 8 bit SI/PO shift register. Serial data is entered through a 2 AND gate synchronous with the LOW to high transition of the clock.

- a) Derive the functional table for the 74x164 and verify it experimentally.
- b) Show that the 74x164 acts as SI/PO shift register. Enter the data 11001010 serially after how many clock pulses can you get the data at the output simultaneously. Fill in the timing diagram below

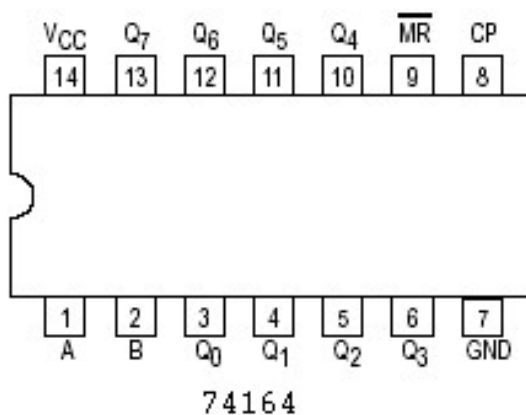


Figure 12.3: Pin specifications of 74164 IC

### 12.4.3 Parallel-In Series-out/Serial-In Serial-out Register

The 74x165 is an 8 bit parallel load or serial in with serial output taken at the last bit Q7.



- Derive the functional table for the 74x165 and verify it experimentally.
- Show that the 74x165 acts as PI/SO shift register. Enter the data 11001010 in parallel mode, after how many clock pulses can you get the data at the output serially.
- Repeat part (b) but use SI/SO mode and count the pulses needed to take the data at the output.

#### **12.4.4 Bidirectional Shift Register**

The 74LS194 is a high speed 4-bit bidirectional universal shift register. It is useful in a wide variety of applications. It may be used in serial-serial, shift left, shift right, serial-parallel, parallel-serial and parallel-parallel data register transfer.

- Connect the 74194 to the breadboard with appropriate inputs and outputs.
- Fill in the timing diagram below

### **12.5 Results and Discussion:**

Excess-3 code is a 4-bit un-weighted code and can be obtained from the corresponding value of BCD code by adding three to each coded number. Excess-3 code is self complementing in nature because 1's complement of the coded number yields 9's complement of number itself.

### **12.6 Conclusion:**

various types of shift register have been implemented and verified using ICs.

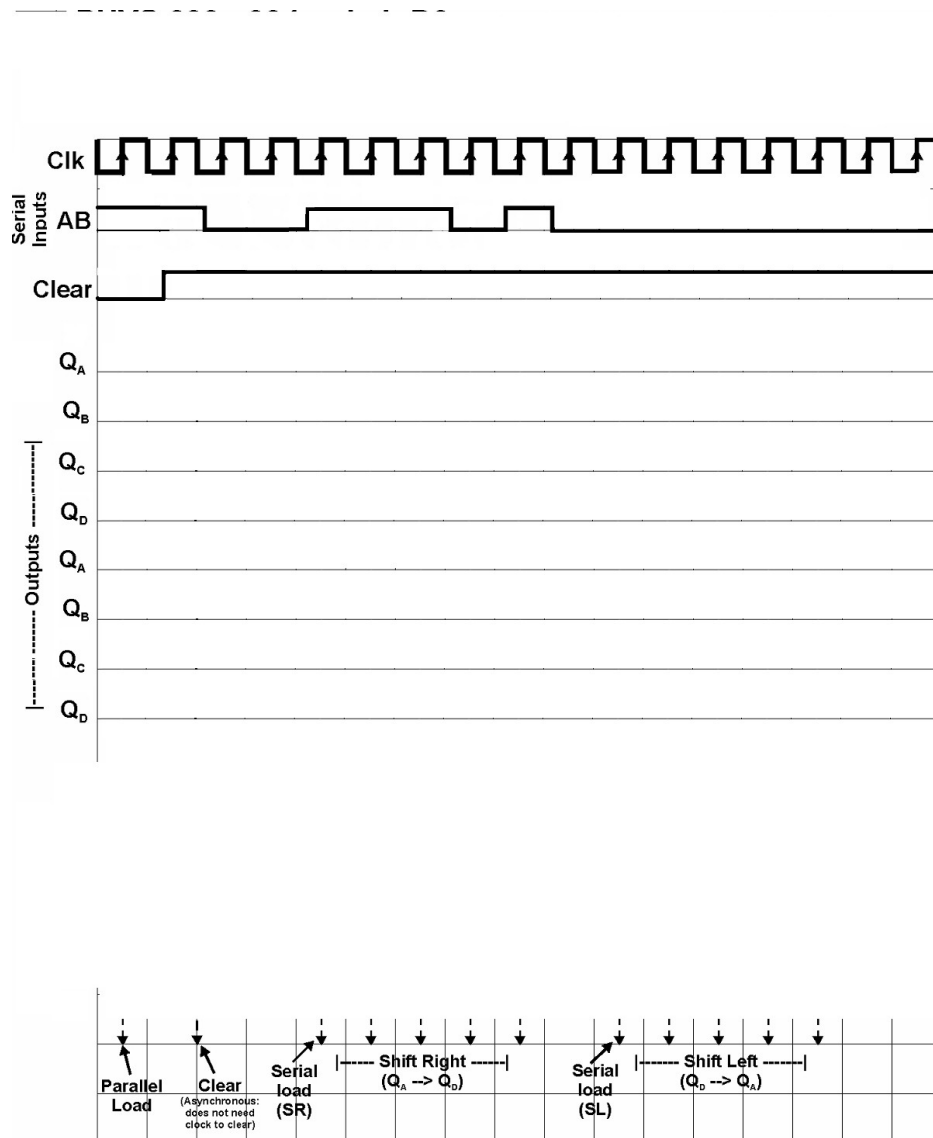


Figure 12.4: Shift Register Timing diagram

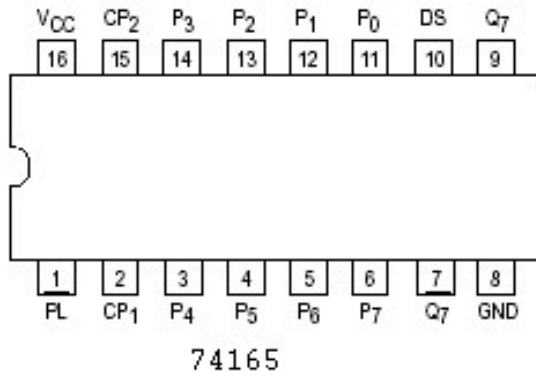


Figure 12.5: Shift Register 74165 IC pin specifications

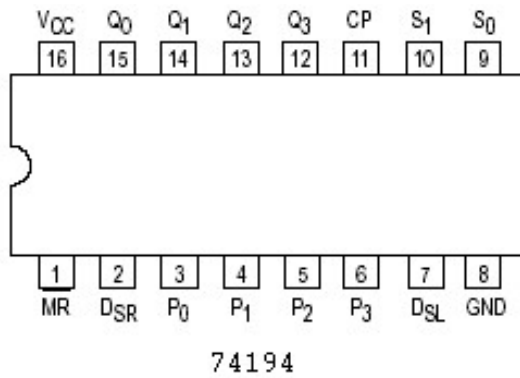


Figure 12.6: Shift Register 74194 IC pin specifications

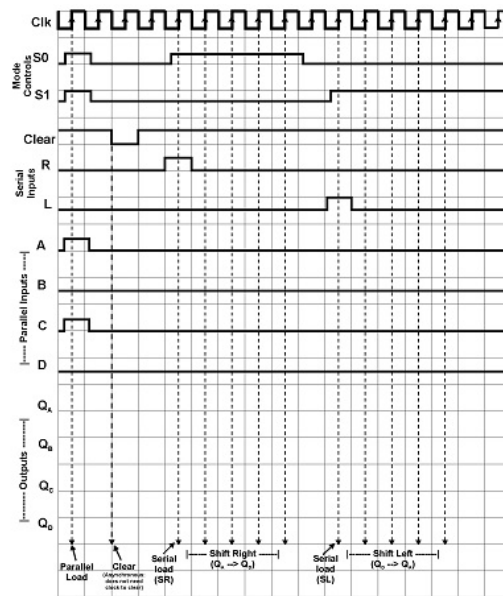


Figure 12.7: Shift Register 74194 IC Timing Diagram

# Lab 13: Counters

---

## 13.1 Objectives

- Introduction to counters. Design and applications.

## 13.2 Background

A counter is a sequential circuit that goes through a prescribed sequence of states upon the application of input pulses. The input pulses (count pulses) of the counter is clock pulses, or from some external source, and may occur at prescribed intervals of time or at random.

### 13.2.1 Classification of Counters

Digital counters are available in a variety of styles. Their characteristics, features, differences, and modes of operation should be observed.

- Asynchronous vs. synchronous.
- Binary vs decade.
- Up vs down.
- Presetable.
- Asynchronous load vs synchronous load.
- Asynchronous clear vs synchronous clear.
- Cascading.
- Counters as binary dividers.
- Divided by N.

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

1. Asynchronous (ripple) counters – the first FF is clocked by the external clock pulse, and then each successive FF is clocked by the Q or Q̄ output of the previous FF.
2. Synchronous counters – all FFs are simultaneously triggered by the same clock.

The counter follows the binary number sequence or other sequence of states. A counter that follows the binary sequence is called a binary counter. An n-bit binary counter consists of n flip-flops and can count in binary from 0 to  $2^n - 1$ .

### 13.2.2 Binary Ripple Counter

A binary ripple counter consists of a series connection of complementing flip flops, with the output of each flip flop connected to the C input of the next higher-order flip-flop. The flip flop holding the least significant bit receives the incoming count pulses. The diagram of a 4-bit binary ripple counter is shown in figure 7.1. The small circle in the C input indicates that the flip-flop complements during a negative-going transition or when output to which it is connected goes from 1 to 0.

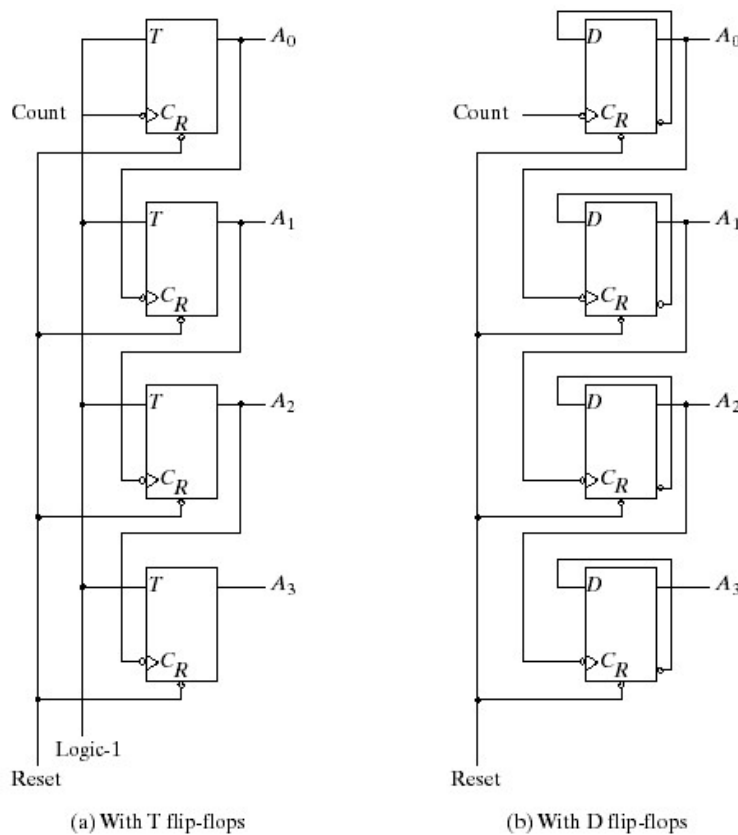


Figure 13.1: 4-bit Binary Ripple Counter

### 13.2.3 BCD Ripple Counter

A decimal counter follows a sequence of ten states and returns to 0 after the count of 9. Such counter must have at least four flip-flops to represent each decimal digit, since a decimal digit is represented by a binary code with at least four bits. The sequence of states in a decimal counter is indicated by the binary code used to represent a decimal digit. If BCD is used, the sequence of states is as shown in the state diagram of figure 7.2. This is similar to a binary counter, except that the state after 1001 (code for decimal 9) is 0000 (code for decimal 0).

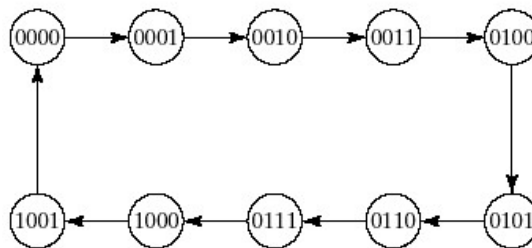


Figure 13.2: 4-bit Binary Ripple Counter

The logic diagram of a BCD ripple counter is shown in figure 7.3. the four outputs are designed by the letter symbol W with a numeric subscript equal to the binary weight of the corresponding bit in the BCD code. The flip-flops trigger on the negative edge. Note that the output Q1 is applied to the C inputs of both Q2 and Q8 and the output of Q2 is applied to the C input of Q4. The J and K inputs are connected either to a permanent 1 signal or to outputs of flip-flops, as shown in the diagram.

The following are the conditions for each flip-flop state transition:-

1. Q1 is complemented on the negative edge of every count pulse.
2. Q2 is complemented if Q8 = 0 and Q1 goes from 1 to 0. Q2 is cleared if Q8 = 1 and Q1 goes from 1 to 0.
3. Q4 is complemented when Q2 goes from 1 to 0.
4. Q8 is complemented when Q4Q2 = 11 and Q1 goes from 1 to 0. Q8 is cleared if either Q4 or Q2 is 0 and Q1 goes from 1 to 0.

### 13.3 Prelab

Derive the truth table and draw a schematic diagram for each experimental part.

### 13.4 Required Equipment:

- 74x93 4-bit binary counter.

- 74x160 synchronous BCD counter.
- 74x190 up/down synchronous BCD counter.
- 74x32 2-input quad OR gate.
- 74x04 hex INV gate.
- 74x00 2-input quad NAND gate.

## 13.5 Procedures:

### 13.5.1 4-bit binary Asynchronous Counter

The 7493 is a 4 stage asynchronous counter containing high speed FF. Features a master reset ( MR1 & MR2 ) to override the clock and force all outputs low. The 74LS93 can be used as:

1. A 4-bit ripple counter - the output Q0 must be externally connected to input CP1. The input count pulses are applied to input CP0. outputs are taken at Q0 to Q3.
2. A 3-bit ripple counter, the input count pulses are applied to input CP1. The outputs are available at Q1 to Q3.

Install the 74x93 in the board and connect the master reset inputs to switches, and the outputs to the respective inputs of a 74x47.

- Derive the functional table of the 74x93, and verify it experimentally.
- Set the clock to its lowest frequency of operation and connect it to the clock input of the 74x93.
- Verify that the 74x93 behaves as a divided by 16 counter (to count 0,1,2,â€¦,15).
- Verify that the 74x93 can also behave as a divided by 8 counter (to count 0,1,2,â€¦,7).
- Design the 74x93 as a divided by 4 counter (to count 0,1,2,3) without any SSI logic.
- Design the 74x93 as a divided by 9 counter (to count 0,1,2,â€¦,8) without any SSI logic.

Note: a counter output can be displayed using the 7-segment display as follows:

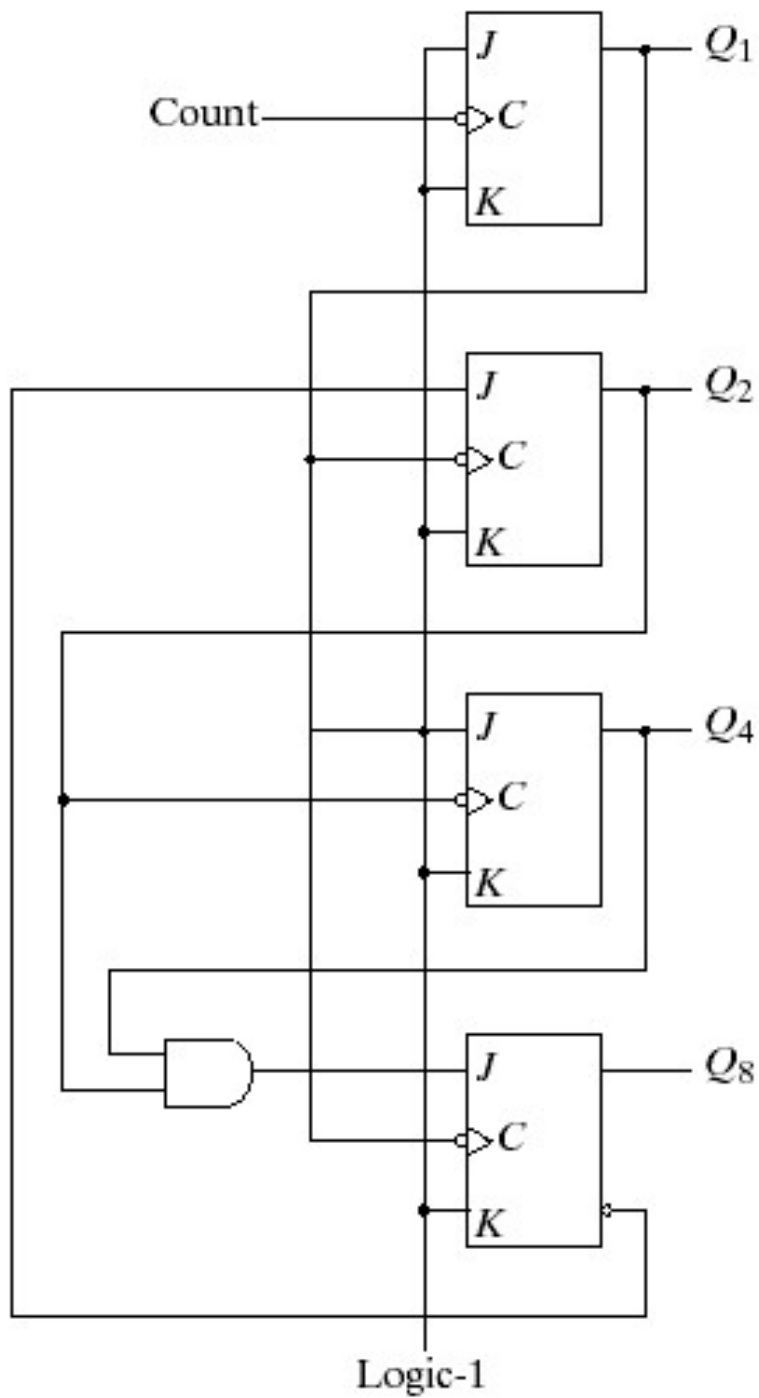


Figure 13.3: BCD Ripple Counter



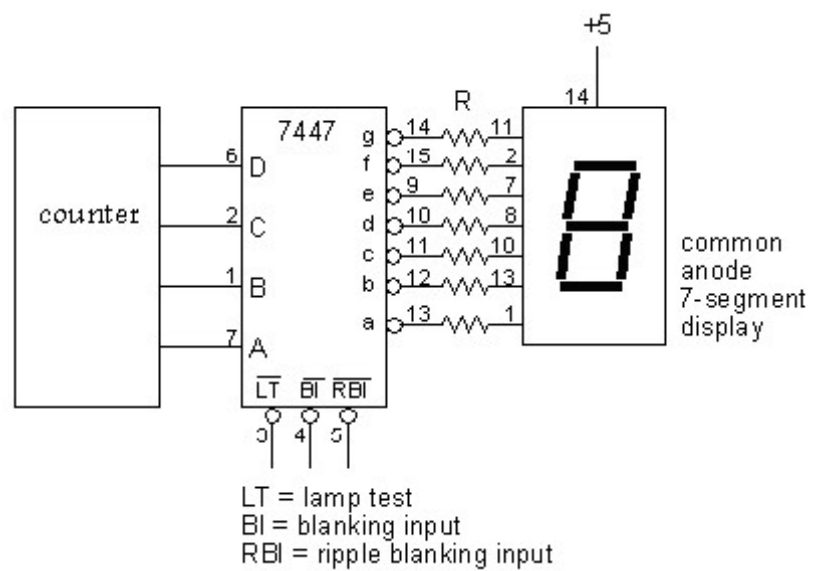


Figure 13.4: 7-Segment Display with Counter

# Lab 14: Synchronous Counter

---

## 14.1 Aim:

Realization of 3-bit synchronous counter design.

## 14.2 Apparatus Required:

IC 7408, IC 7476, IC 7400, IC 7432 etc.

## 14.3 Procedure:

- Connections are made as per circuit diagram.
- 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.
- Verify the Truth table .

## 14.4 Counter Circuit

## 14.5 Synchronous BCD Counter

The 74 x 160 is a high-speed synchronous decade counter with option for applications in programmable loading / dividers array. Install the 74 x 160 in the place where you had the 74 x 93 and connect pins 1, 3, 4, 5, 6, 7, 9 and 10 to switches and the outputs to the decoder, verify load clear & enable input condition from data sheets.

- Derive the functional table of the 74 x 160, and verify it experimentally.
- Arrange the 74 x 160 as a decade counter.
- Design the 74 x 160 as a divided by 8 using only a single INV gate.

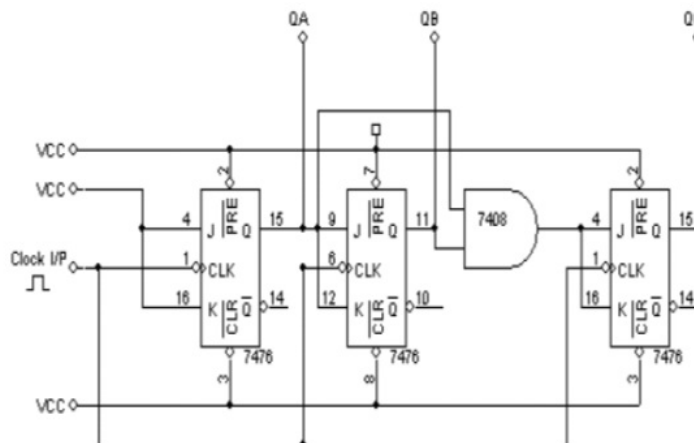


Figure 14.1: 3-bit synchronous counter circuit

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	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0

Figure 14.2: 3-bit synchronous counter truth table

- Design the 74 x 160 as a divided by 6 and program the counter to have the sequence 1,2,3,4,5,6,1,2,3,â€¦. use only a single NAND gate.

### 14.5.1 Synchronous BCD Up/Down Counter

The 74x190 is a decade counter featuring synchronous counting and asynchronous presetting. The preset feature allows the 74x190 to be used in programmable dividers. In addition the 74x190 features up/down (u/d) counter operation.

- Derive the functional table for the 74x190 and verify it experimentally.
- Arrange the 74x190 as a decade up, and then down counter.

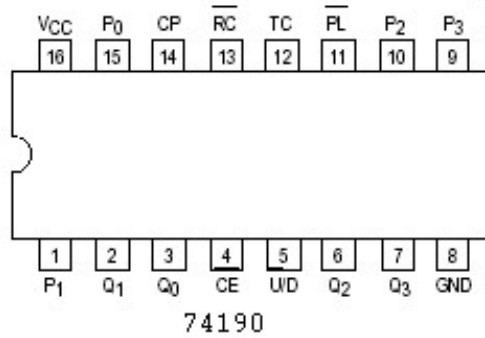


Figure 14.3: Synchronous BCD counter 74160 IC pin specification

- Design the 74x190 as a divided by 7 down counter having the sequence 7,6,5,4,3,2,1,7,6,5, use only a 74x32.
- Design the 74x190 as a divided by 6 up counter having the sequence 3,4,5,6,7,8,3,4,5,6, use only a single NAND gate.

Note: Show schematic diagram for each circuit connections in each part.

## 14.6 Summary