Lab 1: Introduction to Combinational Design

1.1 Introduction

The purpose of this experiment is to introduce you to the basics of logic gates, positive/negative logic and gate behavior. In this lab, you will test the behavior of several of the basic logic gates and you will connect several logic gates together to create simple circuits.

1.2 Hardware Requirement

a. Equipments - Digital IC Trainer Kit
b. Discrete Components - 74LS00 Quad 2-Input NAND gate
   74LS02 Quad 2-Input NOR gate
   74LS04 Quad 2-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

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](image-url)
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 solderless 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

<table>
<thead>
<tr>
<th>Gate</th>
<th>Description</th>
<th>Truth Table</th>
<th>Logic Symbol</th>
<th>Pin Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>OR</td>
<td>The output is active high if any one of the input is in active high state, Mathematically, $Q = A+B$</td>
<td>A</td>
<td>B</td>
<td>Output Q</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>AND</td>
<td>The output is active high only if both the inputs are in active high state, Mathematically, $Q = A \cdot B$</td>
<td>A</td>
<td>B</td>
<td>Output Q</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>NOT</td>
<td>In this gate the output is opposite to the input state, Mathematically, $Q = A$</td>
<td>A</td>
<td>Output Q</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>NOR</td>
<td>The output is active high only if both the inputs are in active low state, Mathematically, $Q = A+B$</td>
<td>A</td>
<td>B</td>
<td>Output Q</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
### NAND

The output is active high only if any one of the input is in active low state, Mathematically,

\[ Q = A \cdot B \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Output Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</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>
</tbody>
</table>

### EXOR

The output is active high only if any one of the input is in active high state, Mathematically,

\[ Q = A \oplus B \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Output Q</th>
</tr>
</thead>
<tbody>
<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>
</tbody>
</table>

### 1.3.2 Pin Diagram of 3-Input Gates

![3-Input Gates Diagram](image)

### 1.4 Pre-lab Questions

1. What is meant by an IC and explain the term DIP and IC 74LSxx.
2. 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?
3. A logic circuit requires HIGH on all its inputs to make the output HIGH. What type of logic circuit is it?
4. 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.
5. Which logic gate is used as a two-bit adder?
6. Which logic gate acts as an Enable/Inhibit Device?
1.5 Lab Procedure

1.5.1 **Experiment #1** A manufacturing plant uses two tanks to store a certain liquid chemical that is required in a manufacturing process. Each tank has a sensor that detects when the chemical level drops to 25% of full. The sensor produces a 5V level when the tanks are more than one-quarter full. When the volume of chemical in a tank drops to one-quarter full, the sensor puts out a 0V level. It is required that a single LED on an indicator panel show when both tanks are more than one-quarter full. Show how a NAND gate can be used to implement this function.

![Diagram of tanks and sensors](image1.png)

**Figure 1.1**

Prepare a truth table, build the circuit and identify the IC pin connection and also give its Boolean equation.

1.5.2 **Experiment #2** Refer to the textbook and data sheet to determine the following:
   a. Draw the logic symbol for a 3-input OR gate using 2-input OR gate. Label the inputs A, B, & C and the output X.
   b. Write the Boolean expression for the 3-input OR gate.
   c. Complete the truth table for a 3-input OR gate.

1.5.3 **Experiment #3** Using TTL logic, design a circuit using individual logic gates to implement the following function. Prepare a truth table, build the circuit, and identify all IC pin connections and verify the circuit operation.

\[ f(x, y, z) = (x \cdot y)' + (x \cdot y \cdot z) \]

**Truth Table**

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
1.6 Post-lab Questions

1. If the two waveforms A and B shown in figure 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.

![Timing Diagram](image)

Figure 1.2

2. Implement the basic gates using Universal gates.

3. Implement NOR using NAND gates and NAND gate using NOR gates.

4. As part of an aircraft’s functional monitoring system, a circuit is required to indicate the status of the landing gears prior to landing. A green LED display turns on if all three gears are properly extended when the “gear down” switch has been activated in preparation for landing. A red LED display turns on if any of the gears fail to extend properly prior to landing. When a landing gear is extended, its sensor produces a LOW voltage. When a landing gear is retracted, its sensor produces a HIGH voltage. Implement a circuit to meet this requirement and give its boolean equation.

![Landing Gear Diagram](image)

5. What type of logic gate does this logic circuit configuration represent

- NAND Gate
- EXOR Gate
- NOR Gate
- EXNOR Gate
Lab Report

Each individual will be required to submit a lab report. Use the format specified in the "Lab Report Requirements" document available on the class web page. Be sure to include the following items in your lab report:

- Lab cover sheet with staff verification for circuit diagram
- Answer the pre-lab questions
- Complete paper design for all three designs including K-maps and minimized equations and the truth table for each of the output signals.
- Answer the post-lab questions

Grading

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-lab Work</td>
<td>20 points</td>
</tr>
<tr>
<td>Lab Performance</td>
<td>30 points</td>
</tr>
<tr>
<td>Post-lab Work</td>
<td>20 points</td>
</tr>
<tr>
<td>Lab report</td>
<td>30 points</td>
</tr>
</tbody>
</table>

For the lab performance - at a minimum, demonstrate the operation of all the circuits to your staff incharge

The lab report will be graded as follows (for the 30 points):

- Logic Diagram along with IC no. and Pin no. for each logic gates  15 points
- K-maps and simplified expressions for all outputs and its truth table  15 points
Lab 2: Design of Binary Adders

2.1 Introduction

The purpose of this experiment is to introduce the design of simple combinational circuits, in this case half adders and full adders. The student should also be able to design an \( n \)-bit ripple carry adder using \( n \) full adders.

2.2 Hardware Requirement

a. Equipments              -    Digital IC Trainer Kit
b. Discrete Components - 74LS08 Quad 2-Input AND gate
                          74LS32 Quad 2-Input OR gate
                          74LS86 Quad 2-Input XOR

2.3 Background

One of the things that led to the “digital revolution” was the ability to perform complex mathematical operations using a computer. During this lab we will see how it is possible to design a circuit to perform one such operation – adding two numbers together.

We will be using two types of adders for this lab and throughout the semester. A half adder adds two bits together and therefore has possible sums of 0, 1, or 2. A full adder adds a carry-in bit to two bits and has possible sums of 0, 1, 2, or 3. See Figure 2.1

To create an \( n \) bit ripple carry adder, you may connect \( n \) full adders together in the fashion of Figure 2.2. The carry-out from a full adder becomes the carry-in for the next significant bit in the
next full adder. In Figure 2.3 you can see the interaction of the input and output signals shown in Figure 2.2.

![Fig. 2.2 2-bit ripple carry adder](image)

![Fig. 2.3 2-bit addition](image)

2.4 Pre-lab Questions

1. How many 74LS283 adders would be required to add two binary numbers each representing decimal numbers up through 100010?
2. Two 4-bit numbers (1101 and 1011) are applied to a 4-bit parallel adder. The input carry is ‘1’. Determine the sum (\(\sum\)) and output carry.
3. What is the difference in mapping a POS expression and an SOP expression?
4. Using K-Map, convert the following SOP in to a Canonical SOP expression and a Minimum POS expression.
5. In a certain chemical-processing plant, a liquid chemical is used in a manufacturing process. The chemical is stored in three different tanks. A level sensor in each tank produces a HIGH voltage when the level of chemical in the tank drops below a specified point. Design a circuit that monitors the chemical level in each tank and indicates when the level in any two tanks drops below the specified point. Write the Boolean SOP and POS expression for the above circuit.

2.5 Lab Procedure

2.5.1 Experiment #1  Half Adder

Using TTL logic, design a circuit using individual logic gates to implement the adder. Your circuit should be capable of taking two 1-digit inputs and displaying the resulting sum. For example:

\[ 1 + 0 = 1 \]

And in general:  \[ X_0 + Y_0 = C_1 S_0 \]

Where C1 is the carry to the two's position.
a. Draw the block diagram and prepare separate truth tables for the 1’s and the 2’s positions for all four combination.

b. Derive the Boolean Equation using K-Map

c. Identify all IC pin connections (i.e., prepare a logic diagram with pin numbers and an IC diagram)

```
A
\downarrow
\uparrow
B
\downarrow
\downarrow
C
```

d. Build the circuit and verify the operation.

2.5.2 Experiment #2 Full Adder

Using TTL logic, design a circuit using individual logic gates to implement the adder.

Your circuit should be capable of taking carry input ‘1’ and two 1-bit inputs and displaying the resulting

a. Draw the block diagram and prepare separate truth tables for all eight combination.

b. Derive the Boolean Equation using K-Map

c. Identify all IC pin connections (i.e., prepare a logic diagram with pin numbers and an IC diagram)

```
A
\downarrow
\uparrow
B
\downarrow
\downarrow
C
```

d. Build the circuit and verify the operation.

2.6 Post-lab Questions

1. Design a 3-bit ripple carry adder using full adders as building blocks. Include the carry-in to the LSB. There should be 7 inputs: (A2A1A0) one 3-bit operand, (B2B1B0) another 3-bit operand, and Cin, the carry-in to the LSB of the adder. There should be 4 outputs.

2. Determine the sum generated by the 4-bit parallel adder and show the intermediate carries when the binary numbers 1101 and 0111 are being added.

3. Show how two 74LS83A adders can be connected to form an 8-bit parallel adder. Show output bits for the following 8-bit input numbers.

\[ A8A7A6A5A4A3A2A1 = 10111001 \quad \text{and} \quad B8B7B6B5B4B3B2B1 = 10011110 \]
4. Design a circuit that is use to subtract two 2-bit numbers. Draw the logic diagram using the truth table and Boolean equation (from K-Map) and design the full adder using two Half subtractor.

5. Give the block diagram of 5-bit Parallel Binary Adder.

Lab Report

Each individual will be required to submit a lab report. Use the format specified in the "Lab Report Requirements" document available on the class web page. Be sure to include the following items in your lab report:
- Lab cover sheet with staff verification for circuit diagram
- Answer the pre-lab questions
- Complete paper design for all three designs including K-maps and minimized equations and the truth table for each of the output signals.
- Answer the post-lab questions

Grading

<table>
<thead>
<tr>
<th></th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-lab Work</td>
<td>20</td>
</tr>
<tr>
<td>Lab Performance</td>
<td>30</td>
</tr>
<tr>
<td>Post-lab Work</td>
<td>20</td>
</tr>
<tr>
<td>Lab report</td>
<td>30</td>
</tr>
</tbody>
</table>

For the lab performance - at a minimum, demonstrate the operation of all the circuits to your staff incharge

The lab report will be graded as follows (for the 30 points):
- Logic Diagram along with IC no. and Pin no. for each logic gates 15 points
- K-maps and simplified expressions for all outputs and its truth table 15 points
Lab 3: Design of Magnitude Comparator

3.1 Introduction

The purpose of this experiment is to introduce the design of Magnitude Comparator. The student should also be able to design a combinational circuit to compare the magnitude of two 2-bit numbers.

3.2 Hardware Requirement

a. Equipments - Digital IC Trainer Kit
b. Discrete Components -
   - 74LS02 Quad 2-Input NOR gate
   - 74LS04 Quad 2-Input NOT gate
   - 74LS08 Quad 2-Input AND gate
   - 74LS11 Triple 3-Input AND gate
   - 74LS32 Quad 2-Input OR gate
   - 74LS86 Quad 2-Input XOR

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 1-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 1-bit comparator below.

3.3.1 1-bit Comparator

The operation of a 1-bit digital comparator is given in the following Truth Table.

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A B</td>
<td>A &gt; B</td>
</tr>
<tr>
<td>0 0</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td>0</td>
</tr>
<tr>
<td>1 0</td>
<td>1</td>
</tr>
<tr>
<td>1 1</td>
<td>0</td>
</tr>
</tbody>
</table>
You may notice two distinct features about the comparator from the above truth table. Firstly, the circuit does not distinguish between either two "0" or two "1"s as an output A = B is produced when they are both equal, either A = B = "0" or A = B = "1". Secondly, the output condition for A = B resembles that of a commonly available logic gate, the Exclusive-NOR or Ex-NOR gate giving $Q = A \oplus B$

Digital comparators actually use Exclusive-NOR gates within their design for comparing the respective pairs of bits in each of the two words with single bit comparators cascaded together to produce Multi-bit comparators so that larger words can be compared.

3.3.2 2-bit Comparator:

To design a comparator for 2 bit binary numbers A (A1A0) and B (B1B0) we do the following steps:

For a 2-bit comparator we have four inputs $A_1A_0$ and $B_1B_0$ and three output $E$ (is 1 if two numbers are equal) $G$ (is 1 when $A > B$) and $L$ (is 1 when $A < B$)

3.3.3 Magnitude Comparator for n-bit

Here we use simpler method to find $E$ (called X) and $G$ (called Y) and $L$ (called Z)

- $A = B$ if all $A_i = B_i$
For 2-Bit: It means $X_0 = A_0B_0 + A'_0B'_0$ and

$$X_1 = A_1B_1 + A'_1B'_1$$

If $X_0=1$ and $X_1=1$ then $A_0=B_0$ and $A_1=B_1$

Thus, if $A=B$ then $X_0X_1 = 1$ it means

$$X = (A_0B_0 + A'_0B'_0)(A_1B_1 + A'_1B'_1)$$ since $(x O y)' = (xy + x'y')$

$$X = (A_0O B_0)' (A_1O B_1)' = ((A_0O B_0) + (A_1O B_1))'$$

It means for $X$ we can NOR the result of two exclusive-OR gates

- **For $A > B$, $A_1 > B_1$ OR $A_1 = B_1$ and $A_0 > B_0$**

<table>
<thead>
<tr>
<th>A1</th>
<th>B1</th>
<th>X1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</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>0</td>
</tr>
</tbody>
</table>

if $A_1=B_1$ ($X_1=1$) then $A_0$ should be 1 and $B_0$ should be 0

<table>
<thead>
<tr>
<th>A0</th>
<th>B0</th>
<th>X0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</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>0</td>
</tr>
</tbody>
</table>

For $A > B$: $A_1 > B_1$ or $A_1 = B_1$ and $A_0 > B_0$

It means $Y = A_1B'_1 + X_1A_0B'_0$ should be 1 for $A > B$
For A < B, A1 < B1 or A1=B1 and A0 < B0

\[ Z = A'1B1 + X1A'0B0 \]

The procedure for binary numbers with more than 2 bits can also be found in the similar way. For example the 4-bit magnitude comparator, in which

\[
\begin{align*}
(A = B) &= x3x2x1x0 \\
(A > B) &= A3B'3 + x3A2B'2 + x3x2A1B'1 + x3x2x1A0B'0 \\
(A < B) &= A'3B3 + x3A'2B2 + x3x2A'1B1 + x3x2x1A'0B0
\end{align*}
\]

3.4 Pre-lab Questions

1. Apply each of the following sets of binary numbers to the comparator inputs in Figure 3.3, and determine the output by following the logic levels through the circuit.

- (a) 10 and 10
- (b) 11 and 10

![Figure 3.3](image)

2. Determine the A = B, A > B, and A < B output for the input numbers shown on the comparator in figure 3.4.

![Figure 3.4](image)
3. Use 74HC85 comparators to compare the magnitude of two 8-bit numbers. Show the comparators with proper interconnections.

3.5 Lab Procedure

3.5.1 Experiment #1  Using TTL logic, design a circuit using individual logic gates to implement the 2-Bit Comparator. Your circuit should be capable comparing the magnitude of A and B and displaying the result for A = B, A > B and A < B.

   a. Draw the logic diagram and prepare separate truth tables for all the combination.
   b. Build the circuit using the general equation and verify the result.

3.6 PostLab questions

1. For each set of binary numbers, determine the output states for the comparator and also draw its logic diagram along with the general equation.
   
   (a) A3A2A1A0 = 1100          B3B2B1B0 = 1001
   (b) A3A2A1A0 = 1000          B3B2B1B0 = 1001
   (c) A3A2A1A0 = 0100          B3B2B1B0 = 1001

2. Expand the circuit in figure 3.4 to 16-bit comparator.

3. Show the logic required to convert a 5-bit binary number to Gray code, and use that logic to convert the following binary numbers to Gray code.

   (a) 10101    (b) 11101    (c) 100101

Lab Report

Each individual will be required to submit a lab report. Use the format specified in the "Lab Report Requirements" document available on the class web page. Be sure to include the following items in your lab report:
- Lab cover sheet with staff verification for circuit diagram
- Answer the pre-lab questions
- Complete paper design for all three designs including K-maps and minimized equations and the truth table for each of the output signals.
- Answer the post-lab questions
Lab 4: Design of Encoders and Decoders

4.1 Introduction

The purpose of this experiment is to introduce you to the basics of Encoders and Decoders. In this lab, you have to implement Priority Encoder and the Boolean function using Decoders.

4.2 Hardware Requirement

a. Equipments - Digital IC Trainer Kit
b. Discrete Components - 74LS00 Quad 2-Input NAND gate
   74LS04 Quad 2-Input NOT gate
   74LS08 Quad 2-Input AND gate
   74LS11 Triple 3-Input AND gate
   74LS32 Quad 2-Input OR gate

4.3 Background

4.3.1 Encoder

Encoder takes all the data inputs one at a time and converts them to a single encoded output. Then, it is a multi-input data line, combinational logic circuit that converts the logic level "1" data at its inputs to an equivalent binary code at its output. Generally encoders produce outputs of 2-bit, 3-bit or 4-bit codes depending upon the number of data input lines and a "n-bit" encoder has 2n input lines with common types that include 4-to-2, 8-to-3 and 16-to-4 line configurations. Encoders are available to encode either a decimal or hexadecimal input pattern to typically a binary or B.C.D. output code.

![Figure 4.1 4-to-2 bit Encoder](image)

One of the main disadvantages of standard encoders is that they can generate the wrong output code when there is more than one input present at logic level "1". For example, if we make inputs D1 and D2 HIGH at logic "1" at the same time, the resulting output is neither at "01" or at "10" but will be at "11" which is an output code that is different to the actual input present. One simple way to overcome this problem is to "Prioritize" the level of each input pin and if there was more than one input at logic level "1" the actual output code would only correspond to the input with the highest designated priority. Then this type of encoder is known as Priority Encoders or P-encoder.
4.3.2 8-to-3 Bit Priority Encoder

Priority encoders are available in standard IC form and the TTL 74LS148 is an 8 to 3 bit priority encoder which has eight active LOW (logic "0") inputs and provides a 3-bit code of the highest ranked input at its output. Priority encoders output the highest order input first for example, if input lines "D2", "D3" and "D5" are applied simultaneously the output code would be for input "D5" ("101") as this has the highest order out of the 3 inputs. Once input "D5" had been removed the next highest output code would be for input "D3" ("011"), and so on.

![Figure 4.2 8-to-3 Bit Priority Encoder](image)

4.3.3 Binary Decoders

A Decoder is the exact opposite to that of an "Encoder" we looked at in the last tutorial. It is basically, a combinational type logic circuit that converts the binary code data at its input into one of a number of different output lines, one at a time producing an equivalent decimal code at its output. Binary Decoders have inputs of 2-bit, 3-bit or 4-bit codes depending upon the number of data input lines, and a "n-bit" decoder has $2^n$ output lines. Typical combinations of decoders include, 2-to-4, 3-to-8 and 4-to-16 line configurations. Binary Decoders are available to "decode" either a Binary or BCD input pattern to typically a Decimal output code.

![Figure 4.3 2-to-4 Binary Decoders](image)

In this simple example of a 2-to-4 line binary decoder, the binary inputs A and B determine which output line from D0 to D3 is "HIGH" at logic level "1" while the remaining outputs are held "LOW" at logic "0". Therefore, whichever output line is "HIGH" identifies the binary code present at the input, in other words it "de-codes" the binary input and these types of binary decoders are commonly used as Address Decoders in microprocessor memory applications.
4.4 Pre-lab Questions

1. Determine the logic required to decode the binary number 1011 by producing a HIGH level on the output.
2. A 3-line-to-8 decoder can be used for octal-to-decimal decoding. When a binary 101 is on the inputs, which output line is activated?
3. How many 74HC154 1-of-16 decoders are necessary to decode a 6-bit binary number?
4. Explain the role of Encoder in code converters?
5. What is the use of Priority Encoder?
6. State the significant application of Encoder and Decoder.
7. Differentiate between Decoder and Demultiplexer.

4.5 Lab Procedure

4.5.1 Experiment #1 Priority Encoder

A priority encoder is an encoder circuit with priority. If two or more inputs are equal to 1, the input with the highest priority will take precedence. The truth table of a 4-input priority encoder is shown below. An output V is added which is set to 1 when one or more inputs are equal to 1, or 0 otherwise. The other two outputs are not inspected when V equals 0 and are specified as don’t-care conditions. Note that whereas X’s in output columns represent don’t-care conditions, the X’s in the input columns are useful for representing a truth table in compact form. For example, input X100 represents the two input combination 0100 and 1100.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>D0 0 1</td>
<td>x X X 0</td>
</tr>
<tr>
<td>D1 0 0</td>
<td>y X 0 1</td>
</tr>
<tr>
<td>D2 X X</td>
<td>z 1 0 1</td>
</tr>
<tr>
<td>D3 X X</td>
<td>1 1 1</td>
</tr>
</tbody>
</table>

(a) Derive the minimal SOP expressions for x, y and V using K-Map
(b) Draw the logic diagram and verify the result
4.5.2 Experiment #2  Implement the Half Adder using 4 x 2 decoder. Use AND and NOT gates to construct the decoder to produce active high output and verify the truth table of Full Adder.

a. Draw the block diagram to implement Half Adder using 4x2 Decoder.
b. Draw the Truth Table of 4x2 Decoder and Half Adder
c. Draw the logic diagram using gates and verify the truth table of Half Adder using the 4x2 Decoder.

4. 6 PostLab questions

1. Given the following two functions:

(a) \( f(x, y, z) = \Sigma m(1, 3, 5) \)

(b) \( g(x, y, z) = \Sigma m(0, 1, 4, 5, 6) \)

How do you implement each of them using a single 3x8 decoder with active-low(negated) outputs as shown in Figure 4.6. Assume that fan-in of each gate is limited to 3.

![Figure 4.6](image)

2. Implement Full Adder using 3x8 Decoder for active low output and draw its truth table.

3. Draw the logic symbol, Truth Table and Logic Diagram of

   (a) Decimal to BCD Encoder

   (b) Octal to Binary Encoder

4. Draw the logic symbol of 4x16 Decoder using 3x8 Decoder.

5. Construct the truth table to decode Binary to 7-Segment

Draw the logic diagram for the above using the Boolean equation.
Lab Report

Each individual will be required to submit a lab report. Use the format specified in the "Lab Report Requirements" document available on the class web page. Be sure to include the following items in your lab report:

- Lab cover sheet with staff verification for circuit diagram
- Answer the pre-lab questions
- Complete paper design for all three designs including K-maps and minimized equations and the truth table for each of the output signals.
- Answer the post-lab questions

Grading

<table>
<thead>
<tr>
<th></th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-lab Work</td>
<td>20</td>
</tr>
<tr>
<td>Lab Performance</td>
<td>30</td>
</tr>
<tr>
<td>Post-lab Work</td>
<td>20</td>
</tr>
<tr>
<td>Lab report</td>
<td>30</td>
</tr>
</tbody>
</table>

For the lab performance - at a minimum, demonstrate the operation of all the circuits to your staff incharge

The lab report will be graded as follows (for the 30 points):

- Logic Diagram along with IC no. and Pin no. for each logic gates 15 points
- K-maps and simplified expressions for all outputs and its truth table 15 points
Lab 5: Design of Multiplexer and De-multiplexer

5.1 Introduction

The purpose of this experiment is to introduce you to the basics of Multiplexer and Demultiplexer. In this lab, you have to implement Demultiplexer and Implement the Boolean function using Multiplexer.

5.2 Hardware Requirement

a. Equipments - Digital IC Trainer Kit
b. Discrete Components -
   - 74LS04 Quad 2-Input NOT gate
   - 74LS08 Quad 2-Input AND gate
   - 74LS11 Triple 3-Input AND gate
   - 74LS32 Quad 2-Input OR gate

5.3 Background

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 2n 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)

![Figure 5.1 4x1 Multiplexer](a) Function Table

<table>
<thead>
<tr>
<th>s1</th>
<th>s0</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>I0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>I1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>I2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>I3</td>
</tr>
</tbody>
</table>

(b) Block Diagram
The Boolean function for a 4x1 Multiplexer is: \( Y = I_0 s_1's_0' + I_1 s_1's_0 + I_2 s_1s_0' + I_3 s_1s_0 \)

**5.3.2 Implement the Function using MUX**

\[ F(A,B,C) = \Sigma(0, 3, 6, 7) \] using a 4 x 1 multiplexer.

<table>
<thead>
<tr>
<th>( A )</th>
<th>( B )</th>
<th>( C )</th>
<th>( F )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Figure 5.3 MUX Implementation Table**

\[ F = C' \text{ (so connect } C' \text{ to MUX input 0)} \]
\[ F = C \]
\[ F = C' \]
\[ F = 0 \]
\[ F = 1 \]

**Figure 5.4 Logic Diagram of \( F(A,B,C) \)**
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.

Figure 5.5 De-Multiplexer Block Diagram

Figure 5.6 1-bit 4-output de-multiplexer using a 2x4 binary decoder

Figure 5.7 1x4 DeMUX Logic Symbol

Figure 5.8 1x4 DeMUX –Truth Table
5.4 Pre-lab Questions

1) Implement a Full Adder logic using suitable MUX only?
2) Implement a 4:1 using 2:1 MUX only?
3) Implement a 1:4 DeMux using 1:2 Demux only?
4) Implement a 2-input EXOR function using suitable multiplexer?
5) Implement a 2-input NAND function using suitable multiplexer?
6) State the significant application of Data selector and data distributor?

5.5 Lab Procedure

5.5.1 Experiment #1 Implement the following Boolean Function $F(x,y,z) = \sum(1,2,6,7)$ using Multiplexer.

   a. Determine the size of the MUX to implement the above Boolean function
   b. Draw the Truth table for the given function.
   c. Draw the block diagram to implement the above function using MUX
   d. Construct the MUX using TTL logic gates and verify the result.

5.5.2 Experiment #2 Design a 1x2 Demux/Data Distributor for active high outputs.

   a. Draw the block diagram and the truth table
   b. Construct the Demux using individual logic gates and verify the truth table.

5.6 PostLab questions

1. Draw the block diagram of 8x1 MUx using two 4x1 MUX and one 2x1 MUX and draw its function table.

2. Implement the function $F(A,B,C,D) = \sum(1,3,4,11,12,13,14,15)$ using MUX. Draw its truth table and its logic symbol.

3. Draw the block diagram of 1x4 DeMUX using 1x2 DeMUX. Draw its truth table.

4. Do the complete paper design for the block diagram given below. Your paper design will have a complete truth table, K-maps and circuit diagrams using the multiplexers and the discrete logic gates. For each MUX, have two designs ready. For this part, the 74LS83 4-bit adder will be used as a 2-bit adder, with the S3 output ($\Sigma$ in 74LS83 datasheet) used as the carry. Design a combinational logic network which accepts the sum and carry from the 2-bit adder and displays the result on a 7-segment display. Figure 5.9 shows a block diagram of the desired circuit.
Lab Report

Each individual will be required to submit a lab report. Use the format specified in the "Lab Report Requirements" document available on the class web page. Be sure to include the following items in your lab report:

- Lab cover sheet with staff verification for circuit diagram
- Answer the pre-lab questions
- Complete paper design for all three designs including K-maps and minimized equations and the truth table for each of the output signals.
- Answer the post-lab questions

Grading

<table>
<thead>
<tr>
<th></th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-lab Work</td>
<td>20</td>
</tr>
<tr>
<td>Lab Performance</td>
<td>30</td>
</tr>
<tr>
<td>Post-lab Work</td>
<td>20</td>
</tr>
<tr>
<td>Lab report</td>
<td>30</td>
</tr>
</tbody>
</table>

For the lab performance - at a minimum, demonstrate the operation of all the circuits to your staff incharge

The lab report will be graded as follows (for the 30 points):

- Logic Diagram along with IC no. and Pin no. for each logic gates 15 points
- K-maps and simplified expressions for all outputs and its truth table 15 points
Lab 6: Introduction to Sequential Design

6.1 Introduction

The purpose of this experiment is to introduce you to the basics of flip-flops. In this lab, you will test the behavior of several flip-flops and you will connect several logic gates together to create simple sequential circuits.

6.2 Hardware Requirement

a. Equipments - Digital IC Trainer Kit
b. Discrete Components -
   - 74LS00 Quad 2-Input NAND gate
   - 74LS02 Quad 2-Input NOR gate
   - 74LS04 Quad 2-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

6.3 Background

Digital electronics 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.

![Sequential Logic Diagram](image)

Figure 6.1 Sequential Logic

The memory 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. 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. There are two types of sequential circuits. Their classification depends on the timing of their signals:

- Synchronous sequential circuits
- Asynchronous sequential circuits

**Synchronous** sequential circuits change their states and output values at discrete instants of time, which are specified by the rising and falling edge of a free-running clock signal. The clock signal is generally some form of square wave as shown in Figure 7.2 below.
From the diagram you can see that the **clock period** is the time between successive transitions in the same direction, that is, between two rising or two falling edges. State transitions in synchronous sequential circuits are made to take place at times when the clock is making a transition from 0 to 1 (rising edge) or from 1 to 0 (falling edge). Between successive clock pulses there is no change in the information stored in memory.

The reciprocal of the clock period is referred to as the **clock frequency**. The **clock width** is defined as the time during which the value of the clock signal is equal to 1. The ratio of the clock width and clock period is referred to as the duty cycle. A clock signal is said to be **active high** if the state changes occur at the clock's rising edge or during the clock width. Otherwise, the clock is said to be **active low**. Synchronous sequential circuits are also known as **clocked sequential circuits**.

Normally Input enable signals of sequential circuit can be of two types

- Level Sensitive or (LATCH)
- Edge Sensitive or (Flip-Flop)

Latches and Flip-flops are one and the same with a slight variation: Latches have level sensitive control signal input and Flip-flops have edge sensitive control signal input. Flip-flops and latches which use this control signals are called synchronous circuits. So if they don't use clock inputs, then they are called asynchronous circuits.

**6.3.1 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 summarising 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 6.1.
### Table 6.1 Flip-flops and their properties

Each of these flip-flops can be uniquely described by its graphical symbol, its characteristic table, its characteristic equation or excitation table. All flip-flops have output signals \( Q \) and \( Q' \).

The **characteristic table** in the third column of Table 7.1 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. The characteristic table for the RS flip-flop shows that the next state is equal to the present state when both inputs \( S \) and \( R \) are equal to 0. When \( R=1 \), the next clock pulse clears the flip-flop. When \( S=1 \), the flip-flop output \( Q \) is set to 1. The equation mark (?) for the next state when \( S \) and \( R \) are both equal to 1 designates an indeterminate next state.

The characteristic table for the JK flip-flop is the same as that of the RS when \( J \) and \( K \) are replaced by \( S \) and \( R \) respectively, except for the indeterminate case. When both \( J \) and \( K \) are equal to 1, the next state is equal to the complement of the present state, that is, \( Q(next) = Q' \).
The next state of the D flip-flop is completely dependent on the input D and independent of the present state.

The next state for the T flip-flop is the same as the present state Q if \( T=0 \) and complemented if \( T=1 \).

The characteristic table is useful during the analysis of sequential circuits when the value of flip-flop inputs are known and we want to find the value of the flip-flop output Q after the rising edge of the clock signal. As with any other truth table, we can use the map method to derive the characteristic equation for each flip-flop, which are shown in the third column of Table 7.1.

During the design process we usually know the transition from present state to the next state and wish to find the flip-flop input conditions that will cause the required transition. For this reason we will need a table that lists the required inputs for a given change of state. Such a list is called the excitation table, which is shown in the fourth column of Table 7.1. There are four possible transitions from present state to the next state. The required input conditions are derived from the information available in the characteristic table. The symbol X in the table represents a "don't care" condition, that is, it does not matter whether the input is 1 or 0.

### 6.4 Pre-lab Questions

1. Describe the main difference between a gated S-R latch and an edge-triggered S-R flip-flop.
2. How does a JK flip-flop differ from an SR flip-flop in its basic operation?
3. Describe the basic difference between pulse-triggered and edge-triggered flip-flops.
4. What is use of characteristic and excitation table?
5. What are synchronous and asynchronous circuits?
6. How many flip flops due you require storing the data 1101?
7. What is propagation delays set up time and hold time?

### 6.5 Lab Procedure

#### 6.5.1 Experiment #1
Study the characteristic of RS Flip-flop constructed using TTL logic gates

![Logic diagram](image)

Figure 6.3 Logic diagram

(a) Draw its logic symbol and write the characteristic equation using K-Map

(b) Verify the Characteristic Table of RS Flip-flop and draw its timing diagram

#### 6.5.2 Experiment #2
Study the characteristic of D Flip-flop constructed using TTL logic gates
(a) Draw its logic symbol and write the characteristic equation using K-Map

(b) Verify the Characteristic Table of RS Flip-flop and draw its timing diagram

6.5.3 Experiment #3 Study the characteristic of JK Flip-flop constructed using TTL logic gates

(a) Draw its logic symbol and write the characteristic equation using K-Map

(b) Verify the Characteristic Table of RS Flip-flop and draw its timing diagram

6.5.4 Experiment #4 Study the characteristic of JK Flip-flop constructed using TTL logic gates

(a) Draw its logic symbol and write the characteristic equation using K-Map

(b) Verify the Characteristic Table of RS Flip-flop and draw its timing diagram
6.5.4 Experiment #5  Verify the function table of IC7474 and IC7473

<table>
<thead>
<tr>
<th>FUNCTION TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>INPUTS</td>
</tr>
<tr>
<td>PRE</td>
</tr>
<tr>
<td>L</td>
</tr>
<tr>
<td>H</td>
</tr>
<tr>
<td>L</td>
</tr>
<tr>
<td>H</td>
</tr>
<tr>
<td>H</td>
</tr>
<tr>
<td>H</td>
</tr>
</tbody>
</table>

Figure 6.7 Pin diagram of D Flip-flop(IC7447)

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>CLK</td>
</tr>
<tr>
<td>L</td>
<td>X</td>
</tr>
<tr>
<td>H</td>
<td>↓</td>
</tr>
<tr>
<td>H</td>
<td>↓</td>
</tr>
<tr>
<td>H</td>
<td>↓</td>
</tr>
<tr>
<td>H</td>
<td>↓</td>
</tr>
</tbody>
</table>

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.  
Q₀ = 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.

Figure 6.8 Pin diagram of JK Flip-flop(IC7473)

6.6 Postlab

1. Discuss the application of flip-flops in data storage.

2. Draw the logic diagram of Master Slave JK flip-flop.

3. A flip-flop is presently in the RESET state and must go to the SET state on the next clock pulse. What must J and K be?

4. Design a T flip-flop and D flip-flop using JK Flip-flop

Lab Report

Each individual will be required to submit a lab report. Use the format specified in the "Lab Report Requirements" document available on the class web page. Be sure to include the following items in your lab report:
- Lab cover sheet with staff verification for circuit diagram
- Answer the pre-lab questions
- Complete paper design for all three designs including K-maps and minimized equations and the truth table for each of the output signals.
- Answer the post-lab questions

**Grading**

<table>
<thead>
<tr>
<th></th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-lab Work</td>
<td>20</td>
</tr>
<tr>
<td>Lab Performance</td>
<td>30</td>
</tr>
<tr>
<td>Post-lab Work</td>
<td>20</td>
</tr>
<tr>
<td>Lab report</td>
<td>30</td>
</tr>
</tbody>
</table>

For the lab performance - at a minimum, demonstrate the operation of all the circuits to your staff incharge

The lab report will be graded as follows (for the 30 points):
- Logic Diagram along with IC no. and Pin no. for each logic gates 15 points
- K-maps and simplified expressions for all outputs and its truth table 15 points
Lab 7: Design of Synchronous Counters

7.1 Introduction

The purpose of this experiment is to introduce the design of Synchronous Counters. The student should also be able to design n-bit synchronous binary counter.

7.2 Hardware Requirement

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

7.3 Background

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.

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.

![Figure 7.1 Logic diagram of 3-bit Synchronous Counter](image)

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.
The count sequence for the 3-bit counter is shown on the right.

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.

7.4 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-6 counter.


7.5 Lab Procedure

7.5.1 Experiment #1 Design a Synchronous Counter for the given state diagram and implement it using JK Flip-flop for active high clock input signal.

- a) Draw the next state table and flip-flop transition table.
- b) Derive the logic expression for flip-flop inputs using K-Map
c) Implement the Counter using JK flip-flop and verify the counter table.
d) Draw its Timing Diagram

Figure 7.4 Logic Diagram

7.6 PostLab questions

1. Draw the state Diagram, state table and Timing Diagram of a 2-bit synchronous counter.

2. Design Mod 8 Up/Down synchronous Counter using JK Flip-flop

3. Design a 3-bit Up/Down Gray Code Counter using D Flip-flop

Lab Report

Each individual will be required to submit a lab report. Use the format specified in the "Lab Report Requirements" document available on the class web page. Be sure to include the following items in your lab report:

- Lab cover sheet with staff verification for circuit diagram
- Answer the pre-lab questions
- Complete paper design for all three designs including K-maps and minimized equations and the truth table for each of the output signals.
- Answer the post-lab questions

Grading

<table>
<thead>
<tr>
<th></th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-lab Work</td>
<td>20</td>
</tr>
<tr>
<td>Lab Performance</td>
<td>30</td>
</tr>
<tr>
<td>Post-lab Work</td>
<td>20</td>
</tr>
<tr>
<td>Lab report</td>
<td>30</td>
</tr>
</tbody>
</table>

For the lab performance - at a minimum, demonstrate the operation of all the circuits to your staff incharge

The lab report will be graded as follows (for the 30 points):

- Logic Diagram along with IC no. and Pin no. for each logic gates 15 points
- K-maps and simplified expressions for all outputs and its truth table 15 points
Lab 8: Design of Asynchronous Counters

8.1 Introduction

The purpose of this experiment is to introduce the design of Asynchronous Counters. The student should also be able to design n-bit asynchronous binary counter.

8.2 Hardware Requirement

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

8.3 Background

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.

2-Bit Ripple Counters

A two-bit asynchronous counter is shown on the left. 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. 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.

![2-Bit Asynchronous Counters diagram](image)

Figure 8.1 2-Bit Asynchronous Counters
Note that for simplicity, the transitions of Q0, Q1 and CLK in the timing diagram above are shown as simultaneous even though this is an asynchronous counter. Actually, there is some small delay between the CLK, Q0 and Q1 transitions.

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.

The 2-bit ripple counter circuit above has four different states, each one corresponding to a count value. Similarly, a counter with \( n \) flip-flops can have \( 2^n \) states. The number of states in a counter is known as its mod (modulo) number. Thus a 2-bit counter is a mod-4 counter.

A mod-\( n \) counter may also described as a divide-by-\( n \) counter. This is because the most significant flip-flop (the furthest flip-flop from the original clock pulse) produces one pulse for every \( n \) pulses at the clock input of the least significant flip-flop (the one triggers by the clock pulse). Thus, the above counter is an example of a divide-by-4 counter.

Two of the most common types of shift register counters are: the 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. The Ring and the Johnson counter must initially be forced into a valid state in the count sequence because they operate on a subset of the available number of states. Otherwise, the ideal sequence will not be followed.

### 8.4 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?

### 8.5 Lab Procedure

#### 8.5.1 Experiment#1 Design Divide-by-Ten Frequency Divider

a. Draw the State table and Logic Diagram.

b. Draw the timing diagram and verify its output.
8.5.2 **Experiment #2** Design a Sequential Circuit that is used to generate the output pattern 110. Connect the Preset input of D flip-flop to Logic’0’/Logic ‘1’ to produce the required pattern.

a. Draw the State table and Logic Diagram.

b. Draw the timing diagram and verify its output.

8.6 **Postlab questions**

a. Draw the logic diagram of Mod 12 Asynchronous Counter and its timing diagram.

b. Design a 4-bit frequency divider.

c. Design a 3-bit Johnson’s Counter. Set the Preset input of D flip-flop to 110.

d. Draw the state diagram for the given logic diagram

e. ![Logic Diagram](image)

Figure 8.3 Logic Diagram

**Lab Report**

Each individual will be required to submit a lab report. Use the format specified in the "Lab Report Requirements" document available on the class web page. Be sure to include the following items in your lab report:

- Lab cover sheet with staff verification for circuit diagram
- Answer the pre-lab questions
- Complete paper design for all three designs including K-maps and minimized equations and the truth table for each of the output signals.
- Answer the post-lab questions
Lab 9: Design of Shift Registers

9.1 Introduction

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.

9.2 Hardware Requirement

a. Equipments              -    Digital IC Trainer Kit
b. Discrete Components - IC7474 Dual JK Flip-flop

9.3 Background

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

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

9.3.1 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.
9.3.2 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.

9.4 Prelab questions

1. Develop the logic diagram for the shift register using JK flip-flop to replace the D flip flop?

2. How many flip-flop is needed to store the data 1001?

3. How many clock pulse are required to enter a byte of data serially into an 8-bit shift register?

4. What are the shift register counters?

5. Explain the function of SHIFT/LOAD input .

9.5 Lab Procedure

9.5.1 Experiment #1 Construct a 3-bit SISO shift register where the input is given to the MSB flip-flop(FF2) and the output is taken from LSB flip-flop(FF0).
### 9.5.2 Experiment #2
Construct a 3-bit PIPO shift register where the input given simultaneously to all the flip-flop and the output data appear on parallel outputs.

<table>
<thead>
<tr>
<th>CLK</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
<th>Q2</th>
<th>Q1</th>
<th>Q0</th>
<th>Data Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a. Draw the logic diagram using IC7474 and enter the data for the given function table.

b. Draw its Timing Diagram

### 9.6 PostLab questions

1. Draw the diagram of 4-Bit Bidirectional shift register.

2. What is PISO shift register and draw its logic diagram.

3. What is Universal Shift Register and what is its IC number.

4. Construct a 4-bit SISO and PIPO shift register. To register the input data 1101

### Lab Report

Each individual will be required to submit a lab report. Use the format specified in the "Lab Report Requirements" document available on the class webpage. Be sure to include the following items in your lab report:

- Lab cover sheet with staff verification for circuit diagram
- Answer the pre-lab questions
- Complete paper design for all three designs including K-maps and minimized equations and the truth table for each of the output signals.
- Answer the post-lab questions

**Grading**

<table>
<thead>
<tr>
<th>Component</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-lab Work</td>
<td>20</td>
</tr>
<tr>
<td>Lab Performance</td>
<td>30</td>
</tr>
<tr>
<td>Post-lab Work</td>
<td>20</td>
</tr>
<tr>
<td>Lab report</td>
<td>30</td>
</tr>
</tbody>
</table>

For the lab performance - at a minimum, demonstrate the operation of all the circuits to your staff incharge

The lab report will be graded as follows (for the 30 points):
- Logic Diagram along with IC no. and Pin no. for each logic gates 15 points
- K-maps and simplified expressions for all outputs and its truth table 15 points