



MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

**DEPARTMENT OF INSTRUMENTATION AND
CONTROL ENGINEERING**

**DIGIAL CIRCUITS LABORATORY
MANUAL – ICE 2161**

III SEMESTER B.Tech

Students Name:

Registration Number:

Signature:

Nov 2021



MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Department of Instrumentation & Control Engineering
Manipal Institute of Technology, Manipal
A Constituent unit of MAHE, Manipal

MANIPAL-576104

CERTIFICATE

This is to certify that the Laboratory Manual for the lab titled Digital Circuits Laboratory submitted by Mr/Ms _____ (Reg. No.: _____) of III Semester of Electronics and Instrumentation Engineering for the academic year 2021-2022 has been submitted as per laboratory course requirements, which has been evaluated and duly certified.

Place:

Date:

Lab In-Charge

Total number of Lab Sessions: 10**CONTENTS**

Sl. No.	Experiment	Page No.
01	Study of Basic Logic Gates and Implementation of Boolean Functions	1
02	Code Converters and Adder – Subtractor Circuits	11
03	Adders and Subtractor Circuits using IC's	12
04	Magnitude comparator and Parity checker / generator	14
05	Multiplexers and Demultiplexers	20
06	Decoders and Encoders	23
07	Flip-Flop Circuits	25
08	Counters	28
09	Shift Registers	33
10	Sequential circuits	35
Appendix A: Pinout Diagrams of various ICs		38

Evaluation plan

- Internal Assessment Marks : 60%
 - ✓ Continuous evaluation component (for each experiment):10 marks
 - ✓ The assessment will depend on punctuality, circuit implementation, maintaining the observation note and answering the questions in viva voce
 - ✓ Total marks of the 10 experiments reduced to marks out of 60
- End semester assessment of 2 hour duration: 40 %

Experiment 1

Study of Basic Logic Gates and Implementation of Boolean Functions

Aim:

- Familiarizing with IC trainer kit.
- Realizing basic logic functions using appropriate gates.
- Constructing simple combinational circuits.

Equipments and components required:

- 7408: Quad 2-input AND gates
- 7432: Quad 2-input OR gates
- 7404: Hex inverters
- 7400: Quad 2-input NAND gates
- 7402: Quad 2-input NOR gates
- 7486: Quad 2-input XOR gates
- IC trainer kit and patch chords.

Note: Refer data manuals/ Appendix for pin diagram and pin descriptions of the above IC's.

Theory:

The basic logic gates are the building blocks of more complex logic circuits. These logic gates perform the basic Boolean functions, such as AND, OR, NAND, NOR, Inversion, Exclusive-OR, Exclusive-NOR.

These basic logic gates are implemented as small-scale integrated circuits (SSICs) or as part of more complex medium scale (MSI) or very large-scale (VLSI) integrated circuits. Digital IC gates are classified not only by their logic operation, but also the specific logic-circuit family to which they belong. Each logic family has its own basic electronic circuit upon which more complex digital circuits and functions are developed. The following logic families are the most frequently used.

TTL - Transistor-transistor logic

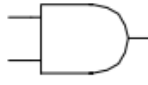

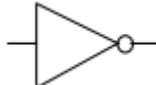

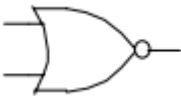
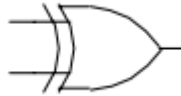
ECL - Emitter-coupled logic

MOS - Metal-oxide semiconductor

CMOS - Complementary metal-oxide semiconductor

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

CMOS logic consumes far less power than MOS logic. There are various commercial integrated circuit chips available. TTL ICs are usually distinguished by numerical designation as the 5400 and 7400 series.

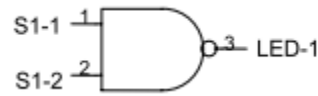
LOGIC	DESCRIPTION	SYMBOL
AND	A multi-input circuit in which the output is 1 only if all inputs are 1.	
OR	A multi-input circuit in which the output is 1 when any one input is 1.	
INVERT (NOT)	The output is 0 when the input is 1, and the output is 1 when the input is 0.	
NAND	AND followed by INVERT.	
NOR	OR followed by INVERT.	
EX-OR	The output is 0 when its two inputs are same and is 1 when its two inputs are different.	

Procedure:

1. Ensure that power supply to IC trainer kit is switched off.
2. Insert the required logic IC into the appropriate ZIF socket.
3. Using connecting wires / patch cords make connections to V_{cc} , GND, inputs and outputs.
4. Switch on the power to IC trainer kit.
5. Perform the experiments by applying proper inputs to logic gate and observe the outputs.
6. Once the experiment is completed, switch off the power to IC trainer kit and then remove the connecting wires and ICs.

Part 1: Realization of Basic Logic Functions:

1. Use one gate for each IC 7400 (NAND), 7402 (NOR), 7408 (AND), 7432 (OR), 7486 (XOR). Each has input pins, 1 and 2, and output pin 3.
2. Connect pin 1 to switch S1-1, pin 2 to switch S1-2, and pin 3 to LED-1 for every gate as shown in Fig. as an example for the NAND gate.



3. Using logic switches S1-1 and S1-2, apply the logic levels 0 and 1 to gate inputs (pin 1, pin 2), in the sequence shown in truth table. Record the output logic levels (LED-1) in table. Repeat the recordings for each gate.

Remember: LED ON = Logic 1 (High), LED OFF = Logic 0 (Low)

IC 7400 (NAND)		
Pin 1 (S1-1)	Pin 2 (S1-2)	Pin 3 (LED)
0	0	
0	1	
1	0	
1	1	

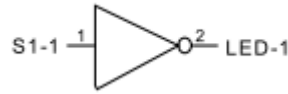
IC 7402 (NOR)		
Pin 1 (S1-1)	Pin 2 (S1-2)	Pin 3 (LED)
0	0	
0	1	
1	0	
1	1	

IC 7486 (XOR)		
Pin 1 (S1-1)	Pin 2 (S1-2)	Pin 3 (LED)
0	0	
0	1	
1	0	
1	1	

IC 7408 (AND)		
Pin 1 (S1-1)	Pin 2 (S1-2)	Pin 3 (LED)
0	0	
0	1	
1	0	
1	1	

IC 7432 (OR)		
Pin 1 (S1-1)	Pin 2 (S1-2)	Pin 3 (LED)
0	0	
0	1	
1	0	
1	1	

4. Use an inverter gate from IC 7404 whose input pin is pin 1 and whose output pin is pin 2. Using logic switches S1-1, apply the logic levels 0 and 1 in the sequence shown in table. Record the output logic levels in the given table.

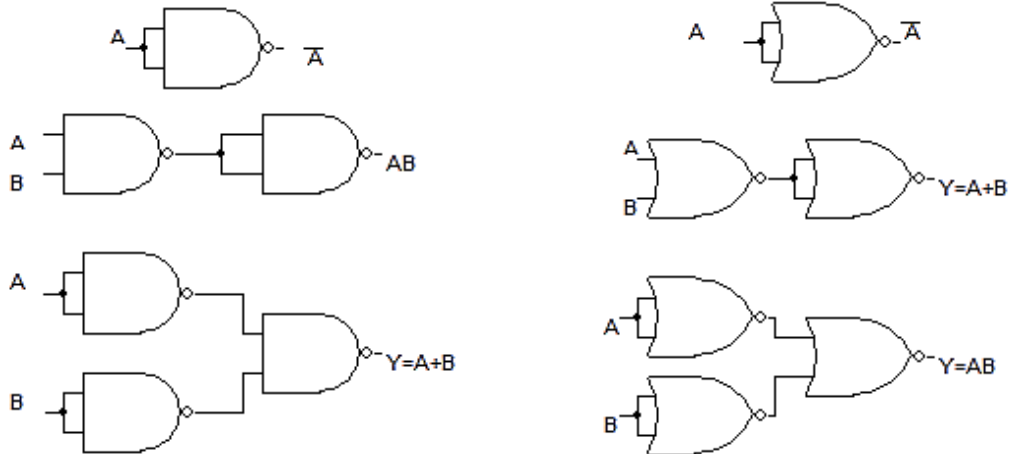


IC 7404 (INVERT)	
Pin 1 (S1-1)	Pin 2 (LED)
0	
1	

Part 2: Universality of NAND and NOR gates:

Using logic switches, apply the logic levels 0 and 1 to gate inputs for the circuit shown below and verify the outputs.

- Basic logic functions using NAND and NOR gates:



Part 3: Construction of Combinational Circuits:

- Design a logic circuit with 4 inputs A,B,C,D that will produce output '1' when two adjacent inputs are 1. Inputs A and D should be treated as adjacent. Implement it using universal logic.

Truth Table:

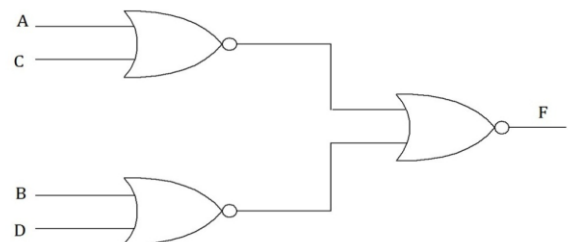
INPUT				OUTPUT
A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

The SOP logic expression can be written as shown below by adding the input variables for F=1:

$$F = \overline{A}\overline{B}CD + \overline{A}BC\overline{D} + \overline{A}BCD + A\overline{B}\overline{C}D + A\overline{B}CD + AB\overline{C}\overline{D} + AB\overline{C}D + ABC\overline{D} + ABCD$$

Solving using k-Map or Boolean theorems, yields: **F = (A+C) (B+D)**

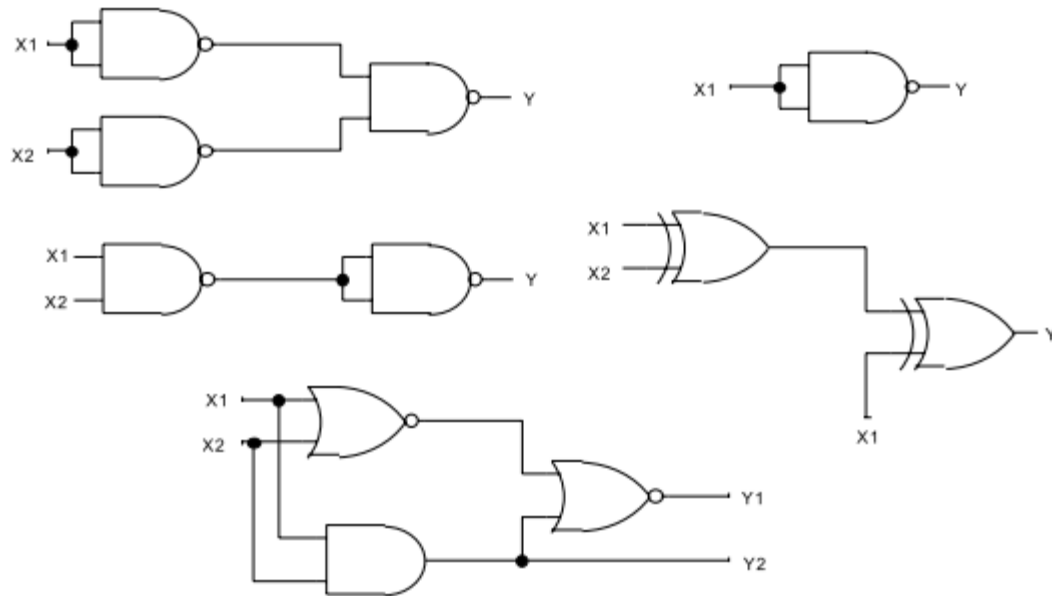
Logic diagram:



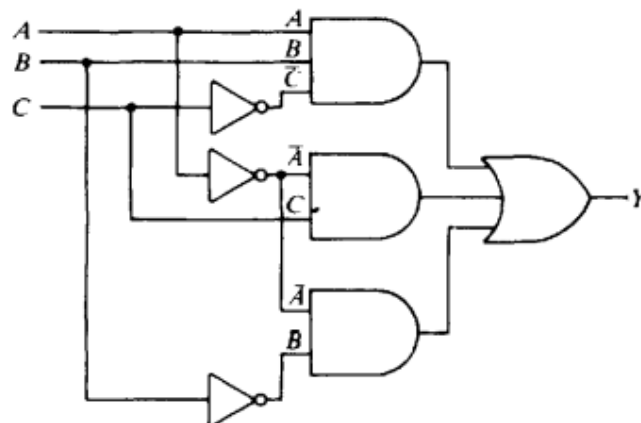
***Note: Follow the same procedure to develop the logics for all the exercise problems given below.**

Exercises:**PART A:**

- Write a truth table for each circuit. Derive Boolean expressions for all outputs. Also verify the results using lab experiments.



- A burglar alarm for a car has a normally low switch on each of four doors. If any door is opened the output of that switch goes HIGH. The alarm is set off with an active-LOW output signal. What type of gate will provide this logic? Support your answer with an explanation.
- Design a circuit that outputs 1 when the input is an odd BCD number.
- Write the Boolean equation for the following circuit and implement it using 2 input NOR gates.



- Design a combinational circuit to produce 2's complement of a 4-bit binary number.

Experiment 2

Code Converters and Adder – Subtractor Circuits

a) Code Converter Circuits

Aim:

- To design and implement the following converters:
 - (a) 4-bit Binary to Gray code
 - (b) BCD to Excess-3 code
 - (c) BCD to 7-segment code

Equipments & Components Required:

- ICs 7408, 7411, 7432, 7404, 7486, 7447, 7730
- IC trainer kit, Connecting wires

Theory:

Code converter is a combinational circuit that translates the input code word into a new corresponding word.

1. Binary to Gray Code Conversion:

Gray code is one of the codes used in digital systems. It has the advantage over binary numbers that only one bit in the code word changes when going from one number to the next. Consider converting a binary number to its Gray code equivalent. Figure 2.1 shows the binary number 0010 being translated into its Gray code equivalent. Start at the MSB of the binary number. Transfer this to the left position in the Gray code as shown by the downward arrow. Now add the 8s bit to the next bit over (4s bit). The sum is 0 ($0 + 0 = 0$), which is transferred down and written as the second bit from the left in the Gray code. The 4s bit is now added to the 2s bit of the binary number. The sum is 1 ($0 + 1 = 1$) and is transferred down and written as in the third bit from the left in the Gray code. The 2s bit is now added to the 1s bit of the binary number. The sum is 1 ($1 + 0 = 1$) and is transferred and written as the right bit in the Gray code. The binary 0010 is then equal to the Gray code number 0011.

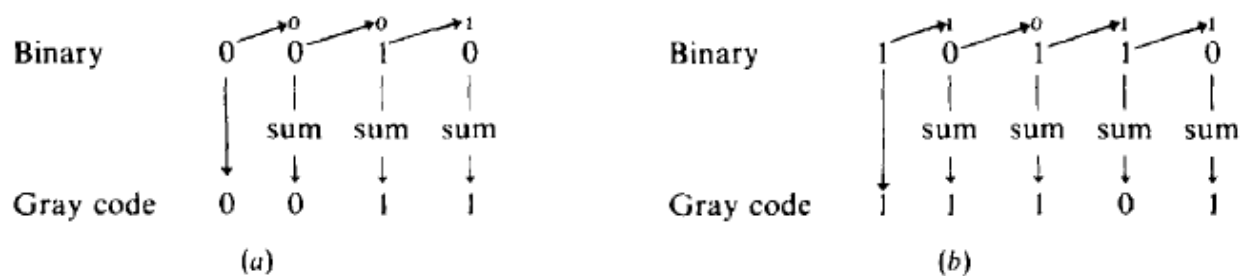


Figure 2.1. Example for Binary to Gray Code Conversion

2. Excess-3 Code:

The excess-3 code digit is obtained by adding 'three' to the corresponding BCD digit. Consider changing the decimal number 62 to an equivalent XS3 number. Step 1 in Figure 2.2(a) shows 3 being added to each decimal digit. Step 2 shows 9 and 5 being converted to their 8421 BCD equivalents. The decimal number 62 then equals the BCD XS3 number 1001 0101. Convert the 8421 BCD number 0100 0000 to its XS3 equivalent. Figure 2.2(b) shows the procedure. The BCD number is divided into 4-bit groups starting at the binary point. Step 1 shows 3 (binary 0011) being added to each 4-bit group. The sum is the resulting XS3 number. Figure 2.2(c) shows the 8421 BCD number 01000000 being converted to its equivalent BCD XS3 number, which is 0111 0011.

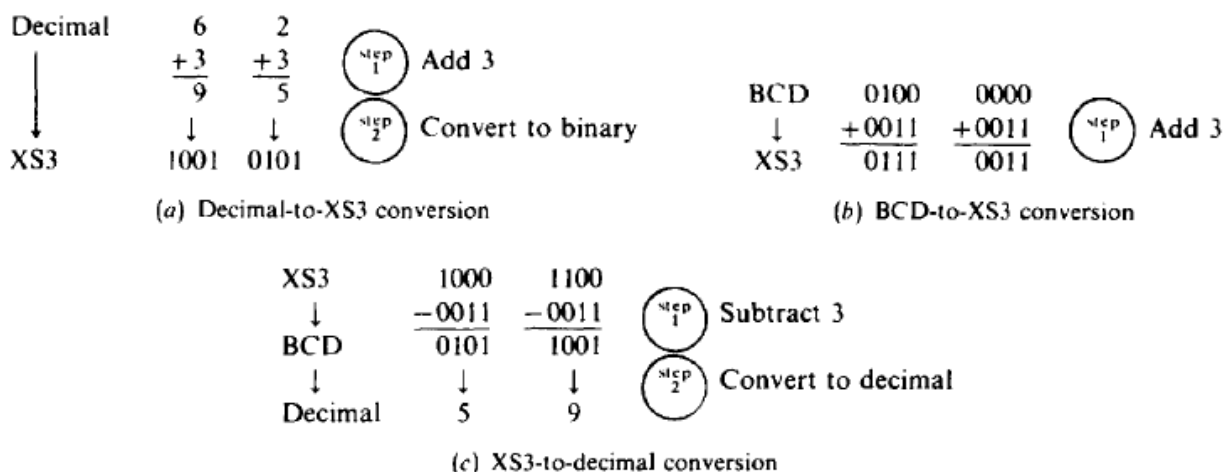


Figure 2.2. Example for Binary to XS3 Code Conversion and vice versa

3. BCD to Seven Segment Decoder:

A common task for a digital circuit is to decode from machine language to decimal numbers. A common output device used to display decimal numbers is the seven segment display shown in Fig. 2.3(a). The seven segments are labeled with standard letters from a through g. The first 10 displays, representing decimal digits 0 through 9, are shown in Fig. 2.3(b). For instance, if segments b and c of the seven-segment display light, a decimal 1 appears. If segments a, b and c light, a decimal 7 appears, and so forth.

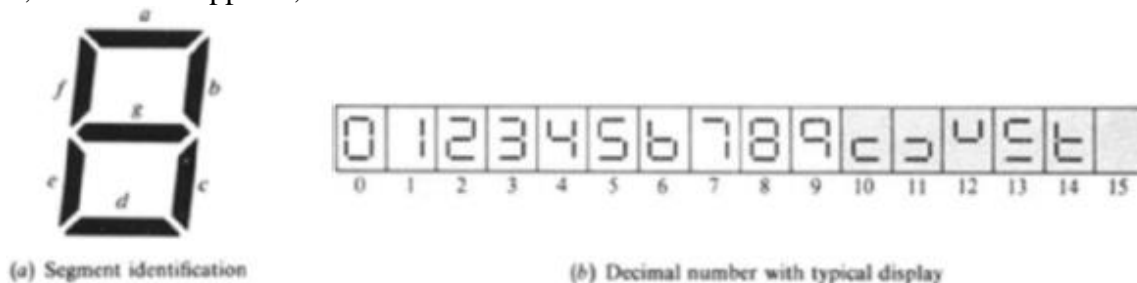


Figure 2.3 Seven Segment Display

Experiment:**Part 1: Implement a 4-bit Binary to Gray code converter and verify the truth table.**

Binary inputs	Gray outputs
$B_3 B_2 B_1 B_0$	$G_3 G_2 G_1 G_0$
0 0 0 0	0 0 0 0
0 0 0 1	0 0 0 1
0 0 1 0	0 0 1 1
0 0 1 1	0 0 1 0
0 1 0 0	0 1 1 0
0 1 0 1	0 1 1 1
0 1 1 0	0 1 0 1
0 1 1 1	0 1 0 0
1 0 0 0	1 1 0 0
1 0 0 1	1 1 0 1
1 0 1 0	1 1 1 1
1 0 1 1	1 1 1 0
1 1 0 0	1 0 1 0
1 1 0 1	1 0 1 1
1 1 1 0	1 0 0 1
1 1 1 1	1 0 0 0

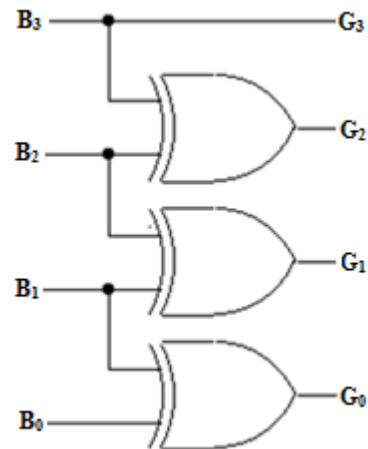
Logic Expression:

$$G_3 = B_3$$

$$G_2 = B_3 \oplus B_2$$

$$G_1 = B_2 \oplus B_1$$

$$G_0 = B_1 \oplus B_0$$

**Part 2: Implement a BCD to Excess-3 converter:**

BCD	Excess-3
A B C D	W X Y Z
0 0 0 0	
0 0 0 1	
0 0 1 0	
0 0 1 1	
0 1 0 0	
0 1 0 1	
0 1 1 0	
0 1 1 1	
1 0 0 0	
1 0 0 1	

Logic Expression:

$$W = A + BC + BD$$

$$X = \bar{B}D + \bar{B}C + B\bar{C}\bar{D}$$

$$Y = \bar{C}\bar{D} + CD$$

$$Z = \bar{D}$$

Implement the above expressions and verify the truth table.

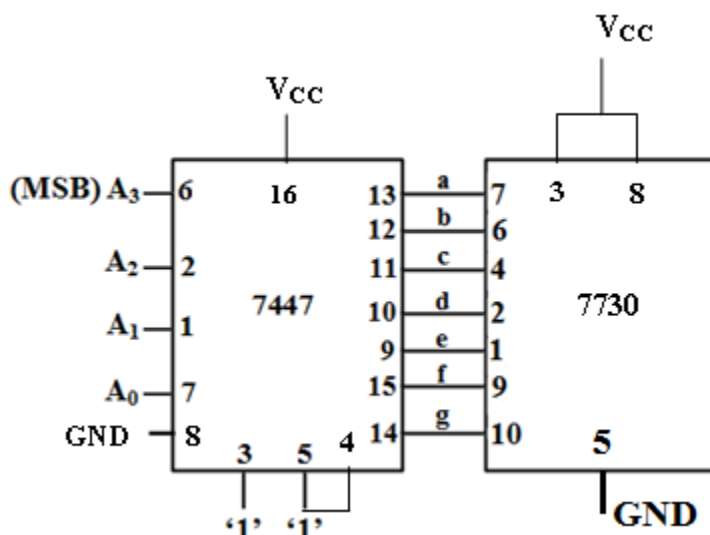
Part 3: Implement a BCD to 7-segment converter:

A seven segment indicator is used for displaying decimal digits 0 through 9.

IC 7447 is a BCD to seven segment decoder/ driver.

IC 7730 is a seven segment display (common anode).

Develop the logic and verify the truth table.



b) Adders and Subtractors

Aim:

- To design and implement full adder and full subtractor circuits.

Equipments & Components Required:

- ICs 7408, 7432, 7404, 7486
- IC trainer kit, Connecting wires

Experiment:

Part 1: Full adder: Full adder adds two bits and also another bit called carry-in that is propagated from the previous stage. In effect, a full adder adds three bits to give out two outputs – sum and carry out. A, B and C_{in} are the three inputs, out of which C_{in} is to be considered as carry propagated from the previous stage. S and C_{out} are the outputs.

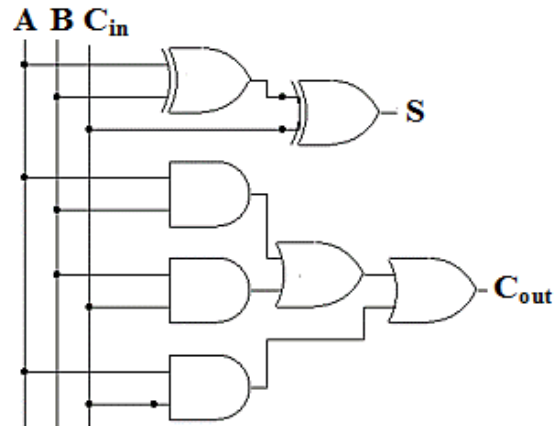
Truth Table

Input			Output	
A	B	C _{in}	C _{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Reduced Boolean Expressions

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

$$C_{out} = AB + BC_{in} + C_{in}A$$



Part 2: Full Subtractor: The full subtractor is a combinational circuit which is used to perform subtraction of three bits. It has three inputs, X (minuend) and Y (subtrahend) and Z (subtrahend) and two outputs D (difference) and B (borrow).

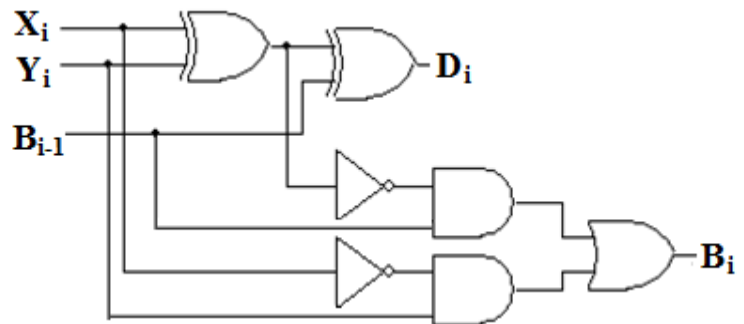
Truth Table

Input			Output	
X _i	Y _i	Z(B _{i-1})	D _i	B _i
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

Reduced Boolean Expressions

$$D_i = X_i \oplus Y_i \oplus B_{i-1}$$

$$B_i = \overline{X_i} \oplus \overline{Y_i} B_{i-1} + \overline{X_i} Y_i$$

**Exercises:**

Design the following converters using discrete gates:

1. 4-bit Gray code to binary
2. 2 4 2 1 BCD to 8 4 2 1 BCD
3. 2 4 2 1 BCD to EXCESS-3 BCD
4. 8 4 2 1 BCD to 8 4 -2 -1 BCD
5. BCD to seven segment code

Experiment 3

Adder – Subtractor Circuits using IC's

Aim:

- Implement a 4-bit parallel adder - IC 7483

Equipments & Components Required:

- ICs 7400, 7486, 7410, 7483
- IC trainer kit, Connecting wires

Experiment:

Part 1: 4 –bit Binary full adder / 2's complement subtractor: The IC 7483 adds two 4-bit binary words plus the incoming carry. This contains 4 inter connected full adder and a look ahead carry adder circuit. Sum outputs are S_3 to S_0 , the out-going carry is C_{out} .

Boolean Expression for Sum (S)

$$S = \overline{A} \overline{B} C_{in} + \overline{A} B \overline{C_{in}} + A \overline{B} \overline{C_{in}} + A B C_{in}$$

$$S = \overline{C_{in}}(\overline{A} B + A \overline{B}) + C_{in}(A B + \overline{A} \overline{B})$$

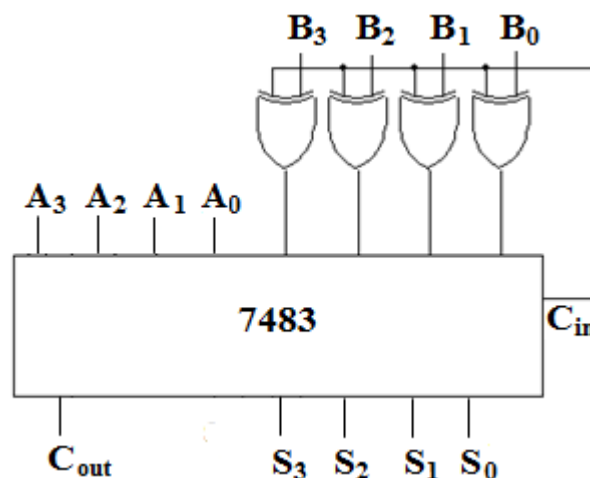
$$S = \overline{C_{in}}(A \oplus B) + C_{in}(\overline{A \oplus B})$$

$$S = \overline{C_{in}}(A \oplus B) + C_{in}(\overline{A \oplus B})$$

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

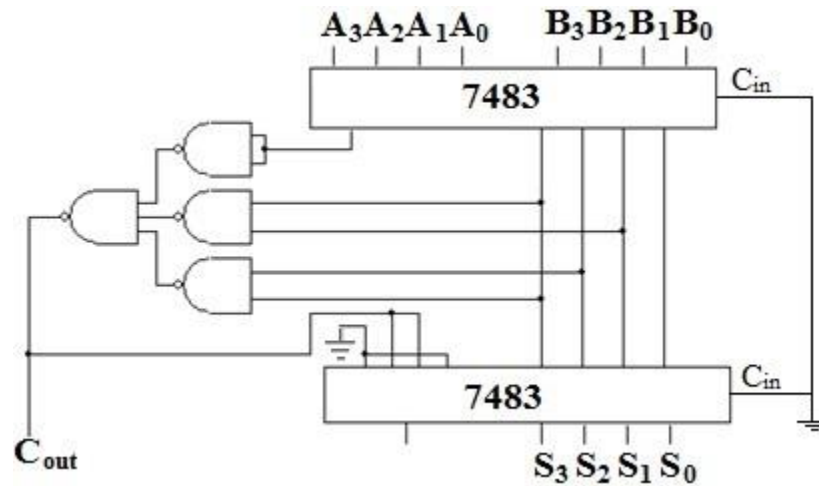
Boolean Expression for Carry (C_{out})

$$C_{out} = AB \oplus BC_{in} \oplus AC_{in}$$



***Note:** If control (C_{in}) input is 0, input $B = B_3 B_2 B_1 B_0$ is added to input $A = A_3 A_2 A_1 A_0$. If control (C_{in}) input is 1, 2's complement of B is added to input A. (i.e., circuit computes $A - B$) Difference is available at $S_3 S_2 S_1 S_0$, borrow output at C_{out} .

Part 2: BCD adder: When two 4 bit BCD numbers are added, if the sum exceeds 9 or if there is a carry, then 6 is added to the sum and a carry is generated to the next decimal digit. Carry from the addition of 6 (if any) is neglected.

**Exercises:**

1. Implement full adder using two half adders and one OR gate.
2. Implement full subtractor using two half subtractors and any additional gate.
3. Implement a circuit which can do the following $3_{(10)} - 8_{(10)}$ using adder ICs
4. Implement a circuit which can do the following $6_{(8)} - 3_{(8)}$.
5. Implement a single digit Excess-3 adder.

Experiment 4

Magnitude Comparators and Parity Generator / Checker

Aim:

- To design, implement and study 1-bit and 2-bit magnitude comparators and 3-bit odd parity generator /checker.
- To study the working of 4-bit magnitude comparator IC 7485 and parity generator / checker IC 74180.

Equipments & Components Required:

- IC 7404, IC 7408, IC 7486, IC 7485, IC 74180, IC 7411
- IC trainer kit & Connecting wires

Experiment:

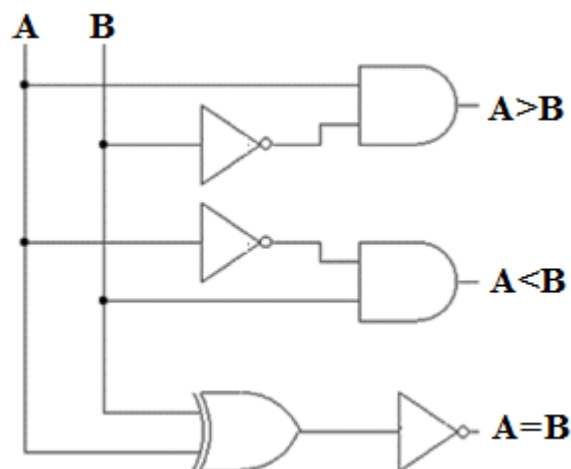
Part 1: One bit magnitude comparator: A and B are two single bit inputs for the comparator and has three output lines, i.e. ($A > B$), ($A < B$), and ($A = B$).

- When input A is greater than input B ($A > B$), output will be high
- When input A is less than input B ($A < B$), output will be high
- When input A is equal to input B ($A = B$), output will be high

***Note:** Only one out of three outputs will be high at any time.

Truth Table:

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



Part 2: Two bit magnitude comparator: Compares two 2-bit inputs, where A_1 and A_0 are the two bits of the first number. B_1 and B_0 are the two bits of the second number.

Truth Table:

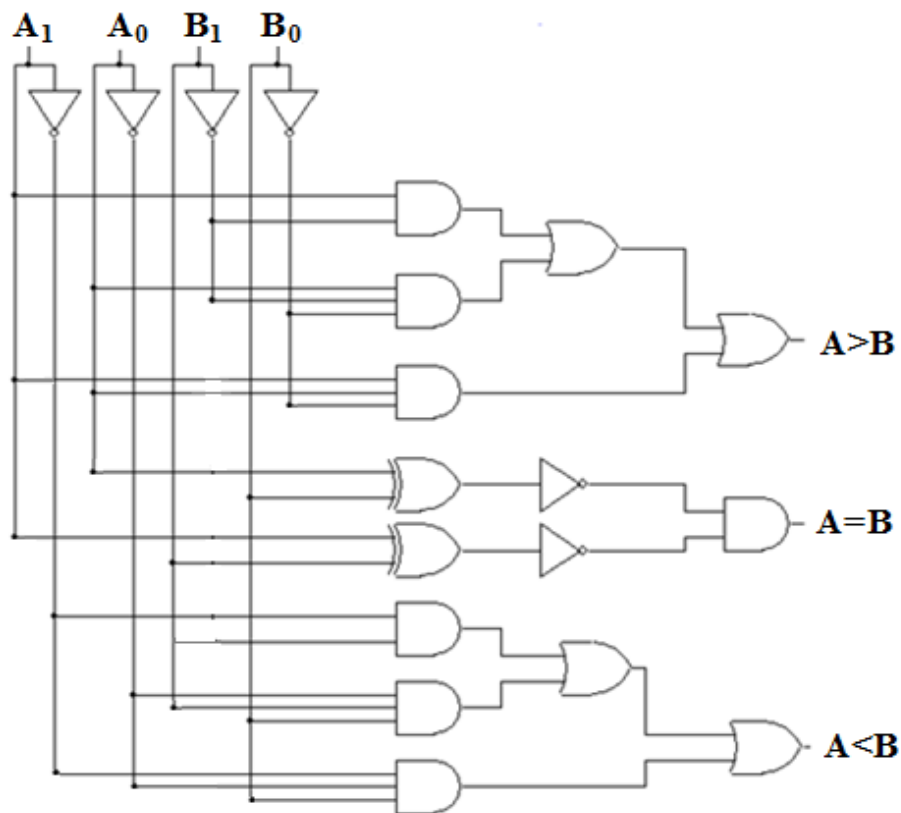
INPUTS				OUTPUTS		
A ₁	A ₀	B ₁	B ₀	A > B	A < B	A = B
0	0	0	0	0	0	1
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	0	0
0	1	0	1	0	0	1
0	1	1	0	0	1	0
0	1	1	1	0	1	0
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	0	1
1	0	1	1	0	1	0
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	0	1

Logic Expression:

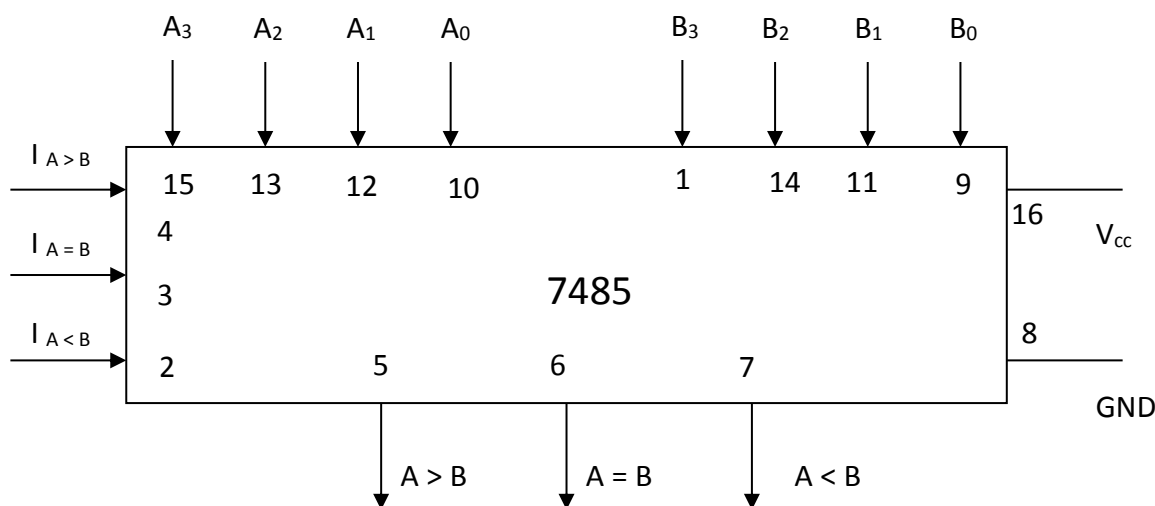
$$(A > B) = A_1 \bar{B}_1 + A_0 \bar{B}_1 \bar{B}_0 + A_1 A_0 \bar{B}_0$$

$$(A = B) = (\bar{A}_0 \oplus \bar{B}_0) \cdot (\bar{A}_1 \oplus \bar{B}_1)$$

$$(A < B) = \bar{A}_1 B_1 + \bar{A}_0 B_1 B_0 + \bar{A}_1 \bar{A}_0 B_0$$

Logic Circuit:

Part 3: 4 – bit magnitude comparator using IC 7485: IC 7485 is a 4-bit magnitude comparator, which can compare two 4–bit binary or BCD codes. These IC's can be cascaded to compare words with more than 4 bits.



Truth Table:

COMPARING INPUTS				CASCADE INPUTS			OUTPUTS		
A ₃ ,B ₃	A ₂ ,B ₂	A ₁ ,B ₁	A ₀ ,B ₀	I _{A>B}	I _{A<B}	I _{A=B}	A>B	A<B	A=B
A ₃ >B ₃	X	X	X	X	X	X	1	0	0
A ₃ <B ₃	X	X	X	X	X	X	0	1	0
A ₃ =B ₃	A ₂ >B ₂	X	X	X	X	X	1	0	0
A ₃ =B ₃	A ₂ <B ₂	X	X	X	X	X	0	1	0
A ₃ =B ₃	A ₂ =B ₂	A ₁ >B ₁	X	X	X	X	1	0	0
A ₃ =B ₃	A ₂ =B ₂	A ₁ <B ₁	X	X	X	X	0	1	0
A ₃ =B ₃	A ₂ =B ₂	A ₁ =B ₁	A ₀ >B ₀	X	X	X	1	0	0
A ₃ =B ₃	A ₂ =B ₂	A ₁ =B ₁	A ₀ <B ₀	X	X	X	0	1	0
A ₃ =B ₃	A ₂ =B ₂	A ₁ =B ₁	A ₀ =B ₀	1	0	0	1	0	0
A ₃ =B ₃	A ₂ =B ₂	A ₁ =B ₁	A ₀ =B ₀	0	1	0	0	1	0
A ₃ =B ₃	A ₂ =B ₂	A ₁ =B ₁	A ₀ =B ₀	0	0	1	0	0	1
A ₃ =B ₃	A ₂ =B ₂	A ₁ =B ₁	A ₀ =B ₀	X	X	1	0	0	1

***Note:**

- The upper part of the table describes normal condition operation that will occur in single device or in series expansion scheme.
- The lower part of the table describes feed forward conditions that exist in parallel expansion scheme.

Part 4: 3-bit odd parity generator/checker: An odd parity generator generates a '1' for an even number of 1's in the message, else 0.

Step 1: Consider a 3-bit message XYZ to be transmitted with an odd parity bit.

Step 2: Obtain an expression for P in terms of the message bits and realize using gates.

X	Y	Z	Parity (P)
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

$$P = \overline{X}\overline{Y}\overline{Z} + \overline{X}YZ + X\overline{Y}\overline{Z} + XY\overline{Z}$$

$$P = \overline{X}(Y \oplus Z) + X(Y \oplus Z)$$

$$P = X \oplus (Y \oplus Z)$$

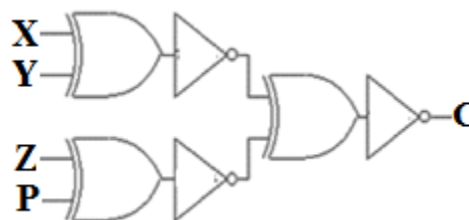


Step3: The odd parity checker consists of a 4-bit data consisting of X, Y, Z, P

X	Y	Z	P	C (Output)
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

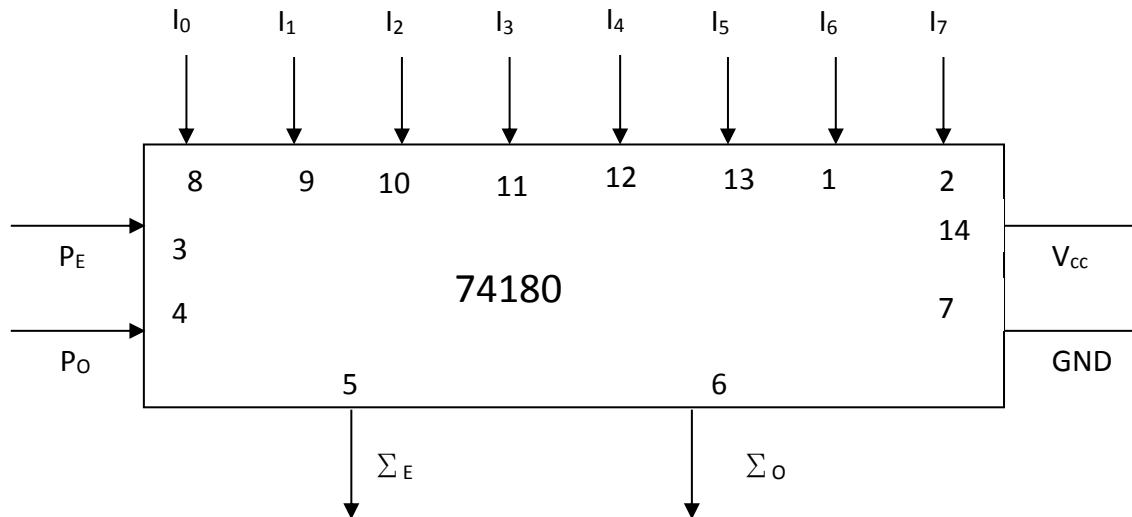
***Note:** Even number of 1's received indicates error which is detected as 1 at the output of the checker.

$$C = X \oplus Y \oplus Z \oplus P$$



Part 5: Study of IC 74180 (parity generator/checker):

- The IC 74180 is a 9 bit (including the parity bit) parity generator or checker, commonly used to detect errors in high speed data transmission or data retrieving systems.
- Both even and odd parity enable inputs and parity outputs are available, for generating or checking parity, on 8 bit of data.

**Truth Table:**

INPUTS			OUTPUTS	
Number of High data Inputs ($I_0 - I_7$)	P_E	P_O	Σ_E	Σ_O
EVEN	H	L	H	L
ODD	H	L	L	H
EVEN	L	H	L	H
ODD	L	H	H	L

***Note:**

- The even parity output (Σ_E) is high when an even number of data inputs ($I_0 - I_7$) are high, P_E is high and P_O is low.
- The odd parity output (Σ_O) is high when an odd number of data inputs are high, P_O is high and P_E is low.

Observe and note down the outputs for the following inputs.

I_7 I_6 I_5 I_4 I_3 I_2 I_1 I_0	P_E P_O	Σ_E Σ_O
1 1 1 1 0 0 0 0	1 0	
1 1 1 1 0 0 0 1	1 0	
1 0 1 0 1 1 0 0	0 1	
1 0 1 0 1 1 0 1	0 1	
0 0 0 0 0 0 0 0	1 0	
1 1 1 1 1 1 1 1	0 1	

EXERCISES:

1. Design and implement 8-bit comparator using 7485 ICs.
2. Design a 5-bit comparator using single IC 7485 and one gate.
3. Design and implement a 4-bit even parity generator/checker.
4. Design a 9-bit odd parity checker using a 74180 and an inverter.
5. Design a 5-bit comparator using discrete gates without using K-Map.

Experiment 5

Multiplexers and De-multiplexers

Aim:

- To design and implement various multiplexer circuits and to generate logic functions using multiplexers.
- To design and implement De-multiplexers.

Components and Equipments Required:

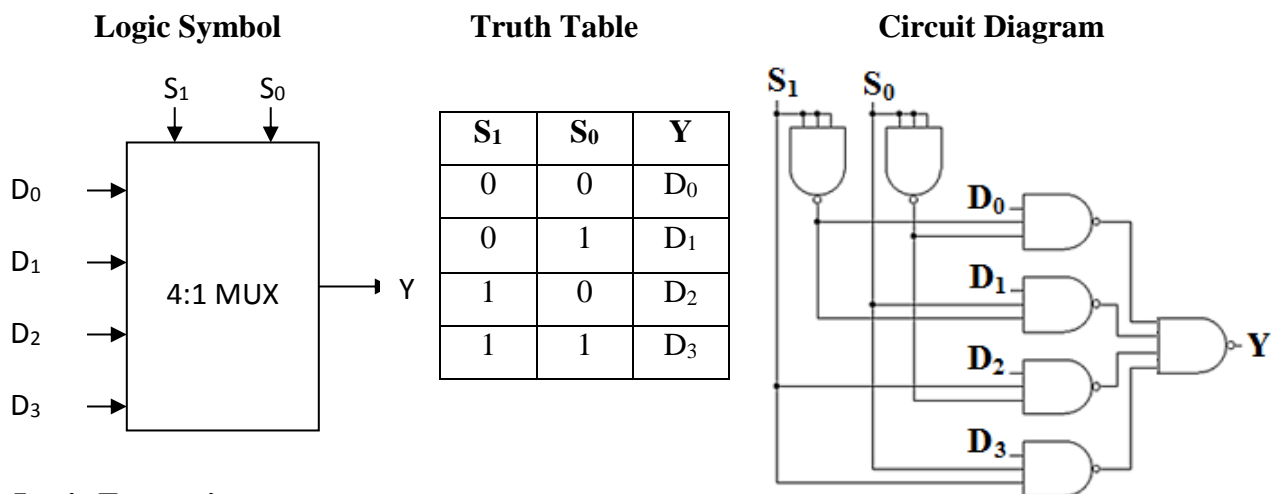
- ICs 74151, 74153
- Basic gates, universal gates
- IC trainer kit, connecting wires

Theory:

Multiplexers – A multiplexer or data selector is a logic circuit that accepts several data inputs and allows only one of them at a time to get through the output. The routing of the desired data input to the output is controlled by SELECT inputs. The multiplexer acts like a digitally controlled multi position switch. The digital code applied to the SELECT inputs determines which data input to be switched to the output.

De-multiplexers – A demultiplexer or data distributor is a circuit that receives information on a single line and transmits this information on one of the 2^n possible output lines. The selection of a specific output line is controlled by n selection lines. A decoder can function as demultiplexer if enable line is taken as data input.

Part 1: 4:1 Multiplexer using only NAND gates:



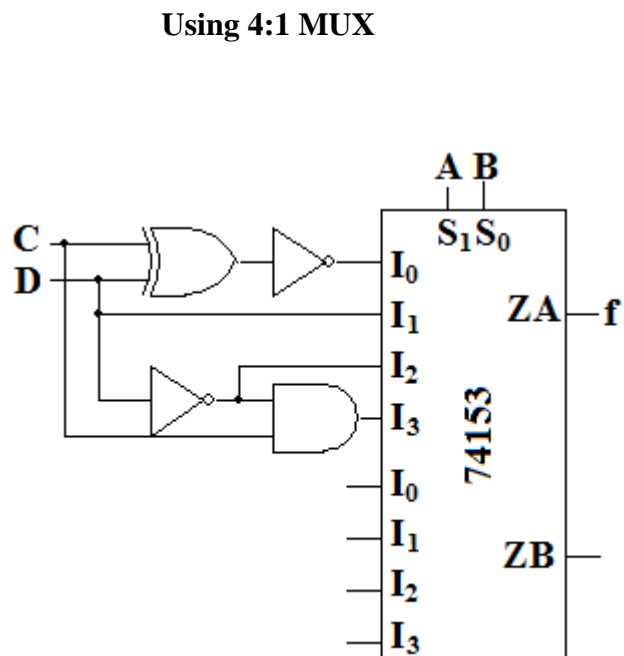
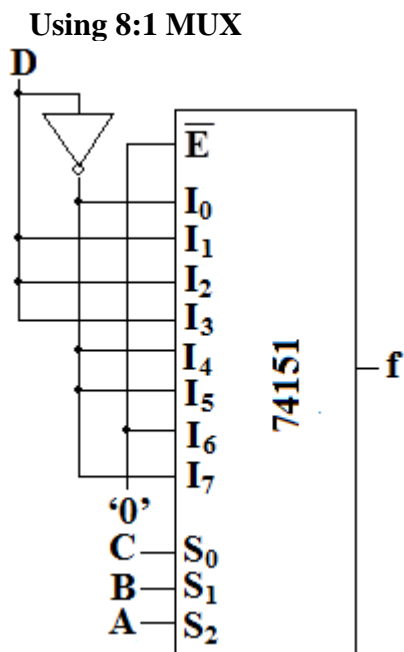
Logic Expression:

$$Y = \bar{S}_1 \bar{S}_0 D_0 + \bar{S}_1 S_0 D_1 + S_1 \bar{S}_0 D_2 + S_1 S_0 D_3$$

Part 2: Test the multiplexer ICs 74153, 74151, 74157 for their functionalities.

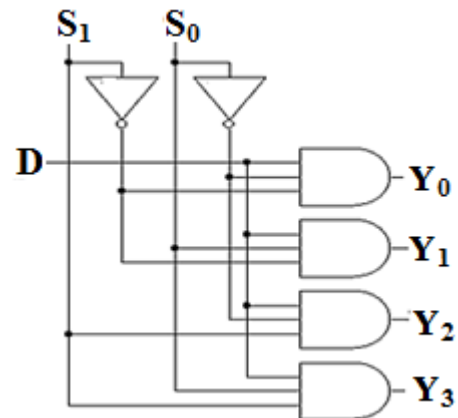
Part 3: Implement $f = \sum m(0, 3, 5, 7, 8, 10, 14)$ using i) 8:1 MUX ii) 4:1 MUX and additional gates:

A	B	C	D	F	Using 8:1 MUX	Using 4:1 MUX
0	0	0	0	1	$I_0 = \overline{D}$	$I_0 = \overline{C \oplus D}$
0	0	0	1	0		
0	0	1	0	0	$I_1 = D$	
0	0	1	1	1		
0	1	0	0	0	$I_2 = D$	$I_1 = D$
0	1	0	1	1		
0	1	1	0	0	$I_3 = D$	
0	1	1	1	1		
1	0	0	0	1	$I_4 = \overline{D}$	$I_2 = \overline{D}$
1	0	0	1	0		
1	0	1	0	1	$I_5 = \overline{D}$	
1	0	1	1	0		
1	1	0	0	0	$I_6 = 0$	$I_3 = C\overline{D}$
1	1	0	1	0		
1	1	1	0	1	$I_7 = \overline{D}$	
1	1	1	1	0		



Part 4: Design a 1:4 De-multiplexer using basic gates:

Select lines		Data	Outputs			
S ₁	S ₀	D	Y ₀	Y ₁	Y ₂	Y ₃
0	0	0 / 1	D	0	0	0
0	1	0 / 1	0	D	0	0
1	0	0 / 1	0	0	D	0
1	1	0 / 1	0	0	0	D

**Part 5: Verify 1:16 DEMUX truth table using 74154 IC.****Exercises:**

1. Implement $f(a, b, c) = \bar{a}\bar{b} + \bar{b}\bar{c} + abc$ using (i) 8:1 MUX (ii) 4:1 MUX
2. Implement full adder using 4:1 MUX and 8:1 MUX
3. Implement 8:1 MUX using 2:1 multiplexers
4. Design 3-bit binary to gray code converter using 4:1 MUX
5. Implement $F = \sum m(0, 5, 7, 11, 15, 16, 18, 25, 29)$ using two 8:1 and one 2:1 MUX

Experiment 6

Decoders and Encoders

Aim:

- To design and implement Decoders and Encoders.

Components and Equipments Required:

- ICs 74139, 74138, 74154, 74148, Basic gates, universal gates
- IC trainer kit, connecting wires

Theory:

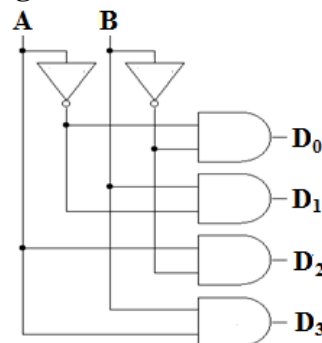
Decoder – It is a logic circuit that converts an N-bit binary input code into M output lines such that only one line is activated for each one of those possible combinations of inputs.

Encoder – An encoder has 2^n (or less) input lines and n output lines. The output lines generate the binary code for 2^n inputs.

Priority Encoder – The priority encoders output corresponds to the currently active input which has the *highest priority*. So when an input with a higher priority is present, all other inputs with a lower priority will be ignored.

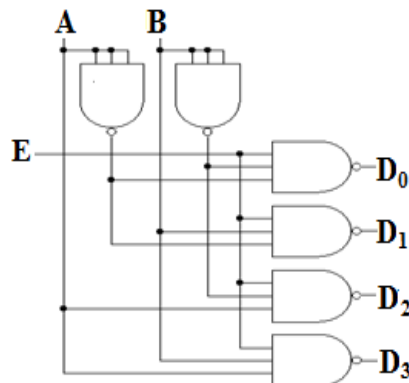
Part 1: Design and implement 2:4 decoder using basic gates:

INPUTS		OUTPUTS			
A	B	D ₀	D ₁	D ₂	D ₃
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



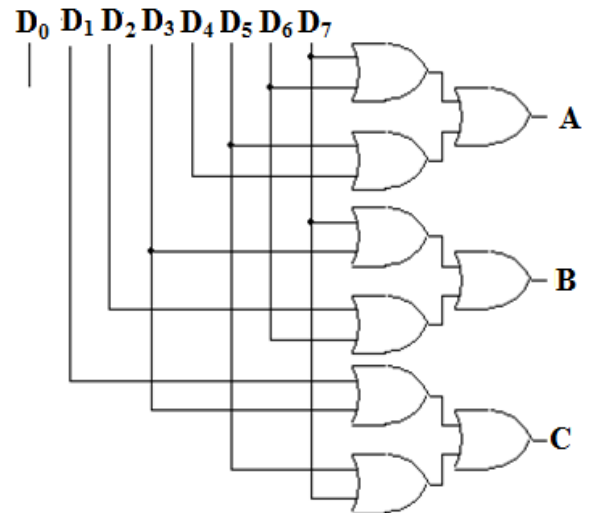
Part 2: Design and implement 2:4 decoder with enable input using only NAND gates

Inputs			Outputs			
E	A	B	D ₀	D ₁	D ₂	D ₃
0	X	X	1	1	1	1
1	0	0	0	1	1	1
1	0	1	1	0	1	1
1	1	0	1	1	0	1
1	1	1	1	1	1	0



Part 3: Design and implement Octal to Binary encoder using basic gates:

Inputs								Outputs		
D7	D6	D5	D4	D3	D2	D1	D0	A	B	C
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1



Part 4: Implement $f = AB + \overline{A}\overline{B}\overline{C}$ using 74138 and additional gate.

Part 5: Design and implement 4 to 16 decoder using two 74138 decoders.

Exercises:

1. Implement a decimal to BCD encoder.
2. Design and implement 4 to 16 decoder using two 74138 decoders.
3. Implement $f(A, B, C, D) = \sum (0, 3, 5, 9)$ using two 74138 ICs and additional gate
4. Implement the following multi output Combinational circuit using a 4 to 16 decoder:
 - i. $F_1(A, B, C, D) = \sum (2, 4, 8, 13)$
 - ii. $F_2(A, B, C, D) = \sum (5, 9, 11, 14)$

Experiment 7

Flip-Flop Circuits

Aim:

- To realize SR Flip-flop, D Flip-flop, JK Flip-flop, T Flip-flop.
- To study the IC –7474 and IC – 7473.

Components and Equipments Required:

- ICs 7400, 7473, 7474, 7404
- IC trainer kit, connecting wires

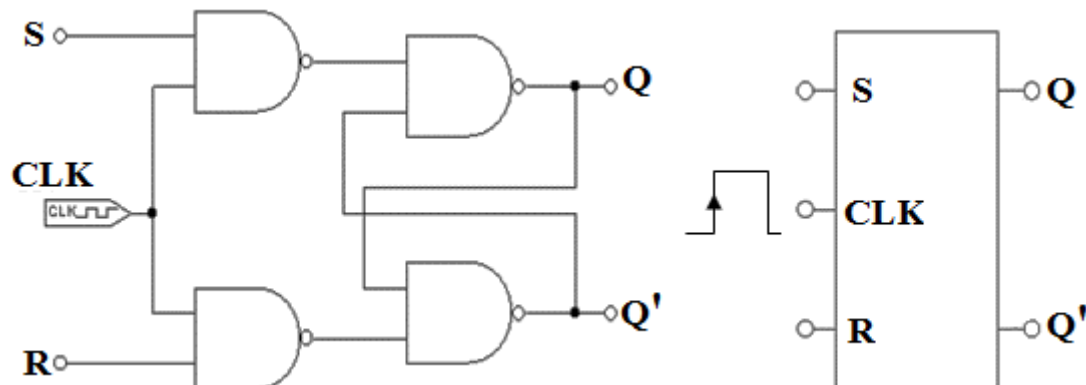
***Note:** Refer the IC data manual, draw the block & pin diagram of ICs 7473 & 7474.

Theory:





Flip-flops, also called bistable gates, are digital logic circuits that can be in one of two states. Flip-flops maintain their state indefinitely until an input pulse called a *trigger* is received. When a trigger is received, the flip-flop outputs change state according to defined rules and remain in those states until another trigger is received. Flip-flop circuits are interconnected to form the logic gates for the digital integrated circuits (IC s) used in memory chips and microprocessors. Flip-flops can be used to store one bit, or binary digit, of data. The data may represent the state of a sequencer, the value of a counter, an ASCII character in a computer's memory or any other piece of information. There are several different kinds of flip-flop circuits, with designators such as *T* (toggle), *S-R* (set/reset) *J-K* (possibly named for Jack Kilby) and *D* (Data or Delay). A flip-flop typically includes zero, one, or two input signals as well as a clock signal and an output signal. Some flip-flops also include a clear input signal to reset the current output.

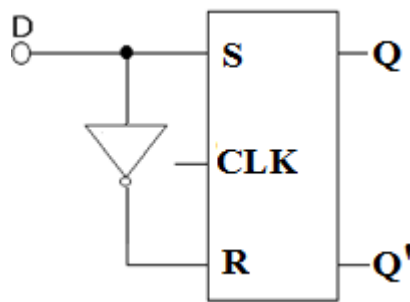
Experiment:



Part 1: SR Flip-flop using NAND gates:



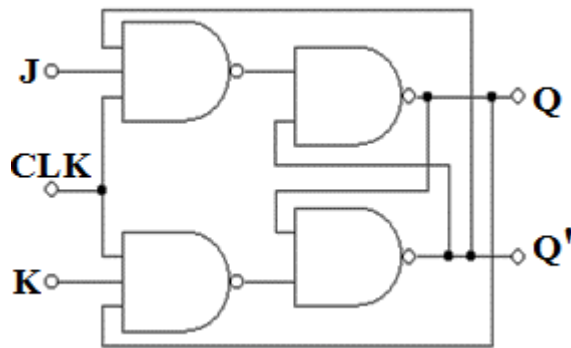
***Note:** Apply high-going mono pulse to CLK





INPUTS			OUTPUTS		OBSERVATIONS	
CLK	S	R	Q_{n+1}	$\overline{Q_{n+1}}$	Q_{n+1}	$\overline{Q_{n+1}}$
	0	0	Q_n	$\overline{Q_n}$		
	0	1	0	1		
	1	0	1	0		
	1	1	Not Allowed			

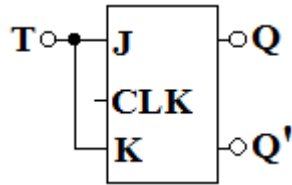
Part 2: D Flip-flop using NAND gates:

INPUTS		OUTPUTS	
CLK	D	Q_{n+1}	$\overline{Q_{n+1}}$
LOW	X		
	0		
	1		

***Note:** Use NAND gates to implement SR Flip-flop

Part 3: JK Flip-Flop using NAND gates:

INPUTS			OUTPUTS	
CLK	J	K	Q_{n+1}	$\overline{Q_{n+1}}$
	0	0		
	0	1		
	1	0		
	1	1		

Part 4: T-Flip Flop using NAND gates:

INPUTS		OUTPUTS	
CLK	T	Q_{n+1}	$\overline{Q_{n+1}}$
LOW	X		
	0		
	1		

Part 5: IC 7474: Positive edge triggered dual D flip-flops with preset and clear:

Inputs				Outputs	
Preset	Clear	Clock	Data	Q_{n+1}	$\overline{Q_{n+1}}$
PR	CLR	CLK	D		
0	1	X	X		
1	0	X	X		
1	1	0	X		
1	1		1		
1	1		0		

Part 6: 74LS73A: Negative edge triggered dual JK Flip-flops with clear:

Inputs				Outputs	
Clear	Clock	J	K	Q_{n+1}	$\overline{Q_{n+1}}$
\overline{CLR}	CLK				
0	X	X	X		
1		0	0		
1		1	0		
1		0	1		
1		1	1		
1	1	X	X		

Exercise:

1. Convert (i) T Flip- flop to JK Flip- flop (ii) Convert D Flip- flop to T Flip- flop
2. Convert JK Flip- flop to SR Flip-flop.
3. Construct SR Flip-flop using NOR gates.

Experiment 8 Counters

Aim:

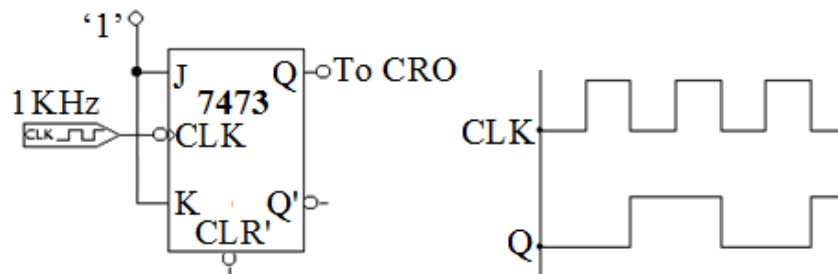
- To design and test asynchronous counters using Flip-Flops.
- To design and test synchronous counters using Flip-Flops.
- To test Johnson and Ring counters.

Components and Equipments Required:

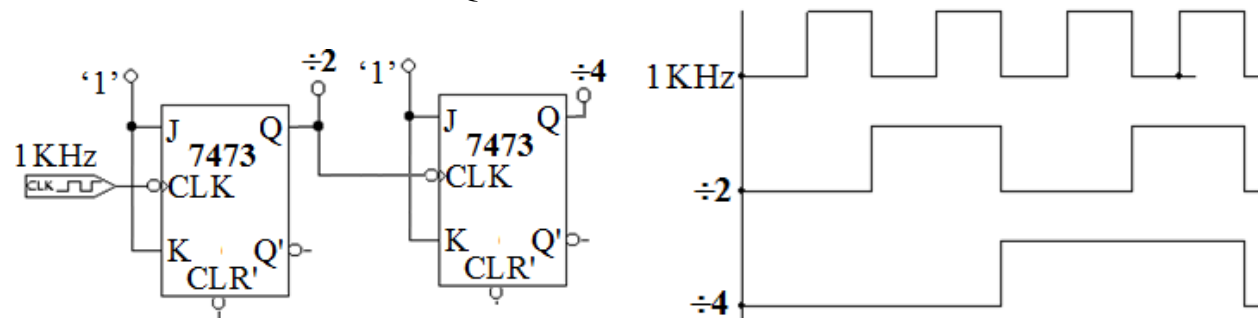
- ICs 7408, 7473, 7474
- IC trainer kit, connecting wires
- Cathode Ray Oscilloscope

Experiment:

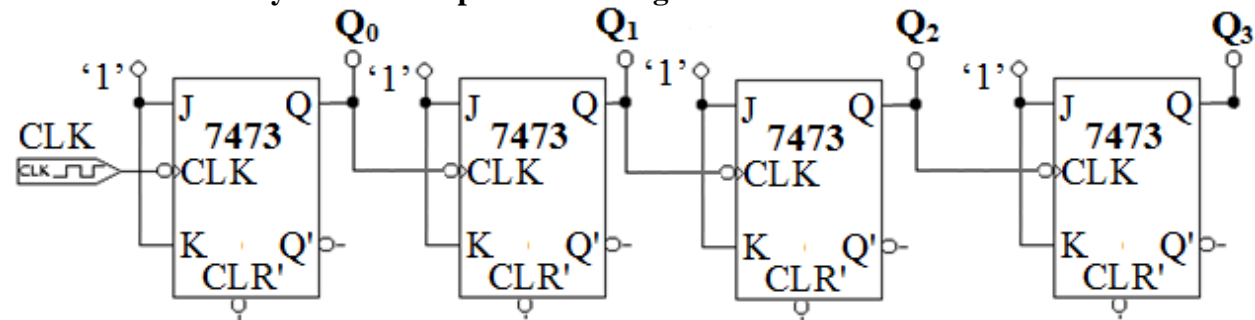
Part 1: Divide by 2 and divide by 4 counters using 7473:



***Note:** Observe the waveform for Q' also.



Part 2: Four bit Asynchronous up counter using two 7473 ICs:



Note down the outputs:

Input	Outputs				Input	Outputs			
Clock pulse	Q _D	Q _C	Q _B	Q _A	Clock pulse	Q _D	Q _C	Q _B	Q _A
0					8				
1					9				
2					10				
3					11				
4					12				
5					13				
6					14				
7					15				

Part 3: 4-bit Synchronous up – counter:

Excitation Table of J K Flip-Flop:

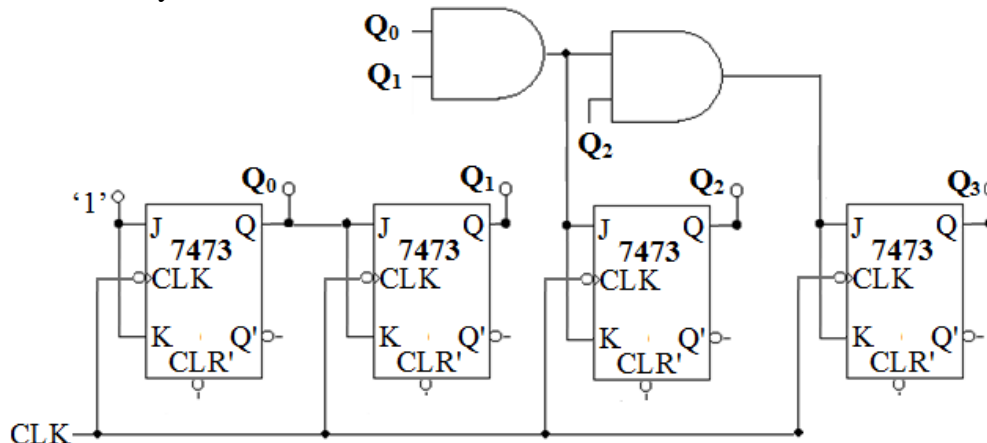
Q _n	Q _{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

- In the Excitation table shown above Q_n and Q_{n+1} are present and next state outputs. For a 4 bit counter, it is required to count from 0 to 15, which requires 4 J K Flip-Flops.
- Let the outputs of these Flip-Flops be Q_D, Q_C, Q_B, Q_A respectively and J_A, K_A be the inputs of Flip-Flop A. J_B, K_B of Flip-Flop B & so on.

State Table:

Q _D	Q _C	Q _B	Q _A	Q _{D+1}	Q _{C+1}	Q _{B+1}	Q _{A+1}	J _D	K _D	J _C	K _C	J _B	K _B	J _A	K _A
0	0	0	0	0	0	0	1	0	X	0	X	0	X	1	X
0	0	0	1	0	0	1	0	0	X	0	X	1	X	X	1
0	0	1	0	0	0	1	1	0	X	0	X	X	0	1	X
0	0	1	1	0	1	0	0	0	X	1	X	X	1	X	1
0	1	0	0	0	1	0	1	0	X	X	0	0	X	1	X
0	1	0	1	0	1	1	0	0	X	X	0	1	X	X	1
0	1	1	0	0	1	1	1	0	X	X	0	X	0	1	X
0	1	1	1	1	0	0	0	1	X	X	1	X	1	X	1
1	0	0	0	1	0	0	1	X	0	0	X	0	X	1	X
1	0	0	1	1	0	1	0	X	0	0	X	1	X	X	1
1	0	1	0	1	0	1	1	X	0	0	X	X	0	1	X
1	0	1	1	1	1	0	0	X	0	1	X	X	1	X	1
1	1	0	0	1	1	0	1	X	0	X	0	0	X	1	X
1	1	0	1	1	1	1	0	X	0	X	0	1	X	X	1
1	1	1	0	1	1	1	1	X	0	X	0	X	0	1	X
1	1	1	1	0	0	0	0	X	1	X	1	X	1	X	1

Draw K-maps for various J_s & K_s and obtain the logical expression for the same. Implement the logic circuit and verify.



Procedure:

1. Connect the circuit.
2. Momentarily clear all the inputs by giving $CLEAR = 0$ & then make it permanently high by giving $CLEAR = 1$.
3. Verify the circuit by observing the output for each clock pulse. After the 15th clock pulse (i.e. for the 16th clock pulse) output DCBA = 0000

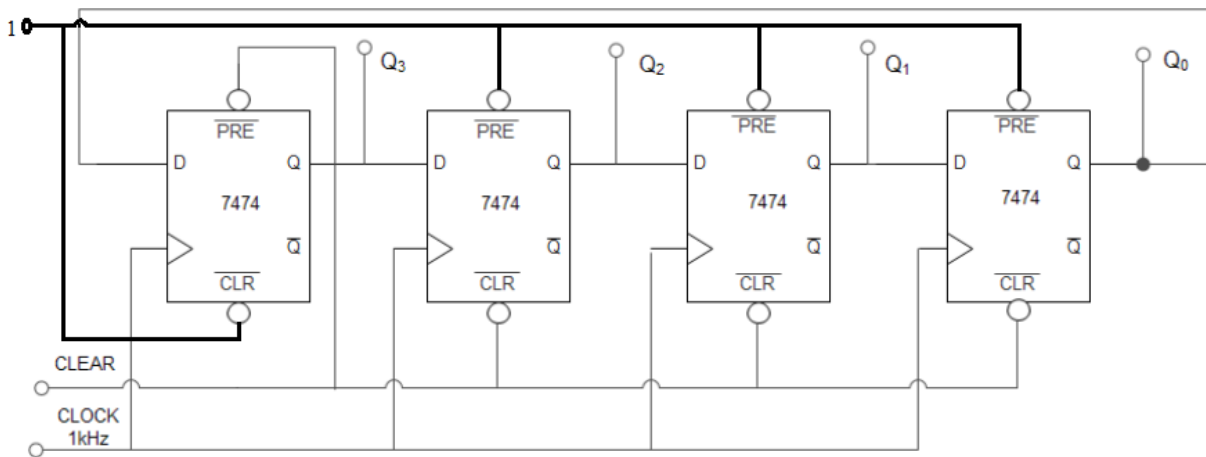
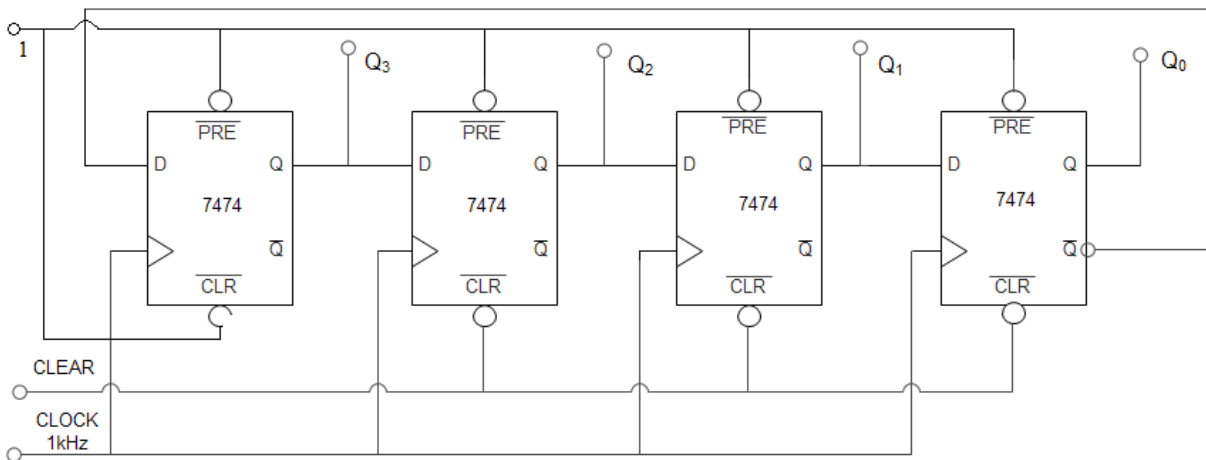
Tabular Column:

CLOCK PULSE	D	C	B	A	CLOCK PULSE	D	C	B	A
0					8				
1					9				
2					10				
3					11				
4					12				
5					13				
6					14				
7					15				

Johnson and Ring counters:

Procedure:

1. Connect the circuit as shown in the circuit diagram. Connect the clear input to logic level switch and Q_3 - Q_0 to LED indicators.
2. Clear all the Flip Flops momentarily by giving clear $CLEAR=0$ and then make it '1' permanently. Observe the output for each clock pulse.

Ring Counter:**Johnson Counter:****Tabular Columns:**

Johnson's counter:

Clock Pulse	Q ₃	Q ₂	Q ₁	Q ₀
0				
1				
2				
3				
4				
5				
6				
7				

Ring counter:

Clock Pulse	Q ₃	Q ₂	Q ₁	Q ₀
1				
2				
3				
4				
5				

Exercises:

1. Design and implement a decade counter and verify its operation.
2. Design and implement a counter for the following sequence.

CLK	Q ₀	Q ₁	Q ₂
0	0	0	0
1	0	0	1
2	0	1	0
3	1	0	0
4	1	0	1
5	1	1	0
6	0	0	0
7	0	0	1

3. Design a 3-bit up-down synchronous counter using JK Flip-Flop.
4. Design and test Synchronous counter for the sequence 0, 1, 3, 2, 6, 7, 5, 4, 0.
5. Design 4-bit Johnson and Ring counters using JK Flip-Flop.

Experiment 9 Shift Registers

Aim:

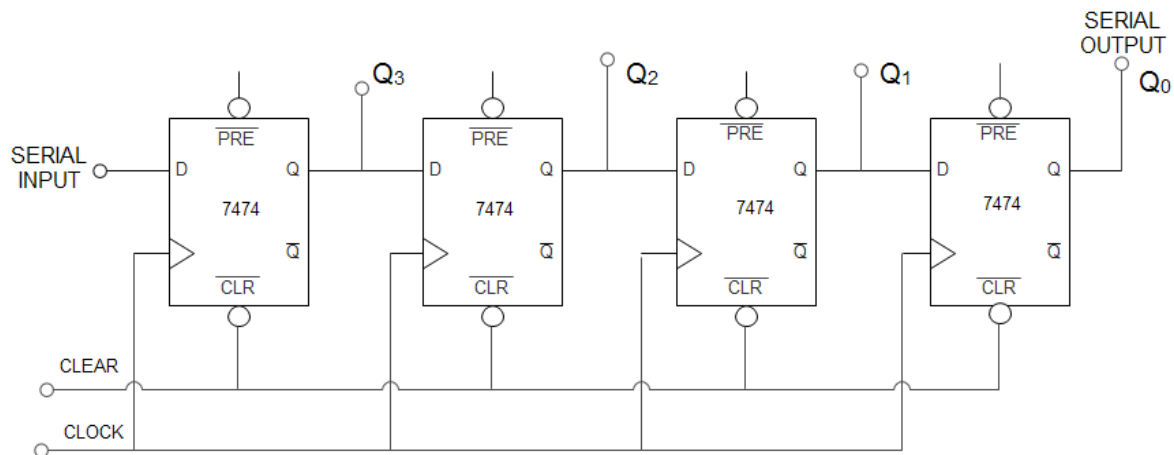
- a) To implement and test the following Shift registers.
 1. Serial-in Serial-out.
 2. Serial-in Parallel-out.
 3. Parallel-in Serial-out.
 4. Parallel-in Parallel-out.
- b) To study the operation of IC 74164.

Components Required:

IC 7474, IC 7408, IC 7400, IC 74164
IC Trainer kit, connecting wires

Part 1: For SISO, SIPO Shift Register:

Circuit diagram:



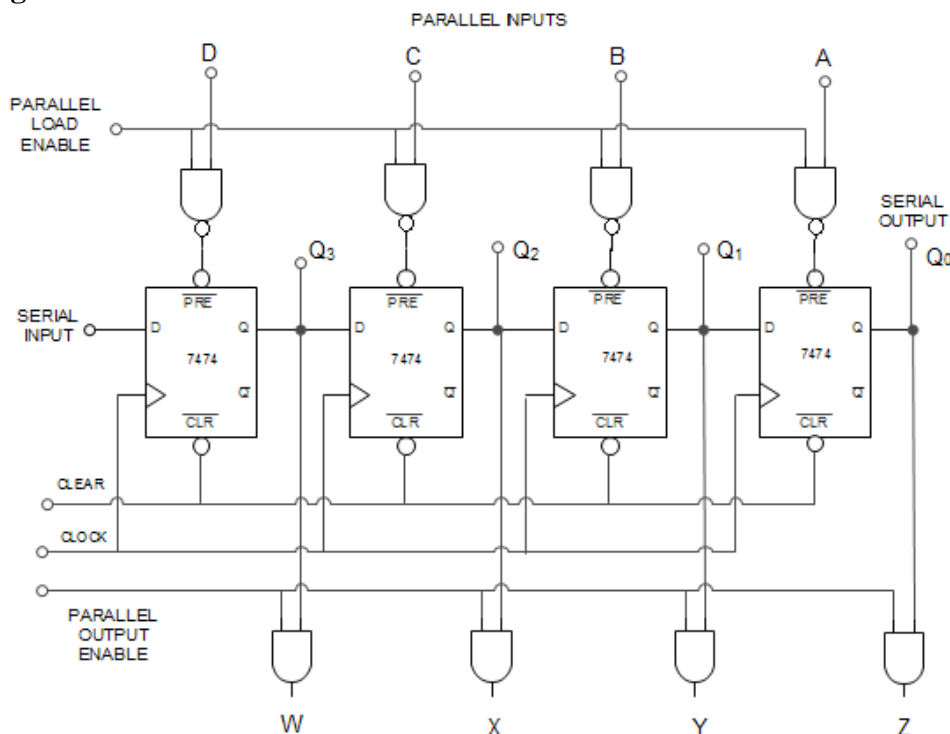
Procedure:

1. Connect the circuit as shown in the circuit diagram
2. Switch on & momentarily clear all the Flip-Flops by giving low to the clear input. Then make it permanently logic 1.
3. Enter any 4-bit serial input using clock pulse for every bit.
4. For example, to enter 1011
5. First make serial input =1, apply a clock pulse,
6. Make serial input=1, again apply a clock pulse,
7. Make serial input=0, again apply a clock pulse,
8. Make serial input=1, apply a clock pulse.
9. This ensures that the register has word 1011 stored in it.
10. Observe the data at the output of each shift register simultaneously.
11. This constitutes parallel output data.

- To observe the output data in serial fashion, apply clock pulse by keeping serial input data zero.

Part 2: For PISO, PIPO Shift Register:

Circuit Diagram:



Procedure:

- Make the clear and parallel enable low i.e. $CLR=0$, $PE=0$.
- To load the input data in parallel fashion, apply 4 bit data ($D C B A$) and make the parallel enable $PE=1$.
- To read the output data in parallel fashion, make parallel output enable =1 i.e. $PE=1$.
- To read the output data in serial fashion, make parallel output enable =0 i.e. $PE=0$ & $CLR=1$. Then apply clock pulse making serial input data=0. Data gets shifted from left Flip-Flop to right Flip-Flop for each clock pulse.

Part 3: To study the operation of IC 74164 – 8 bit shift register:

Exercises:

- Implement a 3-bit bidirectional shift Register.
- Implement a 3-bit universal shift register.
- Implement a 12-bit parallel to serial converter using 7495 ICs and basic gates.
- Implement a 16-bit serial to parallel converter using 7495 ICs and 1 kHz clock frequency.

Experiment 10

Sequential circuits

Aim:

- To design a sequence code detector which will detect the sequence of 1010 in overlapping input sequence.
- To implement maximum length sequence generation using 3-bit shift registers.

Components and Equipments Required:

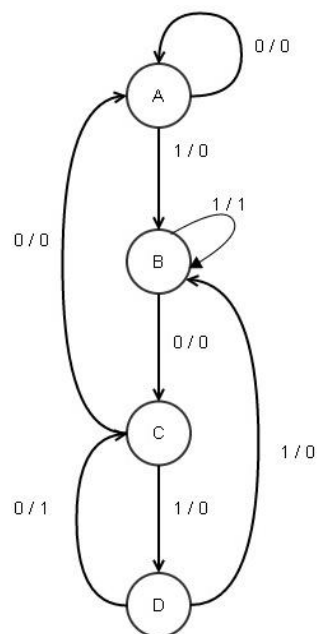
- IC's 7474, 7486, 7473, 7408, IC 7404, IC 7432.
- IC trainer kit, connecting wires.

Experiment:

Part 1: Sequence Detector:

Procedure:

- Develop the state diagram and state table:

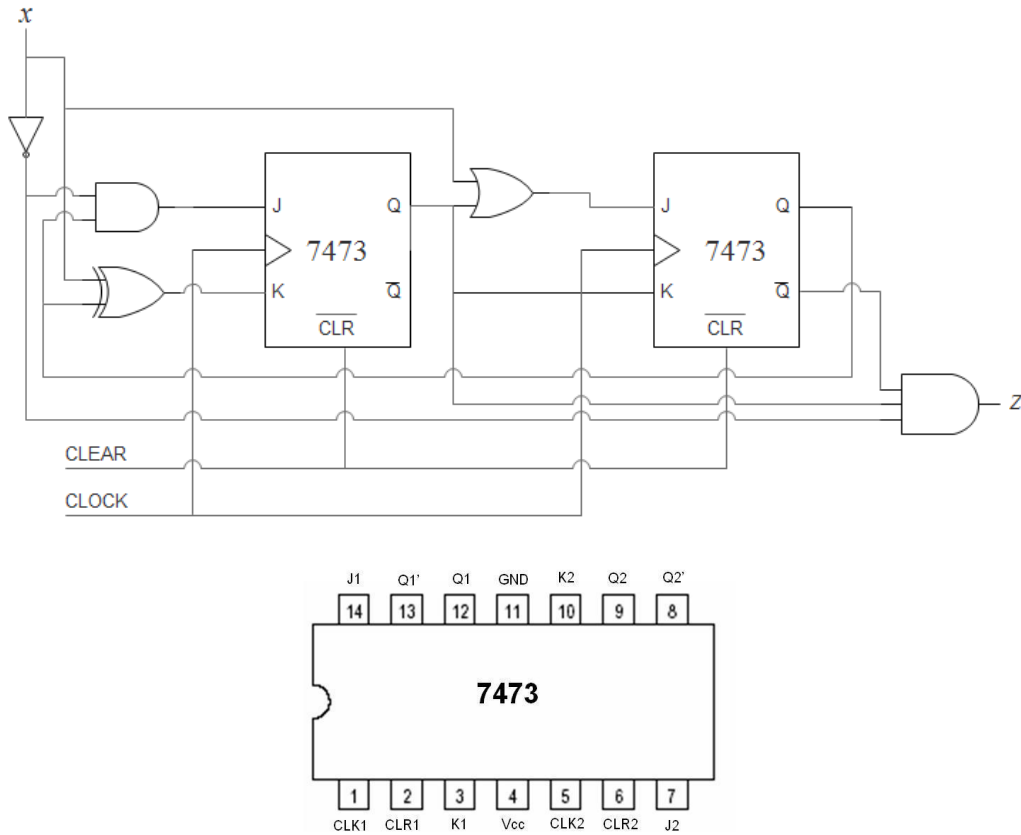


State	Present State		Input	Next State		Output
	A	B		A _n	B _n	
0	0	0	0	0	0	0
1	0	0	1	0	1	0
2	0	1	0	1	1	0
3	0	1	1	0	1	0
6	1	1	0	0	0	0
7	1	1	1	1	0	0
4	1	0	0	1	1	1
5	1	0	1	0	1	0

- On solving using JK flip-flop and K-map following logical expressions are obtained.

$$J_A = BX' \quad K_A = BX' + B'X \quad J_B = X+A \quad K_B = A \quad Z = AB'X'$$

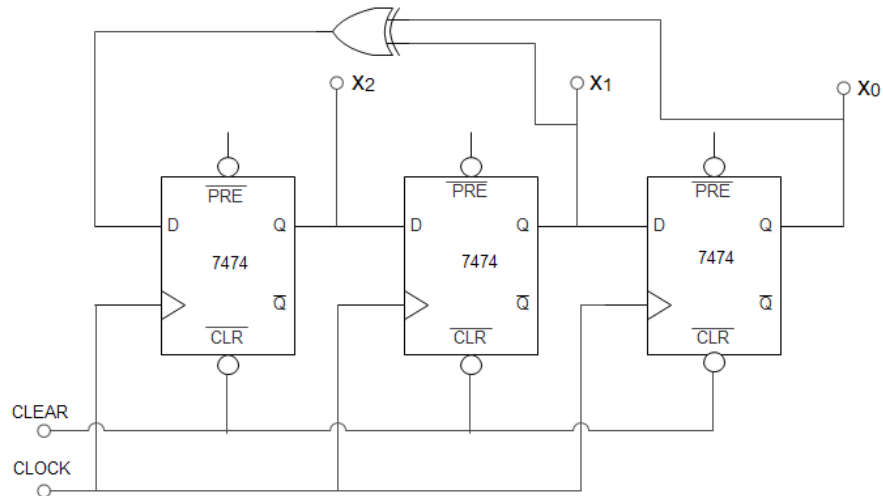
***Note:** Use 7473 IC Dual JK negative edge triggered Flip- flop. The pinout diagram and circuit diagram is shown below.



***Note:** 1. Connect CLR to HIGH

2. CLK1 to HIGH and CLK2 to CLOCK PULSE

Part 2: Sequence Generation:



Procedure:

- Rig up the circuit as shown.
- Load the data other than “000” into the register using asynchronous inputs.
- Apply clock and note down the sequences.

***Note:** assuming the initial data is 100 then sequence will be

100 – 010 – 101 – 110 – 111 – 011 – 001 – 100 – 100

Appendix A:

Pinout Diagrams of various ICs:

