

BACHELOR OF COMPUTER APPLICATION LAB MANUAL

2nd Semester



Prepared By
Pure and Applied Science Dept.
Computer Application

MIDNAPORE CITY COLLEGE



INSTRUCTIONS TO STUDENTS

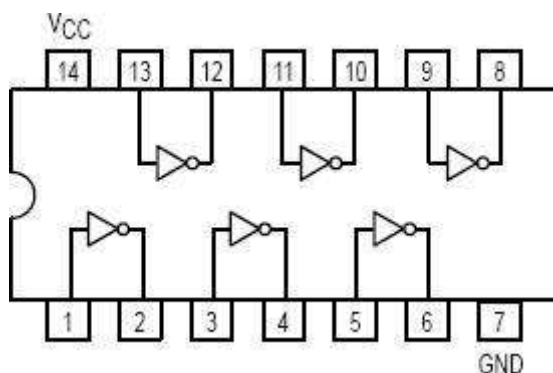
- Before entering the lab, the student should carry the following things (MANDATORY)
 1. Identity card issued by the college.
 2. Class notes
 3. Lab observation book
 4. Lab Manual
 5. Lab Record
- Student must sign in and sign out in the register provided when attending the lab session without fail.
- Come to the laboratory in time. Students, who are late more than 10 min., will not be allowed to attend the lab.
- Students need to maintain 80% attendance in lab if not a strict action will be taken.
- All students must follow a Dress Code while in the laboratory
- Foods, drinks are NOT allowed.
- All bags must be left at the indicated place.
- Refer to the lab staff if you need any help in using the lab.
- Respect the laboratory and its other users.
- Workspace must be kept clean and tidy after experiment is completed.
- Read the Manual carefully before coming to the laboratory and be sure about what you are supposed to do.
- Do the experiments as per the instructions given in the manual.
- Copy all the programs to observation which are taught in class before attending the lab session.
- Students are not supposed to use floppy disks, pen drives without permission of lab-in-charge.
- Lab records need to be submitted on or before the date of submission.

DIGITAL ELECTRONICS
LABORATORY MAUAL
(Course Code: BCAHMJ02P)

CONTENTS

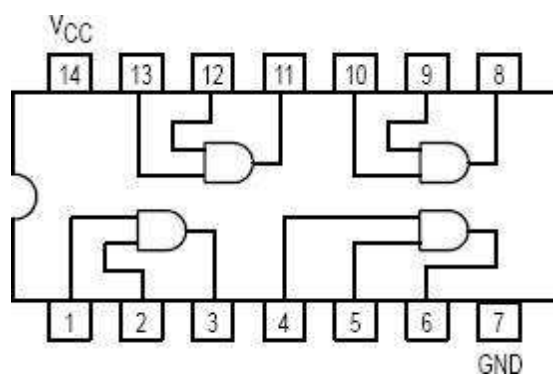
Experiment No

- 1. Verification of Gates***
- 2. Half/Full Adder/Subtractor***
- 3. Parallel Adder/Subtractor***
- 4. Excess-3 to BCD & Vice Versa***
- 5. Binary-Grey & Grey-Binary Converter***
- 6. MUX/DEMUX***
- 7. MUX/DEMUX using only NAND Gates***
- 8. Comparators***
- 9. Encoder/Decoder***
- 10. Flip-Flops***
- 11. Counters***
- 12. Shift Registers***
- 13. Johnson/Ring Counters***
- 14. Sequence Generator***
- 15. Multivibrators***
- 16. Static RAM***



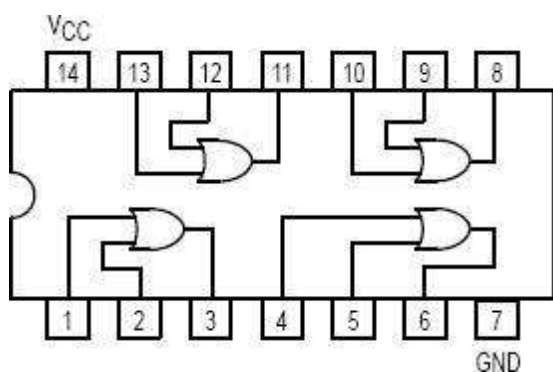
A	O/P	Y1 (V)	Y2 (V)	Y3 (V)	Y4 (V)	Y5 (V)	Y6 (v)
0	1						
1	0						

2-Input AND Gate 7408LS



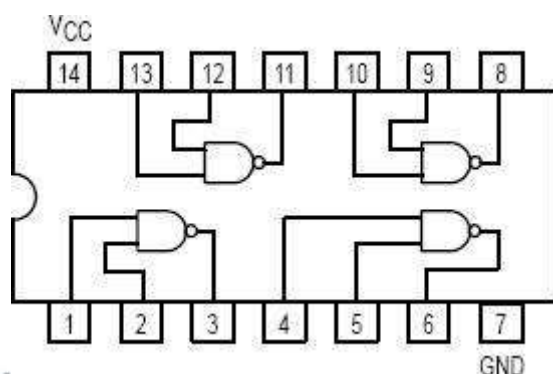
A	B	O/P	Y1 (V)	Y2 (V)	Y3 (V)	Y4 (V)
0	0	0				
0	1	0				
1	0	0				
1	1	1				

2-Input OR Gate 7432LS



A	B	O/P	Y1 (V)	Y2 (V)	Y3 (V)	Y4 (V)
0	0	0				
0	1	0				
1	0	0				
1	1	1				

2-Input NAND Gate 7400LS



A	B	O/P	Y1 (V)	Y2 (V)	Y3 (V)	Y4 (V)
0	0	1				
0	1	0				
1	0	0				
1	1	0				

VERIFICATION OF GATES

Aim: - To study and verify the truth table of logic gates

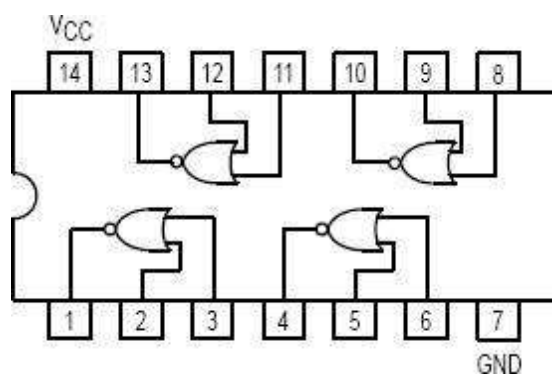
Apparatus Required: -

All the basic gates mention in the fig.

Procedure: -

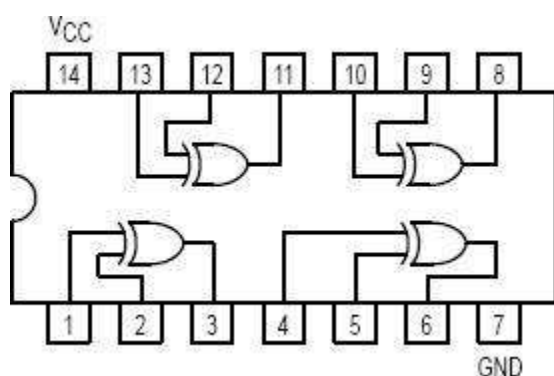
- 1 **Place the IC-on-IC Trainer Kit.**
- 2 **Connect V_{CC} and ground to respective pins of IC Trainer Kit.**
- 3 **Connect the inputs to the input switches provided in the IC Trainer Kit.**
- 4 **Connect the outputs to the switches of O/P LEDs,**
- 5 **Apply various combinations of inputs according to the truth table and observe condition of LEDs.**
- 6 **Disconnect output from the LEDs and note down the corresponding multimeter voltage readings for various combinations of inputs.**

2-Input NOR Gate 7402LS



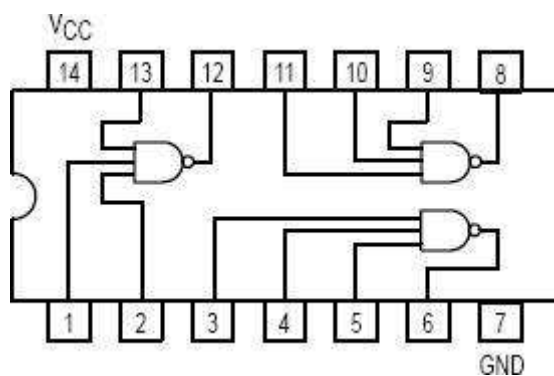
A	B	O/P	Y1 (V)	Y2 (V)	Y3 (V)	Y4 (V)
0	0	1				
0	1	0				
1	0	0				
1	1	0				

2 Input EX-OR Gate 7486LS



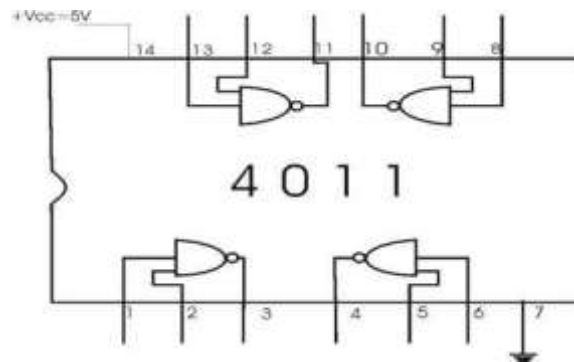
A	B	O/P	Y1 (V)	Y2 (V)	Y3 (V)	Y4 (V)
0	0	0				
0	1	1				
1	0	1				
1	1	0				

3 Input NAND Gate 7410LS



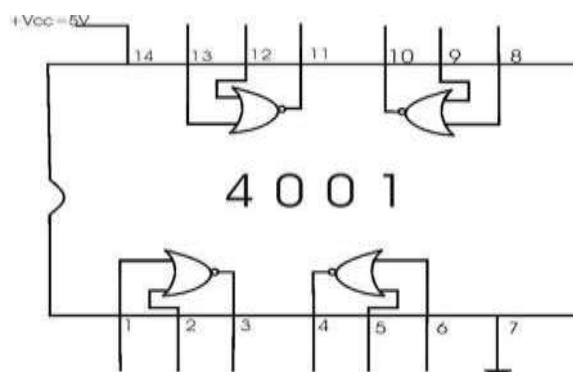
A	B	C	O/P	Y1 (V)	Y2 (V)	Y3 (V)
0	0	0	1			
0	0	1	1			
0	1	0	1			
0	1	1	1			
1	0	0	1			
1	0	1	1			
1	1	0	1			
1	1	1	0			

2-Input NAND Gate CD4011



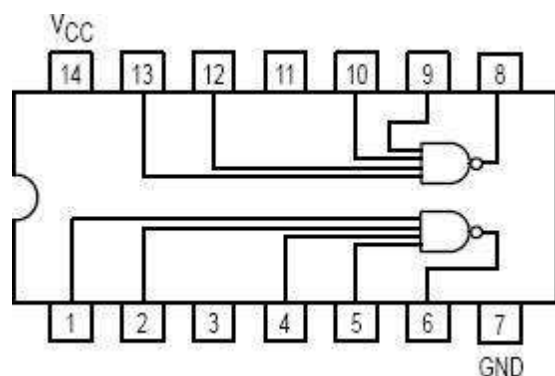
A	B	O/P	Y1 (V)	Y2 (V)	Y3 (V)	Y4 (V)
0	0	1				
0	1	1				
1	0	1				
1	1	0				

2-Input NOR Gate € CD4001



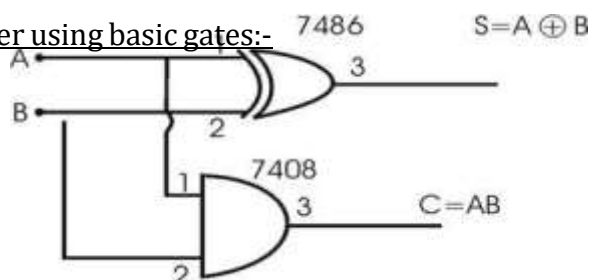
A	B	O/P	Y1 (V)	Y2 (V)	Y3 (V)	Y4 (V)
0	0	1				
0	1	0				
1	0	0				
1	1	0				

4-Input NAND Gate 7420LS

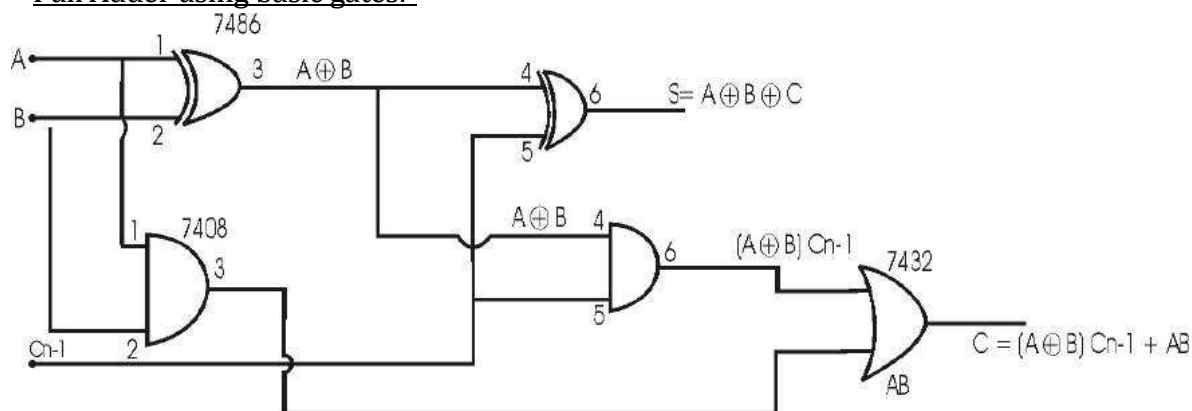


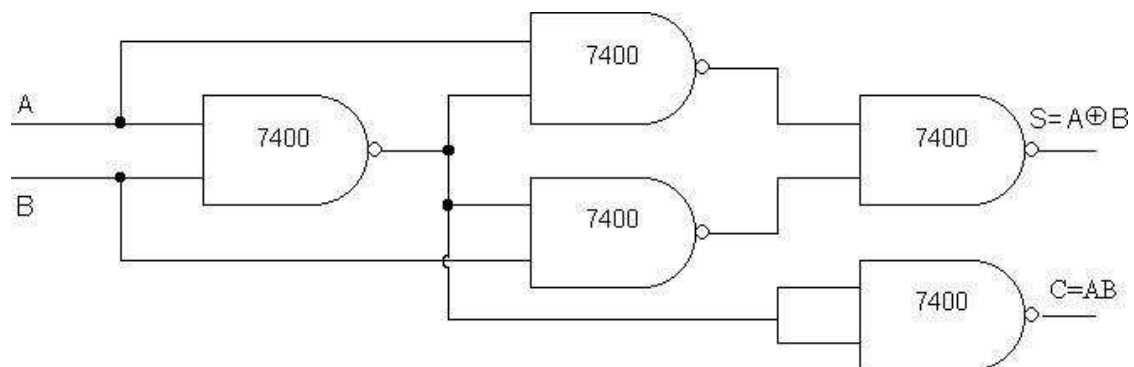
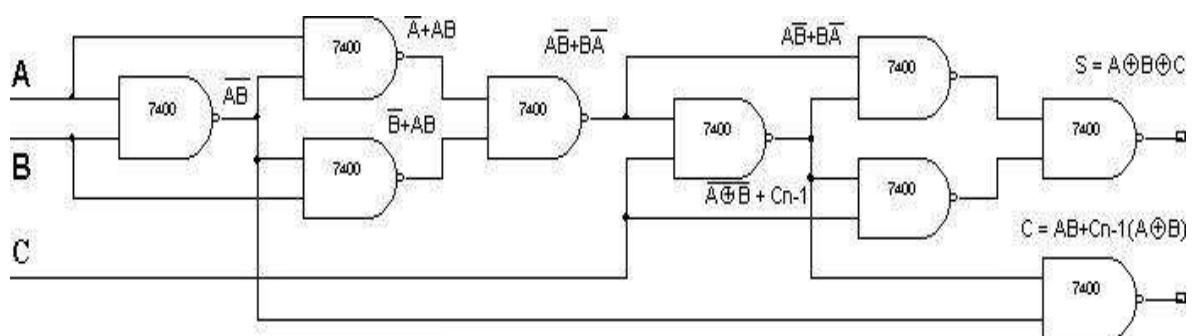
A	B	C	D	O/P	Y1 (V)	Y2 (V)	Y3 (V)
0	0	0	0	1			
0	0	0	1	1			
0	0	1	0	1			
0	0	1	1	1			
0	1	0	0	1			
0	1	0	1	1			
0	1	1	0	1			
0	1	1	1	1			
1	0	0	0	1			
1	0	0	1	1			
1	0	1	0	1			
1	0	1	1	1			
1	1	0	0	1			
1	1	0	1	1			
1	1	1	0	1			
1	1	1	1	0			

Half Adder using basic gates:-



Full Adder using basic gates:-



Half Adder using NAND gates only:-Full Adder using NAND gates only:-

Experiment No:

Date: __/__/

HALF/FULL ADDER & HALF/FULL SUBTRACTOR

Aim: - To realize half/full adder and half/full subtractor.

- i. Using X-OR and basic gates**
- i. Using only nand gates.**

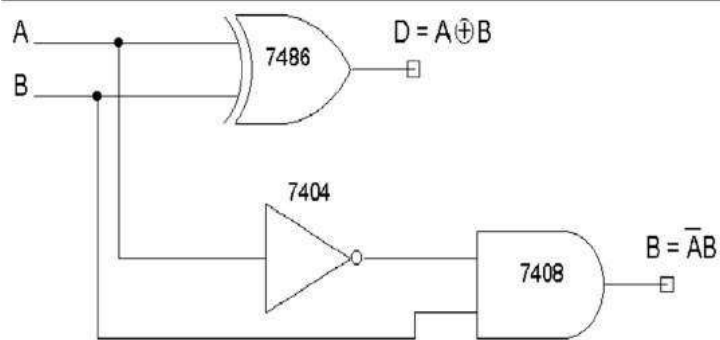
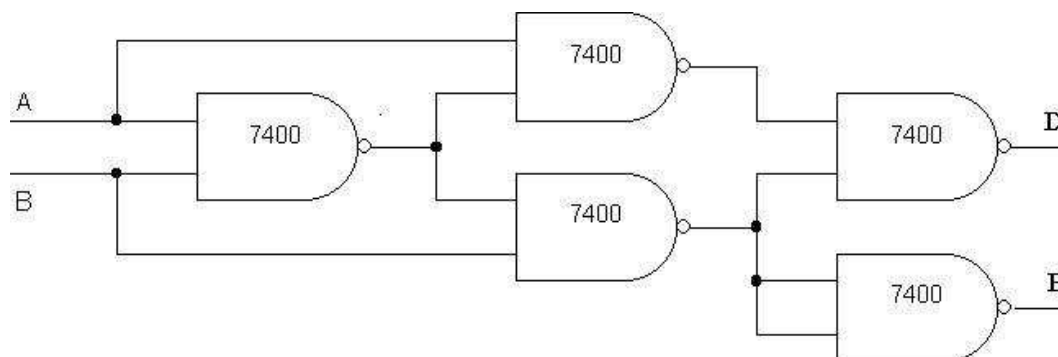
Apparatus Required: -

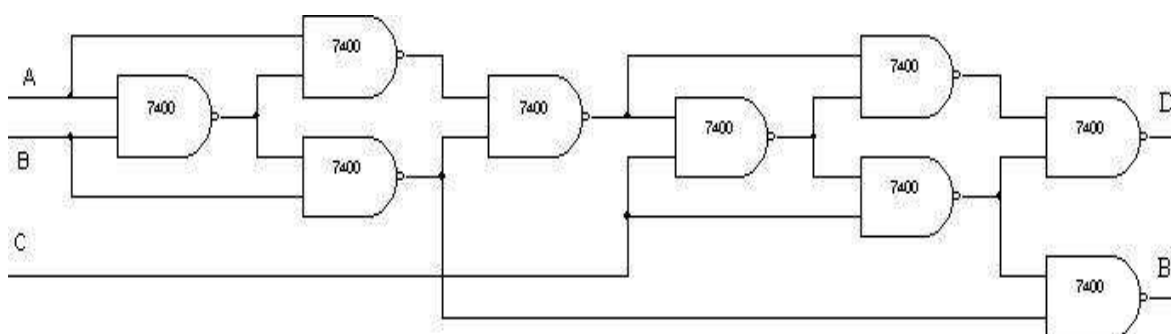
IC 7486, IC 7432, IC 74 08, IC 7400, etc.

Procedure: -

- 1. Verify the gates.**
- 2. Make the connections as per the circuit diagram.**
- 3. Switch on V_{CC} and apply various combinations of input according to the truth table.**
- 4. Note down the output readings for half/full adder and half/full subtractor sum/difference and the carry/borrow bit for different combinations of inputs.**

Using X - OR and Basic Gates (a)Half Subtractor

Full Subtractor**(i) Using only NAND gates (a) Half subtractor****(b) Full Subtractor**



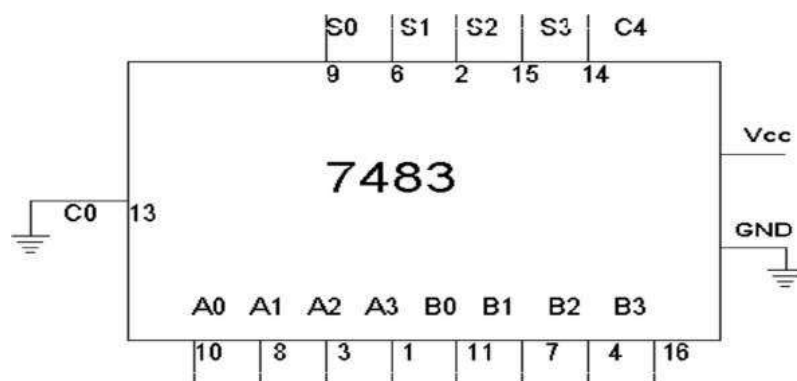
Half Adder					
A	B	S	C	S(V)	C(V)
	0	0	0		
0	1	1	0		
1	0	1	0		
1	1	0	1		

Half Subtractor					
A	B	D	B	D(V)	B(V)
0	0	0	0		
0	1	1	1		
1	0	1	0		
1	1	0	0		

Full Adder						
A	B	Cn-1	S	C	S(V)	C(V)
0	0	0	0	0		
0	0	1	1	0		
0	1	0	1	0		
0	1	1	0	1		
1	0	0	1	0		
1	0	1	0	1		
1	1	0	0	1		
1	1	1	1	1		

Full Subtractor						
A	B	Cn-1	D	B	D(v)	B(v)
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		

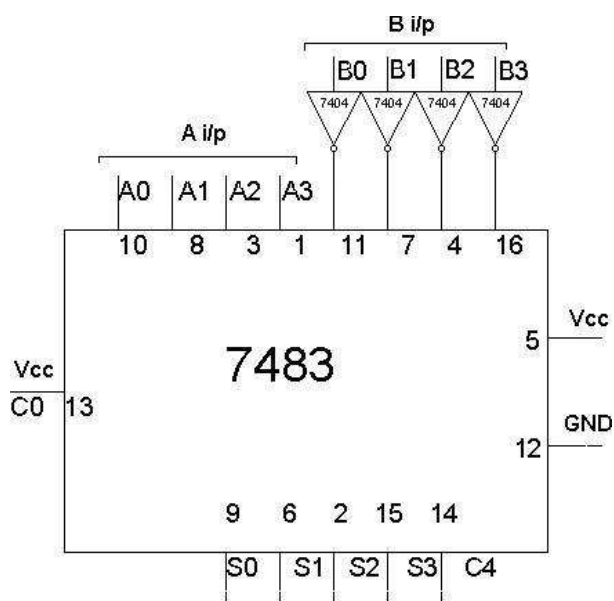
Adder :-



Truth Table :-

A3	A2	A1	A0	B3	B2	B1	B0	C4 (V)	S3(V)	S2(V)	S1(V)	S0(V)
0	0	0	1	0	0	1	0	0	0	0	1	1
0	1	0	1	1	0	1	1	1	1	0	0	0
1	0	1	0	1	0	1	0	1	0	1	0	0
1	1	1	1	1	1	1	1	1	1	1	1	0
0	1	1	1	0	0	1	1	0	1	0	1	0

Subtractor:-



Experiment No: _____

Date: __/__/

PARALLEL ADDER AND SUBTRACTOR USING 7483Aim: - To realize IC7483 as parallel adder / Subtractor.Apparatus Required: -

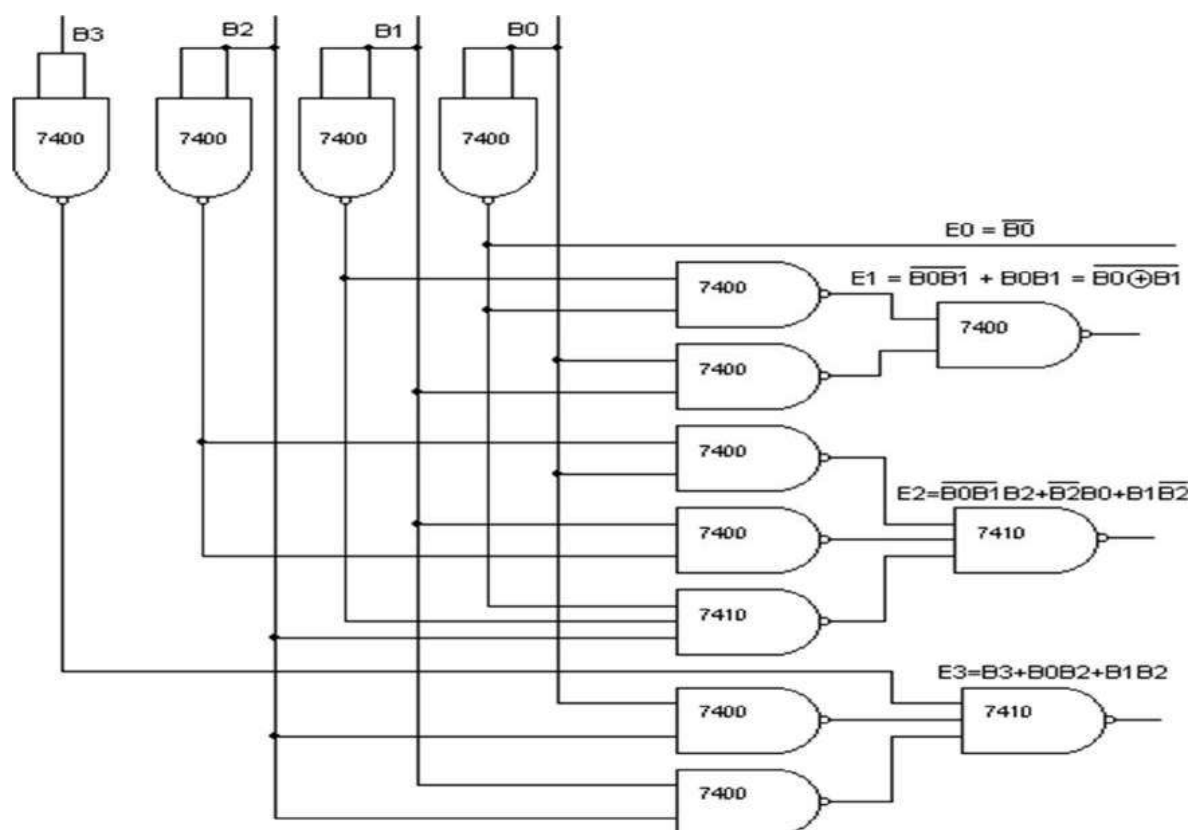
IC 7483, IC 7404, etc.

Procedure: -

- 1 Apply the inputs to A0 to A3 and B0 to B3.
- 2 Connect C0 to the Ground.
- 3 Check the output sum on the S0 to S3 and also C4.
- 4 For subtraction connect C0 to Vcc, Apply the B input through NOT gate, which gives the complement of B.
- 5 The truth table of adder and Subtractor are noted down.

Truth Table for Subtractor

A3	A2	A1	A0	B3	B2	B1	B0	C4(V)	S3(V)	S2(V)	S1(V)	S0(V)
0	0	1	0	0	0	0	1	1	0	0	0	1
0	1	0	1	0	0	1	1	1	0	0	1	0
0	0	1	1	0	1	0	1	0	1	1	1	0
1	0	1	0	0	1	1	0	1	0	1	0	0
1	0	0	0	1	1	1	1	0	1	0	0	1
1	0	1	0	0	1	1	0	1	0	1	0	0
1	0	0	0	1	1	1	1	0	1	0	0	1

BCD To Excess-3Truth Table For Code Conversion: -

Inputs				Outputs			
B3	B2	B1	B0	E3 (v)	E2 (v)	E1 (v)	E0 (v)
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

Experiment No:

Date: __/__/__

BCD to Excess 3 AND Excess 3 to BCD

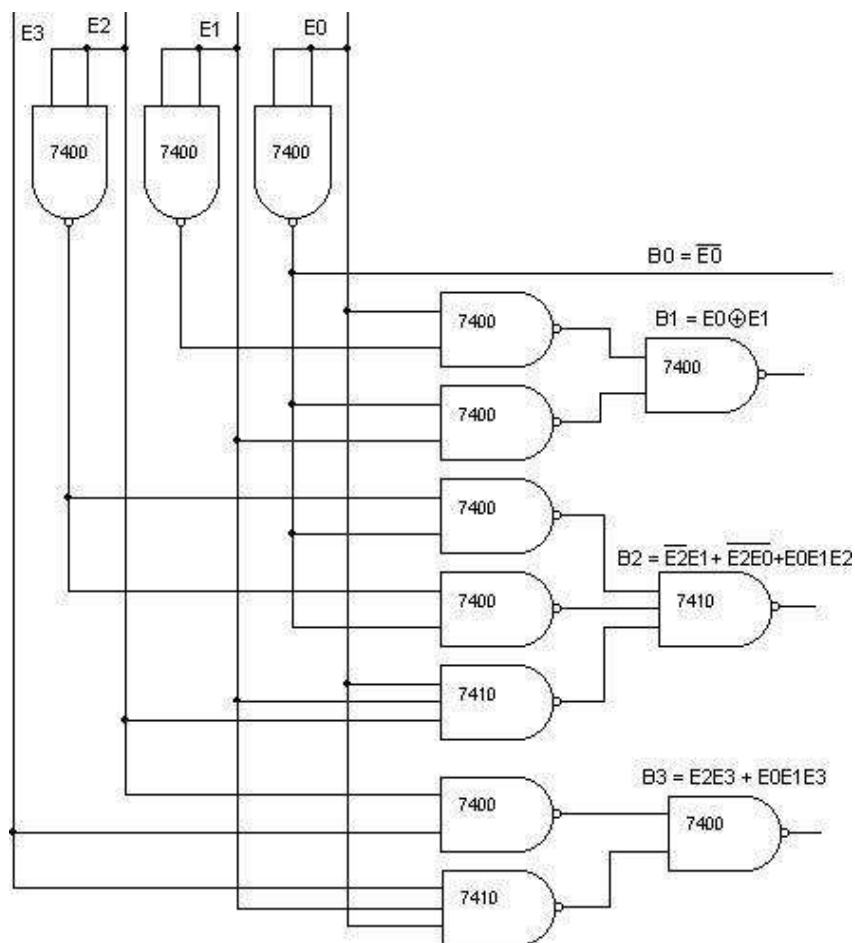
Aim: - To verify BCD to excess -3 code conversion using NAND gates. To study and verify the truth table of excess-3 to BCD code converter

Apparatus Required: -

IC 7400, IC 7404, etc.

Procedure: - (BCD Excess 3 and Vice Versa)

- 1 Make the connections as shown in the fig.
- 2 Pin [14] of all IC'S are connected to +5V and pin [7] to the ground.
- 3 The inputs are applied at E3, E2, E1, and E0 and the corresponding outputs at B3, B2, B1, and B0 are taken for excess - 3 to BCD.
- 4 B3, B2, B1, and B0 are the inputs, and the corresponding outputs are E3, E2, E1 and E0 for BCD to excess - 3.
- 5 Repeat the same procedure for other combinations of inputs.
- 6 Truth table is written.

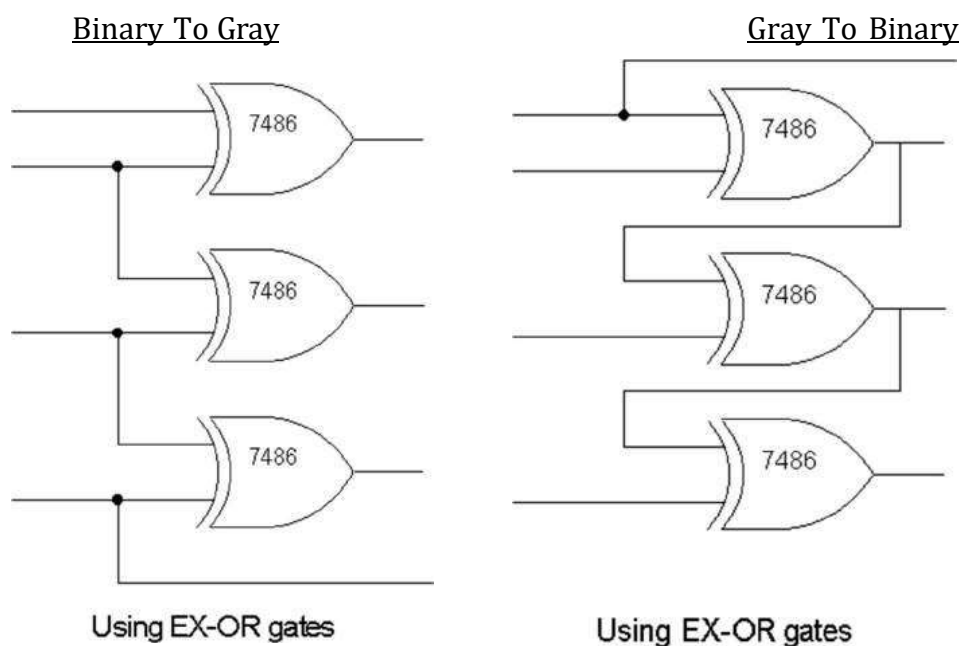
Excess-3 To BCD :-Truth Table For Code Conversion: -

Inputs				Outputs			
E3	E2	E1	E0	B3 (v)	B2 (v)	B1 (v)	B0(v)
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	0
0	1	1	0	0	0	1	1
0	1	1	1	0	1	0	0
1	0	0	0	0	1	0	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	1	1
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	1

Exercise: -

- 1 Obtain the expression for E3, E2, E1 and E0**
- 2 Obtain the expression for B3, B2, B1 and B0**

Circuit Diagram: -



Truth Table For Both: -

Inputs				Outputs			
B3	B2	B1	B0	G3 (V)	G2 (V)	G1 (V)	G0 (V)
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

Experiment No:

Date: __/__/__

BINARY TO GRAY AND GRAY TO BINARY**CONVERSION**

Aim: - To convert given binary numbers to gray codes.

Apparatus Required: -

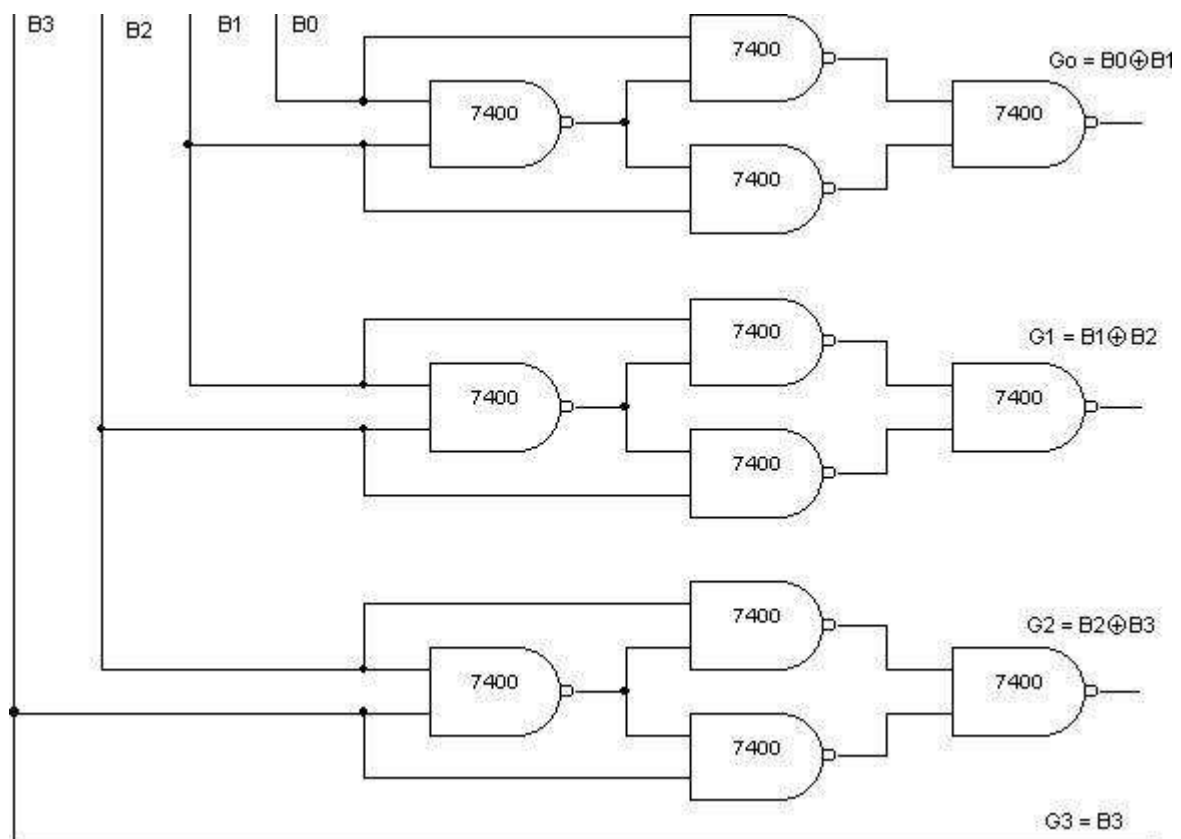
IC 7486, etc

Procedure: -

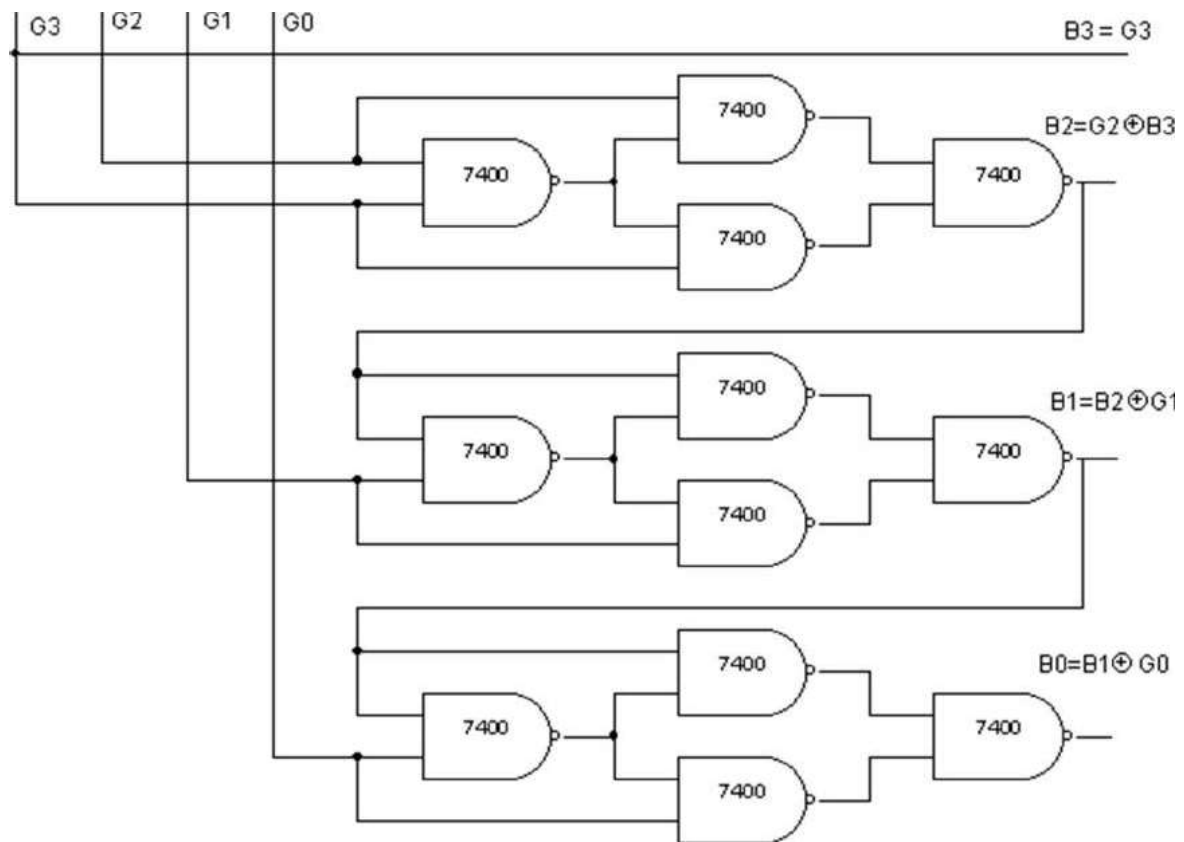
- 1 The circuit connections are made as shown in fig.
- 2 Pin (14) is connected to +Vcc and Pin (7) to ground.
- 3 In the case of binary to gray conversion, the inputs B0, B1, B2 and B3 are given at respective pins and outputs G0, G1, G2, G3 are taken for all the 16 combinations of the input.
- 4 In the case of gray to binary conversion, the inputs G0, G1, G2 and G3 are given at respective pins and outputs B0, B1, B2, and B3 are taken for all the 16 combinations of inputs.
- 5 The values of the outputs are tabulated.

Using Nand Gates

Only: - Binary To Gra

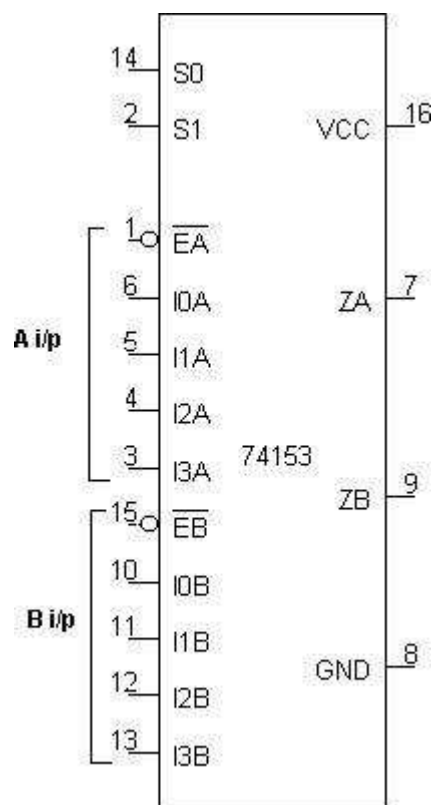


Gray Code



Truth Table For Both: -

Inputs				Outputs			
B3	B2	B1	B0	G3 (V)	G2 (V)	G1 (V)	G0 (V)
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

Pin Details: -Truth Table: -

CHANNEL – A							
INPUTS					SELECT LINES		O/P
$\bar{E}a$	Ioa	I1a	I2a	I3a	S1	S2	Za(v)
1	X	X	X	X	X	X	0
0	0	X	X	X	0	0	0
0	1	X	X	X	0	0	1
0	X	0	X	X	0	1	0
0	X	1	X	X	0	1	1
0	X	X	0	X	1	0	0
0	X	X	1	X	1	0	1
0	X	X	X	0	1	1	0
0	X	X	X	1	1	1	1

CHANNEL – B							
INPUTS					SELECT LINES		O/P
$\bar{E}a$	Iob	I1b	I2b	I3b	S1	S2	Za(v)
1	X	X	X	X	X	X	0
0	0	X	X	X	0	0	0
0	1	X	X	X	0	0	1
0	X	0	X	X	0	1	0
0	X	1	X	X	0	1	1
0	X	X	0	X	1	0	0
0	X	X	1	X	1	0	1
0	X	X	X	0	1	1	0
0	X	X	X	1	1	1	1

Experiment No: _____

Date: __/__/

MUX/DEMUX USING 74153 & 74139

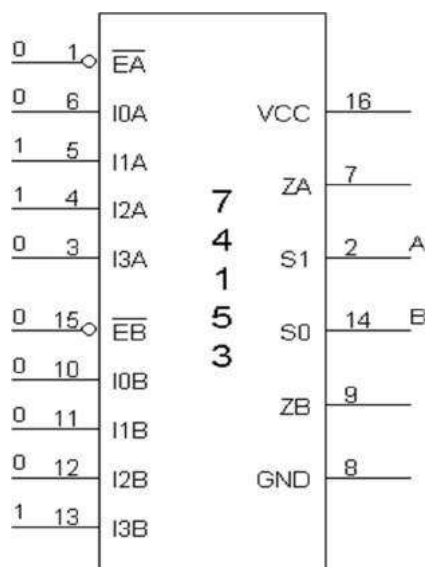
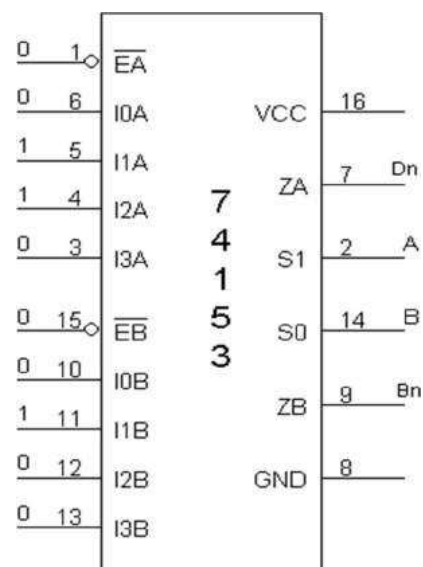
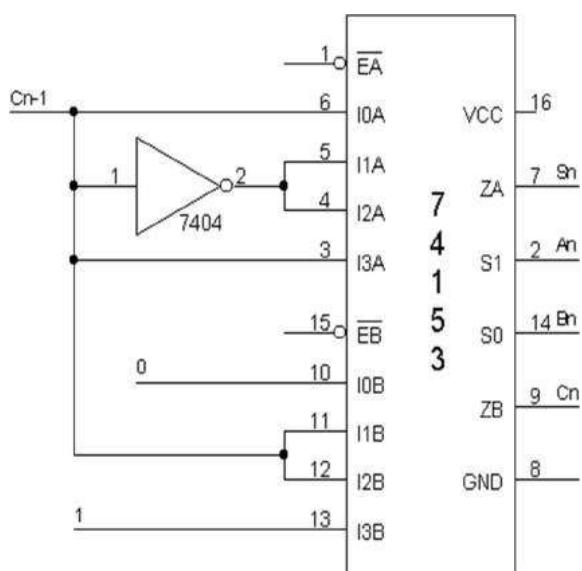
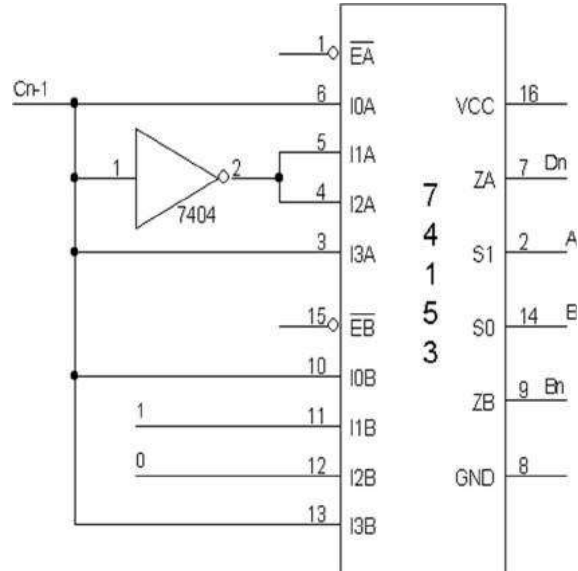
Aim: - To verify the truth table of multiplexer using 74153 & to verify a demultiplexer using 74139. To study the arithmetic circuits half-adder half Subtractor, full adder and full Subtractor using multiplexer.

Apparatus Required: -

IC 74153, IC 74139, IC 7404, etc.

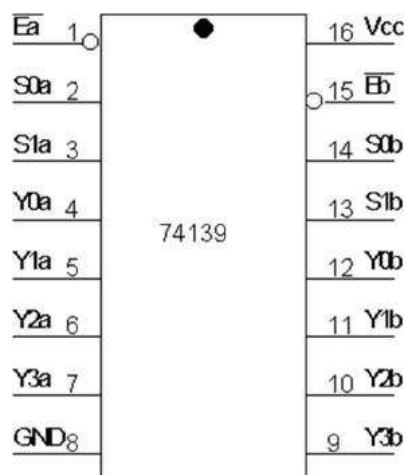
Procedure: - (IC 74153)

1. The Pin [16] is connected to + Vcc.
2. Pin [8] is connected to ground.
3. The inputs are applied either to 'A' input or 'B' input.
4. If MUX 'A' has to be initialized, Ea is made low and if MUX 'B' has to be initialized, Eb is made low.
5. Based on the selection lines one of the inputs will be selected at the output and thus the truth table is verified.
6. In case of half adder using MUX, sum and carry is obtained by applying a constant inputs at I_{0a}, I_{1a}, I_{2a}, I_{3a} and I_{0b}, I_{1b}, I_{2b} and I_{3b} and the corresponding values of select lines are changed as per table and the output is taken at Z_{0a} as sum and Z_{0b} as carry.
7. In this case, the channels A and B are kept at constant inputs according to the table and the inputs A and B are varied. Making Ea and Eb zero and the output is taken at Za, and Zb.
8. In full adder using MUX, the input is applied at C_{n-1}, A_n and B_n. According to the table corresponding outputs are taken at C_n and D_n.

Half Adder Using 74153 -Half Subtractor: -Full Adder Using 74153: -Full Subtractor Using 74153: -

Truth Tables: - Same for both Subtractor and adder

Half adder/subtractor				Full Adder/subtractro				
A	B	Sn/Dn (V)	Cn/Bn (V)	An	Bn	Cn-1	Sn/Dn (V)	Cn/Bn (V)
0	0			0	0	0		
0	1			0	0	1		
1	0			0	1	0		
1	1			0	1	1		
				1	0	0		
				1	0	1		
				1	1	0		
				1	1	1		

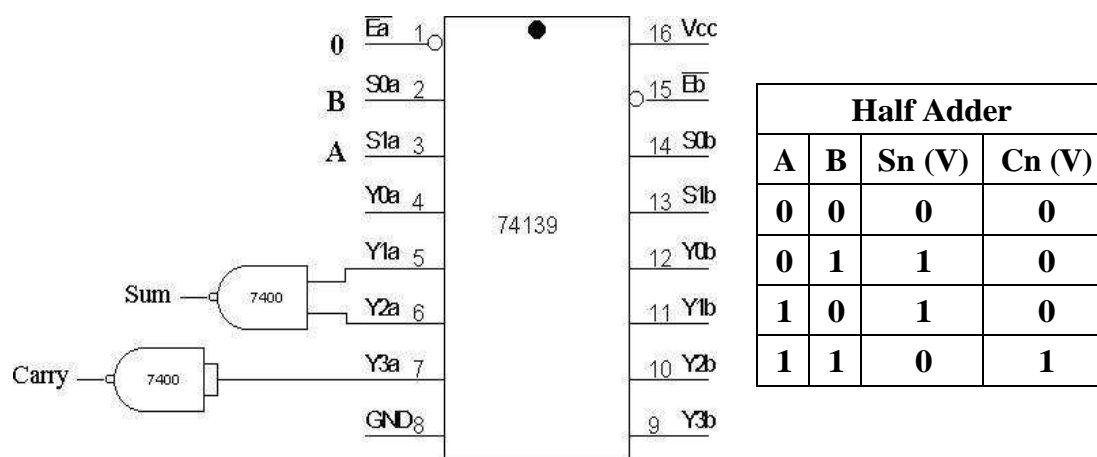
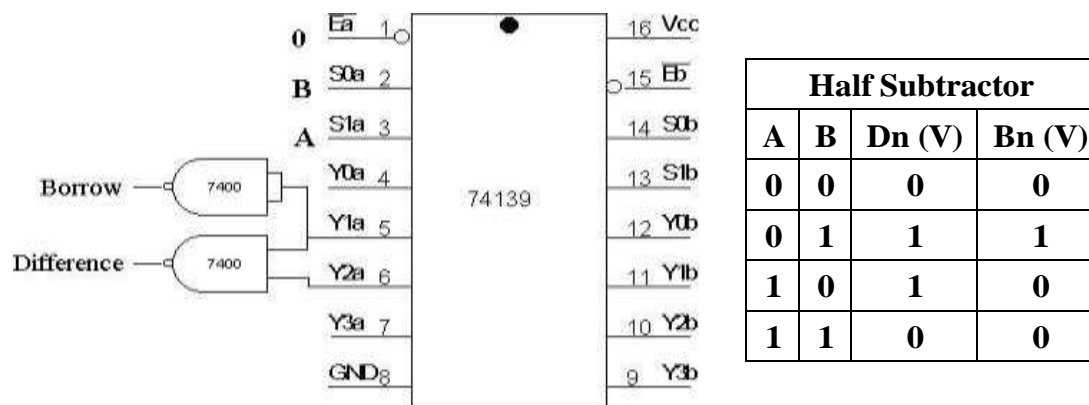
Pin Details: -Truth Table For Demux: -

CHANNEL – A						
Inputs			Outputs			
$\bar{E}a$	S1a	S0a	Y0a	Y1a	Y2a	Y3a
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

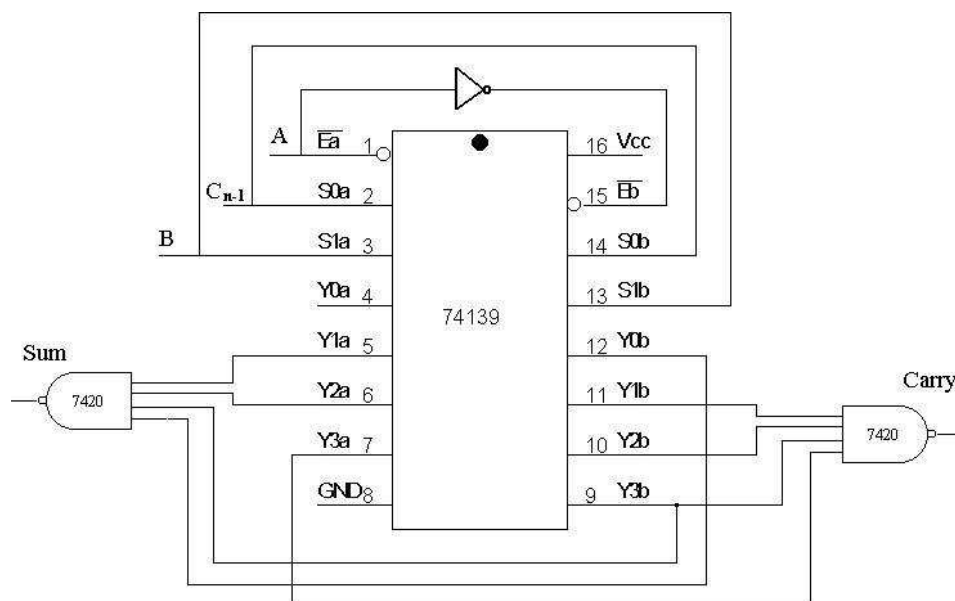
CHANNEL – B						
Inputs			Outputs			
$\bar{E}b$	S1b	S0b	Y0b	Y1b	Y2b	Y3b
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

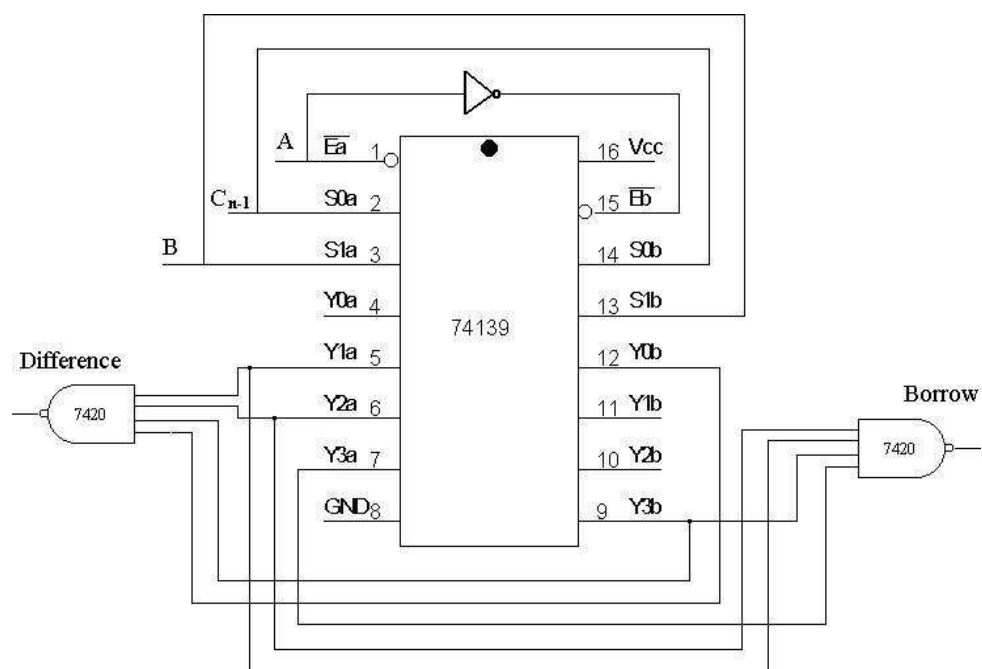
Procedure: - (IC 74139)

1. The inputs are applied to either 'a' input or 'b' input
2. The demux is activated by making Ea low and Eb low.
3. The truth table is verified.

Half adderHalf subtractor:-Exercise:-

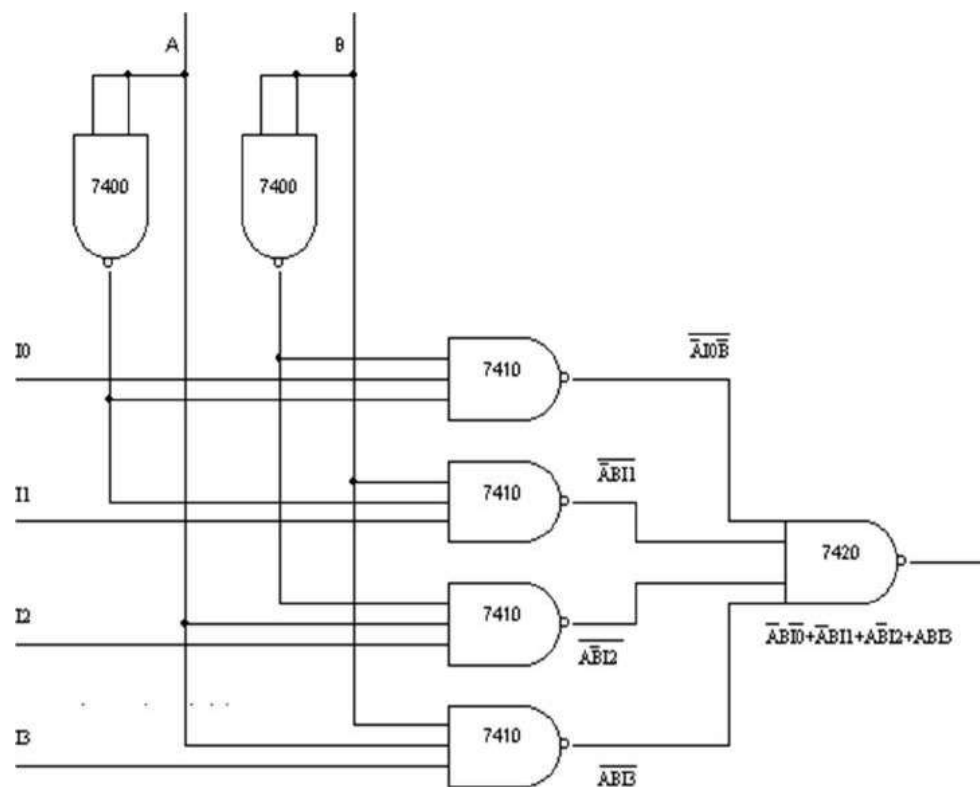
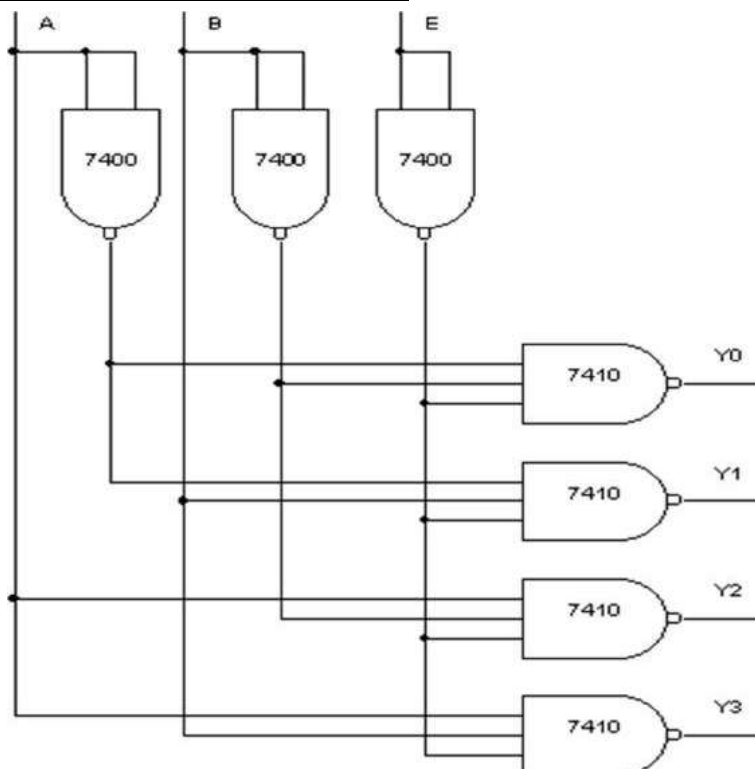
- Repeat the experiment to verify

Channel B. Full Adder using IC 74139:-

Full subtractor using IC 74139:-Truth Tables:-

Full Adder				
An	Bn	Cn-1	Sn (V)	Cn (V)
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Full Subtractor				
An	Bn	Cn-1	Dn (V)	Bn (V)
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

MUX USING NAND GATES ONLY: -DEMUX USING NAND GATES ONLY: -

Experiment No: _____

DATE: __/__/

MUX AND DEMUX USING NAND GATESAIM: - To verify the truth table of MUX and DEMUX using NAND.APPARATUS REQUIRED: -

IC 7400, IC 7410, IC 7420, etc.

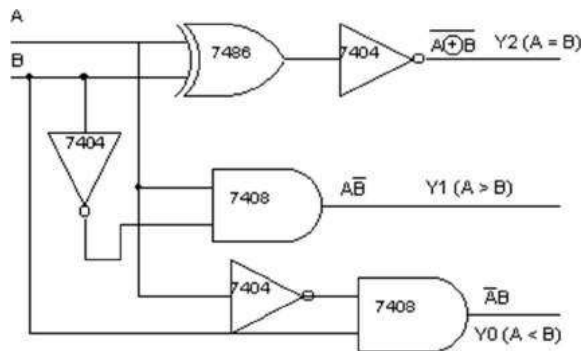
PROCEDURE: -

- 1. Connections are made as shown in the Circuit diagram.**
- 2. Change the values of the inputs as per the truth table and note down the outputs readings using multimeter.**

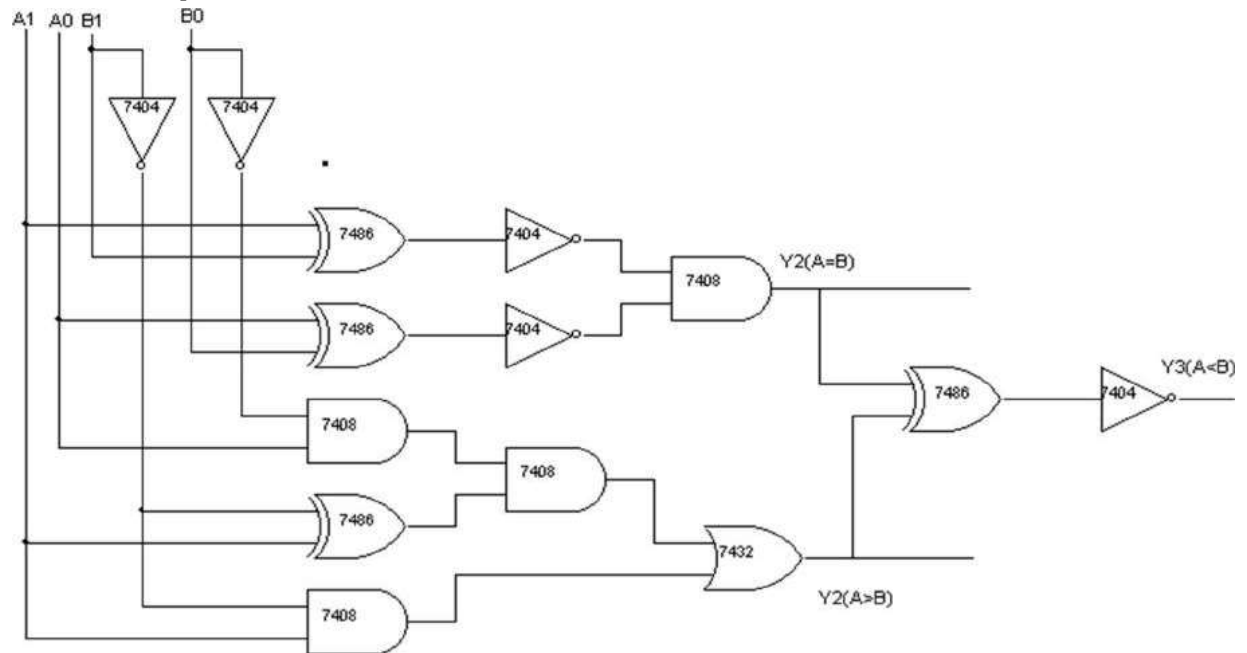
TRUTH TABLES: -

INPUT						OUPUT
A	B	I0	I1	I2	I3	Y (V)
0	0	0	X	X	X	0
0	0	1	X	X	X	1
0	1	X	0	X	X	0
0	1	X	1	X	X	1
1	0	X	X	0	X	0
1	0	X	X	1	X	1
1	1	X	X	X	0	0
1	1	X	X	X	1	1

INPUT			OUPUT			
\bar{E}	A	B	Y0 (V)	Y1 (V)	Y2 (V)	Y3 (V)
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

One Bit Comparator: -

A	B	Y1 (A>B)	Y2 (A = B)	Y3 (A < B)
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

Two Bit Comparator: -Two-Bit Comparator: -

A1	A0	B1	B0	Y1 (A > B)	Y2 (A = B)	Y3 (A < B)
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

Experiment No:

Date: __/__/__

COMPARATORS

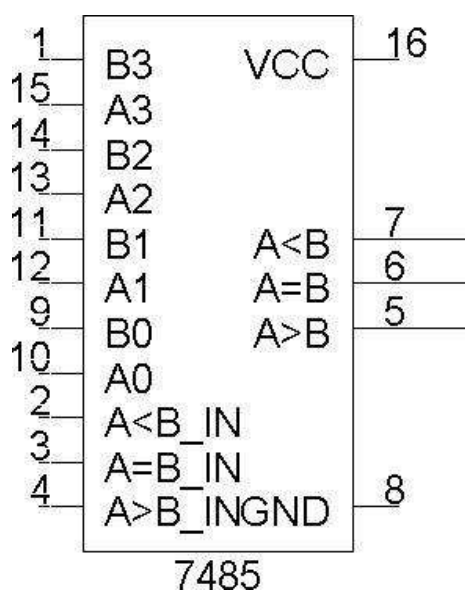
Aim: - To verify the truth table of one bit and two bit comparators using logic gates.

Apparatus Required: -

IC 7486, IC 7404, IC 7408, et c.

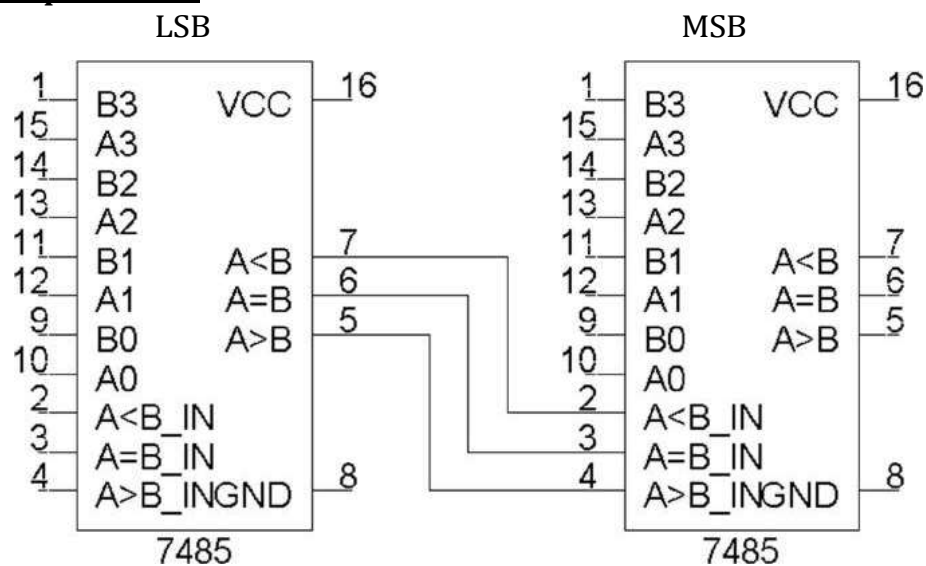
Procedure: -

1. **Verify the gates.**
2. **Make the connections as per the circuit diagram.**
3. **Switch on Vcc.**
4. **Applying i/p and Check for the outputs.**
5. **The voltmeter readings of outputs are taken and tabulated in tabular column.**
6. **The o/p are verified.**

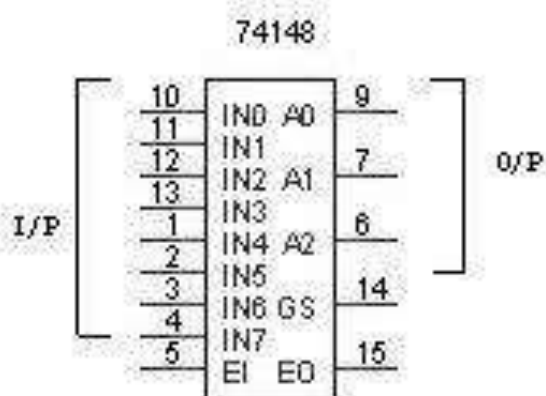
2- **bit Comparator**

Tabular Coloumn For 8-Bit Comparator: -

A ₃ B ₃	A ₂ B ₂	A ₁ B ₁	A ₀ B ₀	A>B	A=B	A<B	A>B	A=B	A<B
A ₃ >B ₃	X	X	X	X	X	X			
A ₃ <B ₃	X	X	X	X	X	X			
A ₃ =B ₃	A ₂ >B ₂	X	X	X	X	X			
A ₃ =B ₃	A ₂ <B ₂	X	X	X	X	X			
A ₃ =B ₃	A ₂ =B ₂	A ₁ >B ₁	X	X	X	X			
A ₃ =B ₃	A ₂ =B ₂	A ₁ <B ₁	X	X	X	X			
A ₃ =B ₃	A ₂ =B ₂	A ₁ =B ₁	A ₀ >B ₀	X	X	X			
A ₃ =B ₃	A ₂ =B ₂	A ₁ =B ₁	A ₀ <B ₀	X	X	X			
A ₃ =B ₃	A ₂ =B ₂	A ₁ =B ₁	A ₀ =B ₀	1	0	0			
A ₃ =B ₃	A ₂ =B ₂	A ₁ =B ₁	A ₀ =B ₀	0	1	0			
A ₃ =B ₃	A ₂ =B ₂	A ₁ =B ₁	A ₀ =B ₀	0	0	1			

8-Bit Comparator: -Exercise:-

- Write the truth table for 8-bit comparator and verify the same for the above circuit.

PIN DETAILS:-TRUTH TABLE:-

E _n	A	B	C	D	E	F	G	H	Q ₂ (V)		Q ₁ (V)		Q ₀ (V)		E _s (V)		E _o (V)	
1	X	X	X	X	X	X	X	X	1		1		1		1		1	
0	0	1	1	1	1	1	1	1	1		1		1		0		1	
0	X	0	1	1	1	1	1	1	1		1		0		0		1	
0	0	X	0	1	1	1	1	1	1		0		1		0		1	
0	0	0	X	0	1	1	1	1	1		0		0		0		1	
0	0	0	0	X	0	1	1	1	0		1		1		0		1	
0	0	0	0	0	X	0	1	1	0		1		0		0		1	
0	0	0	0	0	0	X	0	1	0		0		1		0		1	
0	0	0	0	0	0	0	0	0	0		0		0		0		1	
0	1	1	1	1	1	1	1	1	1		1		1		1		0	

Experiment No:

DATE:___/___/___

ENCODER & DECODER

AIM:-To convert a given octal input to the binary output and to study the LED display using 7447 7-segment decoder/ driver.

APPARATUS REQUIRED: -

IC 74148, IC 7447, 7-segment display, etc.

PROCEDURE: - (Encoder)

- 1 **Connections are made as per circuit diagram.**
- 2 **The octal inputs are given at the corresponding pins.**
- 3 **The outputs are verified at the corresponding output pins.**
















PROCEDURE: - (Decoder)

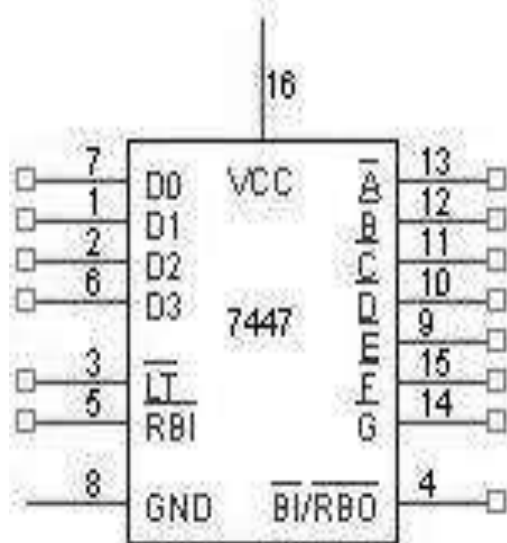
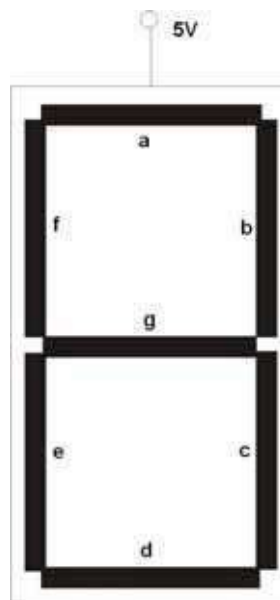
- 1 **Connections are made as per the circuit diagram.**
- 2 **Connect the pins of IC 7447 to the respective pins of the LED display board.**
- 3 **Give different combinations of the inputs and observe the decimal numbers displayed on the board.**

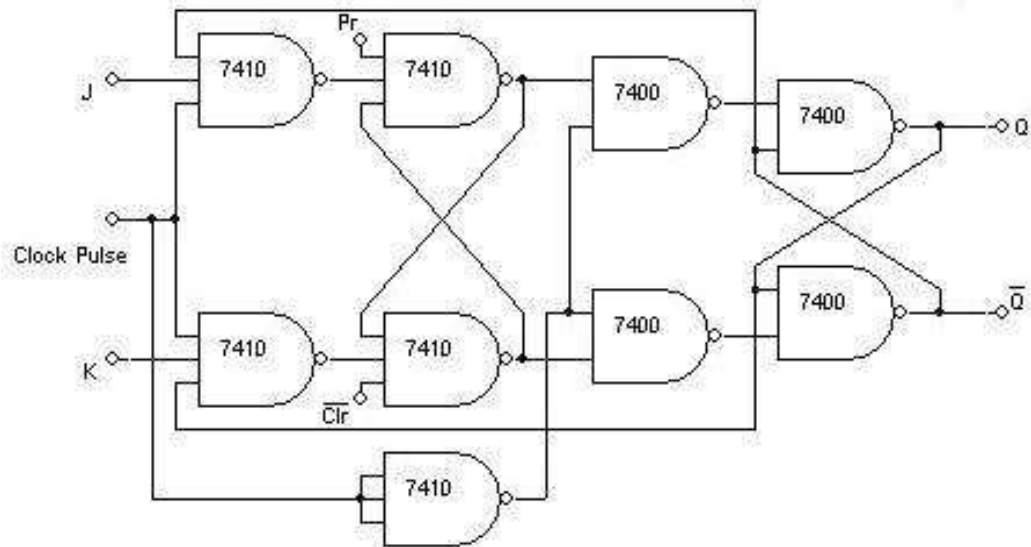
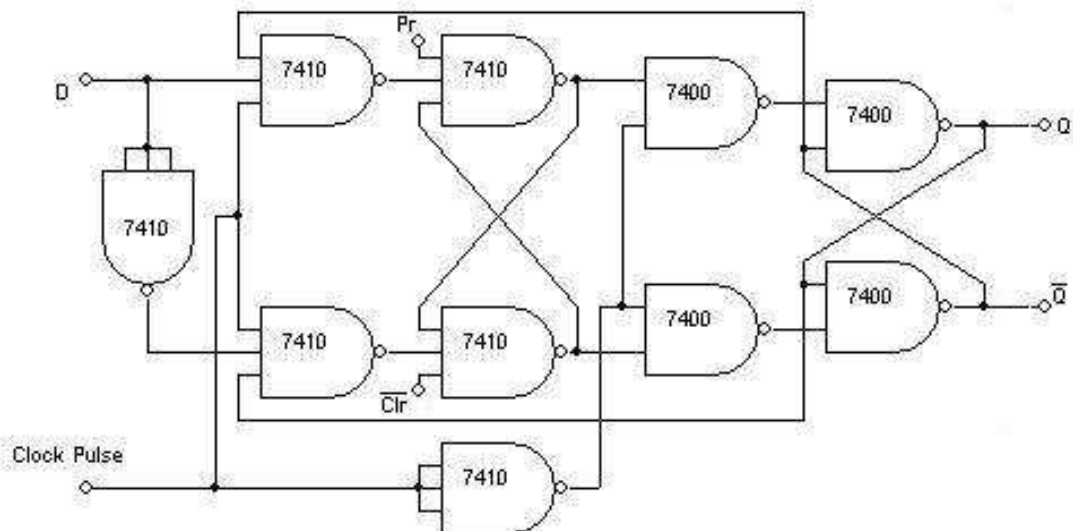
RESULT: -

The given octal numbers are converted into binary numbers. The given data is displayed using 7-segment LED decoder.

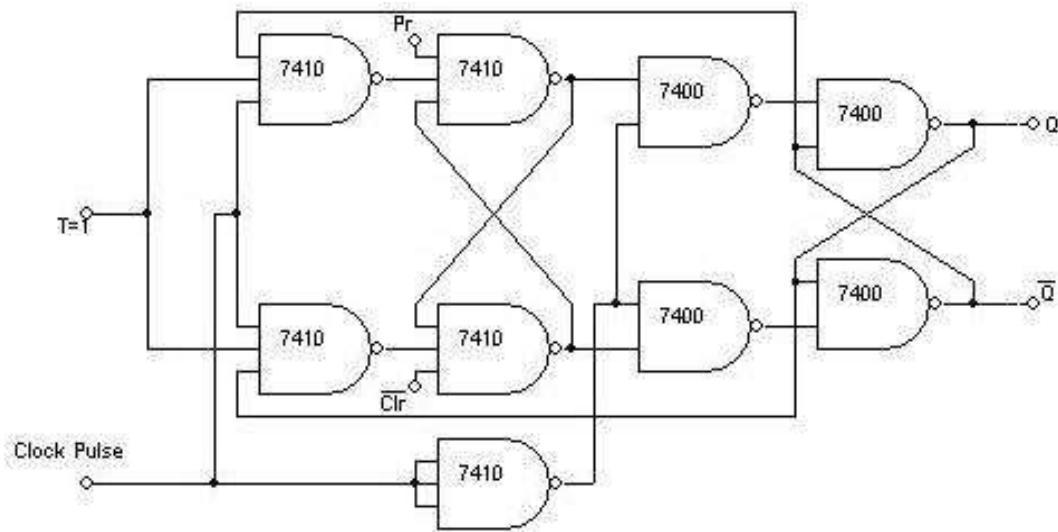
TABULAR COLUMN:-

Q4	Q3	Q2	Q1	O/P	Display	Glowing LEDs
0	0	0	0	0		a,b,c,d,e,f
0	0	0	1	1		b,c
0	0	1	0	2		a,b,d,e,g
0	0	1	1	3		a,b,c,d,g
0	1	0	0	4		b,c,f,g
0	1	0	1	5		a,c,d,f,g
0	1	1	0	6		a.c.d.e.f.g
0	1	1	1	7		a.b.c
1	0	0	0	8		a,b,c,d,e,f,g
1	0	0	1	9		a,b,c,d,f,g
1	0	1	0	10		d,e,g
1	0	1	1	11		c,d,g
1	1	0	0	12		c,d,e
1	1	0	1	13		a,g,d
1	1	1	0	14		d,e,f,g
1	1	1	1	15		blank

PIN DETAILS:-DISPLAY:-Conclusion:-

Circuit Diagram: - (Master Slave JK Flip-Flop)D Flip-Flop:-

T-Flip Flop



Experiment No: _____

Date: __/__/

FLIP-FLOP





Aim:- Truth table verification of Flip-Flops : (i) JK Master Slave
 (ii) **D- Type**
 (iii) **T- Type.**

Apparatus Required: -
 IC 7410, IC 7400, etc.



Procedure: -

- 1 **Connections are made as per circuit diagram.**
- 2 **The truth table is verified for various combinations of inputs.**



Truth Table:- (Master Slave JK Flip-Flop)

Preset	Clear	J	K	Clock	Q _{n+1}	$\overline{Q_n}$	
0	1	X	X	X	1	0	Set
1	0	X	X	X	0	1	Reset
1	1	0	0		Q _n	$\overline{Q_n}$	No Change
1	1	0	1		0	1	Reset
1	1	1	0		1	0	Set
1	1	1	1		$\overline{Q_n}$	Q _n	Toggle

D Flip-Flop:-

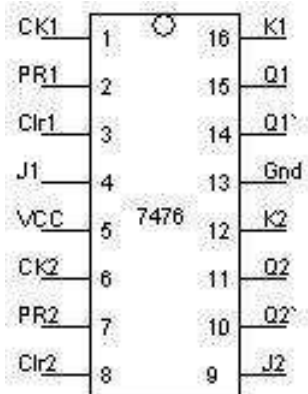
Preset	Clear	D	Clock	Q _{n+1}	$\overline{Q_n}$
1	1	0		0	1
1	1	1		1	0

T Flip-Flop:-

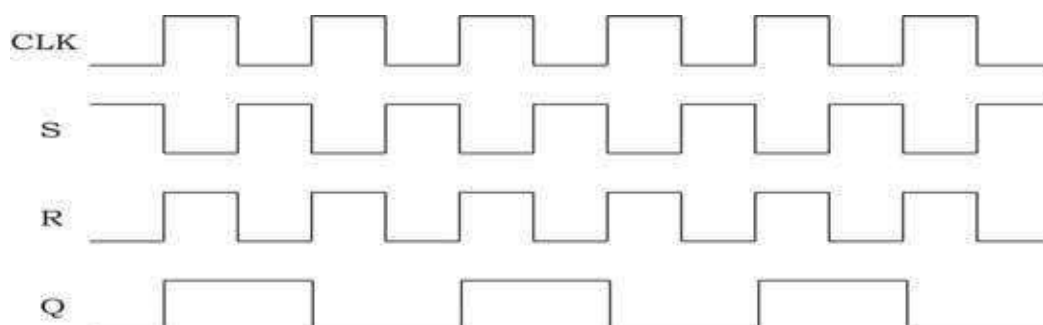
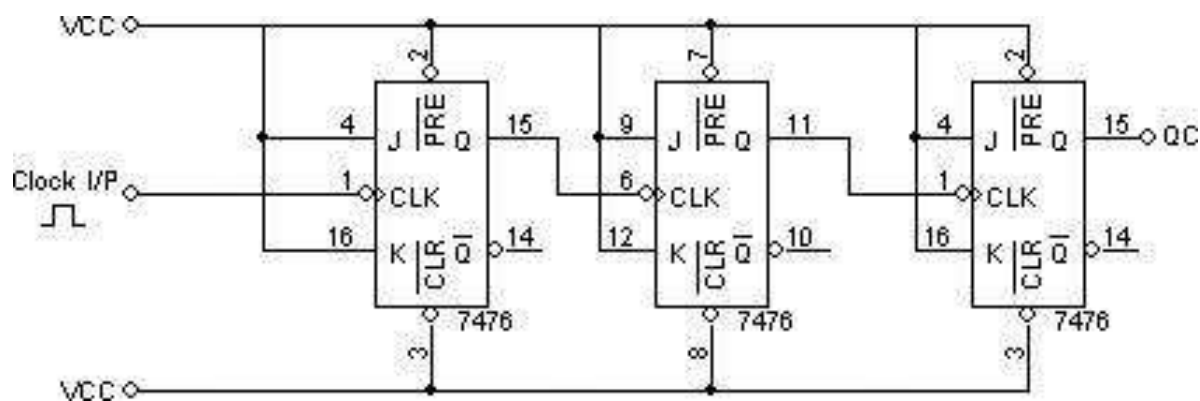
Preset	Clear	T	Clock	Q _{n+1}	$\overline{Q_n}$
1	1	0		Q _n	$\overline{Q_n}$
1	1	1		$\overline{Q_n}$	Q _n

Exercise:-

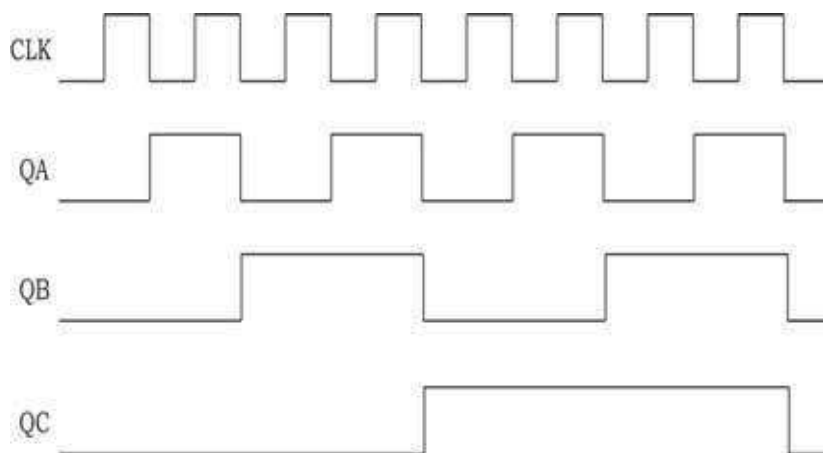
- Write the timing diagrams for all the above Flip-Flops

Pin Details: -Truth Table:-

Clock	QC	QB	QA
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Timing Diagram:-Circuit Diagram: - 3-Bit Asynchronous Up Counter

3-bit Asynchronous up counter			
Clock	QC	QB	QA
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0



No: _____

Experiment
Date: __/__/

COUNTERS

Aim:- Realization of 3-bit counters as a sequential circuit and Mod-N counter design (7476, 7490, 74192, 74193).

Apparatus Required: -

IC 7408, IC 7476, IC 7490, IC 74192, IC 74193, IC 7400, IC 7416, IC 7432
etc.

Procedure: -

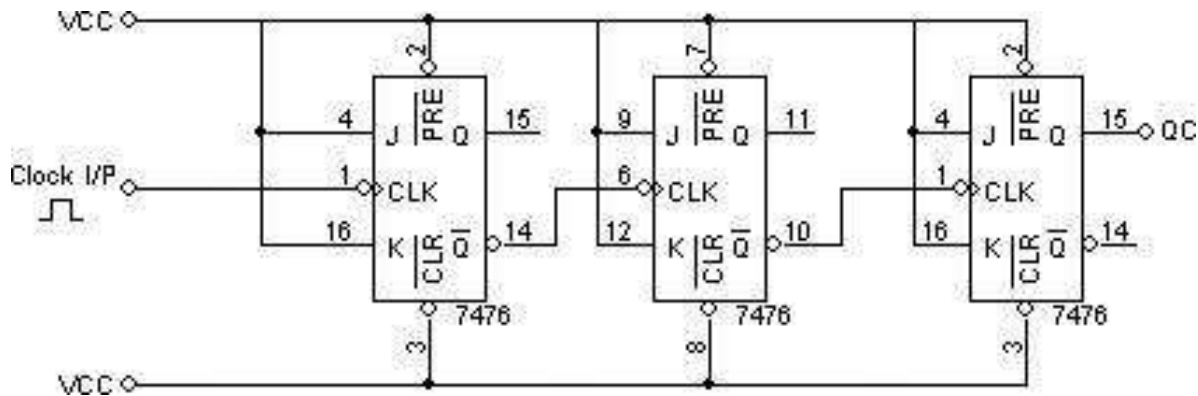
- 1 **Connections are made as per circuit diagram.**
- 2 **Clock pulses are applied one by one at the clock I/P and the O/P is observed at QA, QB & QC for IC 7476.**
- 3 **Truth table is verified.**

Procedure (IC 74192, IC 74193):-

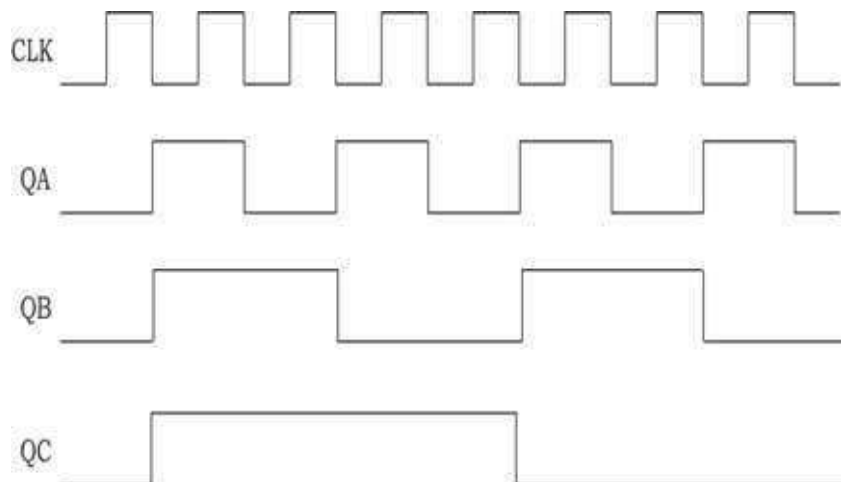
- 1 **Connections are made as per the circuit diagram except the connection from output of NAND gate to the load input.**
- 2 **The data (0011) = 3 is made available at the data i/ps A, B, C & D respectively.**
- 3 **The load pin made low so that the data 0011 appears at QD, QC, QB & QA respectively.**
- 4 **Now connect the output of the NAND gate to the load input.**
- 5 **Clock pulses are applied to “count up” pin and the truth table is verified.**
- 6 **Now apply (1100) = 12 for 12 to 5 counter and remaining is same as for 3 to 8 counter.**

7. The pin diagram of IC 74192 is same as that of 74193. 74192 can be configured to count between 0 and 9 in either direction. The starting value can be any number between 0 and 9.

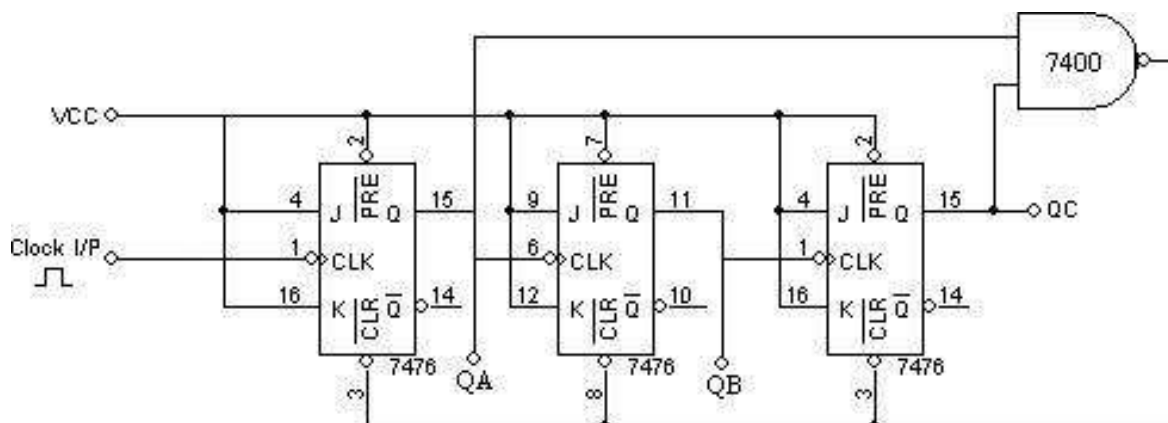
Circuit Diagram: - 3-Bit Asynchronous Down Counter



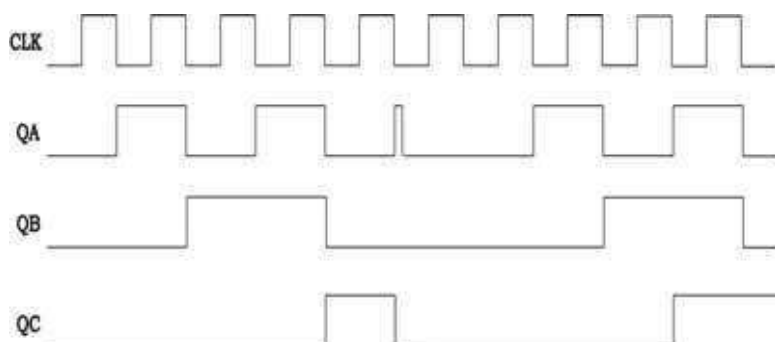
3-bit Asynchronous down counter			
Clock	QC	QB	QA
0	1	1	1
1	1	1	0
2	1	0	1
3	1	0	0
4	0	1	1
5	0	1	0
6	0	0	1
7	0	0	0
8	1	1	1
9	1	1	0



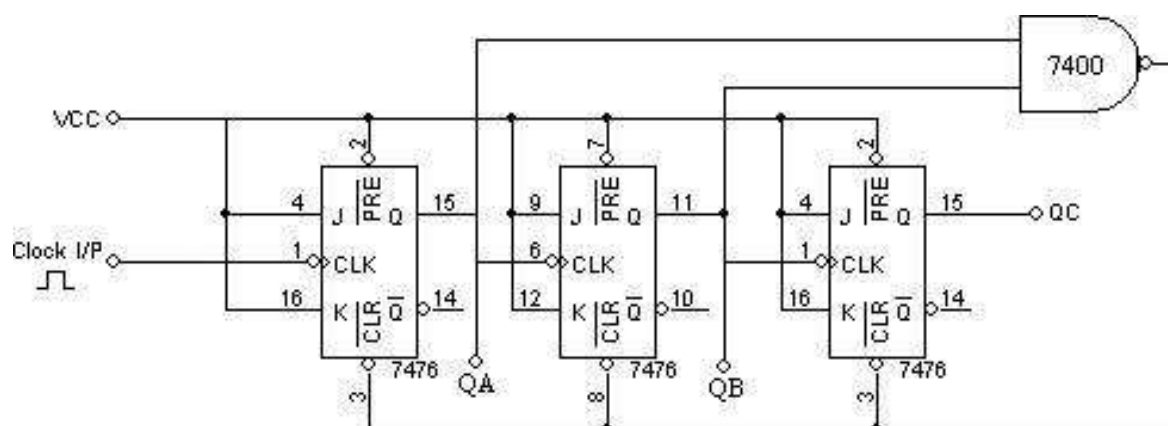
Mod 5 Asynchronous Counter: -



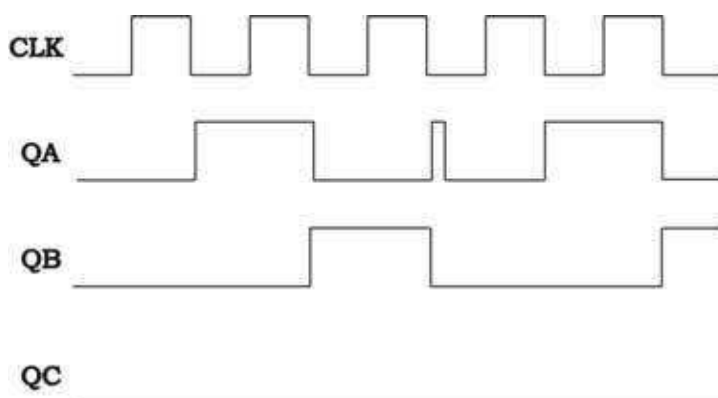
MOD 5 Asynchronous counter			
Clock	QC	QB	QA
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	0	0	0



MOD 3 Asynchronous Counter:-

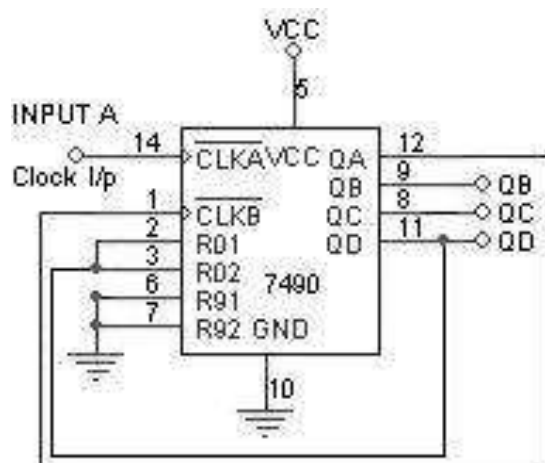


Mod 3 Asynchronous counter			
Clock	QC	QB	QA
0	0	0	0
1	0	0	1
2	0	1	0
3	0	0	0
4	0	0	1
5	0	1	0



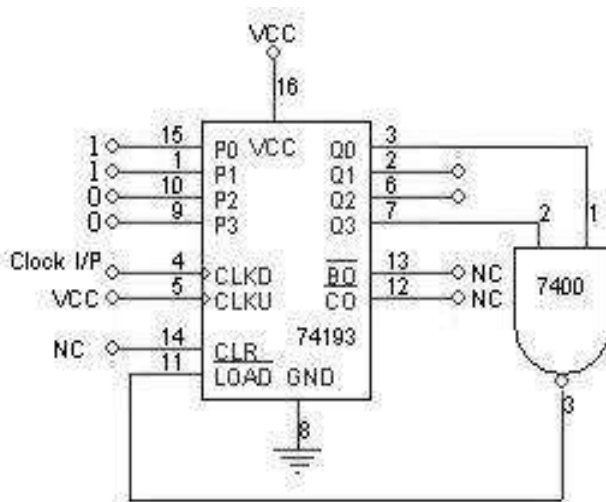
The circuit diagram shows three 7476 J-K flip-flops and a 7408 NAND gate. The clock input (CLK) is connected to the clock input of the first flip-flop (7476). The output of the first flip-flop (QA) is connected to the clock input of the second flip-flop (7476). The output of the second flip-flop (QB) is connected to the clock input of the third flip-flop (7476). The output of the third flip-flop (QC) is connected to the clock input of the first flip-flop (7476). The 7408 NAND gate is connected to the outputs of the first and second flip-flops (QA and QB). The timing diagram shows the CLK signal and the resulting waveforms for QA, QB, and QC, which are square waves with increasing periods.

Clock	QD	QC	QB	QA
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	0	0	0	0

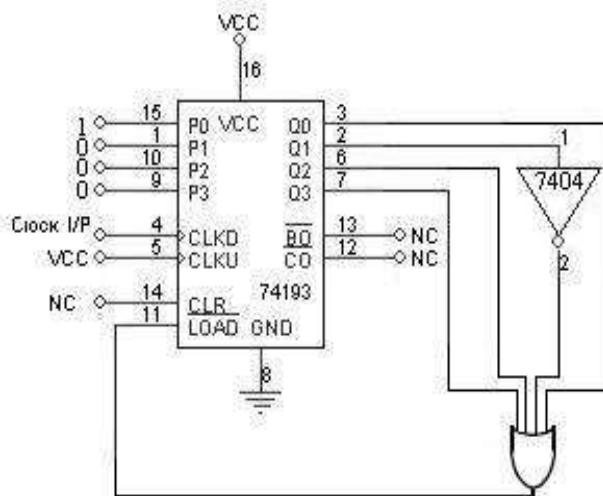


Clock	QD	QC	QB	QA
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	0	0	0	0
9	0	0	0	1

Circuit Diagram (IC 74193) To Count from 3 to 8:-



Clock	QD	QC	QB	QA	Count in Decimal
0	0	0	1	1	3
1	0	1	0	0	4
2	0	1	0	1	5
3	0	1	1	0	6
4	0	1	1	1	7
5	1	0	0	0	8
6	0	0	1	1	3
7	repeats				4

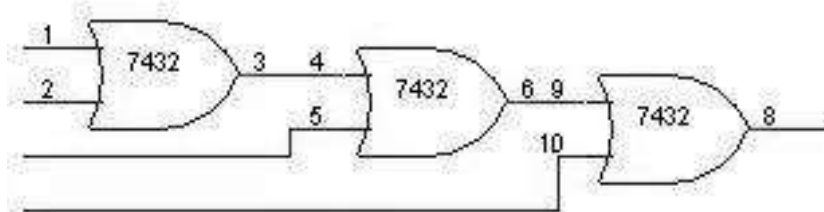
Circuit Diagram (IC 74193) To Count from 8 to 3:-

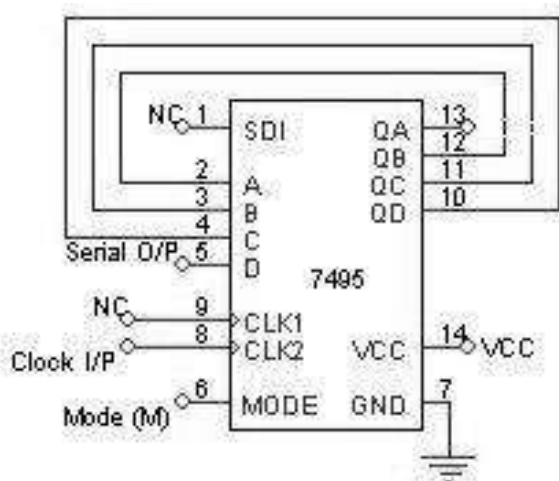
Clock	QD	QC	QB	QA	Count in Decimal
0	0	1	0	1	5
1	0	1	1	0	6
2	0	1	1	1	7
3	1	0	0	0	8
4	1	0	0	1	9
5	1	0	1	0	10
6	1	0	1	1	11
7	1	1	0	0	12
8	0	1	0	1	5
9	repeats				6

Function Table for 7490:-

Clock	R1	R2	S1	S2	QD	QC	QB	QA	
X	H	H	L	X	L	L	L	L	RESET
X	H	H	X	L	L	L	L	L	RESET
X	X	X	H	H	H	L	L	H	SET TO 9
	X	L	X	L	COUNT				
	L	X	L	X	COUNT				
	L	X	X	L	COUNT				
	X	L	L	X	COUNT				

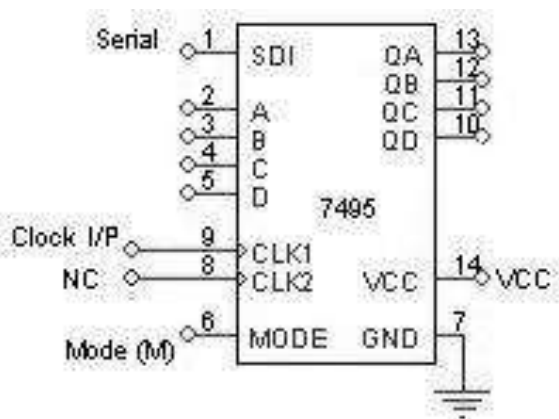
4 I/P OR Gate can be realized as follows:-

Circuit Diagram: - Shift Left



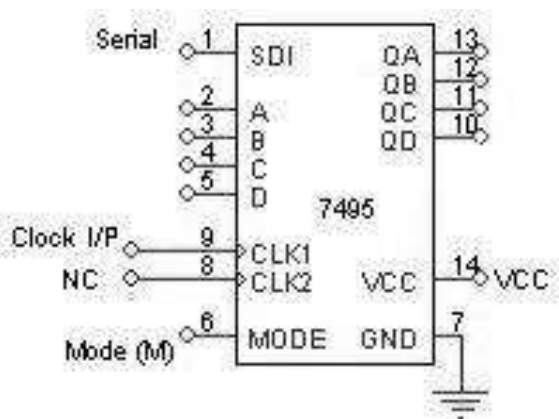
Clock	Serial i/p	QA	QB	QC	QD
1	1	X	X	X	1
2	0	X	X	1	0
3	1	X	1	0	1
4	1	1	0	1	1

SIPO (Right Shift):-



Clock	Serial i/p	QA	QB	QC	QD
1	0	0	X	X	X
2	1	1	0	X	X
3	1	1	1	0	X
4	1	1	1	1	0

SISO:-



Clock	Serial i/p	QA	QB	QC	QD
1	d0=0	0	X	X	X
2	d1=1	1	0	X	X
3	d2=1	1	1	0	X
4	d3=1	1	1	1	0=d0
5	X	X	1	1	1=d1
6	X	X	X	1	1=d2
7	X	X	X	X	1=d3

SHIFT REGISTERS

Aim: - Realization of 3-bit counters as a sequential circuit and Mod-N counter design (7476, 7490, 74192, 74193).

Apparatus Required: -

IC 7495, etc.

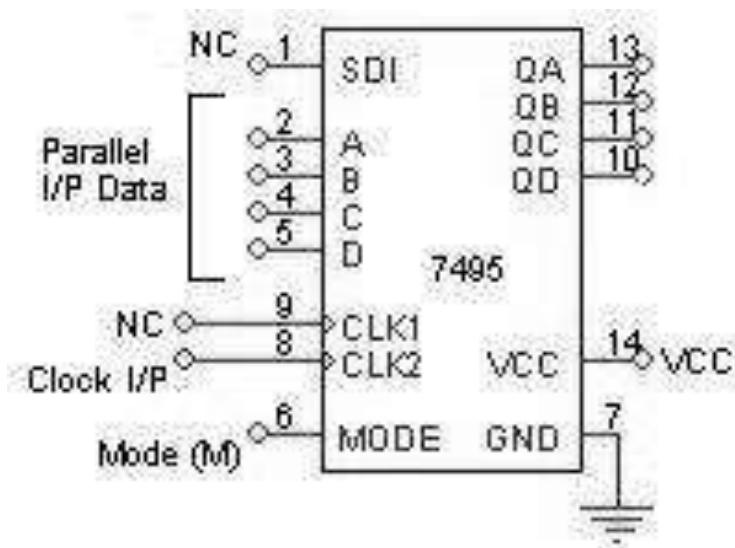
Procedure: -

Serial In Parallel Out:-

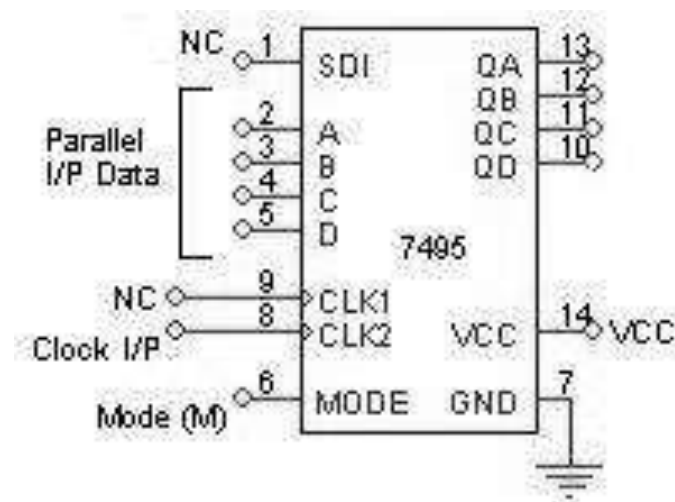
- 1 **Connections are made as per circuit diagram.**
- 2 **Apply the data at serial i/p**
- 3 **Apply one clock pulse at clock 1 (Right Shift) observe this data at QA.**
- 4 **Apply the next data at serial i/p.**
- 5 **Apply one clock pulse at clock 2, observe that the data on QA will shift to QB and the new data applied will appear at QA.**
- 6 **Repeat steps 2 and 3 till all the 4 bits data are entered one by one into the shift register.**

Serial In Serial Out:-

- 1 **Connections are made as per circuit diagram.**
- 2 **Load the shift register with 4 bits of data one by one serially.**
- 3 **At the end of 4th clock pulse the first data 'd0' appears at QD.**
- 4 **Apply another clock pulse; the second data 'd1' appears at QD.**
- 5 **Apply another clock pulse; the third data appears at QD.**
- 6 **Application of next clock pulse will enable the 4th data 'd3' to appear at QD. Thus the data applied serially at the input comes out serially at QD**

PISO:-

Mode	Clock	Parallel i/p				Parallel o/p			
		A	B	C	D	QA	QB	QC	QD
1	1	1	0	1	1	1	0	1	1
0	2	X	X	X	X	X	1	0	1
0	3	X	X	X	X	X	X	1	0
0	4	X	X	X	X	X	X	X	1

PIPO:-

Clock	Parallel i/p				Parallel o/p			
	A	B	C	D	QA	QB	QC	QD
1	1	0	1	1	1	0	1	1

Parallel In Parallel Out:-

- 1 Connections are made as per circuit diagram.
- 2 Apply the 4 bit data at A, B, C and D.
- 3 Apply one clock pulse at Clock 2 (Note: Mode control M=1).
- 4 The 4 bit data at A, B, C and D appears at QA, QB, QC and QD respectively.

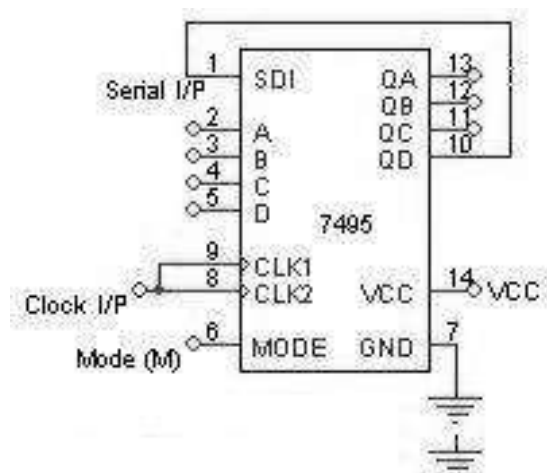
Parallel In Serial Out:-

- 1 Connections are made as per circuit diagram.
- 2 Apply the desired 4 bit data at A, B, C and D.
- 3 Keeping the mode control M=1 apply one clock pulse. The data applied at A, B, C and D will appear at QA, QB, QC and QD respectively.
- 4 Now mode control M=0. Apply clock pulses one by one and observe the data coming out serially at QD.

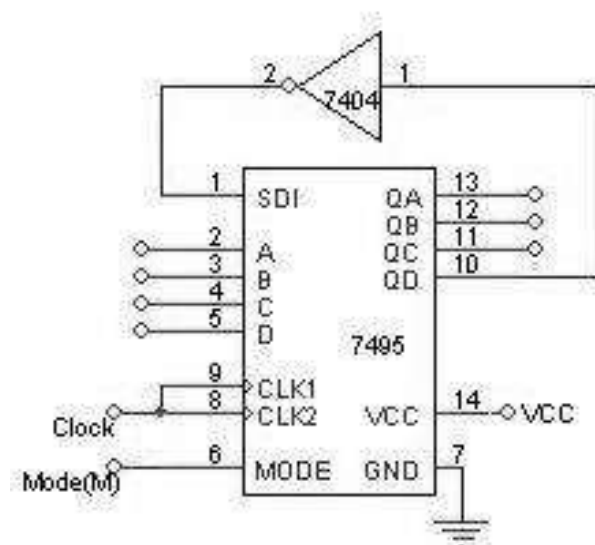
Left Shift:-

- 1 Connections are made as per circuit diagram.
- 2 Apply the first data at D and apply one clock pulse. This data appears at QD.
- 3 Now the second data is made available at D and one clock pulse applied. The data appears at QD to QC and the new data appears at QD.
- 4 Step 3 is repeated until all the 4 bits are entered one by one.
- 5 At the end 4th clock pulse, the 4 bits are available at QA, QB, QC

and QD. Conclusion: -

Circuit Diagram: - Ring Counter

Mode	Clock	QA	QB	QC	QD
1	1	1	0	0	0
0	2	0	1	0	0
0	3	0	0	1	0
0	4	0	0	0	1
0	5	1	0	0	0
0	6	repeats			

Johnson Counter:-

Mode	Clock	QA	QB	QC	QD
1	1	1	0	0	0
0	2	1	1	0	0
0	3	1	1	1	0
0	4	1	1	1	1
0	5	0	1	1	1
0	6	0	0	1	1
0	7	0	0	0	1
0	8	0	0	0	0
0	9	1	0	0	0
0	10	repeats			

Experiment No:

Date: __/__/

JOHNSON COUNTERS / RING COUNTER

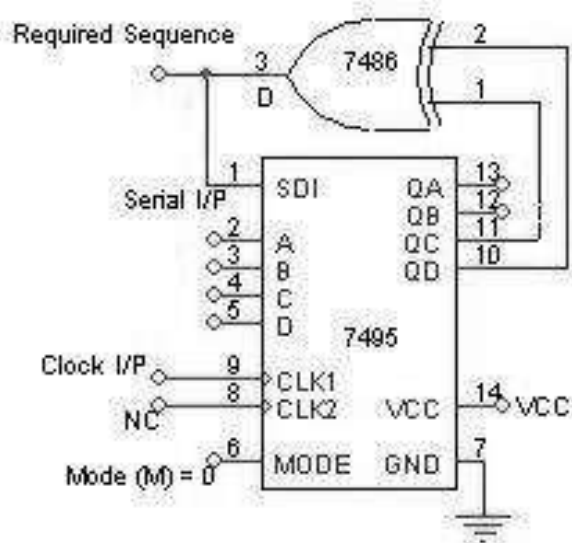
Aim:- Design and testing of Ring counter/ Johnson counter.

Apparatus Required: -

IC 7495, IC 7404, etc.

Procedure: -

- 1 Connections are made as per the circuit diagram.
- 2 Apply the data 1000 at A, B, C and D respectively.
- 3 Keeping the mode $M = 1$, apply one clock pulse.
- 4 Now the mode M is made 0 and clock pulses are applied one by one, and the truth table is verified.
- 5 Above procedure is repeated for Johnson counter also.

Circuit Diagram: - Sequence GeneratorTruth Table:-

Map Value	Clock	QA	QB	QC	QD	o/p D
15	1	1	1	1	1	0
7	2	0	1	1	1	0
3	3	0	0	1	1	0
1	4	0	0	0	1	1
8	5	1	0	0	0	0
4	6	0	1	0	0	0
2	7	0	0	1	0	1
9	8	1	0	0	1	1
12	9	1	1	0	0	0
6	10	0	1	1	0	1
11	11	1	0	1	1	0
5	12	0	1	0	1	1
10	13	1	0	1	0	1
13	14	1	1	0	1	1
14	15	1	1	1	0	1

Karnaugh Map for D:-

		QA QB			
QC QD		00	01	11	10
	00	0	0	0	0
	01	1	1	1	1
	11	0	0	0	0
	10	1	1	1	1

Experiment No:

Date: __/__/

SEQUENCE GENERATOR

Aim:- Design of Sequence Generator.

Apparatus Required: -

IC 7495, IC 7486, etc.

Design:-

To generate a sequence of length S it is necessary to use at least N number of Flip-Flops, which satisfies the condition $S \leq 2^N - 1$.

The given sequence length $S = 15$.

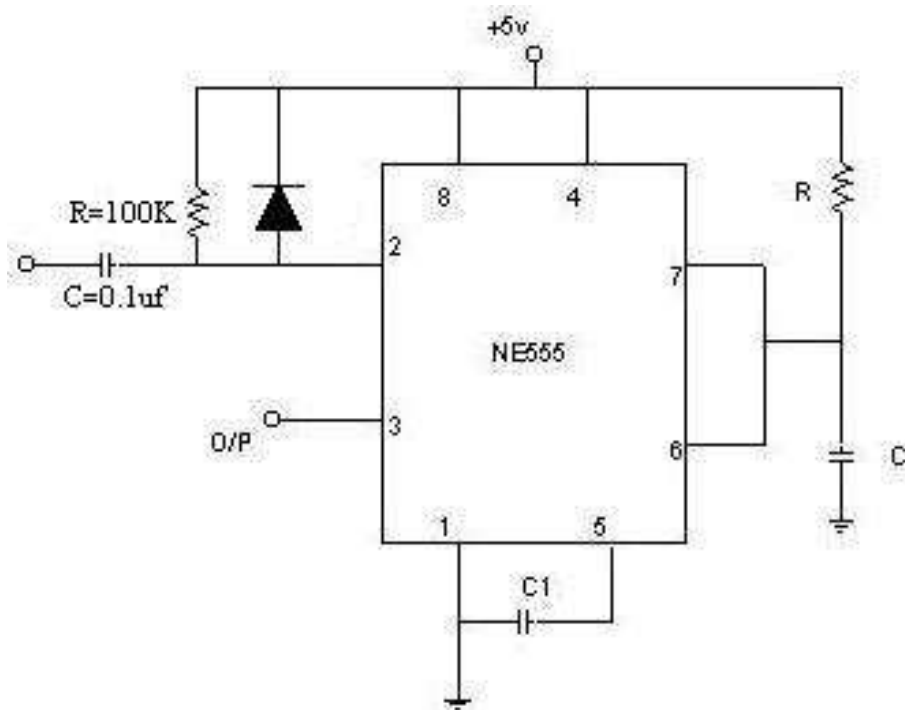
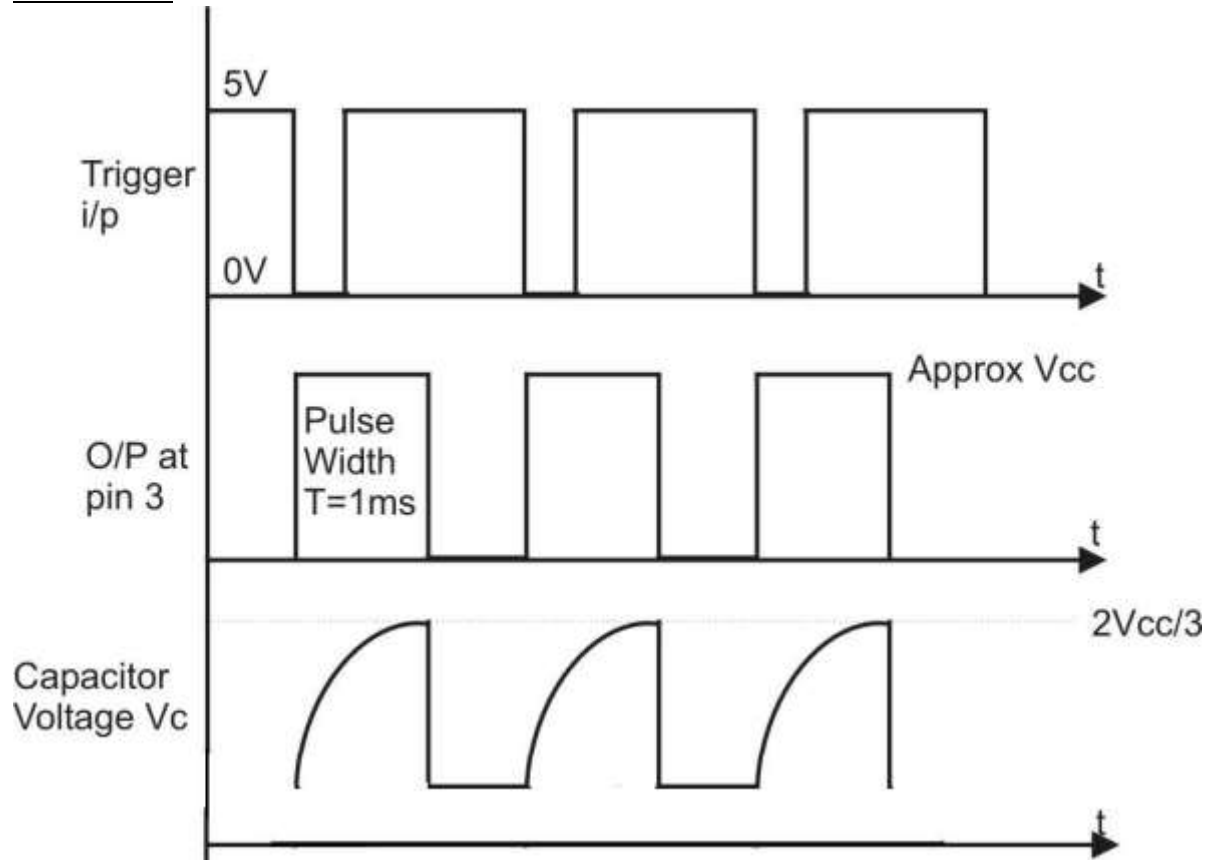
Therefore $N = 4$.

Note: - There is no guarantee that the given sequence can be generated by 4 f/fs. If the sequence is not realizable by 4 f/fs then 5 f/fs must be used and so on.

Procedure: -

- 1. Connections are made as per the circuit diagram.**
- 2. Clock pulses are applied one by one and truth table is verified.**

Conclusion:-

Circuit Diagram: - Monostable MultivibratorWaveform:-

XML PROGRAMMING LABORATORY MAUAL

(Course Code: BCASEC02P)

XML PROGRAMMING

Introduction:

XML is a type of markup language used to store the data. It is very much popular for the transfer of data. It is a case-sensitive language. With the help of XML, we can define the markup elements and customize the markup language. Element is known as the basic element of XML. The XML file should be stored with the extension of **.xml**.

- ❖ XML stands for **EX**tensible **M**arkup **L**anguage.
- ❖ XML is a markup language like HTML.
- ❖ XML is designed to store and transport data.
- ❖ XML is designed to be self-descriptive.
- ❖ It became a W3C Recommendation on February 10, 1998.
- ❖ XML tags are not predefined. You must define your own tags.
- ❖ It is platform independent and language independent.
- ❖ XML is designed in such a way that it can carry the data, but it cannot display the data.

Features of XML:

XML is widely used in the era of web development. It is also used to simplify data storage and data sharing.

The main features or advantages of XML are given below.

1. XML separates data from HTML:

If you need to display dynamic data in your HTML document, it will take a lot of work to edit the HTML each time the data changes. With XML, data can be stored in separate XML files. This way you can focus on using HTML/CSS for display and layout, and be sure that changes in the underlying data will not require any changes to the HTML. With a few lines of JavaScript code, you can read an external XML file and update the data content of your web page.

2. XML simplifies data sharing:

In the real world, computer systems and databases contain data in incompatible formats. XML data is stored in plain text format. This provides a software- and hardware-independent way of storing data. This makes it much easier to create data that can be shared by different applications.

3. XML simplifies data transport:

One of the most time-consuming challenges for developers is to exchange data between incompatible systems over the Internet. Exchanging data as XML greatly reduces this complexity, since the data can be read by different incompatible applications.

4. XML increases data availability:

Different applications can access your data, not only in HTML pages, but also from XML data sources. With XML, your data can be available to all kinds of "reading machines" (Handheld computers, voice machines, news feeds, etc), and make it more available for blind people, or people with other disabilities. XML is designed in such a way that it can carry the data, but it cannot display the data.

5. XML simplifies Platform change:

Upgrading to new systems (hardware or software platforms), is always time consuming. Large amounts of data must be converted and incompatible data is often lost. XML data is stored in text format. This makes it easier to expand or upgrade to new operating systems, new applications, or new browsers, without losing data.

Basic of XML:

Structure of XML:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<root>
```

```
  <child>
```

```
    <subchild>..... </subchild>
```

```
    <subchild>..... </subchild>
```

```
  </child>
```

```
</root>
```

- ❖ XML documents must contain a root element. This element is "the parent" of all other elements.
- ❖ The elements in an XML document form a document tree. The tree starts at the root and branches to the lowest level of the tree.
- ❖ All elements can have sub elements (child elements).
- ❖ The terms parent, child, and sibling are used to describe the relationships between elements. Parent elements have children. Children on the same level are called siblings (brothers or sisters).
- ❖ All elements can have text content and attributes (just like in HTML).

XML Example:

Simple store your & your friend's name –

```
<?xml version="1.0" encoding="UTF-8"?>
<data>
    <friends>
        <name> ABC </name>
        <name> DEF </name>
    </friends>
</data>
```

XML Attributes:

XML elements can have attributes. By the use of attributes, we can add the information about the element.

XML attributes enhance the properties of the elements.

Let us take an example of a book publisher. Here, book is the element and publisher is the attribute –

```
<book publisher="Tata McGraw Hill"></book> Or,
<book publisher='Tata McGraw Hill'></book>
```

Metadata should be stored as attribute and data should be stored as element.

```
<book>
    <book category="computer">
        <author> A & B </author>
</book>
```

In the context of documents, attributes are part of markup, while sub elements are part of the basic document contents.

In the context of data representation, the difference is unclear and may be confusing.

Same information can be represented in two ways:

1st way –

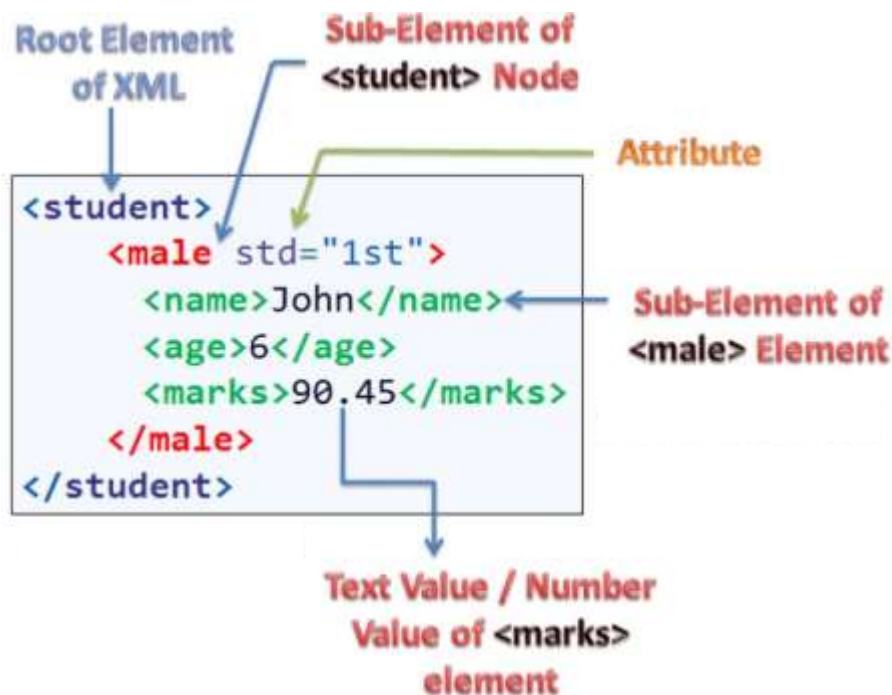
```
<book publisher="Tata McGraw Hill"> </book>
```

2nd way –

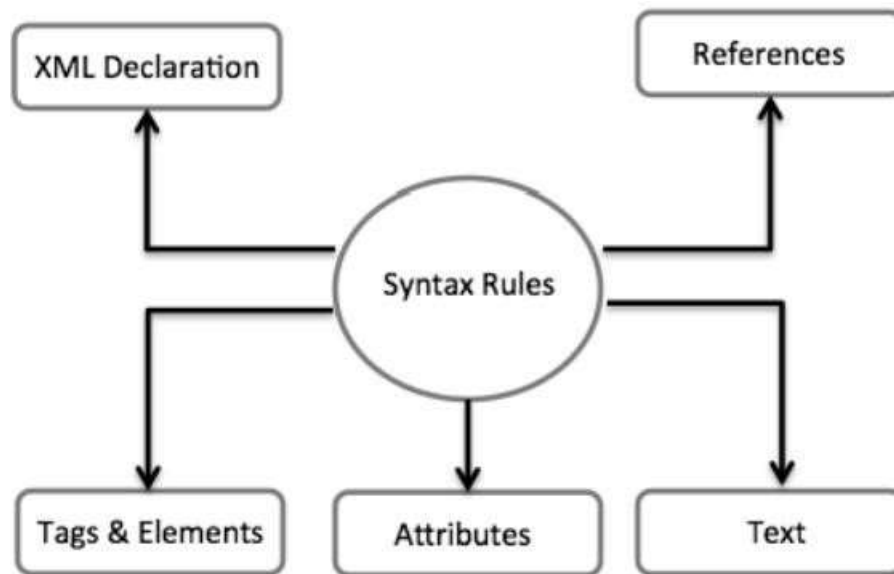
```
<book>
  <publisher> Tata McGraw Hill </publisher>
</book>
```

In the first example publisher is used as an attribute and in the second example publisher is an element.

Both examples provide the same information but it is good practice to avoid attribute in XML and use elements instead of attributes.



XML Syntax Rules:



1. XML Documents Must Have a Root Element

XML documents must contain one root element that is the parent of all other elements:

```

<root>
  <child>
    <subchild>..... </subchild>
  </child>
</root>
  
```

2. The XML Prolog

This line is called the XML prolog:

```
<?xml version="1.0" encoding="UTF-8"?>
```

The XML prolog is optional. If it exists, it must come first in the document.

To avoid errors, you should specify the encoding used, or save your XML files as UTF-8.

UTF-8 is the default character encoding for XML documents.

3. All XML Elements Must Have a Closing Tag

In XML, it is illegal to omit the closing tag. All elements must have a closing tag:

```
<book> This is a book </book>
```

4. XML Tags are Case Sensitive

XML tags are case sensitive. The tag <Letter> is different from the tag <letter>.

Opening and closing tags must be written with the same case:

```
<message>This is correct</message>
```

"Opening and closing tags" are often referred to as "Start and end tags". Use whatever you prefer. It is exactly the same thing.

5. XML Elements Must be Properly Nested

In XML, all elements must be properly nested within each other:

```
<b><i>This text is bold and italic</i></b>
```

In the example above, "Properly nested" simply means that since the <i> element is opened inside the element, it must be closed inside the element.

6. XML Attribute Values Must Always be Quoted

XML elements can have attributes in name/value pairs just like in HTML.

In XML, the attribute values must always be quoted:

```
<note date="12/11/2007">  
  <to>Tove</to>  
  <from>Jani</from>  
</note>
```

7. Entity References

Some characters have a special meaning in XML.

If you place a character like "<" inside an XML element, it will generate an error because the parser interprets it as the start of a new element.

This will generate an XML error:

```
<message>salary < 1000</message>
```

To avoid this error, replace the "<" character with an entity reference:

```
<message>salary &lt; 1000</message>
```

There are 5 pre-defined entity references in XML:

<	<	less than
>	>	greater than
&	&	ampersand
'	'	apostrophe
"	"	quotation mark

Only < and & are strictly illegal in XML, but it is a good habit to replace > with > as well.

8. Comments in XML:

The syntax for writing comments in XML is similar to that of HTML:

```
<!-- This is a comment -->
```

Two dashes in the middle of a comment are not allowed:

```
<!-- This is an invalid -- comment -->
```

9. White-space is Preserved in XML:

XML does not truncate multiple white-spaces (HTML truncates multiple white-spaces to one single white-space):

XML:	Hello Tove
HTML:	Hello Tove

XML vs HTML:

XML (Extensible Markup Language)	HTML (Hypertext Markup Language)
It stores and transports data.	It displays data.
It uses user-defined tags.	It uses predefined tags.
It contains structural data.	It doesn't contain any structural data.
It can distinguish uppercase and lowercase letters (case sensitive).	It can't distinguish uppercase and lowercase letters (case insensitive).
It maintains spacing, tabs, newlines, and any other whitespace formatting.	It doesn't maintain whitespace.
It needs to have an end-tag.	It doesn't need an end-tag.
It needs structure or nesting.	It doesn't need structure.

For Example:

```
<h1>title</h1>
<p>paragraph</p>
<p>paragraph</p>
```

```
<headline>title</headline>
<paragraph>paragraph</paragraph>
<paragraph>paragraph</paragraph>
```

XML Tree Structure:

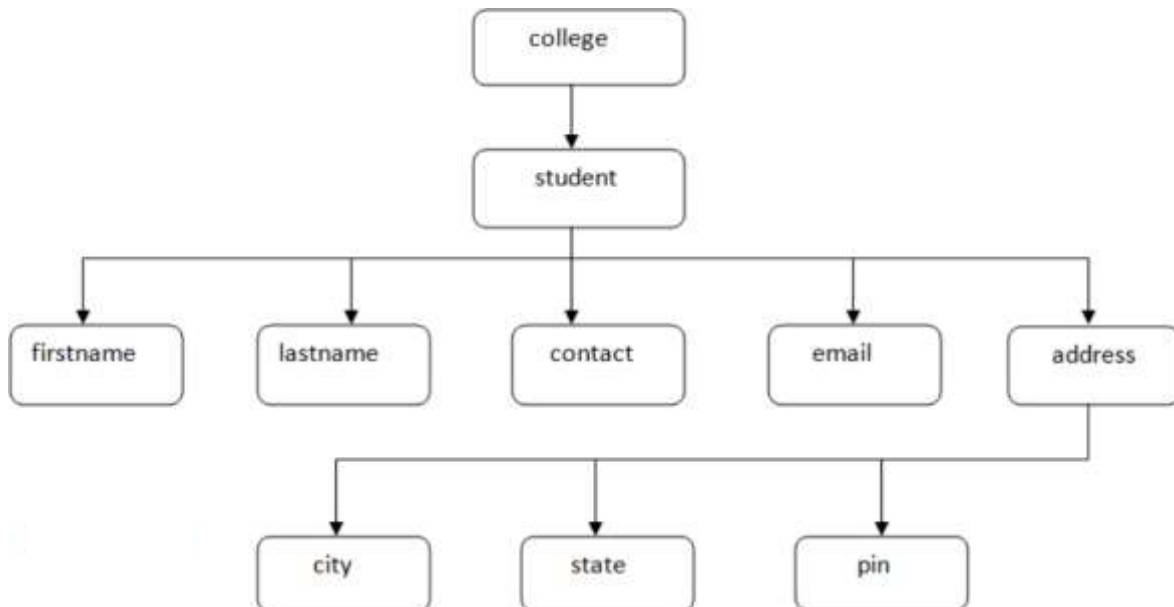
An XML document has a self-descriptive structure. It forms a tree structure which is referred as an XML tree. The tree structure makes easy to describe an XML document.

A tree structure contains root element (as parent), child element and so on. It is very easy to traverse all succeeding branches and sub-branches and leaf nodes starting from the root.

Example:

```
<?xml version="1.0"?>
<college>
  <student>
    <firstname>Tamanna</firstname>
    <lastname>Bhatia</lastname>
    <contact>09990449935</contact>
    <email>tammanabhatia@abc.com</email>
    <address>
      <city>Ghaziabad</city>
      <state>Uttar Pradesh</state>
      <pin>201007</pin>
    </address>
  </student>
</college>
```


Let's see the tree-structure representation of the above example.



In the above example, first line is the XML declaration. It defines the XML version 1.0. Next line shows the root element (college) of the document. Inside that there is one more element (student). Student element contains five branches named <firstname>, <lastname>, <contact>, <Email> and <address>.

<address> branch contains 3 sub-branches named <city>, <state> and <pin>.

XML Validation:

A well-formed XML document can be validated against DTD or Schema.

A well-formed XML document is an XML document with correct syntax. It is very necessary to know about valid XML document before knowing XML validation.

Rules for well-formed XML:

- ❖ It must begin with the XML declaration.
- ❖ It must have one unique root element.
- ❖ All start tags of XML documents must match end tags.
- ❖ XML tags are case sensitive.
- ❖ All elements must be closed.
- ❖ All elements must be properly nested.
- ❖ All attributes' values must be quoted.
- ❖ XML entities must be used for special characters.



XML Namespace:

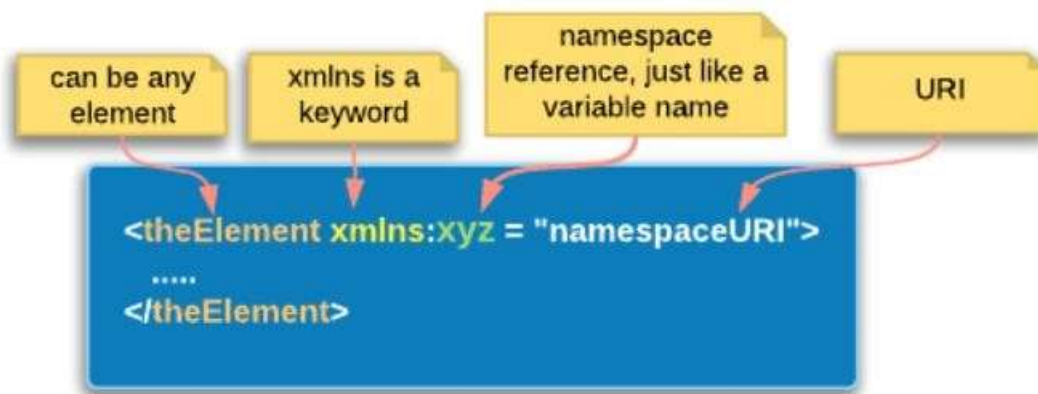
XML Namespace is used to avoid element name conflict in XML document.

An XML namespace is declared using the reserved XML attribute. This attribute name must be started with "xmlns".

Let's see the XML namespace syntax:

```
<element xmlns:name = "URL">
```

Here, namespace starts with keyword "xmlns". The word name is a namespace prefix. The URL is a namespace identifier.



Let's see the example of XML file:

```
<?xml version="1.0" encoding="UTF-8"?>
<cont:contact xmlns:cont="http://kn.com/contact">
  <cont:name>Krishnendu Nanda</cont:name>
  <cont:phone>90645 57423</cont:phone>
</cont:contact>
```

Namespace Prefix: cont

Namespace Identifier: http://kn.com/contact

It specifies that the element name and attribute names with cont prefix belongs to http://kn.com/contact name space.

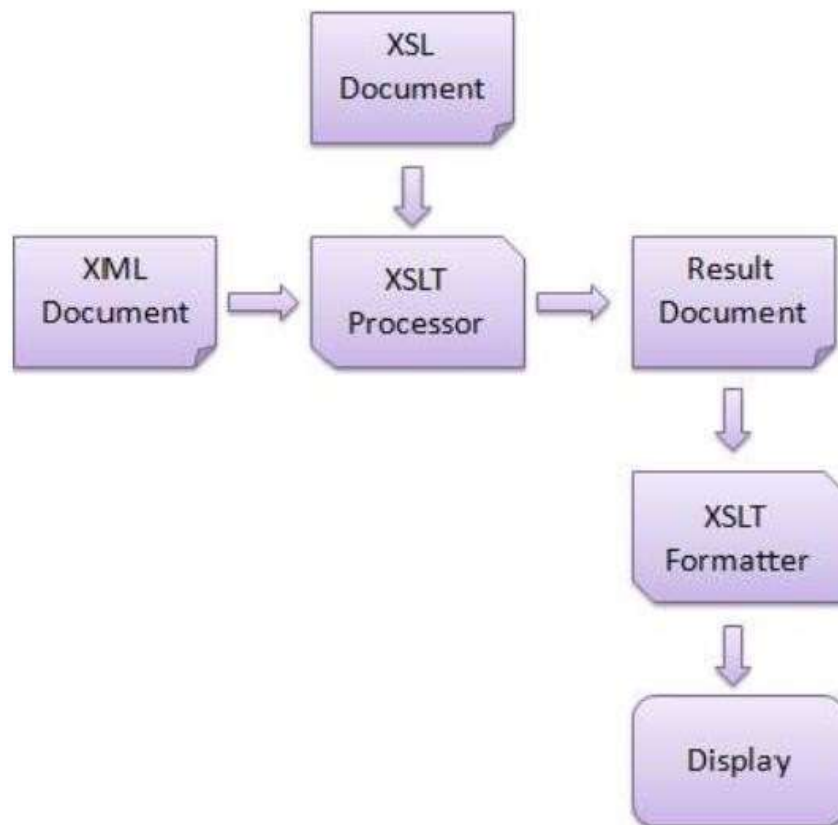
XSLT:

XSLT tutorial provides basic and advanced concepts of XSLT. Our XSLT tutorial is designed for beginners and professionals.

XSL stands for EXtensible Stylesheet Language. XSLT is for Transformation of XML document to other formats.

Our XSLT tutorial includes all topics of Search Engine Optimization such as black and white XSLT techniques, how search engine works, XSLT market research, content research, On Page Optimization techniques, Off Page Optimization techniques, Social Media Optimization, XSLT Tools etc.

XSLT stands for XSL Transformation. It is used to transform XML documents into other formats (like transforming XML into HTML).



Advantages of XSLT:

- ✓ XSLT provides an easy way to merge XML data into presentation because it applies user defined transformations to an XML document and the output can be HTML, XML, or any other structured document.
- ✓ XSLT provides Xpath to locate elements/attribute within an XML document. So it is more convenient way to traverse an XML document rather than a traditional way, by using scripting language.
- ✓ XSLT is template based. So it is more resilient to changes in documents than low level DOM and SAX.
- ✓ By using XML and XSLT, the application UI script will look clean and will be easier to maintain.
- ✓ XSLT templates are based on XPath pattern which is very powerful in terms of performance to process the XML document.
- ✓ XSLT can be used as a validation language as it uses tree-pattern-matching approach.
- ✓ You can change the output simply modifying the transformations in XSL files.

For example:

Suppose,

Create an XML file named **employee.xml**.

Create the XSLT document which satisfies the above requirements. Name it as **employee.xsl** and save it in the same location of employee.xml.

employee.xml

```
<?xml version = "1.0" encoding = "UTF-8"?>
<?xml-stylesheet type = "text/xsl" href = "employee.xsl"?>
```

employee.xsl

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsl:stylesheet version = "1.0" xmlns:xsl =
"http://www.w3.org/1999/XSL/Transform">
```

XSLT Transformations:

1. <xsl:template>
2. <xsl:value-of>
3. <xsl:for-each>
4. <xsl:sort>
5. <xsl:if>
6. <xsl:choose>
7. <xsl:apply-templates>

XPath:

- XPath is a major element in the XSLT standard.
- XPath can be used to navigate through elements and attributes in an XML document.



- XPath stands for XML Path Language
- XPath uses "path like" syntax to identify and navigate nodes in an XML document
- XPath contains over 200 built-in functions
- XPath is a major element in the XSLT standard
- XPath is a W3C recommendation

XPath Path Expressions:

XPath uses path expressions to select nodes or node-sets in an XML document.

These path expressions look very much like the path expressions you use with traditional computer file systems:



XPath Standard Functions:

XPath includes over 200 built-in functions.

There are functions for string values, numeric values, booleans, date and time comparison, node manipulation, sequence manipulation, and much more.

Today XPath expressions can also be used in JavaScript, Java, XML Schema, PHP, Python, C and C++, and lots of other languages.

XPath is a W3C Recommendation:

- ❖ XPath 1.0 became a W3C Recommendation on November 16, 1999.
- ❖ XPath 2.0 became a W3C Recommendation on January 23, 2007.
- ❖ XPath 3.0 became a W3C Recommendation on April 8, 2014.

XPath Nodes Terminology:

Nodes -

In XPath, there are seven kinds of nodes: element, attribute, text, namespace, processing-instruction, comment, and root nodes.

XML documents are treated as trees of nodes. The topmost element of the tree is called the root element.

Look at the following XML document:

```
<?xml version="1.0" encoding="UTF-8"?>

<bookstore>
  <book>
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
</bookstore>
```


Example of nodes in the XML document above:

<bookstore> (root element node)

<author>J K. Rowling</author> (element node) lang="en"

(attribute node)

Relationship of Nodes:**Parent –**

Each element and attribute has one parent.

In the following example; the book element is the parent of the title, author, year, and price:

```
<book>
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
```

Children –

Element nodes may have zero, one or more children.

In the following example; the title, author, year, and price elements are all children of the book element:

```
<book>
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
```

Siblings –

Nodes that have the same parent.

In the following example; the title, author, year, and price elements are all siblings:

```
<book>
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
```

Ancestors –

A node's parent, parent's parent, etc.

In the following example; the ancestors of the title element are the book element and the bookstore element:

```
<bookstore>
  <book>
    <title>Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
</bookstore>
```

Descendants –

A node's children, children's children, etc.

In the following example; descendants of the bookstore element are the book, title, author, year, and price elements:

```
<bookstore>
  <book>
    <title>Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
</bookstore>
```

Selecting Nodes:

XPath uses path expressions to

select nodes in an XML document. The node is selected by following a path or steps. The most useful path expressions are listed below:

Path Expression	Description
<i>nodename</i>	Selects all nodes with the name " <i>nodename</i> "
/	Selects from the root node
//	Selects nodes in the document from the current node that match the selection no matter where they are
.	Selects the current node
..	Selects the parent of the current node
@	Selects attributes

In the table below we have listed some path expressions and the result of the expressions:

Path Expression	Result
bookstore	Selects all nodes with the name "bookstore"
/bookstore	Selects the root element bookstore Note: If the path starts with a slash (/) it always represents an absolute path to an element!
bookstore/book	Selects all book elements that are children of bookstore
//book	Selects all book elements no matter where they are in the document
bookstore//book	Selects all book elements that are descendant of the bookstore element, no matter where they are under the bookstore element
//@lang	Selects all attributes that are named lang

XLink:

- XLink is used to create hyperlinks within XML documents
- Any element in an XML document can behave as a link
- With XLink, the links can be defined outside the linked files
- XLink is a W3C Recommendation

Syntax:

In HTML, the <a> element defines a hyperlink. However, this is not how it works in XML. In XML documents, you can use whatever element names you want - therefore it is impossible for browsers to predict what link elements will be called in XML documents.

Below is a simple example of how to use XLink to create links in an XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
<homepages xmlns:xlink="http://www.w3.org/1999/xlink">
  <homepage xlink:type="simple"
    xlink:href="https://www.w3schools.com">Visit W3Schools</homepage>
  <homepage xlink:type="simple"
    xlink:href="http://www.w3.org">Visit W3C</homepage>
</homepages>
```

- To get access to the XLink features we must declare the XLink namespace. The XLink namespace is: "http://www.w3.org/1999/xlink".
- The xlink:type and the xlink:href attributes in the <homepage> elements come from the XLink namespace.
- The xlink:type="simple" creates a simple "HTML-like" link (means "click here to go there").
- The xlink:href attribute specifies the URL to link to.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore xmlns:xlink="http://www.w3.org/1999/xlink">
  <book title="Harry Potter">
    <description xlink:type="simple"
      xlink:href="/images/HPotter.gif"
      xlink:show="new">
      As his fifth year at Hogwarts School of Witchcraft and Wizardry
      approaches, 15-year-old Harry Potter is.....
    </description>
  </book>
  <book title="XQuery Kick Start">
    <description xlink:type="simple"
      xlink:href="/images/XQuery.gif"
      xlink:show="new">
      XQuery Kick Start delivers a concise introduction to the
      XQuery standard.....
    </description>
  </book>
</bookstore>
```

- The XLink namespace is declared at the top of the document (xmlns:xlink="http://www.w3.org/1999/xlink")
- The xlink:type="simple" creates a simple "HTML-like" link
- The xlink:href attribute specifies the URL to link to (in this case - an image)
- The xlink:show="new" specifies that the link should open in a new window

XHTML:

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>
    <title>Transitional DTD XHTML</title>
</head>

<body bgcolor="#dae1ed">
    <div style="color:#090;font-size:40px;
                font-weight:bold;text-align:center;
                margin-bottom:-25px;">GeeksforGeeks</div>
    <p style="text-align:center;font-size:20px;">
        A computer science portal</p>
    <p style="text-align:center;font-size:20px;">
        Option to choose month:
        <select name="month">
            <option selected="selected">January</option>
            <option>February</option>
            <option>March</option>
            <option>April</option>
            <option>May</option>
            <option>June</option>
            <option>July</option>
            <option>August</option>
            <option>September</option>
            <option>October</option>
            <option>November</option>
            <option>December</option>
        </select>
    </p>
</body>

</html>

```

XSD:

```

<?xml version="1.0" encoding="UTF-8"?>
<shiporder orderid="889923"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="shiporder.xsd">
  <orderperson>John Smith</orderperson>
  <shipto>
    <name>Ola Nordmann</name>
    <address>Langgt 23</address>
    <city>4000 Stavanger</city>
    <country>Norway</country>
  </shipto>
  <item>
    <title>Empire Burlesque</title>
    <note>Special Edition</note>
    <quantity>1</quantity>
    <price>10.90</price>
  </item>
  <item>
    <title>Hide your heart</title>
    <quantity>1</quantity>
    <price>9.90</price>
  </item>
</shiporder>

```


DTD:**XML document with an internal DTD:**

```

<?xml version="1.0"?>
<!DOCTYPE address [
<!ELEMENT address (name, email, phone, birthday)>
<!ELEMENT name (first, last)>
<!ELEMENT first (#PCDATA)>
<!ELEMENT last (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT birthday (year, month, day)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT month (#PCDATA)>
<!ELEMENT day (#PCDATA)>
]>

<address>
  <name>
    <first>Rohit</first>
    <last>Sharma</last>
  </name>
  <email>sharmarohit@gmail.com</email>
  <phone>9876543210</phone>
  <birthday>
    <year>1987</year>
    <month>June</month>
    <day>23</day>
  </birthday>
</address>

```

XML document with an external DTD:**dtd.xml**

```

<?xml version="1.0"?>
<!DOCTYPE address SYSTEM "address.dtd">
<address>
  <name>
    <first>Rohit</first>
    <last>Sharma</last>
  </name>
  <email>sharmarohit@gmail.com</email>
  <phone>9876543210</phone>
  <birthday>
    <year>1987</year>
    <month>June</month>
    <day>23</day>
  </birthday>
</address>

```

address.dtd

- <!ELEMENT address (name, email, phone, birthday)>
- <!ELEMENT name (first, last)>
 - <!ELEMENT first (#PCDATA)>
 - <!ELEMENT last (#PCDATA)>
- <!ELEMENT email (#PCDATA)>
- <!ELEMENT phone (#PCDATA)>
- <!ELEMENT birthday (year, month, day)>
 - <!ELEMENT year (#PCDATA)>
 - <!ELEMENT month (#PCDATA)>
 - <!ELEMENT day (#PCDATA)>

PCDATA (Parsed Character Data):

- **Meaning:** Represents character data that an XML parser will process, potentially interpreting entities and markup.
- **Usage:** Often used when an element can contain text, optionally interspersed with other elements.
- **Example:**
Code

```
<!ELEMENT sender (#PCDATA)>
<sender>Anton Smith</sender>
```

CDATA (Character Data):

- **Meaning:** Represents character data that an XML parser will not process; it's treated as raw text.
- **Usage:** Used to include text, including potential XML fragments, without needing to escape delimiters.
- **Example:**
Code

```
<!ELEMENT square (width, height, #PCDATA)>
<square width="10" height="20">
  This is <p>a paragraph</p> of text.
</square>
```

Feature	#PCDATA	CDATA
Parsing	Parsed by the XML parser	Not parsed by the XML parser
Entity Handling	Entities are expanded by the parser	Entities are not expanded
Markup Handling	Markup (tags) is treated as such	Markup is treated as literal text
Usage in DTD	Defines the content model of an element	Defines the type of attribute value
Example Usage	<sender>John Smith</sender>	<script> <!-- CDATA section --> alert("Hello"); </script>