||JAI SRI GURUDEV||

S J C INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

ANALOG AND DIGITAL ELECTRONICS LABORATORY MANUAL

[18CSL37]

(III SEMESTER)

<u>Prepared By:</u>

Dr. MURTHY SVN
Associate Prof,
Dept. of CSE

Mr. GIRISH BG
Assistant Prof,
Dept. of CSE

S.J.C.INSTITUTE OF TECHNOLOGY, CHICKBALLAPURA

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

2019

DEPARMENT VISION AND MISSION

Vision

To build competent professionals embedded with human values to meet the challenges in the field of Computer Science & Engineering.

Mission

M1 - Empower the graduates with the fundamentals in design and implementation of computational systems through curriculum and research in collaboration with Industry and other organization.

M2 - Imparting Center of Excellence in state-of-Art Infrastructure to make competent graduates by providing professionally committed Faculties

M3 - Inculcate Ethical values, Technical Capabilities & Leadership abilities for meeting the current and future demands of Industry and Society.

Program Educational Objectives (PEOs)

PEO1 : Graduates will have strong foundation in fundamentals to solve Engineering problems in different domains.

PEO2 : Graduates will have successful careers as computer Science Engineers and be able to lead & manage teams.

PEO3: Graduates will instill interpersonal skills and attitudes in the process of Lifelong learning

Program Specific Outcomes (PSO's)

PSO1: Ability to adapt to a rapidly changing environment by learning and employing new programming skills and technologies

PSO2: Ability to use diverse knowledge across the domains with inter-personnel skills to deliver the Industry need

Course objectives:

This laboratory course enables students to get practical experience in design, assembly and evaluation/testing of

- Analog components and circuits including Operational Amplifier, Timer, etc.
- Combinational logic circuits.
- Flip - Flops and their operations
- Counters and registers using flip-flops.
- Synchronous and Asynchronous sequential circuits.
- A/D and D/A converters

Course Outcomes:

The student should be able to:

- Use appropriate design equations / methods to design the given circuit.
- Examine and verify the design of both analog and digital circuits using simulators.
- Make us of electronic components, ICs, instruments and tools for design and testing of circuits for the given the appropriate inputs.
- Compile a laboratory journal which includes; aim, tool/instruments/software/components used, design equations used and designs, schematics, program listing, procedure followed, relevant theory, results as graphs and tables, interpreting and concluding the findings.

Descriptions (if any)

- Simulation packages preferred: Multisim, Modelsim, PSpice or any other relevant.
- For Part A (Analog Electronic Circuits) students must trace the wave form on Tracing sheet / Graph sheet and label trace.
- Continuous evaluation by the faculty must be carried by including performance of a student in
  both hardware implementation and simulation (if any) for the given circuit.
- A batch not exceeding 4 must be formed for conducting the experiment. For simulation individual
  student must execute the program.

Laboratory Programs:

PART A (Analog Electronic Circuits)

1. Design an astable multivibrator ciruit for three cases of duty cycle (50%, <50% and >50%) using NE 555 timer IC. Simulate the same for any one duty cycle.

2. Using ua 741 Opamp, design a 1 kHz Relaxation Oscillator with 50% duty cycle. And simulate the same.

3. Using ua 741 opamap, design a window comparate for any given UTP and LTP. And simulate the same.

PART B (Digital Electronic Circuits)

4. Design and implement Half adder, Full Adder, Half Subtractor, Full Subtractor using basic gates. And implement the same in HDL.

5. Given a 4-variable logic expression, simplify it using appropriate technique and realize the simplified logic expression using 8:1 multiplexer IC. And implement the same in HDL.

6. Realize a J-K Master / Slave Flip-Flop using NAND gates and verify its truth table. And implement the same in HDL.

7. Design and implement code converter I)Binary to Gray (II) Gray to Binary Code using basic gates.

8. Design and implement a mod-n (n<8) synchronous up counter using J-K Flip-Flop ICs and demonstrate its working.

9. Design and implement an asynchronous counter using decade counter IC to count up from 0 to n (n<=9) and demonstrate on 7-segment display (using IC-7447)

Conduct of Practical Examination:

Experiment distribution

- For laboratories having only one part: Students are allowed to pick one experiment from the lot with equal opportunity.
- For laboratories having PART A and PART B: Students are allowed to pick one experiment from PART A and one experiment from PART B, with equal opportunity.
- Change of experiment is allowed only once and marks allotted for procedure to be made zero of the changed part only.
- Marks Distribution *(Courseed to change in accoradance with university regulations)*
- For laboratories having only one part – Procedure + Execution + Viva-Voce: 15+70+15 =100 Marks
- For laboratories having PART A and PART B

  Part A – Procedure + Execution + Viva = 6 + 28 + 6 = 40 Marks

  Part B – Procedure + Execution + Viva = 9 + 42 + 9 = 60 Marks

PART - A

Analog Electronics Experiments

1. Design an astable multivibrator circuit for three cases of duty cycle (50%, <50% and >50%) using NE 555 timer IC. Simulate the same for any one duty cycle.

Description:

Multivibrator is a form of oscillator, which has a non-sinusoidal output. The output waveform is rectangular. The multivibrators are classified as

i)Astable or free running multivibrator It alternates automatically between two states (low and high for a rectangular output) and remains in each state for a time dependent upon the circuit constants. It is just an oscillator as it requires no external pulse for its operation.

ii)Monostable or one shot multivibrators: It has one stable state and one quasi stable. The application of an input pulse triggers the circuit time constants and the output goes to the quazi stable state, after a period of time determined by the time constant, the circuit returns to its initial stable state. The process is repeated upon the application of each trigger pulse.

iii)Bistable Multivibrators: It has both stable states. It requires the application of an external triggering pulse to change the output from one state to other. After the output has changed its state, it remains in that state until the application of next trigger pulse. Flip flop is an example.

Components Required:

555 Timer IC, Resistors of 3.3KΩ, 6.8KΩ, Capacitors of C=0.1 μF, C'=0.01 μF, digital trainer kit(used to give +5v power supply to 555 IC),CRO.

Design:

For astable multivibrator

T_{ON}= 0.693 (R_A+R_B) C

T_{OFF}=0.693 R_B C

With the diode connected in parallel with R_B the effect of R_B is shunted during charging of the capacitor, therefore the equations for T_{ON} and T_{OFF} is given by

TON= 0.693 RA C

T_{OFF}=0.693 R_B C

Case 1: 50% duty cycle

Let Frequency =1kHz, T=1ms, C=0.1 μF

T_{ON} = T_{OFF}=0.5ms

For R_A, 0.5ms= 0.693 * R_A *0.1 *10^{-6}

R_A =7.2 kΩ= R_B

Case 2: >50% duty cycle, let Duty cycle be 75%

Let Frequency =1kHz, T=1ms, C=0.1 μF

TON = 0.75ms

T_{OFF}= 0.25ms

For R_A, 0.75ms= 0.693 * R_A *0.1 *10^{-6}

RA =10 kΩ

For R_B, 0.25ms= 0.693 * R_A *0.1 *10^{-6}

R_B =3.6 kΩ

Case 3: <50% duty cycle, let Duty cycle be 25%
Let Frequency =1kHz, T=1ms, C=0.1 μF
T_{ON} = 0.25ms
T_{OFF}= 0.75ms
For R_A, 0.25ms= 0.693 * R_A *0.1 *10^{-6}
R_A =3.6 kΩ

For R_B, 0.75ms= 0.693 * R_A *0.1 *10^{-6}
RB =10 kΩ

Circuit Diagram:

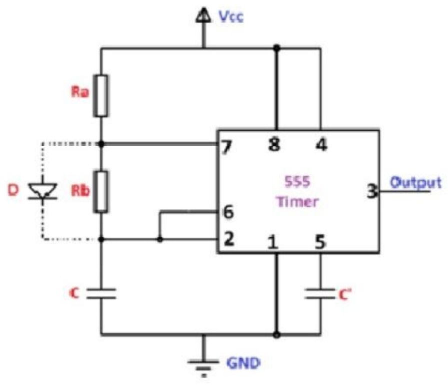


Figure 1: astable mutivibrator
Connect the pin 2 to the CRO to get the capacitor waveform check the amplitude from the waveform to get the UTP and LTP values.
Connect pin 3 to CRO to get the output. Find out the T_H and T_L values.

Procedure:
1. Before making the connections, check the components using multimeter.
2. Make the connections as shown in figure and switch on the power supply.
3. Observe the capacitor voltage waveform at 6th pin of 555 timer on CRO.
4. Observe the output waveform at 3rd pin of 555 timer on CRO (shown below).
5. Note down the amplitude levels, time period and hence calculate duty cycle.

The Vcc determines the upper and lower threshold voltages (observed from the capacitor voltage waveform) as . $V_{UT} = \frac{2}{3}V_{CC} \wedge V_{LT} = \frac{1}{3}V_{CC}$.

Result:
The frequency of the oscillations = 1KHz.

Output Waveforms

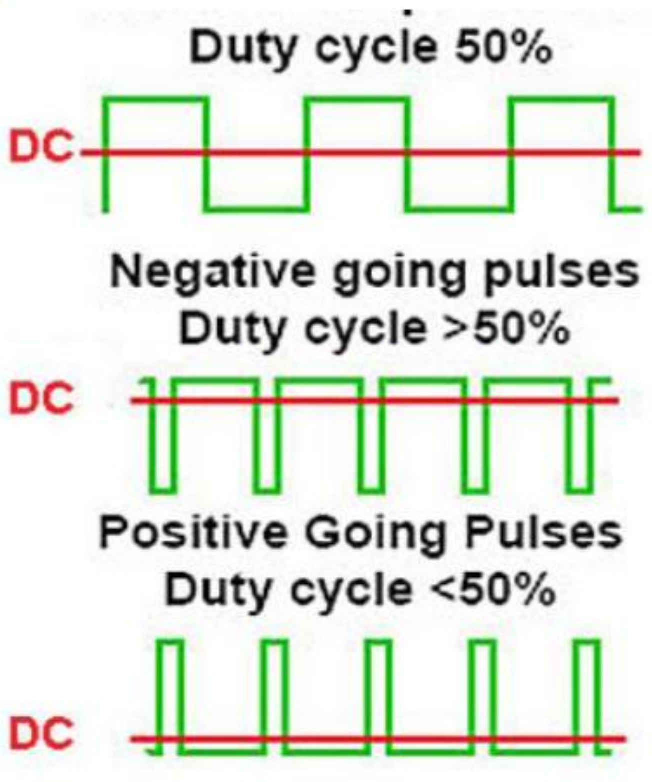


Figure 2: astable multivibrator output waveforms

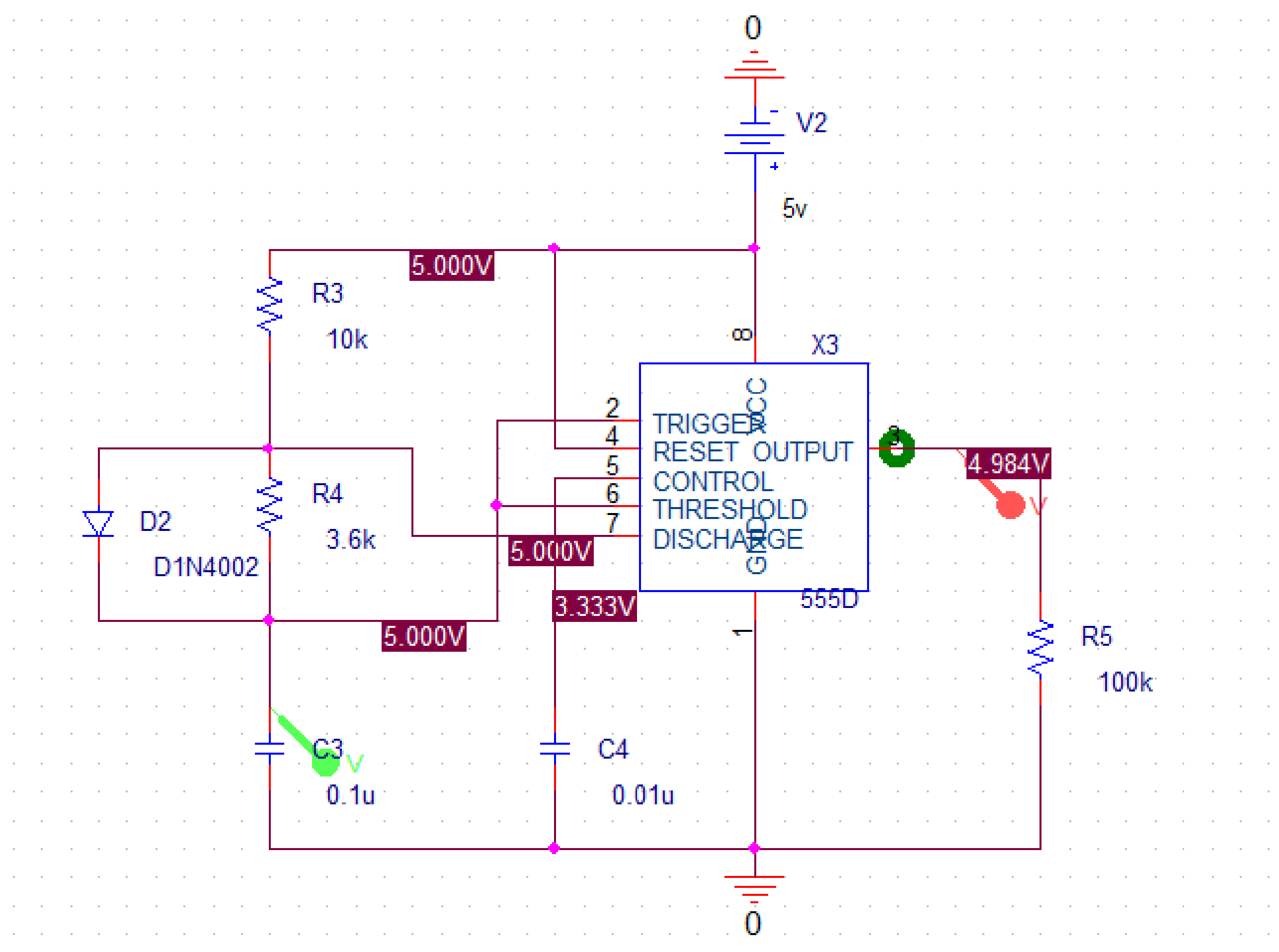
Result:

Note:

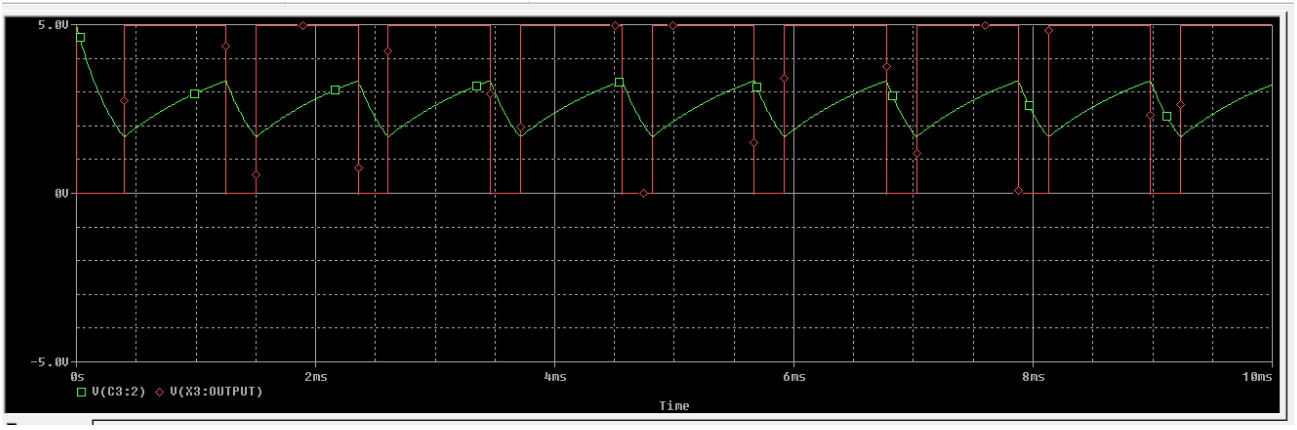
Each division in oscilloscope is 0.2
Time=no of div in x-axis x time base
Amplitude= no of div in y-axis x volt/div
Duty cycle= (Ton/Ton +Toff) *100

Duty cycle	Duty cycle	Ton	Toff
Theoretical	50%	0.5ms	0.5ms
	75%	0.75ms	0.25ms
	25%	0.25ms	0.75ms
Practical	50%		
	75%		
	25%		

Simulation:
Circuit diagram: Astable multivibrator for duty cycle >50%



Output waveform:



Type of analysis: TIME DOMAIN (TRANSIENT)
Run to time: 10m
Step size: 0.01m

2. Using ua 741 Opamp, design a 1 kHz Relaxation Oscillator with 50% duty cycle. And simulate the same.

Components required: Op-amp μA 741, Resistor of 1KΩ, 10KΩ, 20 kΩ Potentiometer, Capacitor of 0.1 μF, Fixed DC power supply ±12v, CRO, connecting board.

Design : The period of the output rectangular wave is given as $T = 2RC\ln\left(\frac{1+\beta}{1-\beta}\right)$ -------(1)

Where, $\beta = \frac{R_1}{R_1 + R_2}$ is the feedback fraction

If $R_1 = R_2$, then from equation (1) we have T = 2RC ln(3)

Another example, if R_2=1.16 R_1, then T = 2RC ----------(2)

Example: Design for a frequency of 1kHz (implies $T = \frac{1}{f} = \frac{1}{10^3} = 10^{-3} = 1ms$)

Use R_2=1.16 R_1, for equation (2) to be applied.
Let R_1 = 10kΩ, then R_2 = 11.6kΩ (use 20kΩ potentiometer as shown in circuit figure)
Choose next a value of C and then calculate value of R from equation (2).

Let C=0.1μF (i.e., 10^{-7}), then $R = \frac{T}{2C} = \frac{10^{-3}}{2 \times 10^{-7}} = 5K\Omega$

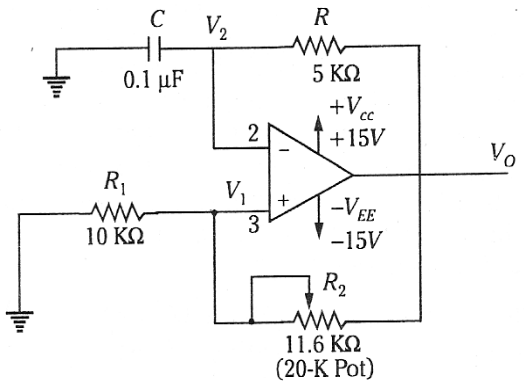
The voltage across the capacitor has a peak voltage of $V_c = \frac{R_1}{R_1 + R_2} V_{sat}$

Procedure :

1. Before making the connections check all the components using multimeter.
2. Make the connections as shown in figure and switch on the power supply.
3. Observe the voltage waveform across the capacitor on one of the channel on CRO.
4. Also observe the output waveform at pin 6 on another channel of CRO. Measure its amplitude and time & find frequency.

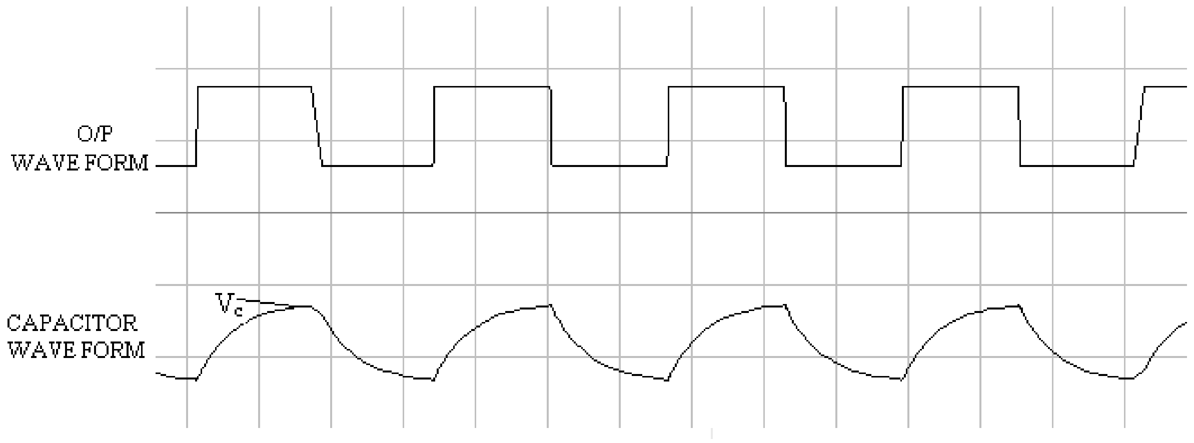
NOTE: There is no I/P signal in this circuit.

Circuit Diagram:



Values
C=0.1μF
R_1 = 10kΩ,
R_2 = 11.6 kΩ,
R = 4.7k/5.1kΩ

Waveforms



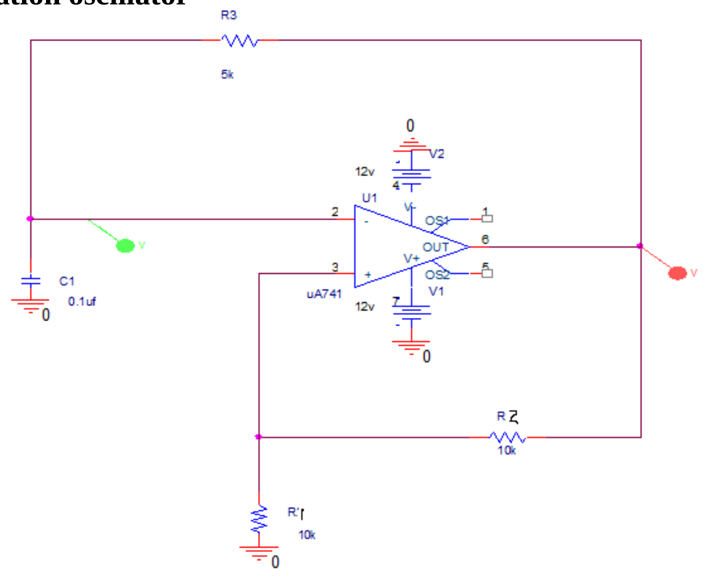
Theory:

Op-Amp Relaxation Oscillator is a simple Square wave generator which is also called as a Free running oscillator or Astable multivibrator or Relaxation oscillator. In this figure the op-amp operates in the saturation region. Here, a fraction (R2/(R1+R2)) of output is fed back to the noninverting input terminal. Thus reference voltage is (R2/(R1+R2)) Vo. And may take values as +(R2/(R1+R2)) Vsat or - (R2/(R1+R2)) Vsat. The output is also fed back to the inverting input terminal after integrating by means of a low-pass RC combination. Thus whenever the voltage at inverting input terminal just exceeds reference voltage, switching takes place resulting in a square wave output.

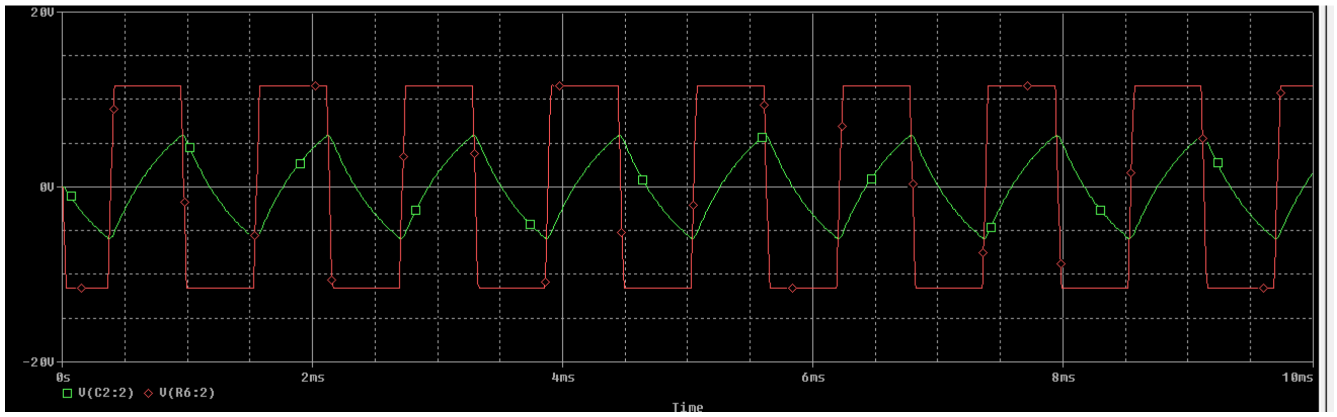
Result: The frequency of the oscillations =_____Hz.

Simulation:

Circuit diagram: Relaxation oscillator



Waveforms from simulation T= 1ms f=1khz



Type of analysis: TIME DOMAIN (TRANSIENT)
Run to time: 10m
Step size: 0.01m

3. Using ua741 Op-amp, design window comparator for any given UTP and LTP. And simulate the same.

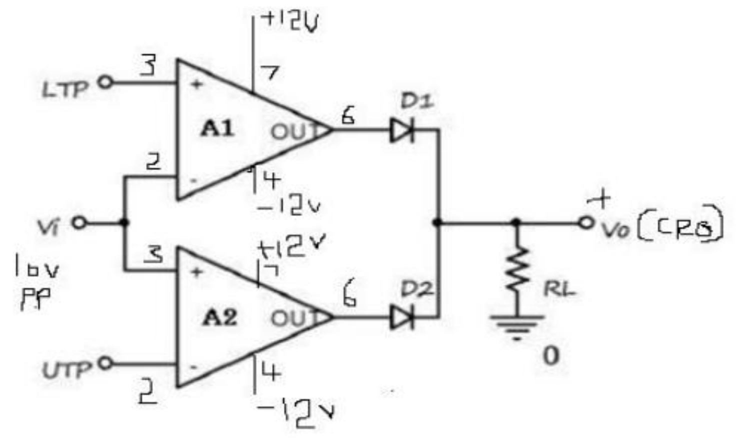
Description:

A Window Comparator is basically the inverting and the non-inverting comparators combined into a single comparator stage. The window comparator detects input voltage levels that are within a specific band or window of voltages, instead of indicating whether a voltage is greater or less than some preset or fixed voltage reference point.

This time, instead of having just one reference voltage value, a window comparator will have two reference voltages implemented by a pair of voltage comparators. One which triggers an op-amp comparator on detection of some upper voltage threshold, $V_{REF(UPPER)}$ and one which triggers an op-amp comparator on detection of a lower voltage threshold level, $V_{REF(LOWER)}$. Then the voltage levels between these two upper and lower reference voltages is called the "window", hence its name.

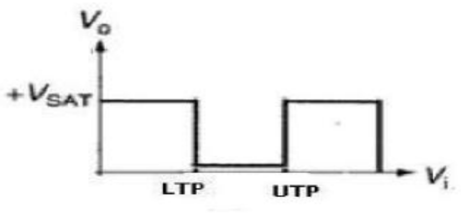
Components Required:
Two Op amp IC μA 741, Two diode 1N4007, Resistor of 1KΩ, DC regulated power Supply, trainer kit (+12v & -12v is given to Op amp from this), Signal generator, CRO.

Circuit:



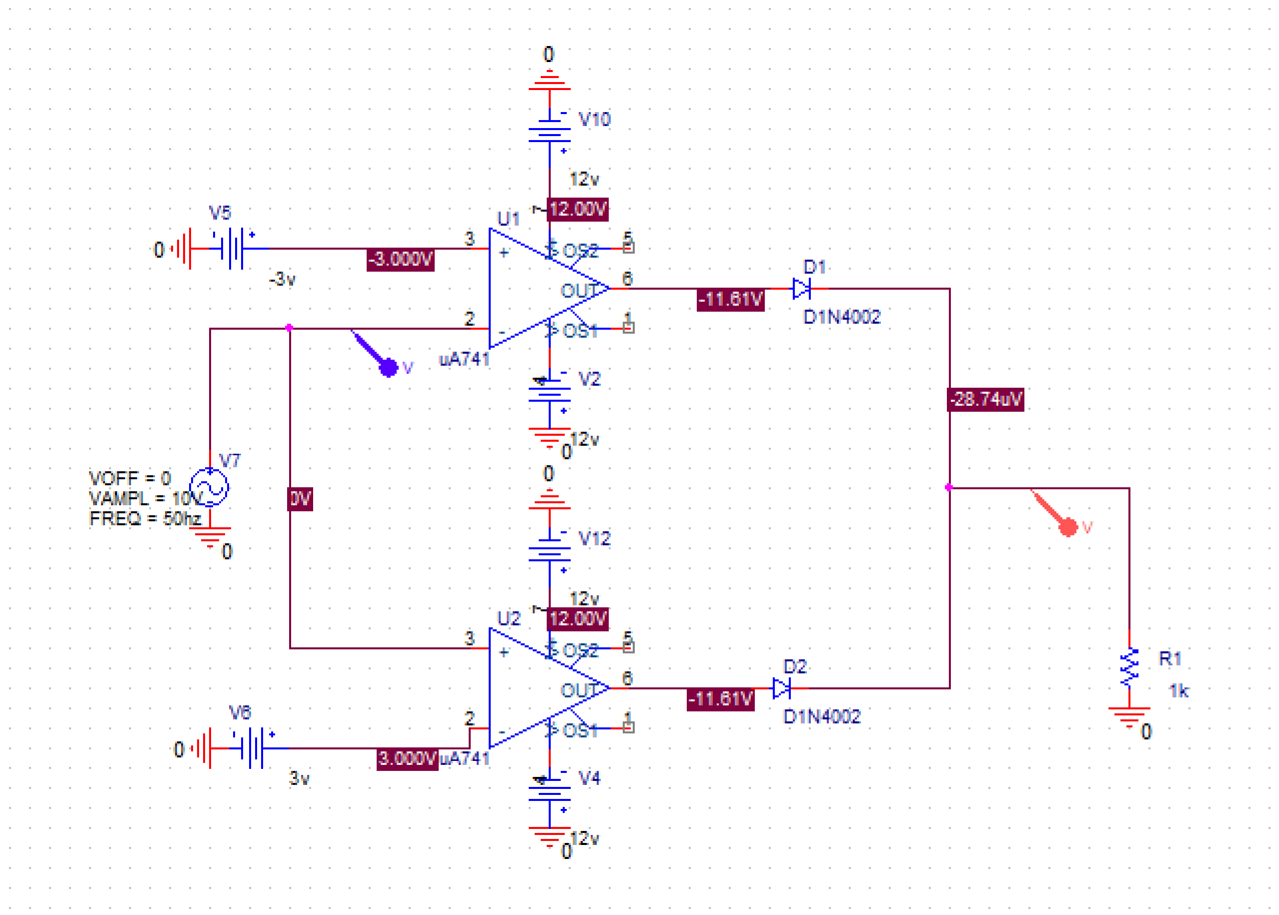
Circuit Diagram for Window comparator

Output waveform

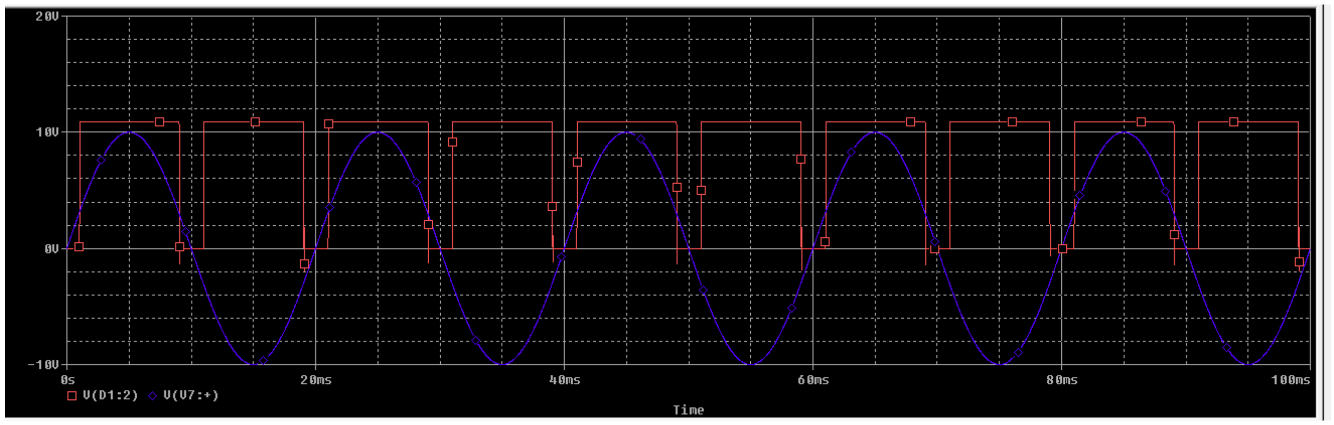


Simulation:
Components to be placed in the schematic:
Two Op amp IC µA 741, Two diode 1N4007, Resistor of 1KΩ, DC regulated power Supply, trainer kit (+12v & -12v is given to Op amp from this).

UTP =3V, LTP = -3V

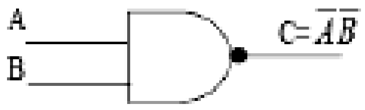
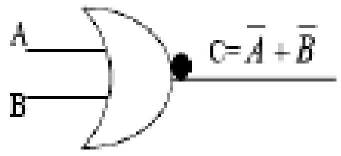
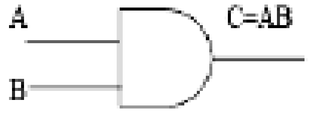
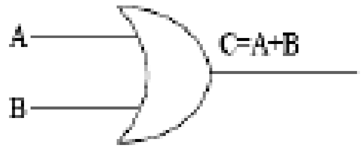
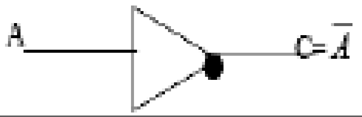
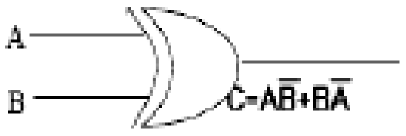


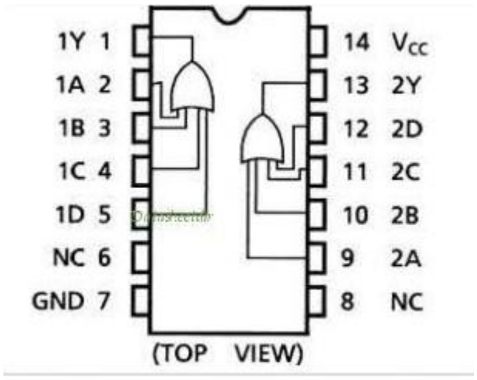
Output waveform:



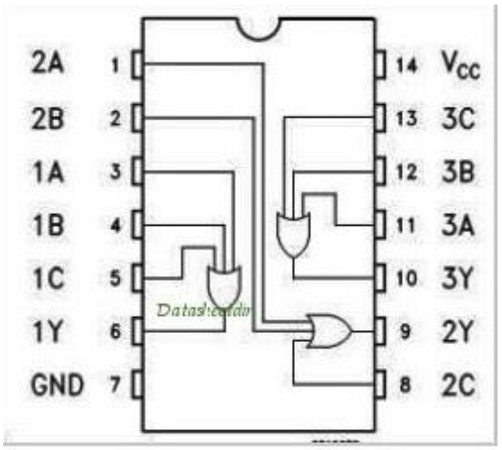
PART - B
DIGITAL ELECTRONICS EXPERIMENTS

BASIC GATES

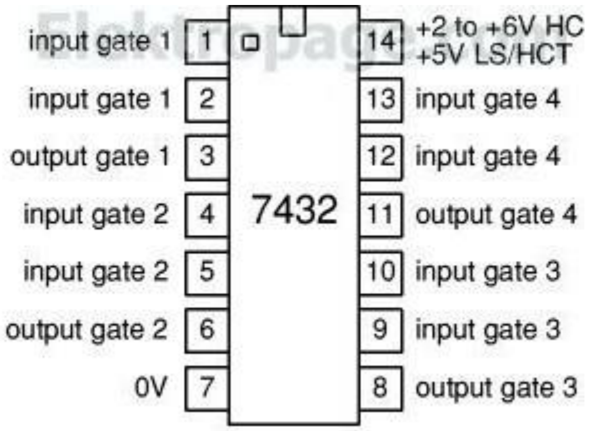
S.NO	GATE	SYMBOL	INPUTS		OUTPUT
			A	B	C
1.	NAND IC 7400		0	0	1
			0	1	1
			1	0	1
			1	1	0
2.	NOR IC 7402		0	0	1
			0	1	0
			1	0	0
			1	1	0
3.	AND IC 7408		0	0	0
			0	1	0
			1	0	0
			1	1	1
4.	OR IC 7432		0	0	0
			0	1	1
			1	0	1
			1	1	1
5.	NOT IC 7404		1	-	0
			0	-	1
6.	EX-OR IC 7486		0	0	0
			0	1	1
			1	0	1
			1	1	0



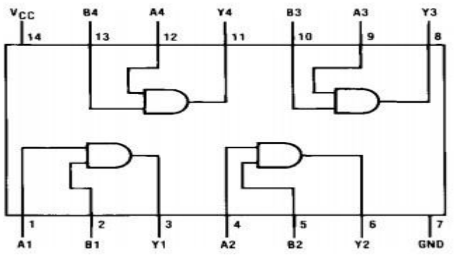
DUAL 4-INPUT OR GATE M74HC4072



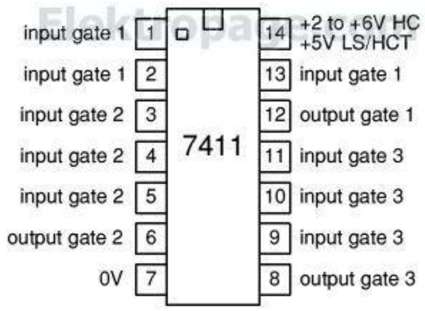
Dual 3 input or gate M54HC4075



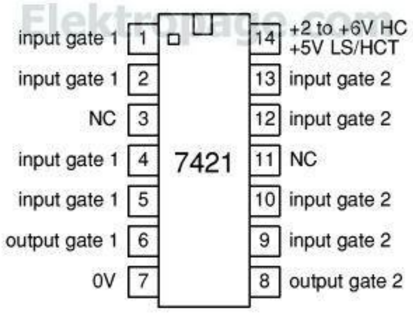
2 INPUT OR GATE



7408 2-INPUT AND GATE



3 INPUT AND GATE



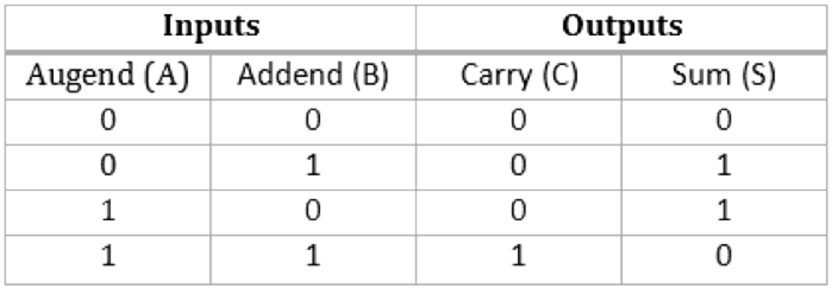
4 INPUT AND GATE

4. Design and implement Half adder, Full Adder, Half Subtractor, Full Subtractor using basic gates. And implement the same using HDL.

HALF ADDER:

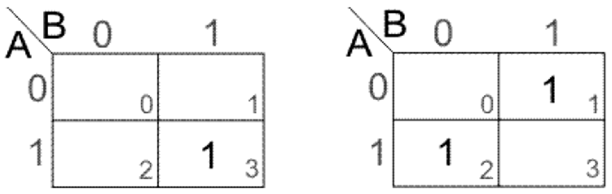
For designing a half adder logic circuit, we first have to draw the truth table for two input variables i.e. the augend and addend bits, two outputs variables carry and sum bits. In first three binary additions, there is no carry hence the carry in these cases are considered as 0.

TRUTH TABLE FOR HALF ADDER:



Inputs		Outputs	
Augend (A)	Addend (B)	Carry (C)	Sum (S)
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Kmap for Half Adder Now from this truth table we can draw Kmap for carries and sums separately.

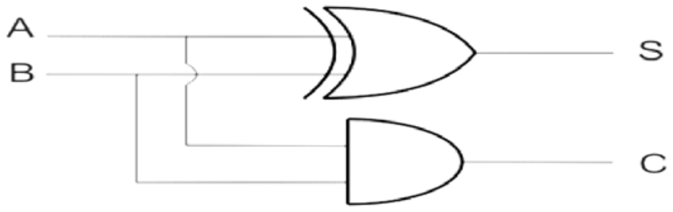


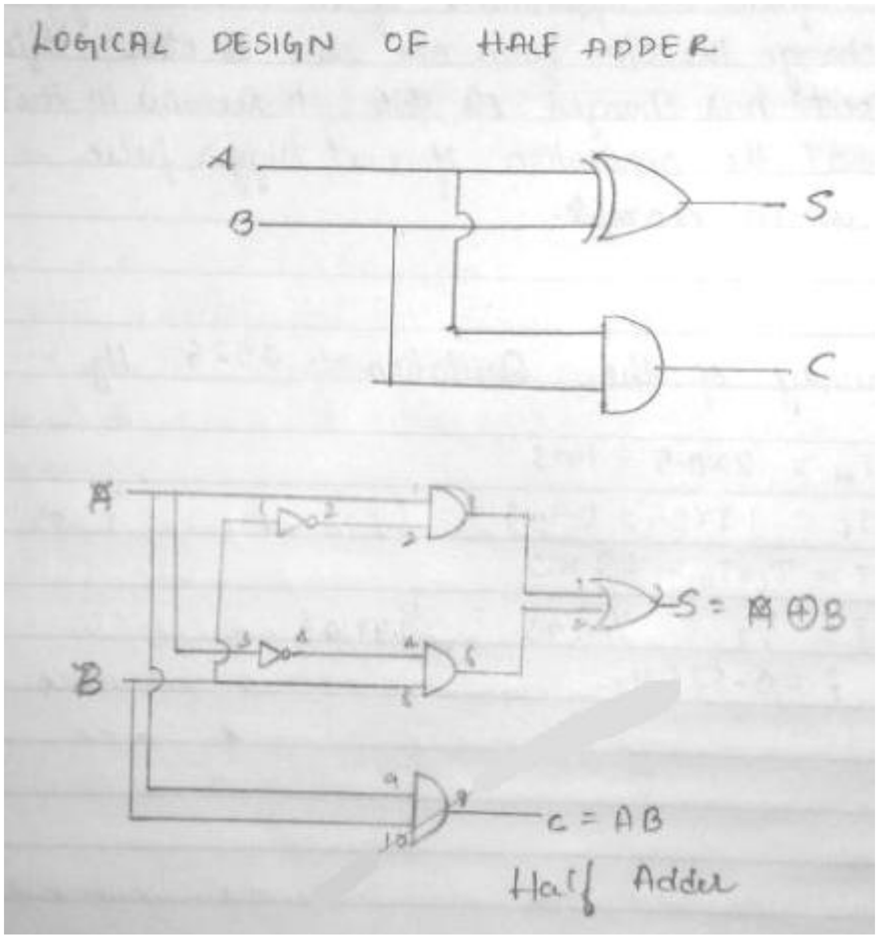
K-map for Carry **K-map for Sum**

For above Kmaps we get,

$$Carry\ \ C = AB\ and\ Sum\ \ C = A\overline{B} + \overline{A}B.$$

Hence, the logical design of Half Adder would be



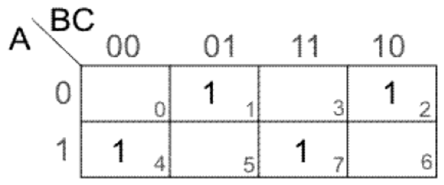


FULL ADDER:

Full adder is a conditional circuit which performs full binary addition that means it adds two bits and a carry and outputs a sum bit and a carry bit. Any bit of augend can either be 1 or 0 and we can represent with variable A, similarly any bit of addend we represent with variable B. The carry after addition of same significant bit of augend and addend can represent by C. Hence truth table for all combinations of A, B and C is as follows,

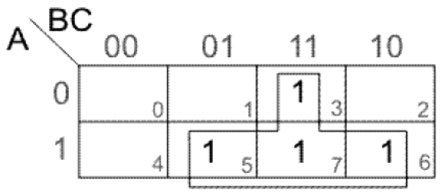
SL No	Augend (A)	Addend (B)	Carry (C)	SUM (S)	Final Carry (C_{out})
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

From the above table, we can draw Kmap for sum (s) and final carry (Cout).



A \ BC	00	01	11	10
0	0	1 1	3	1 2
1	1 4	5	1 7	6

K-map for Sum (S)



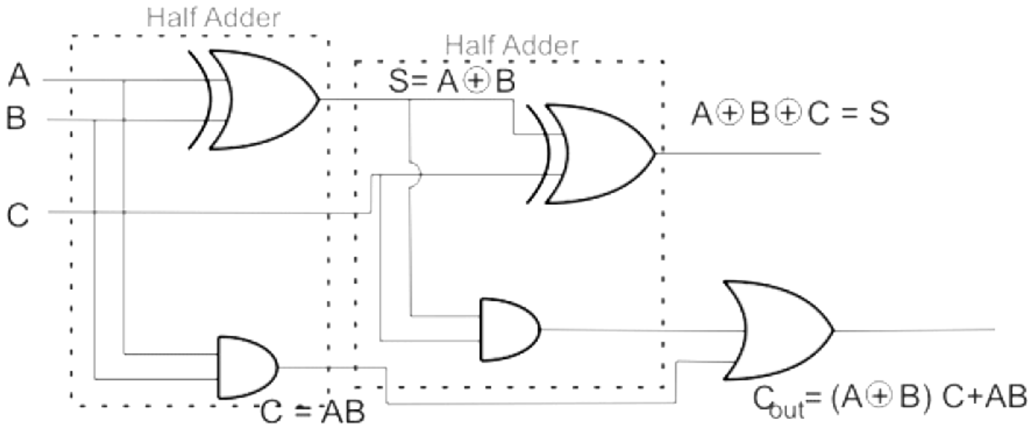
A \ BC	00	01	11	10
0	0	1	1 3	2
1	4	1 5	1 7	1 6

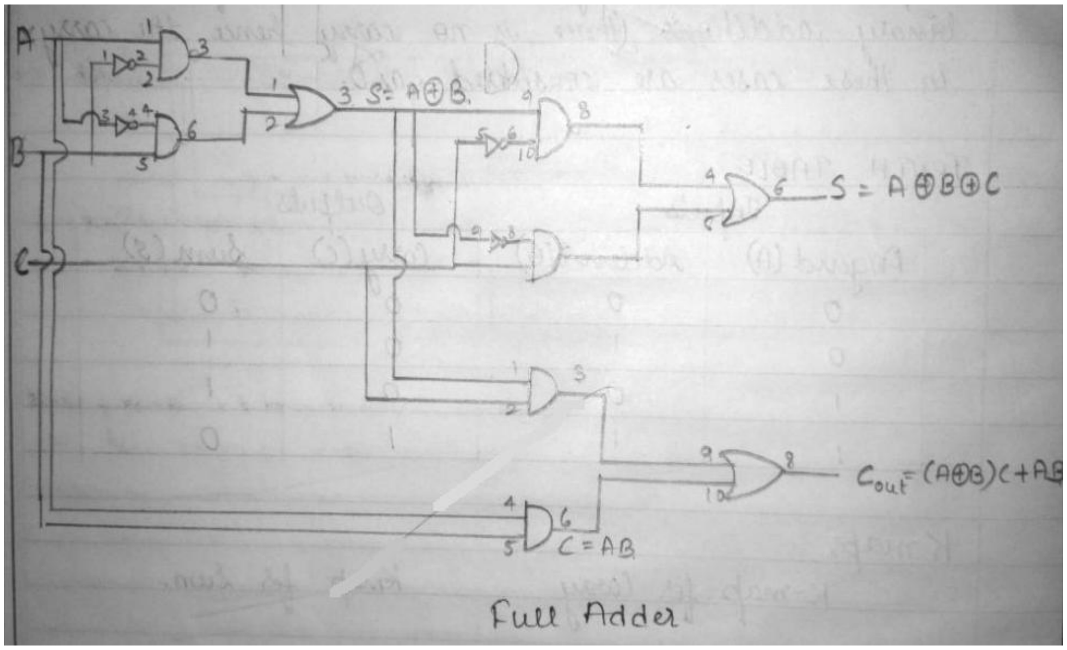
K-map for Carry (C out)

Hence, from Kmaps,

$$\begin{aligned} S &= A\bar{B}\bar{C} + \bar{A}\,\bar{B}C + ABC + \bar{A}B\bar{C} \\ &= C\,(AB + \bar{A}\,\bar{B}) + \bar{C}\,(\bar{A}B + A\,\bar{B}) \\ &= C\,(\overline{\bar{A}B + A\,\bar{B}}) + \bar{C}\,(\bar{A}B + A\,\bar{B}) \\ &= C\,(\overline{A \oplus B}) + \bar{C}\,(A \oplus B) \ = A \oplus B \oplus C \end{aligned}$$

$$\begin{aligned} C_{out} &= \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC \\ &= (\bar{A}B + A\bar{B})\,C + AB\,(\bar{C} + C) \\ &= (A \oplus B).C + AB. \end{aligned}$$





HALF SUBTRACTOR:

Half substractor is a combinational circuit which performs substraction of single bit binary numbers. The substraction combinations of two single bit binary numbers can be,

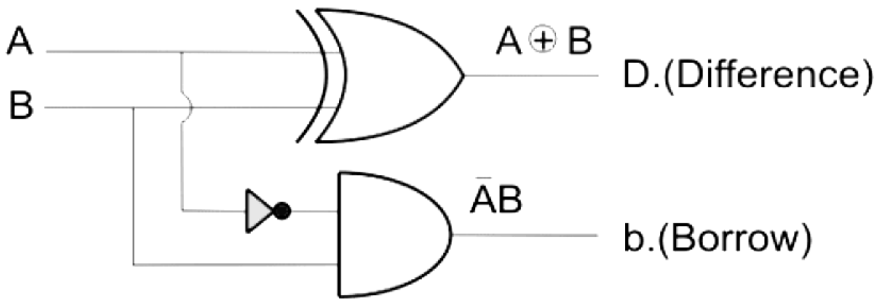
Now if we draw a truth table for that, with all differences (D) and borrow (b), we get,

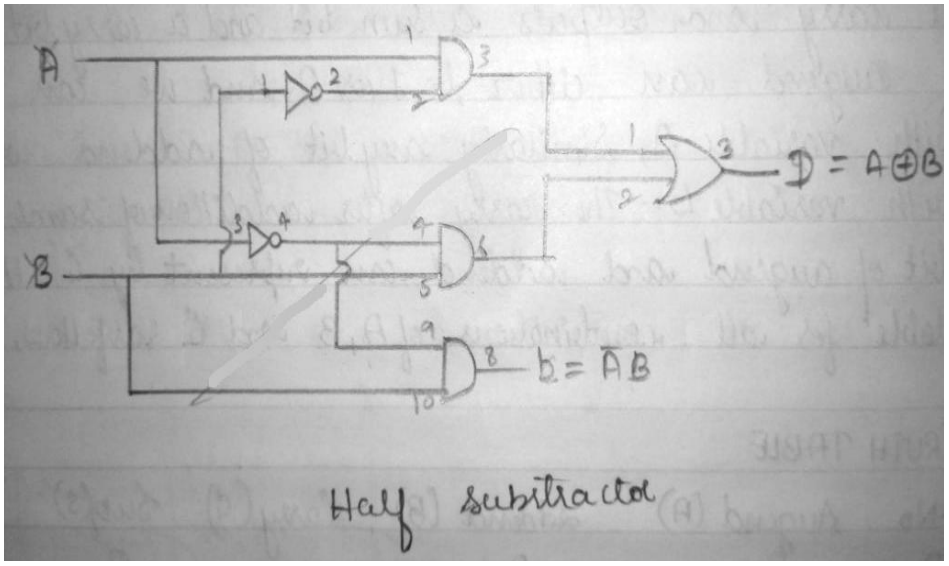
Minuend (A)	Subtrahend (B)	Difference (D)	Borrow (b)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Hence, from truth table it is found that,

$$D = A \oplus B \ and \ b = \overline{A}B$$

The above equations can be represented using logic gates.

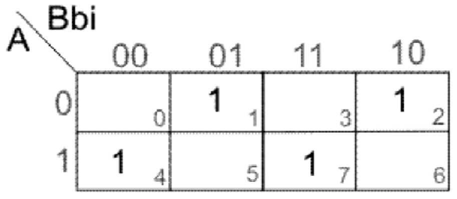




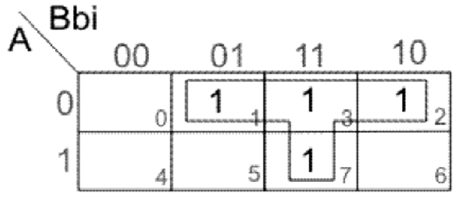
FULL SUBSTRACTOR:

This is not practical to perform substraction only between two single bit binary numbers. Instead binary numbers are always multibits. The substraction of two binary numbers is performed bit by bit from right (LSB) to left (MSB). During substraction of same significant bit of minuend and subtrahend, there may be one borrow bit along with difference bit. This borrow bit (either 0 or 1) is to be added to the next higher significant bit of minuend and then next corresponding bit of subtrahend to be subtracted from this. It will continue up to MSB. The combinational logic circuit performs this operation is called full substractor. Hence, full substractor is similar to half substractor but inputs in full substractor are three instead of two. Two inputs are for the minuend and subtrahend bits and third input is for borrowed which comes from previous bits substraction. The outputs of full adder are similar to that of half adder, these are difference (D) and borrow (b). The combination of minuend bit (A), subtrahend bit (B) and input borrow (bi) and their respective differences (D) and output borrows (b) are represented in a truth table, as follow

Sl No.	Minuend bit (A)	Subtrahend bit (B)	Input borrow (bi)	Difference (D)	Borrow (b)
0	0	0	0	0	0
1	0	0	1	1	1
2	0	1	0	1	1
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	0
6	1	1	0	0	0
7	1	1	1	1	1



K-map for D



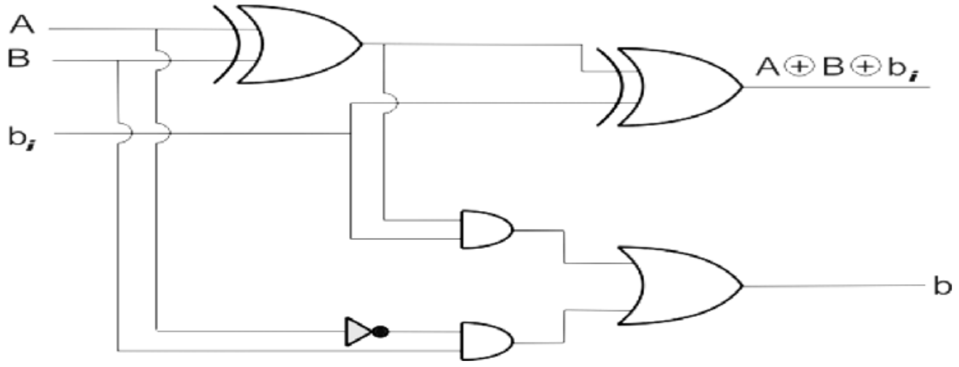
K-map for b

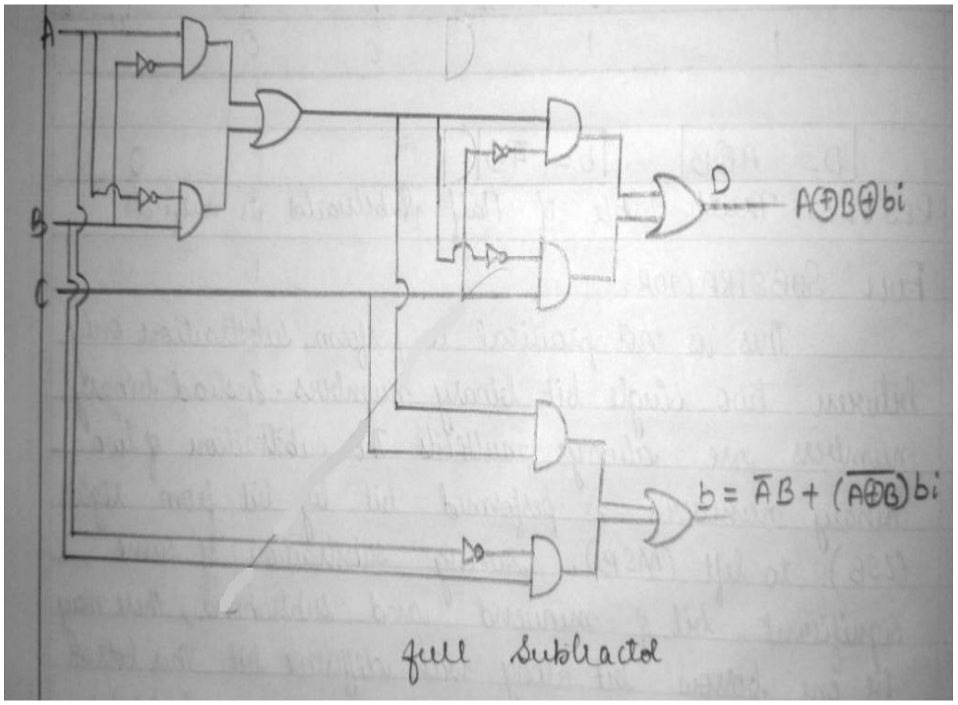
$$D = A\overline{B}\,\overline{bi} + ABbi + \overline{A}B\overline{bi} \;= \overline{bi}(A\overline{B} + \overline{A}B) + bi(\overline{A}\,\overline{B} + AB)$$

$$= \overline{bi}(A\overline{B} + \overline{A}B) + bi(\overline{A\overline{B} + \overline{A}B}) \;= bi \oplus A \oplus B = A \oplus B \oplus bi$$

$$b = \overline{A}\,\overline{B}bi + \overline{A}Bbi + \overline{A}B\overline{bi} + ABbi \;= \overline{A}B\,(bi + \overline{bi}) + (AB + \overline{A}\,\overline{B})\,bi$$

$$= \overline{A}B + (\overline{A \oplus B})\,bi$$





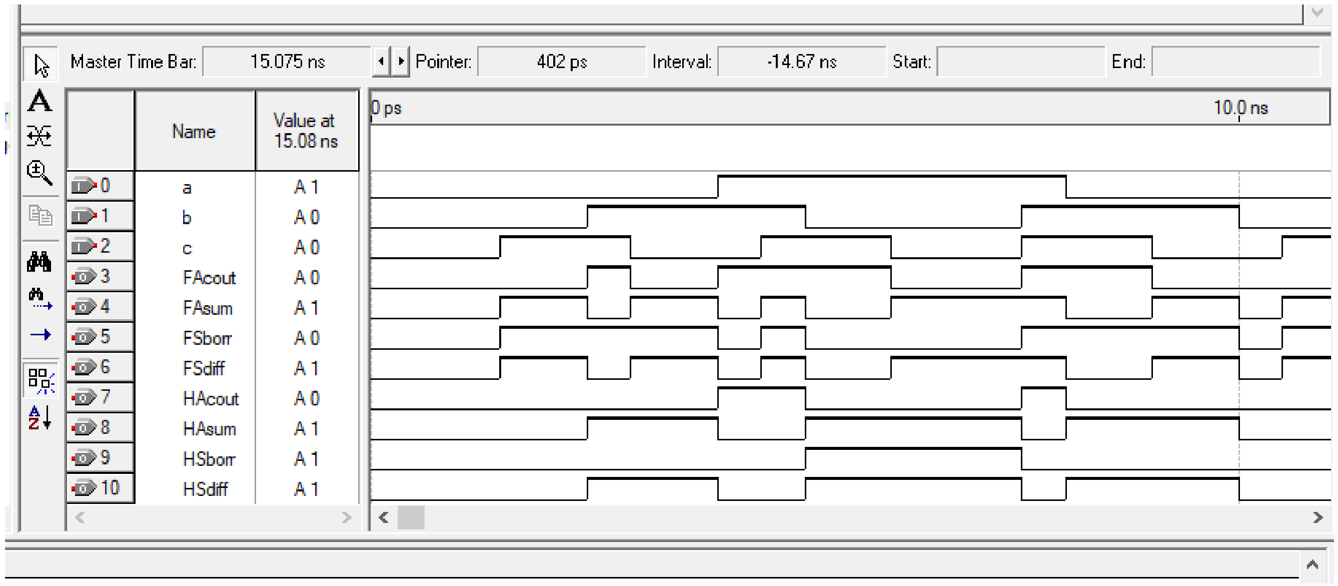
RESULT:
Truth table is verified

Simulation:

VHDL code for adder, subtractor

```
library ieee;
use ieee.std_logic_1164.all;
entity adder is
        port(a,b,c: in std_logic;
        HAsum, HAcout, FAsum, FAcout, HSdiff, HSborr, FSdiff, FSborr: out std_logic);
end adder;
architecture dataflow of adder is
begin
        HAsum<= a xor b;
        HAcout <= a and b;
        FAsum<= a xor b xor c;
        FAcout <= ((a and b)or(b and c) or(a and c));
        HSdiff<= a xor b;
        HSborr <= (a and (not b));
        FSdiff<= a xor b xor c;
        FSborr <= ((b xor c) and (not a)) or (b and c);
end dataflow;
```

Waveform:



Note: File name, project name, entity name should be same

5. Given any 4-variable logic expression, simplify it using appropriate technique and realize the simplified logic expression using 8:1 multiplexer IC. And implement the same in HDL.

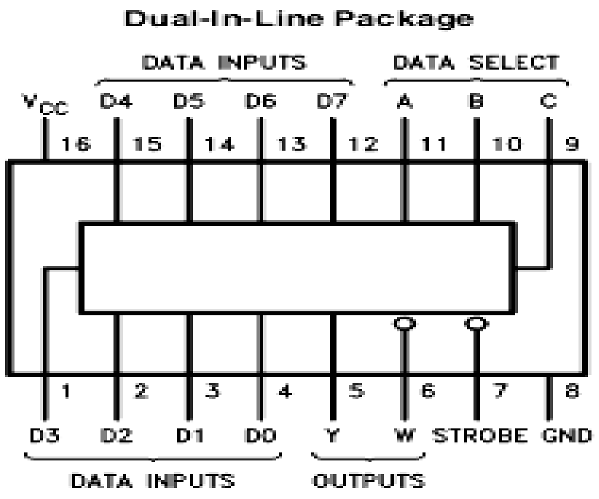
a) E.g., Simplify the function using MEV technique

F(a,b,c,d)=∑m(2,3,4,5,13,15)+dc(8,9,10,11)

Decimal	LSB	f	MEV map entry
0}0 1	0000 0001	0 0	0------Do
1}2 9	0010 0011	1 1	1------D1
2}4 5	0100 0101	1 1	1-----D2
3}6 7	0110 0111	0 0	0-----D3
4}8 9	1000 1001	X X	X-----D4
5}10 11	1010 1011	X X	X-----D5
6}12 13	1100 1101	0 1	d----D6
7}14 15	1110 1111	0 1	d----D7

<u>Components Used:</u> IC 74LS151, Patch Chords, Trainer kit.

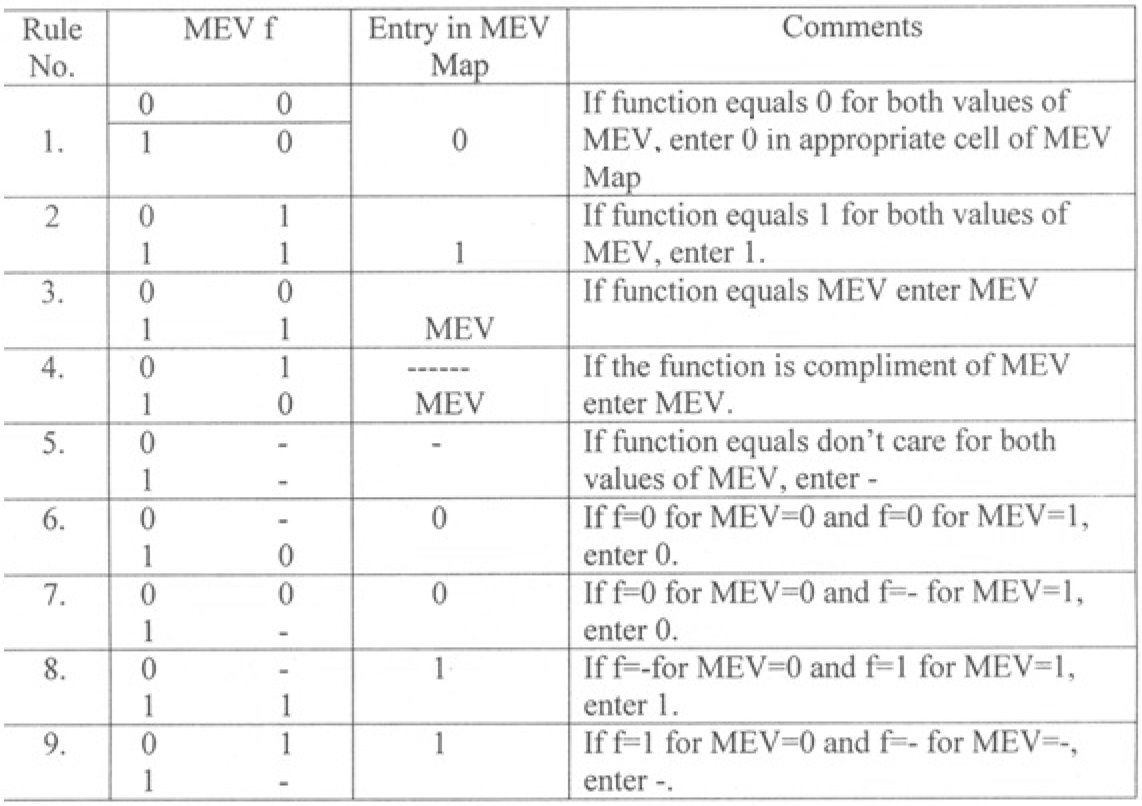
Pin Diagram of Ics Used:



Theory:

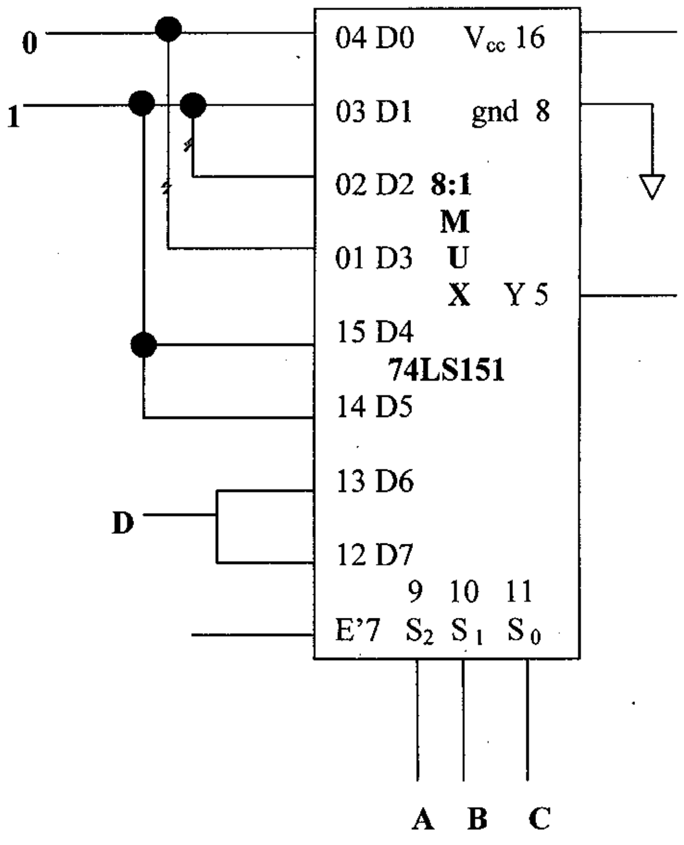
Map Entered Variable Method:

Rules for entering values in a MEV K Map:



Rule No.	MEV	f	Entry in MEV Map	Comments
1.	0 1	0 0	0	If function equals 0 for both values of MEV, enter 0 in appropriate cell of MEV Map
2	0 1	1 1	1	If function equals 1 for both values of MEV, enter 1.
3.	0 1	0 1	MEV	If function equals MEV enter MEV
4.	0 1	1 0	$\overline{MEV}$	If the function is compliment of MEV enter $\overline{MEV}$.
5.	0 1	- -	-	If function equals don't care for both values of MEV, enter -
6.	0 1	- 0	0	If f=0 for MEV=0 and f=0 for MEV=1, enter 0.
7.	0 1	0 -	0	If f=0 for MEV=0 and f=- for MEV=1, enter 0.
8.	0 1	- 1	1	If f=-for MEV=0 and f=1 for MEV=1, enter 1.
9.	0 1	1 -	1	If f=1 for MEV=0 and f=- for MEV=-, enter -.

Circuit Diagram:



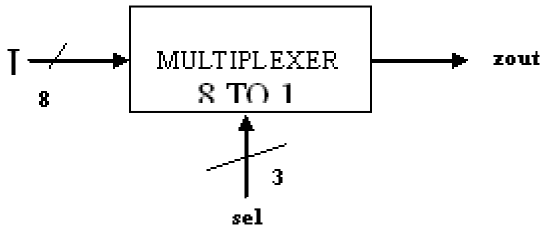
Procedure:

(1) Verify all components and patch chords whether they are in good condition not.
(2) Make connection as shown in the circuit diagram.
(3) Give supply to the trainer kit.
(4) Provide input data to circuit via switches.
(5) Verify truth table sequence and observe outputs.

Result:

Simulation: VHDL code for 8:1 MUX

Description:
An 8:1 multiplexer has 8 inputs and one output. The data stored in one of these 8 input line is transferred serially to the output based on the value of the selection bits.



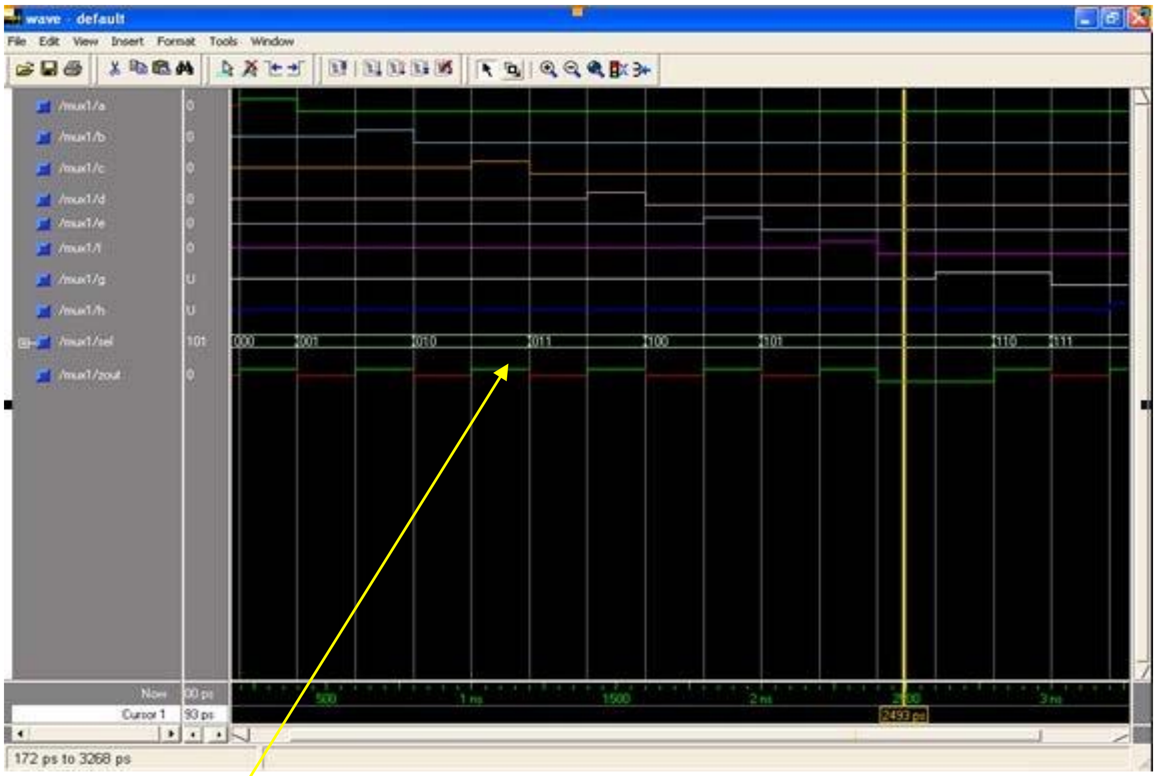
VHDL code for 8 to 1 MUX (behavioral modeling):

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL; // includes the standard library
entity mux1 is
        Port ( I : in std_logic_vector(7 downto 0);
        sel : in std_logic_vector(2 downto 0); //Input and output is declared as ports
        zout : out std_logic);
end mux1;
architecture Behavioral of mux1 is
begin
        zout<= I(0) when sel="000" else // Based on the value of selection the value of data
        I(1) when sel="001" else //stored in the array I is stored in zout
        I(2) when sel="010" else
        I(3) when sel="011" else
        I(4) when sel="100" else
        I(5) when sel="101" else
        I(6) when sel="110" else
        I(7);
end Behavioral;
```

TruthTable

INPUTS			OUTPUTS
SEL (2)	SEL (1)	SEL (0)	Zout
0	0	0	I(0)
0	0	1	I(1)
0	1	0	I(2)
0	1	1	I(3)
1	0	0	I(4)
1	0	1	I(5)
0	1	1	I(6)
1	1	1	I(7)

Wavefrom:

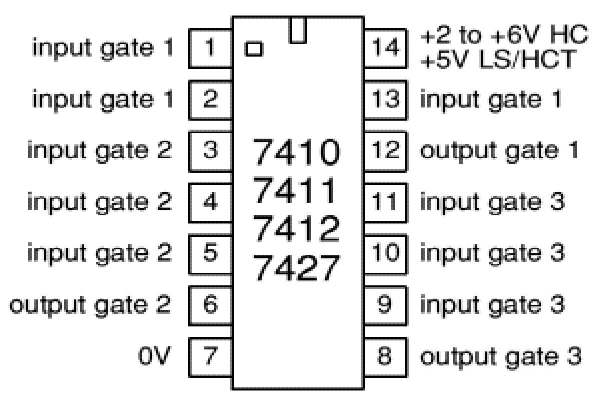


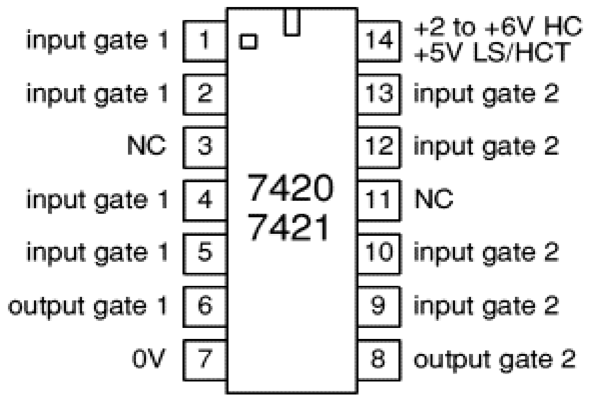
output

5.Realize a J-K Master/Slave FF using NAND gates and verify its truth table. And implement the same in HDL.

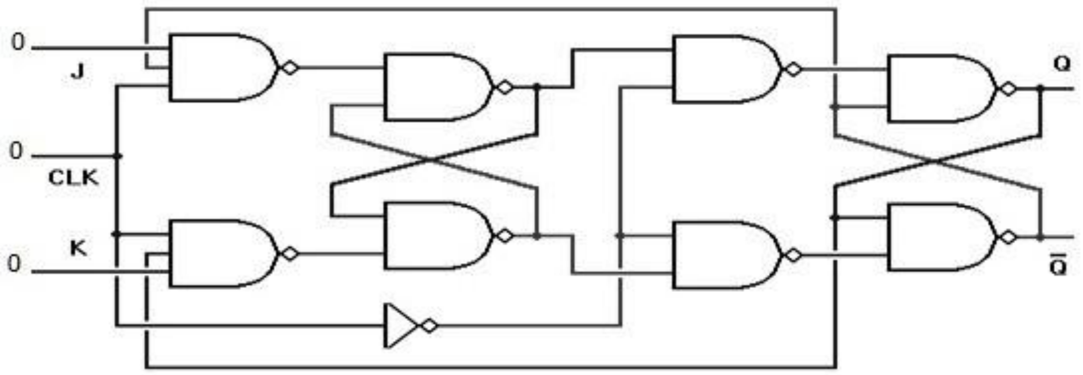
Components used: IC 74LS00, IC 74LS10, IC 74LS20, Power chords, Patch chords, Trainer kit.

Pin Details of the ICs:





Theory:
The circuit below shows the solution. To the RS flip-flop we have added two new connections from the Q and Q' outputs back to the original input gates. Remember that a NAND gate may have any number of inputs, so this causes no trouble. To show that we have done this, we change the designations of the logic inputs and of the flip-flop itself. The inputs are now designated J (instead of S) and K (instead of R). The entire circuit is known as a *JK flip-flop*.



In most ways, the JK flip-flop behaves just like the RS flip-flop. The Q and Q' outputs will only change state on the falling edge of the CLK signal, and the J and K inputs will control the future output state pretty much as before. However, there are some important differences.

Since one of the two logic inputs is always disabled according to the output state of the overall flip-flop, the master latch cannot change state back and forth while the CLK input is at logic 1. Instead, the enabled input can change the state of the master latch *once*, after which this latch will not change again. This was not true of the RS flip-flop.

If both the J and K inputs are held at logic 1 and the CLK signal continues to change, the Q and Q' outputs will simply change state with each falling edge of the CLK signal. (The master latch circuit will change state with each *rising* edge of CLK.) We can use this characteristic to advantage in a number of ways. A flip-flop built specifically to operate this way is typically designated as a *T* (for *Toggle*) flip-flop. The lone T input is in fact the CLK input for other types of flip-flops.

The JK flip-flop *must* be edge triggered in this manner. Any level-triggered JK latch circuit will oscillate rapidly if all three inputs are held at logic 1. This is not very useful. For the same reason, the T flip-flop must also be edge triggered. For both types, this is the only way to ensure that the flip-flop will change state only once on any given clock pulse. Because the behavior of the JK flip-flop is completely predictable under all conditions, this is the preferred type of flip-flop for most logic circuit designs. The RS flip-flop is only used in applications where it can be guaranteed that both R and S cannot be logic 1 at the same time.

At the same time, there are some additional useful configurations of both latches and flip-flops. In the next pages, we will look first at the major configurations and note their properties. Then we will see how multiple flip-flops or latches can be combined to perform useful functions and operations.

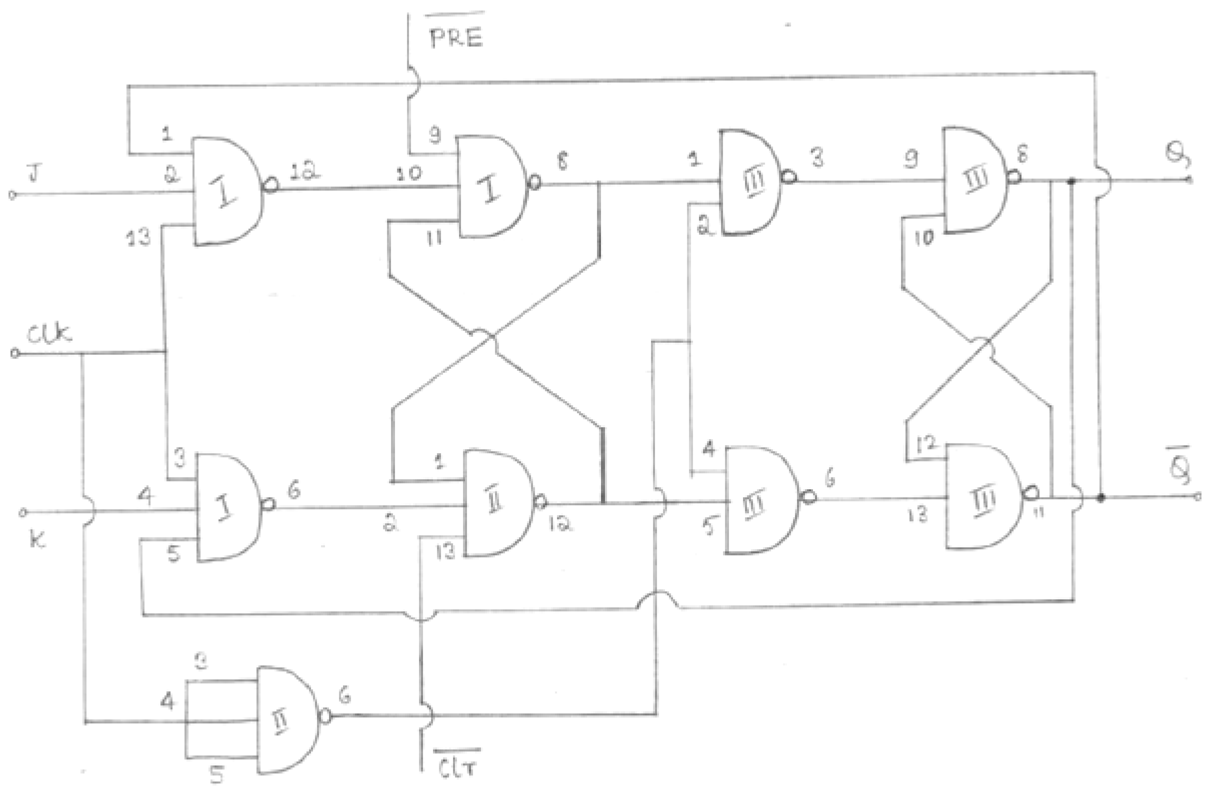
Master Slave Flip Flop: The control inputs to a clocked flip flop will be making a transition at approximately the same times as triggering edge of the clock input occurs. This can lead to unpredictable triggering.

A JK master flip flop is positive edge triggered, where as slave is negative edge triggered. Therefore master first responds to J and K inputs and then slave. If J=0 and K=1, master resets on arrival of positive clock edge. High output of the master drives the K input of the slave. For the trailing edge of the clock pulse the slave is forced to reset. If both the inputs are high, it changes the state or toggles on the arrival of thepositive clock edge and the slave toggles on the negative clock edge. The slave does exactly what the master does.

Function Table:

Clk	J	K	Q	$\overline{Q}$	comment
⎍	0	0	Q_0	$\overline{Q_0}$	No change
⎍	0	1	0	1	Reset
⎍	1	0	1	0	Set
⎍	1	1	Q_0	Q_0	toggle

Circuit Diagram:



Procedure:

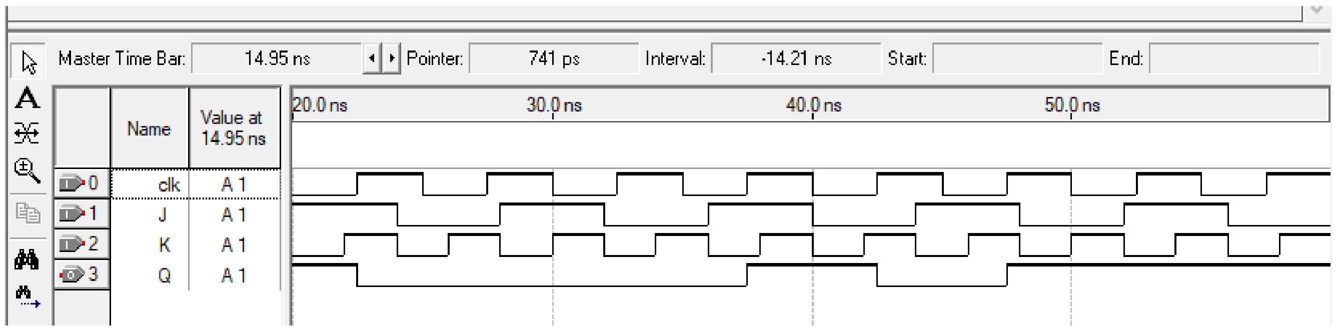
- Verify all components and patch chords whether they are in good condition are not.
- Make connection as shown in the circuit diagram.
- Give supply to the trainer kit.
- Verify the output with truth table.

Result:
Truth Table is verified

Simulation: VHDL code for JK master slave Flipflop

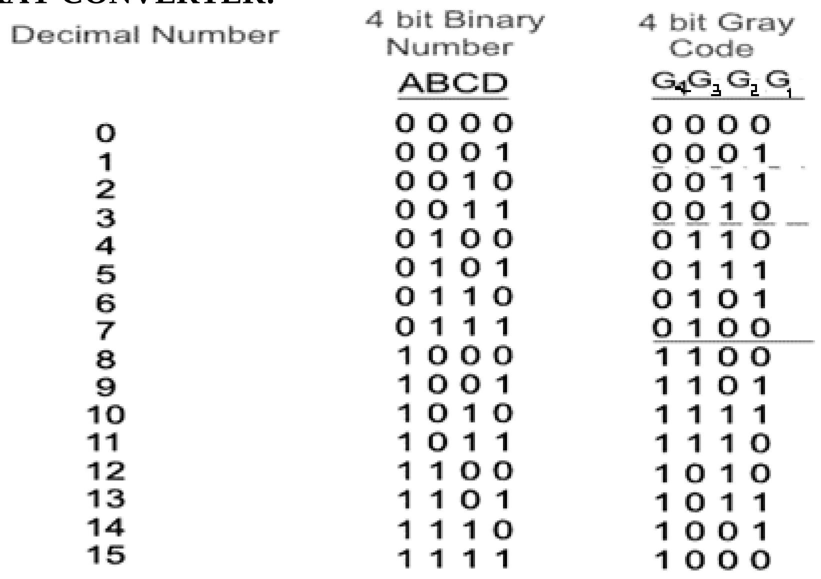
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity jkflip is
Port ( J, K, clk : in std_logic;
Q : buffer std_logic);
end jkflip;
architecture Behavioral of jkflip is
begin
process(clk)
begin
if rising_edge(clk) then
Q<= ((J and (not Q)) or ((not K) and Q));
end if;
end process;
end Behavioral;
```

Waveform:



7. Design and implement code converter I) Binary to Gray II) Gray to Binary Code using basic gates.

BINARY TO GRAY CONVERTER:



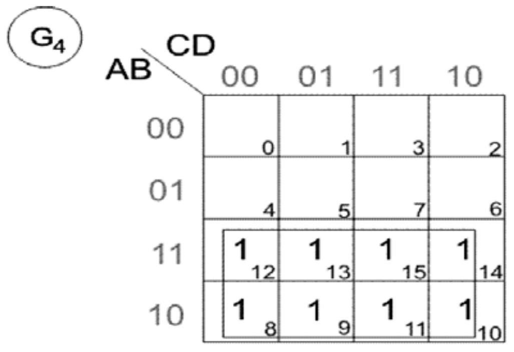
Decimal Number	4 bit Binary Number ABCD	4 bit Gray Code $G_4G_3G_2G_1$
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1
10	1 0 1 0	1 1 1 1
11	1 0 1 1	1 1 1 0
12	1 1 0 0	1 0 1 0
13	1 1 0 1	1 0 1 1
14	1 1 1 0	1 0 0 1
15	1 1 1 1	1 0 0 0

The logical circuit which converts binary code to equivalent gray code is known as binary to gray code converter. The gray code is a non weighted code. The successive gray code differs in one bit position only that means it is a unit distance code. It is also referred as cyclic code. It is not suitable for arithmetic operations. It is the most popular of the unit distance codes. It is also a reflective code. An n bit Gray code can be obtained by reflecting an n-1 bit code about an axis after 2^{n-1} rows, and putting the MSB of 0 above the axis and the MSB of 1 below the axis. Reflection of Gray codes is shown below. The 4 bits binary to gray code conversion table is given below,

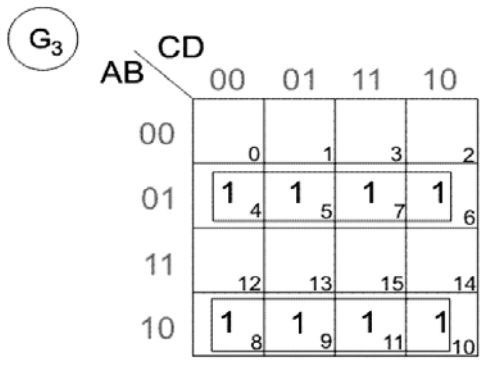
That means, in 4 bit gray code, (4-1) or 3 bit code is reflected against the axis drawn after $(2^{4-1})^{th}$ or 8th row. The bits of 4 bit gray code are considered as G4G3G2G1. Now from conversion table,

$$G4 = \sum m\,(8, 9, 10, 11, 12, 13, 14, 15)$$
$$G3 = \sum m\,(4, 5, 6, 7, 8, 9, 10, 11)$$
$$G2 = \sum m\,(2, 3, 4, 5, 10, 11, 12, 13)$$
$$G1 = \sum m\,(1, 2, 5, 6, 9, 10, 13, 14)$$

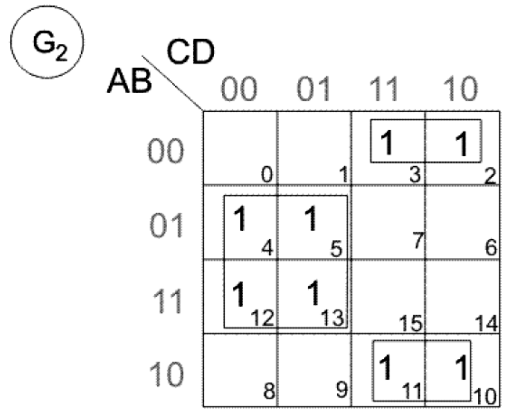
From above SOPs, let us draw K maps for G4, G3, G2 and G1.



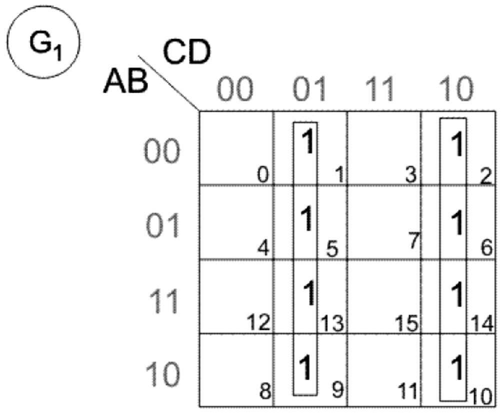
$G_4 = A$



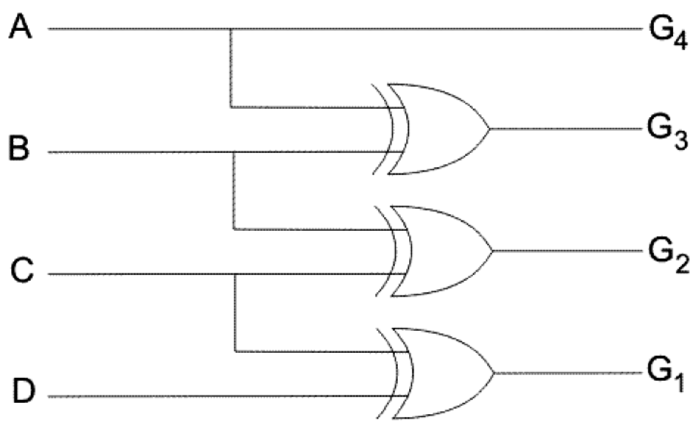
$G_3 = \overline{A}B + A\overline{B} = A \oplus B$

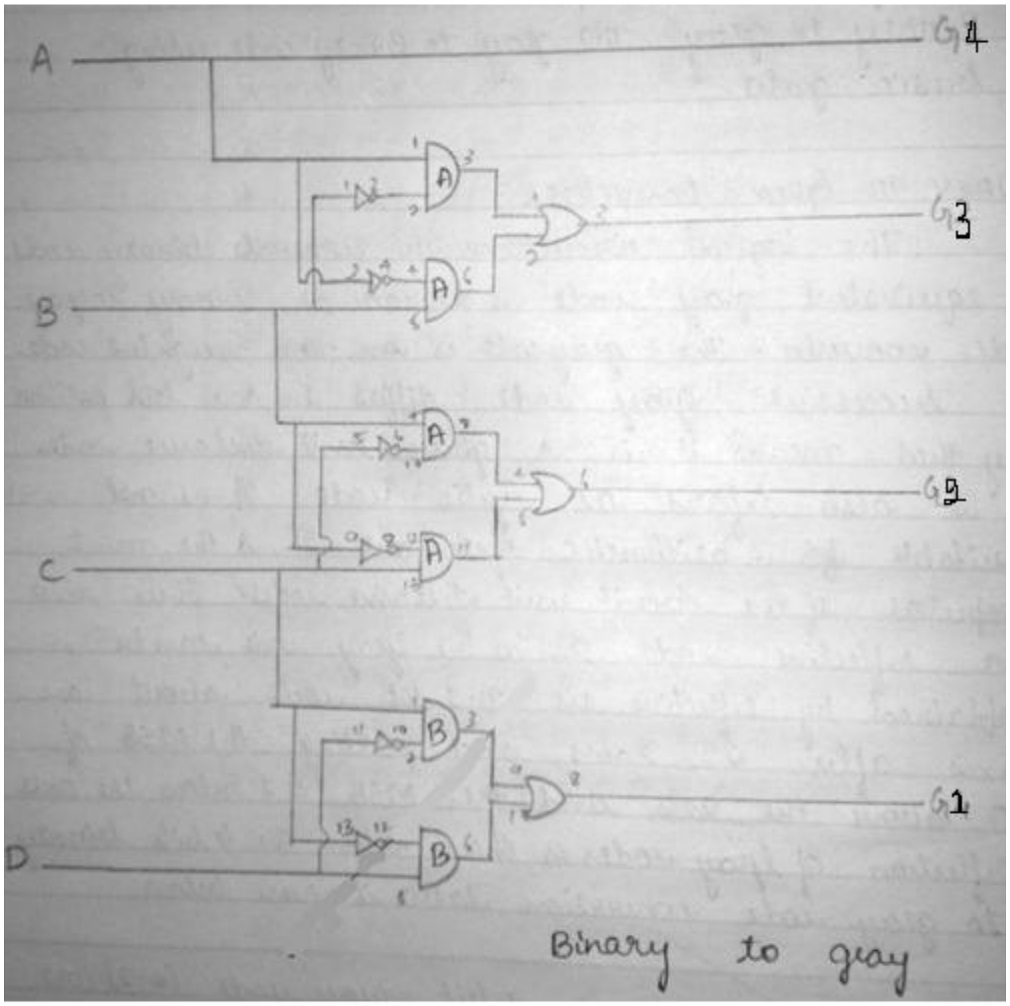


$G_2 = B\overline{C} + \overline{B}C = B \oplus C$



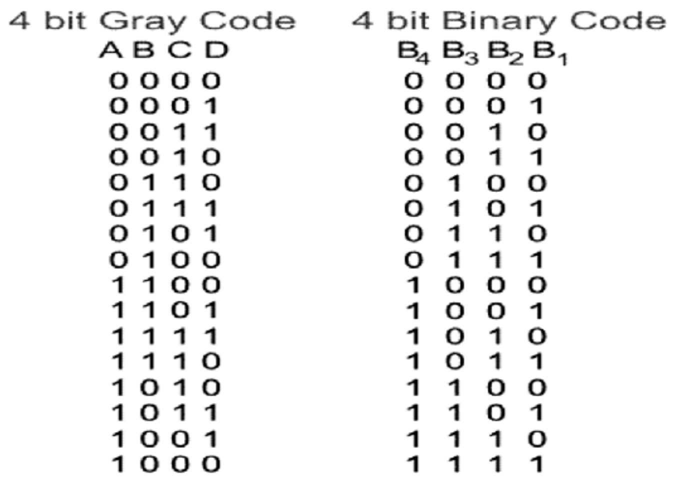
$G_1 = \overline{C}D + C\overline{D} = C \oplus D$



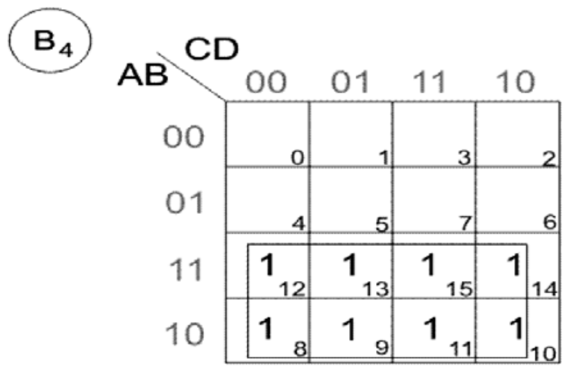


GREY TO BINARY CODE CONVERTER:

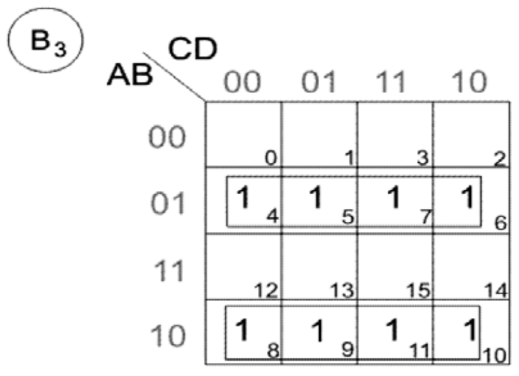
In gray to binary code converter, input is a multiplies gray code and output is its equivalent binary code. Let us consider a 4 bit gray to binary code converter. To design a 4 bit gray to binary code converter, we first have to draw a conversion table.



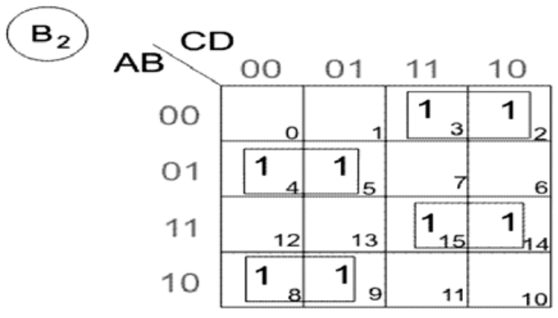
4 bit Gray Code	4 bit Binary Code
A B C D	B_4 B_3 B_2 B_1
0 0 0 0	0 0 0 0
0 0 0 1	0 0 0 1
0 0 1 1	0 0 1 0
0 0 1 0	0 0 1 1
0 1 1 0	0 1 0 0
0 1 1 1	0 1 0 1
0 1 0 1	0 1 1 0
0 1 0 0	0 1 1 1
1 1 0 0	1 0 0 0
1 1 0 1	1 0 0 1
1 1 1 1	1 0 1 0
1 1 1 0	1 0 1 1
1 0 1 0	1 1 0 0
1 0 1 1	1 1 0 1
1 0 0 1	1 1 1 0
1 0 0 0	1 1 1 1



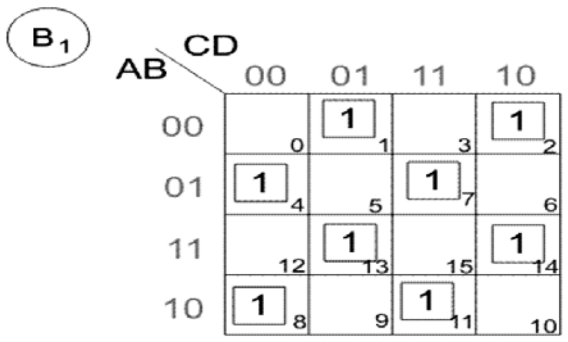
$B_4 = A$



$B_3 = \overline{A}B + A\overline{B} = A \oplus B$

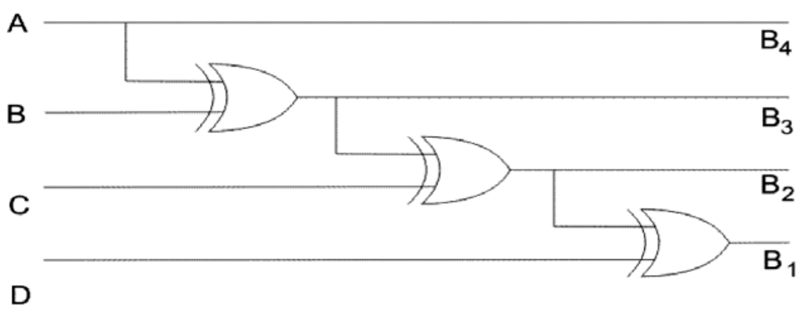


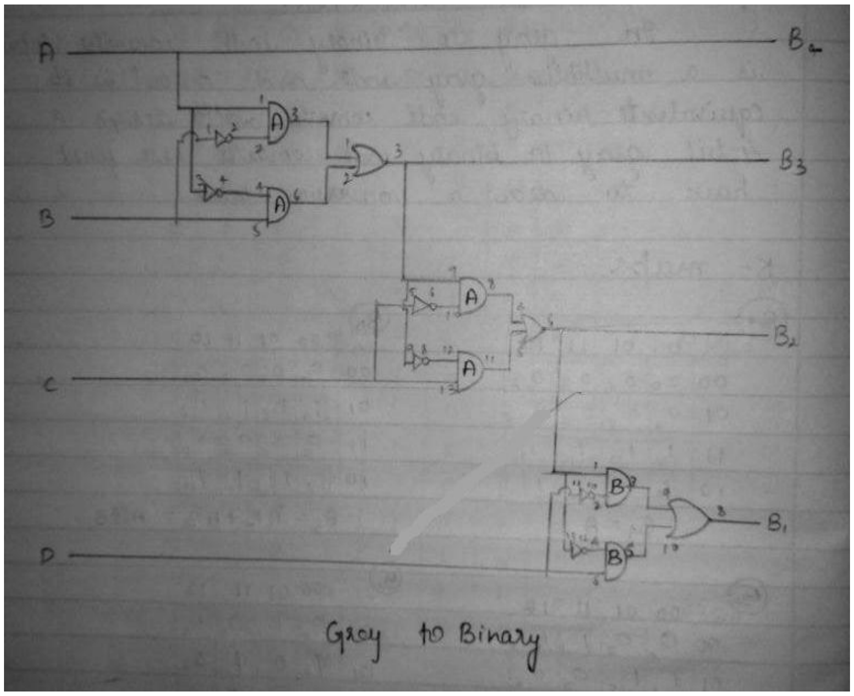
$$\begin{aligned}
B_2 &= \overline{A}B\overline{C} + A\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + ABC \\
&= A(\overline{B}\,\overline{C} + BC) + \overline{A}(B\overline{C} + \overline{B}C) \\
&= A(\overline{B\overline{C} + \overline{B}C}) + \overline{A}(B\overline{C} + \overline{B}C) \\
&= A(\overline{B \oplus C}) + \overline{A}(\overline{B \oplus C}) = A \oplus B \oplus C
\end{aligned}$$



$$\begin{aligned}
B_1 &= \overline{A}\,\overline{B}\,\overline{C}\,D + \overline{A}\,\overline{B}\,C\,\overline{D} + \overline{A}\,B\,\overline{C}\,\overline{D} + \overline{A}\,BCD + AB\overline{C}\,D + ABC\overline{D} + A\overline{B}\,\overline{C}\,\overline{D} \\
&+ A\overline{B}CD = A \oplus B \oplus C \oplus D
\end{aligned}$$

From above gray code we get,





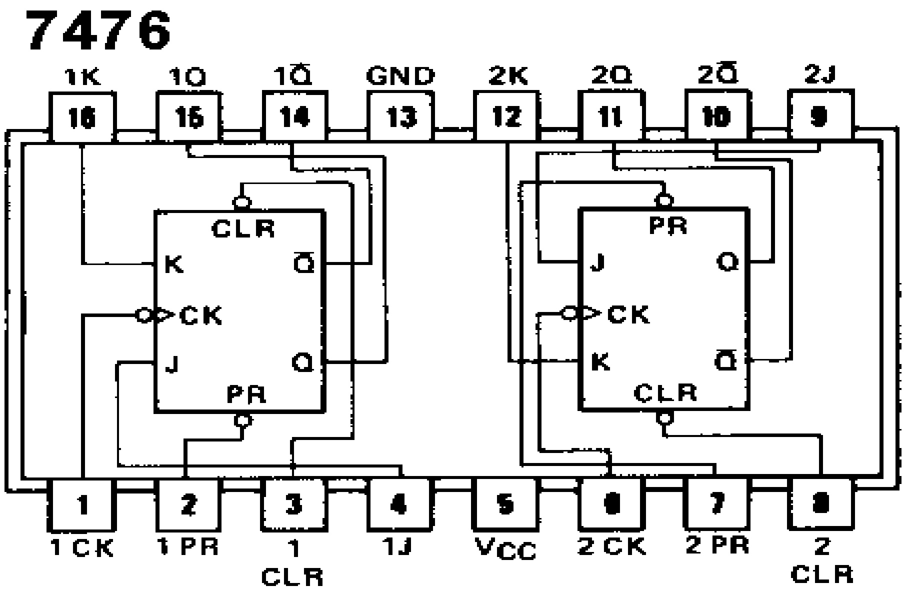
RESULT:

Truth Table is verified.

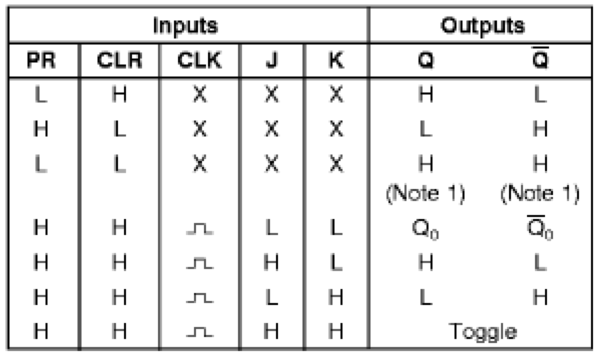
8. Design and implement a mod n (n<8) synchronous up counter using JK Flip Flop ICs and demonstrate its working.

Components used: IC 74LS76, IC 74LS08, Patch chords, power chords, and Trainer kit.

Pin diagram of 7476



Function Table



Inputs					Outputs	
PR	CLR	CLK	J	K	Q	Q̄
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H (Note 1)	H (Note 1)
H	H	⎍	L	L	Q_0	$\overline{Q}_0$
H	H	⎍	H	L	H	L
H	H	⎍	L	H	L	H
H	H	⎍	H	H	Toggle	

Qn	Qn+1	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Theory:

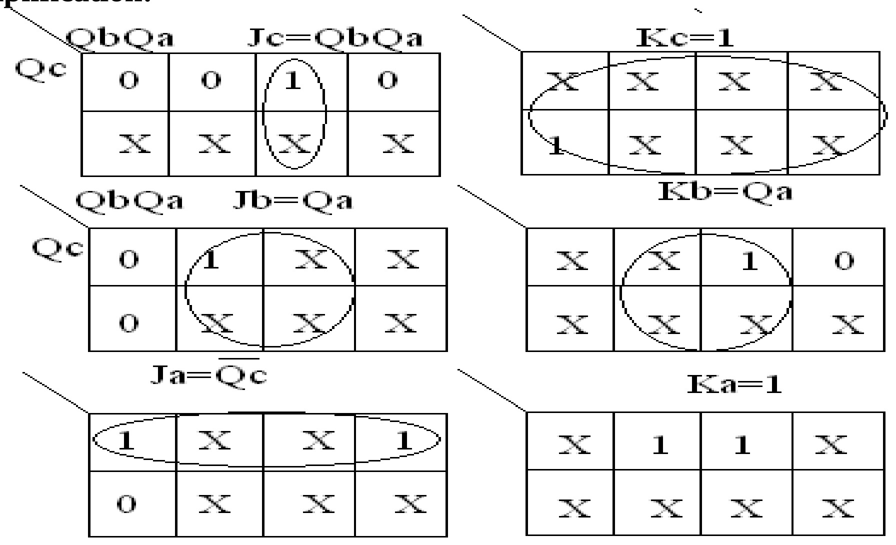
The ripple counter requires a finite amount of time for each flip flop to change state. This problem can be solved by using a synchronous parallel counter where every flip flop is triggered in synchronism with the clock, and all the output which are scheduled to change do so simultaneously. The counter progresses counting upwards in a natural binary sequence from count 000 to count 100 advancing count with every negative clock transition and get back to 000 after this cycle.

Designing:

a)Transition Table: Mod 5 (0 – 4)

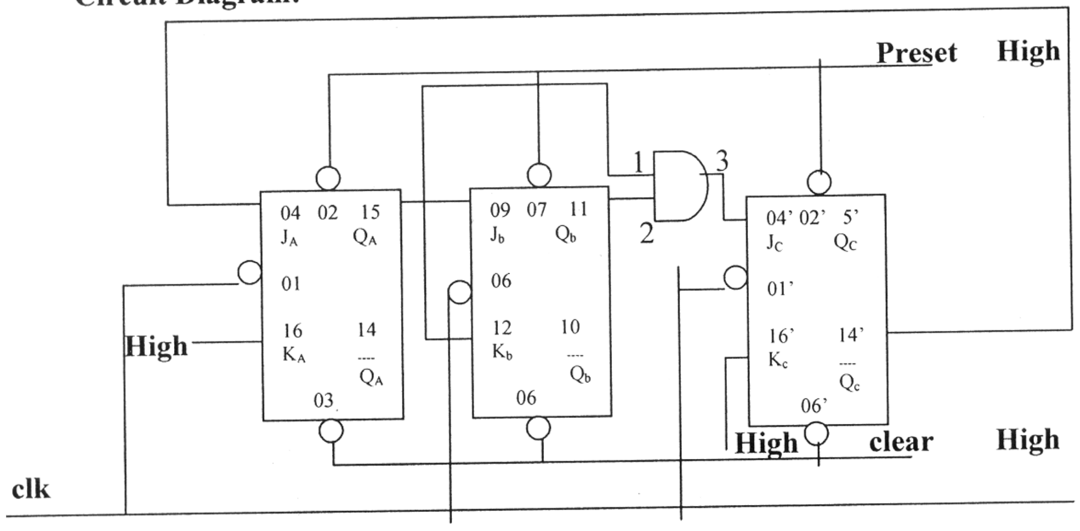
Present State			Nert State								
Qc	Qb	Qa	Qc+1	Qb+1	Qa+1	Jc	Kc	Jb	Kb	Ja	Ka
0	0	0	0	0	1	0	x	0	x	1	x
0	0	1	0	1	0	0	x	1	x	x	1
0	1	0	0	1	1	0	x	x	0	1	x
0	1	1	1	0	0	1	x	x	1	x	1
1	0	0	0	0	0	x	1	0	x	0	x
Not used			So filled with don't			care		values			
1	0	1	x	x	x	x	x	x	x	x	x
1	1	0	x	x	x	x	x	x	x	x	x
1	1	1	x	x	x	x	x	x	x	x	x

b)K-map Simplification:



c) Diagram & implementation

Circuit Diagram:



Procedure:

(1) Verify all components and patch chords whether they are in good condition or not.
(2) Make connection as shown in the circuit diagram.
(3) Give supply to the trainer kit.
(4) Provide input data to circuit via switches.
(5) Verify truth table sequence and observe outputs.

Result:

Truth Table is verified

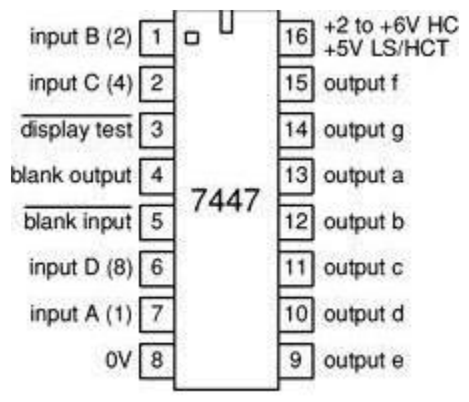
9. **Design and implement asynchronous counter using decade counter IC to count up from 0 to n (n≤9) and demonstrate on seven segment display (using IC-7447).**

Components Required: ICs used: 7490,7447, 507

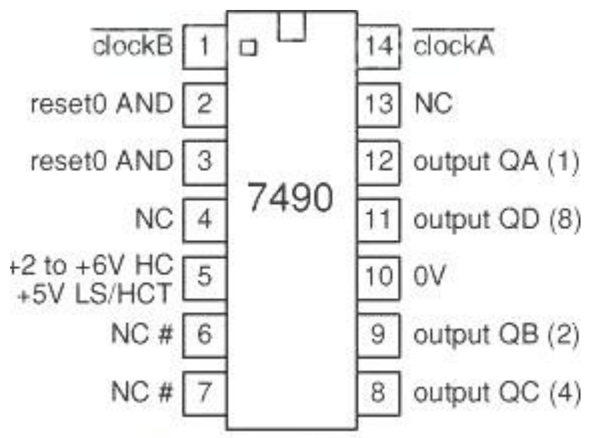
Pin Details of the ICs: PIN diagram of 7490,7447,FND 507

Description:

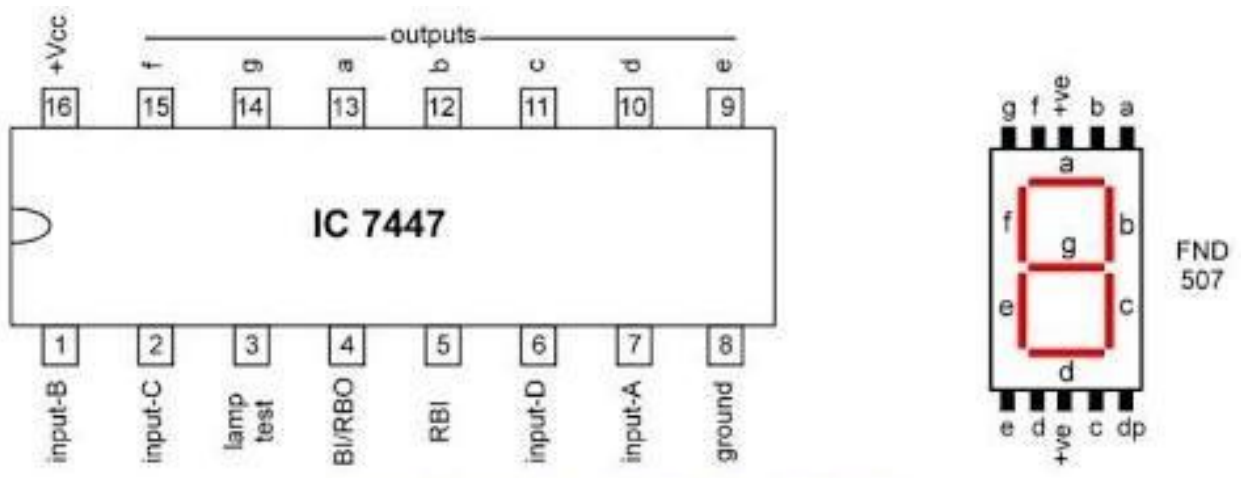
Asynchronous counter is a counter in which the clock signal is connected to the clock input of only first stage flip flop. The clock input of the second stage flip flop is triggered by the output of the first stage flip flop and so on. This introduces an inherent propagation delay time through a flip flop. A transition of input clock pulse and a transition of the output of a flip flop can never occur exactly at the same time. Therefore, the two flip flops are never simultaneously triggered, which results in asynchronous counter operation.



Pin diagram of 7447

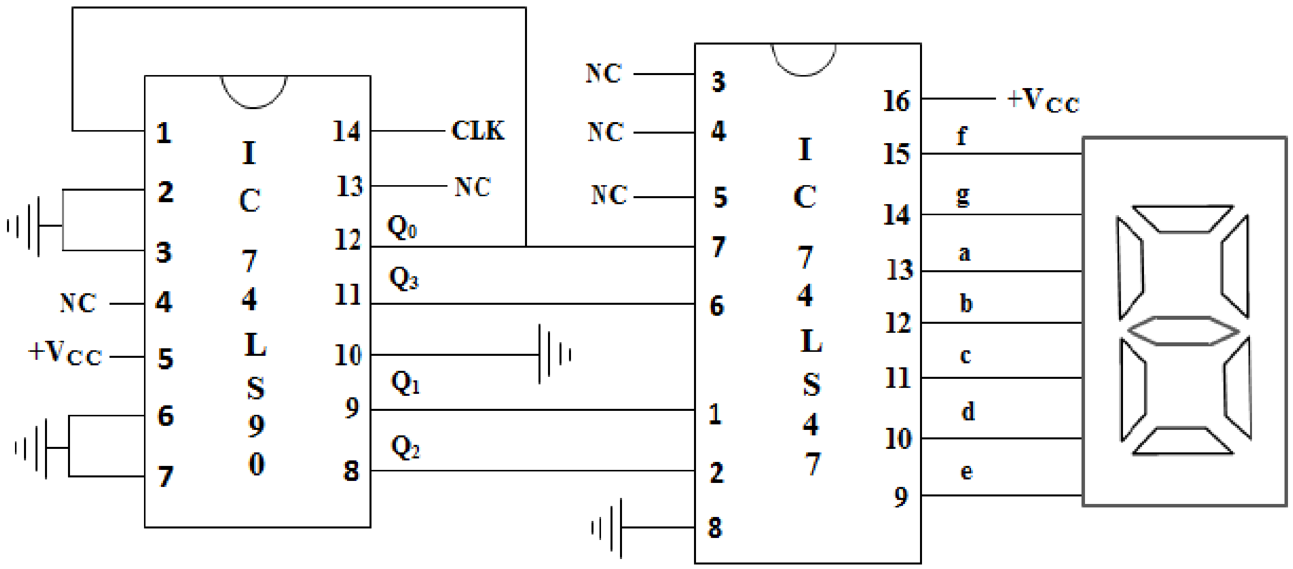


Pin diagram of 7490



Pin configuration of IC 7447 and FND 507

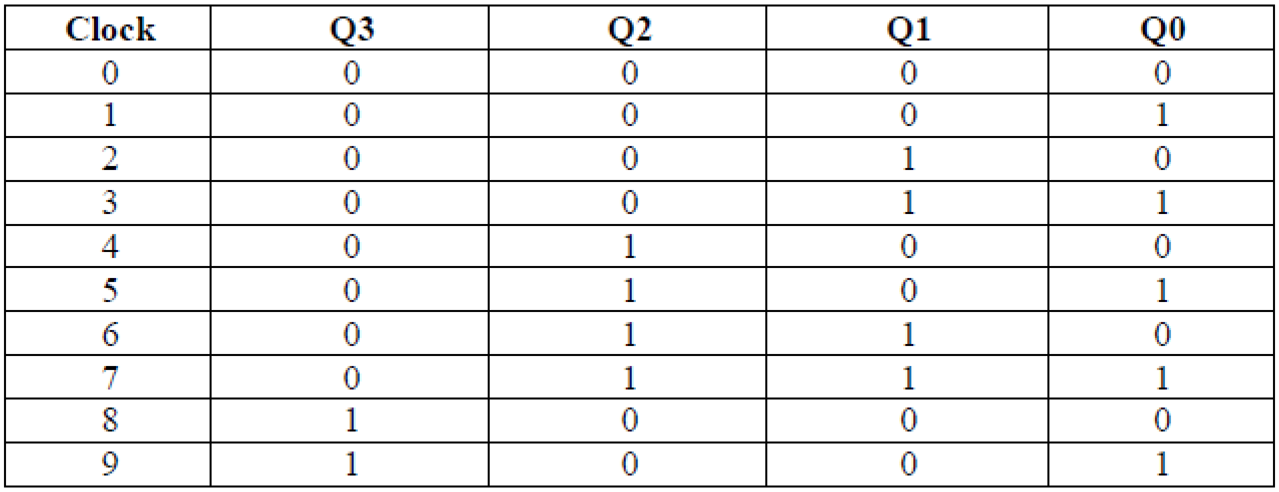
Circuit Diagram



For mod 9
connect Q0 and Q3 to reset(clear) through an AND gate. Reset should not be connected to the switch
For mod8
Connect Q3 to reset
For mod7
Connect Q2, Q1,Q0 to reset through an And Gate
For Mod 6
Connect Q2 and Q1 to reset through an AND gate
For mod 5
Connect Q0 and Q2 to reset through an AND gate
For Mod 4
Connect Q2 to reset
For mod 3
Connect Q1 and Q0 to reset through an AND gate
For mod 2
Connect Q1 to reset

Procedure:
1. Verify all components & patch chords whether they are in good condition or not.
2. Make connections as shown in the circuit diagram.
3. Give supply to the trainer kit.
4. Provide input data to circuit via switches.
5. Verify truth table sequence & observe outputs.

Function Table:



Clock	Q3	Q2	Q1	Q0
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

Result:
mod n<=9 counter implemented using the decade counter IC

Analog Electronics Software Experiments with PSPICE

The steps to simulation

1. Create a simulation project
2. Draw schematic to simulate
3. Establish a simulation profile
4. Set up simulation type
5. Simulate circuit
6. Analyze results in Probe

General Procedure to use for all simulation experiments:

Start → Programs → Orcad family lite edition → Caputre lite edition → File

1) Select new & blank project, select analog & mixed mode for new simulation (use open project for already existing project).
2) Layout appears select components from parts & place at required position on layout, connect components using wire, apply voltage marker at i/p & o/p to see wave forms.
3) If components are not available it is required to add by using add library present in window.
4) From menu bar select PSpice, new simulation profile to Set simulation profile like
   Select Analysis type: Time domain (or) AC sweep as per experiment.
   Set Start value, Step size & End valu then Save settings.
5) Run simulation observe & note the output waveform.
6) Use Edit profile for any changes required in profile.
7) Simulation can be done for different values of component & supply.

Digital Electronics Software Experiments with Xilinx

Software package used for Digital experiment Xilinx 6.1 with Modelsim 9.2 , It is one of most popular software tool used to synthesize VHDL code. This tool Includes many steps. To make user feel comfortable with the tool the steps are given below:-

These steps are common to all Digital experiment for simulation & synthesis part.

- Double click on Project navigator. (Assumed icon is present on desktop).
- Select **NEW PROJECT** in **FILE MENU.**
  Enter following details as per your convenience

Project name	: new
Project location	: C:\create your folder
Top level module	: HDL

- In **NEW PROJECT** dropdown Dialog box, Choose your appropriate device specification.
  Example is given below:

Device family	: Spartan2
Device	: xc2s200
Package	: PQ208
TOP Level Module	: HDL
Synthesis Tool	: XST
Simulation	: Modelsim / others
Generate sim lang	: VHDL

- In source window right click on specification, select new source
  Enter the following details
  Entity: sample
  Architecture : **Behavioral**
  Enter the input and output port and modes.
  This will create **sample.VHD** source file. Click Next and finish the initial Project preparation.

- Double click on synthesis. If error occurs edit and correct VHDL code.
- Top level module appears, again select new source & set test bench waveform.
- Double click on Lunch modelsim (or any equivalent simulator if you are using) for functional simulation of your design.

Sample Viva Questions

1. Why operational amplifier is called by its name?
2. Explain the advantages of OPAMP over transistor amplifiers.
3. List the OPAMP ideal characteristics.
4. Give the symbol of OPAMP
5. Explain the various applications of OPAMP
6. Define UTP and LTP
7. Mention the applications of schmitt trigger
8. What is a square wave generator/ Regenerative comparator?
9. Give the hysterisis curve of a schmitt trigger
10. What is a bipolar and unipolar devices? Give examples
11. Define resolution
12. Explain the need of D/A and A/D converters.
13. List the different types of A/D and D/ A converters
14. What is a multivibrators?
15. What is a bistable multivibrators?
16. Give the applications of monostable and astable multivibrators
17. Explain the working of 555 timer as astable and monostable multivibrator
18. Why astable multivibrator is called as free running multivibrato
19. Define duty cycle.
20. List the applications of 555 timer
21. Explain 555 timer as astable multivibrator to generate a rectangular wave of duty cycle of less than 0.5
22. Define a logic gate.
23. What are basic gates?
24. Why NAND and NOR gates are called as universal gates?
25. State De morgans theorem
26. Give examples for SOP and POS
27. Explain how transistor can be used as NOT gate
28. Realize logic gates using NAND and NOR gates only
29. List the applications of EX-OR and EX~NOR gates
30. What is a half adder?
31. What is a full adder?
32. Differentiate between combinational and sequential circuits. Give examples
33. Give the applications of combinational and sequential circuits
34. Define flip flop
35. What is an excitation table?
36. What is race around condition?
37. How do you eliminate race around condition?
38. What is minterm an d max term?
39. Define multiplexer/ data selector
40. What is a demultiplexer?
41. Give the applications of mux and demux
42. What is a encoder and decoder?
43. Compare mux and encoder

44. Compare demux and decoder
45. What is a priority encoder?
46. What are counters? Give their applications.
47. Compare synchronous and asynchronous counters
48. What is modulus of a number?
49. What is a shift register?
50. What does LS stand for, in 74LSOO?
51. What is positive logic and negative logic?
52. What are code converters?
53. What is the necessity of code conversions?
54. What is gray code?
55. Realize the Boolean expressions for
    a Binary to gray code conversion
    b Gray to binary code conversion

Note:
All the above questions are the most commonly asked and the depth of it may vary based on the answers which you give during the viva voice procedure.

All the very best!