CANARA ENGINEERING COLLEGE

BENJANAPADAVU – 574219

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

ANALOG & DIGITAL ELECTRONICS LABORATORY

(18CSL37)

MANUAL

(Effective from the academic year 2018 -2019)

SEMESTER - III

Prepared by:

Mrs. Anupama V
Assistant Professor
Department of Computer Science & Engineering
Canara Engineering College

Mr. Dhananjaya B
Assistant Professor
Department of Computer Science & Engineering
Canara Engineering College

VISION

To be recognized as a center of knowledge dissemination in Computer Science and engineering by imparting value-added education to transform budding minds into competent computer professionals.

MISSION

- Provide a learning environment enriched with ethics that helps in enhancing problem solving skills of students and, cater to the needs of the society and industry.
- Expose the students to cutting-edge technologies and state-of-the-art tools in the many areas of Computer Science & Engineering.
- Create opportunities for all round development of students through co-curricular and extra-curricular activities.
- Promote research, innovation and development activities among staff and students.

DEPARTMENT PROFILE

The department was started in the year 2001 to offer under graduate degree programme i.e. **Bachelor of Engineering (BE) in Computer Science & Engineering (CS&E).** The department also offers post graduate programme in engineering i.e. **MTech in Computer Science Engineering (CS&E)** since 2011. The department has dedicated, qualified and experienced faculty members to guide the students in academics. The faculty members are actively involved in teaching, product development and research. The faculty members have published number of research and review papers/articles in referred International journals and reputed International conferences which are archived at IEEE/ACM/Springer and other renowned digital libraries.

The department frequently organizes training programmes for the faculty, technical staff and students. The faculty frequently attends staff development programmes (SDP/FDP/Seminar) to update themselves in technological advancements and conferences to present research findings. The department aims at building the students' career by placing special emphasis on all-round development through continuous interaction with Industry. Interactive sessions with experts from academia, research laboratories and industry are constantly held so as to enable students to gain knowledge on diverse and emerging fields. The campus placement has been scaling higher and higher peaks right from its inception with multinational companies recruiting students in large numbers. To increase the opportunity of placements to students, the department conducts soft skills training programmes, technical skill development activities and initiatives on self-learning (Spoken Tutorial programmes by IIT Bombay).

The department promotes extracurricular activities under the umbrella of the students' association & SPECS. The department brings out Annual technical magazine and newsletter which provides an opportunity for the students and staff to publish innovative ideas, programming tips and articles on current trends in computing and technology. The students 'association and National Service Scheme (NSS) wing frequently conducts various programmes to strengthen leadership skills, teamwork and communication; and awareness on protection of environment and social responsibilities.

PROGRAMME EDUCATIONAL OBJECTIVES (PEO)

1. Graduates will work productively as computer science engineers exhibiting ethical qualities and leadership roles in multi-disciplinary teams.
2. Graduates will adapt to the changing technologies, tools and societal requirements.
3. Graduates will design and deploy software that meets the needs of individuals and the industries
4. Graduates will take up higher education and/or be associated with the field so that they can keep themselves abreast of Research & Development

PROGRAMME OUTCOMES (PO)

Engineering graduates in Computer Science and Engineering will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods, including design of experiments, analysis and interpretation of data and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Select/Create and apply appropriate techniques, resources and modern engineering and IT tools, including prediction and modelling to complex engineering activities, taking comprehensive cognizance of their limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and Sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts and demonstrate the knowledge of and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the relevant scientific and/or engineering practices.
9. **Individual and team work:** Function effectively as an individual and as a member or leader in diverse teams and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with the society-at-large, such as being able to comprehend and write effective reports and design documentation, make effective presentations and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work as a member and leader in a team to manage projects and in multidisciplinary environments.

COURSE OBJECTIVES:

1	Develop, design, observe and analyze simple analog and digital electronic circuits.
2	Acquire experience in building and troubleshooting simple analog and digital electronic circuits.
3	Explain and analyze simple analog and digital electronic circuits.
4	Be able to present developed simple analog and digital electronic circuits.
5	Have practical knowledge on different digital electronic devices and instruments.

COURSE OUTCOMES (COs):

SL. NO	DESCRIPTION	PO MAPPING	PSO MAPPING
CO:1	Conduct experiments to know the working of Analog and digital electronic components and demonstrate its working.	1, 3, 5	1
CO:2	Troubleshooting simple analog & digital electronic circuits and tabulation of experimental data and conclusion.	1, 3, 5	1
CO:3	Express the theoretical aspects used in conducting different experiments of Analog & Digital Electronics.	1, 10	1
CO:4	Record Experimental process and results.	1, 10	1
CO:5	Conduct any given experiment in Analog & Digital Electronics laboratory and communicate results effectively.	1, 3, 5, 10	1

COURSE ARTICULATION MATRIX:

Course code	Name of the Course	PO Mapped	PSO Mapped
17CSL37	Analog & Digital Electronics lab	PO1-5, PO3-3, PO5-3, PO10-3	PSO1-5

Sl. No.	Course Outcomes	Mapping				
		PO				PSO
		1	3	5	10	1
1	Conduct experiments to know the working of Analog and digital electronic components and demonstrate its working.	2	3	3		2
2	Troubleshooting simple analog & digital electronic circuits and tabulation of experimental data and conclusion.	2	3	3		2
3	Express the theoretical aspects used in conducting different experiments of Analog & Digital Electronics.	2			2	2
4	Record Experimental process and results.	2			2	2
5	Conduct any given experiment in Analog & Digital Electronics laboratory and communicate results effectively.	2	3	3	2	2
Average COs mapping to POs / PSOs		**2**	**3**	**3**	**2**	**2**

1– Course Outcome addresses program outcome 1–slightly, 2– Moderate, 3 – Substantial

ANALOG AND DIGITAL ELECTRONICS LABORATORY
(Effective from the academic year 2018 -2019)
SEMESTER – III

Course Code	**18CSL37**	**CIE Marks**	40
Number of Contact Hours/Week	0:2:2	**SEE Marks**	60
Total Number of Lab Contact Hours	36	**Exam Hours**	03

Credits – 2

Course Learning Objectives: This course (18CSL37) will enable students to:

This laboratory course enable 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

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)

Laboratory 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.

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)*
  a) For laboratories having only one part – Procedure + Execution + Viva-Voce: 15+70+15 = 100 Marks
  b) For laboratories having PART A and PART B
     i. Part A – Procedure + Execution + Viva = 6 + 28 + 6 = 40 Marks
     ii. Part B – Procedure + Execution + Viva = 9 + 42 + 9 = 60 Marks

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.

AIM:

1) To design and implement an astable multivibrator using 555 Timer for 50%, <50%, >50% duty cycle.
2) To simulate an astable multivibrator using 555 Timer any one duty cycle using PSpice.

EQUIPMENT REQUIRED:

Sl. No	Components	Quantity
1	555 Timer IC	1
2	3.3KΩ Resistor	1
3	6.8KΩ Resistor	1
4	0.1uF Capacitor	1
5	0.01uF Capacitor	1
6	DC regulated power supply	1
7	Signal generator	1
8	CRO	1
9	Wires	As required

THEORY: Multivibrator is a form of oscillator, which has a non-sinusoidal output. The output waveform is rectangular.

The multivibrators are classified as: 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.

Monostable or one shot multivibrator: It has one stable state and one quasi stable. The application of an input pulse triggers the circuit time constants. 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.

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.

DESIGN:

The time period T =1/f = 1ms = t_H + t_L

Where t_H is the time the output is high and t_L is the time the output is low.

From the theory of astable multivibrator using 555 Timer), we have

$$t_L = 0.693\ R_B\ C$$

$$R_B = \frac{t_L}{0.693 \times C} \quad \text{----------(1)}$$

$$t_H = 0.693\ (R_A + R_B)C$$

$$R_A = \frac{t_H}{0.693 \times C} - R_B \quad \text{----------(2)}$$

Case 1: frequency (f) = 1 KHz, duty cycle D= 75% = 0.75

Therefore T=1/f=1ms= t_H + t_L =1ms

Duty cycle = t_H / T = 0.75

t_H = 0.75T = 0.75ms

$t_L = T - t_H = 0.25ms$.

Let C=0.1µF and substituting in the equations (1) and (2),

$R_B = \frac{0.25x10^{-3}}{0.693x0.1x10^{-6}}$

$R_B = 3.6k\Omega \backsimeq 3.3k\Omega$

$R_A = \frac{0.75x10^{-3}}{0.693x0.1x10^{-6}} - 3.6x10^3$

$R_A = 7.2k\Omega \backsimeq 6.8k\Omega$

Case 2: frequency (f) = 1 KHz duty cycle D = 40% = 0.4

Therefore T=1/f=1ms= $t_H + t_L$ =1ms

Duty cycle = t_H / T = 0.4

$t_H = 0.4T = 0.4ms$

$t_L = T - 0.4T = 0.4T = 0.6ms$

Let C=0.1µF and substituting in the equations (1) and (2),

$0.6x10^{-3} = 0.693R_B \times 0.1x10^{-6}$

$R_B = 8.66k\Omega$

$R_B = \frac{0.4x10^{-3}}{0.693x0.1x10^{-6}}$

$R_B = 5.7k\Omega$

$R_A = \frac{0.6x10^{-3}}{0.693x0.1x10^{-6}} - 5.7x10^3$

$R_A = 8.6k\Omega$

Case 3: frequency (f) = 1 KHz, duty cycle D = 50% = 0.5

Therefore T=1/f=1ms= $t_H + t_L$ =1ms

Duty cycle = t_H / T = 0.5

Hence $t_H = 0.5T = 0.5ms$

$t_L = T - 0.5T = 0.5T = 0.5ms$

Let C=0.1µF and substituting in the equations (1) and (2),

$R_B = \frac{0.5x10^{-3}}{0.693x0.1x10^{-6}}$

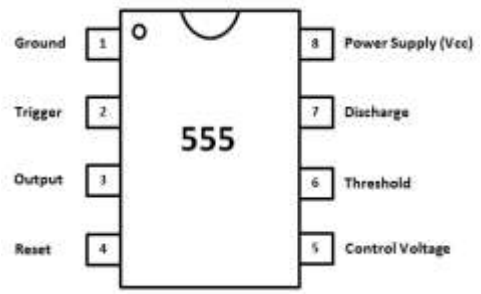
$R_B = 7.2k\Omega$

$R_A = \frac{0.5x10^{-3}}{0.693x0.1x10^{-6}}$

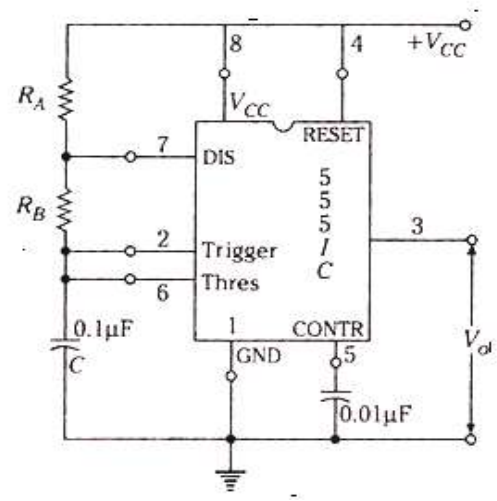
$R_A = 7.2k\Omega$

BLOCK / CIRCUIT DIAGRAM:

555 Timer Pin Diagram:



Circuit Diagram for Astable Multivibrator:



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.

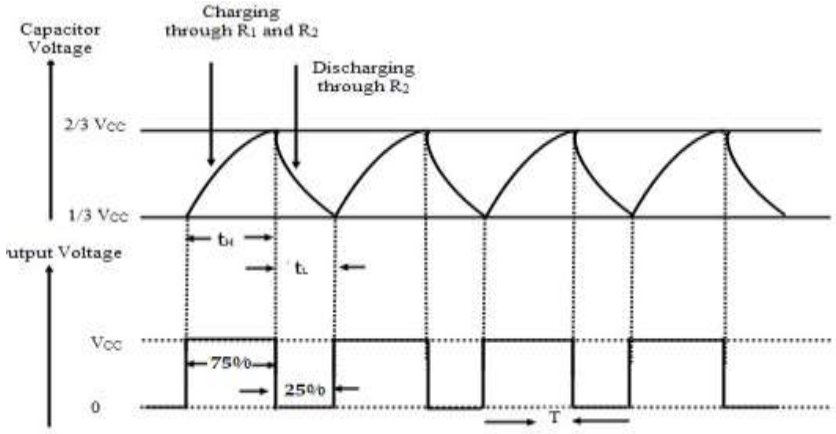
3. Observe the capacitor voltage waveform at 6th pin of 555 timer on CRO.

4. Observe the output waveform at 3rd pin of 555 timer on CRO (shown below).

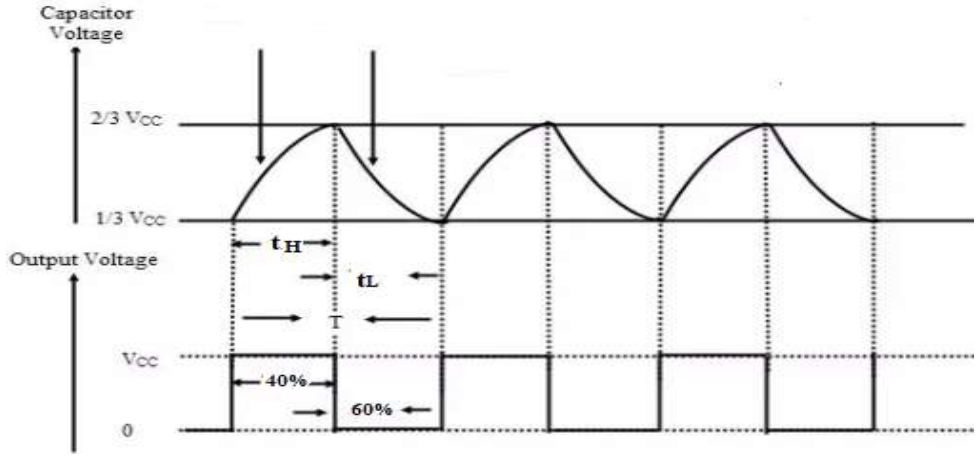
5. Note down the amplitude levels, time period and hence calculate duty cycle.

INPUT/OUTPUT WAVEFORMS

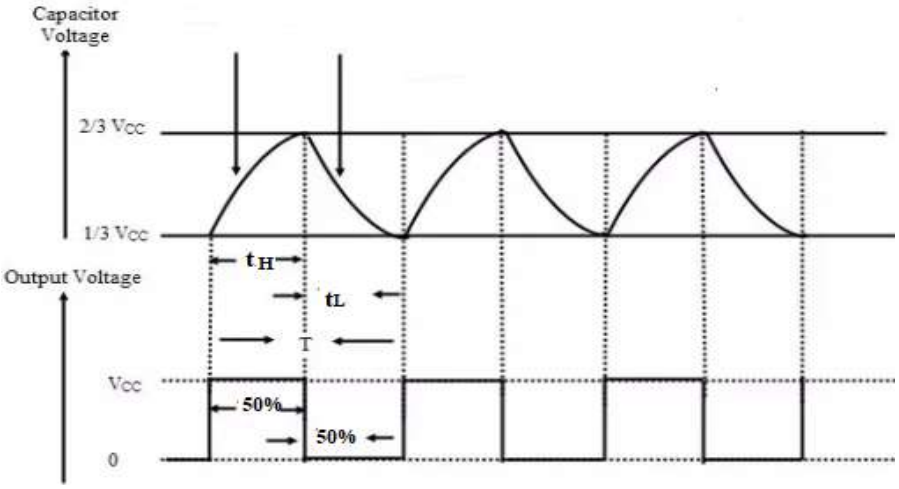
Case 1:



Case 2:



Case 3:

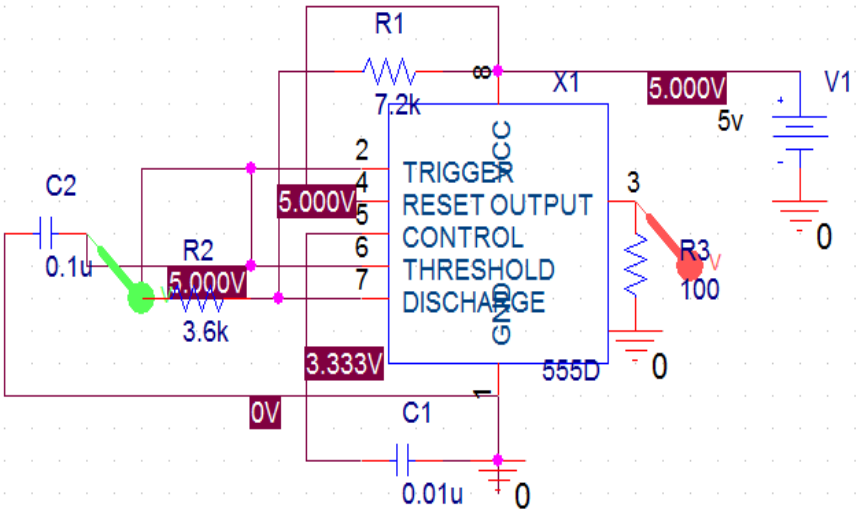


SIMULATION:

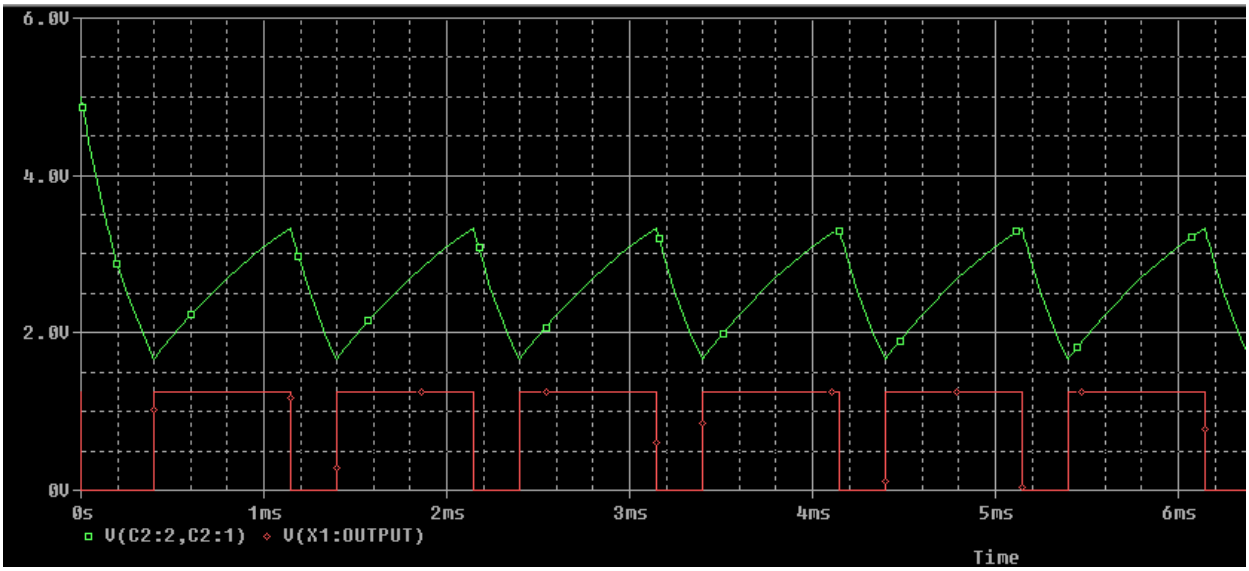
- Start Orcad and create new project by clicking File → New → Project → Enter the Project Name and select Analog or Mixed A/D
- By clicking Place → Part, select the following components

Part	Library	Quantity
555 D	Eval	1
Resistors	Analog	3
Capacitor	Analog	2
V_{DC}	Source	1
0/CAPSYM	Place Ground	3

- Connect all the components using Wires.
- Set the values of the component.
- Place the probes at the input and output terminal of the circuit.
- Then click on Save
- Create a new simulation profile with Analysis type: Time domain (Transient), Run to Time: 10ms
- Save and Run the simulation.
- Observe the waveform and note down.



SIMULATION WAVEFORM:



RESULT & CONCLUSION:

1) An astable multivibrator of given duty cycle and frequency is designed and implemented.
2) An astable multivibrator of given duty cycle and frequency is simulated using PSpice.

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

AIM:

1) To design and implement an op-amp relaxation oscillator for 1 kHz frequency 50% duty cycle
2) To simulate an op-amp relaxation oscillator for 1 kHz frequency 50% duty cycle using PSpice.

EQUIPMENT REQUIRED:

Sl. No	Components	Quantity
1	IC µA741	1
2	5KΩ Resistor	1
3	10KΩ Resistor	1
4	Potentiometer	1
5	0.1uF Capacitor	1
6	DC regulated power supply	1
7	Signal generator	1
8	CRO	1
9	Wires	As required

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 (R1/ (R1+R2)) of output is fed back to the non inverting input terminal. Thus reference voltage is (R1/ (R1+R2)) Vo. And may take values as + (R1/ (R1+R2)) Vsat or - (R1/ (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 which resulting in a square wave output.

DESIGN:

The period of the output rectangular wave given as

$$T = 2RC \ln\left(\frac{1+\beta}{1-\beta}\right) \quad \text{----------(1)}$$

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

If $R_2 = 1.16R_1$ then from equation (1) we have T = 2RC ----------(2)

Case 1: Design for a frequency of 1 kHz (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)

Let C=0.1μF

Then $R = \frac{T}{2C} = \frac{10^{-3}}{2 \times 10^{-7}} = 5K\Omega$

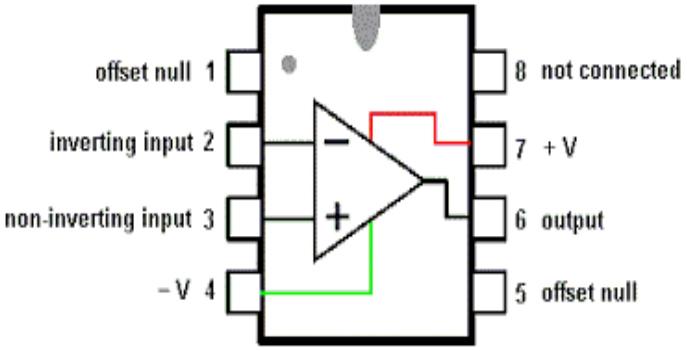
The voltage across the capacitor has a peak voltage of $V_c = \frac{R_1}{R_1 + R_2} V_{sat}$

PROCEDURE:

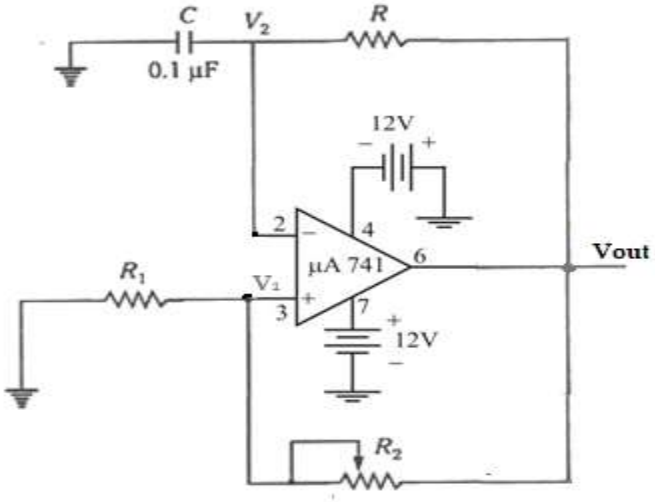
1. Before making the connections check all the components using multimeter.
2. Make the connections as shown in figure and switch on the power supply.
3. Observe the voltage waveform across the capacitor on CRO.
4. Also observe the output waveform on CRO. Measure its amplitude and frequency.

BLOCK / CIRCUIT DIAGRAM:

ICμA741 Pin Diagram:

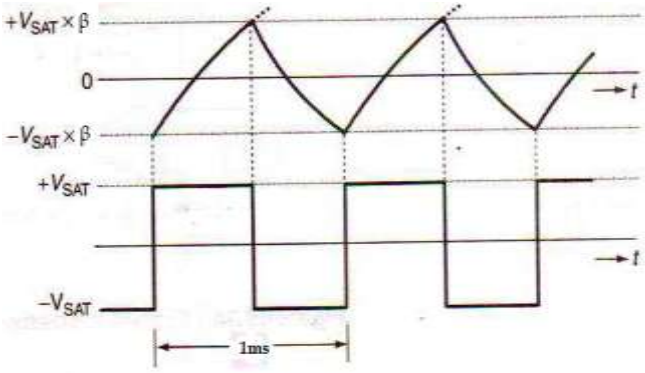


Circuit Diagram for Relaxation Oscillator:



INPUT/OUTPUT WAVEFORMS

Case 1: f=1kHz

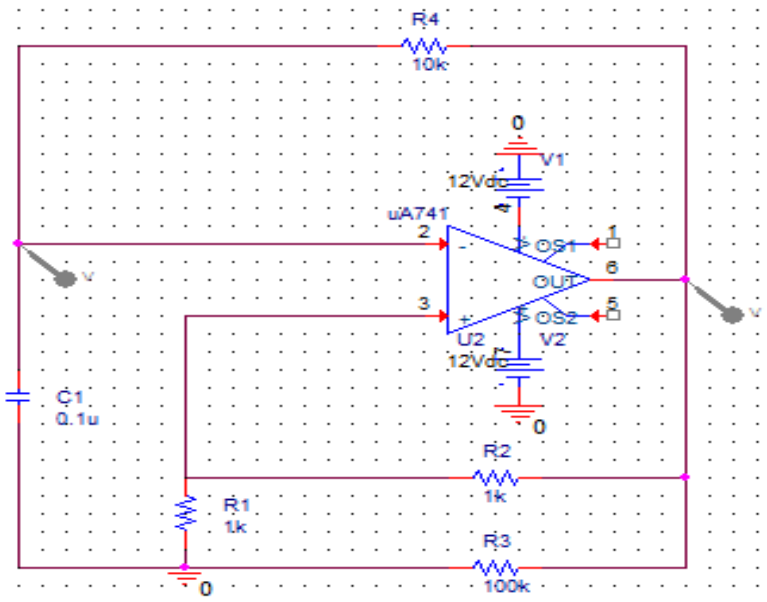


SIMULATION:

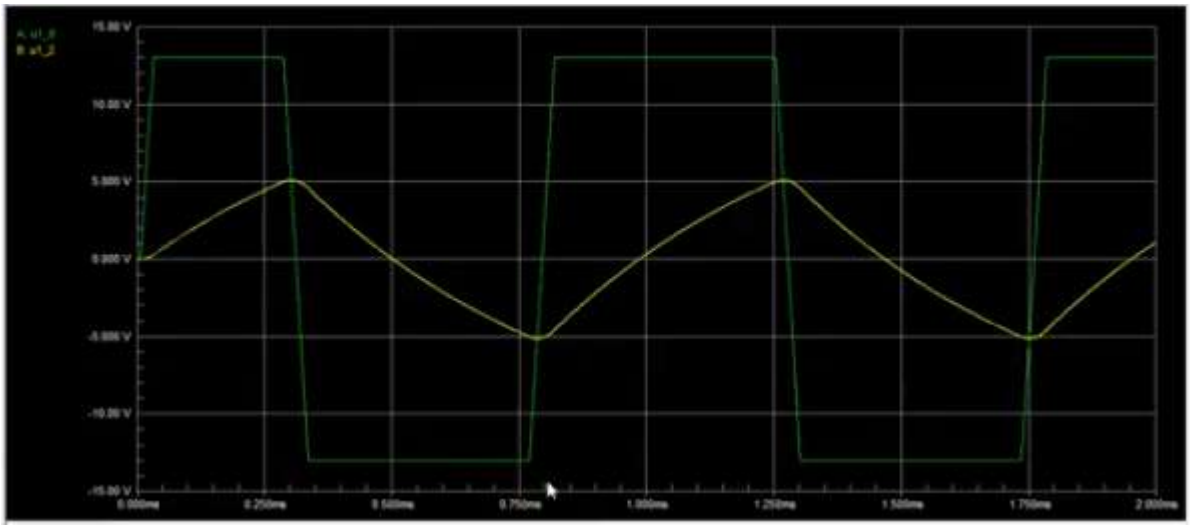
- Start Orcad and create new project by clicking File → New → Project → Enter the Project Name and select Analog or Mixed A/D
- By clicking Place → Part, select the following components

Part	Library	Quantity
µA741	Eval	1
Resistors	Analog	3
Capacitor	Analog	1
V_{SIN}	Source	1
V_{DC}	Source	2
0/CAPSYM	Place Ground	3

- Connect all the components using Wires.
- Set the values of the component.
- Place the probes at the input and output terminal of the circuit.
- Then click on Save
- Create a new simulation profile with Analysis type: Time domain (Transient), Run to Time: 100ms
- Save and Run the simulation.
- Observe the waveform and note down.



SIMULATION WAVEFORM:



RESULT & CONCLUSION:

1) An op-amp relaxation oscillator with 1 kHz frequency 50% duty cycle is designed and implemented.
2) An op-amp relaxation oscillator with 1 kHz frequency 50% duty cycle is simulated using PSpice.

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

1) To design and implement a window comparator for any given UTP and LTP
2) To simulate a window comparator for any given UTP and LTP using PSpice

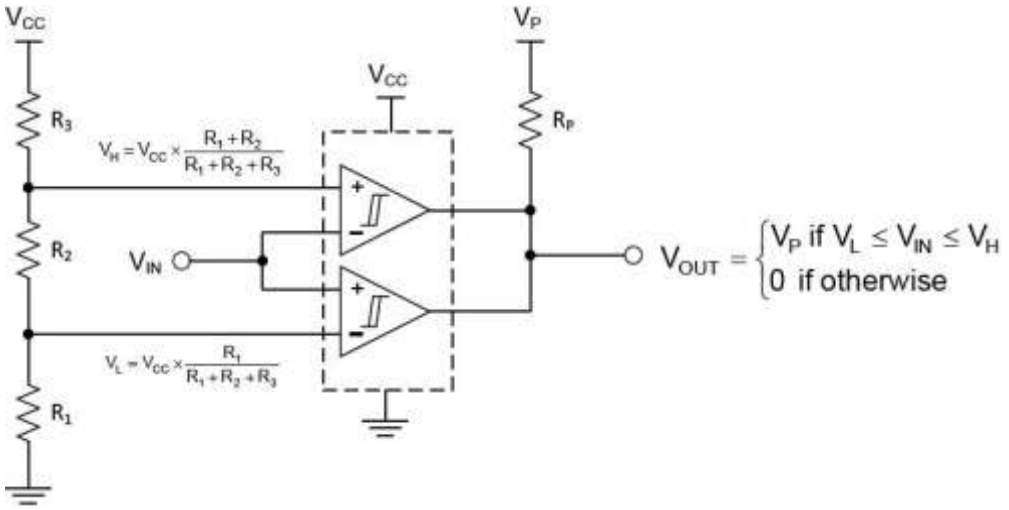
EQUIPMENT REQUIRED:

Sl. No	Components	Quantity
1	IC µA741	2
2	6.5KΩ Resistor	2
3	10KΩ Resistor	3
4	DC regulated power supply	1
5	Signal generator	1
6	CRO	1
7	Wires	As required

THEORY: A Window Comparator is basically the inverting and the non-inverting comparators above 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.

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.

In the voltage divider network, if we now use three equal value resistors so that R1 = R2 = R3 = R we can create a very simple window comparator circuit as shown. Also as the resistive values are all equal, the voltage drops across each resistor will also be equal at one-third the supply voltage, 1/3Vcc. We can set the upper reference voltage to 2/3 Vcc and the lower reference voltage to 1/3 Vcc. Consider the window comparator circuit below.



DESIGN:

- Supply Voltage: 5 V
- Input Range: 0 V – 5V
- Window Range: 1.66 V – 3.33 V Output:
- 0 V to 10 V (36 V maximum)

A reference voltage, V_{CC}, is divided down by resistors R_1-R_3. The two node voltages, V_H and V_L, define the upper window voltage and lower window voltage, respectively. When the input voltage is between V_H and V_L, the output is 'high', or V_P. When outside the window voltage, the output is pulled down to 0 V.

Equations (1) and (2) define V_H and V_L, respectively.

$$V_H = V_{CC} \times \frac{R_1 + R_2}{R_1 + R_2 + R_3} \qquad (1)$$

$$V_H = V_{CC} \times \frac{R_1 + R_2}{R_1 + R_2 + R_3} \qquad (2)$$

Solving Equations (1) and (2) for VCC, setting them equal to each other, then simplifying yields Equation (3)

$$V_H\left(\frac{R_1 + R_2 + R_3}{R_1 + R_2}\right) = V_L\left(\frac{R_1 + R_2 + R_3}{R_1}\right) \rightarrow \frac{V_H}{V_L} = \frac{R_1 + R_2}{R_1} \qquad (3)$$

Given V_H=3.33 V and V_L=1.66 V, the ratio of R_1 and R_2 is calculated in Equation (4).

$$\frac{V_H}{V_L} = \frac{R_1 + R_2}{R_1} \rightarrow \frac{3.33}{1.66} = 2 = 1 + \frac{R_2}{R_1} \rightarrow 1 = \frac{R_2}{R_1} \qquad (4)$$

i.e. $R_1 = R_2$

To limit the current drawn from the reference voltage source, R1 and R2 were selected as 10 KΩ

While the values of R1 and R2 are related to the ratio of the window voltages, R3 determines the voltage value. R3 is calculated in Equation (5)

$$V_L = V_{CC} \times \frac{R_1}{R_1 + R_2 + R_3} \rightarrow R_3 = \frac{R_1 \times V_{CC}}{V_L} - (R_1 + R_2) = 10k\Omega \qquad (5)$$

In order for the output voltage to swing close to the rail, the current should be limited to less than~4 mA.

Given a up voltage of 10 V, RP should be 2.5 kΩ or greater. Therefore, RP was selected to be

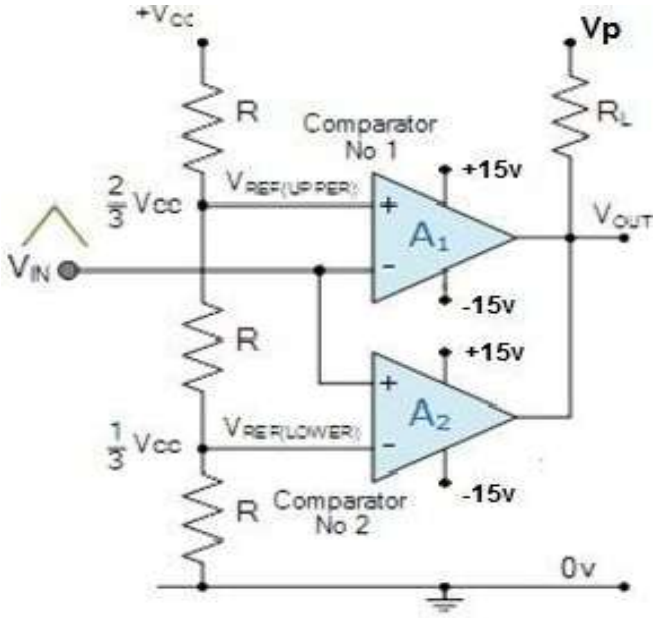
5.1 kΩ. If the pull up voltage is increased, RP may have to be increased in order to obtain good output swing to the negative rail. Notice that when the input, VIN, is between VH and VL, the output goes to the pull-up voltage,

V_P. V_{CC} = 5V, R_P = 5.1 K Ω, V_p = 10V, UTP = (2/3)V_{CC}, LTP=(1/3)V_{CC}

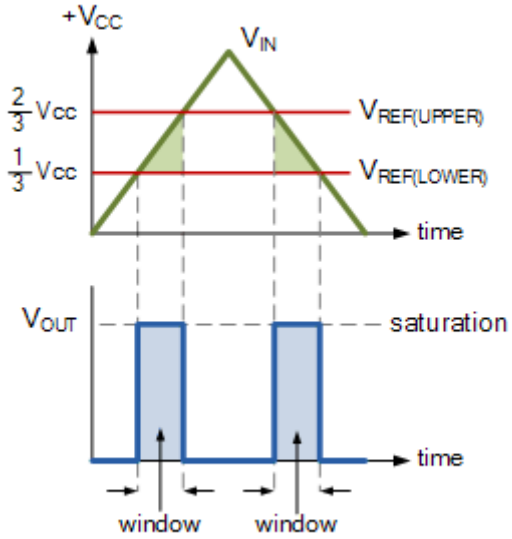
PROCEDURE:

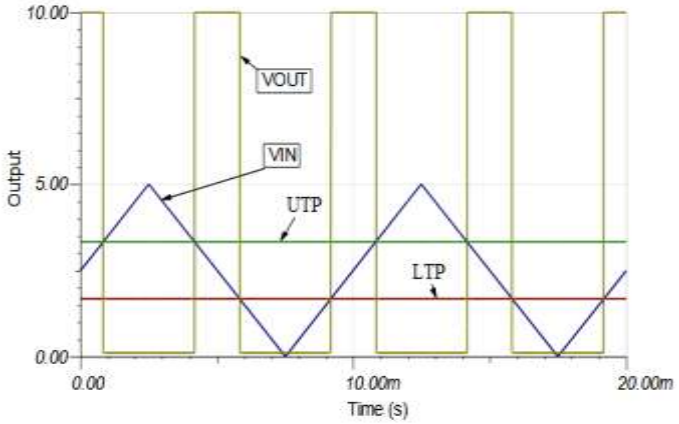
1. Before making the connections check all the components using multimeter.
2. Make the connections as shown in figure and switch on the power supply.
3. Observe the voltage waveform across the capacitor on CRO.
4. Also observe the output waveform on CRO. Measure its amplitude and frequency.

BLOCK / CIRCUIT DIAGRAM:



INPUT/OUTPUT WAVEFORMS:





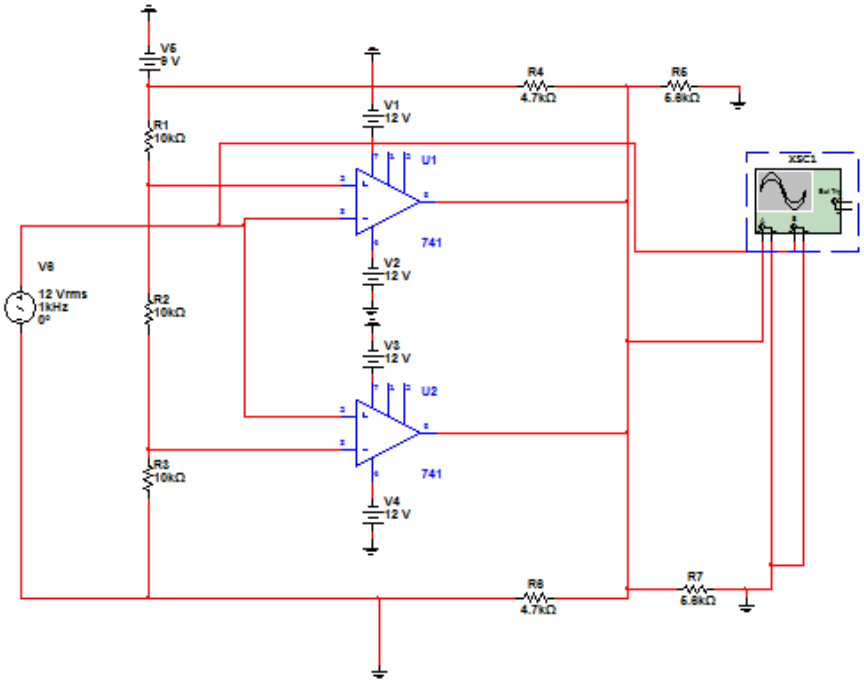
SIMULATION:

- Start Orcad and create new project by clicking File → New → Project → Enter the Project Name and select Analog or Mixed A/D

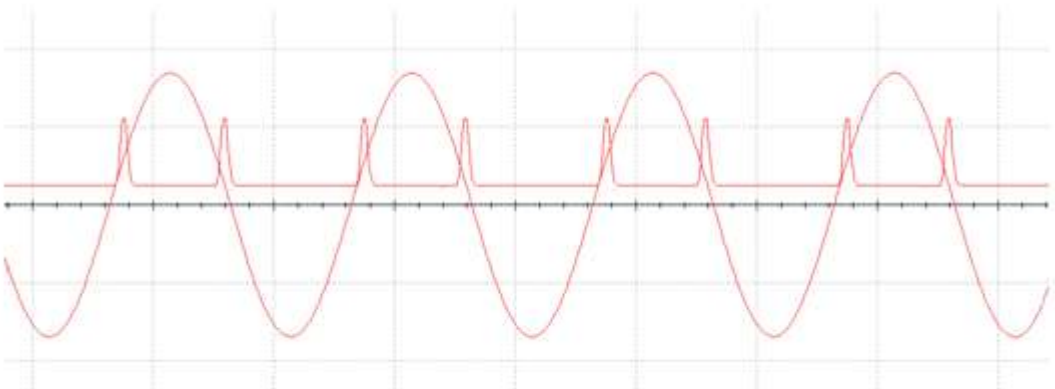
- By clicking Place → Part, select the following components

Part	Library	Quantity
µA741	Eval	2
Resistors	Analog	7
V_{SIN}	Source	1
V_{DC}	Source	5
0/CAPSYM	Place Ground	6

- Connect all the components using Wires.
- Set the values of the component.
- Place the probes at the input and output terminal of the circuit.
- Then click on Save
- Create a new simulation profile with Analysis type: Time domain (Transient), Run to Time: 100ms
- Save and Run the simulation.
- Observe the waveform and note down.



SIMULATION WAVEFORM:



RESULT & CONCLUSION:

1) A window comparator is designed and implemented for any given UTP and LTP
2) A window comparator is simulated for any given UTP and LTP using PSpice

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

AIM:

1) To design and implement half adder, full adder, half subtractor and full subtractor using basic logic gates.
2) To simulate half adder, full adder, half subtractor and full subtractor in VHDL using Xilinx ISE simulator.

EQUIPMENT REQUIRED:

Sl. No	Components	Quantity
1	NOT gate IC7404	1
2	2 input AND gate IC7408	2
3	2 input OR gate IC7432	1
4	Digital Trainer Kit	1
5	Patch Cords	As required

THEORY:

- Adder circuit is a combinational digital circuit that is used for adding numbers. A typical adder circuit produces a sum bit (denoted by S) and a carry bit (denoted by C) as the output. Adders are used in the arithmetic logic units, in other parts of the processor, where they are used to calculate addresses, table indices, increment and decrement operators.

- Half-Adder: A combinational logic circuit that adds two single binary digits *A* and *B*. It has two outputs, sum (*S*) and carry (*C*). The carry signal represents an overflow into the next digit of a multi-digit addition.

- Full-Adder: The half-adder does not take the carry bit from its previous stage into account. This carry bit from its previous stage is called carry-in bit (C_{in}). A combinational logic circuit that adds two data bits A, B, and a carry-in bit C_{in}, is called a full-adder. It has two outputs, sum (*S*) and carry (*C*).

- Subtractor circuit is a combinational digital circuit that is used for subtracting numbers. A typical subtractor circuit produces a diffrence bit (denoted by D) and a brrow bit (denoted by B) as the output.

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

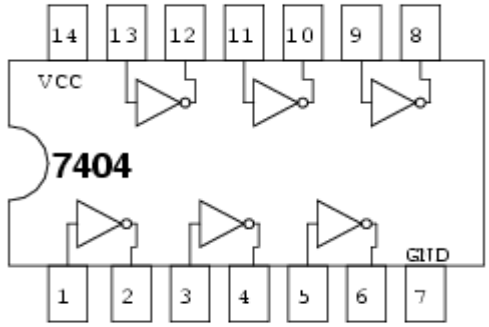
- Full Subtractor: A combinational circuit of full-subtractor performs the operation of subtraction of three bits—the minuend, subtrahend, and borrow generated from the subtraction operation of previous significant digits and produces the outputs difference and borrow.

PROCEDURE:

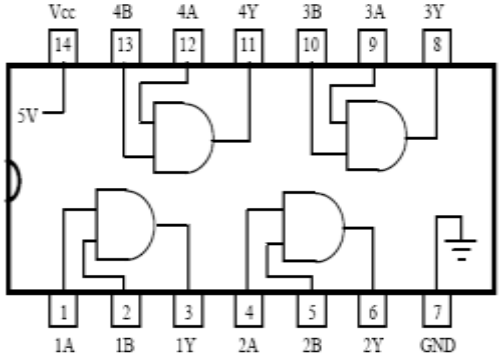
1. Check all the components for their working.
2. Insert the appropriate IC into the IC base.
3. Make connections as shown in the circuit diagram.
4. Give supply to the trainer kit.
5. Provide input data to the circuit via switches.
6. Observe the outputs and verify the Truth Table.

BLOCK / CIRCUIT DIAGRAM:

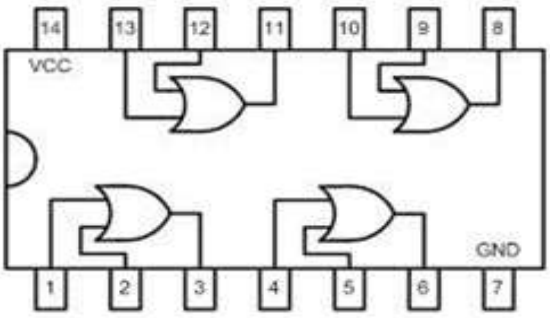
IC 7404 Pin Diagram



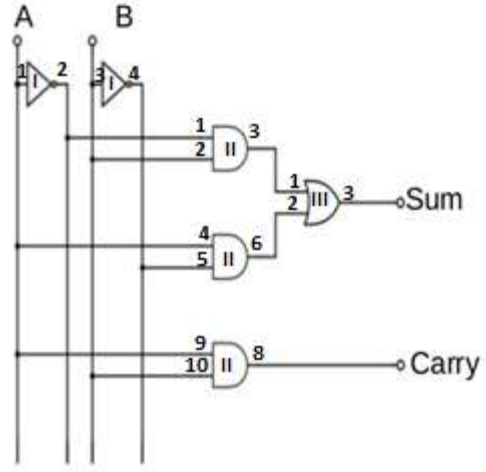
IC 7408 Pin Diagram



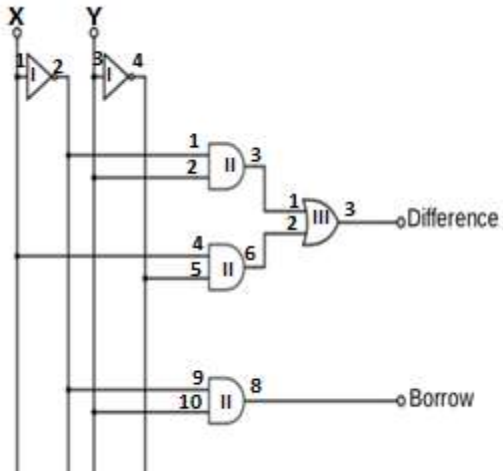
IC 7432 Pin Diagram



Half Adder Circuit

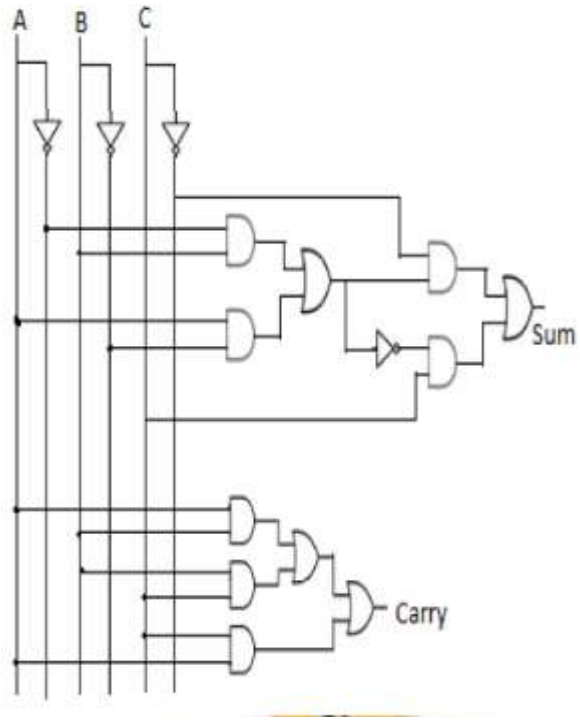


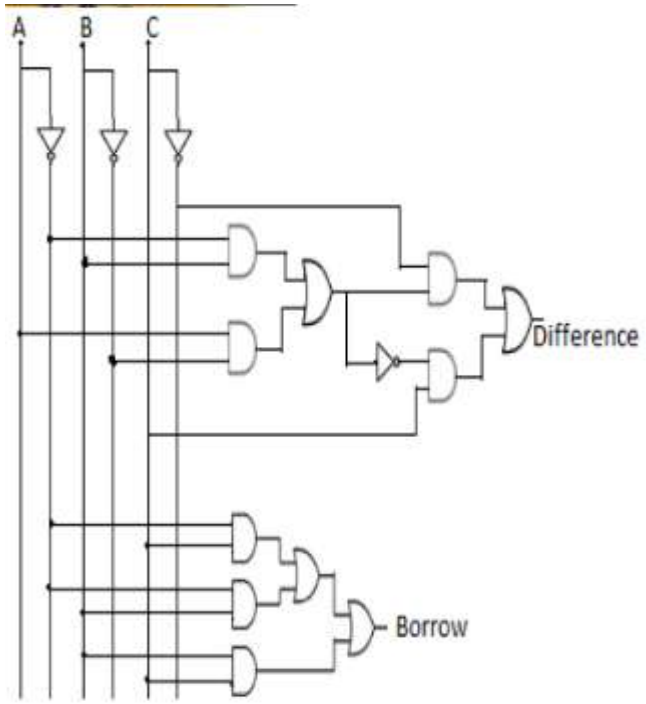
Half Subtractor Circuit



Full Adder Circuit

Full Subtractor Circuit





OBSERVATION TABLE:

Truth Table for Half Adder:

INPUT		OUTPUT	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

SUM = $\bar{A} \cdot B + A \cdot \bar{B}$
CARRY = $A \cdot B$

Truth Table for Half Subtractor:

INPUT		OUTPUT	
X	Y	Difference	Barrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Difference = $\bar{A} \cdot B + A \cdot \bar{B}$
Borrow = $\bar{A} \cdot B$

Truth Table for Full Adder:

INPUT			OUTPUT	
A	B	C_{in}	Sum	Carry
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	0

$$\text{Sum} = \bar{A}.\bar{B}.C + \bar{A}.B.\bar{C} + A.\bar{B}.\bar{C} + A.B.C$$
$$= \bar{A}.B.\bar{C} + A.\bar{B}.\bar{C} + \bar{A}.\bar{B}.C + A.B.C$$
$$= (\bar{A}.B + A.\bar{B})\bar{C} + (\bar{A}.\bar{B} + A.B)C$$
$$= (A \oplus B)\bar{C} + \overline{(A \oplus B)}C$$

Carry = $\sum(3, 5, 6)$
(after simplifying using K map)
Carry = $A.B + B.C + C.A$

Truth Table for Full Subtractor:

INPUT			OUTPUT	
A	B	B_{in}(C)	Difference	Barrow
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

$$\text{Difference} = A.B.C + A.B.C + A.B.C + A.B.C$$
$$= \bar{A}.B.\bar{C} + A.\bar{B}.\bar{C} + \bar{A}.\bar{B}.C + A.B.C$$
$$= (\bar{A}.B + A.\bar{B})\bar{C} + (\bar{A}.\bar{B} + A.B)C$$
$$= (A \oplus B)\bar{C} + \overline{(A \oplus B)}C$$

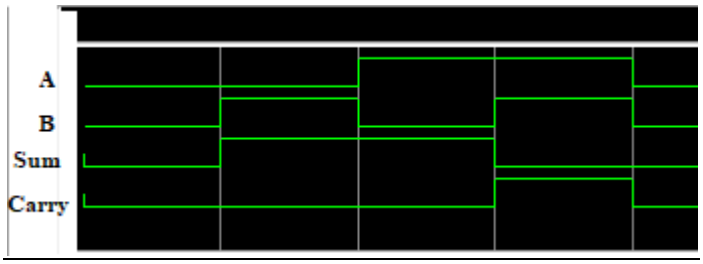
Borrow = $\sum(1, 2, 3, 7)$
(after simplifying using K map)
Borrow= $\bar{A}.C + \bar{A}.B + B.C$

SIMULATION:

Half Adder VHDL Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity HalfAdder is
   Port ( A, B : in  STD_LOGIC;
          Sum, Carry : out  STD_LOGIC);
end HalfAdder;
architecture equation of HalfAdder is
begin
sum <= ((not A)and B)or(A and(not B));
carry <= A and B;
end equation;
```

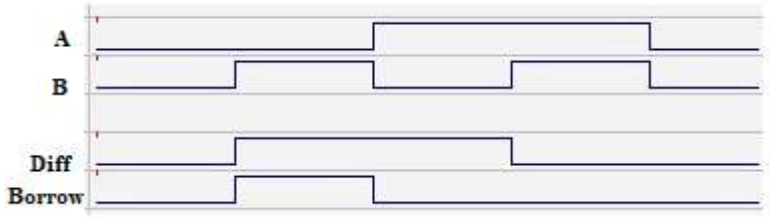
Output Waveform for Half Adder:



Half Subtractor VHDL Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity HalfSub is
   Port ( A,B : in  STD_LOGIC;
          Diff,Borrow : out  STD_LOGIC);
end HalfSub;
architecture equation of HalfSub is
begin
 Diff<=((not A)and B)or(A and(not B));
 Borrow<=((not A)and B);
end equation;
```

Output Waveform for Half Subtractor:

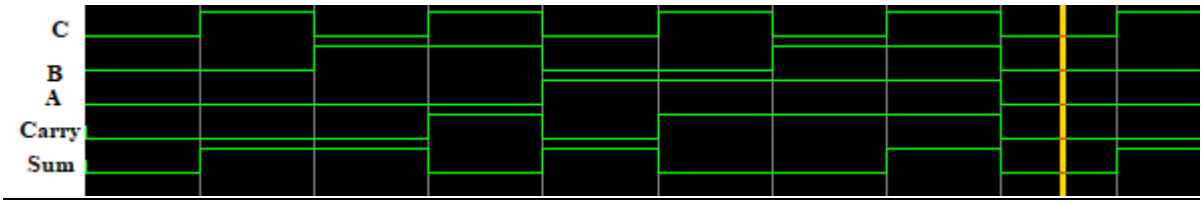


Full Adder VHDL Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity FullAdder is
```

```
    Port ( A,B,C : in  STD_LOGIC;
           Sum,Carry : out  STD_LOGIC);
end FullAdder;
architecture equation of FullAdder is
begin
sum<=(((not A)and(not b)and c)or((not A)and B and(not C))or(a and(not B)and(not C))or(A and B and
C));
carry<=(A and B)or(B and C)or(A and C);
end equation;
```

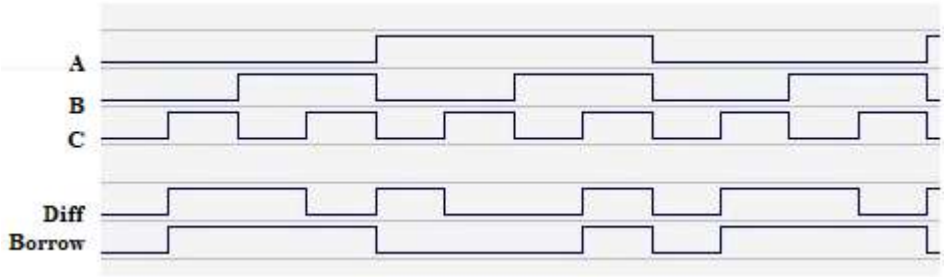
Output Waveform for Full Adder:



Full Subtractor VHDL Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity FullSubtractor is
    Port ( A,B,C : in  STD_LOGIC;
           Diff,Borrow : out  STD_LOGIC);
end FullSubtractor;
architecture equation of FullSubtractor is
begin
Diff <= (((not A)and(not b)and c)or((not A)and B and(not C))or(a and(not B)and(not C))or(A and B
and C));
Borrow <= (A and B)or(B and C)or(A and C);
end equation;
```

Output Waveform for Full Subtractor:



RESULTS & CONCLUSIONS:

1) The truth table of half adder, half subtractor, full adder and full subtractor is verified.
2) The output waveform of half adder, half subtractor, full adder and full subtractor is simulated and verified.

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.

AIM:

1) To simplify 4 variable logic expression using Entered Variable Map and realize the simplified logic expression using 8:1 Multiplexer IC.
2) To simulate 8:1 Multiplexer in VHDL using Xilinx ISE simulator.

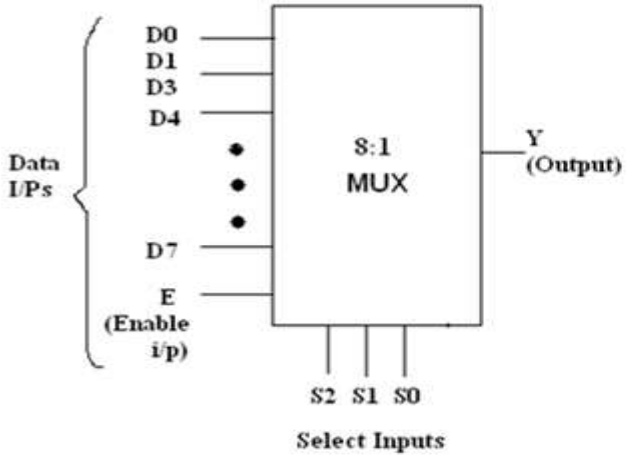
EQUIPMENT REQUIRED:

Sl. No	Components	Quantity
1	8:1 Multiplexer IC74151	1
2	NOT gate IC7404	1
3	Digital Trainer Kit	1
4	Patch Cords	As required

THEORY:

- A digital multiplexer is a combinational circuit that selects binary information from one of the many inputs and transmits to a single output line.

- An 8 to 1 multiplexer has 8 data inputs, 3 selector inputs and 1 output.



Enable	Select Inputs			Output
E	S2	S1	S0	Y
0	X	X	X	0
1	0	0	0	D0
1	0	0	1	D1
1	0	1	0	D2
1	0	1	1	D3
1	1	0	0	D4
1	1	0	1	D5
1	1	1	0	D6
1	1	1	1	D7

- Entering variable into Karnaugh map along with 0's, 1's and don't care conditions is called Entered Variable K Map or Map Entered Variable K map.

- Rules for entering values in a Map Entered Variable K map are:

Rule No.	MEV f	Map Entry	Comments
1	0 0 1 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 0 1 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 X 1 X	X	If function equals X, enter X
6	0 0 1 X	0	f=0 for MEV=0 and f=X for MEV=1,enter 0
7	0 X 1 0	0	f=X for MEV=0 and f=0 for MEV=1,enter 0
8	0 1 1 X	1	f=1 for MEV=0 and f=X for MEV=1,enter 1
9	0 X 1 1	1	f=X for MEV=0 and f=1 for MEV=1,enter 1

PROCEDURE:

1. Assume that the 4-variable Boolean function Y = F(A,B,C,D) = ∑ (2, 4, 5, 7, 10, 14) is to be implemented using 8:1 multiplexer IC 74151.

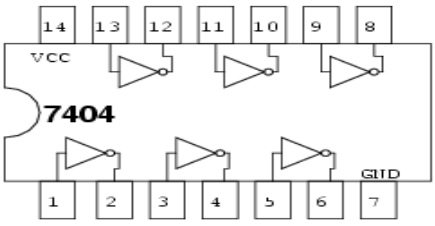
2. Considering A, B, C as the control inputs and D as the data input, implantation table will be

	D0	D1	D2	D3	D4	D5	D6	D7
$\overline{D}$	0 0	1 2	1 4	0 6	0 8	1 10	0 12	1 14
D	0 1	0 3	1 5	1 7	0 9	0 11	0 13	0 15
Data Input to MUX	0	$\overline{D}$	1	D	0	$\overline{D}$	0	$\overline{D}$

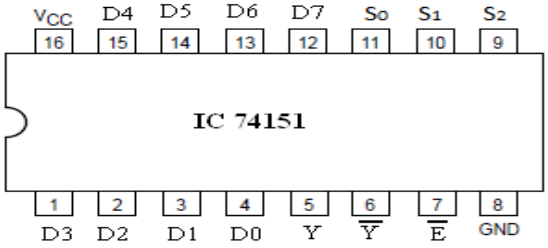
3. Check all the components for their working.
4. Insert the appropriate IC into the IC base.
5. Make connections as shown in the circuit diagram.
6. Give supply to the trainer kit.
7. Provide input data to the circuit via switches.
8. Observe the outputs and verify the Truth Table.

BLOCK / CIRCUIT DIAGRAM:

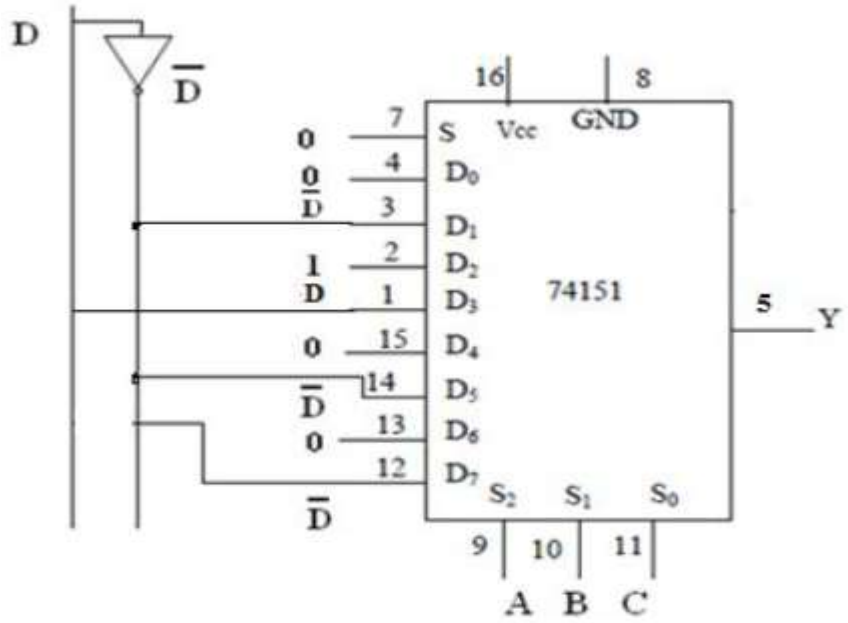
IC 7404 Pin Diagram:



IC 74151 Pin Diagram:



4-variable Boolean function design using IC74151:



OBSERVATION TABLE:

Truth table for Y = F(A,B,C,D) = ∑ (2, 4, 5, 7, 10, 14)

INPUT				OUTPUT
A	B	C	D	Y

0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

SIMULATION:

8:1 multiplexer VHDL Code:

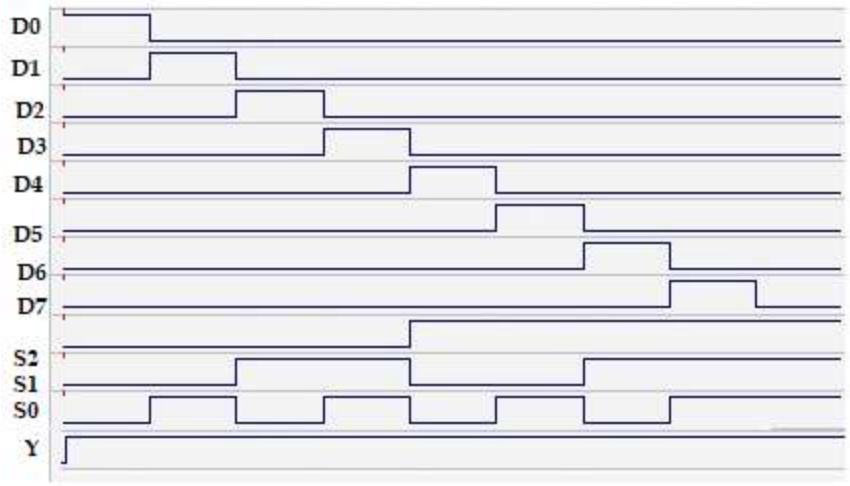
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity mux8to1 is
   Port ( S : in  STD_LOGIC_VECTOR (2 downto 0);
          D : in  STD_LOGIC_VECTOR (7 downto 0);
          Y : out  STD_LOGIC);
end mux8to1;
architecture Dataflow of mux8to1 is
begin
with S select
Y <= D(0) when "000",
D(1) when "001",
D(2) when "010",
D(3) when "011",
D(4) when "100",
D(5) when "101",
D(6) when "110",
D(7) when others;
end Dataflow;
```

Observation Table for 8:1 multiplexer:

Selector Inputs			Data Inputs								Output
A	B	C	D0	D1	D2	D3	D4	D5	D6	D7	Y
0	0	0	1	0	0	0	0	0	0	0	1
0	0	1	0	1	0	0	0	0	0	0	1
0	1	0	0	0	1	0	0	0	0	0	1
0	1	1	0	0	0	1	0	0	0	0	1
1	0	0	0	0	0	0	1	0	0	0	1
1	0	1	0	0	0	0	0	1	0	0	1
1	1	0	0	0	0	0	0	0	1	0	1

1	1	1	0	0	0	0	0	0	0	1	1

Output Waveform for 8:1 multiplexer:



RESULTS & CONCLUSIONS:

1) Given 4 variable Boolean function is implemented using 8:1 multiplexer of IC 74151.

2) The output waveform of 8:1 multiplexer is simulated and verified.

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

AIM:

1) To realize a J-K Master/Slave flip flop using NAND gates and verify its truth table.
2) To simulate J-K Master/Slave flip flop in VHDL using Xilinx ISE simulator.

EQUIPMENT REQUIRED:

Sl. No	Components	Quantity
1	NOT gate IC7404	1
2	3 input NAND gate IC7410	2
3	2 input NAND gate IC7400	1
4	Digital Trainer Kit	1
5	Patch Cords	As required

THEORY:

- The basic 1-bit digital memory circuit is known as a flip-flop. It can have only two states, either the 1 state or the 0 state. A flip-flop, on the other hand, is edge-triggered and only changes state when a control signal goes from high to low or low to high.
- JK Master Slave Flip Flop: A JK master flip flop is positive edge triggered, whereas 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 the positive clock edge and the slave toggles on the negative clock edge. The slave does exactly what the master does.

PROCEDURE:

1. Check all the components for their working.
2. Insert the appropriate IC into the IC base.

3. Make connections as shown in the circuit diagram.
4. Give supply to the trainer kit.
5. Provide input data to the circuit via switches.
6. Observe the outputs and verify the Truth Table.

OBSERVATION TABLE:

Truth Table for JK-Master/Slave Flip-Flop

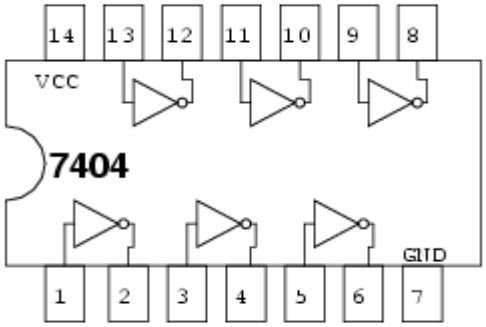
INPUTS		Q(t) BEFORE CP (Establish using PRSET & CLEAR)	OUTPUTS	
J	K		Q(t+1)	Q(t+1)'
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	0	1

(Simplified Table)

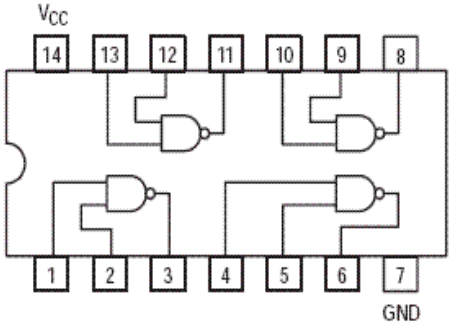
INPUTS		NORMAL OUTPUT
J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	Q(t)'

BLOCK / CIRCUIT DIAGRAM:

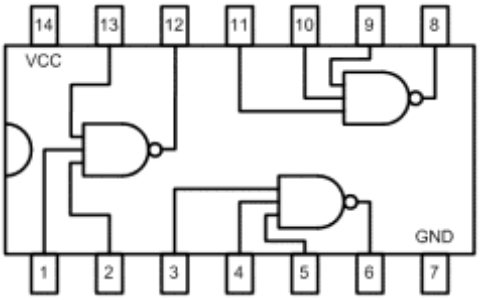
IC 7404 Pin Diagram:



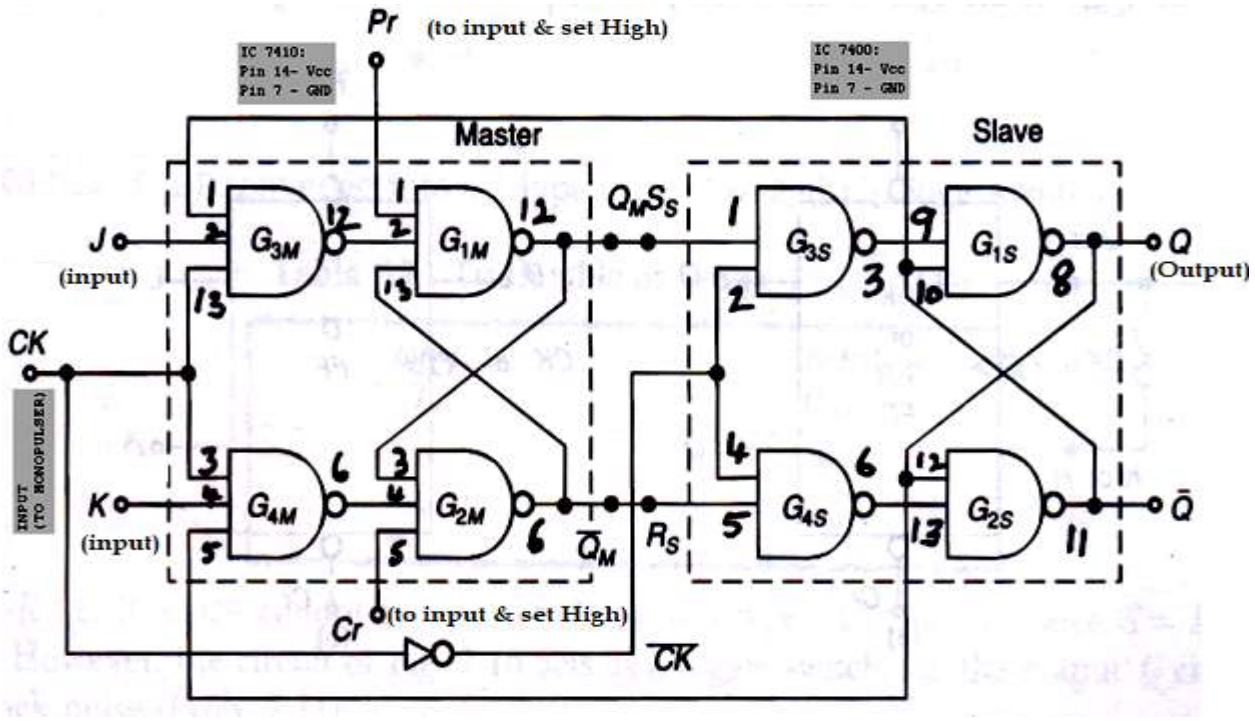
IC 7400 Pin Diagram:



IC 7410 Pin Diagram:



JK Master Slave Flip Flop Circuit:



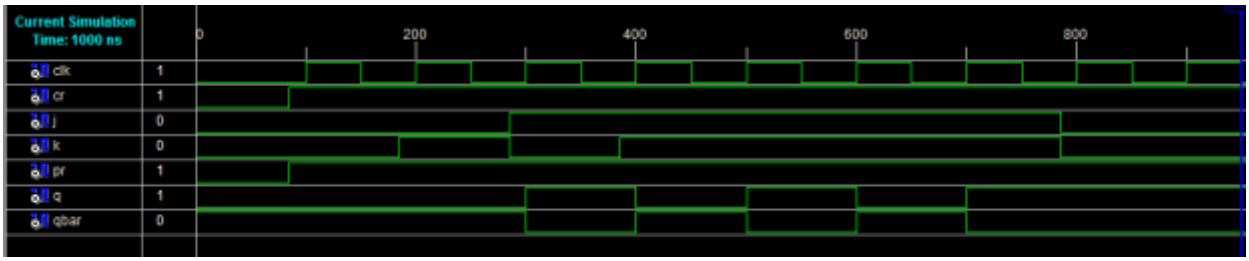
SIMULATION:

J-K Master/Slave flip flop VHDL Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity jk_ff is
        port(j,k,cr,pr,clk:in std_logic;
        q,qbar:out std_logic);
end jk_ff;
architecture behavioural of jk_ff is
signal input:std_logic_vector(1 downto 0);
begin
        input<=j&k;
        process(clk,j,k,pr,cr)
        variable temp:std_logic:='0';
        begin
                if(cr='1' and pr='1')then
                        if rising_edge(clk) then
                                case input is
                                        when "10"=> temp:='1';
                                        when "01"=> temp:='0';
                                        when "11"=> temp:=not temp;
                                        when others=> null;
                                end case;
                        end if;
                else
                        temp:='0';
```

```
            end if;
            q<=temp;
            qbar<=not temp;
        end process;
end behavioural;
```

Output Waveform for J-K flip flop:



RESULTS & CONCLUSIONS:

1) J-K Master/Slave flip flop is realized using NAND gates and its truth table is verified.
2) The output waveform of J-K Master/Slave flip flop is simulated and verified.

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

AIM:

1) To design and implement 4-bit Binary to Gray code converter using basic gates.
2) To design and implement 4-bit Gray to binary code converter using basic gates.

EQUIPMENT REQUIRED:

Sl. No	Components	Quantity
1	NOT gate IC7404	1
2	2 input AND gate IC7408	1
3	2 input OR gate IC7432	1
4	Digital Trainer Kit	1
5	Patch Cords	As required

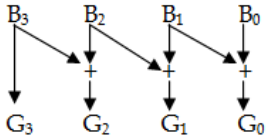
THEORY:

- Code is a symbolic representation of discrete information /data.
- The binary number system is a radix-2 number system with '0' and '1' as the two independent symbols/digits. All larger binary numbers are represented in terms of '0' and '1'. A binary digit is called as a bit. A binary number consists of sequence of bits, each of which is either a 0 or 1. Each bit carries a weight based on its position relative to the binary point. The weight of each bit position is one power of 2 greater than the weight of the position to its immediate right.
  eg: $(15)_{10} = (1111)_2$

- Gray code belongs to a class of code known as minimum change code/cyclic code, in which a number changes by only one bit as it proceeds from one number to the next. Hence this code is not useful for arithmetic operations. This code finds extensive use for shaft encoders, in some types of analog-to-digital converters, etc. Gray code is reflected/mirror code. The Gray code is not a weighted code.
  eg: $(15)_{10} = (1000)_{gray}$

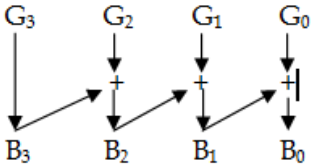
- Any binary number can be converted into equivalent Gray code by the following steps:
  (i) The MSB of the Gray code is the same as the MSB of the binary number.
  (ii) The second MSB (adjacent to the MSB) of the Gray code equals the Ex-OR of the MSB and second MSB of the binary number.

(iii) The third MSB (adjacent to the second MSB) of the Gray code equals the exclusive-OR of the second and third MSB bits of the binary number.
(iv) The process continues until we obtain the LSB of the Gray code number by the addition of the LSB and the next higher adjacent bit of the binary number.



- Any Gray code can be converted into equivalent binary number by the following steps:

(i) The MSB of the binary number is the same as the MSB of the Gray code.
(ii) The second MSB (adjacent to the MSB) of the binary number equals the Ex-OR of the MSB of the binary number and second MSB (adjacent to the MSB) of the Gray code.
(iii) The third MSB (adjacent to the MSB) of the binary number equals the exclusive-OR of the second MSB of the binary number and third MSB of the Gray code.
(iv) The process continues until we obtain the LSB of the binary number.



PROCEDURE:

1. Check all the components for their working.
2. Insert the appropriate IC into the IC base.
3. Make connections as shown in the circuit diagram.
4. Give supply to the trainer kit.
5. Provide input data to the circuit via switches.
6. Observe the outputs and verify the Truth Table.

OBSERVATION TABLE:

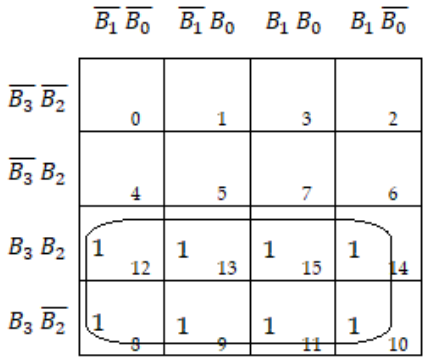
Truth table for Binary to Gray Code Conversion:

BINARY				GRAY			
B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

Truth table for Gray Code to Binary Conversion:

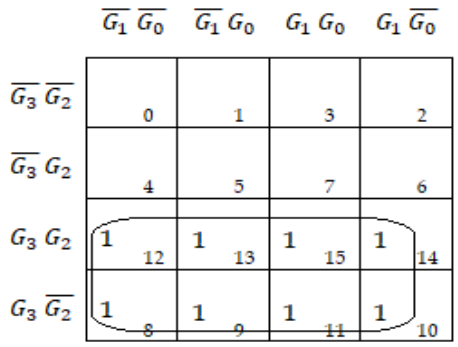
GRAY				BINARY			
G_3	G_2	G_1	G_0	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
1	0	0	0	1	1	1	1
1	0	0	1	1	1	1	0
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	0	1	0	1	1
1	1	1	1	1	0	1	0

$G_3 = \sum m\ (8, 9, 10, 11, 12, 13, 14, 15)$



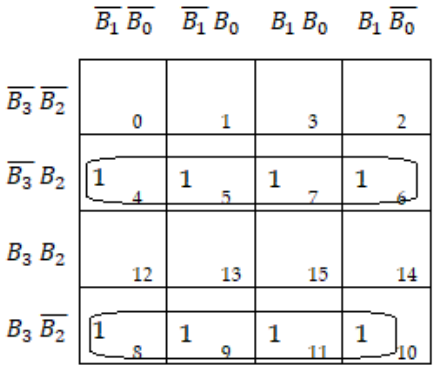
$G_3 = B_3$

$B_3 = \sum m\ (8, 9, 10, 11, 12, 13, 14, 15)$



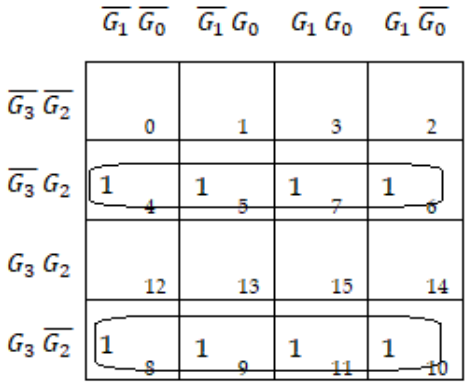
$B_3 = G_3$

$G_2 = \sum m\ (4, 5, 6, 7, 8, 9, 10, 11)$



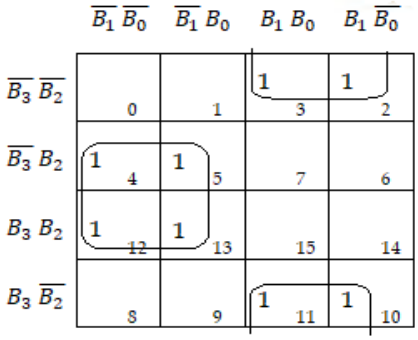
$G_2 = \overline{B_3}\,B_2 + B_3\,\overline{B_2}$
$G_2 = B_2 \oplus B_3$

$B_2 = \sum m\ (4, 5, 6, 7, 8, 9, 10, 11)$



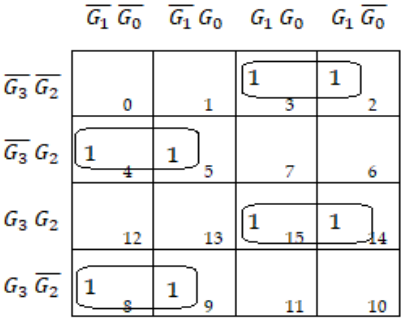
$B_2 = \overline{G_3}\,G_2 + G_3\,\overline{G_2}$
$B_2 = G_2 \oplus G_3$

$G_1 = \sum m\ (2, 3, 4, 5, 10, 11, 12, 13)$



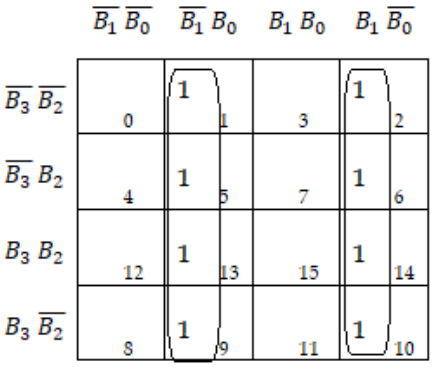
$G_1 = \overline{B_1}\,B_2 + B_1\,\overline{B_2}$
$G_1 = B_1 \oplus B_2$

$B_1 = \sum m\ (2, 3, 4, 5, 8, 9, 14, 15)$

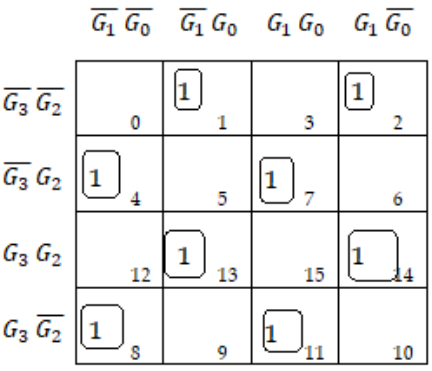


$B_1 = \overline{G_3}\overline{G_2}G_1 + \overline{G_3}G_2\overline{G_1} + G_3G_2G_1 + G_3\overline{G_2}\overline{G_1}$
$B_1 = G_1 \oplus G_2 \oplus G_3$

$G_0 = \sum m\ (1, 2, 5, 6, 9, 10, 13, 14)$



$B_0 = \sum m\ (1, 2, 4, 7, 8, 11, 13, 14)$

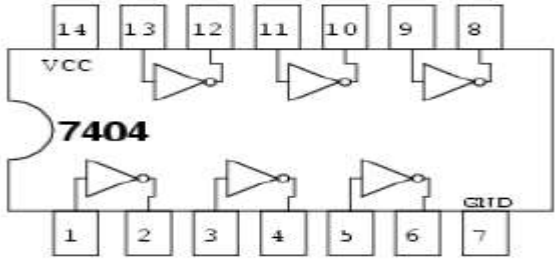


$G_0 = \overline{B_1}\,B_0 + B_1\,\overline{B_0}$
$G_0 = B_0 \oplus B_1$

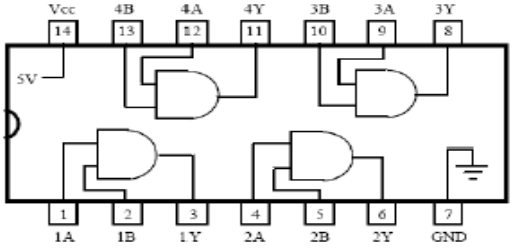
$B_0 = \overline{G}_3\overline{G}_2\overline{G}_1G_0 + \overline{G}_3\overline{G}_2G_1\overline{G}_0 + \overline{G}_3G_2\overline{G}_1\overline{G}_0 + \overline{G}_3G_2G_1G_0 + G_3G_2\overline{G}_1G_0 + G_3G_2G_1\overline{G}_0 + G_3\overline{G}_2\overline{G}_1\overline{G}_0 + G_3\overline{G}_2G_1G_0$
$B_0 = G_0 \oplus G_1 \oplus G_2 \oplus G_3$

BLOCK / CIRCUIT DIAGRAM:

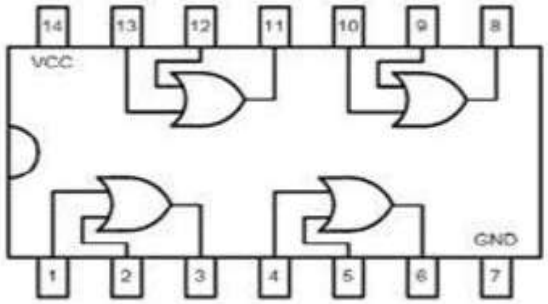
IC 7404 Pin Diagram:



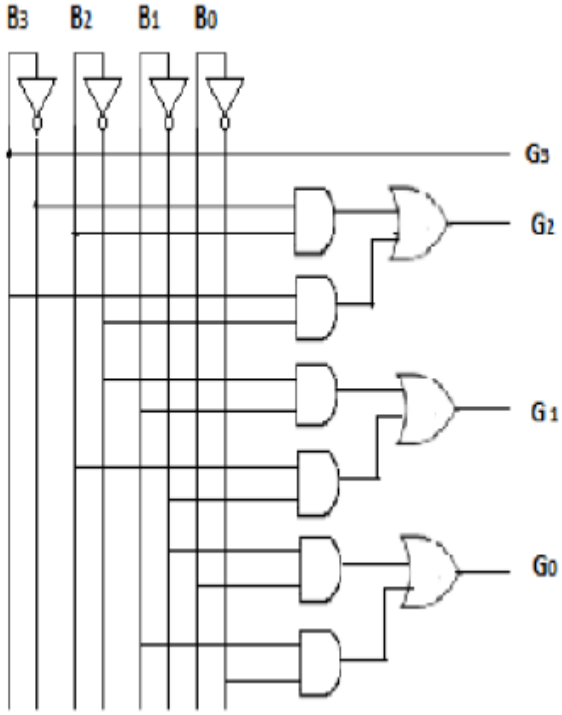
IC 7408 Pin Diagram:



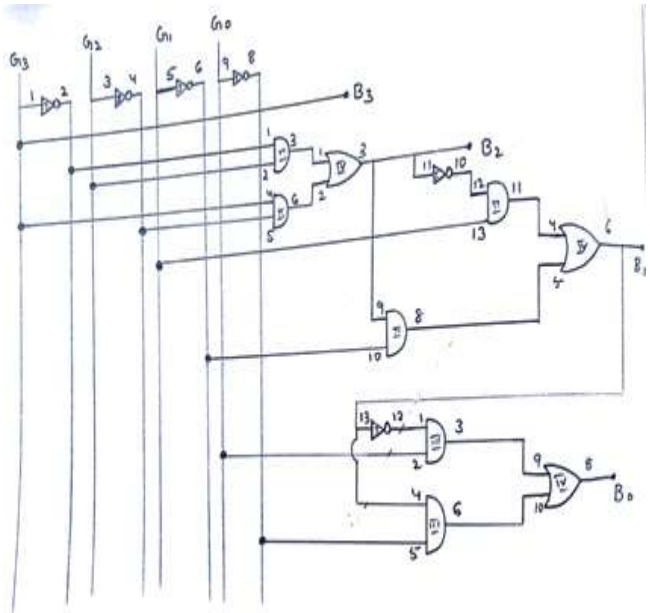
IC 7432 Pin Diagram:



Binary to Gray Code Circuit:



Gray Code to Binary Circuit:



RESULTS & CONCLUSIONS:

1) Binary to gray code conversion is realized using basic gates and truth table is verified.
2) Gray to binary code conversion is realized using basic gates and truth table is verified.

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

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

EQUIPMENT REQUIRED:

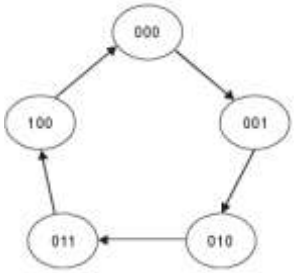
Sl. No	Components	Quantity
1	Dual JK Master/Slave Flip-flop IC 7476	2
2	2 input AND gate IC 7408	1
3	NOT gate IC7400	1
4	Digital Trainer Kit	1
5	Patch Cords	As required

THEORY:

- In digital logic and computing, a counter is a device which stores and displays the number of times a particular event or process has occurred, often in relationship to a clock signal.
- A synchronous counter is one whose output bits change state simultaneously. Such a counter circuit can be built from JK flip-flop by connecting all the clock inputs together, so that each and every flip-flop receives the exact same clock pulse at the exact same time. This results in all the individual output bits changing state at exactly the same time in response to the common clock signal with no ripple effect i.e. with no propagation delay.

Design of MOD 5 counter using JK flip flop:

In order to design a MOD-5, which has five distinct states (N=5), the number of flip-flops required is 3. The state diagram for MOD 5 counter is shown below.



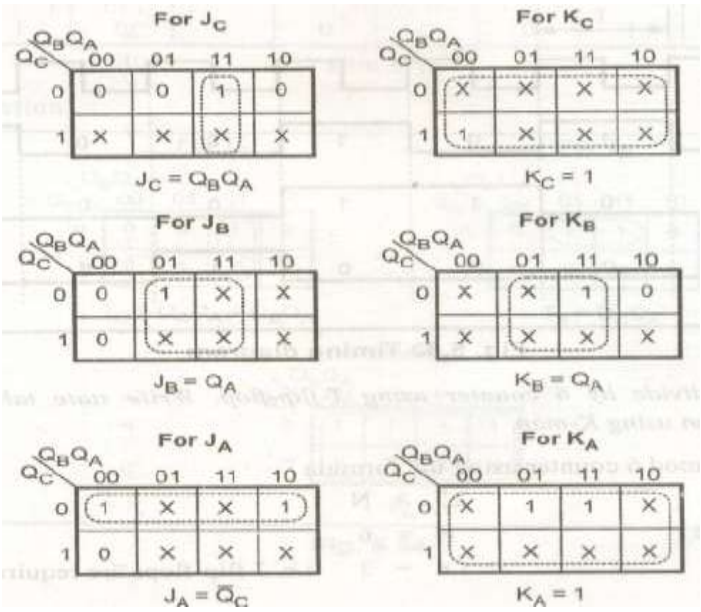
State table of the counter is derived with the excitation table of the JK flip-flop given below.

Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

State table

Present State			Next State			Flip-flop Inputs					
Q_C	Q_B	Q_A	Q_{c+1}	Q_{B+1}	Q_{A+1}	J_C	K_C	J_B	K_B	J_A	K_A
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
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

After K-Map simplification, expressions for the flip-flop inputs are as shown below:



PROCEDURE:

1. Check all the components for their working.
2. Insert the appropriate IC into the IC base.
3. Make connections as shown in the circuit diagram.
4. Give supply to the trainer kit.
5. Provide input data to the circuit via switches.
6. Observe the outputs and verify the Truth Table.

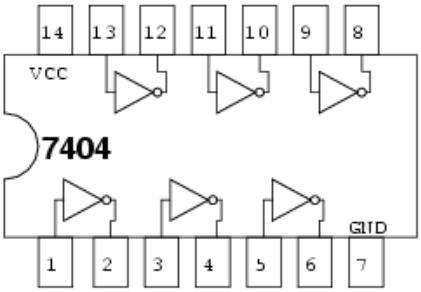
OBSERVATION TABLE:

Truth Table for JK-Master/Slave Flip-Flop

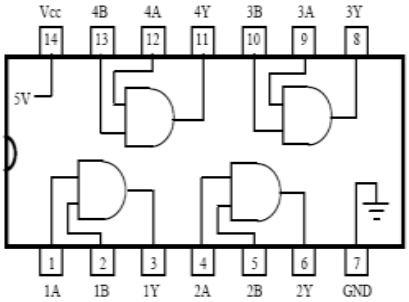
Clock Count	Outputs		
	Q_C	Q_B	Q_A
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	0	0	0

BLOCK / CIRCUIT DIAGRAM:

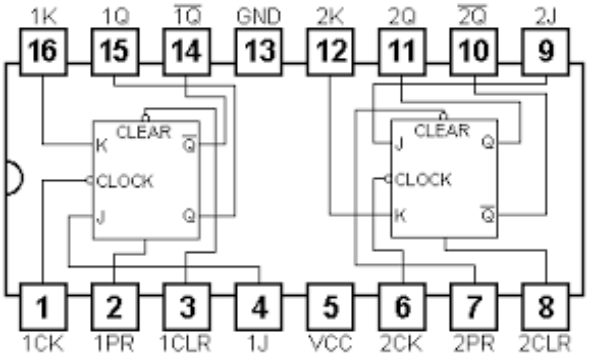
IC 7404 Pin Diagram:



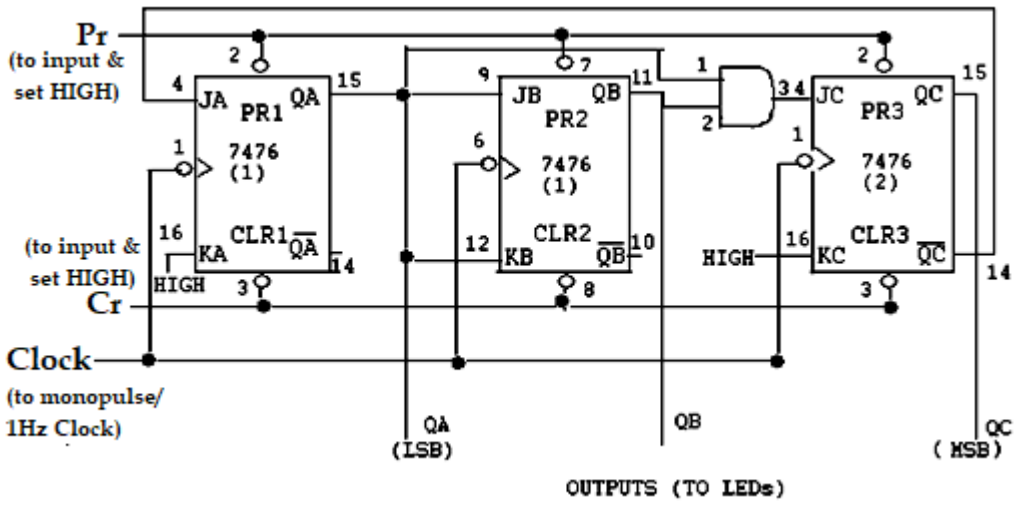
IC 7408 Pin Diagram:



IC 7476 Pin Diagram:



MOD 5 Synchronous Up Counter Circuit:



RESULTS & CONCLUSIONS: A mod-5 synchronous up counter is realized using J-K Flip-Flop ICs and its 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).

AIM: To 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).

EQUIPMENT REQUIRED:

Sl. No	Components	Quantity
1	Asynchronous Decade Counter IC7490	1
2	BCD to seven segment decoder IC7447	1
3	2 input AND gate IC7408	1
4	Digital Trainer Kit	1
5	Patch Cords	As required

THEORY:

- 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.
- A binary coded decimal (BCD) is a serial digital counter that counts ten digits .And it resets for every new clock input. As it can go through 10 unique combinations of output, it is also called as "Decade counter". A BCD counter can count 0000, 0001, 0010, 1000, 1001, 1010, 1011, 1110, 1111, 0000, and 0001 and so on.
- 7490 is an asynchronous decade counter.

PROCEDURE:

1. Check all the components for their working.
2. Insert the appropriate IC into the IC base.
3. Make connections as shown in the circuit diagram.
4. Give supply to the trainer kit.
5. Provide input data to the circuit via switches.
6. Observe the outputs and verify the Truth Table.

OBSERVATION TABLE:

Truth Table for MOD10 asynchronous counter

Clock Count	Output				
	7490				7447
	Q_D	Q_C	Q_B	Q_A	
0(by reset)	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	2
3	0	0	1	1	3
4	0	1	0	0	4
5	0	1	0	1	5
6	0	1	1	0	6
7	0	1	1	1	7
8	1	0	0	0	8
9	1	0	0	1	9
10(repeats)	0	0	0	0	0

Truth Table for MOD 7 asynchronous counter

Clock Count	Output			
	7490			7447
	Q_C	Q_B	Q_A	
0(by reset)	0	0	0	0
1	0	0	1	1
2	0	1	0	2
3	0	1	1	3
4	1	0	0	4
5	1	0	1	5
6	1	1	0	6
7(repeats)	0	0	0	0

Truth Table for MOD 6 asynchronous counter

Clock Count	Output			
	7490			7447
	Q_C	Q_B	Q_A	
0(by reset)	0	0	0	0
1	0	0	1	1
2	0	1	0	2
3	0	1	1	3
4	1	0	0	4
5	1	0	1	5
6(repeats)	0	0	0	0

Truth Table for MOD 5 asynchronous counter

Clock Count	Output			
	7490			7447
	Q_C	Q_B	Q_A	
0(by reset)	0	0	0	0
1	0	0	1	1
2	0	1	0	2
3	0	1	1	3
4	1	0	0	4
5(repeats)	0	0	0	0

Truth Table for MOD 4 asynchronous counter

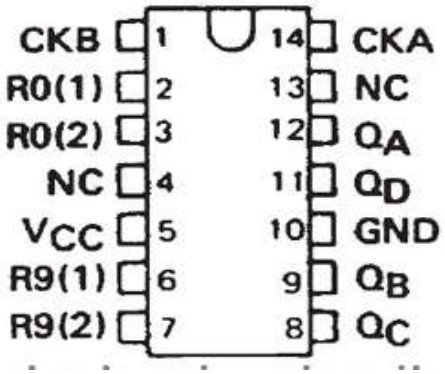
Clock Count	Output		
	7490		7447
	Q_B	Q_A	
0(by reset)	0	0	0
1	0	1	1
2	1	0	2
3	1	1	3
4(repeats)	0	0	0

Truth Table for MOD 3 asynchronous counter

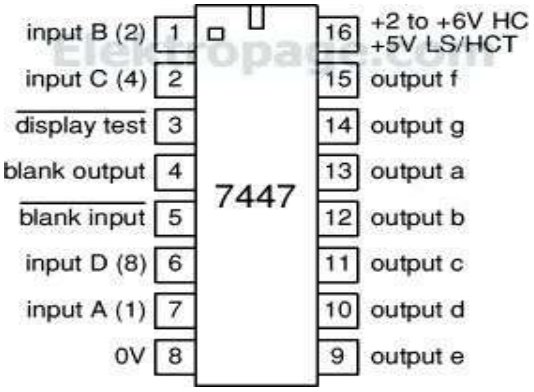
Clock Count	Output		
	7490		7447
	Q_B	Q_A	
0(by reset)	0	0	0
1	0	1	1
2	1	0	2
3(repeats)	0	0	0

BLOCK / CIRCUIT DIAGRAM:

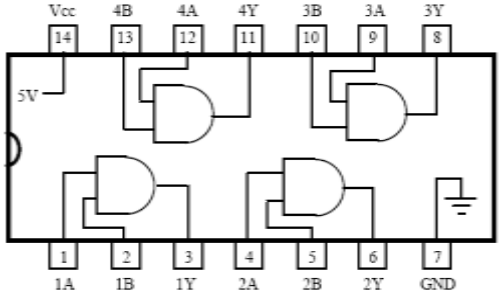
IC 7490 Pin Diagram:



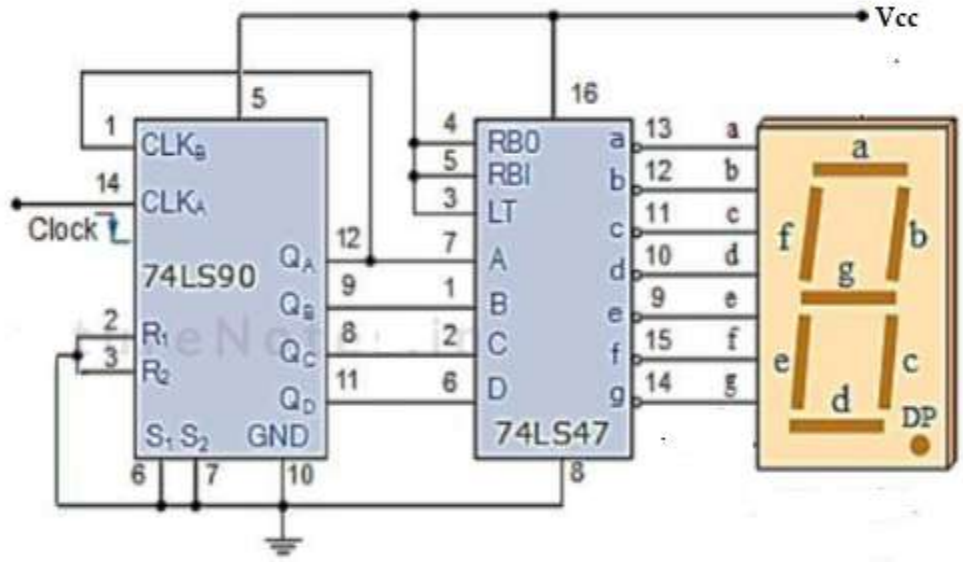
IC 7447 Pin Diagram:



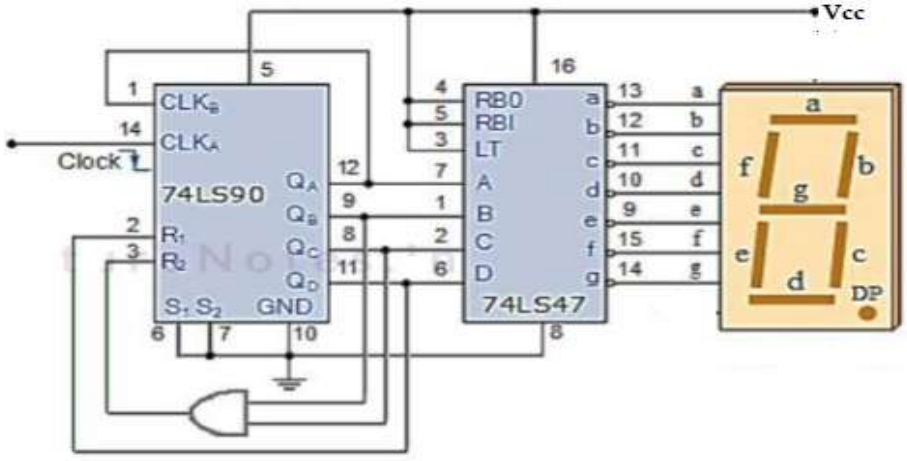
IC 7408 Pin Diagram:



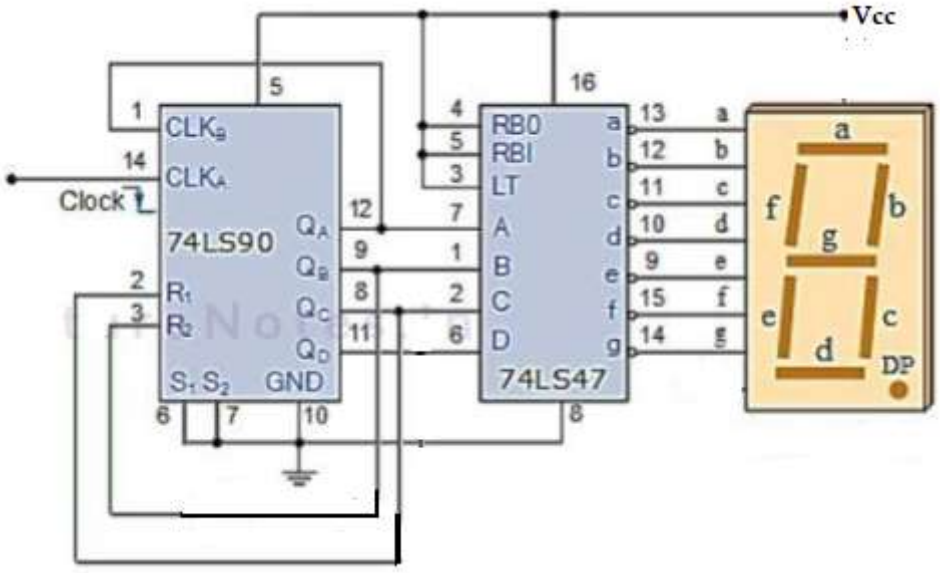
MOD 10 asynchronous counter with 7 segment display Circuit



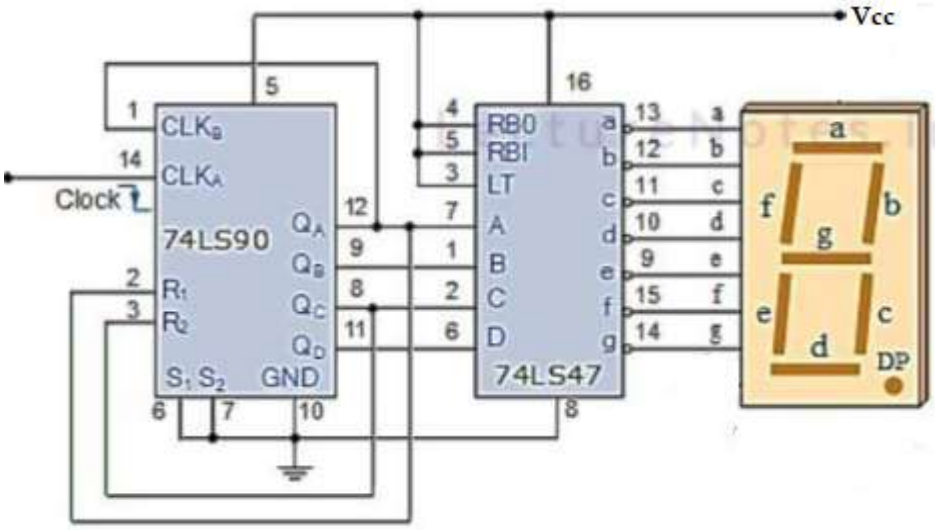
MOD 7 asynchronous counter with 7 segment display Circuit



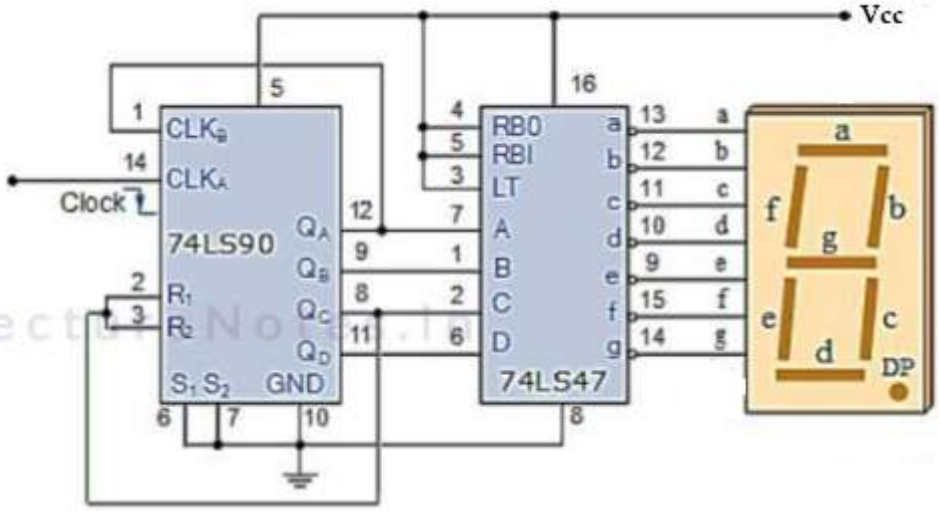
MOD 6 asynchronous counter with 7 segment display Circuit



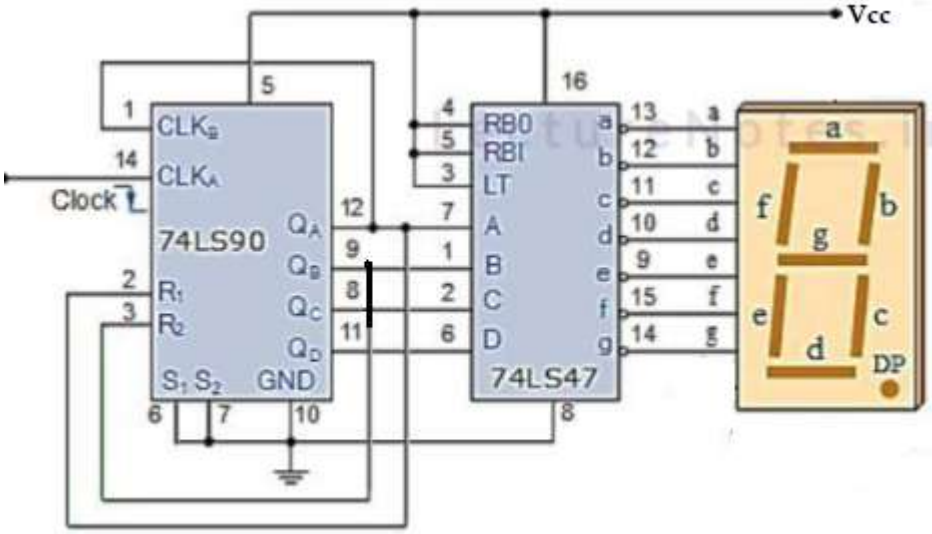
MOD 5 asynchronous counter with 7 segment display Circuit



MOD 4 asynchronous counter with 7 segment display Circuit



MOD 3 asynchronous counter with 7 segment display Circuit



RESULTS & CONCLUSIONS: Asynchronous counter to count up from 0 to n (n≤9) is realized using decade counter IC and seven segment display IC-7447.