



**DEPARTMENT  
OF  
ELECTRONICS AND TELECOMMUNICATION ENGINEERING**

**RAMAIAH INSTITUTE OF TECHNOLOGY  
(Autonomous Institute, Affiliated to VTU)  
BENGALURU – 560054**

**III SEMESTER  
DIGITAL CIRCUIT DESIGN LAB MANUAL (ETL37)  
2023-2024**

**Faculty In-Charge:**

**Dr. H R RAMYA  
Dr. PARIMALA PRABHAKAR  
Dr. K R SHOBHA**

**RAMAIAH INSTITUTE OF TECHNOLOGY  
(Autonomous Institute, Affiliated to VTU)  
BANGALORE – 54**

### About the Institute:

Dr. M. S. Ramaiah a philanthropist, founded 'Gokula Education Foundation' in 1962 with an objective of serving the society. M S Ramaiah Institute of Technology (MSRIT) was established under the aegis of this foundation in the same year, creating a landmark in technical education in India. MSRIT offers 17 UG programs and 15 PG programs. All these programs are approved by AICTE. All eligible UG and PG programs are accredited by National Board of Accreditation (NBA). The institute is accredited with '**A+**' **grade by NAAC in March 2021** for 5 years. University Grants Commission (UGC) & Visvesvaraya Technological University (VTU) have conferred Autonomous Status to MSRIT for both UG and PG Programs since 2007. The institute has also been conferred autonomous status for Ph.D. program since 2021. The institute is a participant to the Technical Education Quality Improvement Program (TEQIP), an initiative of the Government of India. The institute has 380 competent faculty out of which 67% are doctorates. Some of the distinguished features of MSRIT are: State of the art laboratories, individual computing facility for all faculty members, all research departments active with sponsored funded projects and more than 300 scholars pursuing Ph.D. To promote research culture, the institute has established Centre of Excellence for Imaging Technologies, Centre for Advanced Materials Technology, Centre for Antennas and Radio Frequency systems (CARFS), Center for Cyber Physical Systems, Schneider Centre of Excellence & Centre for Bio and Energy Materials Innovation. **Ramaiah Institute of Technology has obtained "Scimago Institutions Rankings" All India Rank 107 & world ranking 600 for the year 2022.**

The Entrepreneurship Development Cell (EDC) and Section 8 company "Ramaiah Evolute" have been set up on campus to incubate startups. **M S Ramaiah Institute of Technology is recognized by Atal Ranking of Institutions on Innovation Achievements (ARIIA), MoE, Govt. of India.** MSRIT has a strong Placement and Training department with a committed team, a good Mentoring/Proctorial system, a fully equipped Sports department, large air-conditioned library with good collection of book volumes and subscription to International and National Journals. The Digital Library subscribes to online e-journals from Elsevier Science Direct, IEEE, Taylor & Francis, Springer Link, etc. The Institute is a member of DELNET, CMTI and VTU E-Library Consortium. The Institute has a modern auditorium, recording studio, and several hi-tech conference halls with video conferencing facilities. The institute has excellent hostel facilities for boys and girls. MSRIT Alumni have distinguished themselves by occupying high positions in India and abroad and are in touch with the institute through an active Alumni Association.

**As per the National Institutional Ranking Framework (NIRF), MoE, Government of India, Ramaiah Institute of Technology has achieved 78<sup>th</sup> rank among 1314 top Engineering Institutions & 23<sup>rd</sup> Rank for School of Architecture in India for the year 2023.**

### About the department:

The Department of Electronics & Telecommunication Engineering (Formerly known as Department of Telecommunication Engineering) was established in 1996 to address the increasing demand for professionals with expertise in communication and networking technology in India. The Department has state of the art laboratories, equipment's, resources and committed faculty having best of the academic and industry recognition. The Department started a **M.Tech program in Digital Communication in the year 2004**. The Department also started a **Research Centre** in the year 2012 and currently has 07 Research Scholars carrying

out their Research. Department has collaborations with some of the leading industries like **Ansys, Rohde & Schwarz, JV Micronics, Nokia, Huawei Technologies, Intel, Samsung**, and with leading national and international universities like **Bradley University, IIT-M**, enabling the department to focus on R&D, and thus providing new avenues for PG/UG students for placement and higher studies. Both UG and PG Programs are accredited by the **National Board of Accreditation**. There are **5 Funded Research projects** (Industry and Government) ongoing in the department involving students to carry out innovative projects. Many professional activities are organized regularly to the students under various professional societies like IEEE Sensor Council, IEEE Communication Society, IEEE Antenna and Propagation Society, IETE Bangalore and IEEE MTTs student Branch.

### **VISION OF THE INSTITUTE**

The department of ETE has established the Centre of Excellence – Centre for Antennas and Radio Frequency Systems (CARFS) Jointly with ECE department on 24th March 2021 to engage in advanced Research leading to innovation in the areas of Antennas & RF Systems. The CARFS has the State of the art Facilities to collaborate with Researchers in other Institutions across the Country and World in various projects.

To be an Institution of International Eminence, renowned for imparting quality technical education, cutting edge research and innovation to meet global socio- economic needs

### **MISSION OF THE INSTITUTE**

**RIT shall meet the global socio-economic needs through**

- Imparting quality technical education by nurturing a conducive learning environment through continuous improvement and customization
- Establishing research clusters in emerging areas in collaboration with globally reputed organizations
- Establishing innovative skills development, techno-entrepreneurial activities and consultancy for socio-economic needs

### **QUALITY POLICY**

We at Ramaiah Institute of Technology strive to deliver comprehensive, continually enhanced, global quality technical and management education through an established Quality Management System complemented by the synergistic interaction of the stake holders concerned

### **VISION OF THE DEPARTMENT**

To provide an ambience for the students to excel in studies, research and innovation, focusing on meeting global socio-economic needs from a Telecommunication Engineering perspective

#### **MISSION OF THE DEPARTMENT**

- Providing high quality technical education to create world class Telecommunication engineers.
- Creating an ambience for skill development, research and entrepreneurial activities to meet socio-economic needs

#### **PROGRAM EDUCATIONAL OBJECTIVES (PEOs):**

- **PEO1:** Graduates will excel in professional careers in Industry, Academia and Research to meet Socio-Economic needs.
- **PEO2:** Graduates will analyze problems specific to Telecommunication Engineering and multidisciplinary domains providing technically feasible solutions.
- **PEO3:** Graduates will exhibit professional communication skills, teamwork, leadership qualities, ethical behavior and lifelong learning.

#### **PROGRAM OUTCOMES (POs):**

- **PO1: Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- **PO2: 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.
- **PO3: 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.
- **PO4: 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.
- **PO5: Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

- **PO6: 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.
- **PO7: 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.
- **PO8: Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- **PO9: Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- **PO10: Communication:** Communicate effectively on complex engineering activities with the engineering community and with 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.
- **PO11: 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.
- **PO12: Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

**PROGRAM SPECIFIC OUTCOMES (PSOs):**

- 
- **PSO1:** Identify, analyze, formulate, design and demonstrate applications relevant to telecommunication engineering using electronic devices.
- **PSO2:** Use current technology and modern tools to address solutions for telecommunication products by taking into account safety, healthy environmental requirements.
- **PSO3:** Apply project management tools to solve Telecommunication systems by exhibiting teamwork, lifelong learning.

**COURSE DESIGN, DELIVERY AND ASSESMENT**

Course Code and Title: <b>Digital Circuit Design Lab, ETL37</b>	Course Credits : 0:0:1
CIE : <b>50 Marks</b>	SEE : <b>50 Marks</b>
Total No of Lab Hours : <b>28 Hrs</b>	
Prepared by :Dr.Ramya H R	Date : <b>15/11/2023</b>
Reviewd by : Dr.K R Shobha	Date : <b>15/11/2023</b>

**Prerequisites**

<b>Prerequisite Courses with codes : Basic Electronics (EC13)</b>
---

**COURSE OBJECTIVES:**

1. Demonstrating to students how to simplify, analyse and design combinational digital circuit.
2. Delivering to students how to build a sequential circuit (memory) using gates practically.
3. Appreciate the importance of Verilog HDL to design and verify digital circuit using EDA tools.
4. Understand the lexical conventions of VERILOG HDL at dataflow, gate level, structural, Behavioral and RTL levels.
5. Interpret Verilog constructs for logic synthesis using CAD tools. Discriminate between manual and automated logic synthesis and their impact on design. Discussing different FPGA architectures.

**Note:** I) Student should design the logic circuit using **gates** and wiring the circuit using IC's & **trainer kit** to verify the design.

II) Student should write the Verilog HDL module to simulate and synthesize the logic circuit using FPGA **XC3S400 IC** hardware board with **Xilinx 14.7i** software.

## Content Sheet

SL NO	LIST OF EXPERIMENTS	Page No	
<b>1</b>	Simplification, realization of Boolean expressions using logic gates/Universal gates and Verilog HDL module	<b>1-7</b>	
<b>2</b>	Realization of Half/Full adder and Half/Full subtractions using logic gates and Verilog HDL module	<b>8-13</b>	
<b>3</b>	Realization of 4-bit parallel adder/subtractions using 7483 IC	<b>14-18</b>	
<b>4</b>	Realization of 3-bit Binary to Gray code conversion and vice versa.	<b>19-23</b>	
<b>5</b>	MUX/DEMUX – use of 74153, 74139 for arithmetic circuits and code converter	<b>24-35</b>	
<b>6</b>	MUX/DEMUX using Verilog HDL module	<b>36-39</b>	
<b>7</b>	Realization of One/Two/Four-bit comparator/multiplier using gates ,7485 Ic and Verilog HDL module	<b>40-44</b>	
<b>8</b>	BCD to 7 Segment Decoder/Driver , Design of Encoder with & without priority (74147) and Verilog HDL module	<b>45-55</b>	
<b>9</b>	Truth table verification of Flip-Flops: (i) JK Master slave (ii) T type and (iii) D type. Realization of 3 bit counters as a sequential circuit and MOD – N counter (7476)	<b>45-55</b>	
<b>10</b>	Realization of 4 bit counters as a sequential circuit and MOD – N counter design (7490, 74192) and using Verilog HDL module	<b>56-63</b>	
<b>11</b>	Shift left, shift right, SIPO, SISO, PISO, PIPO operations using 74S95 and using Verilog HDL module	<b>64-68</b>	
<b>12</b>	Writing the Verilog module to simulate and synthesize Ring/Johnson Counter and using Verilog HDL module	<b>69-72</b>	
<b>13</b>	Writing a Verilog module to interface Stepper motor and DC-motor to FPGA.	<b>73-76</b>	
<b>14</b>	Writing a Verilog module to interface DAC to FPGA.	<b>73-76</b>	

## TEXT BOOKS

1. John M Yarbrough, “Digital Logic Applications and Design”, Thomson Learning, 2001.
2. Donald D Givone, “Digital Principles and Design “, Tata McGraw Hill Edition, 2002.
3. Samir Palnitkar, VERILOG HDL-A Guide to digital design and synthesis- 2nd edition, Pearson education.2003.
4. Wayne Wolf, FPGA based system design- Reprint 2005, Pearson Education.

## REFERENCE BOOKS:

1. R D Sudhaker Samuel, “Logic Design – A simplified approach” , Sanguine Technical Publishers, 2004.
2. Stephen Brown, Zvonko Vranesic, Fundamentals of Digital logic with VERILOG design, TMH.

## WEB LINKS AND VIDEO LECTURES (e-Resources):

- <https://dld-iitb.vlabs.ac.in>
- <https://de-iitr.vlabs.ac.in>
- <https://de-iitg.vlabs.ac.in>
- <http://vlabs.iitkgp.ac.in>

## Course Contents and Lecture Schedule:

Lesson Plan	Topics	Duration	CO's
1	Simplification, realization of Boolean expressions using logic gates/Universal gates and Verilog HDL module	2 hr	CO1,3
2	Realization of Half/Full adder and Half/Full subtractions using logic gates and Verilog HDL module	2 hr	CO1,3
3	Realization of 4-bit parallel adder/subtractions using 7483 IC	2 hr	CO1,3
4	Realization of 3-bit Binary to Gray code conversion and vice versa.	2 hr	CO1,3
5	MUX/DEMUX – use of 74153, 74139 for arithmetic circuits and code converter	2 hr	CO1
6	MUX/DEMUX using Verilog HDL module	2 hr	CO1,3
7	Realization of One/Two/Four-bit comparator/multiplier using gates ,7485 Ic and Verilog HDL module	2 hr	CO1,3

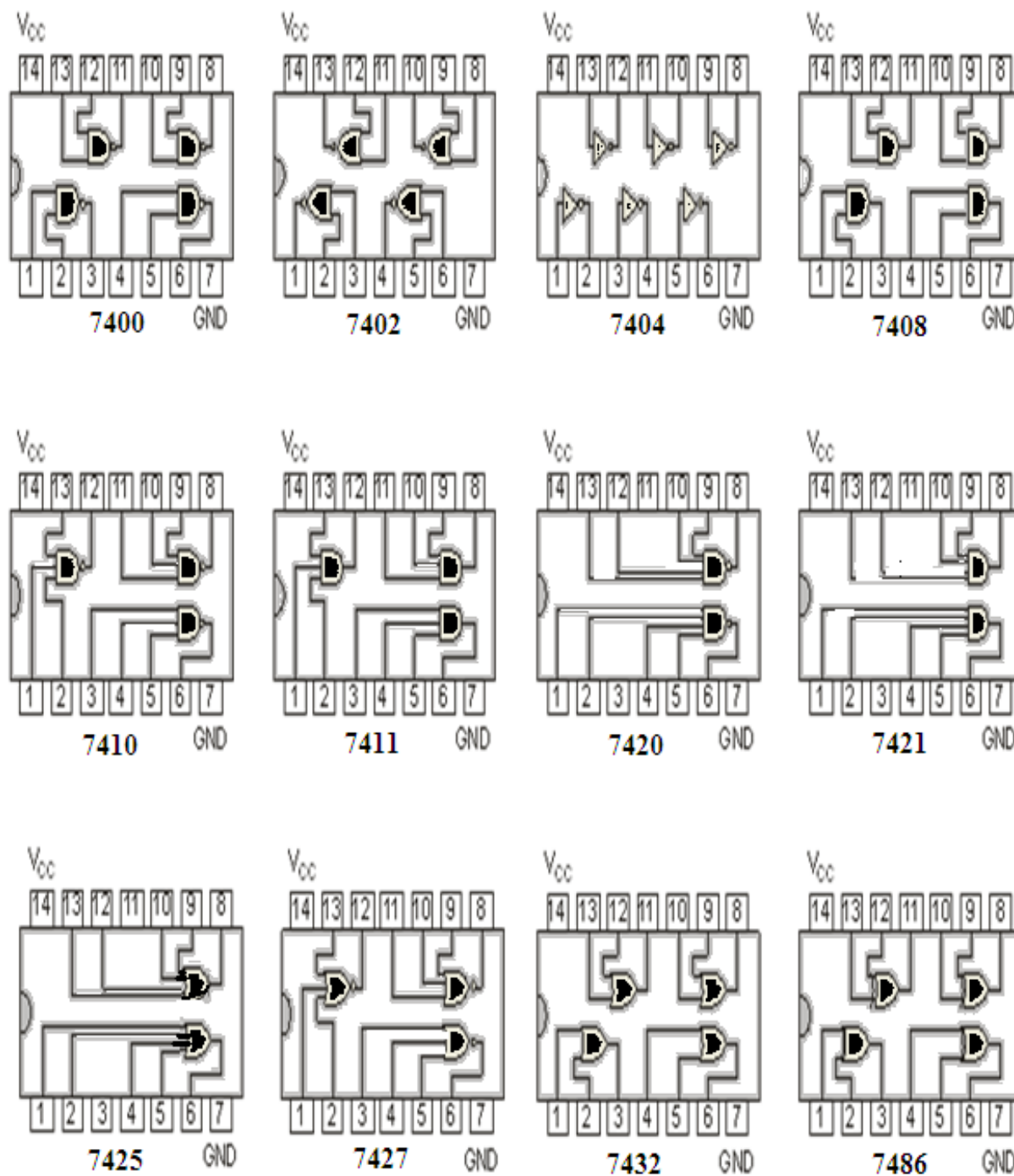


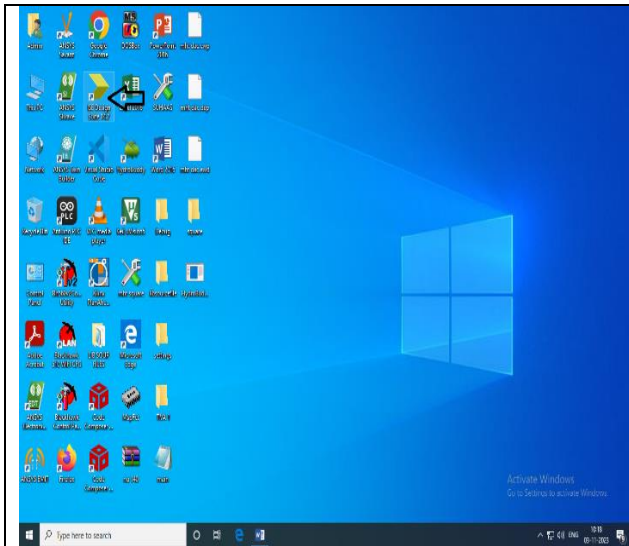
8	BCD to 7 Segment Decoder/Driver , Design of Encoder with & without priority (74147) and Verilog HDL module	2 hr	CO2,4
9	Truth table verification of Flip-Flops: (i) JK Master slave (ii) T type and (iii) D type. Realization of 3 bit counters as a sequential circuit and MOD – N counter (7476)	2 hr	CO2,4
10	Realization of 4 bit counters as a sequential circuit and MOD – N counter design (7490, 74192) and using Verilog HDL module	2 hr	CO2,4
11	Shift left, shift right, SIPO, SISO, PISO, PIPO operations using 74S95 and using Verilog HDL module	2 hr	CO2,4
12	Writing the Verilog module to simulate and synthesize Ring/Johnson Counter and using Verilog HDL module	2 hr	CO2,4
13	Writing a Verilog module to interface Stepper motor and DC-motor to FPGA	2 hr	CO4,5
14	Writing a Verilog module to interface DAC to FPGA.	2 hr	CO4,5

### Course Outcomes

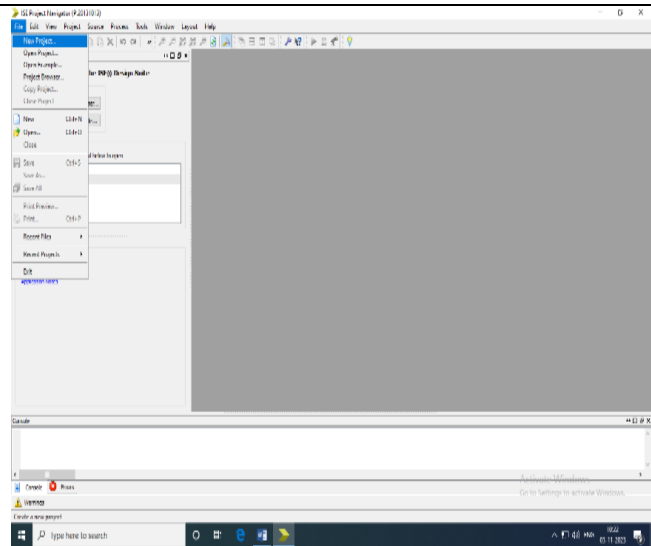
1. Ability to design and evaluate combinational logic circuits with minimum gates. **(PO1, 2, 3, 8,9, 10, 12) (PSO1, 2, 3)**
2. Ability to design and evaluate sequential networks with a minimum number of gates. **(PO1, 2, 3,8,9,10,12) (PSO 1, 2, 3)**
3. Ability to design and evaluate simple or complex logical circuits using Verilog HDL. **(PO1, 2, 3, , 8,9, 10, 12) (PSO1, 2, 3)**
4. Ability to design & evaluate memory blocks with minimum number of gates using FPGA architecture. **(PO1, 2, 3, , 8,9,10, 12) (PSO 1, 2, 3).**
5. Design and synthesize digital circuit on FPGA processor using EDA tools. **(PO1, 2, 3, , 8, 9, 10, 12) (PSO1, 2, 3)**

### IC Internal Pin Configuration:

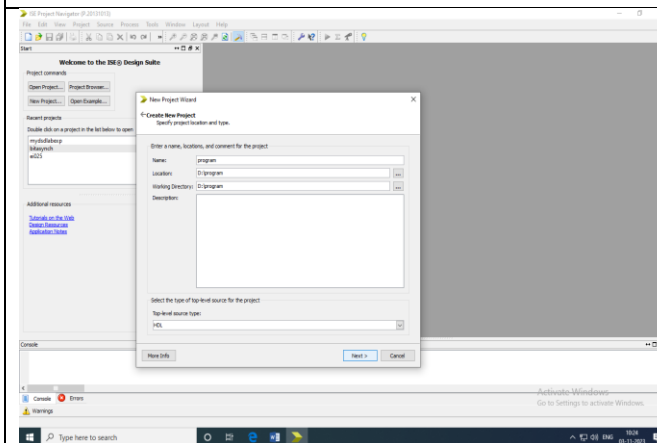




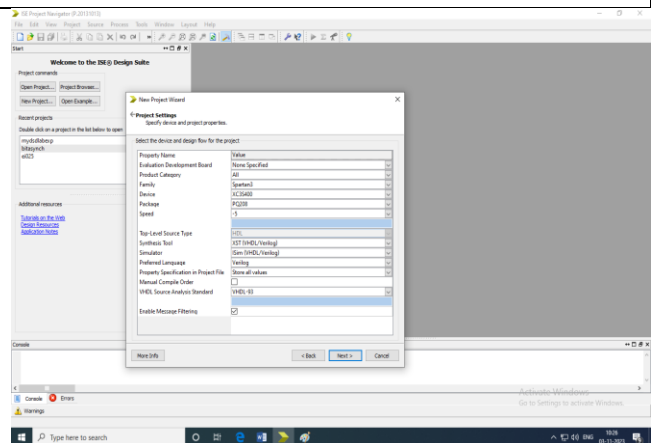
Step 1: open ISE Design suite 14.7



Step 2: File – New project



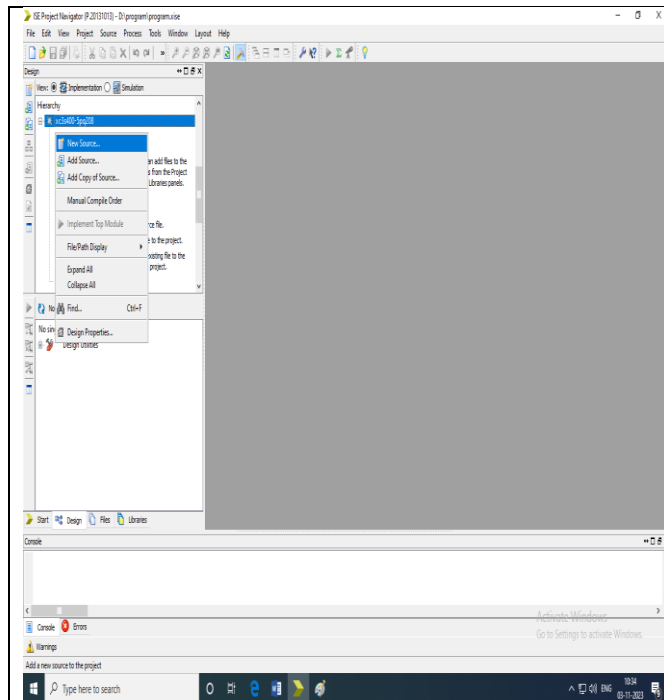
Step 3: Give file name, select location D drive and click next



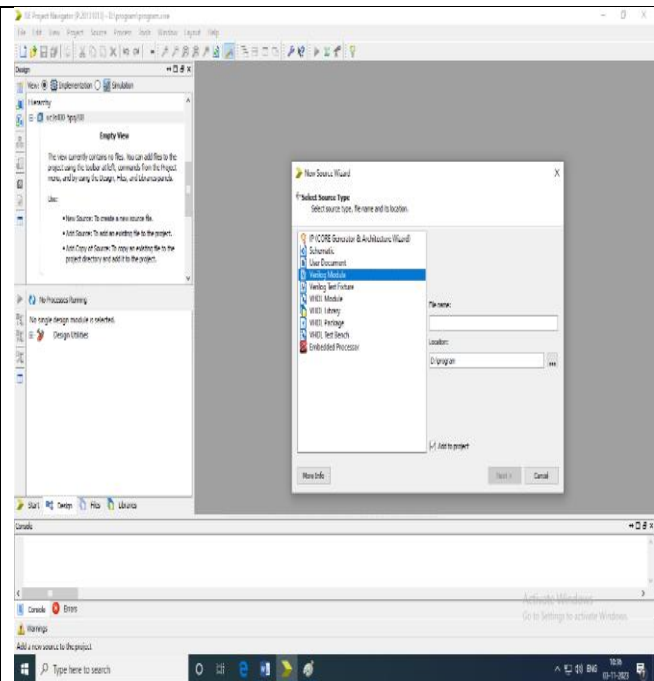
Step 4: Select Family-Spartan3, Device-xc3s400, Package -PQ208, Speed -5, Simulator-ISIM, Language- Verilog

Step 5: click next and finish

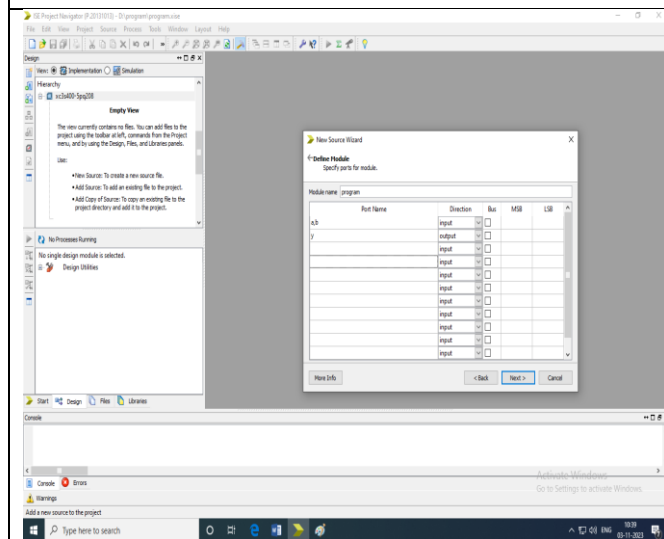
## Digital Circuit Design Lab Manual (ETL37)



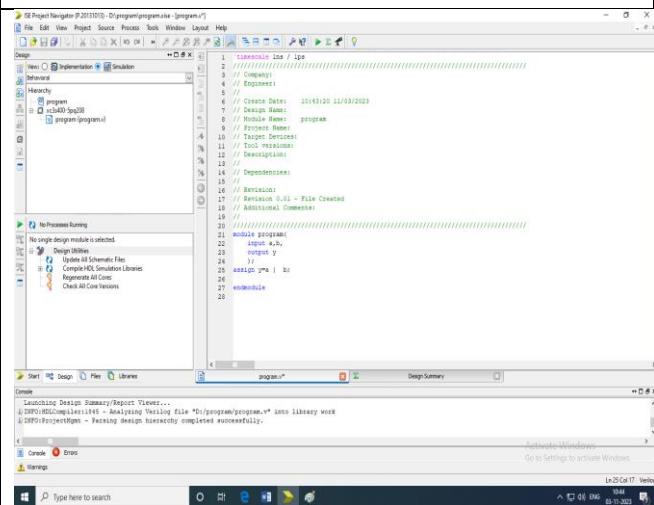
Step 6: Right click on xc3s400 new source



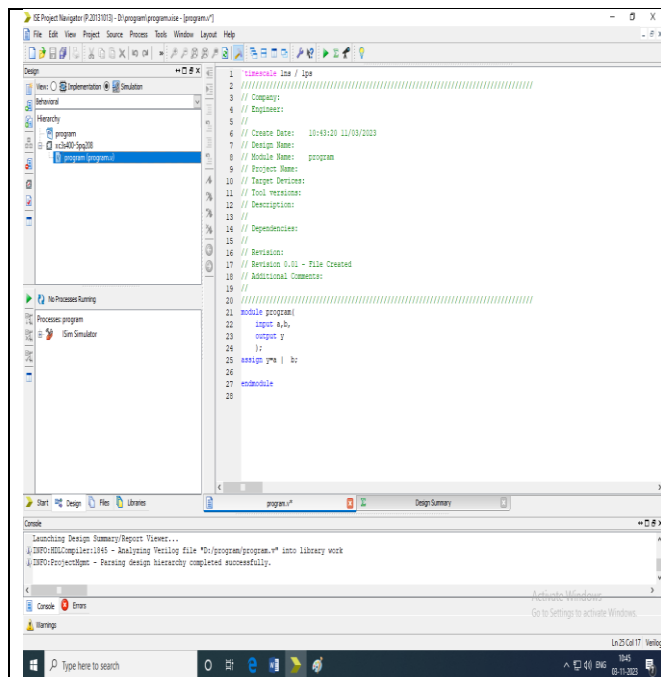
Step 7: Select Verilog module, give file name, click next



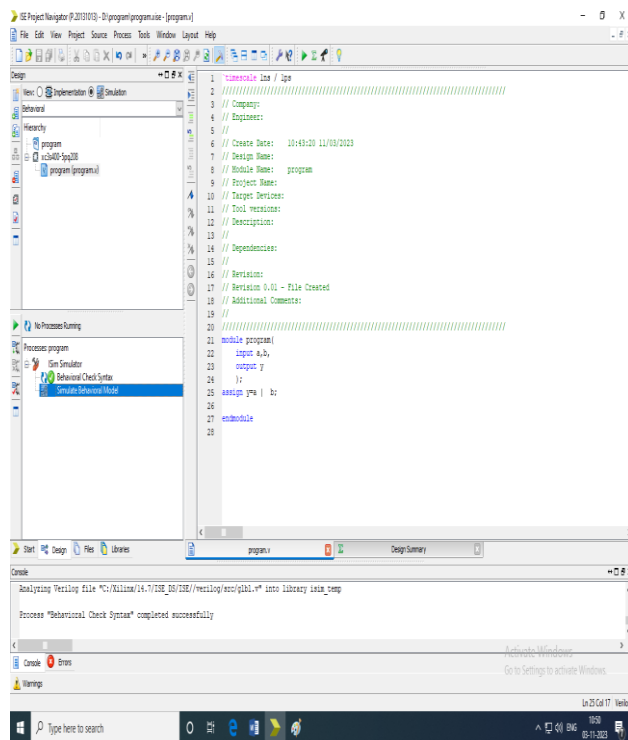
Step 8: Declare ports select direction input, output, if necessary MSB, LSB and, next, finish



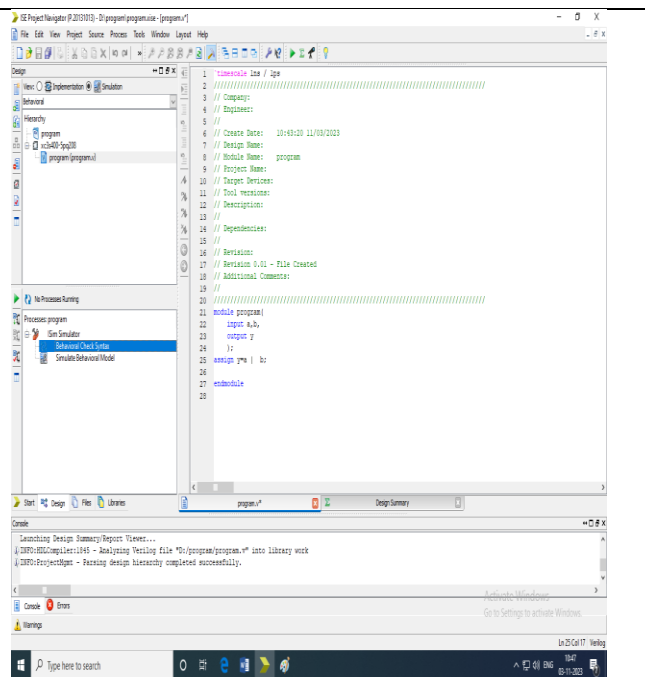
Step 9: Wright program select simulation



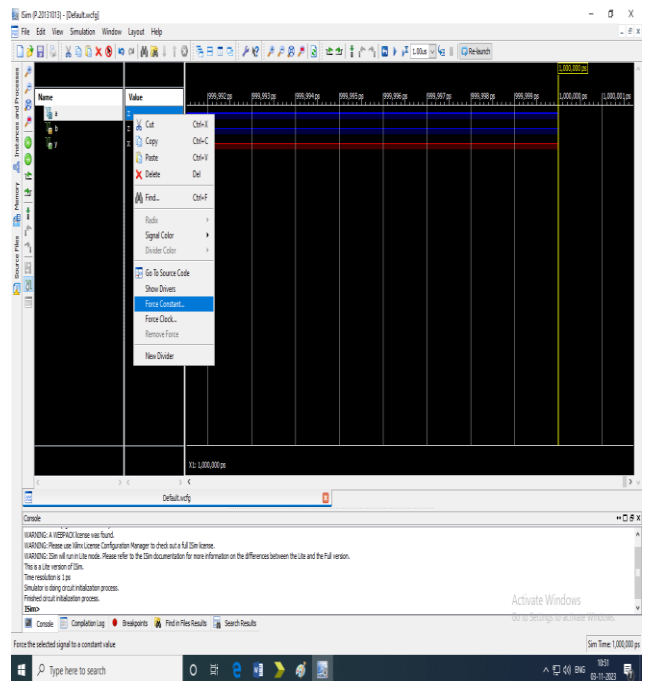
Step 10: select your program



Step 12: Double click on simulate behavioral Model, new window will be open.

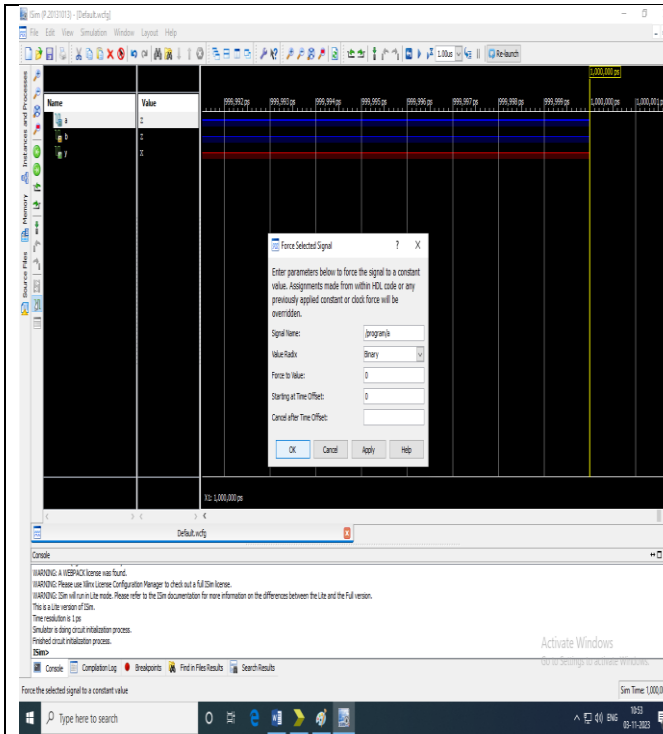


Step 11: Double click on behavioral check syntax under Isim Simulator, if are any error clear it.

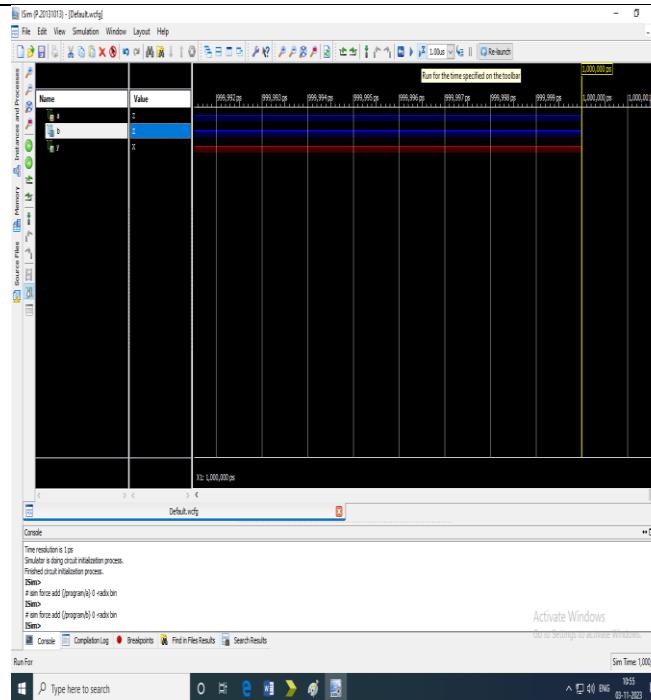


Step 13: Right click on value-force constant.

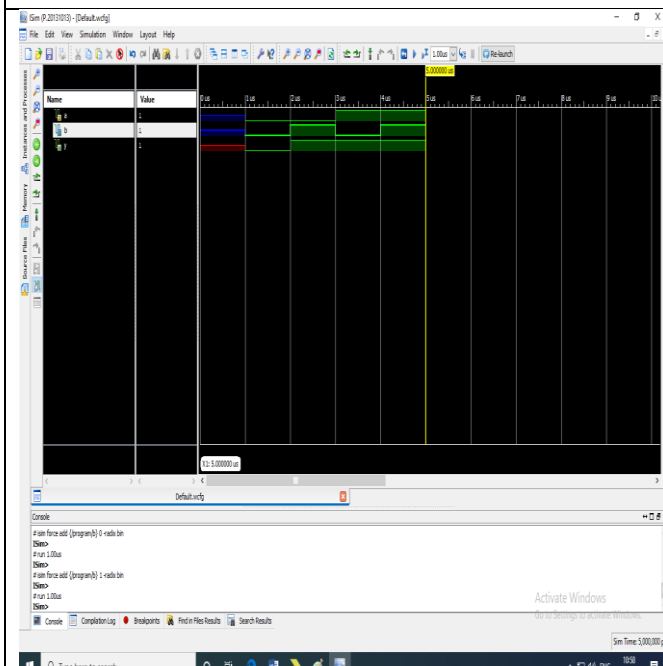
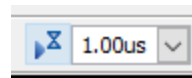
## Digital Circuit Design Lab Manual (ETL37)



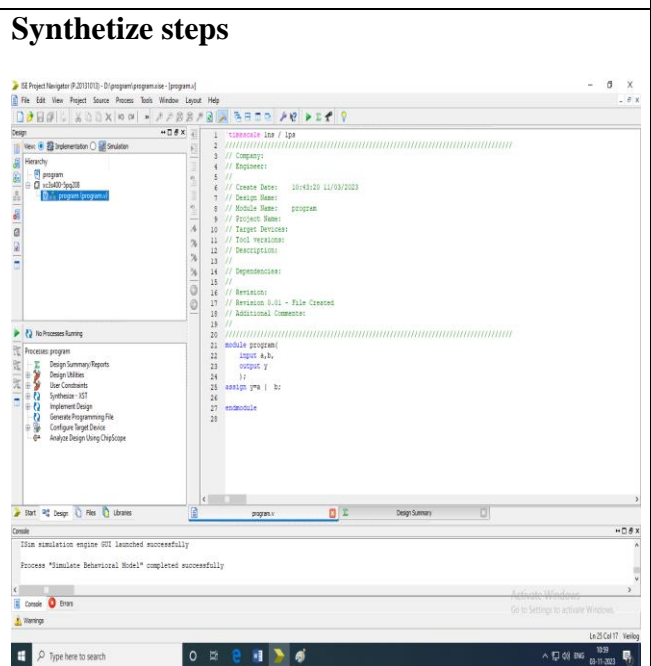
Step 13: force value as per your truth table then click ok.



Step 14: click on run for time specified time.

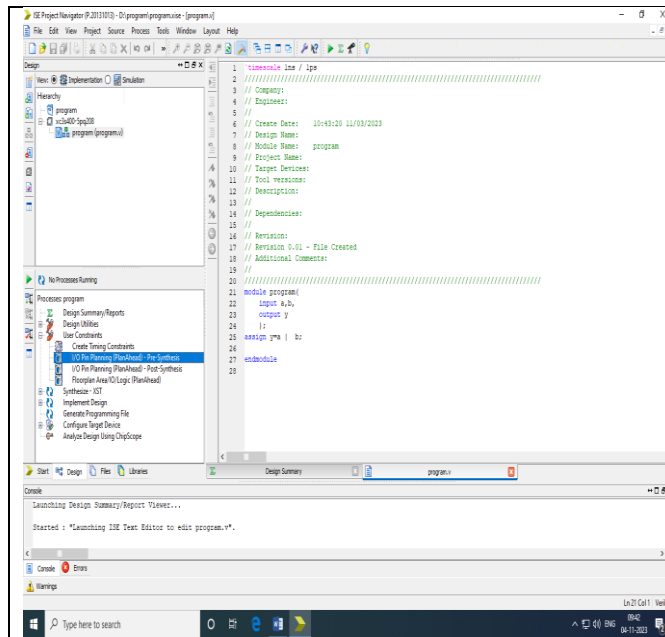


### Step 15: Observe wave form

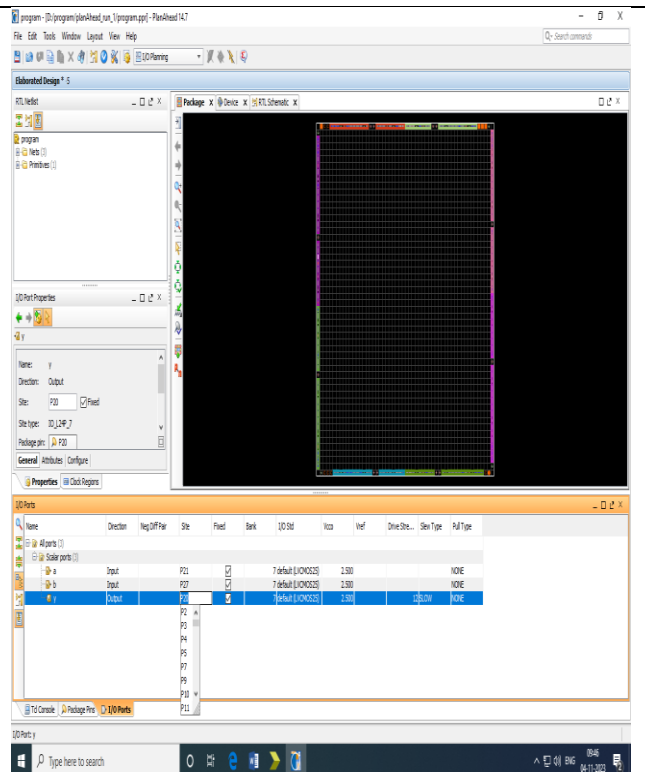


Step 16: click on implementation.

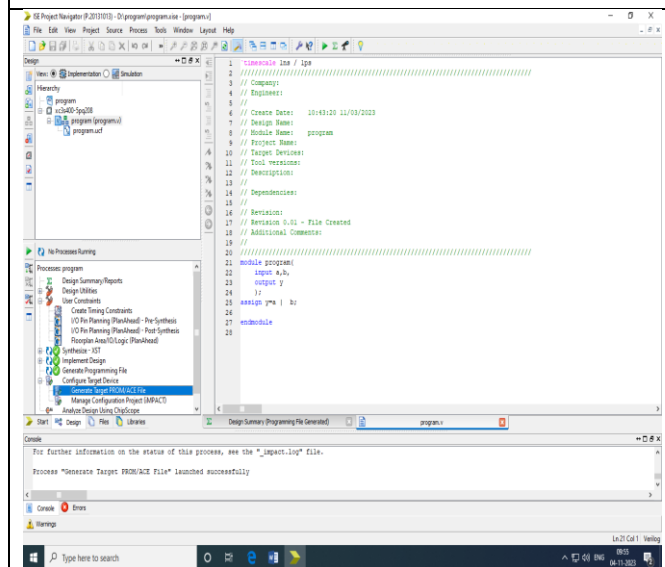
## Digital Circuit Design Lab Manual (ETL37)



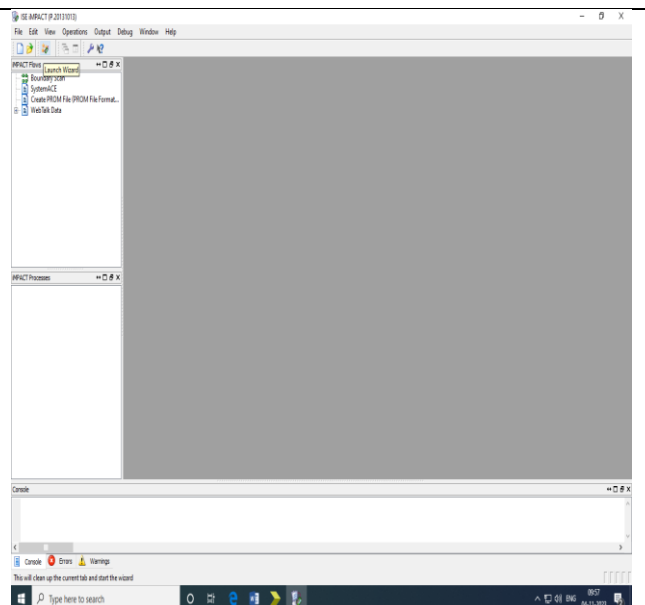
Step 17: click on I/o Planning under user Constraints, new window will be open yes.



Step 17: Declare the input and output pin by referring to pin detail data sheet xc3s400 then save and close.

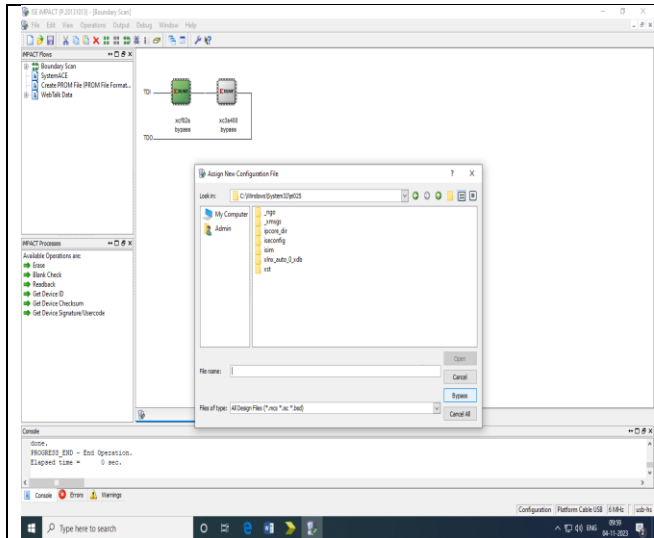


Step 18: Double click on generate target PROM/AEC File under Configure Target Device new will be open.

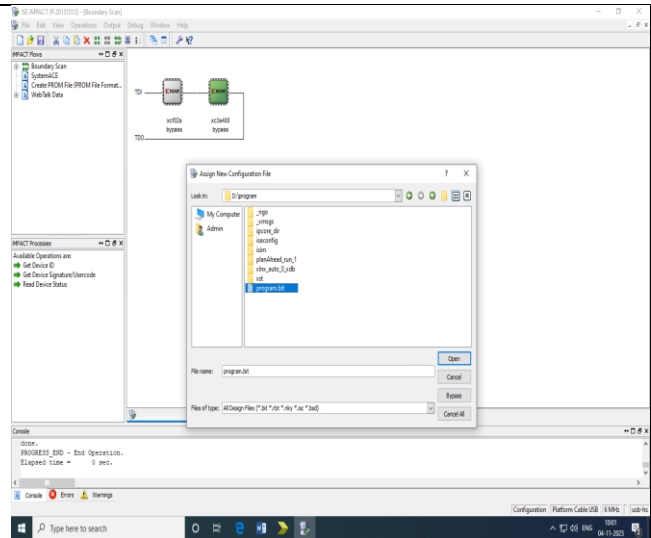


Step 18: Click on Launch wizard ok, yes.

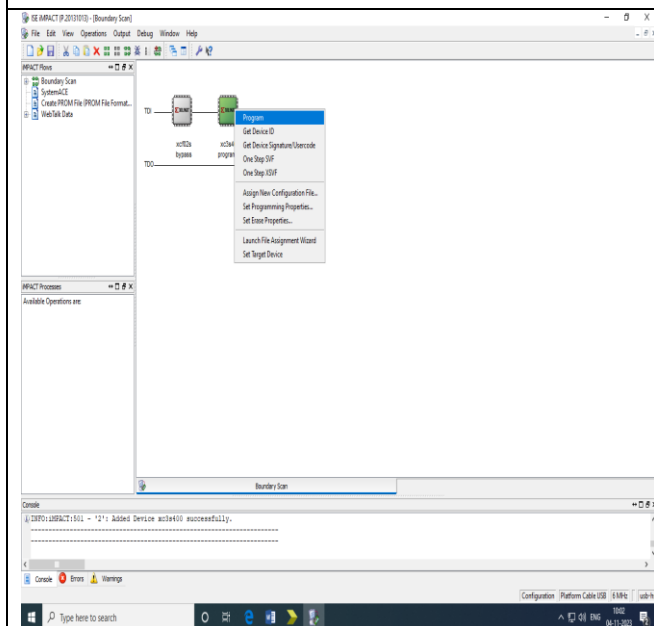
## Digital Circuit Design Lab Manual (ETL37)



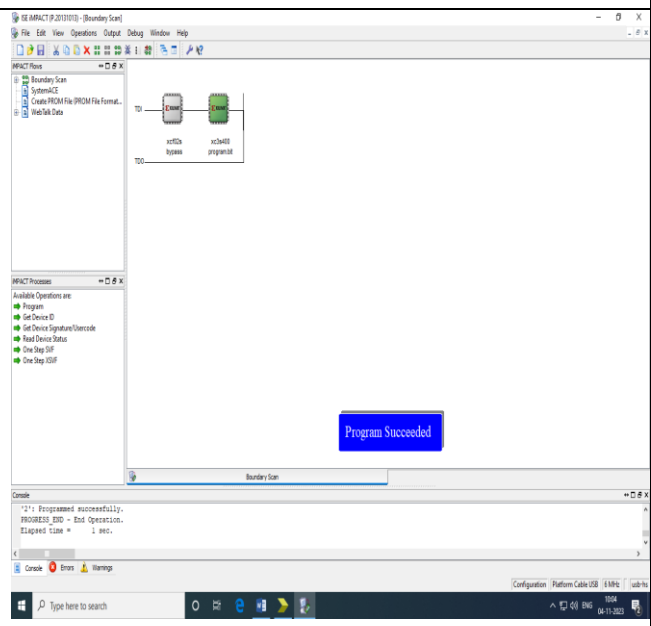
Step 19: Click on bypass, yes.



Step 19: go to your location select your bit file then click open, then ok.



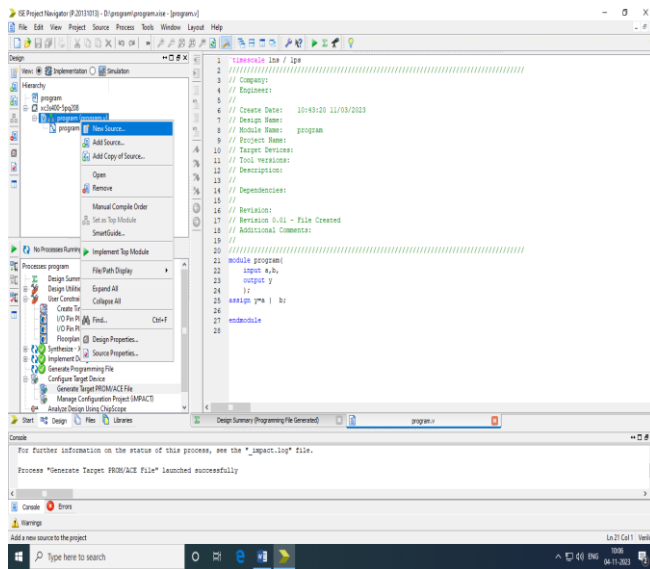
Step 19: right click on 3s400 select program, ok.



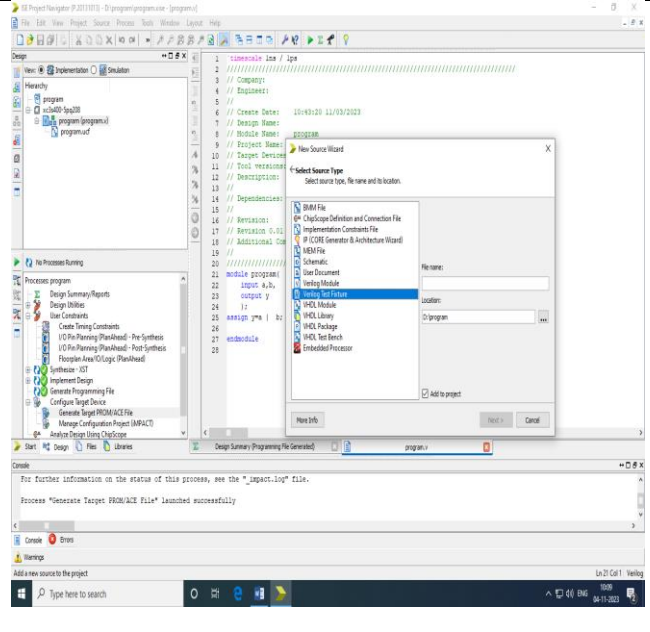
Step 19: after receiving program is succeeded very the same on FPGA KIT.



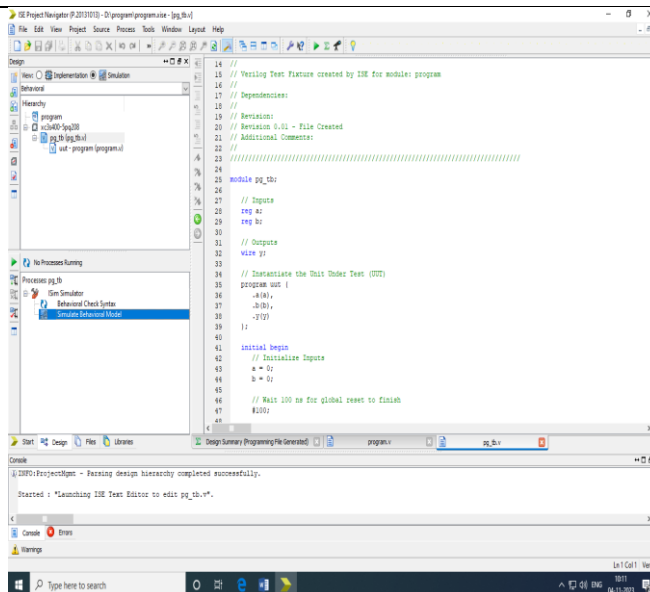
## Test Bench Steps



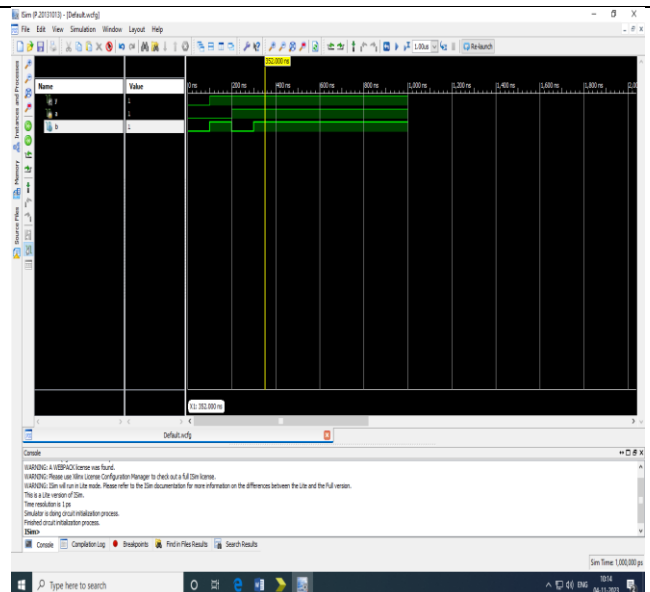
Step 20: Right click on program new source.



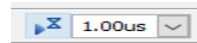
Step 20: Verilog text Fixture, give file name next, next finish.



Step 21: select simulation and then Double click simulate Behavioral model under isim simulator.



Step 22: click on run for time specified time.



Observe wave form.

**Pin Details of 3S400 Data Sheet****2. INPUT DIP SWITCHES**

There are 4 DIP (DIP1 to DIP4) switches provided on the baseboard which are connected to 32 input LEDs (LG1 to LG32) and to the connectors connecting the Daughter Board. DIP switches are represented on baseboard from DIP1 through DIP4. When the drag button of the dip switch is kept in the **ON** side, the input is high (1) the corresponding O/p LED will go **ON** and when the drag button of the dip switch is kept in the **OFF** side the input is low (0) and the corresponding O/p LED will go **OFF**.

**DIP1 (1) to DIP1 (8)** are represented as **IN1** to **IN8**, **DIP2 (1) to DIP2 (8)** are represented as **IN9** to **IN16**, **DIP3 (1) and DIP3 (8)** are represented as **IN17** to **IN24** & **DIP4 (1) and DIP4 (8)** are represented as **IN25** to **IN32** respectively. If IN1 is ON LED LG1 will be ON otherwise LG1 will be OFF and so on.

**PIN DETAILS**

UCF	Connector pin	3S50 IC Pins	3S400 IC Pins	XC9572 IC Pins
INPUT<0>	CN9/20	P21	P21	P31
INPUT<1>	CN9/22	P27	P27	P33
INPUT<2>	CN9/24	P29	P29	P35
INPUT<3>	CN9/26	P35	P35	P37
INPUT<4>	CN9/28	P37	P37	P40
INPUT<5>	CN9/30	P40	P40	P43
INPUT<6>	CN9/32	P43	P43	P45
INPUT<7>	CN9/34	P45	P45	P47
INPUT<8>	CN9/36	P48	P48	P50
INPUT<9>	CN9/38	P52	P52	P52
INPUT<10>	CN10/6	P58	P58	P54
INPUT<11>	CN10/8	P62	P62	P56
INPUT<12>	CN10/10	P64	P64	NA
INPUT<13>	CN10/12	P67	P67	NA
INPUT<14>	CN10/14	P71	P71	NA
INPUT<15>	CN10/16	P74	P74	NA
INPUT<16>	CN10/24	P80	P80	NA
INPUT<17>	CN11/37	P162	P162	NA
INPUT<18>	CN10/28	P86	P86	NA
INPUT<19>	CN10/30	P90	P90	NA
INPUT<20>	CN10/32	P94	P94	NA
INPUT<21>	CN10/34	P100	P100	NA
INPUT<22>	CN10/36	P102	P102	NA
INPUT<23>	CN11/4	P107	P107	NA
INPUT<24>	CN11/6	P113	P113	NA
INPUT<25>	CN11/8	P115	P115	NA
INPUT<26>	CN11/10	P117	P117	NA
INPUT<27>	CN11/12	P120	P120	NA
INPUT<28>	CN11/14	P123	P123	NA
INPUT<29>	CN11/16	P125	P125	NA
INPUT<30>	CN11/18	P131	P131	NA
INPUT<31>	CN11/20	P133	P133	NA

**Output LEDs**

There are 32 O/P LEDs (LR1 to LR32) which are connected to FPGA/CPLD output pins. These LEDs are active high type, i.e., they switch ON to indicate a logical '1' status.

**PIN DETAILS**

UCF	Connector PIN	3S50 IC pins	3S400 IC pins	XC9572 IC pins
OUTPUT<0>	CN9/19	P20	P20	P26
OUTPUT<1>	CN9/21	P26	P26	P32
OUTPUT<2>	CN9/23	P28	P28	P34
OUTPUT<3>	CN9/25	P34	P34	P36
OUTPUT<4>	CN9/27	P36	P36	P39
OUTPUT<5>	CN9/29	P39	P39	P41
OUTPUT<6>	CN9/31	P42	P42	P44
OUTPUT<7>	CN9/33	P44	P44	P46
OUTPUT<8>	CN9/35	P46	P46	P48
OUTPUT<9>	CN9/37	P51	P51	P51
OUTPUT<10>	CN10/5	P57	P57	P53
OUTPUT<11>	CN10/7	P61	P61	P55
OUTPUT<12>	CN10/9	P63	P63	NA
OUTPUT<13>	CN10/11	P65	P65	NA
OUTPUT<14>	CN10/13	P68	P68	NA
OUTPUT<15>	CN10/15	P72	P72	NA
OUTPUT<16>	CN10/23	P78	P78	NA
OUTPUT<17>	CN10/25	P81	P81	NA
OUTPUT<18>	CN10/27	P85	P85	NA
OUTPUT<19>	CN10/29	P87	P87	NA
OUTPUT<20>	CN10/31	P93	P93	NA
OUTPUT<21>	CN10/33	P95	P95	NA
OUTPUT<22>	CN10/35	P101	P101	NA
OUTPUT<23>	CN11/3	P106	P106	NA
OUTPUT<24>	CN11/5	P111	P111	NA
OUTPUT<25>	CN11/7	P114	P114	NA
OUTPUT<26>	CN11/9	P116	P116	NA
OUTPUT<27>	CN11/11	P119	P119	NA
OUTPUT<28>	CN11/13	P122	P122	NA
OUTPUT<29>	CN11/15	P124	P124	NA
OUTPUT<30>	CN11/17	P130	P130	NA
OUTPUT<31>	CN11/19	P132	P132	NA

**NOTE:** Download **sw\_led\_test.bit/jed** files for FPGA/CPLD

**CONNECTIONS:** Connect 26 pin FRC cable between J1 to J3 on SPARTAN6 DB.

**RESULT:** DIP1 to DIP4 are input pins & LR1 to LR32 used to see the corresponding output in LEDs.

## 6. SPEED & DIRECTION CONTROL OF DC MOTOR

PWM method is used to run the DC motor. First 3 pins of DIP1 are used to control the speed of the DC motor. Switch (SW5) is used to control the direction of DC motor. DC motor should be connected to the 2-pin Relimate connector RM2.

**CONNECTIONS:** Connect 10 pin FRC cable from CN22 to CN24.  
Connect 26 pin FRC cable between J1 to J2 on SPARTAN6 DB.



### PIN DETAILS

UCF	Connector pin	3S50 IC PINS	3S400 IC PINS	XC9572 IC PINS
CLK	CN10/18	P79	P79	P9
PDCM	CN12/35	P203	P203	P10
PSW<0>	CN9/20	P21	P21	P31
PSW<1>	CN9/22	P27	P27	P33
PSW<2>	CN9/24	P29	P29	P35

**NOTE:** Download [dc\\_motor.bit/jed](#) files for FPGA/CPLD

**RESULT:** connect the DC motor to relaymate connector RM2 & by selecting the above explained switches we can vary the speed of DC motor & to change the direction of rotation of DC motor use switch SW5.

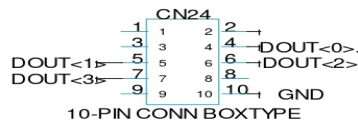
## 7. SPEED & DIRECTION CONTROL OF STEPPER MOTOR

**Note:** "We can run a Stepper motor which draws up to 300mA current.

**Caution:** "If it draws the current more than 300mA, the U1 IC will get heated".

Stepper motor can be interfaced directly to the power-mate connector PM1. Direction of the stepper motor can be changed by changing the DIP1 pin 1. Speed of the stepper motor can be controlled through the software by changing the counter value.

**CONNECTIONS:** Connect 10 pin FRC cable from CN22 to CN24.  
Connect 26 pin FRC cable between J1 to J2 on SPARTAN6 DB.



### PIN DETAILS

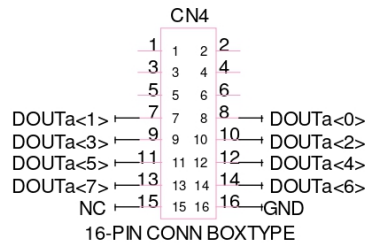
UCF	Connector pin	3S50 IC PIN	3S400 IC PIN	XC9572 IC PINS
CLK	CN10/18	P79	P79	P9
CNTRL	CN9/20	P21	P21	P31
DOUT<0>	CN12/14	P169	P169	P69
DOUT<1>	CN12/17	P175	P175	P72
DOUT<2>	CN12/18	P176	P176	P74
DOUT<3>	CN12/19	P178	P178	P75

Connection details of the power-mate connector PM1 are as follows:

Signal	PM1
VCC	PIN 1
DOUT<3>	PIN 2
DOUT<2>	PIN 3
DOUT<1>	PIN 4
DOUT<0>	PIN 5

**NOTE:** Download [stepper\\_motor.bit/jed](#) files for FPGA/CPLD

**RESULT:** connect the stepper motor to powermate connector PM1 & by selecting the switch DIP1/pin1 we can change the direction of rotation of stepper motor.

**PIN Details**

UCF	CONNECTOR PINS	XC3S50 IC PIN	XC3S400 IC PIN	XC9572 IC PIN
CLKIN	CN10/18	PIN 79	PIN 79	PIN 9
DOUTA0	CN12/26	PIN 187	PIN 187	PIN 83
DOUTA1	CN12/25	PIN 185	PIN 185	PIN 82
DOUTA2	CN12/28	PIN 190	PIN 190	PIN 1
DOUTA3	CN12/27	PIN 189	PIN 189	PIN 84
DOUTA4	CN12/30	PIN 194	PIN 194	PIN 3
DOUTA5	CN12/29	PIN 191	PIN 191	PIN 2
DOUTA6	CN12/32	PIN 197	PIN 197	PIN 5
DOUTA7	CN12/31	PIN 196	PIN 196	PIN 4

**NOTE:**

1. Download dacsqr.bit/jed file for FPGA/CPLD for Square wave
2. Download dacsine.bit/jed file for FPGA/CPLD for Sine wave
3. Download dacramp.bit/jed file for FPGA/CPLD for Ramp wave
4. Download dactri.bit/jed file for FPGA/CPLD for Triangle wave

**RESULT:** Observe the sine, square, ramp & triangular waves on CRO. Jumper JP4 is provided for UNI polar or Bi polar output.

**11. ELEVATOR INTERFACE****OPERATION:**

There are 4 keys on the interface board, which are used to send a request for the elevator. Each key is associated with one floor. Key request is acknowledged by a Red LED, which is a part of the hardware of the interface board.

There are 10 LEDs L1 to L10. L1, L4, L7 and L10 are green LEDs corresponding to the floor. Remaining 6 LEDs are amber in color, which indicate the shift position. The software generates a 4-bit code to switch on these LEDs. Whenever a key (SW1 to SW4) is pressed,

## **EXPERIMENT: 1**

### **REALIZATION AND VERIFICATION OF LOGIC GATES**

AIM: To study and verify the truth table of logic gates

LEARNING OBJECTIVE:

- ☐ Identify various ICs and their specification.

COMPONENTS REQUIRED:

- ☐ Logic gates (IC) trainer kit.
- ☐ Connecting patch chords.
- ☐ IC 7400, IC 7402, IC 7404, IC 7408, IC 7432, IC 7486.

THEORY:

The basic logic gates are the building blocks of more complex logic circuits. These logic gates perform the basic Boolean functions, such as AND, OR, NAND, NOR, Inversion, Exclusive-OR, Exclusive-NOR. Fig. below shows the circuit symbol, Boolean function, and truth. It is seen from the Fig that each gate has one or two binary inputs, A and B, and one binary output, C. The small circle on the output of the circuit symbols designates the logic complement. The AND, OR, NAND, and NOR gates can be extended to have more than two inputs. A gate can be extended to have multiple inputs if the binary operation it represents is commutative and associative.

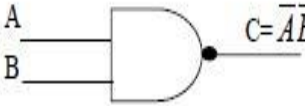
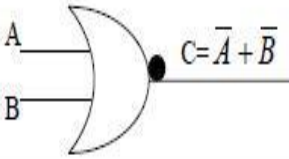
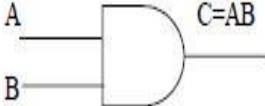
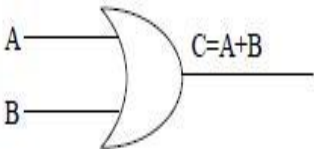
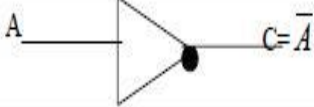
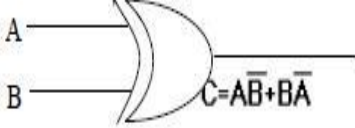
These basic logic gates are implemented as small-scale integrated circuits (SSICs) or as part of more complex medium scale (MSI) or very large-scale (VLSI) integrated circuits. Digital IC gates are classified not only by their logic operation, but also the specific logic-circuit family to which they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

TTL    Transistor-transistor logic  
ECL    Emitter-coupled logic  
MOS    Metal-oxide semiconductor  
CMOS   Complementary metal-oxide semiconductor

TTL and ECL are based upon bipolar transistors. TTL has a well-established popularity among logic families. ECL is used only in systems requiring high-speed operation. MOS and CMOS, are based on field effect transistors. They are widely used in large scale integrated circuits because of their high component density and relatively low power consumption. CMOS logic consumes far less power than MOS logic. There are various commercial integrated circuit chips available. TTL ICs are usually distinguished by numerical designation as the 5400 and 7400 series

PROCEDURE:

1. Check the components for their working.
2. Insert the appropriate IC into the IC base.
3. Make connections as shown in the circuit diagram.
4. Provide the input data via the input switches and observe the output on output LEDs

S.NO	GATE	SYMBOL	INPUTS		OUTPUT
			A	B	
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

**VIVA QUESTIONS:**

Why NAND & NOR gates are called universal gates?

Realize the EX – OR gates using minimum number of NAND gates.

Give the truth table for EX-NOR and realize using NAND gates?

What are the logic low and High levels of TTL IC's and CMOS IC's?

Compare TTL logic family with CMOS family?

Which logic family is fastest and which has low power dissipation?

## REALIZATION OF A BOOLEAN FUNCTION.

AIM: To simplify the given expression and to realize it using Basic gates and Universal gates

### LEARNING OBJECTIVE:

To simplify the Boolean expression and to build the logic circuit.

Given a Truth table to derive the Boolean expressions and build the logic circuit to realize it.

### COMPONENTS REQUIRED:

IC 7400, IC 7408, IC 7432, IC 7406, IC 7402, Patch Cords & IC Trainer Kit.

### THEORY:

*Canonical Forms (Normal Forms):* Any Boolean function can be written in disjunctive normal form (sum of min-terms) or conjunctive normal form (product of max-terms).

A Boolean function can be represented by a Karnaugh map in which each cell corresponds to a minterm. The cells are arranged in such a way that any two immediately adjacent cells correspond to two minterms of distance 1. There is more than one way to construct a map with this property.

### **Karnaugh Maps**

For a function of two variables, say,  $f(x, y)$ ,

	$x'$	$x$
$y'$	$f(0,0)$	$f(1,0)$
$y$	$f(0,1)$	$f(1,1)$

For a function of three variables, say,  $f(x, y, z)$

	$x'y'$	$x'y$	$xy$	$xy'$
$z'$	$f(0,0,0)$	$f(0,1,0)$	$f(1,1,0)$	$f(1,0,0)$
$z$	$f(0,0,1)$	$f(0,1,1)$	$f(1,1,1)$	$f(1,0,1)$

For a function of four variables:  $f(w, x, y, z)$

	$w'x'$	$w'x$	$wx$	$wx'$
$y'z'$	0	4	12	8
$y'z$	1	5	13	9
$yz$	3	7	15	11
$yz'$	2	6	14	10



Realization of Boolean expression:

$$1) \quad Y = \bar{A} \bar{B} C \bar{D} + \bar{A} B C \bar{D} + A B C \bar{D} + A \bar{B} C \bar{D} + A \bar{B} \bar{C} \bar{D} + A \bar{B} \bar{C} D + A \bar{B} C D$$

AB

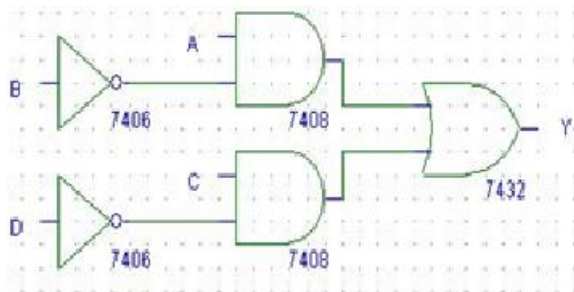
			1	
			1	
			1	
1	1	1	1	

After simplifying using K-Map method we  
get Realization using Basic gates

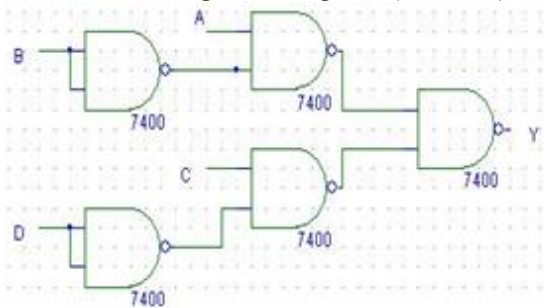
$$Y = \bar{A} B + C D$$

**TRUTH TABLE**

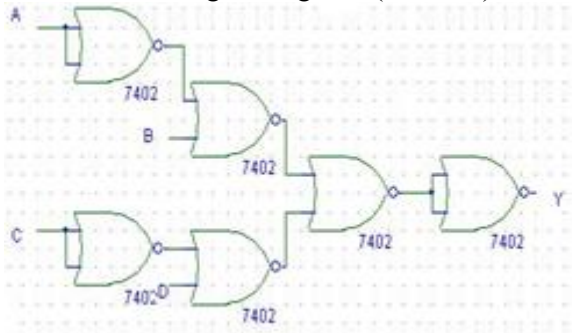
INPUTS				OUTPUT
A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0



Realization using NAND gates (IC 7400)



Realization using NOR gates. (IC 7402)



2) For the given Truth Table, realize a logical circuit using basic gates and NAND gates

Inputs				Output
A	B	C	D	Y
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

#### PROCEDURE:

Check the components for their working.

Insert the appropriate IC into the IC base.

Make connections as shown in the circuit diagram.

Provide the input data via the input switches and observe the output on output

LEDs Verify the Truth Table

RESULT: Simplified and verified the Boolean function using basic gates and universal gates

#### VIVA QUESTIONS:

- 1) What are the different methods to obtain minimal expression?
- 2) What is a Min term and Max term
- 3) State the difference between SOP and POS.
- 4) What is meant by canonical representation?
- 5) What is K-map? Why is it used?
- 6) What are universal gates?

VERILOG MODULE FOR BASIC GATES (DATA FLOW)

```
module basic_gates(input a, input b, output [7:0]y);
assign y[0] = a & b;
assign y[1] = ~(a & b);
assign y[2] = a | b;
assign y[3] = ~(a | b);
assign y[4] = a ^ b;
assign y[5] = ~(a ^ b);
assign y[6] = ~ a;
assign y[7] = ~ b;
endmodule
```

BASIC GATES (GATE LEVEL)

```
module basic_gates(input a, input b, output [7:0]y);
and a1(y[0], a, b);
nand a2(y[1], a, b);
or a3(y[2], a, b);
nor a4(y[3], a, b);
xor a5(y[4], a, b);
xnor a6(y[5], a, b);
not a7(y[6], a);
not a8(y[7], b);
endmodule
```



## EXPERIMENT: 2

### ADDERS AND SUBTRACTORS

AIM: To realize

- i) Half Adder and Full Adder
- ii) Half subtraction and Full Subtraction by using Basic gates and NAND gates

LEARNING OBJECTIVE:

- To realize the adder and subtraction circuits using basic gates and universal gates to realize full adder using two half adders
- To realize a full subtraction using two half subtractions

COMPONENTS REQUIRED:

IC 7400, IC 7408, IC 7486, IC 7432, Patch Cords & IC Trainer Kit.

THEORY:

*Half-Adder:* A combinational logic circuit that performs the addition of two data bits, A and B, is called a half-adder. Addition will result in two output bits; one of which is the sum bit, S, and the other is the carry bit, C. The Boolean functions describing the half-adder are:

$$S = A \oplus B$$

$$C = A B$$

*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. A combinational logic circuit that adds two data bits, A and B, and a carry-in bit, Cin, is called a full-adder. The Boolean functions describing the full-adder are:

$$S = (x \oplus y) \oplus \text{Cin}$$

$$C = xy + \text{Cin} (x \oplus y)$$

*Half Subtractor:* Subtracting a single-bit binary value B from another A (i.e. A - B) produces a difference bit D and a borrow out bit B-out. This operation is called half subtraction and the circuit to realize it is called a half subtractor. The Boolean functions describing the half-Subtractor are:

$$S = A \oplus B$$

$$C = A' B$$

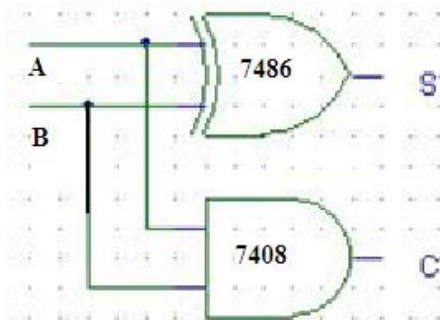
*Full Subtractor:* Subtracting two single-bit binary values, B, Cin from a single-bit value A produces a difference bit D and a borrow out Br bit. This is called full subtraction. The Boolean functions describing the full-subtractor are:

$$D = (x \oplus y) \oplus \text{Cin}$$

$$\text{Br} = A' B + A' (\text{Cin}) + B(\text{Cin})$$

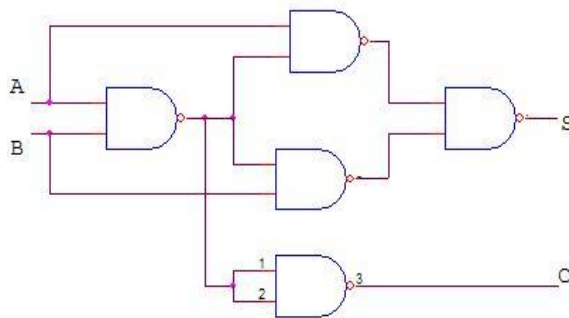
**I. TO REALIZE HALF ADDER****TRUTH TABLE**

INPUTS		OUTPUTS	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

**i) Basic Gates****BOOLEAN EXPRESSIONS:**

$$S = A \oplus B$$

$$C = A B$$

**ii) NAND Gates****II. FULL ADDER****TRUTH TABLE**

INPUTS			OUTPUTS	
A	B	Cin	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

```

module ha2(input a,
input b, output sum,
output carry );
xor a1( sum, a,
b);
and a2( carry,
a, b);
endmodule

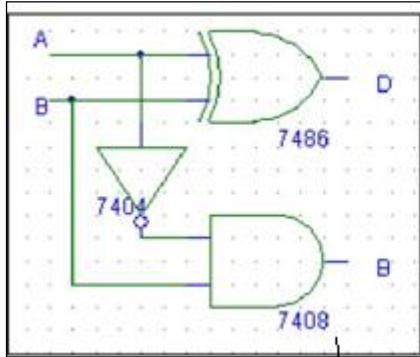
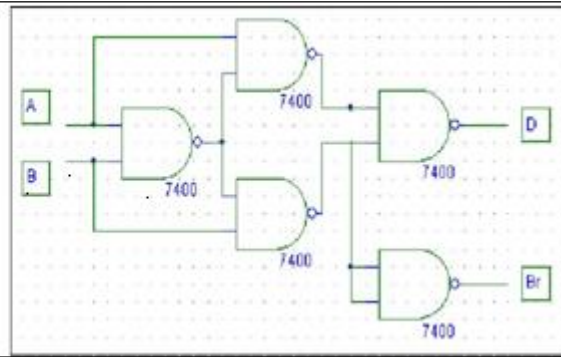
```

**BOOLEAN EXPRESSIONS:**

$$S = A \oplus B \oplus C$$

$$C = A B + B C_{in} + A C_{in}$$



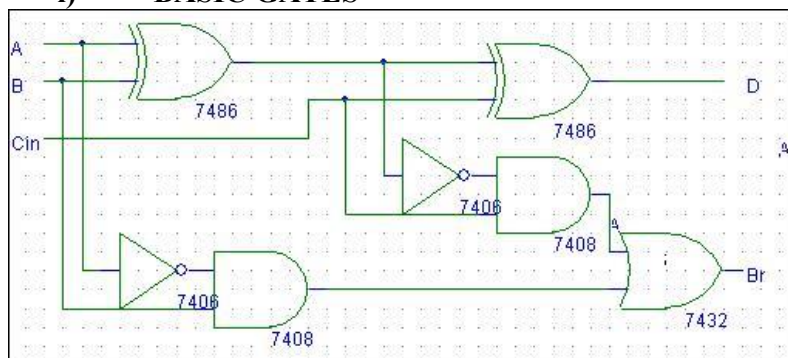
**i) BASIC GATES****ii) NAND Gates (IC 7400)****IV. FULL SUBTRACTOR****TRUTH TABLE**

INPUTS			OUTPUTS	
A	B	Cin	D	Br
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

**BOOLEAN EXPRESSIONS:**

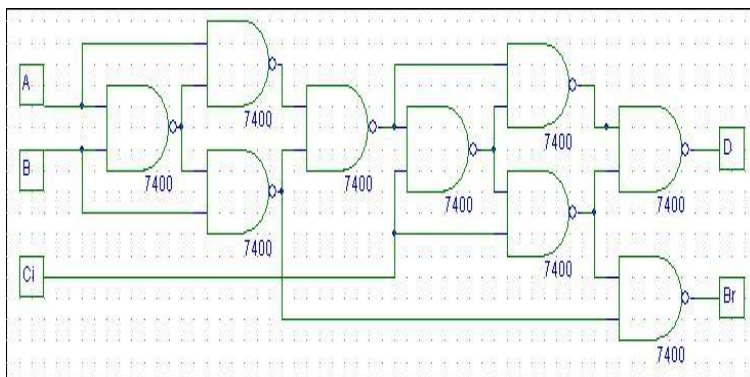
$$D = A \oplus B \oplus C$$

$$Br = \bar{A} B + B Cin + \bar{A} Cin$$

**i) BASIC GATES**



**ii) To Realize the Full subtractor using NAND Gates only. (IC 7400)**



```
module fulladder(a,b,c, sum,carry);
```

```
    input a,b,c;
```

```
    output sum,carry;
```

```
    assign sum=a^b^c;
```

```
    assign carry=(a&b)|(b&c)|(a&c);
```

```
endmodule
```

**PROCEDURE:**

- ☐ Check the components for their working.
- ☐ Insert the appropriate IC into the IC base.
- ☐ Make connections as shown in the circuit diagram.
- ☐ Verify the Truth Table and observe the outputs

**RESULT:** The truth table of the above circuits is

verified.

**VIVA QUESTIONS:**

- 1) What is a half adder?
- 2) What is a full adder?
- 3) What are the applications of adders?
- 4) What is a half subtractor?
- 5) What is a full subtractor?
- 6) What are the applications of subtractors?
- 7) Obtain the minimal expression for above circuits.
- 8) Realize a full adder using two half adders
- 9) Realize a full subtractors using two half subtractors



## EXPERIMENT: 3

### PARALLEL ADDER AND SUBTRACTOR

**AIM:** To design and set up the following circuit using IC 7483.

- i) A 4-bit binary parallel adder.
- ii) A 4-bit binary parallel subtractor.

#### LEARNING OBJECTIVE:

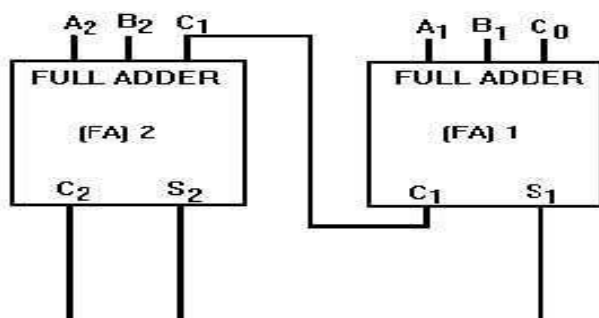
- To learn about IC 7483 and its internal structure.
- To realize a subtractor using adder IC 7483

#### COMPONENTS REQUIRED:

IC 7483, IC 7486, Patch Cords & IC Trainer Kit.

#### THEORY:

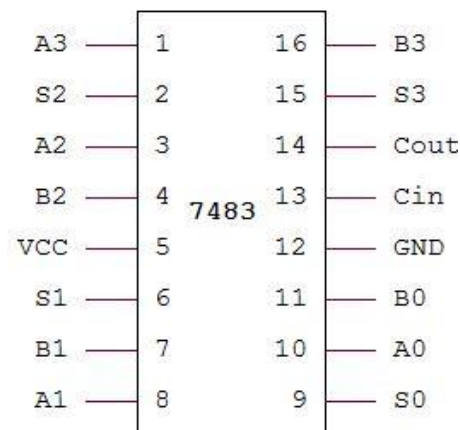
The Full adder can add single-digit binary numbers and carries. The largest sum that can be obtained using a full adder is 11<sub>2</sub>. Parallel adders can add multiple-digit numbers. If full adders are placed in parallel, we can add two- or four-digit numbers or any other size desired. Figure below uses STANDARD SYMBOLS to show a parallel adder capable of adding two, two-digit binary numbers. The addend would be on A inputs, and the augend on the B inputs. For this explanation we will assume there is no input to C<sub>0</sub> (carry from a previous circuit)



To add 10<sub>2</sub> (addend) and 01<sub>2</sub> (augend), the addend inputs will be 1 on A<sub>2</sub> and 0 on A<sub>1</sub>. The augend inputs will be 0 on B<sub>2</sub> and 1 on B<sub>1</sub>. Working from right to left, as we do in normal addition, let's calculate the outputs of each full adder. With A<sub>1</sub> at 0 and B<sub>1</sub> at 1, the output of adder1 will be a sum (S<sub>1</sub>) of 1 with no carry (C<sub>1</sub>). Since A<sub>2</sub> is 1 and B<sub>2</sub> is 0, we have a sum (S<sub>2</sub>) of 1 with no carry (C<sub>2</sub>) from adder1. To determine the sum, read the outputs (C<sub>2</sub>, S<sub>2</sub>, and S<sub>1</sub>) from left to right. In this case, C<sub>2</sub> = 0, S<sub>2</sub> = 1, and S<sub>1</sub> = 1. The sum, then, of 10<sub>2</sub> and 01<sub>2</sub>

is 0112. To add four bits we require four full adders arranged in parallel. IC 7483 is a 4- bit parallel adder whose pin diagram is shown.

	MSB				LSB
INPUTS					Cin
		A <sub>3</sub>	A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>
		B <sub>3</sub>	B <sub>2</sub>	B <sub>1</sub>	B <sub>0</sub>
OUTPUT	Cout	S <sub>3</sub>	S <sub>2</sub>	S <sub>1</sub>	S <sub>0</sub>



IC 7483 pin diagram

i) 4-Bit Binary Adder

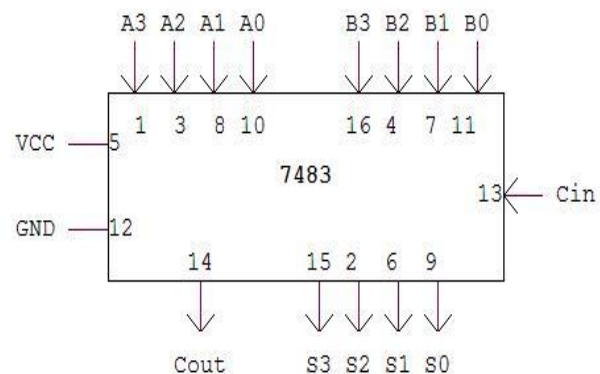
An Example: 7+2=11 (1001)

7 is realized at A<sub>3</sub> A<sub>2</sub> A<sub>1</sub> A<sub>0</sub> = 0111

2 is realized at B<sub>3</sub> B<sub>2</sub> B<sub>1</sub> B<sub>0</sub> = 0010

Sum = 1001

ADDER CIRCUIT:



**PROCEDURE:**

- ☐ Check all the components for their working.
- ☐ Insert the appropriate IC into the IC base.
- ☐ Make connections as shown in the circuit diagram.
- ☐ Apply augend and addend bits on A and B and cin=0.
- ☐ Verify the results and observe the outputs.

ii) **4-BIT BINARY SUBTRACTOR.**

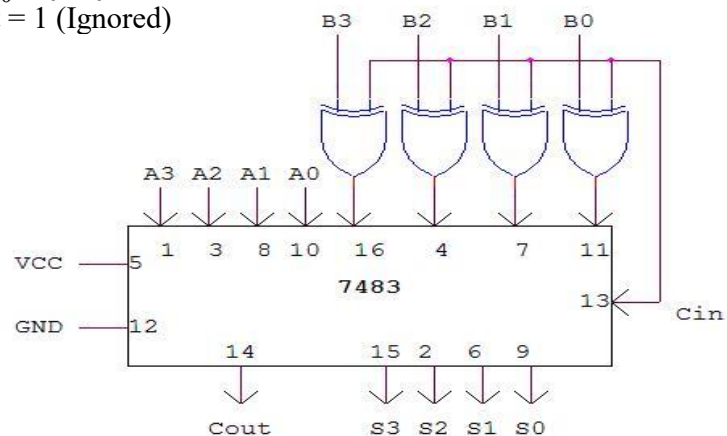
Subtraction is carried out by adding 2's complement of the subtrahend.

Example:  $8 - 3 = 5$  (0101)

- ☐ 8 is realized at A<sub>3</sub> A<sub>2</sub> A<sub>1</sub> A<sub>0</sub> = 1000
- ☐ 3 is realized at B<sub>3</sub> B<sub>2</sub> B<sub>1</sub> B<sub>0</sub> through X-OR gates = 0011
- ☐ Output of X-OR gate is 1's complement of 3 = 1100
- ☐ 2's Complement can be obtained by adding C<sub>in</sub> = 1

Therefore

$$\begin{array}{rcl}
 C_{in} & = & 1 \\
 A_3 A_2 A_1 A_0 & = & 1000 \\
 B_3 B_2 B_1 B_0 & = & 1100 \\
 S_3 S_2 S_1 S_0 & = & 0101 \\
 C_{out} & = & 1 \text{ (Ignored)}
 \end{array}$$

**PROCEDURE:**

- ☐ Check all the components for their working.
- ☐ Insert the appropriate IC into the IC base.
- ☐ Make connections as shown in the circuit diagram.
- ☐ Apply Minuend and subtrahend bits on A and B and cin=1.
- ☐ Verify the results and observe the outputs.

**RESULTS:** Verified the working of IC 7483 as adder and subtractor.

```
module pa4bit(input [4:1] a,input [4:1] b,input cin,output [4:1] s,output cout );
wire [3:1]t;
    fa f1 (a[1],b[1],cin,s[1],t[1]);
    fa f2 (a[2],b[2],t[1],s[2],t[2]) ;
    fa f3 (a[3],b[3],t[2],s[3],t[3]) ;
    fa f4 (a[4],b[4],t[3],s[4],cout) ;
endmodule
```

#### VIVA QUESTIONS:

- 1) What is the internal structure of 7483 IC?
- 2) What do you mean by code conversion?
- 3) What are the applications of code conversion?
- 4) How do you realize a subtractor using full adder?
- 5) What is a ripple Adder? What are its disadvantages?



## EXPERIMENT: 4

### BINARY TO GRAY CODE CONVERTER

AIM: To realize Binary to Gray code converter and vice versa

LEARNING OBJECTIVE:

To learn the importance of non-weighted code

To learn to generate gray code

COMPONENTS REQUIRED:

IC 7400, IC 7486, and IC 7408, Patch Cords & IC Trainer Kit

#### I) BINARY TO GRAY CONVERSION

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

$$G_3 = B_3$$

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

$$G_2 = B_3 \oplus B_2$$

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

$$G_1 = B_1 \oplus B_2$$

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

$$G_0 = B_1 \oplus B_0$$

Binary				Gray			
B3	B2	B1	B0	G3	G2	G1	G0
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



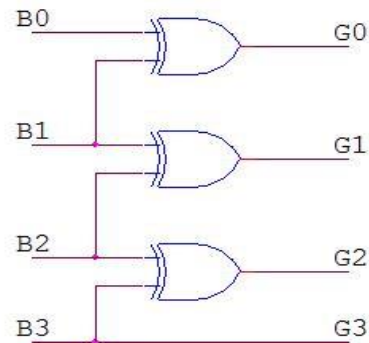
### BINARY TO GRAY CODE USING EX-OR GATES

*BOOLEAN EXPRESSIONS:*

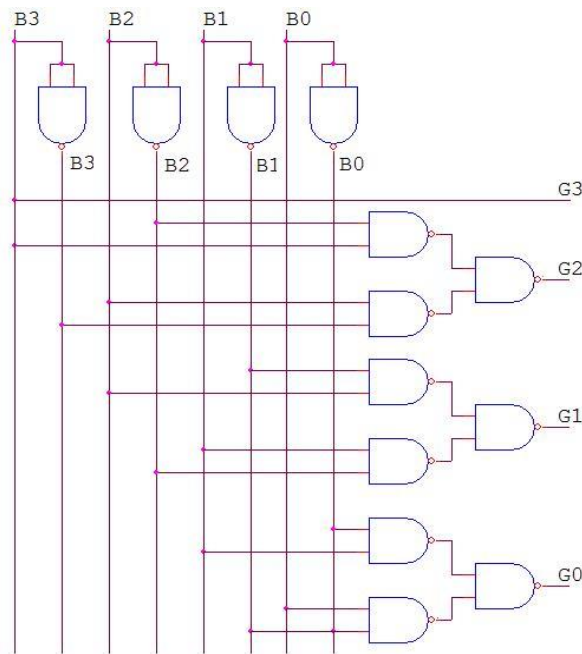
$$G_3 = B_3 \quad G_2 = B_3 \oplus B_2$$

$$G_1 = B_1 \oplus B_2$$

$$G_0 = B_1 \oplus B_0$$



REALIZATION USING NAND GATES: (IC 7400)



### II) GRAY TO BINARY CONVERSION

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

$$B_3 = G_3$$

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

$$B2 = G3 \oplus G2$$

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

$$B1 = G3 \oplus G2 \oplus G1$$

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

$$B0 = G3 \oplus G2 \oplus G1 \oplus G0$$

### BOOLEAN EXPRESSIONS:

$$B3 = G3$$

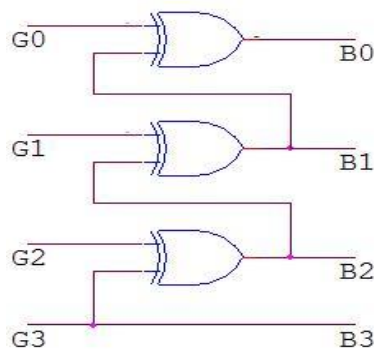
$$B2 = G3 \oplus G2$$

$$B1 = G3 \oplus G2 \oplus G1$$

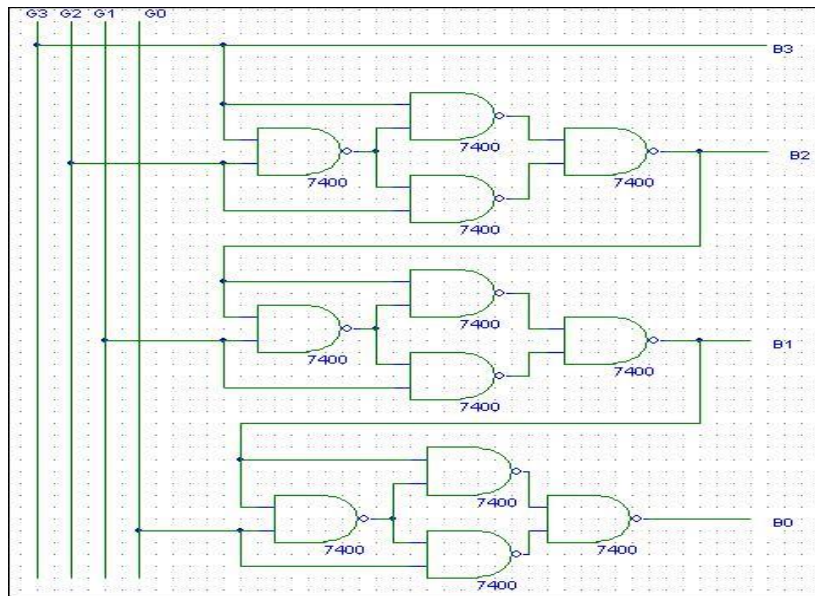
$$B0 = G3 \oplus G2 \oplus G1 \oplus G0$$

Gray				Binary			
G3	G2	G1	G0	B3	B2	B1	B0
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

### GRAY TO BINARY CODE CONVERSION USING EX-OR GATE



## REALIZATION USING NAND GATES: (IC 7400)

PROCEDURE:

- ☐ Check all the components for their working.
- ☐ Insert the appropriate IC into the IC base.
- ☐ Make connections as shown in the circuit diagram.
- ☐ Verify the Truth Table and observe the outputs.

RESULT: Binary to gray code conversion and vice versa is realized using EX-OR gates and NAND gates.

```

module btog(b, g);
    input [3:0] b;
    output [3:0] g;
    assign g[0]=b[1]^b[0];
    assign g[1]=b[2]^b[1];
    assign g[2]=b[3]^b[2];
    assign g[3]=b[3];
endmodule

```

VIVA QUESTIONS:

- 1) What are code converters?
- 2) What is the necessity of code conversions?
- 3) What is gray code?
- 4) Realize the Boolean expressions for
  - a) Binary to gray code conversion
  - b) Gray to binary code conversion



## EXPERIMENT: 5

### MULTIPLEXER AND DEMULTIPLEXER

**AIM:** To design and set up the following circuit

- 1) To design and set up a 4:1 Multiplexer (MUX) using only NAND gates.
- 2) To design and set up a 1:4 Demultiplexer (DE-MUX) using only NAND gates.
- 3) To verify the various functions of IC 74153 (MUX) and IC 74139 (DEMUX).
- 4) To set up a Half/Full Adder and Half/Full Subtractor using IC 74153.

**LEARNING OBJECTIVE:**

- To learn about various applications of multiplexer and de-multiplexer
- To learn and understand the working of IC 74153 and IC 74139
- To learn to realize any function using Multiplexer

**THEORY:**

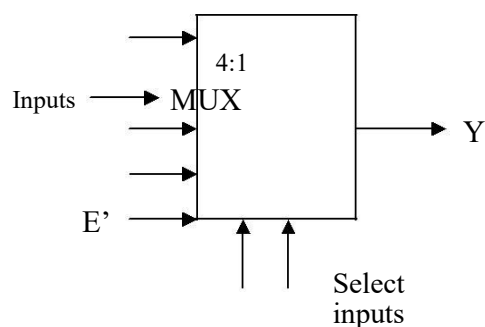
Multiplexers are very useful components in digital systems. They transfer a large number of information units over a smaller number of channels, (usually one channel) under the control of selection signals. Multiplexer means many to one. A multiplexer is a circuit with many inputs but only one output. By using control signals (select lines) we can select any input to the output. Multiplexer is also called as data selector because the output bit depends on the input data bit that is selected. The general multiplexer circuit has  $2^n$  input signals,  $n$  control/select signals and 1 output signal.

De-multiplexers perform the opposite function of multiplexers. They transfer a small number of information units (usually one unit) over a larger number of channels under the control of selection signals. The general de-multiplexer circuit has 1 input signal,  $n$  control/select signals and  $2^n$  output signals. De-multiplexer circuit can also be realized using a decoder circuit with enable.

**COMPONENTS REQUIRED:**

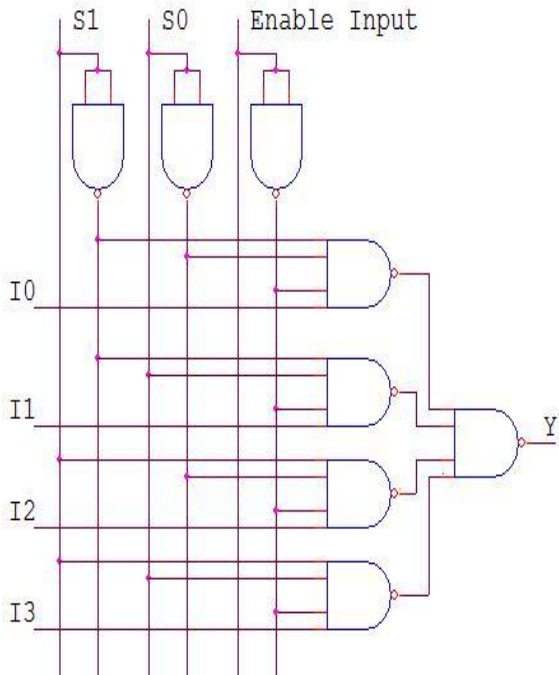
IC 7400, IC 7410, IC 7420, IC 7404, IC 74153, IC 74139, Patch Cords & IC Trainer Kit.

**i) 4:1 MULTIPLEXER**



$$\text{Output } Y = E'S_1'S_0'I_0 + E'S_1'S_0'I_1 + E'S_1S_0'I_2 + E'S_1S_0'I_3$$

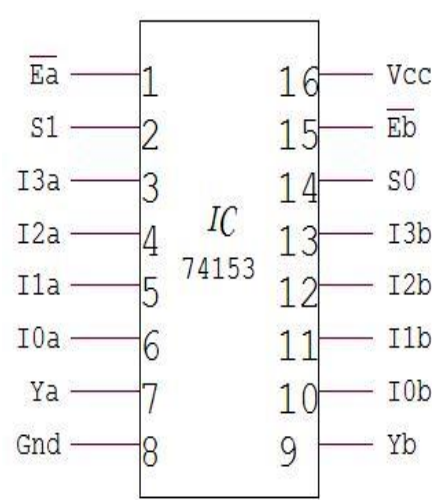
REALIZATION USING NAND GATES



TRUTH TABLE

Select Inputs		Enable Input	Inputs				Out puts
S1	S0	E	I0	I1	I2	I3	Y
X	X	1	X	X	X	X	0
0	0	0	0	X	X	X	0
0	0	0	1	X	X	X	1
0	1	0	X	0	X	X	0
0	1	0	X	1	X	X	1
1	0	0	X	X	0	X	0
1	0	0	X	X	1	X	1
1	1	0	X	X	X	0	0
1	1	0	X	X	X	1	1

VERIFY IC 74153 MUX (DUAL 4:1 MULTIPLEXE

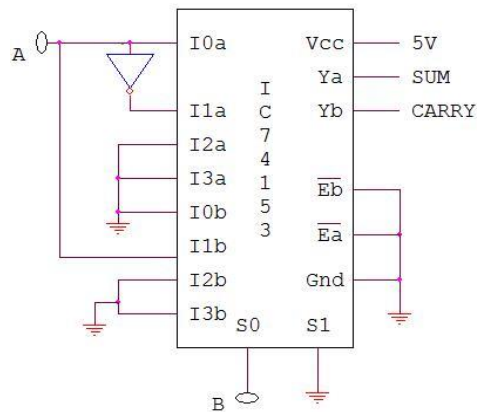


**HALF ADDER USING MUX:****DESIGN:**

SUM		CARRY	
I0	I1	I0	I1
0	1	0	1
2	3	2	3
A	A'	0	A

**TRUTH TABLE**

Inputs		Outputs	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

**CIRCUIT:****FULL ADDER USING MUX:****DESIGN:**

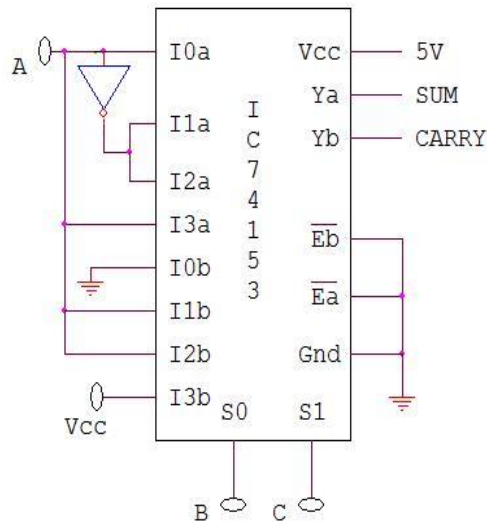
I0	I1	I3	I3
0	1	2	3
4	5	6	7
A	A'	A'	A

I0	I1	I3	I3
0	1	2	3
4	5	6	7
0	A	A	1

**TRUTH TABLE**

Inputs			Outputs	
A	B	C	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

FULL ADDER CIRCUIT



HALF SUBTRACTOR USING MUX:

DESIGN:

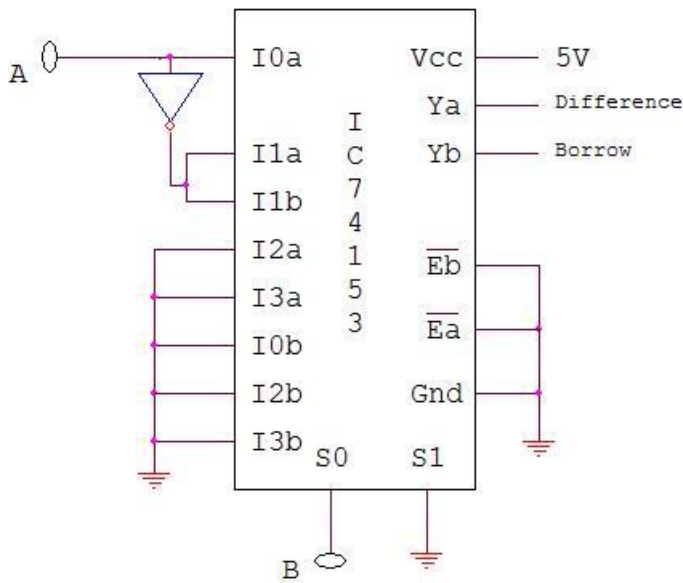
DIFFERENCE

I0	I1
0	1
2	3
A	A'

BORROW

I0	I1
0	1
2	3
0	A'

CIRCUIT:



TRUTH TABLE

Inputs		Outputs	
A	B	D	Br
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



**FULL SUBTRACTOR USING MUX:**

DESIGN:

DIFFERENCE

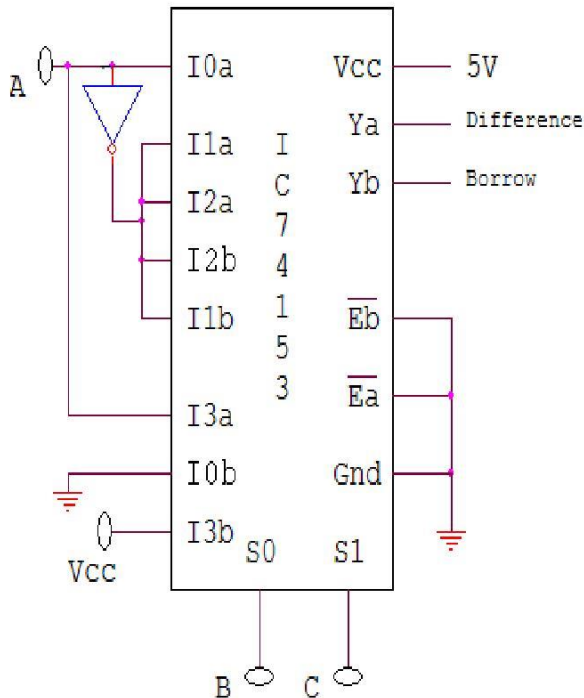
I0	I1	I2	I3
0	1	2	3
4	5	6	7
A	A'	A'	A

BORROW

I0	I1	I2	I3
0	1	2	3
4	5	6	7
0	A'	A'	1

**TRUTH TABLE**

Inputs			Outputs	
A	B	C	D	Br
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

**PROCEDURE:**

- ☐ Check all the components for their working.
- ☐ Insert the appropriate IC into the IC base.
- ☐ Make connections as shown in the circuit diagram.
- ☐ Verify the Truth Table and observe the outputs.

```

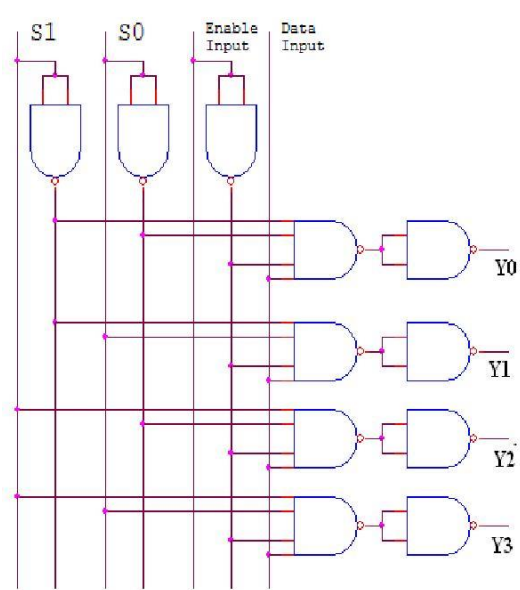
module mux4_1_synth_conditional(w, s, f );
    input [3:0] w;
    input [1:0] s;
    output reg f;
    always @ (*)
        f = s[1]?(s[0]?w[3]:w[2]):(s[0]?w[1]:w[0]);
endmodule

```

**RESULT:** Adder and subtractor circuits are realized using multiplexer IC 74153.

**STUDY DEMULTIPLEXER OF IC 74LS139:**

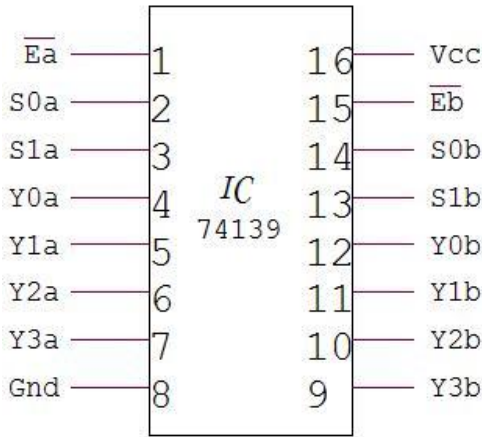
**iii) DE-MUX USING NAND GATES**



Enable Inputs		Data Input		Select Inputs		Outputs			
E	D	S <sub>1</sub>	S <sub>0</sub>	Y <sub>3</sub>	Y <sub>2</sub>	Y <sub>1</sub>	Y <sub>0</sub>		
1	0	X	X	X	X	X	X		
0	1	0	0	0	0	0	1		
0	1	0	1	0	0	1	0		
0	1	1	0	0	1	0	0		
0	1	1	1	1	0	0	0		

**VERIFICATION OF IC 74139 (DEMUX)**

**TRUTH TABLE**



Inputs			Outputs			
Ea	S <sub>1</sub>	S <sub>0</sub>	Y <sub>3</sub>	Y <sub>2</sub>	Y <sub>1</sub>	Y <sub>0</sub>
1	X	X	1	1	1	1
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1

```

module demux1_4(z,a,b,en);
  input a,b,en;
  output [3:0]z;
  reg z;
  always@(a or b or en)
  case({en,a,b})
  default: z = 4'b1111;
  3'b100 : z = 4'b1110;
  4'b110 : z = 4'b1101;
  3'b101 : z = 4'b1011;
  4'b111 : z = 4'b0111;
  endcase
endmodule

```

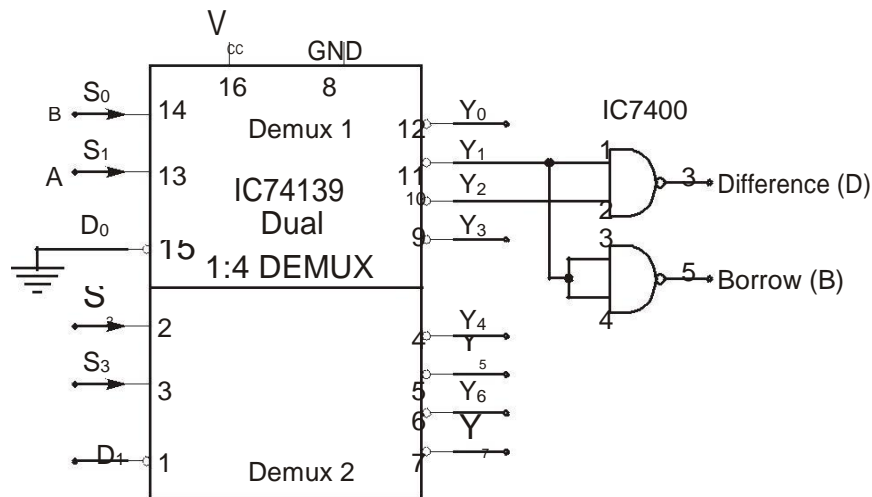
```

module decoder(l, en, Y);
  input [1:0] l;
  input en;
  output [3:0] Y;
  reg [3:0] Y;
  always @(en,l)
  begin
  if(en==1)
  Y=4'd0;
  else
  case(l)
  2'd0 : Y= 4'b0001;
  2'd1 : Y= 4'b0010;
  2'd2 : Y= 4'b0100;
  2'd3 : Y= 4'b1000;
  default : Y = 4'bXXXX;
  endcase
  end
endmodule

```

#### PROCEDURE:

- ☐ Check all the components for their working.
- ☐ Insert the appropriate IC into the IC base.
- ☐ Make connections as shown in the circuit diagram.
- ☐ Verify the Truth Table and observe the outputs.

**Realization of Half Subtractor and Full Subtractor circuits using IC 74139 Chip.****i) Half Subtractor****Logic Diagram:****Figure-2: Realization of Half Subtractor using IC 74139****Truth Table:**

Inputs		Outputs	
A	B	Difference (D)	Borrow (B)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

$$\text{Difference (D)} = \sum_m(1,2)$$

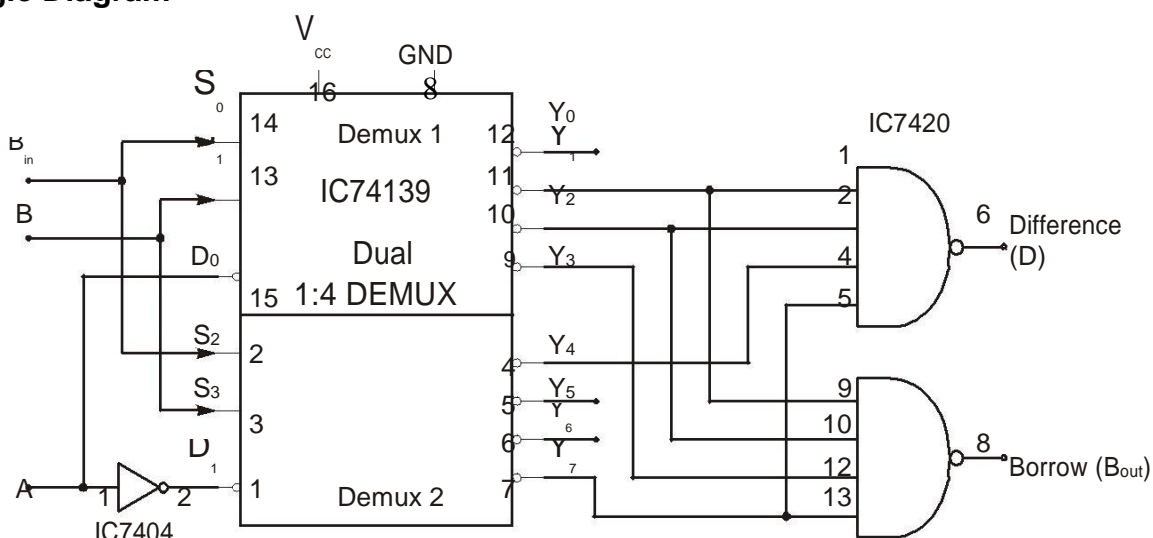
$$\text{Borrow (B}_{out}) = \sum_m(1)$$

$$\text{Difference (D)} = \bar{A}B + A\bar{B} = A \oplus B$$

$$\text{Borrow (B}_{out}) = \bar{A}B$$

**Procedure:**

- 1) Connect the circuit as shown in figure-2 using the pin details of the IC used.
- 2) Switch on the power supply and verify the truth table of half subtractor. circuit for various input combinat

**Logic Diagram****Figure-3: Realization of Full Subtractor using IC 74139****Truth Table:**

Inputs			Outputs	
A	B	B <sub>in</sub>	Difference (D)	Borrow (B <sub>out</sub> )
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

$$\text{Difference (D)} = \sum_{m(1,2,4,7)}$$

$$\text{Borrow (B}_{\text{out}}\text{)} = \sum_{m(1,2,3,7)}$$

$$\text{Difference (D)} = \bar{A}\bar{B}B_{in} + \bar{A}B\bar{B}_{in} + A\bar{B}\bar{B}_{in} + AB B_{in}$$

$$\text{Borrow (B}_{\text{out}}\text{)} = \bar{A}B + \bar{A}B_{in} + BB_{in}$$

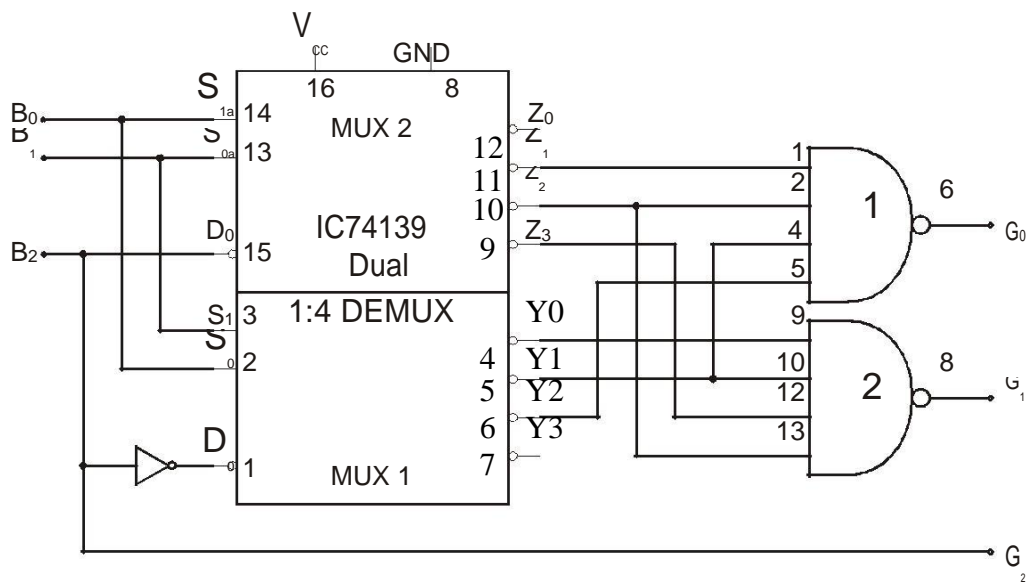
**Procedure:**

- 1) Connect the circuit as shown in figure-3 using the pin details of the IC used.
- 2) Switch on the power supply and verify the truth table of full subtractor circuit for various input combinations.

**Result / Conclusion:**

Truth-table of 1 : 4 Demux IC, Half subtractor and Full subtractor circuits are verified.

**Code Converters using 74139 IC (1 : 4 DEMUX)****Realization of 3-bit Binary to Gray code converter using IC 74139**



Truth Table of Binary to Gray

Input Binary Codes			Output Gray Codes		
B <sub>2</sub>	B <sub>1</sub>	B <sub>0</sub>	G <sub>2</sub>	G <sub>1</sub>	G <sub>0</sub>
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

Truth Table of Gray to Binary

Input Gray Codes			Output Binary Codes		
G <sub>2</sub>	G <sub>1</sub>	G <sub>0</sub>	B <sub>2</sub>	B <sub>1</sub>	B <sub>0</sub>
0	0	0	0	0	0
0	0	1	0	0	1
0	1	1	0	1	0
0	1	0	0	1	1
1	1	0	1	0	0
1	1	1	1	0	1
1	0	1	1	1	0
1	0	0	1	1	1

Expressions of Binary to Gray

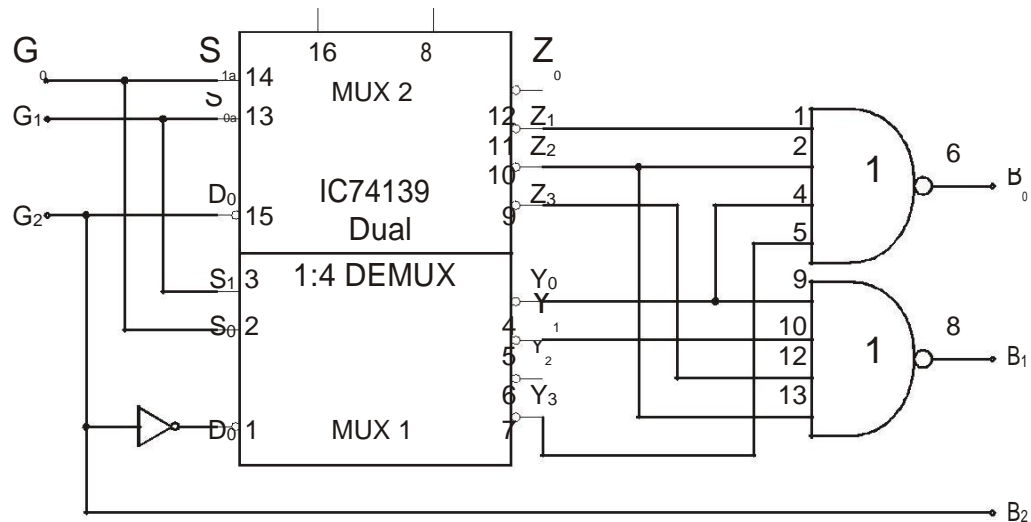
$$\begin{aligned} G_0 &= \sum_m (1,2,5,6) \\ G_1 &= \sum_m (2,3,4,5) \\ G_2 &= \sum_m (4,5,6,7) \end{aligned}$$

Expressions of Gray to Binary

$$\begin{aligned} B_0 &= \sum_m (1,2,4,7) \\ B_1 &= \sum_m (2,3,4,5) \\ B_2 &= \sum_m (4,5,6,7) = G_2 \end{aligned}$$

Realization of 3-bit Gray to Binary code converter using IC 74139





### Realization of 4-bit BCD to Excess-3 code converter using IC 74139

Expressions for Z, Y, X, W are given by

$$\begin{aligned}
 Z &= \sum_m (0, 2, 4, 6, 8) \\
 Y &= \sum_m (0, 3, 4, 7, 8) \\
 X &= \sum_m (1, 2, 3, 4, 9) \\
 W &= \sum_m (5, 6, 7, 8, 9)
 \end{aligned}$$

### VIVA QUESTIONS:

- 1) What is a multiplexer?
- 2) What is a de-multiplexer?
- 3) What are the applications of multiplexer and de-multiplexer?
- 4) Derive the Boolean expression for multiplexer and de-multiplexer.
- 5) How do you realize a given function using multiplexer
- 6) What is the difference between multiplexer & demultiplexer?
- 7) In  $2^n$  to 1 multiplexer how many selection lines are there?
- 8) How to get higher order multiplexers?
- 9) Implement an 8:1 mux using 4:1 muxes?





**EXPERIMENT: 6****COMPARATORS/MULTIPLIERS**

**AIM:** To realize One & Two Bit Comparator and study of 7485 magnitude comparator.

**LEARNING OBJECTIVE:**

To learn about various applications of comparator

To learn and understand the working of IC 7485 magnitude comparator

To learn to realize 8-bit comparator using 4-bit comparator

**THEORY:**

Magnitude Comparator is a logical circuit, which compares two signals A and B and generates three logical outputs, whether  $A > B$ ,  $A = B$ , or  $A < B$ . IC 7485 is a high speed 4-bit Magnitude comparator, which compares two 4-bit words. The  $A = B$  Input must be held high for proper compare operation.

**COMPONENTS REQUIRED:**

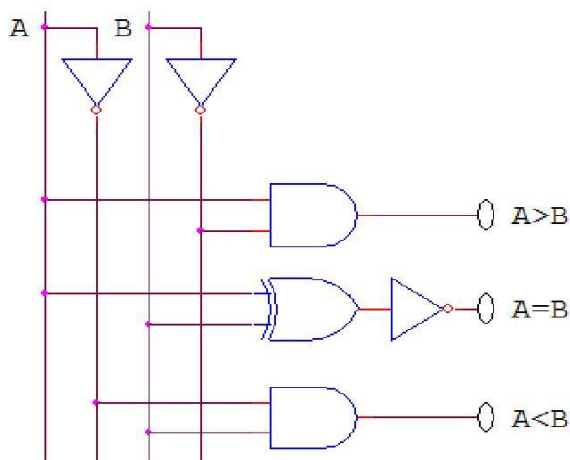
IC 7400, IC 7410, IC 7420, IC 7432, IC 7486, IC 7402, IC 7408, IC 7404, IC 7485, Patch Cords & IC Trainer Kit.

**1) 1- BIT COMPARATOR****TRUTH TABLE**

$$A > B = A \bar{B}$$

$$A < B = \bar{A} B$$

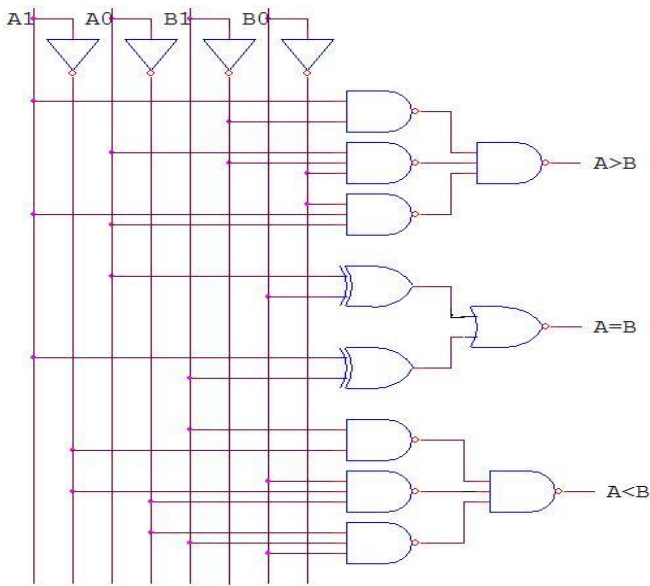
$$A = B = \bar{A} \bar{B} + AB$$



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

2- BIT COMPARATOR

$$(A>B)= A_1 B_1 + A_0 B_1 B_0 + B_0 A_1 A_0$$
$$(A=B) = (A_0 \oplus B_0) (A_1 \oplus B_1)$$
$$(A<B) = B_1 A_1 + B_0 A_1 A_0 + A_0 B_1 B_0$$



2-bit comparator circuit diagram

TRUTH TABLE

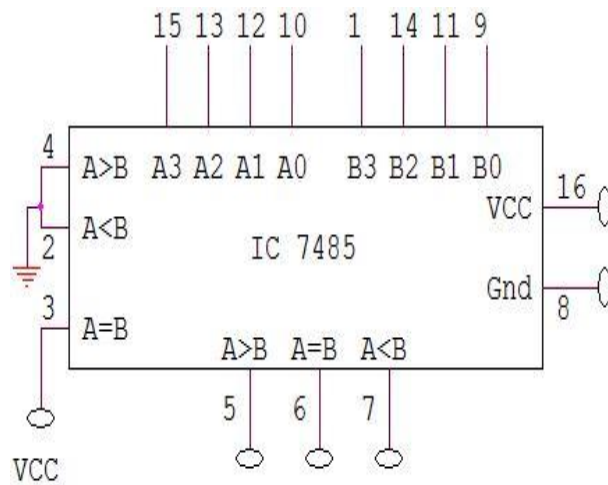
INPUTS				OUTPUTS		
A <sub>1</sub>	A <sub>0</sub>	B <sub>1</sub>	B <sub>0</sub>	A > B	A = B	A < B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0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

```

module comparator(a,b, c,d,e);
    input [1:0] a,b;
    output c,d,e;
    assign c=(a[1] & ~(b[1]))|(a[0] & ~(b[1]) & ~(b[0])) | (a[1] & a[0] & ~(b[0]));
    assign d=~(c | e);
    assign e=(b[0] & ~(a[1]) & ~(a[0]))|(b[1] & ~(a[1]))|(b[0] & b[1] & ~(a[0]));
endmodule

```

### 3. TO COMPARE THE GIVEN DATA USING 7485 CHIP.



A				B				Result
A3	A2	A1	A0	B3	B2	B1	B0	
0	0	0	1	0	0	0	0	A > B
0	0	0	1	0	0	0	1	A = B
0	0	0	0	0	0	0	1	A < B

#### PROCEDURE:

- ☐ Check all the components for their working.
- ☐ Insert the appropriate IC into the IC base.
- ☐ Make connections as shown in the circuit diagram.
- ☐ Verify the Truth Table and observe the outputs.

**RESULT:** One bit, two bit and four bit comparators are verified using basic gates and magnitude comparator IC7485

#### VIVA QUESTIONS:

- 1) What is a comparator?
- 2) What are the applications of comparator?
- 3) Derive the Boolean expressions of one bit comparator and two bit comparators.
- 4) How do you realize a higher magnitude comparator using lower bit comparator
- 5) Design a 2 bit comparator using a single Logic gates?
- 6) Design an 8 bit comparator using a two numbers of IC 7485?



## EXPERIMENT: 7

### BCD TO 7-SEGMENT DECODER/DRIVER

**AIM:** To set up and test a 7-segment static display system to display numbers 0 to 9.

**LEARNING OBJECTIVE:**

- To learn about various applications of decoder
- To learn and understand the working of IC 7447
- To learn about types of seven-segment display

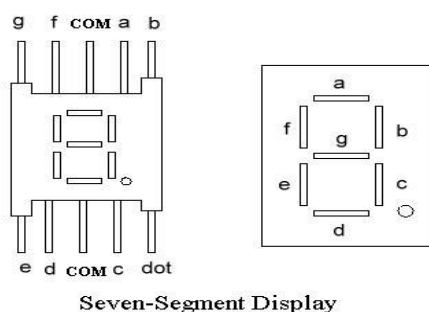
**COMPONENTS REQUIRED:**

IC7447, 7-Segment display (common anode), Patch chords, resistor (1K ) & IC Trainer Kit

**THEORY:**

The Light Emitting Diode (LED) finds its place in many applications in these modern electronic fields. One of them is the Seven Segment Display. Seven-segment displays contains the arrangement of the LEDs in “Eight” (8) passion, and a Dot (.) with a common electrode, lead (Anode or Cathode). The purpose of arranging it in that passion is that we can make any number out of that by switching ON and OFF the particular LED's. Here is the block diagram of the Seven Segment LED arrangement.

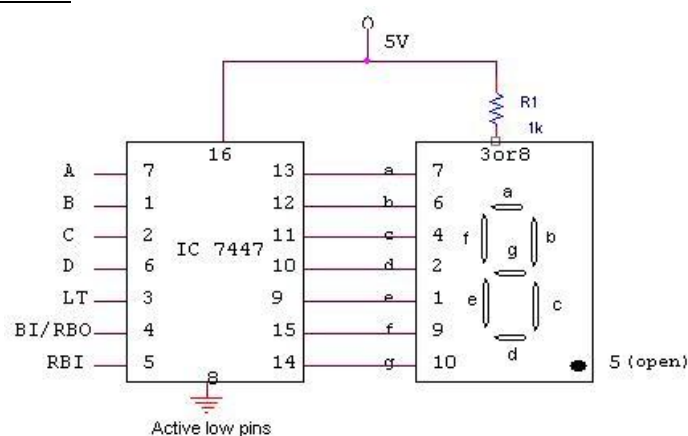
The Light Emitting Diode (LED), finds its place in many applications in this modern electronic fields. One of them is the Seven Segment Display. Seven-segment displays contains the arrangement of the LEDs in “Eight” (8) passion, and a Dot (.) with a common electrode, lead (Anode or Cathode). The purpose of arranging it in that passion is that we can make any number out of that by switching ON and OFF the particular LED's. Here is the block diagram of the Seven Segment LED arrangement.



LED's are basically of two types-

- Common Cathode (CC) -All the 8 anode legs uses only one cathode, which is common.
- Common Anode (CA)-The common leg for all the cathode is of Anode type.

A decoder is a combinational circuit that connects the binary information from 'n' input lines to a maximum of  $2^n$  unique output lines. The IC7447 is a BCD to 7-segment pattern converter. The IC7447 takes the Binary Coded Decimal (BCD) as the input and outputs the relevant 7 segment code.

CIRCUIT DIAGRAM:TRUTH TABLE:

BCD Inputs				Output Logic Levels from IC 7447 to 7-segments							Decimal number display
D	C	B	A	a	b	c	d	e	f	g	
0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	1	0	0	1	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0	2
0	0	1	1	0	0	0	0	1	1	0	3
0	1	0	0	1	0	0	1	1	0	0	4
0	1	0	1	0	1	0	0	1	0	0	5
0	1	1	0	1	1	0	0	0	0	0	6
0	1	1	1	0	0	0	1	1	1	1	7
1	0	0	0	0	0	0	0	0	0	0	8
1	0	0	1	0	0	0	1	1	0	0	9

PROCEDURE:

- ☐ Check all the components for their working.
- ☐ Insert the appropriate IC into the IC base.
- ☐ Make connections as shown in the circuit diagram.
- ☐ Verify the Truth Table and observe the outputs.

VIVA QUESTIONS:

1. What are the different types of LEDs?
2. Draw the internal circuit diagram of an LED.
3. What are the applications of LEDs?

## ENCODERS:

### AIM:

1. To set up a circuit of Decimal-to-BCD Encoder using IC 74147.
2. To design and set up a circuit of Hexadecimal-to-Binary Encoder using IC 74148 Encoders.

### LEARNING OBJECTIVE:

- To learn about various applications of Encoders
- To learn and understand the working of IC 74147, IC 74148.
- To learn to do code conversion using encoders

### COMPONENTS REQUIRED:

IC 74147, IC 74148, IC 74157, Patch chords & IC Trainer Kit

### THEORY:

An encoder performs a function that is the opposite of decoder. It receives one or more signals in an encoded format and output a code that can be processed by another logic circuit. One of the advantages of encoding data, or more often data addresses in computers, is that it reduces the number of required bits to represent data or addresses. For example, if a memory has 16 different locations, in order to access these 16 different locations, 16 lines (bits) are required if the addressing signals are in 1 out of  $n$  format. However, if we code the 16 different addresses

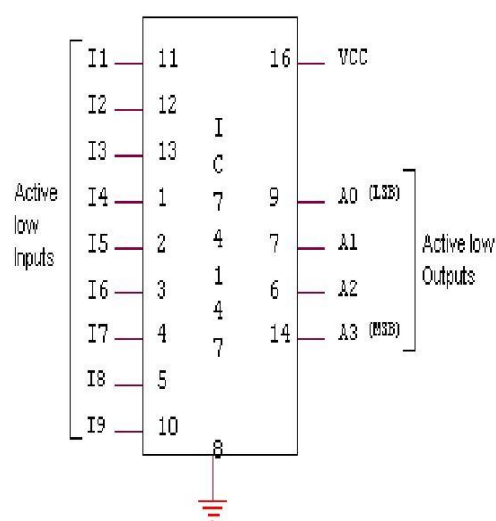
into a binary format, then only 4 lines (bits) are required. Such a reduction improves the speed of information processing in digital systems.

### CIRCUIT DIAGRAM:

#### **1) DECIMAL-TO-BCD ENCODER USING IC 74147.**

##### **TRUTH TABLE**

INPUTS									OUTPUTS			
I <sub>1</sub>	I <sub>2</sub>	I <sub>3</sub>	I <sub>4</sub>	I <sub>5</sub>	I <sub>6</sub>	I <sub>7</sub>	I <sub>8</sub>	I <sub>9</sub>	A <sub>3</sub>	A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>
1	1	1	1	1	1	1	1	0	0	1	1	0
X	X	X	X	X	X	X	0	1	0	1	1	1
X	X	X	X	X	X	0	1	1	1	0	0	0
X	X	X	X	X	0	1	1	1	1	0	0	1
X	X	X	X	0	1	1	1	1	1	0	1	0
X	X	X	0	1	1	1	1	1	1	0	1	1
X	X	0	1	1	1	1	1	1	1	1	0	0
X	0	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1



<pre> module encoder(en, i, y);   input en;   input [7:0] i;   output [2:0] y;   reg [2:0] y;   always @(en,i)   begin     if(en==1)       y=3'd0;     else       case(i)         8'b00000001 : y=3'd0;         8'b00000010 : y=3'd1;         8'b00000100 : y=3'd2;         8'b00001000 : y=3'd3;         8'b00010000 : y=3'd4;         8'b00100000 : y=3'd5;         8'b01000000 : y=3'd6;         8'b10000000 : y=3'd7;         default : y=3'bXXX;       endcase     end   endmodule </pre>	<pre> module priority(en, i, y);   input en;   input [7:0] i;   output [2:0] y;   reg [2:0] y;   always @(en,i)   begin     if(en==1)       y=3'd0;     else       casex(i)         8'b00000001 : y=3'd0;         8'b0000001X : y=3'd1;         8'b000001XX : y=3'd2;         8'b00001XXX : y=3'd3;         8'b0001XXXX : y=3'd4;         8'b001XXXXX : y=3'd5;         8'b01XXXXXX : y=3'd6;         8'b1XXXXXXX : y=3'd7;         default : y=3'b000;       endcase     end   endmodule </pre>
--	--



**EXPERIMENT: 8 & 9****(I) FLIP FLOPS & (II) STUDY OF 3 BIT COUNTERS AS A SEQUENTIAL CIRCUIT and MOD-N COUNTERS**AIM: Truth Table verification of

- a. JK Master Slave Flip Flop.
- b. T type Flip Flop.
- c. D type Flip Flop. .

LEARNING OBJECTIVE:

- To learn about various Flip-Flops
- To learn and understand the working of Master slave FF
- To learn about applications of FFs
- Conversion of one type of Flip flop to another

COMPONENTS REQUIRED:

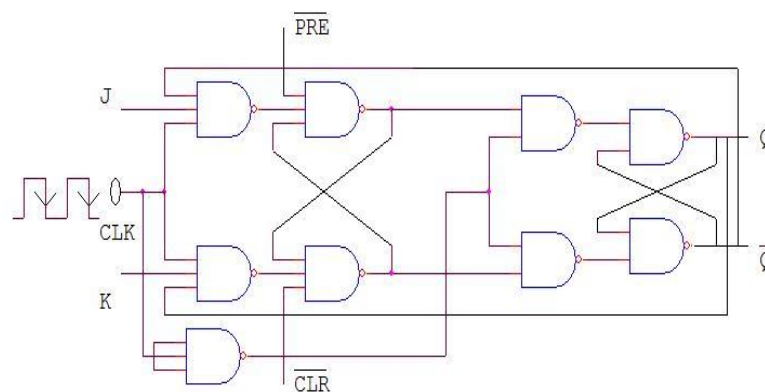
IC 7400, Patch Cords &amp; IC Trainer Kit.

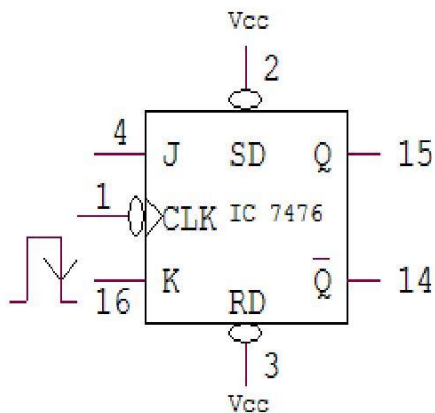
THEORY:

Logic circuits that incorporate memory cells are called *sequential logic circuits*; their output depends not only upon the present value of the input but also upon the previous values. Sequential logic circuits often require a timing generator (a clock) for their operation.

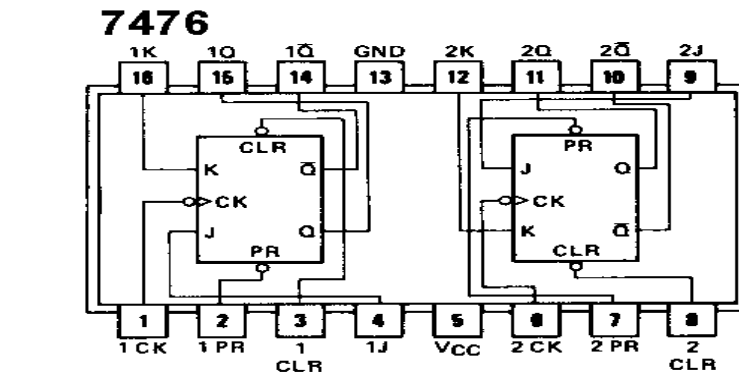
The latch (flip-flop) is a basic bi-stable memory element widely used in sequential logic circuits. Usually there are two outputs, Q and its complementary value.

Some of the most widely used latches are listed below.

**JK MASTER SLAVE FLIP FLOP****LOGIC DIAGRAM**

**TRUTH TABLE****LOGIC DIAGRAM IC7476**

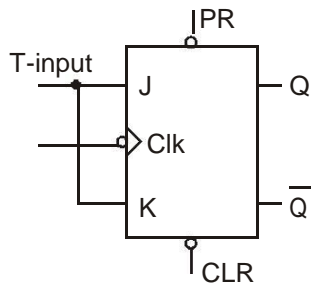
SD	RD	Clock	J	K	Q	Q'	Comment
0	0	Not Allowed					
0	1	X	X	X	1	0	Set
1	0	X	X	X	0	1	Reset
1	1	1	0	0	NC	NC	Memory
1	1	1	0	1	0	1	Reset
1	1	1	1	0	1	0	Set
1	1	1	1	1	Q'	Q	Toggle

**Pin Details of 7476 IC****PROCEDURE:**

- ☐ Check all the components for their working.
- ☐ Insert the appropriate IC into the IC base.
- ☐ Make connections as shown in the circuit diagram.
- ☐ Verify the Truth Table and observe the outputs.

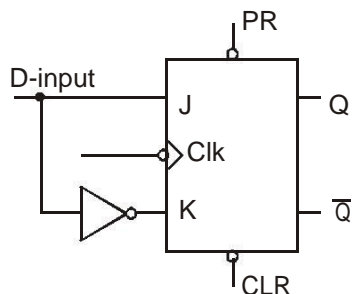
**VIVA QUESTIONS:**

1. What is the difference between Flip-Flop & latch?
2. Give examples for synchronous & asynchronous inputs?
3. What are the applications of different Flip-Flops?
4. What is the advantage of Edge triggering over level triggering?
5. What is the relation between propagation delay & clock frequency of flip-flop?
6. What is race around in flip-flop & how to over come it?
7. Convert the J K Flip-Flop into D flip-flop and T flip-flop?
8. List the functions of asynchronous inputs?

**Realization of T - Flip Flop using IC 7476.****Realization of T-F/F  
using J-K MS F/F****Truth Table**

Inputs		Outputs	Comments
$I_{in}$	CLK	$Q_{n+1}$	
0	X	$Q_n$	Reset
0		0	No Change
1		1	Toogle

**Note:** Keep PR and CLR = 1 for verifying the truth tables of JKMS Flip Flop.

**c) Realization of D - Flip Flop using J-K Master Slave Flip Flop.****Realization of D-F/F  
using J-K MS F/F****Truth Table**

Inputs		Outputs	Comments
D	CLK	$Q_{n+1}$	
0	X	$Q_n$	Reset
0		0	Data Transfer
1		1	Data Transfer

**Procedure:**

1. Make connections as per the circuit diagram for T and D flip-flop.
2. Feed the T and D input bit binary patterns using logic input switches.
3. Connect the clock (CLK) pin to the bounceless pulser high or low.
4. Monitor the output using logic output LED indicators.
5. Verify the output according to the truth table.

**JK FLIP-FLOP**

```

module jkff(j,k,clk, q,qb);
  input j,k,clk;
  output q,qb;
  reg q=1'b0;
  reg qb=1'b1;
  reg [20:0]clk1=21'd0;
  always@(posedge(clk))
  begin
    clk1=clk1+1;
  end
  always@(posedge(clk1[20]))
  begin
    case({j,k})
      2'b00:q=q;
      2'b01:q=0;
      2'b10:q=1;
      2'b11:q=~q;// for SRFF q=1'bX;
    endcase
    qb=~q;
  end
endmodule

```

**D FLIP-FLOP**

```

module dff(clk,d, q,qb);
  input clk,d;
  output q,qb;
  reg q,qb;
  always @(posedge(clk))
  begin
    q=d;
    qb=~q;
  end
endmodule

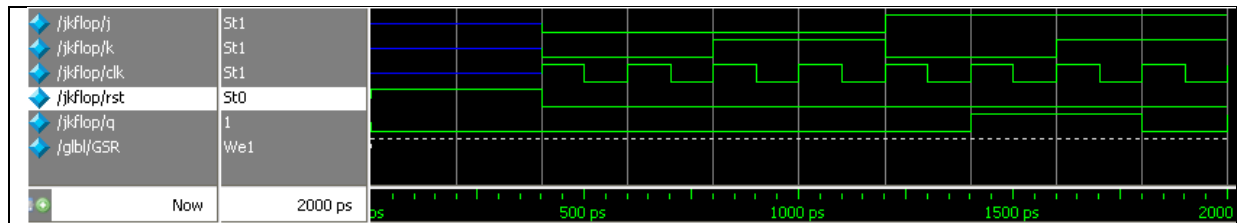
```

**T FLIP-FLOP**

```

module tff(t,clk, q,qb);
  input t,clk;
  output q,qb;
  reg q=1'b0;
  reg qb=1'b1;
  reg [20:0]clk1=21'd0;
  always@(posedge(clk))
  begin
    clk1=clk1+1;
  end
  always@(posedge(clk1[20]))
  begin
    if(t==1)
      q=~q;
      qb=~q;
    end
  end
endmodule

```



## (II) STUDY OF 3 BIT COUNTERS AS A SEQUENTIAL CIRCUIT and MOD-N COUNTERS

### ASYNCHRONOUS COUNTER

**AIM:** To design and test 3-bit binary asynchronous counter using flip-flop IC 7476 for the given sequence

#### LEARNING OBJECTIVE:

To learn about Asynchronous Counter and its application

To learn the design of asynchronous up counter and down counter

#### COMPONENTS REQUIRED:

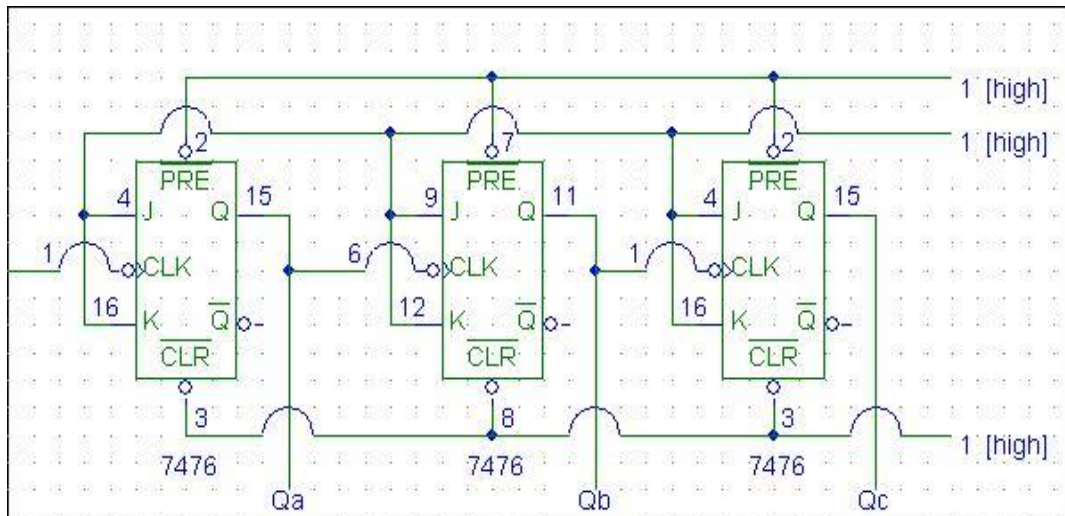
IC 7476, Patch Cords & IC Trainer Kit

#### THEORY:

A counter in which each flip-flop is triggered by the output goes to previous flip-flop. As all the flip-flops do not change state simultaneously spike occur at the output. To avoid this, strobe pulse is required. Because of the propagation delay the operating speed of asynchronous counter is low. Asynchronous counter are easy and simple to construct

### **MOD-8 UP COUNTER**

#### **CIRCUIT DIAGRAM:**



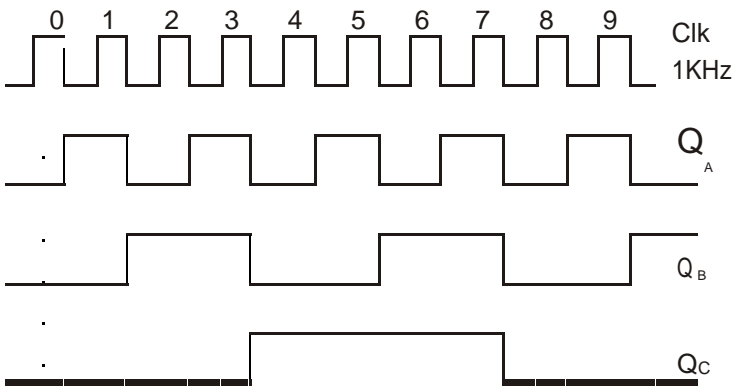
#### PROCEDURE:

- Check all the components for their working.
- Insert the appropriate IC into the IC base.
- Make connections as shown in the circuit diagram.
- Verify the Truth Table and observe the outputs

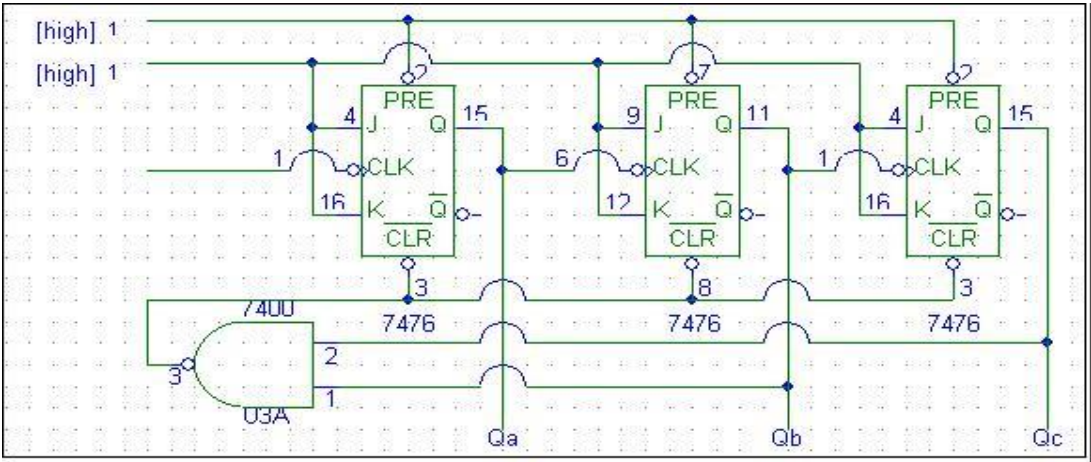
TRUTH TABLE

CLK	QC	QB	QA
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0

Waveform:



MOD 6 UP COUNTER  
CIRCUIT DIAGRAM:



TRUTH TABLE

CLK	QC	QB	QA
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	0	0	0



VIVA QUESTIONS:

What is an asynchronous counter?

How is it different from a synchronous counter?

Realize asynchronous counter using T flip-flop

**3bit counter:**

```
module counter3bit(q,clk,rst);  
output reg[2:0] q;  
input clk,rst;  
reg [20:0]clk1=21'd0;  
always@(posedge(clk))  
begin  
clk1=clk1+1;  
end  
always@(posedge(clk1[20]))  
begin  
if(rst)  
q=0;  
else  
q=q+1; //q=q-1; for down counter.  
end  
endmodule
```



**SYNCHRONOUS COUNTERS**

**AIM:** To design and test 3-bit binary synchronous counter using flip-flop IC 7476 for the given sequence.

**LEARNING OBJECTIVE:**

To learn about synchronous Counter and its application To learn the design of synchronous counter counter

**COMPONENTS REQUIRED:**

IC 7476, Patch Cords & IC Trainer Kit

**THEORY:**

A counter in which each flip-flop is triggered by the output goes to previous flip-flop. As all the flip-flops do not change states simultaneously in asynchronous counter, spike occur at the output. To avoid this, strobe pulse is required. Because of the propagation delay the operating speed of asynchronous counter is low. This problem can be solved by triggering all the flip-flops in synchronous with the clock signal and such counters are called synchronous counters.

**PROCEDURE:**

- ☐ Check all the components for their working.
- ☐ Insert the appropriate IC into the IC base.
- ☐ Make connections as shown in the circuit diagram.
- ☐ Verify the Truth Table and observe the outputs.

**MOD 5 COUNTER:****TRUTH TABLE:**

QC	QB	QA
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
0	0	0

**Present count**

QC	QB	QA
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	0

**next count**

QC	QB	QA
0	0	1
0	1	0
0	1	1
1	0	0
0	0	0

**JK FF excitation table:**

Q	Q <sup>+</sup>	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

**DESIGN:**

1	X	X	1
0	X	X	X

$$J_A = \overline{Q_C}$$

X	1	X	X
X	X	X	X

$$K_A = 1$$

0	1	X	X
0	X	X	X

$$J_B = Q_A$$

X	X	1	0
X	X	X	X

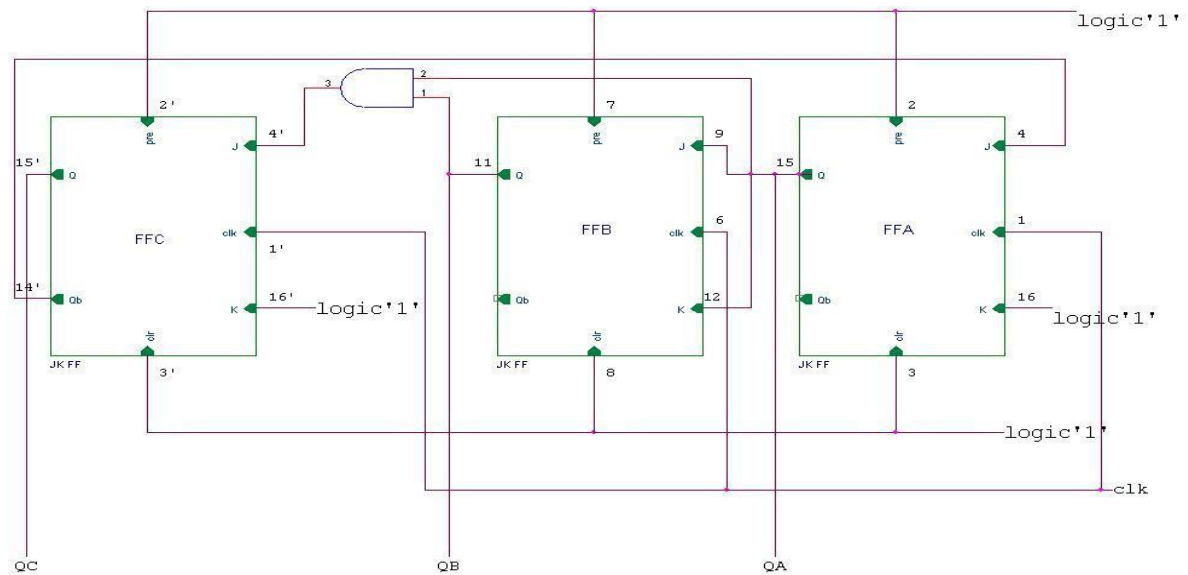
$$K_B = Q_A$$

0	0	1	0
X	X	X	X

$$J_C = Q_B Q_A$$

X	X	X	X
1	X	X	X

$$K_C = 1$$



### VIVA QUESTIONS:

What are synchronous counters?

What are the advantages of synchronous counters?

What is an excitation table?

Write the excitation table for D, T FF

Design mod-5 synchronous counter using T FF



EXPERIMENT: 10

STUDY OF 4 BIT COUNTERS AS A SEQUENTIAL CIRCUIT and MOD-N COUNTERS

STUDY ASYNCHRONOUS OF 7490 BCD COUNTER

AIM: To design IC 7490 as a decade counter with BCD count sequence

LEARNING OBJECTIVE:

- To learn about decade Counter
- To use it as a divide by N counter [ $N \leq 10$ , say  $N=7, N=5$ ]

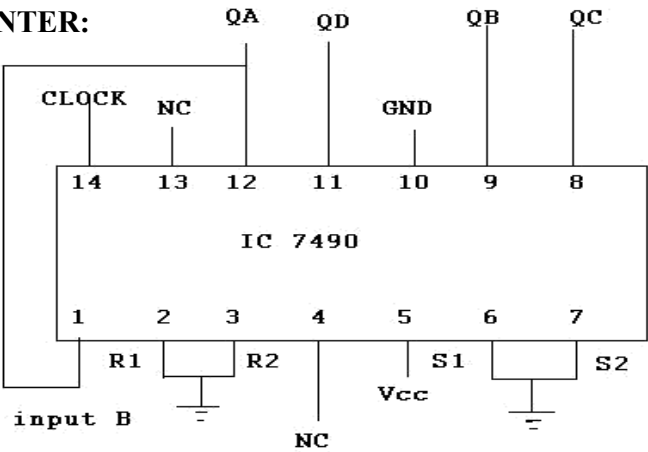
COMPONENTS REQUIRED:

IC 7490, Patch Cords & IC Trainer Kit

PROCEDURE:

- Check all the components for their working.
- Insert the appropriate IC into the IC base.
- Make connections as shown in the circuit diagram.
- Verify the Truth Table and observe the outputs.

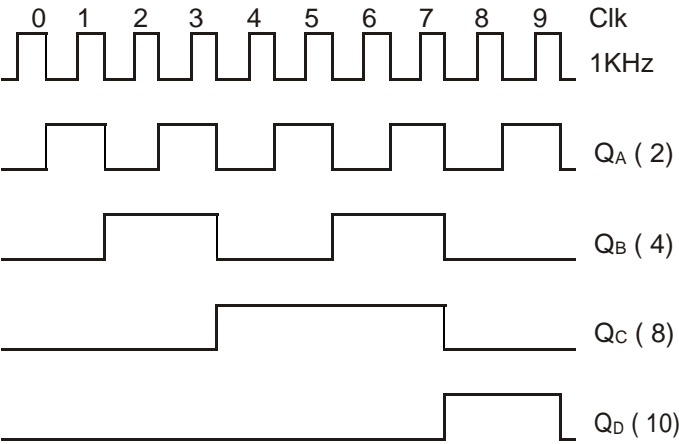
DECADE COUNTER:

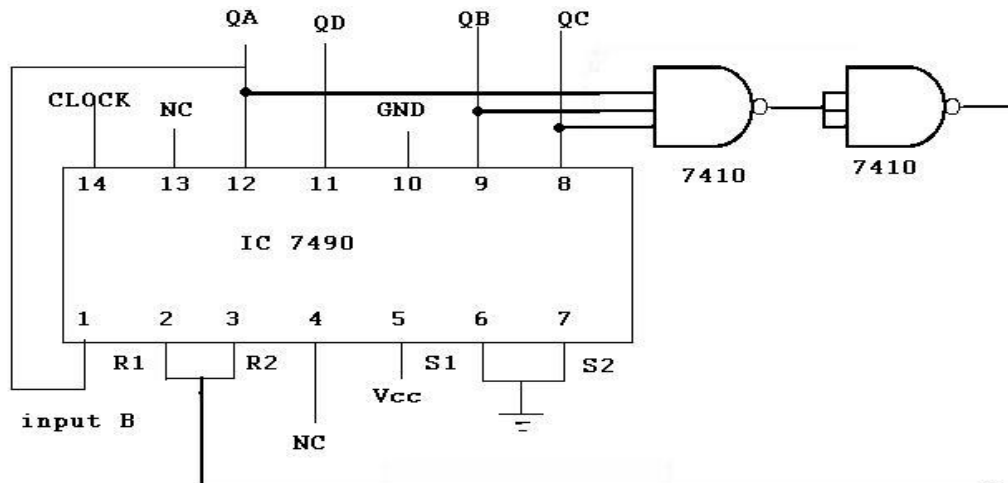


TRUTH TABLE

Waveform:

QD	QC	QB	QA
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
0	0	0	0



**7490 AS A DIVIDE BY N COUNTER (N=7):**

TRUTH TABLE:

Q <sub>D</sub>	Q <sub>C</sub>	Q <sub>B</sub>	Q <sub>A</sub>
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	0	0	0

VIVA QUESTIONS:

What is a decade counter?

What do you mean by a ripple counter?

Explain the design of Modulo-N counter ( $N \leq 9$ ) using IC 7490**PRESETTABLE 4-BIT SYNCHRONOUS BINARY UP/DOWN**

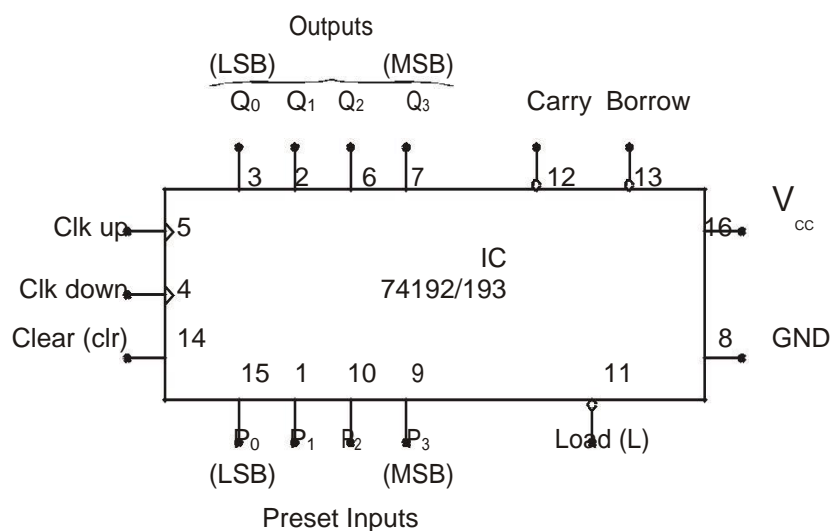
## COUNTER

### Study of Synchronous Binary Counter using IC74192 & IC 74193.

#### **Apparatus Required:**

IC's: 74192 or 74193, 7404 and 7420 chips.

#### **Logic Diagram:**

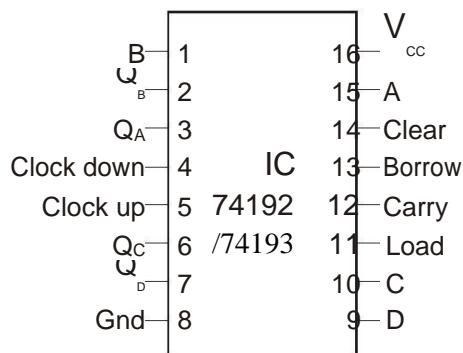


*Figure-1: Realization of 4-bit up/down decade counter*

#### **Function Table of 74192/74193**

Load	Clear	Clk-Up	Clk-down	Mode
X	1	X	X	Rest to zero
1	0		1	Up-count
1	0	1		Down-count
0	0	X	X	Preset
1	0	1	1	Stop count

#### **Pin Details of 74192 IC**



Clear = 0, Load = 1 and Clock Down = 1 for up counter operation.

Clear = 0, Load = 1 and Clock Up = 1 for down counter operation.

1. P1,P2,P3 and P0 are parallel data inputs
2. Q0,Q1,Q2 and Q3 are flip-flop outputs
3. MR: Asynchronous master reset
4. PL: Asynchronous parallel load(active low) input
5. TCd : Terminal count down output
6. TCu : Terminal count up output

#### **Note:**

- The carry and borrow outputs are normally at logic 1.
- If the outputs  $Q_D, Q_C, Q_B, Q_A$  changes from 1001 to 0000, then carry is logic 0, else carry is equal to 1.
- Carry and Borrow are mainly used for cascading the counters.
- Output waveforms is similar to that of synchronous 4-bit up-down counter using 7476IC, except that the flip flop output changes states when the clock makes a low to high transition.

**a) Realization of 4-bit Up-Counter :**

**Truth Table (Count Sequence Table):**

Number of	Outputs			
input pulses	$Q_3$	$Q_2$	$Q_1$	$Q_0$
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
00	0	0	0	0

Number of	Outputs			
input pulses	$Q_3$	$Q_2$	$Q_1$	$Q_0$
0	1	1	1	1
1	1	1	1	0
2	1	1	0	1
3	1	1	0	0
4	1	0	1	1
5	1	0	1	0
6	1	0	0	1
7	1	0	0	0
8	0	1	1	1
9	0	1	1	0
10	0	1	0	1
11	0	1	0	0
12	0	0	1	1
13	0	0	1	0
14	0	0	0	1
15	0	0	0	0
0	1	1	1	1

**Procedure:**

- 1) Connect the circuit as shown in figure-1 using the pin details of the IC used.
- 2) Switch on the power supply. For up-counting, initially clear the outputs by connecting the clear input to logic '1' level then keep Clear =0, Load = 1, Clk-down = 1 levels and apply 1Hz clock or mono pulse to clock-up input.
- 3) Observe the count sequence at the outputs  $Q_3, Q_2, Q_1$  and  $Q_0$ . The count sequence is given by: 0000 to 1111 (i.e., 0, 1, 2, ....., 14,15, 0, 1, 2, .....

**Note:** The carry and borrow outputs are normally at logic '1' level. If the outputs  $Q_3, Q_2, Q_1$  and  $Q_0$  changes from 1001 to 0000, then carry = 0, else carry = 1.

1. Connect the circuit as shown in figure-1 using the pin details of the IC used.
2. Switch on the power supply. For down-counting, initially clear the outputs by connecting the clear input to logic '1' level then keep Clear =0, Load = 1, Clk-Up = 1 levels and apply 1Hz clock or mono pulse to clock-down input.
3. Observe the count sequence at the outputs  $Q_3, Q_2, Q_1$  and  $Q_0$ . The count sequence is given by: 1001 to 0000 (i.e., 15,14,13,, ....., 2, 1, 0, 15,14,13, .....

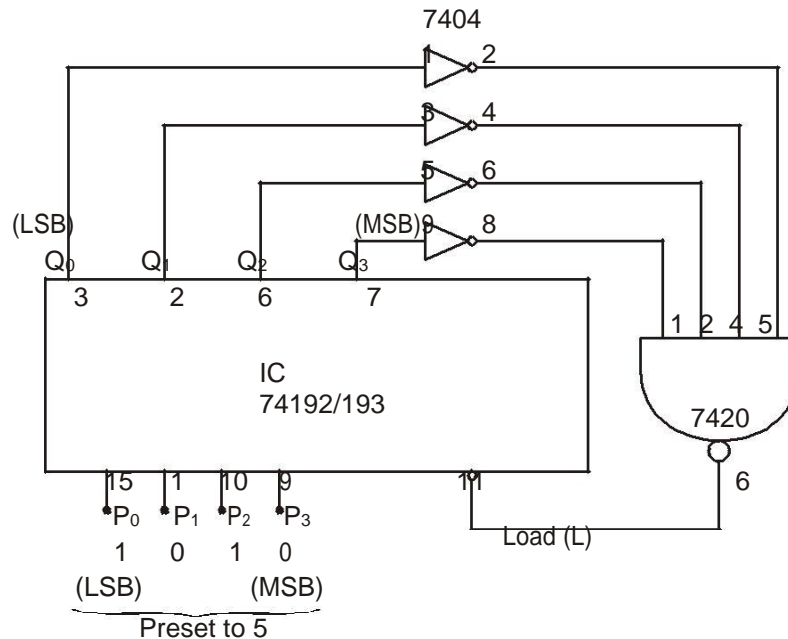
**Note:** If the outputs  $Q_3$ ,  $Q_2$ ,  $Q_1$  and  $Q_0$  changes from 0000 to 1001, then borrow = 0, else borrow = 1.

c) Realization of Mod-N Presettable Counter:

i) Mod-N Presettable Up-Counter:

Example: To count from 5 to 15.

Logic Diagram:



**Figure-2**

**Truth Table (Count Sequence Table):**

Number of	Outputs			
input pulses	$Q_3$	$Q_2$	$Q_1$	$Q_0$
0	0	1	0	1
1	0	1	1	0
2	0	1	1	1
3	1	0	0	0
4	1	0	0	1
5	1	0	1	0
6	1	0	1	1
7	1	1	0	0
8	1	1	0	1
9	1	1	1	0
10	1	1	1	1
11	0	1	0	1

**Procedure:**

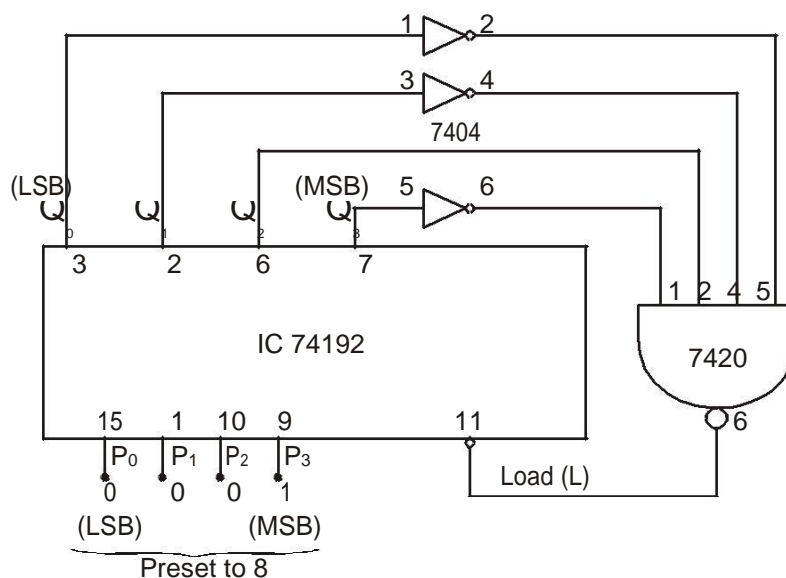
- 1) Connect the circuit as shown in figure-2 using the pin details of the IC used.
- 2) Switch on the power supply. Preset the data inputs to value 5 (i.e., connect the preset input pins  $P_2$  and  $P_0$  to logic '1' levels and  $P_3$  and  $P_1$  to logic '0' levels, i.e.,  $P_3P_2P_1P_0 = 0101$ ).
- 3) Set up for count-up mode i.e., keep clear = 0, clk down = 1 and apply 1Hz clock or mono



- pulse to clk-up input.
- 4) Observe the count sequence at the outputs  $Q_3$ ,  $Q_2$ ,  $Q_1$  and  $Q_0$  for the external logic shown. The count sequence is given by 0101, 0110, 0111, 1000, 1001, 0101, 0110, ..... (i.e., 5, 6, 7, 8, 9, 5, 6, 7, 8, 9, 5, 6, .....).
  - 5) **Note:** When the count sequence goes from 1001 (9) to 0000 (0) the data value 0101 (5) will be loaded (preset) into counter, as the load becomes zero. Hence the counter counts from 0101 (5) instead of 0000 (0).

**Example:** To count from 8 to 5.

### Logic Diagram:



**Figure-3**

### Truth Table (Count Sequence Table):

Number of input pulses	Outputs			
	$Q_3$	$Q_2$	$Q_1$	$Q_0$
0	1	0	0	0
1	0	1	1	1
2	0	1	1	0
3	0	1	0	1
4	1	0	0	0

### Procedure:

- 1) Connect the circuit as shown in figure-3 using the pin details of the IC used.
- 2) Switch on the power supply. Preset the data input 8 (connect the preset input pins  $P_3$  to logic '1' level and  $P_2$ ,  $P_1$ , and  $P_0$  to logic '0' levels, i.e.,  $P_3P_2P_1P_0 = 1000$ ).
- 3) Set up for countdown mode i.e., keep clear = 0, clk up = 1 and apply 1Hz clock or mono pulse to clk-down input.

- 4) Observe the count sequence at the outputs  $Q_3$ ,  $Q_2$ ,  $Q_1$  and  $Q_0$  for the external logic shown. The count sequence is given by 1000, 0111, 0110, 0101, 1000, 0111, ..... (i.e., 8, 7, 6, 5, 8, 7, 6, 5, 8, 7, .....).

**Result / Conclusion:**

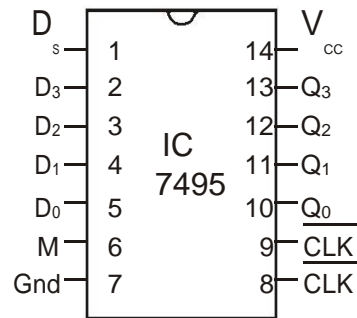
Mod-N counter operations are verified.

<pre> 4-bit Up_down counter: module aUpdown_counter(cnt, up_down,clk,rst);     output [3:0] cnt;     input up_down,clk,rst;     reg [3:0] cnt;     reg [15:0] clk1=16'd0;     always @(posedge(clk))     begin     clk1=clk1+1;     end     always @(posedge rst or posedge clk1[15])     begin     if(rst)     cnt=4'b0000;     else if(clk1[15])     if(up_down==0)     cnt=cnt-1;     else     cnt=cnt+1;     end     endmodule Synchronous mod-5 any sequence counter: Counting sequence – 2,5,7,8,9,2,5,..... module synany(clk,rst, cnt);     input clk,rst;     output [3:0] cnt;     reg [3:0] cnt;     reg [15:0] clk1=16'd0;     always@(posedge(clk))     begin     clk1=clk1+1;     end     always@(posedge(clk1[15]))     if(rst==1)     cnt=4'b0010;     else     case(cnt)     4'b0010: cnt=4'b0101;     4'b0101: cnt=4'b0111;     4'b0111: cnt=4'b1000;     4'b1000: cnt=4'b1001;     4'b1001: cnt=4'b0010;     default : cnt=4'b0000;     endcase     endmodule </pre>	<pre> Asynchronous any sequence counter: module asynany(clk,rst, cnt);     input clk,rst;     output [3:0] cnt;     reg [3:0] cnt;      reg [15:0] clk1=16'd0;     always @(posedge(clk))     begin     clk1=clk1+1;     end     always @(posedge rst or posedge clk1[15])     begin     if(rst)     cnt=4'b0010;     else     case(cnt)     4'b0010: cnt=4'b0101;     4'b0101: cnt=4'b0111;     4'b0111: cnt=4'b1000;     4'b1000: cnt=4'b1001;     4'b1001: cnt=4'b0010;     default : cnt=4'b0000;     endcase     end     endmodule  module bcd_counter(cnt, rst,clk);     output [3:0] cnt;     input rst,clk;     reg [3:0] cnt;     reg [15:0] clk1=16'd0;     always @(posedge(clk))     begin     clk1=clk1+1;     end     always @(posedge(clk1[15]))     begin     if(rst==1)     cnt=4'b0000;     else     cnt=cnt+1;     if(cnt==4'b1010)     cnt=4'b0000;     endendmodule </pre>
---	---



**EXPERIMENT: 11****SHIFT REGISTERS**

**Aim: To study Shift Left, Shift Right, SIPO, SISO, PISO, PIPO operation using IC 7495.**

**Pin Details of 7495 IC Shift Register**

$D_s$  : Serial input data (to be right shifted)

$D_3, D_2, D_1, D_0$  : Parallel data inputs.

$M$  : Mode Control

Keep  $M$  1 for loading parallel data and enable clock 2.  $M$  0 for loading parallel data and enable clock 1.

Clock 1: For loading parallel input data and for the operation of shift left of data, Clock 2: For right shift of data.

$Q_3, Q_2, Q_1, Q_0$  : Parallel outputs of the shift register.

**Shift Right Operation: Serial Input Parallel Output (SIPO) Operation**

- Mode control is set to logic '0'.
- Clock 1 (Pin no.9) of IC 7495 is connected to bounceless pulser high or low.
- The serial input to be converted to parallel output is fed by serial input data pin  $D_s$  of IC 7495.
- After four clock pulses the serial input data appears in parallel form at  $Q_3, Q_2, Q_1, Q_0$ .

**Serial Input Serial Output (SISO) Operation**

- Mode control is set to logic '0'.
- Clock 1 (Pin no.9) of IC 7495 is connected to bounceless pulser high or low.
- The serial input to be converted to parallel output is fed by serial input data pin  $D_s$  of IC 7495.
- After four clock pulses the serial input data appears in parallel form at  $Q_3, Q_2, Q_1, Q_0$ .
- The next three pulses will move the data out of the shift register serially at  $Q_0$  terminal.

**Parallel in Parallel Output (PIPO) Operation**

- Mode control is set to logic '1'.
- The parallel inputs to be loaded into shift register are given to D<sub>3</sub>, D<sub>2</sub>, D<sub>1</sub>, D<sub>0</sub> inputs of IC 7495.
- Clock 2 (Pin no.8) of IC 7495 is connected to bounceless pulser high or low and is pulsed once.
- Now D<sub>3</sub>, D<sub>2</sub>, D<sub>1</sub>, D<sub>0</sub> parallel inputs appears on Q<sub>3</sub>, Q<sub>2</sub>, Q<sub>1</sub>, Q<sub>0</sub> lines.

**Parallel in Serial Output (PISO) Operation**

- Mode control is set to logic '1'.
- The parallel inputs to be loaded into shift register are given to D<sub>3</sub>, D<sub>2</sub>, D<sub>1</sub>, D<sub>0</sub> inputs of IC 7495.
- Clock 1 (Pin no.9) of IC 7495 is connected to bounceless pulser high or low.
- With each clock pulse data shifts right by one place and hence after three clock pulses parallel input data is converted into serial output data at Q<sub>0</sub>.

**Truth-Table for SIPO and SISO**

Mode Control	Clock	Pulse	Serial Input Data	Outputs			
				Q <sub>3</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>
0	Clock-1	1	1	1	X	X	X
		2	0	0	1	X	X
		3	1	1	0	1	X
		4	0	0	1	0	1
		5					0
		6					1
		7					0

Parallel Output

Serial Output

**Truth-Table for PIPO and PISO**

Mode Control	Clock	Parallel Inputs				Pulse	Outputs			
		D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>		Q <sub>3</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>
1	Clk-2	1	0	1	0	1	1	0	1	0
0	Clk-1					2				1
						3				0
						4				1

Parallel Output

Serial Output

## Shift Left Operation

- Short Q<sub>0</sub> and D<sub>1</sub>, Q<sub>1</sub> and D<sub>2</sub> and Q<sub>2</sub> and D<sub>3</sub> pins correspondingly
- Mode control is set to logic '1'.
- The serial input data is fed to D<sub>0</sub> input pin.
- Clock 2 of IC 7495 is connected to bounceless pulser and is pulsed. Each time it is pulsed the shift register contents are left shifted once at it appears at Q<sub>3</sub> pin of IC 7495.

### Shift-Left Shift-Right register

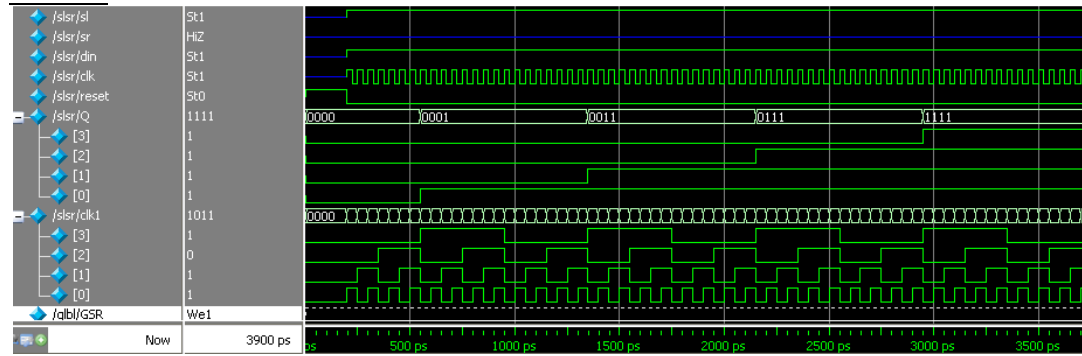
```

module slsr(sl, sr, din, clk, reset, Q);
input sl, sr, din, clk, reset;
output [3:0] Q;
reg [3:0] Q;
reg [3:0] clk1=4'd0;
always@(posedge clk)
    clk1=clk1+1;
always @ (posedge clk1[3] or posedge reset)
begin
    if (reset)
        Q<= 4'b0000;
    else if (sl)
        Q <= {Q[2:0],din};
    else if (sr)
        Q <= {din, Q[3:1]};
    end
endmodule

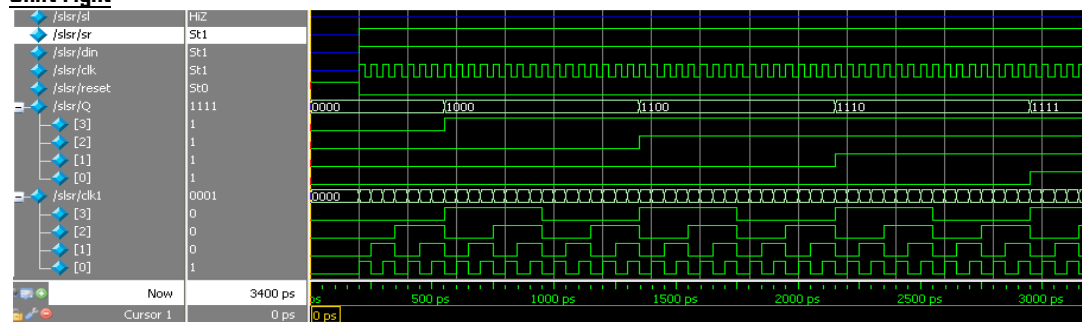
```

### Shift-Left Shift-Right register (simulation result)

#### 1.Shift left



#### Shift right



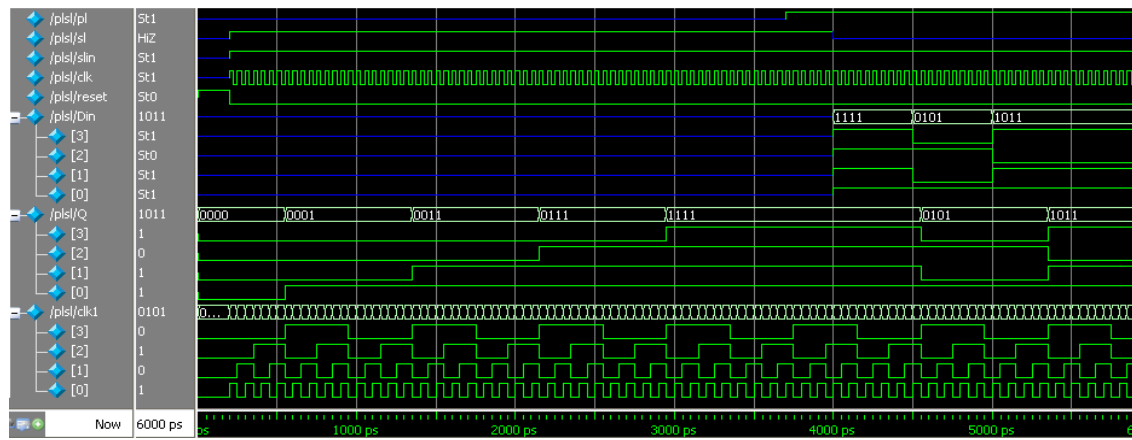
## 1. Parallel Load -Shift Left Register

```

module plsl(pl, sl, slin, Din, clk, reset, Q);
input pl, sl, slin, clk, reset;
input [3:0] Din;
output [3:0] Q;
reg [3:0] Q;
reg [3:0] clk1=4'd0;
always@(posedge clk)
clk1=clk1+1;
always @ (posedge clk1[3] or posedge reset)
begin
if (reset)
begin
Q <= 4'b0000;
else if (sl)
Q <= {Q[2:0],slin};
else if (pl)
Q <= Din;
end
end
endmodule

```

### Parallel Load -Shift Left Register(simulation output)







**EXPERIMENT: 12****RING COUNTER AND JOHNSON COUNTER**

AIM: To realize and study Ring Counter and Johnson counter.

LEARNING OBJECTIVE:

To learn about Ring Counter and its application

To learn about Johnson Counter and its application

COMPONENTS REQUIRED:

IC 7495, IC 7404, Patch Cords & IC Trainer Kit.

THEORY:

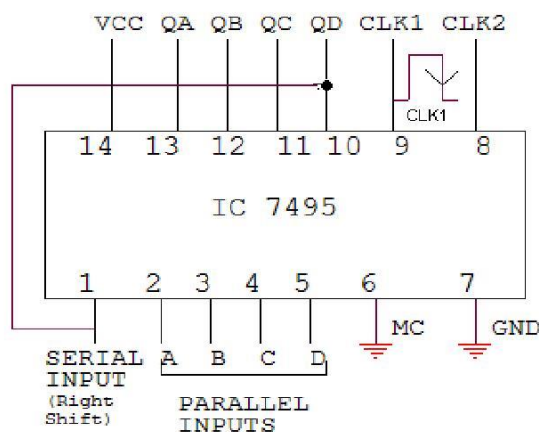
Ring counter is a basic register with direct feedback such that the contents of the register simply circulate around the register when the clock is running. Here the last output that is Q<sub>D</sub> in a shift register is connected back to the serial input.

A basic ring counter can be slightly modified to produce another type of shift register counter called Johnson counter. Here complement of last output is connected back to the not gate input and not gate output is connected back to serial input. A four bit Johnson counter gives 8 state output.

CIRCUIT DIAGRAM:

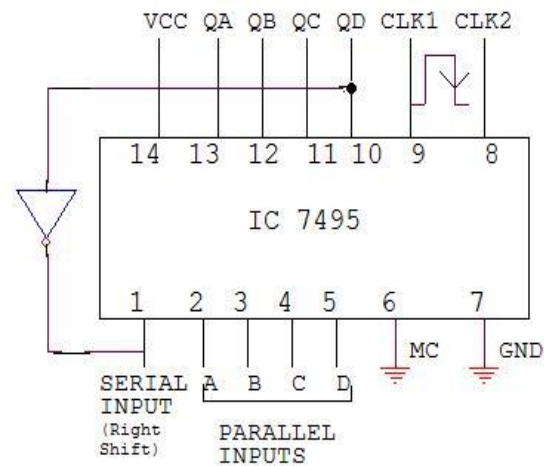
## 1) RING COUNTER

## TRUTH TABLE



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

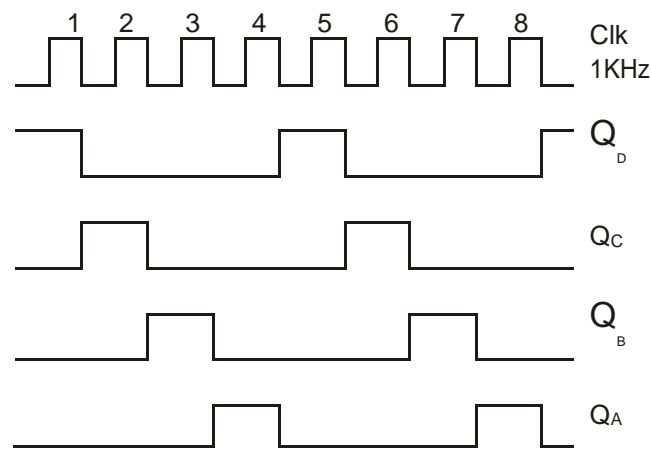
2) JOHNSON COUNTER



TRUTH TABLE

Clock pulses	Q <sub>A</sub>	Q <sub>B</sub>	Q <sub>C</sub>	Q <sub>D</sub>
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1
8	0	0	0	0

Waveform: Ring Counter



PROCEDURE:

- ☐ Check all the components for their working.
- ☐ Insert the appropriate IC into the IC base.
- ☐ Make connections as shown in the circuit diagram.
- ☐ Apply clock to pin number 9 and observe the output

RESULT: The truth table & working of Ring and Johnson counters is verified

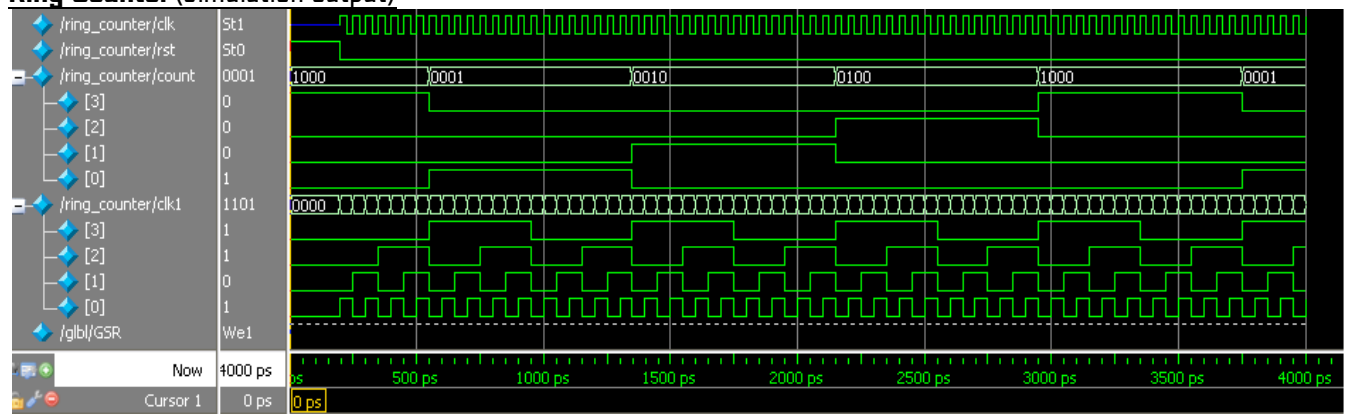
### Ring Counter

```

module ring_counter (clk, rst, count); (synthesis// simulation)
input clk, rst; output [3:0] count;
reg [3:0] count;
reg [15:0] clk1=16'd0; // reg [3:0] clk1=4'd0;
always@(negedge clk)
clk1=clk1+1;
always @ (posedge clk1[15]) // always @(posedge clk1[3])
begin
if (rst)
count <= 4'b1000;
else
begin
count <= count << 1;
count[0] <= count[3];
end
end
endmodule

```

### Ring Counter(simulation output)





## EXPERIMENT: 13 & 14

**Writing Verilog module to interface Stepper motor, DC-motor and DAC to FPGA.**

### Stepper motor

```
module stepper(clk,rst,dir,dout);
input clk,rst,dir;
output reg[3:0]dout;
reg[1:0]count;
reg[15:0]clk1;
always@(posedge clk)
begin
if(rst == 1)
clk1=1'b0;
else
clk1 = clk1+1;
end
always@(posedge clk1[15] or posedge rst)
begin
if(rst==1) count<=2'b00;
else
if(dir==1)count<=count+1;
else count<=count-1;
end
always@(count)
begin
case(count)
2'b00:dout<=4'b1100;
2'b01:dout<=4'b0110;
2'b10:dout<=4'b0011;
default:dout<=4'b1001;
endcase
end
endmodule
```

**DAC**

```
module dac(clk,rst,s1,s2,dacout);
input clk,rst,s1,s2;
output reg[7:0]dacout;
reg[7:0]count;
always@(posedge clk or posedge rst)
begin
if(rst==1)count<=8'd0;
else count<=count+1;
end
always@(s1 or s2 or count)
begin
case({s1,s2})
2'b00:if(count[7]==1)
dacout<=8'b11111111;
else
dacout<=8'd0;
2'b01: dacout<=count;
2'b10:if(count[7]==1)
dacout<=count; else
dacout<= ~count;
default: dacout<=8'd0;
endcase
end
endmodule
```

**DC MOTOR:**

```

module dcmotor(
    input [2:0] psw,
    input clk,
    output reg pdcm
);
reg [11:0] clk1=12'd0;
reg vdcn;

always@(posedge clk)
begin
    clk1=clk1+1;
    if (clk1==12'b101110111000)
    clk1=12'd0;
end

always@(clk1,psw)
begin

    if(clk1 == 12'd0)
    vdcn = 1;

        if (psw ==3'b000 && clk1 == 12'b000111110100) vdcn =0;
    else if (psw ==3'b001 && clk1 == 12'b001100100000) vdcn =0;
    else if (psw ==3'b010 && clk1 == 12'b010001001100) vdcn =0;
    else if (psw ==3'b011 && clk1 == 12'b010101111000) vdcn =0;
    else if (psw ==3'b100 && clk1 == 12'b011010100100) vdcn =0;
    else if (psw ==3'b101 && clk1 == 12'b011111010000) vdcn =0;
    else if (psw ==3'b110 && clk1 == 12'b100011111100) vdcn =0;
    else if (psw ==3'b111 && clk1 == 12'b100111000100) vdcn =0;

    if(vdcn ==1)
        pdcm =1;
    else
        pdcm =0;

end

endmodule

```



