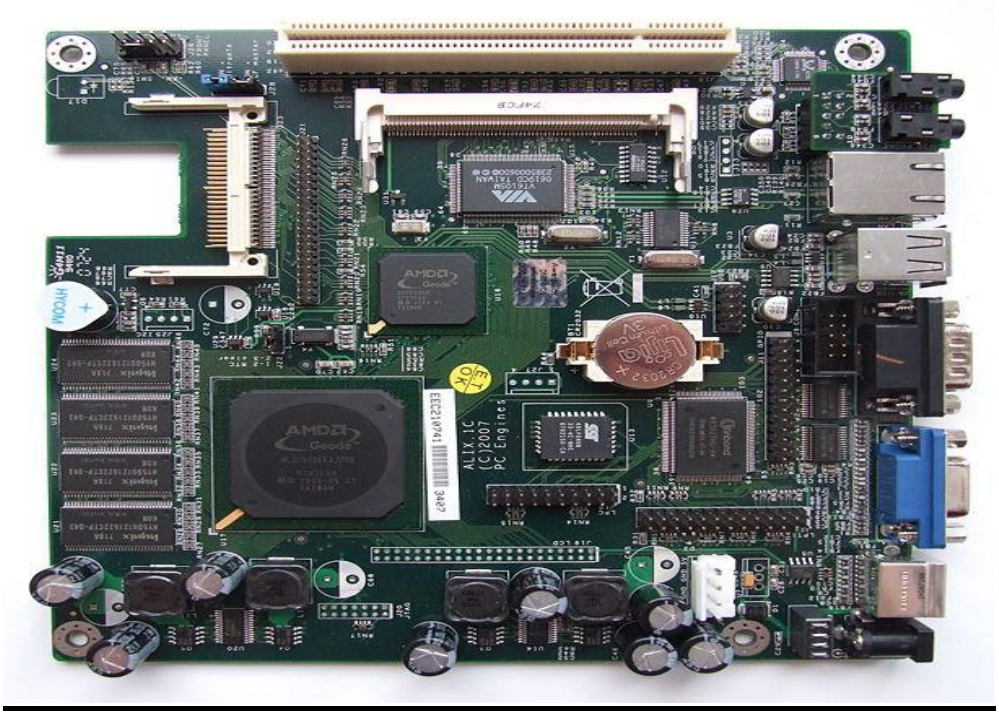
MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

LABORATARY MANUAL OF

DIGITAL LOGIC DESIGN LAB

II YEAR B.TECH I-SEM ECE (R-25)

A.Y:2026-27



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

PREPARED BY
D.MALATHI RANI
S. SINDHU REKHA

CERTIFICATE

This is to certify that this manual is a bonafide record of practical work in the ***Digital Logic Design Laboratory*** in **Third Semester of II- year B.Tech (ECE) Programme** during the academic year **2026-2027.** This book is prepared by **Mrs. D.Malathi Rani (Assistant Professor), Mrs.S.Sindhu Rekha (Assistant Professor)** Department of Electronics and Communication Engineering.

LAB I/C **Head of the Department**

PREFACE

This laboratory lays a strong foundation for Electronics and Communication Engineering students during the second year of their course. The Digital Logic Design Laboratory is divided into two parts: Part-A and Part-B.

In Part-A, students design and analyze circuits related to Digital Logic. In Part-B, students will design the circuits of digital system and implement them in Cadence tool. After performing all the experiments included in this Laboratory, it is hoped the student receives good training to handle any electronic equipment available in electronics field.

By,

Mrs. D.Malathi Rani
Mrs. S. Sindhu Rekha

ACKNOWLEDGEMENT

It was really a good experience, working with *Digital Logic Design Laboratory*. First we would like to thank Dr. N. Srinivas, Associate Professor, HOD of Department of Electronics and Communication Engineering, Marri Laxman Reddy Institute of technology & Management for his concern and giving the technical support in preparing the document.

We are deeply indebted and gratefully acknowledge the constant support and valuable patronage of Dr. Ravi Prasad, Dean, Marri Laxman Reddy Institute of technology & Management for giving us this wonderful opportunity for preparing the *Digital Logic Design Laboratory* manual.

We express our hearty thanks to Dr. R. Murali Prasad, Principal, Marri Laxman Reddy Institute of technology & Management, for timely corrections and scholarly guidance.

We express our hearty thanks to Dr. P. Sridhar, Director, Marri Laxman Reddy Institute of technology & Management, for timely corrections and scholarly guidance.

At last, but not the least I would like to thanks the entire ECE Department faculty those who had inspired and helped us to achieve our goal.

By,

Mrs. D.Malathi Rani
Mrs. S. Sindhu Rekha

GENERAL INSTRUCTIONS

1. Students are instructed to come to *Digital Logic Design* laboratory on time. Late comers are not entertained in the lab.
2. Students should be punctual to the lab. If not, the conducted experiments will not be repeated.
3. Students are expected to come prepared at home with the experiments which are going to be performed.
4. Students are instructed to display their identity cards before entering into the lab.
5. Students are instructed not to bring mobile phones to the lab.
6. Any damage/loss of equipments like IC's Trainer kits etc., during the lab session, it is student's responsibility and penalty or fine will be collected from the student.
7. Students should update the records and lab observation books session wise. Before leaving the lab the student should get his lab observation book signed by the faculty.
8. Students should submit the lab records by the next lab to the concerned faculty members in the staffroom for their correction and return.
9. Students should not move around the lab during the lab session.
10. If any emergency arises, the student should take the permission from faculty member concerned in written format.
11. The faculty members may suspend any student from the lab session on disciplinary grounds.
12. Never copy the output from other students. Write down your own outputs.

SAFETY MEASURES

1. While working in the laboratory suitable precautions should be observed to prevent accidents.
2. Always follow the experimental instructions strictly.
3. Use the first aid box in case of any accident/mishap.
4. Never work in the laboratory unless a demonstrator or teaching assistant in present.
5. When the experiment is completed, students should disconnect the setup made by them, and should return all the components/instruments taken for the purpose.

INSTITUTION VISION AND MISSION

VISION

To be a globally recognized institution that fosters innovation, excellence, and leadership in education, research, and technology development, empowering students to create sustainable solutions for the advancement of society.

MISSION

To foster a transformative learning environment that empowers students to excel in engineering, innovation, and leadership.

To produce skilled, ethical, and socially responsible engineers who contribute to sustainable technological advancements and address global challenges.

To shape future leaders through cutting-edge research, industry collaboration, and community engagement.

Quality Policy

The management is committed in assuring quality service to all its stakeholders, students, parents, alumni, employees, employers, and the community.

Our commitment and dedication are built into our policy of continual quality improvement by establishing and implementing mechanisms and modalities ensuring accountability at all levels, transparency in procedures, and access to information and actions.

DEPARTMENT VISION, MISSION, PROGRAM OUTCOMES AND PROGRAM SPECIFIC OUTCOMES

Vision of the Department

To provide quality technical education in Electronics and Communication Engineering through research, innovation, striving for global recognition in specified domain, leadership, and sustainable societal solutions.

Mission of the Department

- **DM1:** To create a transformative learning environment that empowers students in electronics and communication engineering, fostering excellence in technical skills and leadership.
- **DM2:** To drive innovation through research, deliver a transformative education grounded in ethical principles, and nurture the development of professionals
- **DM3:** To cultivate strong industry partnerships, and engaging actively with the community for societal and technological progress.

Program Outcomes (POs) Engineering Graduates will be able to:

- **PO1: Engineering Knowledge**: Apply knowledge of mathematics, natural science, computing, engineering fundamentals and an engineering specialization as specified in WK1 to WK4 respectively to develop to the solution of complex engineering problems.
- **PO2: Problem Analysis:** Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development
- **PO3: Design/Development of Solutions:** Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required.
- **PO4: Conduct Investigations of Complex Problems:** Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions.
- **PO5: Engineering Tool Usage:** Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems.
- **PO6: The Engineer and The World:** Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy,
- **PO7: Ethics:** Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws.
- **PO8: Individual and Collaborative Team work:** Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.
- **PO9: Communication:** Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences
- **PO10: Project Management and Finance:** Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments.
- **PO11: Life-Long Learning:** Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change.

Program Specific Outcomes (PSOs)

1. Design, develop, fabricate and commission the electrical systems involving power generation, transmission, distribution and utilization.
2. Focus on the components of electrical drives with its converter topologies for energy conversion, management and auditing in specific applications of industry and sustainable rural development.
3. Gain the hands-on competency skills and other computing tools necessary for entry level position to meet the requirements of the employer.

Program Educational Objectives (PEOs)

PEO 1: Have Successful career in Industry

Graduates will excel in the Electronics and Communication industry with a strong foundation in technical expertise, continuous learning, and innovation.

PEO 2: Show Excellence in higher studies/Research

Graduates will excel in higher studies and research in Electronics and Communication Engineering (ECE) through a combination of rigorous academic dedication, cutting-edge innovation, and a deep understanding of emerging technologies.

PEO 3: Show Good Competency towards Entrepreneurship

Graduates will have to show good competency towards entrepreneurship in the field of Electronics and Communication Engineering, one must demonstrate an in-depth understanding of emerging technologies, market trends, and the ability to innovate within this rapidly evolving industry.

Course Structure:

Level	Credits	Periods/Week	Prerequisites
UG	1	3	Entire Subject of Digital System Design & VLSI

Evaluation Scheme:

Each laboratory will be evaluated for a total of 100 marks consisting of 40 marks for Continuous Internal Evaluation (CIE) and 60 marks for semester end lab examination. Out of 40 marks for internal evaluation:

- A write-up on day-to-day experiment(aim,components/procedure, expected outcome) which shall be evaluated for 10 marks
- 10 marks for viva-voce/ tutorial/ case study/ application/ poster presentation.
- Internal practical examination shall be evaluated for 10 marks.
- The remaining 10 marks are for Laboratory Project (Design/ Software / Hardware Model/ App Development/ Prototype).

Table 1: CIE marks distribution

Component				
Type of Assessment	Day to Day performance and viva voce examination	Final internal lab assessment	Laboratory Report / Project and Presentation	Total Marks
CIE marks	20	10	10	40

Continuous Internal Evaluation (CIE): Two CIE exams shall be conducted at the 8th week and 16th week of the semester; the average of the two CIEs will be taken into account. The CIE exam is conducted for 10 marks.

The Semester End Examination shall be conducted with an external examiner and the laboratory teacher. The external examiner shall be appointed from the other colleges which will be decided by the Head of the institution.

In the Semester End Examination held for 3 hours, total 60 marks are divided and allocated as shown below:

- 10 marks for write-up
- 15 for experiment/program
- 15 for evaluation of results
- 10 marks for presentation on another experiment/program in the same laboratory course and
- 10 marks for viva-voce on concerned laboratory course.

Course Objectives:

The students will try to learn

- Practical skills in analysing and simplifying Boolean expressions.
- Hands-on experience in designing combinational and sequential logic circuits.
- Digital system modelling using Verilog HDL.
- Simulating and verifying designs with EDA tools.
- Construction of modular digital systems such as counters, FSMs, and shift registers

Course Outcomes:

After successful completion of the course, students shall be able to

- Analyse and simplify Boolean expressions and implement them using logic gates and ICs.
- Design and realize combinational and sequential logic circuits using logic gate hardware.
- Model digital systems in Verilog HDL using dataflow, behavioural, and structural styles.
- Simulate and verify digital designs using industry-standard EDA tools and testbenches.
- Build modular and hierarchical designs such as counters, FSMs, and shift registers.

Course Outcomes (CO's) – Program Outcomes (PO's) Mapping:

CO1: Analyse and simplify Boolean expressions and implement them using logic gates and ICs.
CO2: Design and realize combinational and sequential logic circuits using logic gate hardware.
CO3: Model digital systems in Verilog HDL using dataflow, behavioural, and structural styles.
CO4: Simulate and verify digital designs using industry-standard EDA tools and testbenches.
CO5: Build modular and hierarchical designs such as counters, FSMs, and shift registers.

MAPPING OF EACH CO WITH PO(s),PSO(s):

Course Outcomes	PROGRAM OUTCOMES														
	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO 3
CO1	✓	✓	✓	✓	✓	✓	-	-	-	-	-	✓	✓		✓
CO2	✓	✓	✓	✓	✓	✓	-	-	-	-	-	✓	-	-	✓
CO3	✓	✓	✓	✓	✓	✓	-	-	-	-	-	✓	-	-	✓
CO4	✓	✓	✓	✓	✓	✓	-	-	-	-	-	✓	✓	✓	✓
CO5	✓	✓	✓	✓	✓	-	-	-	-	-	-	✓	✓	✓	✓

JUSTIFICATIONS FOR CO – PO / PSO MAPPING - DIRECT:

Course Outcomes	PO'S/ PSO'S	Justification for mapping (Students will be able to)	No. of Key Competencies
CO1	PO1	1. Scientific principles and methodology 2. Mathematical principles 3. Own and / or other engineering disciplines to integrate / support study of their own engineering discipline.	3
	PO2	1. Problem or opportunity identification 2. Problem statement and system definition 3. Problem formulation and abstraction 4. Information and data collection 5. Model translation 6. Validation 7. Experimental design 8. Solution development or experimentation /Implementation 9. Interpretation of results	9
	PO3	1. Investigate and define a problem and identify constraints including environmental and sustainability limitations, health and safety and risk assessment issues; 2. Understand customer and user needs and the	6

		importance of considerations such as aesthetics; 3. Use creativity to establish innovative solutions; 4. Manage the design process and evaluate outcomes. 5. Knowledge of management techniques which may be used to achieve engineering objectives within that context; 6. Understanding of the requirement for engineering activities to promote sustainable development;	
	PO4	1. Knowledge of characteristics of particular materials, equipment, processes, or products; 2. Workshop and laboratory skills; 3. Understanding of contexts in which engineering knowledge can be applied (example, operations and management, technology development, etc.); 4. Understanding use of technical literature and other information sources Awareness of nature of intellectual property and contractual issues; 5. Understanding of appropriate codes of practice and industry standards; 6. Ability to work with technical uncertainty. 7. Understanding of engineering principles and the ability to apply them to analyze key engineering processes; 8. Ability to apply quantitative methods and computer software relevant to their engineering discipline, in order to solve engineering problems; 9. Understanding of and ability to apply a systems approach to engineering problems.	9
	PO5	1. Computer software/ simulation packages/ diagnostic equipment/ technical library resources/ literature search tools.	`1
	PO6	1. Knowledge of management techniques which may be used to achieve engineering objectives within that context; 2. Understanding of the requirement for engineering activities to promote sustainable development;	2

	PO12	1. Keeping current in ECE and advanced engineering concepts 2. Personal continuing education efforts 3. Ongoing learning – stays up with industry trends/ new technology 4. Continued personal development	4
	PSO1	1. Adopt the engineering professional code and conduct 2. Sustainable and Compliant Design	2
	PSO3	1. Adopt technical library resources and literature search. 2. Research, analysis, problem solving	2
CO2	PO1	1. Scientific principles and methodology 2. Mathematical principles 3. Own and / or other engineering disciplines to integrate / support study of their own engineering discipline.	3
	PO2	1. Problem or opportunity identification 2. Problem statement and system definition 3. Problem formulation and abstraction 4. Information and data collection 5. Model translation 6. Validation 7. Experimental design 8. Solution development or experimentation /Implementation 9. Interpretation of results	9
	PO3	1. Investigate and define a problem and identify constraints including environmental and sustainability limitations, health and safety and risk assessment issues; 2. Understand customer and user needs and the importance of considerations such as aesthetics; 3. Use creativity to establish innovative solutions; 4. Manage the design process and evaluate outcomes. 5. Knowledge of management techniques which may be used to achieve engineering objectives within that context; 6. Understanding of the requirement for	6

		engineering activities to promote sustainable development;	
	PO4	1. Knowledge of characteristics of particular materials, equipment, processes, or products; 2. Workshop and laboratory skills; 3. Understanding of contexts in which engineering knowledge can be applied (example, operations and management, technology development, etc.); 4. Understanding use of technical literature and other information sources Awareness of nature of intellectual property and contractual issues; 5. Understanding of appropriate codes of practice and industry standards; 6. Ability to work with technical uncertainty. 7. Understanding of engineering principles and the ability to apply them to analyze key engineering processes; 8. Ability to apply quantitative methods and computer software relevant to their engineering discipline, in order to solve engineering problems; 9. Understanding of and ability to apply a systems approach to engineering problems.	9
	PO5	1. Computer software/simulation packages/diagnostic equipment/technical library resources/ literature search tools.	1
	PO6	1. Knowledge of management techniques which may be used to achieve engineering objectives within that context; 2. Understanding of the requirement for engineering activities to promote sustainable development;	2
	PO12	1. Keeping current in ECE and advanced engineering concepts 2. Personal continuing education efforts 3. Ongoing learning – stays up with industry trends/ new technology 4. Continued personal development	4
	PSO3	1. Adopt technical library resources and literature search. 2. Research, analysis, problem solving	2

CO3	PO1	1. Scientific principles and methodology 2. Mathematical principles 3. Own and / or other engineering disciplines to integrate / support study of their own engineering discipline.	3
	PO2	1. Problem or opportunity identification 2. Problem statement and system definition 3. Problem formulation and abstraction 4. Information and data collection 5. Model translation 6. Validation 7. Experimental design 8. Solution development or experimentation /Implementation 9. Interpretation of results	9
	PO3	1. Investigate and define a problem and identify constraints including environmental and sustainability limitations, health and safety and risk assessment issues; 2. Understand customer and user needs and the importance of considerations such as aesthetics; 3. Use creativity to establish innovative solutions; 4. Manage the design process and evaluate outcomes. 5. Knowledge of management techniques which may be used to achieve engineering objectives within that context; 6. Understanding of the requirement for engineering activities to promote sustainable development;	6
	PO4	1. Knowledge of characteristics of particular materials, equipment, processes, or products; 2. Workshop and laboratory skills; 3. Understanding of contexts in which engineering knowledge can be applied (example, operations and management, technology development, etc.); 4. Understanding use of technical literature and other information sources Awareness of nature of intellectual property and contractual issues; 5. Understanding of appropriate codes of practice and industry standards; 6. Ability to work with technical uncertainty.	9

		7. Understanding of engineering principles and the ability to apply them to analyze key engineering processes; 8. Ability to apply quantitative methods and computer software relevant to their engineering discipline, in order to solve engineering problems; 9. Understanding of and ability to apply a systems approach to engineering problems.	
	PO5	1. Computer software/ simulation packages/ diagnostic equipment/ technical library resources/ literature search tools.	1
	PO6	1. Knowledge of management techniques which may be used to achieve engineering objectives within that context; 2. Understanding of the requirement for engineering activities to promote sustainable development;	2
	PO12	1. Keeping current in ECE and advanced engineering concepts 2. Personal continuing education efforts 3. Ongoing learning – stays up with industry trends/ new technology 4. Continued personal development	4
	PSO3	1. Adopt technical library resources and literature search. 2. Research, analysis, problem solving	2
CO4	PO1	1. Scientific principles and methodology 2. Mathematical principles 3. Own and / or other engineering disciplines to integrate / support study of their own engineering discipline.	3
	PO2	1. Problem or opportunity identification 2. Problem statement and system definition 3. Problem formulation and abstraction 4. Information and data collection 5. Model translation 6. Validation 7. Experimental design 8. Solution development or experimentation /Implementation 9. Interpretation of results	9
	PO3	1. Investigate and define a problem and identify	6

		constraints including environmental and sustainability limitations, health and safety and risk assessment issues; 2. Understand customer and user needs and the importance of considerations such as aesthetics; 3. Use creativity to establish innovative solutions; 4. Manage the design process and evaluate outcomes. 5. Knowledge of management techniques which may be used to achieve engineering objectives within that context; 6. Understanding of the requirement for engineering activities to promote sustainable development;	
	PO4	1. Knowledge of characteristics of particular materials, equipment, processes, or products; 2. Workshop and laboratory skills; 3. Understanding of contexts in which engineering knowledge can be applied (example, operations and management, technology development, etc.); 4. Understanding use of technical literature and other information sources Awareness of nature of intellectual property and contractual issues; 5. Understanding of appropriate codes of practice and industry standards; 6. Ability to work with technical uncertainty. 7. Understanding of engineering principles and the ability to apply them to analyze key engineering processes; 8. Ability to apply quantitative methods and computer software relevant to their engineering discipline, in order to solve engineering problems; 9. Understanding of and ability to apply a systems approach to engineering problems.	9
	PO5	1. Computer software/ simulation packages/ diagnostic equipment/ technical library resources/ literature search tools.	1
	PO6	1. Knowledge of management techniques which may be used to achieve engineering objectives within that context;	2

		2. Understanding of the requirement for engineering activities to promote sustainable development;	
	PO12	1. Keeping current in ECE and advanced engineering concepts 2. Personal continuing education efforts 3. Ongoing learning – stays up with industry trends/ new technology 4. Continued personal development	4
	PSO1	1. Adopt the engineering professional code and conduct	1
	PSO2	1. Adopt the engineering professional code and conduct.	1
	PSO3	1. Adopt technical library resources and literature search. 2. Research, analysis, problem solving	2
CO5	PO1	1. Scientific principles and methodology 2. Mathematical principles 3. Own and / or other engineering disciplines to integrate / support study of their own engineering discipline.	3
	PO2	1. Problem or opportunity identification 2. Problem statement and system definition 3. Problem formulation and abstraction 4. Information and data collection 5. Model translation 6. Validation 7. Experimental design 8. Solution development or experimentation /Implementation 9. Interpretation of results	9
	PO3	1. Investigate and define a problem and identify constraints including environmental and sustainability limitations, health and safety and risk assessment issues; 2. Understand customer and user needs and the importance of considerations such as aesthetics; 3. Use creativity to establish innovative solutions; 4. Manage the design process and evaluate outcomes. 5. Knowledge of management techniques which may be used to achieve engineering objectives	6

		within that context; 6. Understanding of the requirement for engineering activities to promote sustainable development;	
	PO4	1. Knowledge of characteristics of particular materials, equipment, processes, or products; 2. Workshop and laboratory skills; 3. Understanding of contexts in which engineering knowledge can be applied (example, operations and management, technology development, etc.); 4. Understanding use of technical literature and other information sources Awareness of nature of intellectual property and contractual issues; 5. Understanding of appropriate codes of practice and industry standards; 6. Ability to work with technical uncertainty. 7. Understanding of engineering principles and the ability to apply them to analyze key engineering processes; 8. Ability to apply quantitative methods and computer software relevant to their engineering discipline, in order to solve engineering problems; 9. Understanding of and ability to apply a systems approach to engineering problems.	9
	PO5	1. Computer software/ simulation packages/ diagnostic equipment/ technical library resources/ literature search tools.	1
	PO6	1. Knowledge of management techniques which may be used to achieve engineering objectives within that context; 2. Understanding of the requirement for engineering activities to promote sustainable development;	2
	PO12	1. Keeping current in ECE and advanced engineering concepts 2. Personal continuing education efforts 3. Ongoing learning – stays up with industry trends/ new technology	4

		4. Continued personal development	
	PSO1	1. Adopt the engineering professional code and conduct	1
	PSO2	1. Adopt the engineering professional code and conduct.	1
	PSO3	1. Adopt technical library resources and literature search. 2. Research, analysis, problem solving	2

TOTAL COUNT OF KEY COMPETENCIES FOR CO – (PO, PSO) MAPPING:

Course Outcomes	**PROGRAM OUTCOMES**														
	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO 3
	4	10	10	10	4	5	4	4	10	5	10	8	6	10	8
CO1	3	9	6	9	1	2	-	-	-	-	-	4	2	-	2
CO2	3	9	6	9	1	2	-	-	-	-	-	4	-	-	2
CO3	3	9	6	9	1	2	-	-	-	-	-	4	-	-	2
CO4	3	9	6	9	1	2	-	-	-	-	-	4	1	1	2
CO5	3	9	6	9	1	2	-	-	-	-	-	4	1	1	2

PERCENTAGE OF KEY COMPETENCIES FOR CO – (PO/ PSO):

Course Outcomes	PROGRAM OUTCOMES														
	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO 3
CO1	75	90	60	90	25	40	-	-	-	-	-	50	33	-	25
CO2	75	90	60	90	25	40	-	-	-	-	-	50	-	-	25
CO3	75	90	60	90	25	40	-	-	-	-	-	50	-	-	25
CO4	75	90	60	90	25	40	-	-	-	-	-	50	17	10	25
CO5	75	90	60	90	25	40	-	-	-	-	-	50	17	10	25

COURSE ARTICULATION MATRIX (PO – PSO MAPPING):

CO'S and PO'S, CO'S and PSO'S on the scale of 0 to 3, 0 being no correlation, 1 being the low correlation, 2 being medium correlation and 3 being high correlation.

0 - 0≤ C≤ 5% – No correlation, 2 - 40 % <C < 60% –Moderate

1-5 <C≤ 40% – Low/ Slight 3 - 60% ≤ C < 100% – Substantial /High

Course Outcomes	PROGRAM OUTCOMES														
	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO 3
CO1	3	3	3	3	1	2	-	-	-	-	-	2	1	-	1
CO2	3	3	3	3	1	2	-	-	-	-	-	2	-	-	1
CO3	3	3	3	3	1	2	-	-	-	-	-	2	-	-	1
CO4	3	3	3	3	1	2	-	-	-	-	-	2	1	1	1
CO5	3	3	3	3	1	2	-	-	-	-	-	2	1	1	1
Total	15	15	15	15	5	10	-	-	-	-	-	10	3	2	5
Average	3	3	3	3	1	2	-	-	-	-	-	2	1	1	1

LAB TIME TABLE

ROOM NO: SR-110

NAME OF THE LAB: DLD LAB **A.Y:** 2026-2027

BRANCH: ECE **W.E. F:**

SEMESTER: I

PERIOD	1st	2nd	3rd	12:25 - 01:15	4th	5th	6th
TIME / DAY	09:40 - 10:35	10:35 - 11:30	11:30 - 12:25		01:15 - 02:10	02:10 - 03:05	03:05 - 04:00
MON				LUNCH			
TUE							
WED							
THU							
FRI							
SAT							

Time Table I/C **Time Tables C/D** **HOD – ECE**

COURSE CONTENT

DIGITAL LOGIC DESIGN LABORATORY								
III Semester: ECE								
Course Code	**Category**	**Hours/ Week**			**Credits**	**Maximum Marks**		
2530473	**Core**	**L**	**T**	**P**	**C**	**CIA**	**SEE**	**Total**
		0	**0**	**2**	**1**	**40**	**60**	**100**
Contact Classes: Nill	**Tutorial Classes: Nil**	**Practical Classes: 32**				**Total Classes:32**		
Prerequisites: Nill								

Couse Overview:

This laboratory course offers practical experience in designing, analysing, and simulating digital circuits. Students start with basic hardware implementation using logic gate ICs, covering Boolean simplification, arithmetic circuits, code converters, and other combinational modules. The course then progresses to Verilog HDL-based design, introducing dataflow, behavioural, and structural modelling with simulation tools. Emphasis is placed on strengthening core logic concepts while developing skills for modern digital system design.

Course Objectives:

The students will try to learn

- Practical skills in analysing and simplifying Boolean expressions.
- Hands-on experience in designing combinational and sequential logic circuits.
- Digital system modelling using Verilog HDL.
- Simulating and verifying designs with EDA tools.
- Construction of modular digital systems such as counters, FSMs, and shift registers.

Course Outcomes:

After successful completion of the course, students shall be able to

- Analyse and simplify Boolean expressions and implement them using logic gates and ICs.
- Design and realize combinational and sequential logic circuits using logic gate hardware.
- Model digital systems in Verilog HDL using dataflow, behavioural, and structural styles.
- Simulate and verify digital designs using industry-standard EDA tools and testbenches.
- Build modular and hierarchical designs such as counters, FSMs, and shift registers.

List of Experiments:

A. Realization in Hardware Laboratory (Using Logic ICs)

These are fundamental hands-on experiments conducted using logic ICs such as AND, OR, NOT, NAND, NOR, XOR gates, flip-flops, multiplexers, and decoders.

1. Realize and minimize Boolean functions using basic gates and universal gates (NAND/NOR) in SOP/POS form.
2. Design and implement Half Adder, Full Adder using logic gates.
3. Design and implement Half Subtractor, and Full Subtractor using logic gates.
4. Construct and analyse basic logic gates (AND, OR, NOT, XOR, XNOR) using only NAND and NOR gates.
5. Design and implement code converters such as Binary to Gray, Gray to Binary, using gates.
6. Design and implement simple combinational circuits: 2-to-1 multiplexer, 1-bit comparator

B.Verilog HDL-Based Digital Design Experiments (Simulation-Based)

These experiments are implemented using Verilog HDL with different modelling styles (dataflow, behavioural, structural) and simulated using tools like Vivado/Model Sim/Xilinx ISE/cadence/ any equivalent.

1. Design and simulate a 2-bit comparator using dataflow modelling; extend it to 4-bit using structural modelling.
2. Implement a 2:1 multiplexer using dataflow modelling and design an 8:1 multiplexer using structural modelling.
3. Design a 2-to-4 decoder using dataflow modelling and realize a 3-to-8 decoder using structural modelling.
4. Implement a given Boolean function using a decoder-based approach in behavioural modelling.
5. Design and simulate a universal n-bit shift register (left, right, hold, parallel load) using behavioural modelling.
6. Design a synchronous MOD-n counter using behavioural modelling with D or JK flip-flops.
7. Design and simulate an asynchronous (ripple) counter for a custom sequence using structural modelling.
8. Implement a sequence detector for a given binary pattern using FSM (Moore/Mealy) in behavioural modelling.

Open Ended Experiments

Implementation of binary multiplier
Design and simulation of Registers
Implementation of binary counter and simulate using simulation tool

NOTE: Minimum 5 experiments from each PART to be conducted.

Reference Link

1. https://onlinecourses.nptel.ac.in/noc25_cs25/preview
2. https://nptel.ac.in/courses/106103358
3. https://www.nptelprep.in/courses/106103358
4. https://onlinecourses.nptel.ac.in/noc25_ee180/preview
5. https://onlinecourses.nptel.ac.in/noc26_ee71/preview
6. https://onlinecourses-archive.nptel.ac.in/noc18_cs30
7. https://onlinecourses-archive.nptel.ac.in/noc18_cs48/course

Materials Online:

1. Lab Manual
2. Open-ended experiments

DIGITAL LOGIC DESGN LABORATORY

Virtual lab details

Name of the Virtual Lab: DLD with Virtual Laboratory Virtual

Lab Host Institute: IIT ROORKE, India

URL/Link to Lab: https://de-iitr.vlabs.ac.in/

Academic Year: 2026-2027

Semester: III

List of Experiments Available in Virtual Lab

1.Realize and minimize Boolean functions using basic gates and universal gates (NAND/NOR) in SOP/POS form.
2.Design and implement Half Adder, Full Adder using logic gates.
3.Design and implement Half Subtractor, and Full Subtractor using logic gates.
4.Construct and analyse basic logic gates (AND, OR, NOT, XOR, XNOR) using only NAND and NOR gates.
5.Design and implement code converters such as Binary to Gray, Gray to Binary, using gates.
6.Design and implement simple combinational circuits: 2-to-1 multiplexer, 1-bit comparator

7.Design and simulate a 2-bit comparator using dataflow modelling; extend it to 4-bit using structural modelling.
8.Implement a 2:1 multiplexer using dataflow modelling and design an 8:1 multiplexer using structural modelling.
9.Design a 2-to-4 decoder using dataflow modelling and realize a 3-to-8 decoder using structural modelling.
10.Implement a given Boolean function using a decoder-based approach in behavioural modelling.
11.Design and simulate a universal n-bit shift register (left, right, hold, parallel load) using behavioural modelling.
12.Design a synchronous MOD-n counter using behavioural modelling with D or JK flip-flops.
13..Design and simulate an asynchronous (ripple) counter for a custom sequence using structural modelling.
14.Implement a sequence detector for a given binary pattern using FSM (Moore/Mealy) in behavioural modelling.

DIGITAL LOGIC DESGN LABORATORY

LAB PLANNER

S.No	Experiment	CO	Virtual Lab Availability	Date planned	Date conducted
1	Realize and minimize Boolean functions using basic gates and universal gates (NAND/NOR) in SOP/POS form.	1	Yes		
2	Design and implement Half Adder, Full Adder using logic gates.	1	Yes		
3	Design and implement Half Subtractor, and Full Subtractor using logic gates.	2	Yes		
4	Construct and analyse basic logic gates (AND, OR, NOT, XOR, XNOR) using only NAND and NOR gates.	1	No		
5	Design and implement code converters such as Binary to Gray, Gray to Binary, using gates.	1	Yes		
6	Design and implement simple combinational circuits: 2-to-1 multiplexer, 1-bit comparator	1	No		
	MID-I				
7	Design and simulate a 2-bit comparator using dataflow modelling; extend it to 4-bit using structural modelling.	5	Yes		
8	Implement a 2:1 multiplexer using dataflow modelling and design an 8:1 multiplexer using structural modelling.	5	Yes		
9	Design a 2-to-4 decoder using dataflow modelling and realize a 3-to-8 decoder using structural modelling.	4	Yes		
10	Implement a given Boolean function using a decoder-based approach in behavioural modelling.	4	Yes		
11	Design and simulate a universal n-bit shift register (left, right, hold, parallel load) using behavioural modelling.	4	Yes		
12	Design a synchronous MOD-n counter using behavioural modelling with D or JK flip-flops.	4	Yes		

13	Design and simulate an asynchronous (ripple) counter for a custom sequence using structural modelling.	4	Yes		
14	Implement a sequence detector for a given binary pattern using FSM (Moore/Mealy) in behavioural modelling.	5	Yes		
	MID-II				

DIGITAL LOGIC DESGN LABORATORY

LAB PLANNER

Date planed																		
Date conducted																		
Roll Number	**Exp No**	**CO**	**VL**	**Exp No**	**CO**	**VL**	**Exp No**	**CO**	**VL**	**Exp No**	**CO**	**VL**	**Exp No**	**CO**	**VL**	**Exp No**	**CO**	**VL**	
	1	1	Y	2	1	Y	3	2	Y	4	1	N	5	1	Y	6	1	N	
	1	1	Y	2	1	Y	3	2	Y	4	1	N	5	1	Y	6	1	N	**M**
	1	1	Y	2	1	Y	3	2	Y	4	1	N	5	1	Y	6	1	N	**I**
	1	1	Y	2	1	Y	3	2	Y	4	1	N	5	1	Y	6	1	N	**D**
	1	1	Y	2	1	Y	3	2	Y	4	1	N	5	1	Y	6	1	N	-
	1	1	Y	2	1	Y	3	2	Y	4	1	N	5	1	Y	6	1	N	**I**

Note: VL*-Virtual Lab Availability

Date planed																			
Date conducted																			
Roll Number	**Exp No**	**CO**	**VL**	**Exp No**	**CO**	**VL**	**Exp No**	**CO**	**VL**	**Exp No**	**CO**	**VL**	**Exp No**	**CO**	**VL**	**Exp No**	**CO**	**VL**	
	7	4	Y	8	4	Y	9	4	Y	10	4	Y	11	4	Y	12	5	Y	
	7	4	Y	8	4	Y	9	4	Y	10	4	Y	11	4	Y	12	5	Y	**M**
	7	4	Y	8	4	Y	9	4	Y	10	4	Y	11	4	Y	12	5	Y	**I**
	7	4	Y	8	4	Y	9	4	Y	10	4	Y	11	4	Y	12	5	Y	**D**
	7	4	Y	8	4	Y	9	4	Y	10	4	Y	11	4	Y	12	5	Y	**-**
	7	4	Y	8	4	Y	9	4	Y	10	4	Y	11	4	Y	12	5	Y	**II**

Note: VL*-Virtual Lab Availability

DIGITAL LOGIC DESGN LABORATORY

RUBRICS USED TO ASSESS LEARNINGS IN LABORATORIES

1. RUBRICS FOR DAY TO DAY EVALUATION

Parameter	Max Marks	Level-1 (Very Poor)	Level-2 (Poor)	Level-3 (Average)	Level-4 (Good)	Level-5 (Excellent)
Observation Book	05	No observations or irrelevant data. (0–1)	Incomplete or incorrect data. (2)	Basic values with some errors. (3)	Mostly correct with good format. (4)	Fully correct, clear, and well-formatted. (5)
Record Writing	05	Not submitted. (0-1)	Submitted but mostly incomplete. (2)	Submitted with some missing/wrong parts. (3)	Submitted with minor issues. (4)	Fully complete, correct algorithm & flowchart. (5)
Result	05	No result or major errors. (0–1)	Result partially obtained. (2)	Acceptable result with limited error. (3)	Near-correct result and reasonable error. (4)	Accurate result. (5)
Viva-Voce	05	Did not answer any questions. (1)	Answered very few questions. (2)	Answered some questions with help. (3)	Answered most questions correctly. (4)	Answered all questions accurately. (5)

DIGITAL LOGIC DESGN LABORATORY

2. RUBRICS FOR INTERNAL EVALUATION

Criterion	Max Marks	Level-1 *(Very Poor)*	Level-2 *(Poor)*	Level-3 *(Average)*	Level-4 *(Good)*	Level-5 *(Excellent)*
Design/Tool/Apparatus Selection	2 Marks	Incorrect tool/design and no reasoning. **(0)**	Tool/design selection attempted with unclear logic. **(0.5)**	Satisfactory selection with partial justification. **(1)**	Correct selection and proper analysis with few errors. **(1.5)**	Smart selection with accurate, relevant analysis. **(2)**
Execution (Code/Debug/Run) /Analysis/Method Used	4 Marks	Did not attempt or completely failed to execute. **(0)**	Attempted but unable to proceed or with major errors. **(1)**	Partial execution with some logic/syntax errors. **(2)**	Mostly correct execution with minimal help. **(3)**	Fully correct and independently executed program. **(4)**
Results& Documentation	2 Marks	Incomplete or poorly presented. **(0)**	Basic structure but lacks clarity or formatting. **(0.5)**	Complete but generic or with formatting issues. **(1)**	Well-structured and mostly clear. **(1.5)**	Well-organized, professional, and engaging documentation. **(2)**
Viva-Voce (Understanding of Concepts)	2 Marks	No understanding; could not answer questions. **(0)**	Answered a few with difficulty. **(0.5)**	Answered half the questions with basic clarity. **(1)**	Good understanding with confident answers. **(1.5)**	Answered all questions with clarity and depth. **(2)**

DIGITAL LOGIC DESGN LABORATORY

3. RUBRICS FOR SEMESTER END EXAMINATIONS

Criterion	Max Marks	Level-1 (Very Poor (*0–2 marks)*	Level-2 (Poor)*(3–4 marks)*	Level-3 (Average)*(5–6 marks)*	Level-4 (Good)*(7–9 marks)*	Level-5 (Excellent)*(10–12 marks)*
Preparedness for the Experiment	12 marks	No clarity on objective or procedure. Unable to explain basics.	Limited idea of the objective/procedure. Needed prompting.	Has basic understanding; minor gaps in concept or preparation.	Well-prepared, with clear understanding of steps and background.	Fully prepared with strong conceptual clarity and confident explanation.
Performance in the Laboratory	12 marks	Unable to perform experiment. Relied entirely on examiner's help.	Performed with multiple errors and constant support.	Performed with some errors; required occasional help.	Performed mostly independently with minimal support.	Performed independently, efficiently, and with precision.
Calculations & Graphs	12 marks	No or incorrect calculations. Graphs missing or irrelevant.	Multiple calculation errors. Graphs/plots inaccurate or poorly labelled.	Calculations partially correct. Graphs present but with some flaws.	Correct calculations and graphs with minor errors.	Accurate calculations and well-labelled graphs with proper interpretation.
Results & Error Analysis	12 marks	No result or invalid result. No error analysis attempted.	Incorrect result with vague or no error discussion.	Acceptable result. Error analysis attempted but limited.	Correct result with sound error discussion.	Accurate result with detailed and relevant error analysis.
Viva-Voce (Subject Knowledge)	12 marks	Unable to answer any questions. No conceptual understanding.	Answered few questions with poor logic.	Answered half of the questions with average understanding.	Answered most questions with clarity and confidence.	Answered all questions with depth, clarity, and reasoning.

INDEX

14	Implement a sequence detector for a given binary pattern using FSM (Moore/Mealy) in behavioural modelling.	
OPEN ENDED EXPERIMENTS		
1.	Implementation of Demorgan's Theorem	147

PART-A

Experiment No. 01

Realize and minimize Boolean functions using basic gates and universal gates (NAND/NOR) in SOP/POS form

AIM:

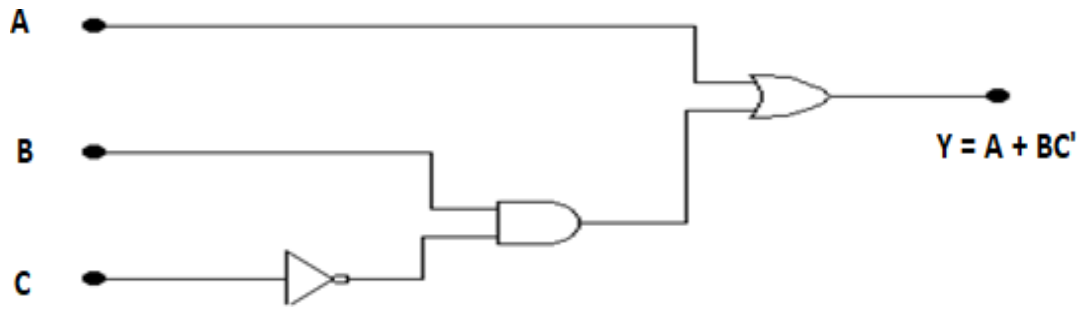
Realize and minimize Boolean functions using basic gates and universal gates (NAND/NOR) in SOP/POS form.

APPARATUS:

Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	AND GATE	IC 7408	1
2.	OR GATE	IC 7432	1
3.	NOT GATE	IC 7404	1
4.	DIGITAL IC TRAINER KIT		1
5.	CONNECTING WIRES		1

LOGIC DIAGRAM 1:

Figure 1.1 Logic Diagram 1

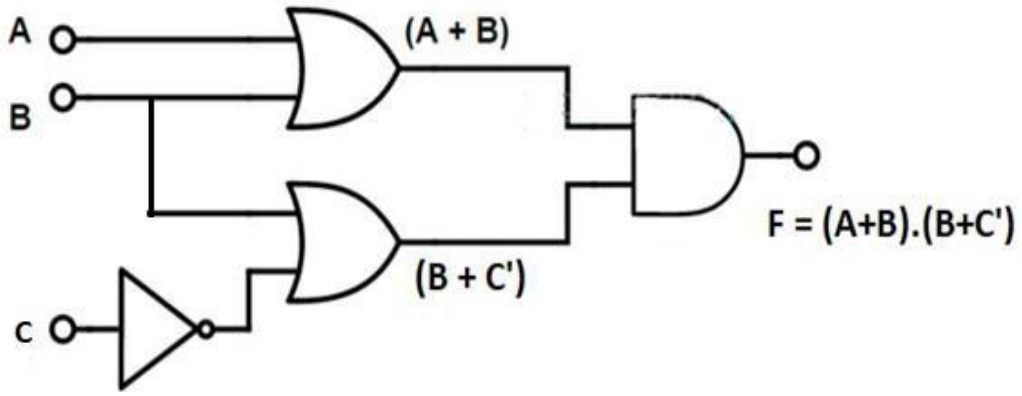


TRUTH TABLE:

A	B	C	C’	BC’	Y = A + BC’
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	1	1	1
0	1	1	0	0	0
1	0	0	1	0	1
1	0	1	0	0	1
1	1	0	1	1	1
1	1	1	0	0	1

LOGIC DIAGRAM 2

Figure 1.2 Logic Diagram 2

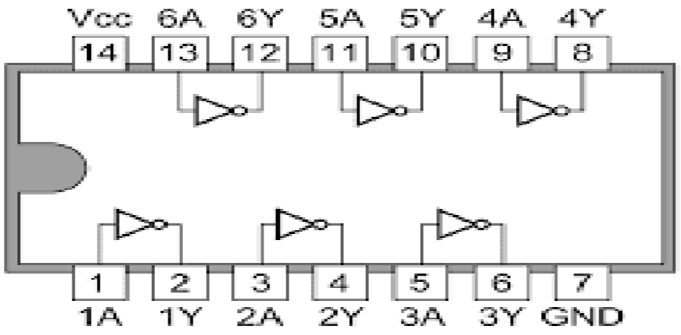


TRUTH TABLE:

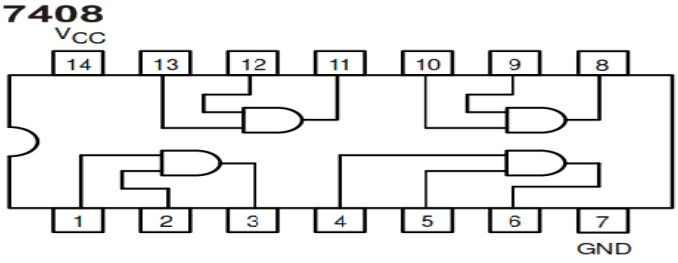
A	B	C	A+B	B+C'	F = (A + B).(B+C')
0	0	0	0	1	0
0	0	1	0	0	0
0	1	0	1	1	1
0	1	1	1	1	1
1	0	0	1	1	1
1	0	1	1	0	0
1	1	0	1	1	1
1	1	1	1	1	1

IC PIN DIAGRAMS:

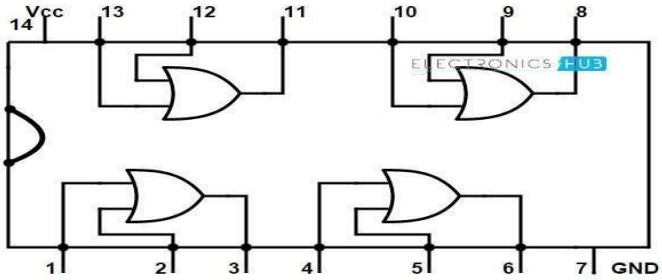
7404 Hex Inverters



7408



7432



To simplify the following Boolean Functions using Karnaugh map and implement the simplified functions using logic gates.

(i) $F_1 (X, Y, Z) = \Sigma m (3, 4, 6, 7)$

(ii) $F_2 (A, B, C, D) = \Sigma m (2, 6, 8, 9, 10, 11, 14)$

TRUTH TABLE:

$F_1 (X, Y, Z) = \Sigma m (3, 4, 6, 7)$

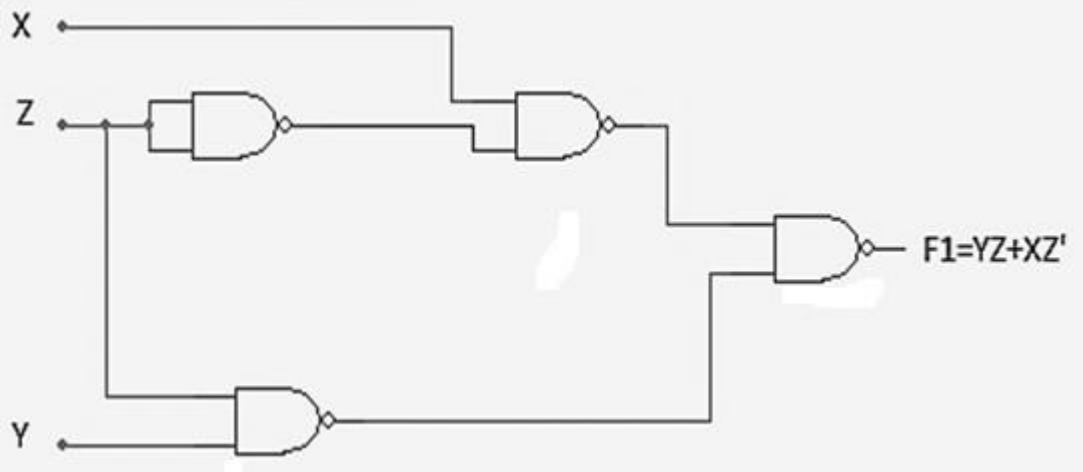
X	Y	Z	F_1
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

K-Map Simplification:

X \ YZ	00	01	11	10
0			1	
1	1		1	1

Simplified expression for F_1 using K-map is $F_1 = YZ + XZ'$

LOGIC DIAGRAM:

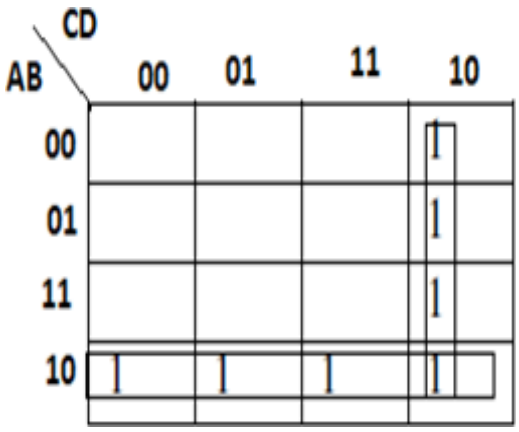


TRUTH TABLE:

F_2 (A, B, C, D) = Σm (2, 6, 8, 9, 10, 11, 14)

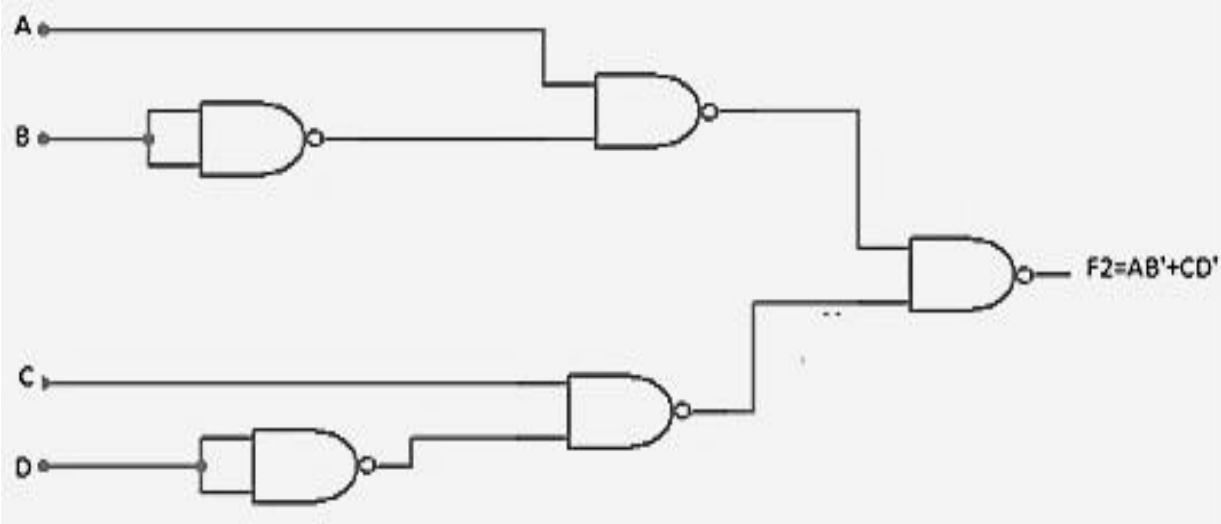
INPUTS				OUTPUT
A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

K-Map Simplification:



Simplified expression for F_2 using K-map is F_2 = AB'+CD'

LOGIC DIAGRAM

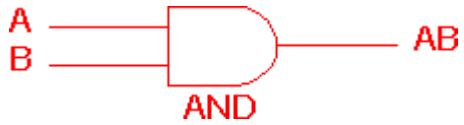


THEORY:

Logic gates

Digital systems are said to be constructed by using logic gates. These gates are the AND, OR, NOT, NAND, NOR, EXOR and EXNOR gates. The basic operations are described below with the aid of truth tables.

AND gate



2 Input AND gate		
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

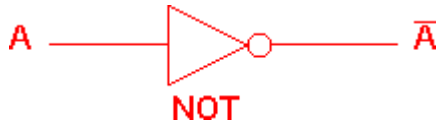
The AND gate is an electronic circuit that gives a **high** output (1) only if **all** its inputs are high. A dot (.) is used to show the AND operation i.e. A.B. Bear in mind that this dot is sometimes omitted i.e. AB

OR gate

2 Input OR gate		
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

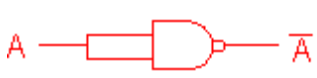
The OR gate is an electronic circuit that gives a high output (1) if **one or more** of its inputs are high. A plus (+) is used to show the OR operation.

NOT gate



NOT gate	
A	$\overline{A}$
0	1
1	0

The NOT gate is an electronic circuit that produces an inverted version of the input at its output. It is also known as an *inverter*. If the input variable is A, the inverted output is known as NOT A. This is also shown as A', or A with a bar over the top, as shown at the outputs. The diagrams below show two ways that the NAND logic gate can be configured to produce a NOT gate. It can also be done using NOR logic gates in the same way.

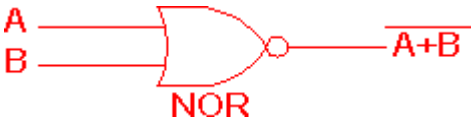


NAND gate

2 Input NAND gate		
A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

This is a NOT-AND gate which is equal to an AND gate followed by a NOT gate. The outputs of all NAND gates are high if **any** of the inputs are low. The symbol is an AND gate with a small circle on the output. The small circle represents inversion.

NOR gate



2 Input NOR gate		
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

This is a NOT-OR gate which is equal to an OR gate followed by a NOT gate. The outputs of all NOR gates are low if **any** of the inputs are high.

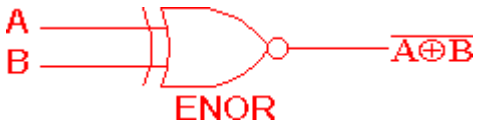
The symbol is an OR gate with a small circle on the output. The small circle represents inversion.

EXOR gate

2 Input EXOR gate		
A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

The '**Exclusive-OR**' gate is a circuit which will give a high output if **either, but not both**, of its two inputs are high. An encircled plus sign (⊕) is used to show the EOR operation.

EXNOR gate



2 Input EXNOR gate		
A	B	$\overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1

The '**Exclusive-NOR**' gate circuit does the opposite to the EOR gate. It will give a low output if **either, but not both**, of its two inputs are high. The symbol is an EXOR gate with a small circle on the output. The small circle represents inversion.

The NAND and NOR gates are called *universal functions* since with either one the AND and OR functions and NOT can be generated.

Note:

A function in *sum of products* form can be implemented using NAND gates by replacing all AND and OR gates by NAND gates.

A function in *product of sums* form can be implemented using NOR gates by replacing all AND and OR gates by NOR gates

Table 1: Logic gate symbols

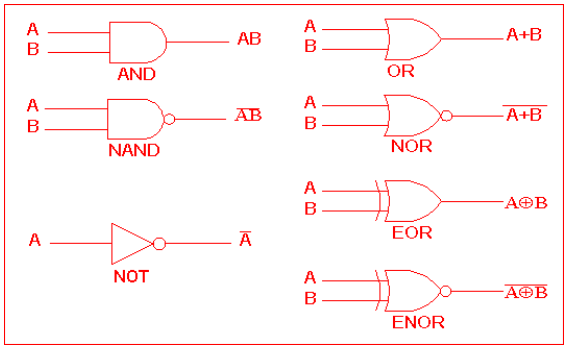


Table 2 is a summary truth table of the input/output combinations for the NOT gate together with all possible input/output combinations for the other gate functions. Also note that a truth table with 'n' inputs has 2^n rows. You can compare the outputs of different gates.

Table 2: Logic gates representation using the Truth table

INPUTS		OUTPUTS					
A	B	AND	NAND	OR	NOR	EXOR	EXNOR
0	0	0	1	0	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	0	1	0	0	1

NOT gate	
A	$\overline{A}$
0	1
1	0

PROCEDURE:

1. Make the connections as per the circuit diagram
2. Switch on the power supply
3. Verify the truth table

PRECAUTIONS:

1. The power supply pins must be checked whether power is available at those pins using test probes.
2. No loose connections should be there and care must be taken to avoid shorting of pins.

REAL TIME APPLICATIONS:

Realization of Boolean expressions using logic gates is used to design digital circuits like adders and multiplexers.

It forms the basic operation of computers, calculators, and microprocessors.

Applications include control systems, communication devices, and digital electronics hardware.

RESULT:

Realize and minimize Boolean functions using basic gates and universal gates (NAND/NOR) in SOP/POS form have been verified.

VIVA QUESTIONS:

1. What is a Boolean function?
2. What is meant by SOP (Sum of Products) form?
3. What is meant by POS (Product of Sums) form?
4. What are minterms and maxterms?
5. Define canonical SOP form.
6. Define canonical POS form.
7. What are basic logic gates?
8. What are universal gates?
9. Why are NAND and NOR called universal gates?
10. What is the difference between canonical and standard forms?
11. How do you convert a truth table into SOP form?
12. How do you convert a truth table into POS form?
13. How do you simplify Boolean expressions using Boolean algebra?
14. What is a Karnaugh Map (K-map)?
15. How is K-map used for minimization of Boolean functions?
16. What are prime implicants and essential prime implicants?
17. How do you realize SOP expression using basic gates?
18. How do you realize POS expression using basic gates?
19. How can a Boolean function be implemented using only NAND gates?
20. How can a Boolean function be implemented using only NOR gates?
21. Convert a given SOP expression into NAND-only implementation.
22. Convert a given POS expression into NOR-only implementation.
23. Compare SOP and POS implementations in terms of gate count.
24. What are the advantages of using NAND gates over basic gates?
25. How does minimization reduce hardware complexity in digital circuits?

Experiment No. 02

Design and implement Half Adder, Full Adder using logic gates.

AIM:

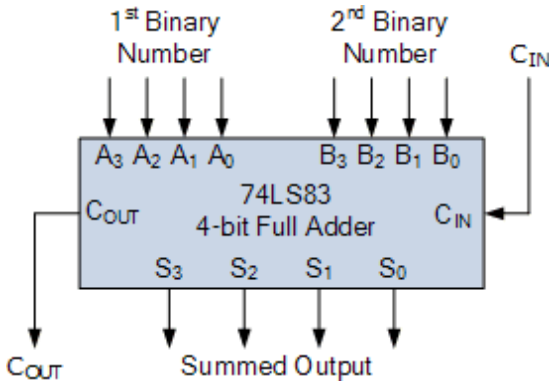
To design and realize Half Adder Full Adder circuits using logic gates and verify the truth tables.

APPARATUS:

1. IC 7408
2. IC 7432
3. IC 7486
4. Bread Board
5. Connecting wires

BLOCK DIAGRAM:

4-BIT ADDER USING 74LS83:



LOGIC DIAGRAM OF HALF ADDER

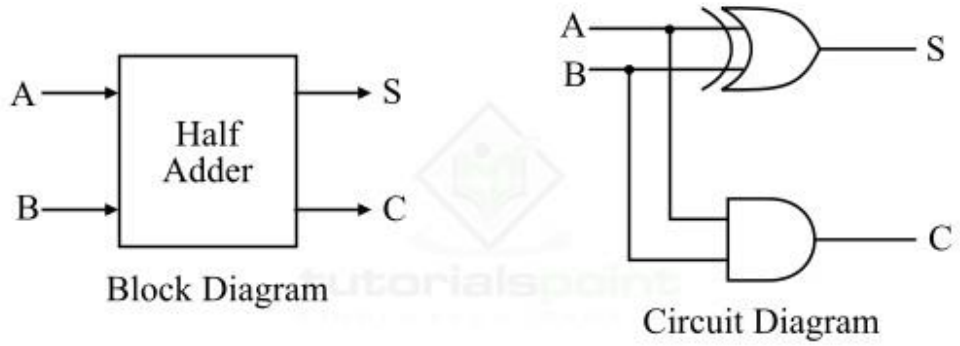
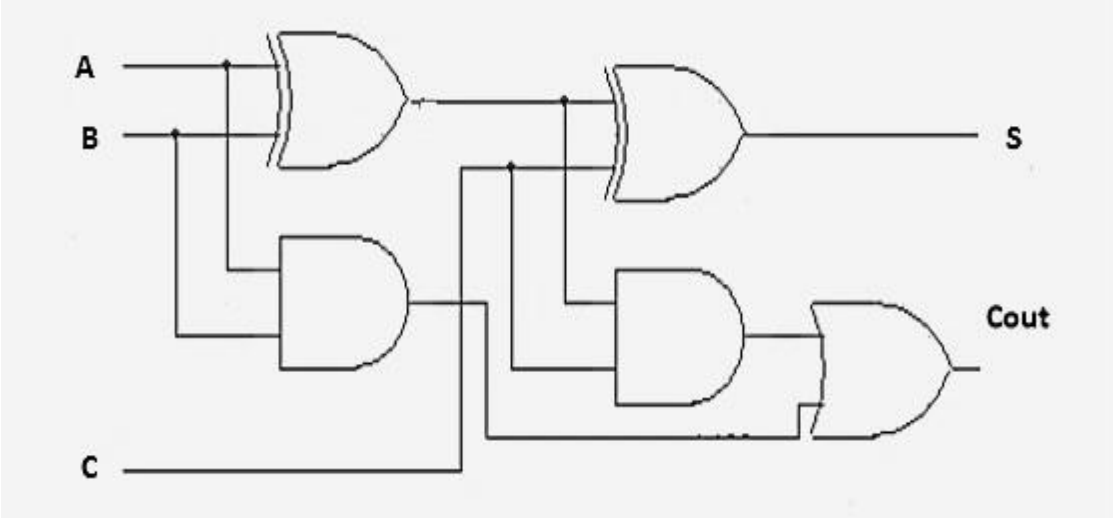


Figure 1 - Half Adder

Truth Table

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

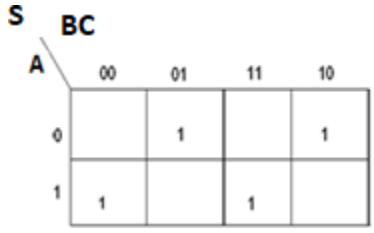
LOGIC DIAGRAM OF FULL ADDER:



TRUTH TABLE OF FULL ADDER:

A	B	C	Cout	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

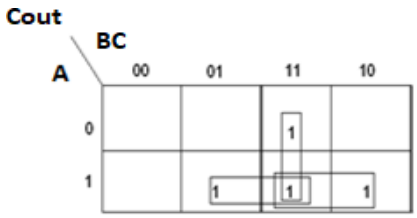
K-Map for SUM output:



S = A'B'C + A'BC' + AB'C' + ABC

S = A ⊕ B ⊕ C

K-Map for CARRY output:



Cout = AB + BC + AC

Cout = (A ⊕ B)C + AB

THEORY:

HALF ADDER:

A combinational logic circuit which is designed to add two binary digits is called as a half adder. The half adder provides the output along with a carry value (if any). The half adder circuit is designed by connecting an EX-OR gate and one AND gate. It has two input terminals and two output terminals for sum and carry. The block diagram and circuit diagram of a half adder are shown in Figure-1.

From the logic circuit diagram of half adder, it is clear that A and B are the two input bits, S is the output sum, and C is the output carry bit.

In the case of a half adder, the output of the EX-OR gate is the sum of two bits and the output of the AND gate is the carry. Although, the carry obtained in one addition will not be forwarded in the next addition because of this it is known as half adder.

FULL ADDER:

A full adder is a combinational circuit that performs the arithmetic sum of three input bits. It consists of three inputs and two outputs. Two of the input variables, denoted by A and B, represent the two significant bits to be added. The third input Cin represents the carry from the previous lower significant position. Two outputs are necessary because the arithmetic sum of three binary digits ranges in value from 0 to 3, and binary 2 or 3 needs two digits. The two outputs are designated by the symbols S for sum and Cout for output carry.

$S = A \square B \square C$

$Cout = A.B + A.C + B.C$

PROCEDURE:

1. Insert the appropriate IC's into the Breadboard.
2. Make connections as shown in the circuit diagram.
3. Connect Pin-14 of all ICs to +5V and Pin-7 to ground.
4. Provide the input data via input switches.
5. Feed the logic 0 (0V) or logic 1 (5V) in different combinations at the inputs according to truth table.
6. Observe the output on output LEDs.
7. Give all possible combinations of inputs and note down the output.
8. Verify the operation of the circuits using truth table.

PRECAUTIONS:

1. The power supply pins must be checked whether power is available at those pins using test probes.
2. No loose connections should be there and care must be taken to avoid shorting of pins.

RESULT: Thus the Half Adder and Full Adder circuits are designed and the truth tables are verified.

VIVA QUESTIONS:

1. Implement a Full Adder using only NAND gates.
2. Implement a Half Adder using only NOR gates.
3. What is the difference between combinational and sequential circuits?
4. Is a Half Adder a combinational circuit? Justify.
5. How can propagation delay affect adder performance?
6. What is ripple carry adder and how is it related to Full Adders?
7. How can you minimize the Boolean expressions of a Full Adder?
8. Design a 4-bit adder using Full Adders.
9. What are the limitations of ripple carry adders?
10. How many Half Adders are required to construct a Full Adder?
11. Explain how a Full Adder can be implemented using two Half Adders.
12. What is the role of Carry-in (Cin) in a Full Adder?
13. Compare Half Adder and Full Adder.
14. Why is a Half Adder not sufficient for multi-bit addition?
15. How does carry propagation occur in a Full Adder?
16. What are the inputs and outputs of a Full Adder?
17. Write the truth table of a Half Adder.
18. Write the truth table of a Full Adder.
19. Which logic gates are used to implement a Half Adder?
20. Which logic gates are used to implement a Full Adder?
21. What is the function of the Sum output in an adder?
22. What is the function of the Carry output?
23. What is a Half Adder?
24. What are the inputs and outputs of a Half Adder?
25. What is a Full Adder?

EXPERIMENT: 03

Design and implement Half Subtractor, Full Subtractor using logic gates.

AIM:To design and realize Half Subtractor Full Subtractor circuits using logic gates and verify the truth tables.

APPARATUS REQUIRED:

1. IC 7408
2. IC 7432
3. IC 7486
4. Bread Board
5. Connecting wires

LOGIC DIAGRAM OF HALF SUBTRACTOR:

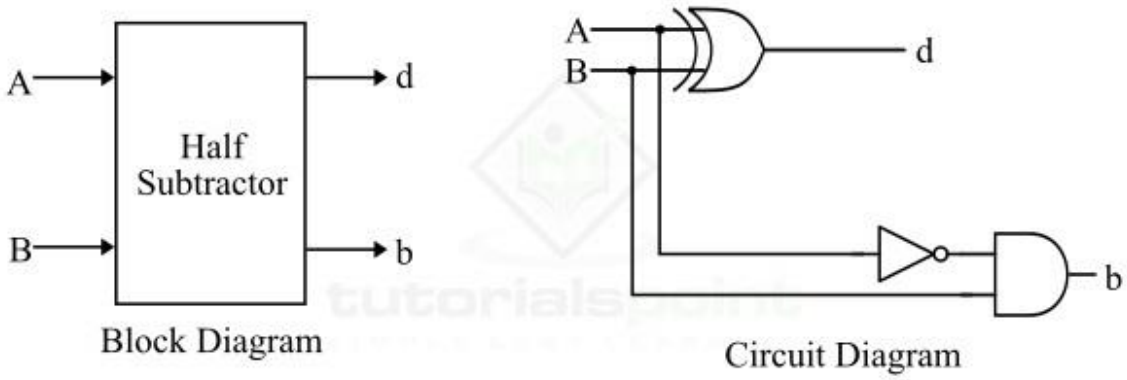


Figure 1 - Half Subtractor

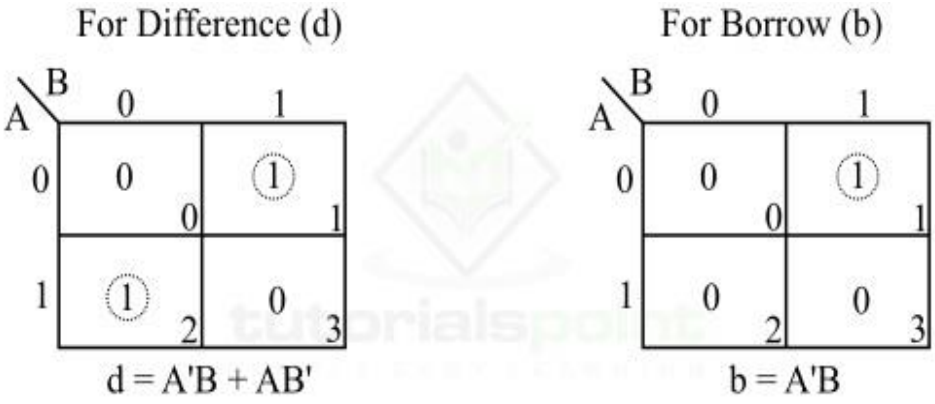
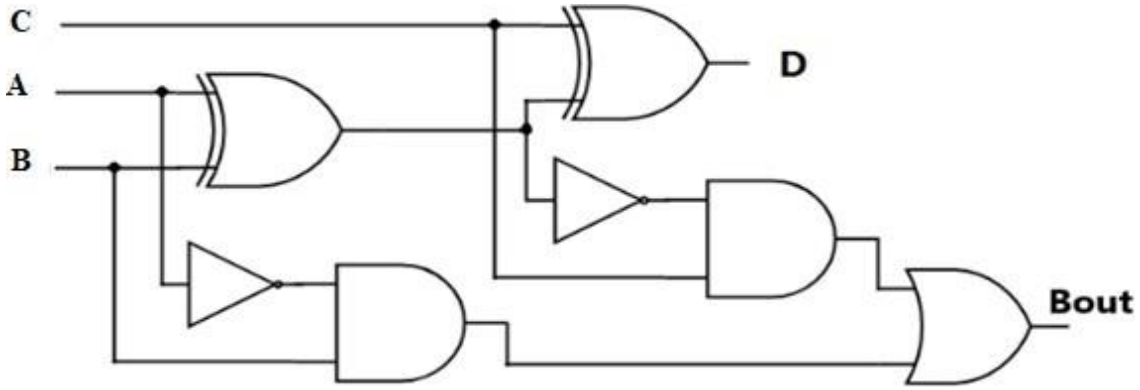


Figure 2 - K Map for Half Subtractor

LOGIC DIAGRAM OF FULL SUBTRACTOR*:*

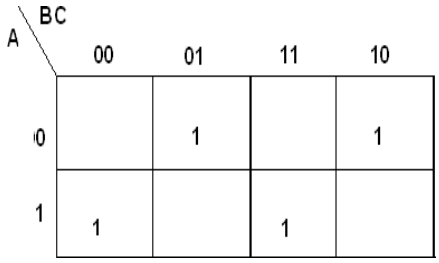


FULL SUBTRACTOR:

TRUTH TABLE:

A	B	C	Bout	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

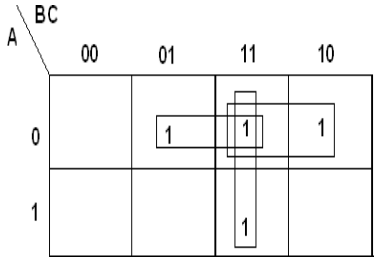
K-Map for Difference output:



D = A'B'C + A'BC' + AB'C' + ABC

D = A ⊕ B ⊕ C

K-Map for Borrow output:



B_{out} = A'B + BC + A'C

B_{out} = (A⊕B)'C + A'B

THEORY:

HALF SUBTRACTOR:

In digital electronics, a subtractor is a combinational logic circuit that can perform the subtraction of two number (binary numbers) and produce the difference between them. It is a combinational circuit that means its output depends on its present inputs only. Although, in practice, the subtraction of two binary number is accomplished by taking the 1's or 2's compliment of the subtrahend and adding it to the minuend.

In this way, the subtraction operation of binary numbers can be converted into simple addition operation which makes hardware construction simple and less expensive. There are two types of subtractors namely, Half Subtractor and Full Subtractor.

In this article, we will discuss the half subtractor, its basic definition, circuit diagram, truth table, characteristic equation, etc. So let's begin with the basic definition of half subtractor.

FULL SUBTRACTOR:

Full subtractor is combinational circuit that performs subtraction of 3-bits. It has 3 inputs & 2 outputs. Here Outputs are known as borrow (Bout) and difference (D).

D = A □ B □ C

Bout = A'.B + BC + A'.C

PROCEDURE:

1. Connections are made as per the circuit diagram
2. Switch on the supply
3. Apply the input values
4. Verify the truth table for different input values
5. Apply the input values
6. Verify the truth table for different input values

PRECAUTIONS:

1. Connection should be tight.
2. O/P should be finding sequentially.
3. IC's should be handled carefully.

RESULT:

Thus the Half Subtractor and Full Subtractor circuits are designed and the truth tables are verified.

VIVA QUESTIONS:

1. What is a Half Subtractor?
2. What are the inputs and outputs of a Half Subtractor?
3. What is a Full Subtractor?
4. What are the inputs and outputs of a Full Subtractor?
5. Define Difference and Borrow in subtraction circuits.
6. Write the truth table of a Half Subtractor.
7. Write the truth table of a Full Subtractor.
8. Which logic gates are used to implement a Half Subtractor?
9. Which logic gates are used to implement a Full Subtractor?
10. What is the purpose of the Borrow output?
11. Derive the Boolean expression for the Difference of a Half Subtractor.
12. Derive the Boolean expression for the Borrow of a Half Subtractor.

13. Derive the Boolean expression for the Difference of a Full Subtractor.
14. Derive the Boolean expression for the Borrow of a Full Subtractor.
15. How many Half Subtractors are required to construct a Full Subtractor?
16. Explain how a Full Subtractor can be implemented using two Half Subtractors.
17. What is the role of Borrow-in (Bin) in a Full Subtractor?
18. Compare Half Subtractor and Full Subtractor.
19. Why is a Half Subtractor not sufficient for multi-bit subtraction?
20. How does borrow propagation occur in a Full Subtractor?
21. Implement a Full Subtractor using only NAND gates.
22. Implement a Half Subtractor using only NOR gates.
23. What is the difference between adder and subtractor circuits?
24. How can a Full Subtractor be implemented using adders (2's complement method)?
25. Design a 4-bit subtractor using Full Subtractors.

EXPERIMENT:04

Construct and analyse basic logic gates (AND, OR, NOT, XOR, XNOR) using only NAND and NOR gates.

AIM: Design and realization logic gates using universal gates

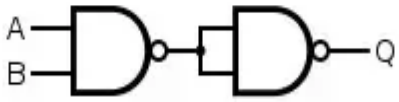
APPARATUS REQUIRED:

IC 7400, IC 7402 Trainer kit, Connecting wires, patch cards.

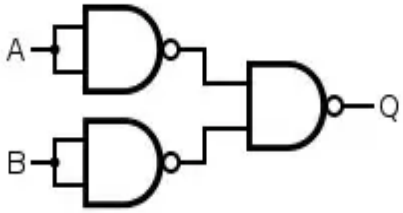
REALIZATION OF BASIC GATE OPERATIONS USING NAND GATES:

NOT

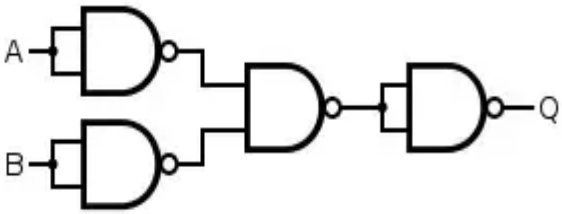
AND



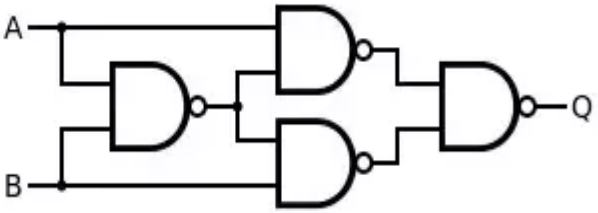
OR



NOR

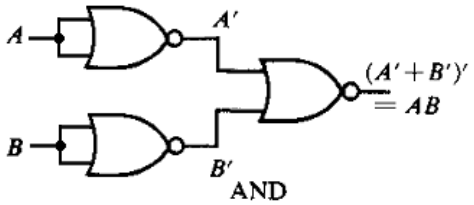


XOR

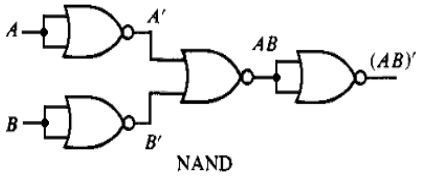


REALIZATION OF BASIC GATE OPERATIONS USING NOR GATE:

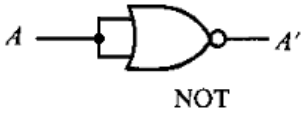
1. **AND Operation**:



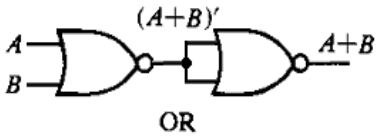
2. **NAND OPERATION:**



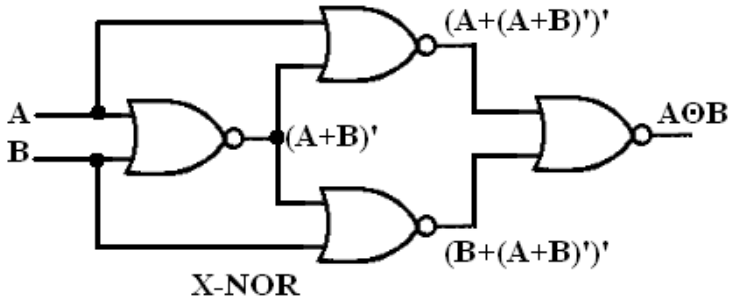
3. **NOT Operation:**

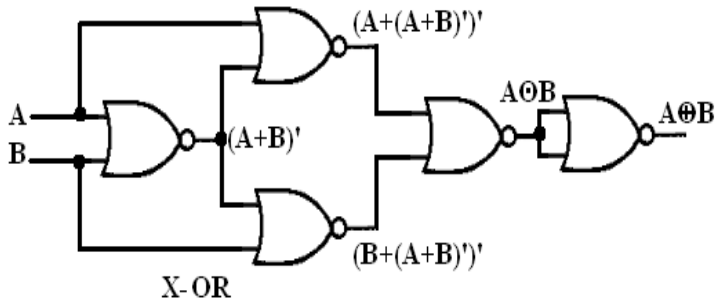


4. **OR Operation:**



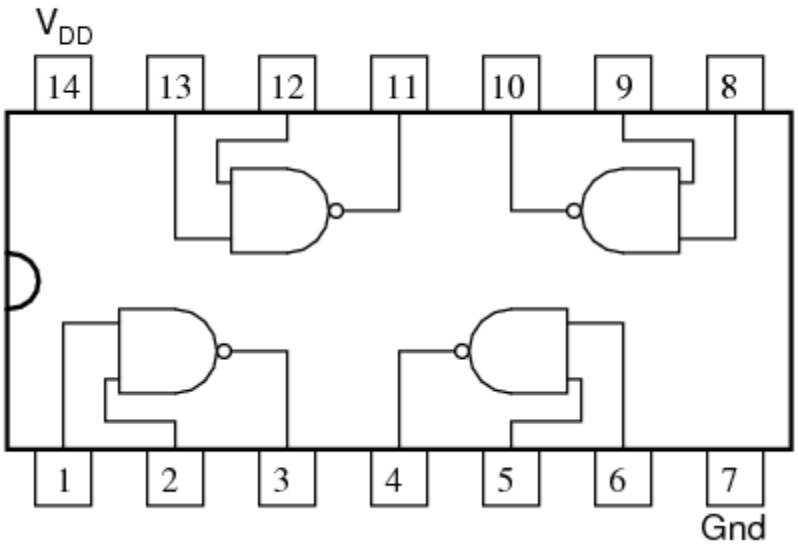
5. **XNOR & XOR Operation**:



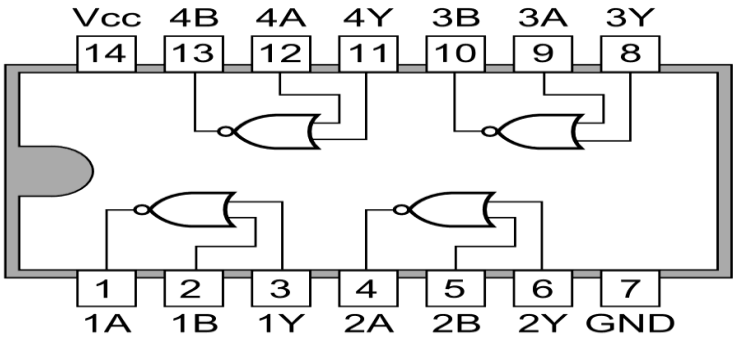


IC PIN DIAGRAMS:

"Pinout," or "connection" diagram for the 4011 quad NAND gate



7402 Quad 2-input NOR Gates



THEORY:

UNIVERSAL GATES

Universal Gates: A universal gate is a gate which can implement any Boolean function without need to use any other gate type. The NAND and NOR gates are universal gates. In practice, this is advantageous since NAND and NOR gates are economical and easier to fabricate and are the basic gates used in all IC digital logic families. In fact, an AND gate is typically implemented as a NAND gate followed by an inverter not the other way around!! Likewise, an OR gate is typically implemented as a NOR gate followed by an inverter.

Truth Table of 2 input Gates

A	B	AND	OR	NAND	NOR	XOR	XNOR

Truth table Single input Gates

A	NOT	BUFFER

PROCEDURE:

1.Supply connections are given at the corresponding pins of ICs.

2.For realization of individual gates using NAND gates alone, the connections are made as per the logic diagrams.

3.Inputs are given and the truth tables of individual gates are verified.

4.The same procedure is repeated for realization of individual gates using NOR gates.

PRECAUTIONS:

1. The power supply pins must be checked whether power is available at those pins using test probes.
2. No loose connections should be there and care must be taken to avoid shorting of pins.

RESULT:

The truth tables of realization of logic gates using universal gates have been verified

VIVA QUESTIONS:

1. How many basic binary subtraction operations are possible?
2. What are the two types of basic adder circuits?
3. A binary parallel adder produces arithmetic sum in what?
4. Total number of inputs in a half adder is
5. Controlled inverter is also known as
6. A logic circuit that provides a HIGH output for both inputs HIGH or both inputs LOW is
7. What is the major difference between half-adders and full-adders?
8. How many basic binary subtraction operations are possible?
9. What are the two types of basic adder circuits?
10. When performing subtraction by addition in the 2's-complement system:
11. What is a 4 bit adder?
12. What is a subtractor?
13. Describe the function of a full adder?
14. Design a full adder using two half adders?
15. What is the difference between half adder and a full adder?
16. Write the applications of adders and subtractors?
17. Draw the logic diagram of adder?
18. Draw the logic diagram of a subtractor?
19. What is the use of adder in digital circuits?
20. Is it possible to construct a circuit to perform both addition and subtraction at the same time?
21. How are adders used in processors? Explain.
22. Carry is obtained in which operation?
23. What are the limitations of half adders?
24. How many logic gates are required for designing a full adder and what are they?
25. How many number of inputs does a half adder and a full adder have?
26. For upto how many bits can a half subtractor perform subtraction?
27. What are the bits which undergo subtraction operation in half subtractors?
28. How many outputs are required for a adder and a subtractor?
29. Describe minuend?
30. Describe subtrahend?

Experiment No. 05

Design and implement code converters such as Binary to Gray, Gray to Binary, using gates

Aim: To Design and realization a 4 bit gray to Binary and Binary to Gray Converter

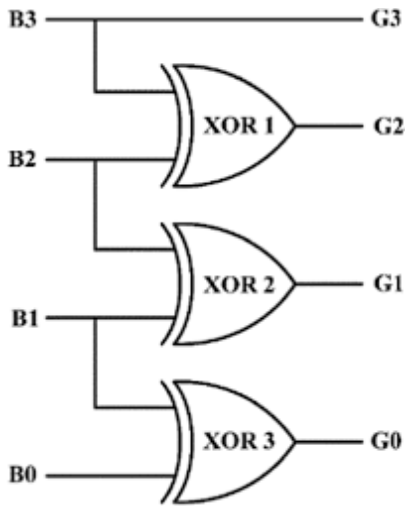
Apparatus: Binary to gray logic trainer kit, Patch cords

Theory:

Binary to Gray Code Converter

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

The 4 bits binary to gray code conversion table & Circuit Diagram is given below



TRUTH TABLE:

4 bit Binary to Gray code converter

Decimal Number	4 bit Binary Number ABCD	4 bit Gray Code $G_1G_2G_3G_4$
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1
10	1 0 1 0	1 1 1 1
11	1 0 1 1	1 1 1 0
12	1 1 0 0	1 0 1 0
13	1 1 0 1	1 0 1 1
14	1 1 1 0	1 0 0 1
15	1 1 1 1	1 0 0 0

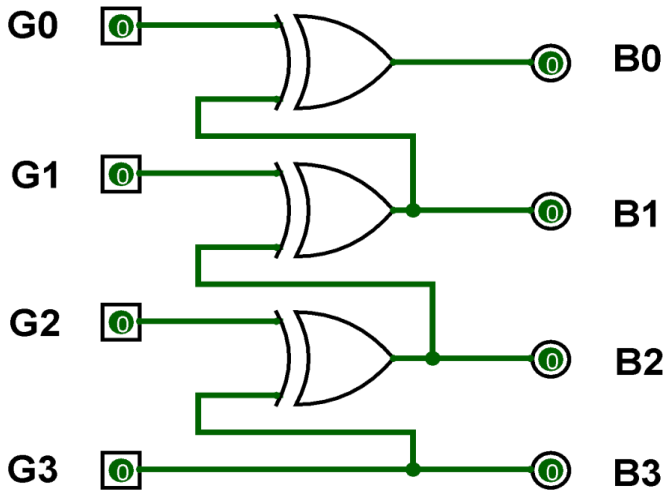
$B_3 = G_3$

$B_2 \oplus B_3 = G_2$

$B_1 \oplus B_2 = G_1$

$B_0 \oplus B_1 = G_0$

The 4 bits Gray to Binary code conversion table & Circuit Diagram is given below



4 bit Gray to Binary code converter

GARY CODE NUMBERS				CONVERTED BINARY NUMBERS			
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

$G_0 \oplus G_1 \oplus G_2 \oplus G_3 = B_0$

$G_1 \oplus G_2 \oplus G_3 = B_1$

$G_2 \oplus G_3 = B_2$

$G_3 = B_3$

Procedure:

1. Connect the circuit as per the circuit diagram.
2. Connect the inputs to the switch and outputs to the logic indicators.
3. Apply the different combinations of inputs and observe the outputs.

PRECAUTIONS:

1. The power supply pins must be checked whether power is available at those pins using test probes.
2. No loose connections should be there and care must be taken to avoid shorting of pins.

Real-Time Applications of Binary to Gray (BTOG) and Gray to Binary (GTOB):

1. **Rotary Position Encoders**

   Gray code is widely used in **rotary encoders** to detect the position of rotating shafts in motors and robotic arms. Binary values are converted to Gray code (BTOG) to reduce errors during position detection, and then converted back to binary (GTOB) for digital processing.

2. **Analog to Digital Converters (ADCs)**
   In some ADC designs, **Gray code is used to minimize switching errors** between adjacent values. The Gray code output is later converted back to binary for computation.
3. **Digital Communication Systems**
   Gray code is used in **modulation techniques** (like QAM) to reduce bit errors during signal transmission. Conversion between binary and Gray code helps improve reliability.
4. **Karnaugh Map (K-Map) Arrangement**
   Gray code is used in arranging **K-maps** so that only one variable changes between adjacent cells, simplifying Boolean function minimization.
5. **Error Reduction in Digital Systems**
   Gray code ensures that **only one bit changes between successive numbers**, reducing errors in high-speed digital circuits.
6. **Robotics and Automation Systems**
   Used in **robot joint position sensing** and automated machines where precise movement tracking is required.
7. **Counters and Digital Instruments**
   Gray code counters are used in **digital measurement systems** to avoid transition errors.
8. **Memory Addressing and Data Conversion**
   Some systems convert **binary addresses to Gray code** to reduce switching noise and then convert back to binary for processing.

RESULT:

Thus, the Binary to Gray code conversion was successfully implemented**,** and the obtained Gray code outputs matched the expected values for the given binary inputs**.**

Thus, the Gray to Binary code conversion was successfully implemented**,** and the obtained binary outputs matched the expected values for the given Gray code inputs

VIVA QUESTIONS:

1. What is a Binary code?
2. What is Gray code?
3. Why is Gray code called reflected binary code?
4. What is the main advantage of Gray code?
5. Where is Gray code commonly used?
6. What is meant by code conversion in digital electronics?
7. What is Binary to Gray code conversion (BTOG)?
8. What is the rule for converting Binary to Gray code?
9. How is the first Gray bit obtained from binary?
10. How are the remaining Gray bits calculated?
11. Which logic gate is used for Binary to Gray conversion?
12. Write the logic expression for Binary to Gray conversion.
13. What is Gray to Binary conversion (GTOB)?
14. What is the rule for converting Gray code to Binary?
15. How is the first binary bit obtained from Gray code?
16. How are the remaining binary bits calculated?
17. Which logic gate is mainly used in Gray to Binary conversion?
18. Why is XOR gate important in code conversion circuits?
19. How many bits change between two successive Gray codes?
20. Why is Gray code preferred in digital systems?
21. What is the difference between Binary code and Gray code?
22. What type of errors can Gray code reduce?
23. Where is Gray code used in encoders?
24. Why is Gray code used in position encoders?
25. What are the applications of code converters?
26. Can combinational circuits be used for code conversion?

27. Draw the logic circuit for Binary to Gray converter.
28. Draw the logic circuit for Gray to Binary converter.
29. What are the advantages of Gray code over binary code?
30. What will happen if more than one bit changes at a time in binary code?

EXPERIMENT: 06

Design and implement simple combinational circuits: 2-to-1 multiplexer, 1-bit comparator

AIM:

Design and implement simple combinational circuits: 2-to-1 multiplexer, 1-bit comparator

APPARATUS:

S.NO.	Component	Type	Quantity
1.	2 to 1 multiplexer	IC 74157	1
2.	1-bit Digital Comparator	IC 7485	1
3.	Digital IC trainer	-	1
4.	patch cards.		

Objective: To verify the Multiplexer operation using IC-74153(2:1) MUX

Circuit Diagram: Figure below shows the block diagram of a Multiplexer.

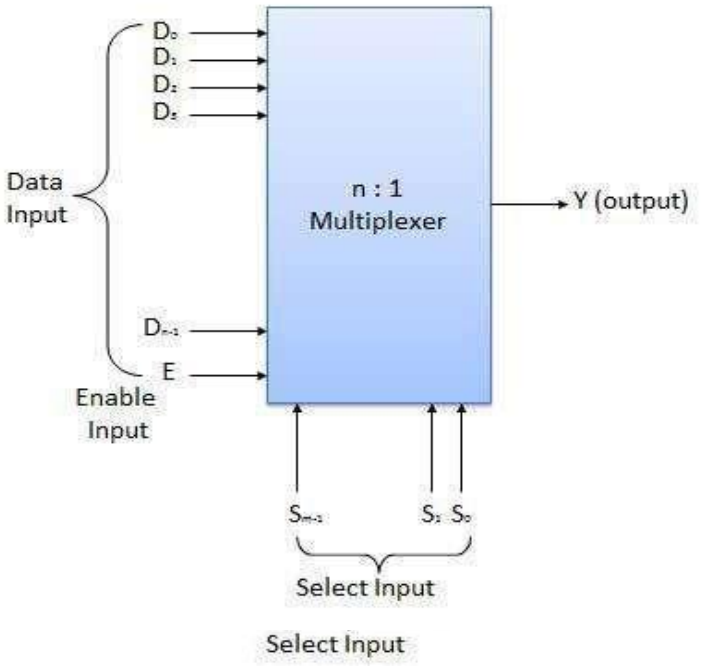
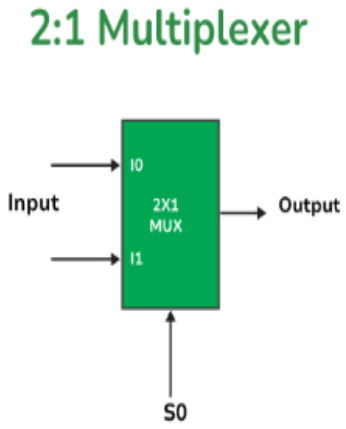


Fig: Block diagram of a Multiplexer

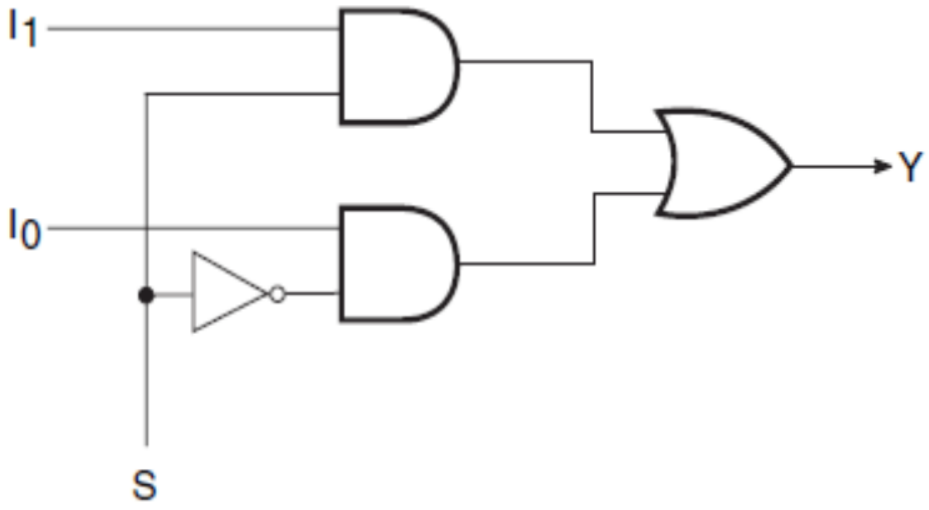
2-to-1 multiplexer Block Diagram



Truth Table

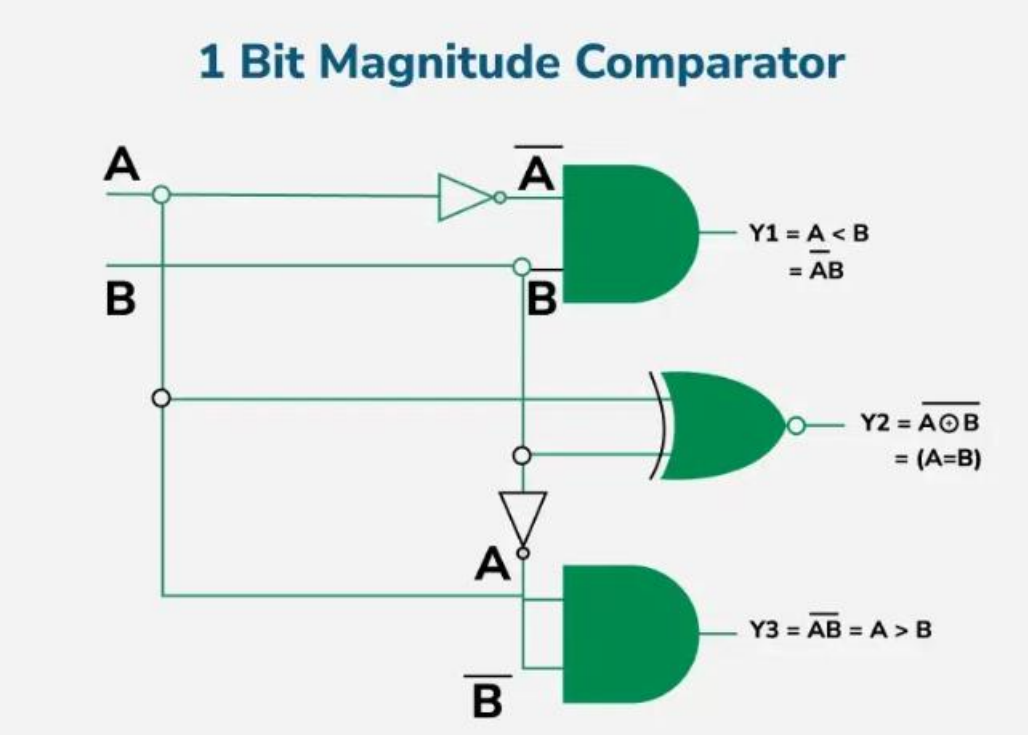
S_0	I_0	I_1	Y
0	0	X	0
0	1	X	1
1	X	0	0
1	X	1	1

Logic Diagram



1-bit comparator Block Diagram

Logic Diagram



Truth Table

A	B	A<B	A=B	A>B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

Figure-2: Truth Table of 1-Bit Comparator

Procedure

1.Assemble the circuit on bread board, as per above Pin diagram.
2.Give the logical inputs and check for the proper output, as per the function table.

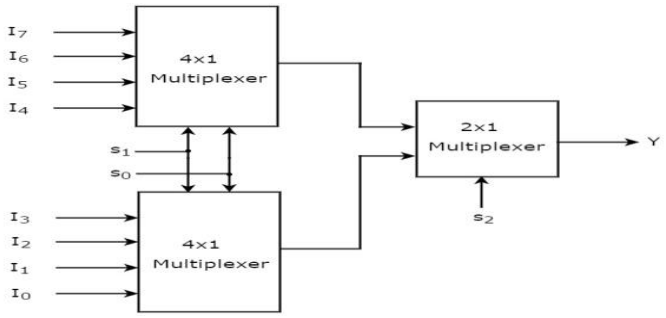
THEORY:

In this section, let us implement 8x1 Multiplexer using 4x1 Multiplexers and 2x1 Multiplexer. We know that 4x1 Multiplexer has 4 data inputs, 2 selection lines and one output. Whereas, 8x1 Multiplexer has 8 data inputs, 3 selection lines and one output.

So, we require two 4x1 Multiplexers in first stage in order to get the 8 data inputs. Since, each 4x1 Multiplexer produces one output, we require a 2x1 Multiplexer in second stage by considering the outputs of first stage as inputs and to produce the final output.

Let the 8x1 Multiplexer has eight data inputs I7 to I0, three selection lines s2, s1 & s0 and one output Y. The Truth table of 8x1 Multiplexer is shown below.

We can implement 8x1 Multiplexer using lower order Multiplexers easily by considering the above Truth table. The block diagram of 8x1 Multiplexer is shown in the following figure.



The same selection lines, s1 & s0 are applied to both 4x1 Multiplexers. The data inputs of upper 4x1 Multiplexer are I7 to I4 and the data inputs of lower 4x1 Multiplexer are I3 to I0. Therefore, each 4x1 Multiplexer produces an output based on the values of selection lines, s1 & s0.

The outputs of first stage 4x1 Multiplexers are applied as inputs of 2x1 Multiplexer that is present in second stage. The other selection line, s2 is applied to 2x1 Multiplexer.

If s2 is zero, then the output of 2x1 Multiplexer will be one of the 4 inputs I3 to I0 based on the

values of selection lines s1 & s0.

If s2 is one, then the output of 2x1 Multiplexer will be one of the 4 inputs I7 to I4 based on the values of selection lines s1 & s0.

Therefore, the overall combination of two 4x1 Multiplexers and one 2x1 Multiplexer performs as one 8x1 Multiplexer.

PRECAUTIONS:

The power supply pins must be checked whether power is available at those pins using test probes.

No loose connections should be there and care must be taken to avoid shorting of pins.

RESULT:

Implementation of simple combinational circuits: 2-to-1 multiplexer, 1-bit comparator are sucessfully verified with the help of truth table

VIVA QUESTIONS

1. What is a multiplexer?
2. Which combinational circuit is renowned for selecting a single input from multiple inputs & directing the binary information to output line?
3. Which is the major functioning responsibility of the multiplexing combinational circuit?
4. How many select lines would be required for an 8-line-to-1-line multiplexer?
5. How many NOT gates are required for the construction of a 4-to-1 multiplexer?
6. What are the applications of multiplexer and de-multiplexer?
7. In 2n 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
10. What is the difference between multiplexer & demultiplexer?
11. How many data inputs does a mux have?
12. What are the components of a mux?
13. What is a combinational circuit?

14. Hat is the ic used in the design of 16*1 mux using two 8*1 mux?
15. What is an integrated circuit?
16. Draw the pin configuration of IC74151
17. How many selectionlines are required for 8*1 mux?
18. What is Magnitude Comparator?
19. List out the applications of comparators?
20. What is digital comparator?
21. Realize a single bit comparator?
22. Which logic gate is a basic comparator?
23. When are two numbers are equal?
24. What are the different types of comparator?
25. What are the advantages of comparator?

PART-A

Realization in Hardware Laboratory (Using Logic ICs)

VIRTUAL LAB :01

1.Realize and minimize Boolean functions using basic gates and universal gates (NAND/NOR) in SOP/POS form.

Aim of the experiment:

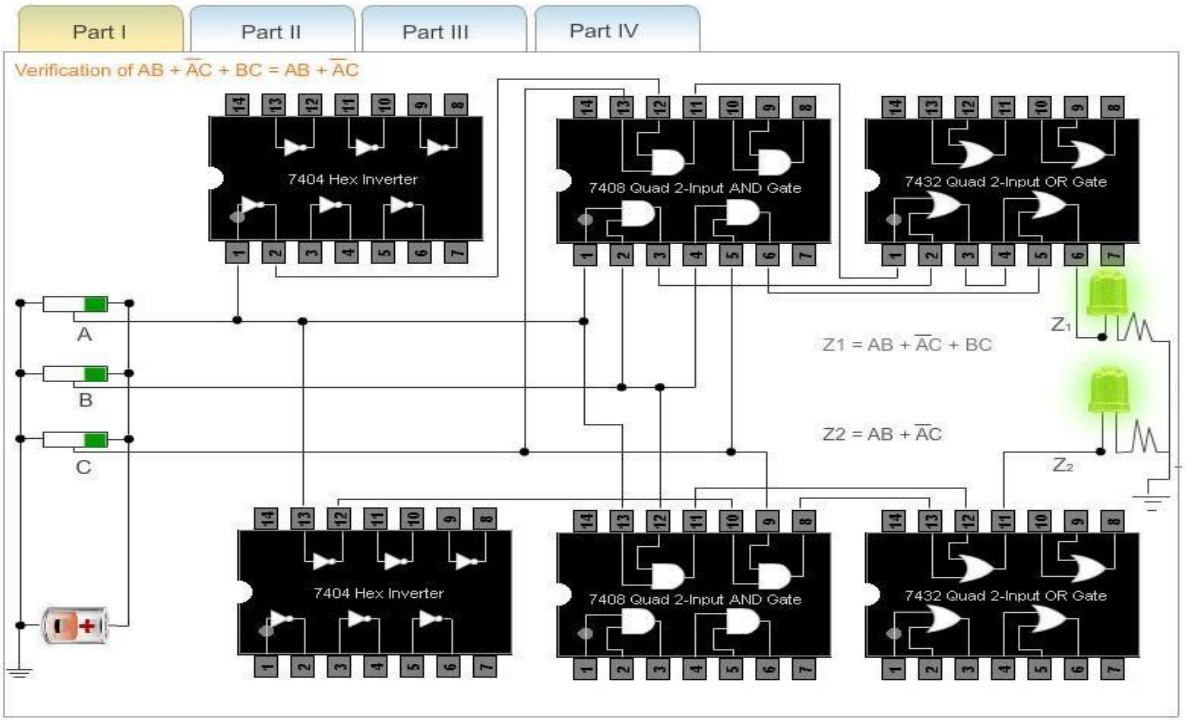
The principal objective of this experiment is to fully understand the function and use of logic gates such as 7400(quad 2-input NAND gates),7402(quad 2-input NOR gates),7404(Hex inverter) 7408(quad 2-input AND gates) and 7432(quad 2-input OR gates).

Procedure:

Please follow these steps to do the experiment.

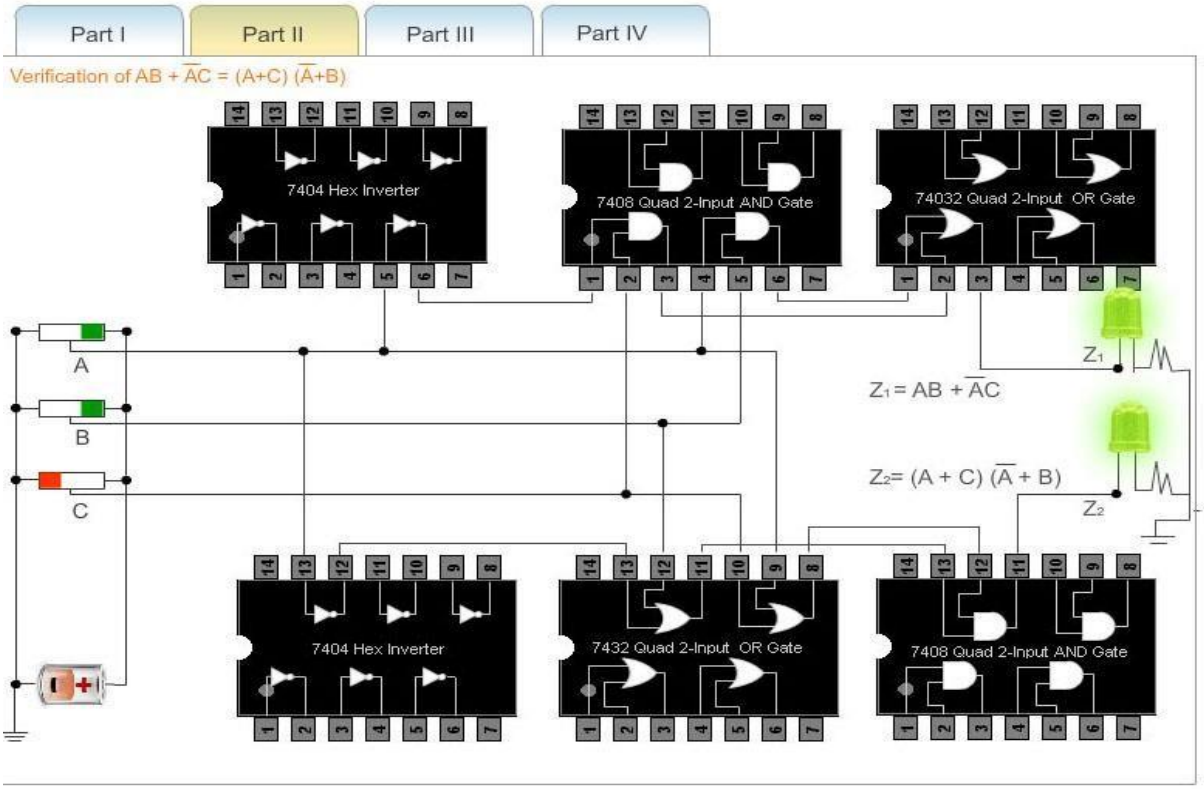
Part 1:

1. At first go through the structure of 7404 Hex inverter, 7408(quad 2-input AND gates), 7432(quad 2-input OR gates).
2. Next, apply a high level voltage to all the inputs A,B,C.
3. Next, check that both LEDs glow. This is because both the outputs z1 and z2 attain the same value.
4. Thus, AB+AC+BC=AB+AC holds for the condition A=B=C="1".
5. For all the combinations of the variables A,B, and C verify that both the LEDs are glowing or not glowing. If the LED glows, it indicates that the corresponding output has reached logic 1 level. Similarly a dark LED indicates low level output voltage.


Part I
Part II
Part III
Part IV
Verification of AB + $\overline{A}C$ + BC = AB + $\overline{A}C$
7404 Hex Inverter
7408 Quad 2-Input AND Gate
7432 Quad 2-Input OR Gate
A
B
C
Z1 = AB + $\overline{A}C$ + BC
Z2 = AB + $\overline{A}C$
Z_1
Z_2
7404 Hex Inverter
7408 Quad 2-Input AND Gate
7432 Quad 2-Input OR Gate

Part 2:

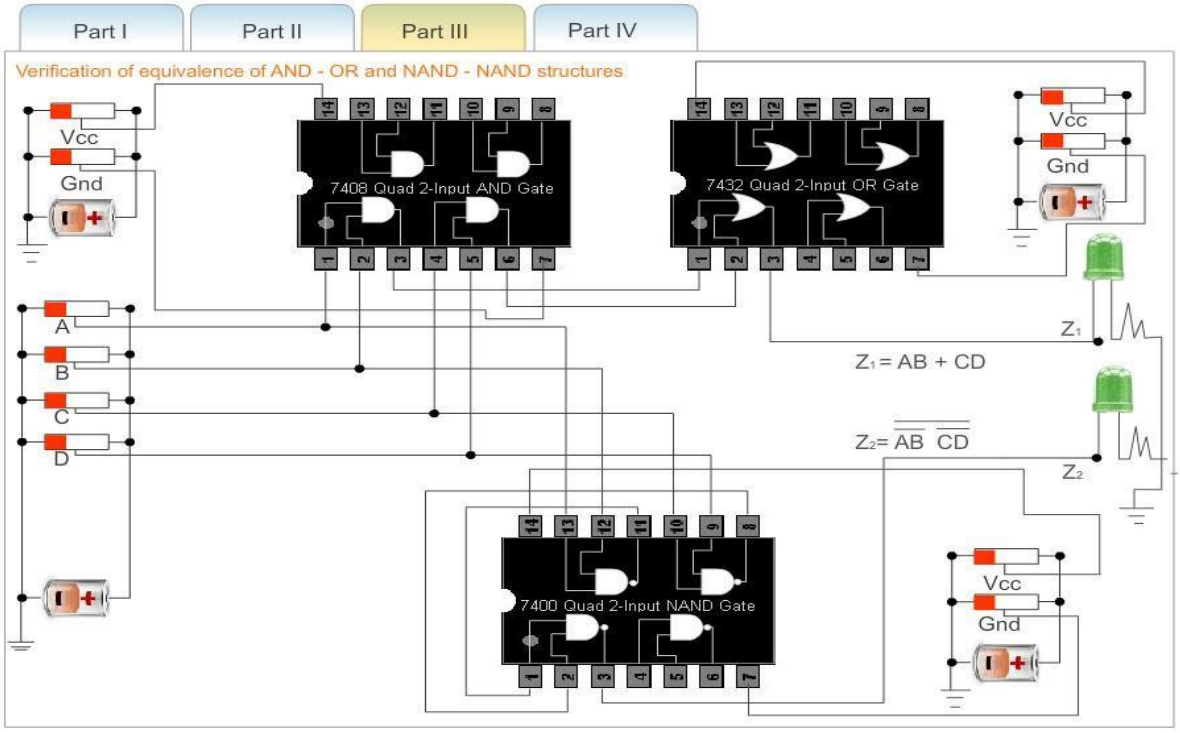
1. At first go through the structure of 7400(quad 2-input NAND gates), 7408(quad 2-input AND gates), 7432(quad 2-input OR gates).
2. Next, apply a high level voltage to inputs A,B.and apply low level voltage to the input C.
3. Next, check that both LEDs glow. This is because both the outputs z1 and z2 attain the same value.
4. So, the equivalence of AND-OR and NAND NAND structure can be verified.
5. For all the combinations of the variables A,B and C verify that both the LEDs are glowing or not glowing. If the LED glows, it indicates that the corresponding output has reached logic 1 level. Similarly a dark LED indicates low level. output voltage.

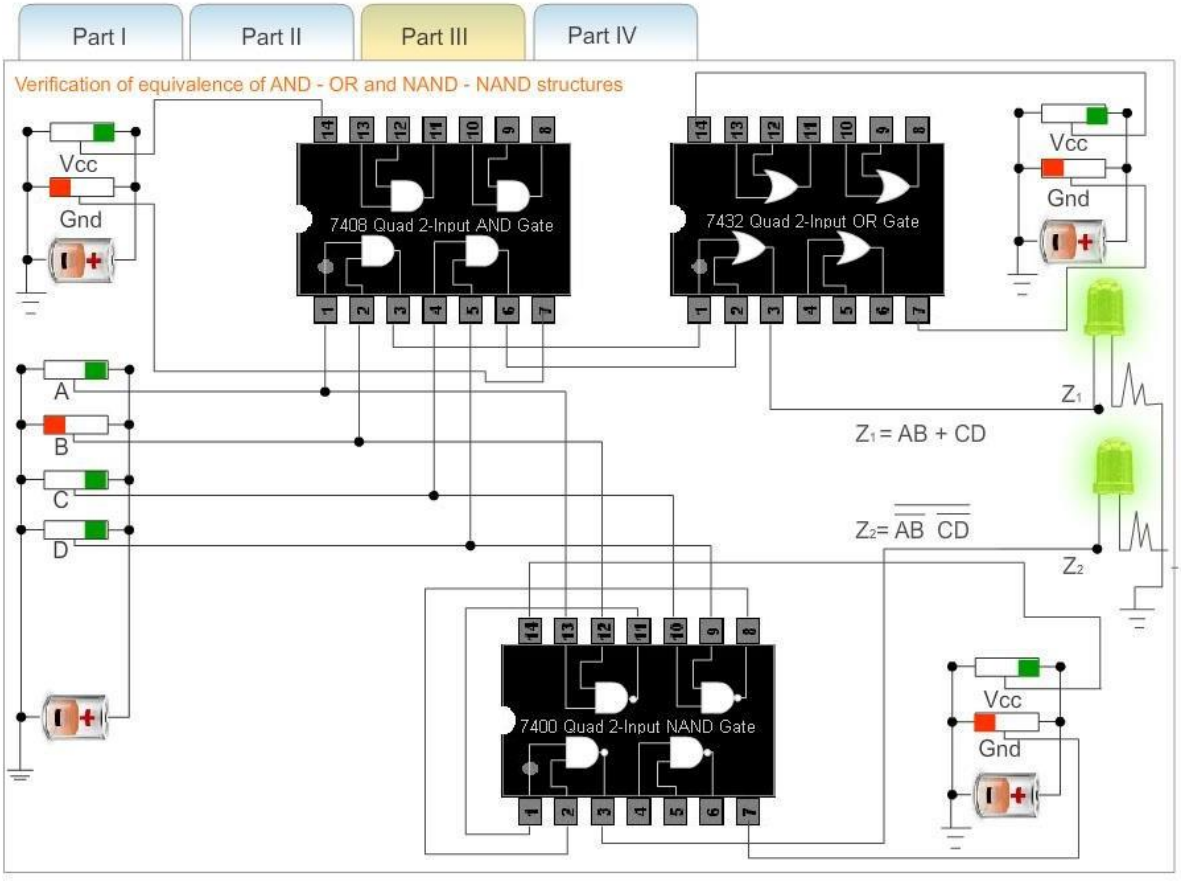


Part 3:

1. At first go through the structure of 7404 Hex inverter, 7408(quad 2-input AND gates), 7432(quad 2-input OR gates), 7400(quad 2-input NAND gates).
2. Next, apply a high level voltage to all the Vcc inputs and keep low level voltage to all the Gndinputs.IfVcc and ground are not connected properly then error message will be appeared and no output will be generated.
3. Next, apply a high level voltage to the inputs A,B and apply low level voltage to the input C.
4. Next, check that both LEDs glow.This is because both the outputs z1 and z2 attain the same value.
5. Thus, AB+AC+BC=AB+AC =(A+C)(A+B) holds for the condition A=B="1" and C=D="0"

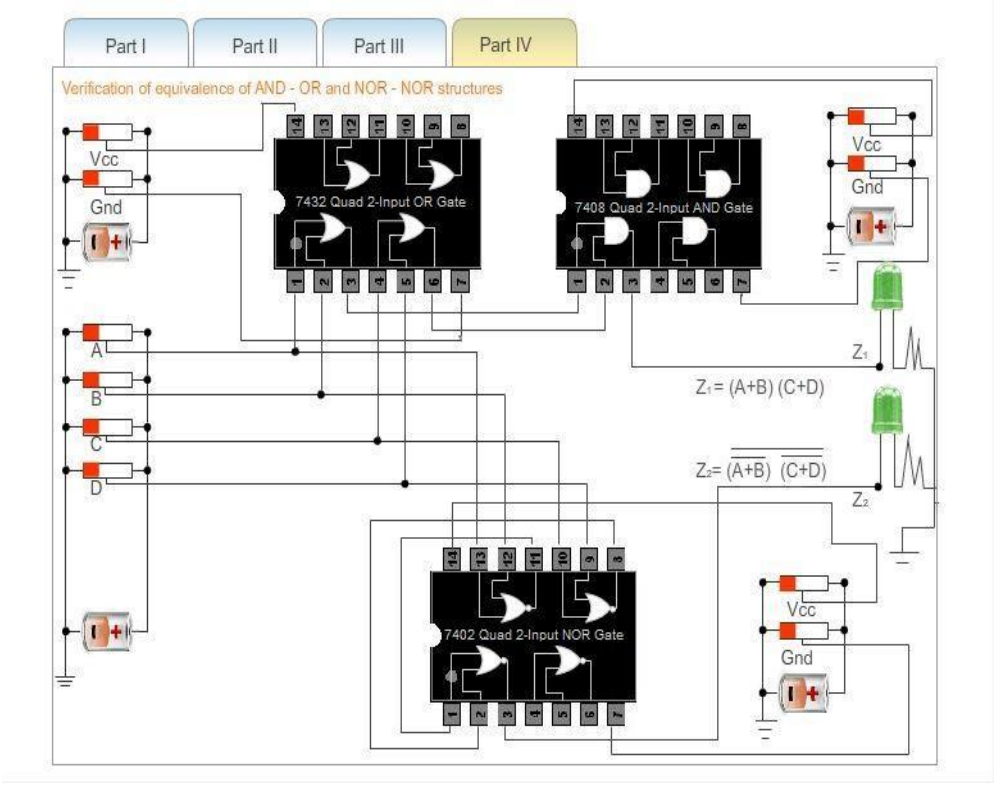
6. For all the combinations of the variables A,B and C verify that both the LEDs are glowing or not glowing. If the LED glows, it indicates that the corresponding output has reached logic 1 level. Similarly a dark LED indicates low level output voltage

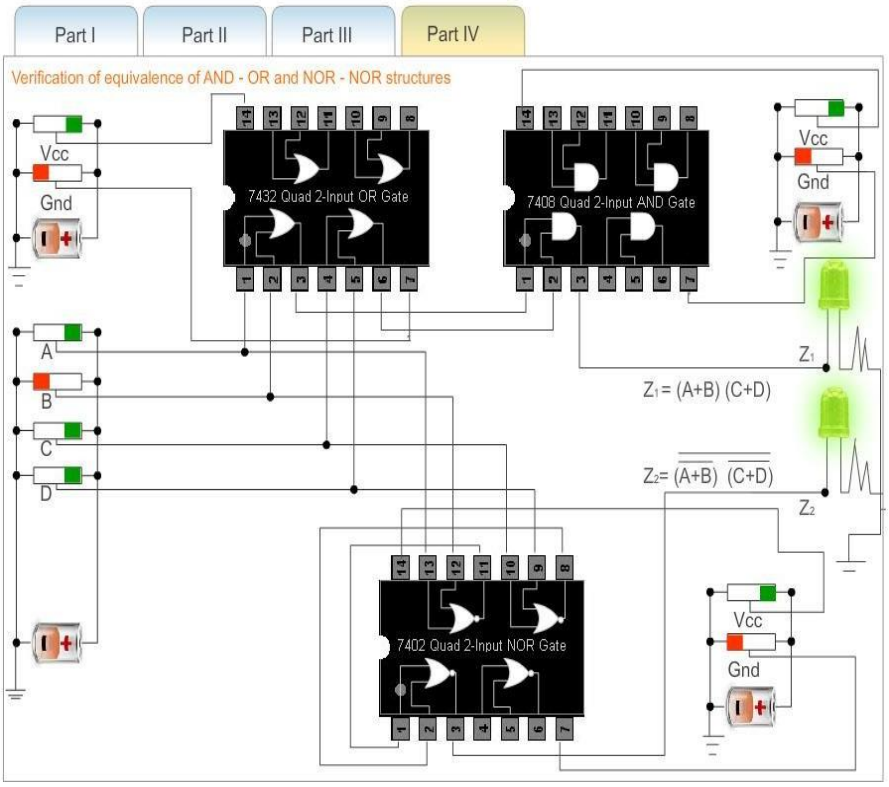




Part 4:

1. At first go through the structure of 7402(quad 2-input NOR gates), 7408(quad 2-input AND gates), 7432(quad 2-input OR gates).
2. Next, apply a high level voltage to all the Vcc inputs and apply low level voltage to all the Gndinputs.IfVcc and ground are not connected properly then error message will be appeared and no output will be generated.
3. Next, apply a high level voltage to all the inputs A,C and apply low level voltage to the inputs B,D.
4. Next, check that both LEDs glow.This is because both the outputs z1 and z2 attain the same value.
5. So, the equivalence of OR-AND and NOR-NOR structure can be verified.
6. For all the combinations of the variables A,B,C and D verify that both the LEDs are glowing or not glowing. If the LED glows, it indicates that the corresponding output has reached logic 1 level. Similarly a dark LED indicates low level output voltage.





Url: **https://dec-iitkgp.vlabs.ac.in/exp/basic-logic-gates/**

Conclusion:

The experiment demonstrates that Boolean expressions can be effectively analyzed and simplified using Boolean algebra and implemented using basic logic gates.

Thus, the required logical function is successfully synthesized, confirming the correctness of the Boolean expression and gate-level realization.

VIRTUAL LAB :02

Design and implement Half Adder, Full Adder using logic gates

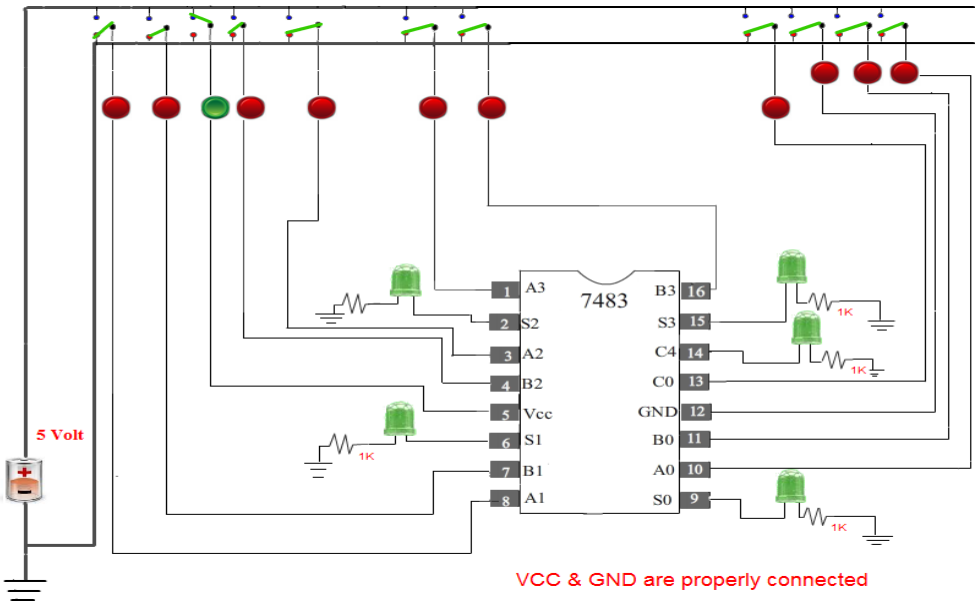
AIM: Design and implement Half Adder, Full Adder using logic gates

- The objective of part 1 of the experiment is to fully understand the functionality of 4 bit binary full adder.
- The objective of part 2 of the experiment is to fully understand the functionality of 8 bit full adder by cascading to 7483 chips(two 4 bit full adder).
- The objective of part 3 of the experiment is to fully understand the functiponality and implementation of 4 bit adder/subtractor

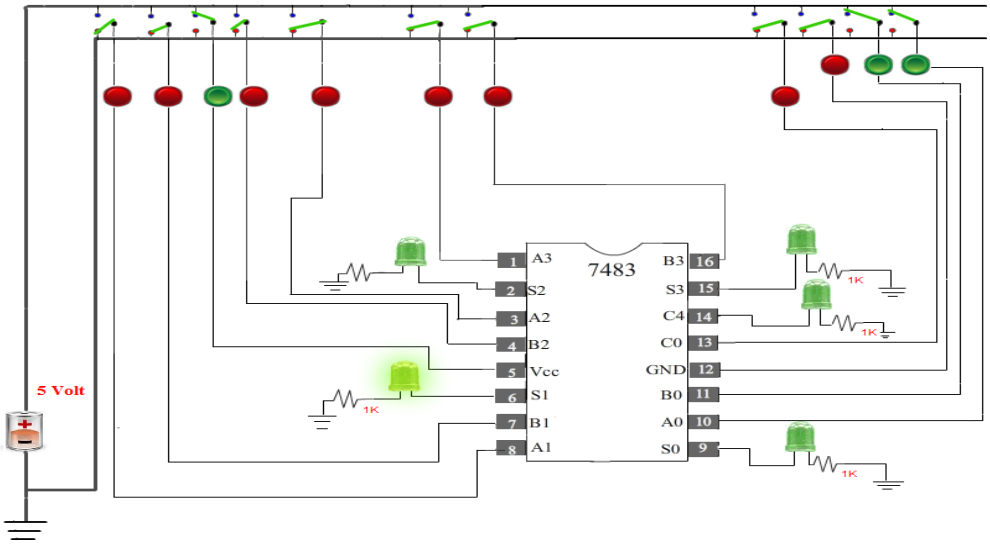
Procedure:

Please follow these steps to do the experiment:

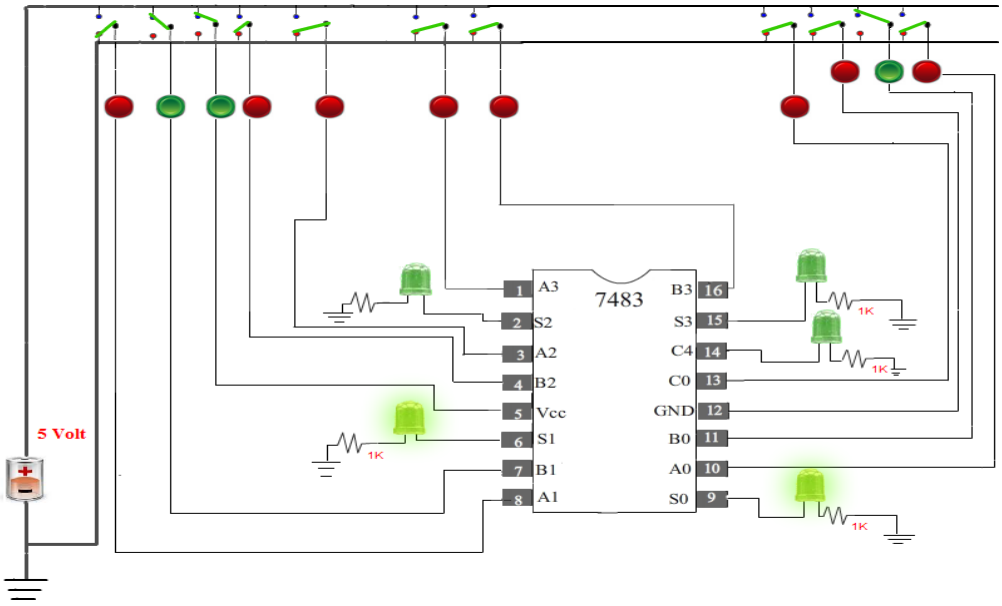
1. At first click on the Vcc switch that means Vcc = 1 and GND = 0 , show message Vcc& GND properly connected.



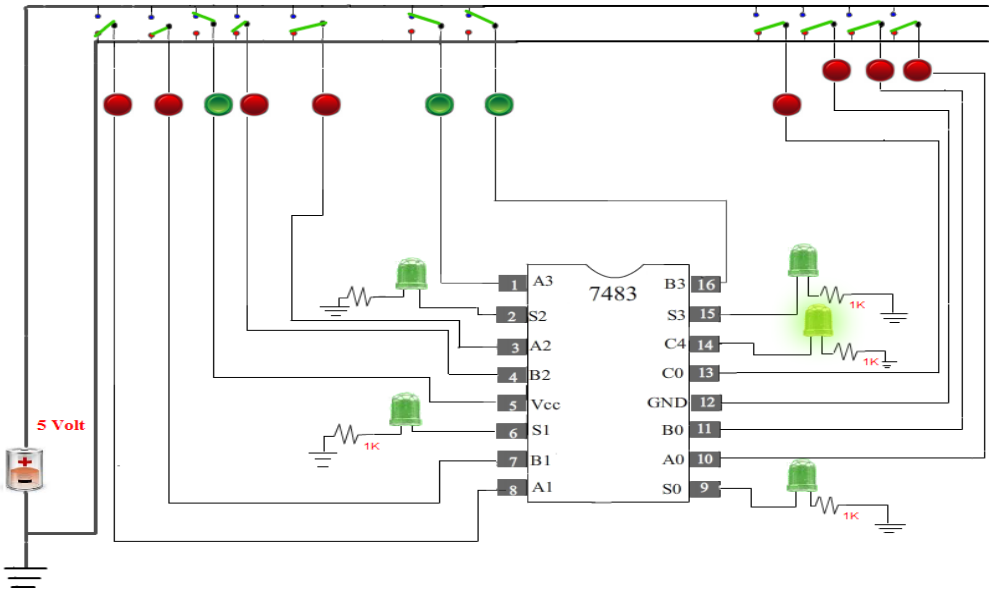
2. Next, A_0= 1, A_1= 0, A_2= 0, A_3= 0 and B_0= 1, B_1= 0, B_2= 0, B_3= 0 now you can see the output result of S_0= 0, S_1= 1, S_2= 0, S_3= 0 and C_4= 0.



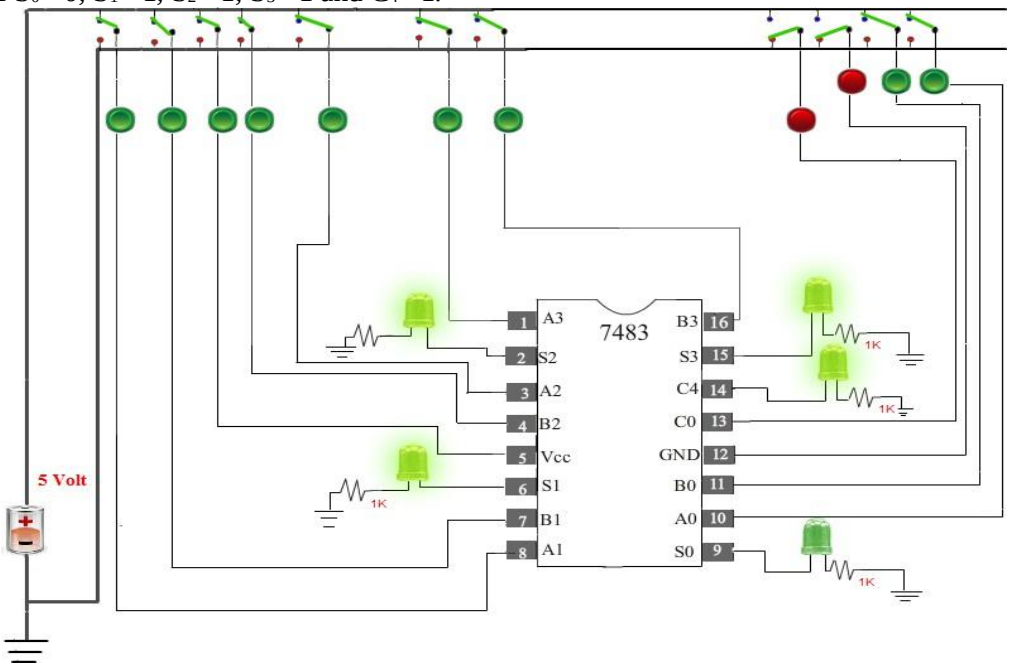
3. Next, A_0= 0, A_1= 0, A_2= 0, A_3= 0 and B_0= 1, B_1= 1, B_2= 0, B_3= 0 now you can see the output result of S_0= 1, S_1= 1, S_2= 0, S_3= 0 and C_4= 0.



4. Next, A_0= 0, A_1= 0, A_2= 0, A_3= 1 and B_0= 0, B_1= 0, B_2= 0, B_3= 1 now you can see the output result of S_0= 0, S_1= 0, S_2= 0, S_3= 0 and C_4= 1.



5. Next, A_0= 1, A_1= 1, A_2= 1, A_3= 1 and B_0= 1, B_1= 1, B_2= 1, B_3= 1 now you can see the output result of S_0= 0, S_1= 1, S_2= 1, S_3= 1 and C_4= 1.



Url: https://dec-iitkgp.vlabs.ac.in/exp/arithmetic-expressions/

Conclusion:

The experiment confirms that arithmetic expressions can be correctly analyzed and implemented using adders and subtractors. The designed circuits perform accurate addition and subtraction operations, validating the logical design and functionality of arithmetic circuits

1)HALF ADDER

Simulator 1:

Step-1) Connect the supply(+5V)  to the circuit.

Step-2) First press "ADD" button to add basic state of your output in the given table.

Step-3) Press the switches to select the required inputs "A" and "B".

Step-4) Press "ADD" button to add your inputs and outputs in the given table.

Step-5) Repeat steps 3 & 4 for next state of inputs and their corresponding outputs.

Step-6) Press the "PRINT" button after completing your simulation to get your results.

Step-7) Press the "RESET" button whenever you want to refresh your simulator.

Simulator 2:

Step-1) Enter the Boolean input "A" and "B".

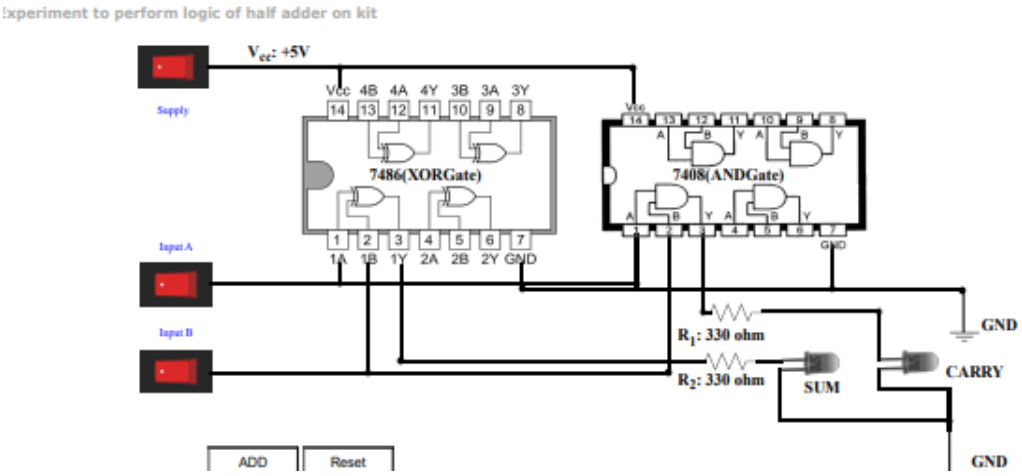
Step-2) Enter the Boolean output for your corresponding inputs.

Step-3) Click on "Circuit" button to check the circuit diagram for half adder.

Step-4) Click on "Check" Button to verify your output.

Step-5) Click "Print" if you want to get print out of Truth Table.

Step-6) Click on "Reset" button if you want to reset input and outputs.



2)FULL ADDER

Simulator 1:

Step-1) Connect the supply(+5V)  to the circuit.

Step-2) First press "ADD" button to add basic state of your output in the given table.

Step-3) Press the switches to select the required inputs "A" and "B" and "C_{in}".

Step-4) Press "ADD" button to add your inputs and outputs in the given table.

Step-5) Repeat steps 3 & 4 for next state of inputs and their corresponding outputs.

Step-6) Press the "PRINT" button after completing your simulation to get your results.

Step-7) Press the "RESET" button whenever you want to refresh your simulator.

Simulator 2:

Step-1) Enter the Boolean input "A" and "B" and "Cin".

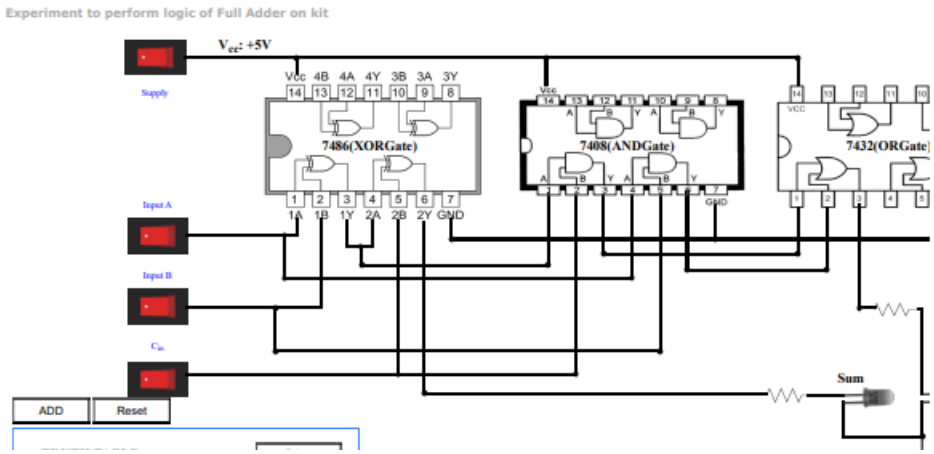
Step-2) Enter the Boolean output for your corresponding inputs.

Step-3) Click on "Circuit" button to check the circuit diagram for full adder.

Step-4) Click on "Check" Button to verify your output.

Step-5) Click "Print" if you want to get print out of Truth Table.

Step-6) Click on "Reset" button if you want to reset input and outputs.



Conclusion:

The experiment demonstrates that Boolean expressions can be effectively analyzed and simplified using implement Half Adder, Full Adder and implemented using basic logic gates.

Thus, the required logical function is successfully synthesized, confirming the correctness of the Boolean expression and gate-level realization.

VIRTUAL LAB :03

Design and implement Half Subtractor, and Full Subtractor using logic gates

AIM: To verify the truth table of half subtractor by using the ICs of XOR, NOT and AND gates and of full subtractor by using the ICs of XOR, AND, NOT and OR gates respectively and analyse the working of half subtractor and full subtractor circuit with the help of LEDs in simulator 1 and verify the truth table only of half subtractor and full subtractor in simulator 2.

PROCEDURE:

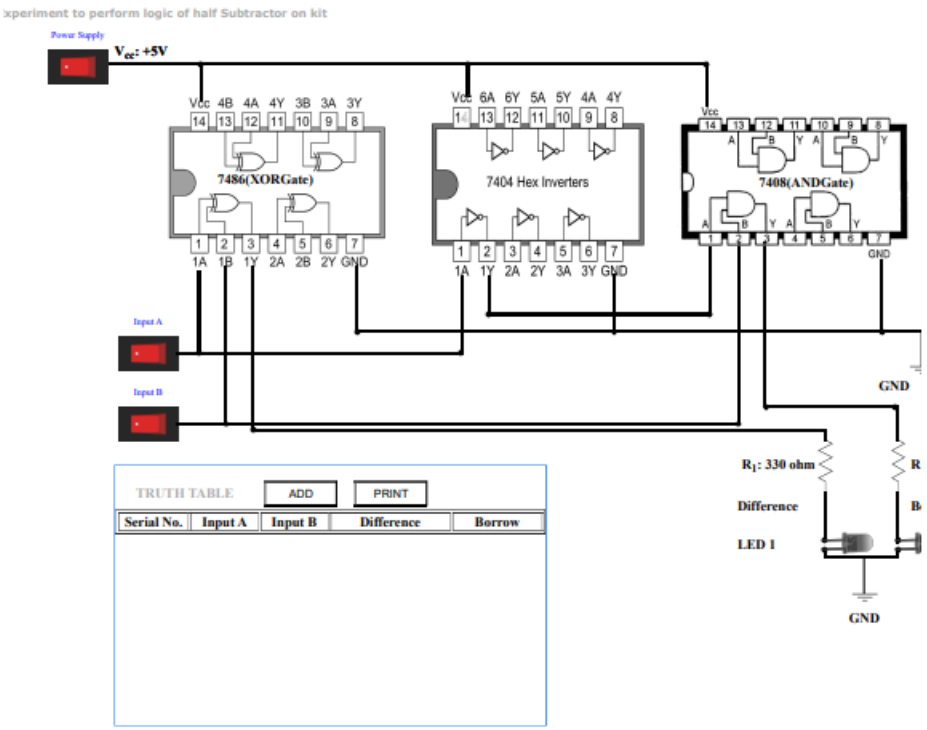
1)HALF SUBTRACTOR

Simulator 1:

Step-1) Connect the Supply(+5V)  to the circuit.
Step-2) First press "ADD" button to add basic state of your output in the given table.
Step-3) Press the switches to select the required inputs "A" and "B".
Step-4) Press "ADD" button to add your inputs and outputs in the given table.
Step-5) Repeat steps 3&4 for next state of inputs and their corresponding outputs.
Step-6) Press the "PRINT" button after completing your simulation to get your results.

Simulator 2:

Step-1) Enter the Boolean input "A" and "B".
Step-2) Enter the Boolean output for your corresponding inputs.
Step-3) Click on "Check" Button to verify your output.
Step-4) Click "Print" if you want to get print out of Truth Table.
Step-5) Click "Reset" if you want to reset inputs and outputs.



2)**FULL SUBTRACTOR**

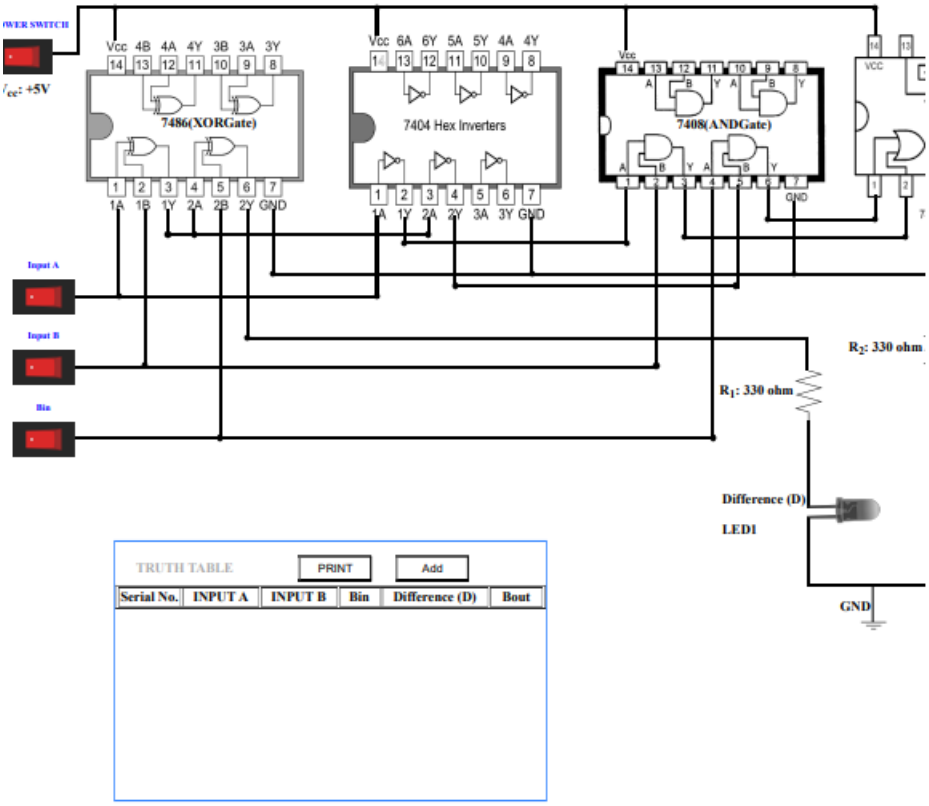
Simulator 1:

Step-1) Connect the Supply(+5V)  to the circuit.
Step-2) First press "ADD" button to add basic state of your output in the given table.
Step-3) Press the switches to select the required inputs "A" and "B" and "Bin".
Step-4) Press "ADD" button to add your inputs and outputs in the given table.
Step-5) Repeat steps 3&4 for next state of inputs and their corresponding outputs.
Step-6) Press the "PRINT" button after completing your simulation to get your results.

Simulator 2:

Step-1) Enter the Boolean inputs "A" and "B" and "Bin".
Step-2) Enter the Boolean output for your corresponding inputs.
Step-3) Click on "Check" Button to verify your output.
Step-4) Click "Print" if you want to get print out of Truth Table.
Step-5) Click "Reset" if you want to reset inputs and outputs.

Experiment to perform logic of Full Subtractor on kit



TRUTH TABLE PRINT Add

Serial No.	INPUT A	INPUT B	Bin	Difference (D)	Bout

Conclusion:

The experiment demonstrates that Boolean expressions can be effectively analyzed and simplified using implement Half Subtractor, and Full Subtractor and implemented using basic logic gates.

Thus, the required logical function is successfully synthesized, confirming the correctness of the Boolean expression and gate-level realization.

VIRTUAL LAB :04

Construct and analyse basic logic gates (AND, OR, NOT, XOR, XNOR) using only NAND and NOR gates.

Aim: To implement the logic functions i.e. AND, OR, NOT, Ex-OR, Ex- NOR and a logical expression with the help of NAND and NOR universal gates respectively.

Procedure:

1. Click on the **Component** button to place components on the table.

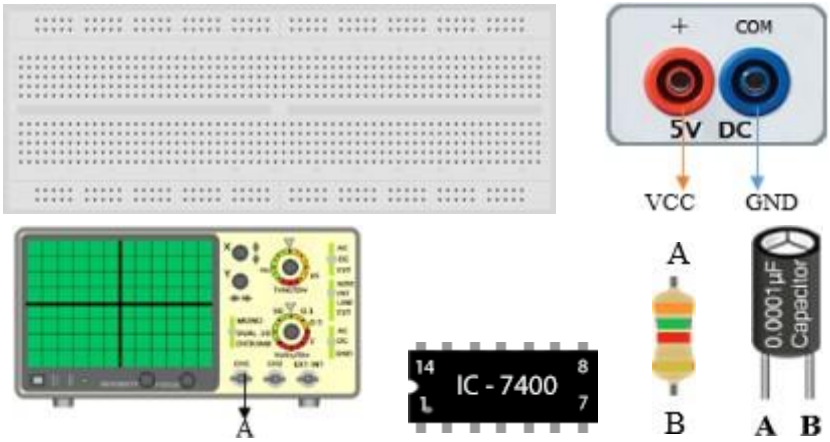


Fig. 1 Components

2. Make connections as per the circuit diagram and pin diagram of IC or according to connection table.
3. Connect the C.R.O on output terminal of circuit (Refer connection table).
4. Step-1) Select and drag "  " for generating wire of the circuit.
   Step-2)Join the wire to perform the required logic.
   Step-3) Click on check button to check the connections.
   Step-4) If connections are wrong click on reset button to reset connections.

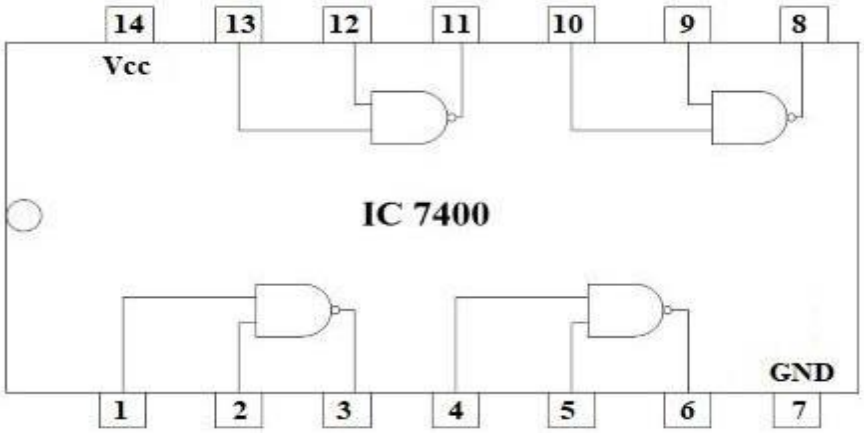


Fig. 3 Pin diagram of IC 7400 Table 1:

Connection table

S. No.	Source	Target
1	IC7400 _ Terminal 14	Supply VCC
2	IC7400 _ Terminal 07	Supply VCC
3	Capacitor (C) _ Terminal A	Supply GND
4	Capacitor (C) _ terminal B	Resistance (R_1) _ Terminal A
5	Capacitor (C) _ Terminal B	IC7400 _ Terminal 13
6	Capacitor (C) _ Terminal B	IC7400 _ Terminal 12
7	Resistance (R_1) _ Terminal B	IC7400 _ Terminal 11
8	IC7400 _ Terminal 11	CRO _ Terminal A

5. Click on **Check Connections** button. If connections are right, click on **'OK'**, then **Simulation** will become active.
6. Connect CH1/CH2 of C.R.O. to output terminal of the circuit.
7. Observe output wave on C.R.O by adjusting C.R.O channel **CH1/CH2** and **TIME** knobs.
8. Use **X** Shift and **Y** Shift buttons for wave shifting.
9. Measure the time period of output wave at C.R.O. and calculate the frequency by feeding the time period in the given box.
10. Compare both experimental and theoretical frequencies.
11. Click on the **Reset** button to reset the page.

Url: https://ade-iitr.vlabs.ac.in/exp/generation-of-clock/procedure.html

Conclusion :

The experiment on generation of clock pulses using NAND gates was successfully performed. It is observed that by connecting NAND gates with suitable RC components, a periodic square wave (clock signal) can be generated. The circuit operates as an astable multivibrator, continuously switching between logic HIGH and logic LOW states without any external triggering.

VIRTUAL LAB :05

AIM: To analyze the truth table of binary to gray and gray to binary converter using combination of NAND gates and to understand the working of binary to gray and gray to binary converter with the help of LEDs display.

Gray to Binary Conversion

Step-1) Connect battery to supply 5V to the circuit. Step-2)

Press Switches for different inputs.

The switch in ON state is and the switch in OFF state is

Step-3) The corresponding combination of input and output LEDs lit up for different combination of inputs.

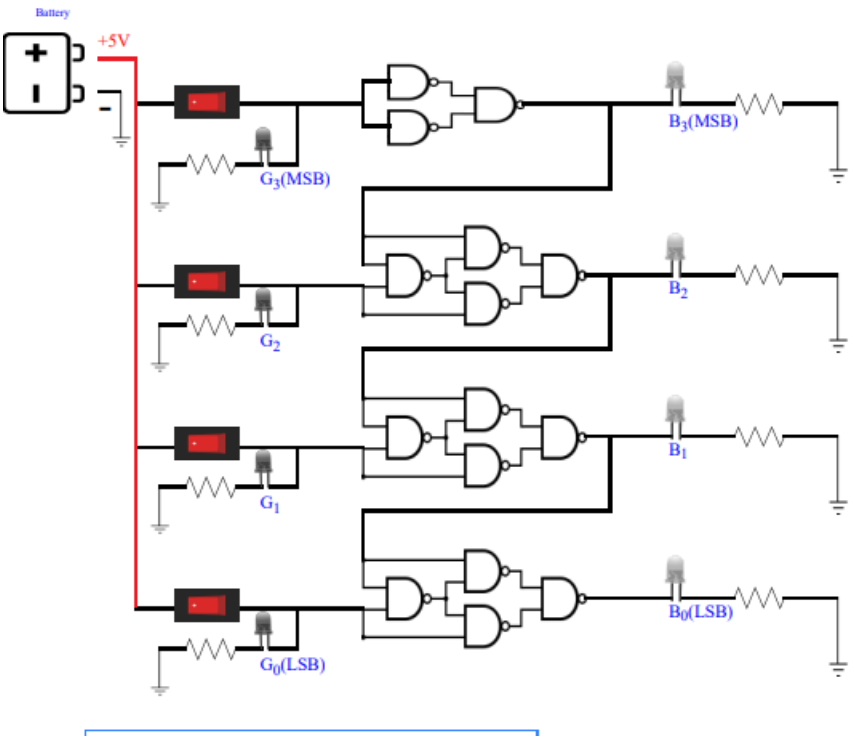
The input gray code LEDs are G3,G2,G1 and G0 and the output binary code LEDs B3,B2,B1 and B0 glow accordingly.

The input LED in OFF state is and in ON state is . The output

LED in OFF state is and in ON state is .

Step-4) Click "Add" to add the values to the Truth Table. Step-5)

Click "Print" to get the print out of the Truth Table.



Binary to Gray Code Conversion

Step-1) Connect battery to supply 5V to the circuit. Step-2)

Press Switches for different inputs.

The switch in ON state is and the switch in OFF state is

Step-3) The corresponding combination of input and output LEDs lit up for different combination of inputs.

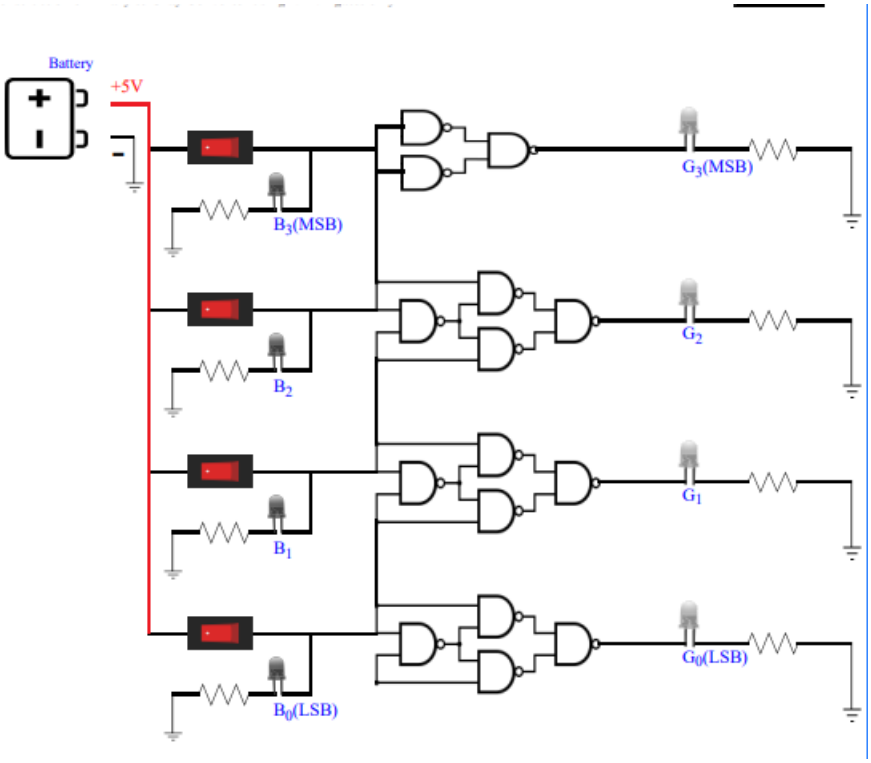
The input binary code LEDs are B3,B2,B1 and B0 and the output gray code LEDs G3,G2,G1 and G0 glow accordingly.

The input LED in OFF state is and in ON state is . The output

LED in OFF state is and in ON state is .

Step-4) Click "Add" to add the values to the Truth Table. Step-5)

Click "Print" to get the print out of the Truth Table.



Url: https://de-iitr.vlabs.ac.in/exp/binary-conversion/index.html

Conclusion :

The experiment on Binary to Gray (BTOG) and Gray to Binary (GTOB) code conversion was successfully carried out using logic gates. From the results obtained, it is observed that the binary number can be correctly converted into its corresponding Gray code and vice versa.

VIRTUAL LAB :06

Design and implement simple combinational circuits: 2-to-1 multiplexer, 1-bit comparator

Aim of the experiment:

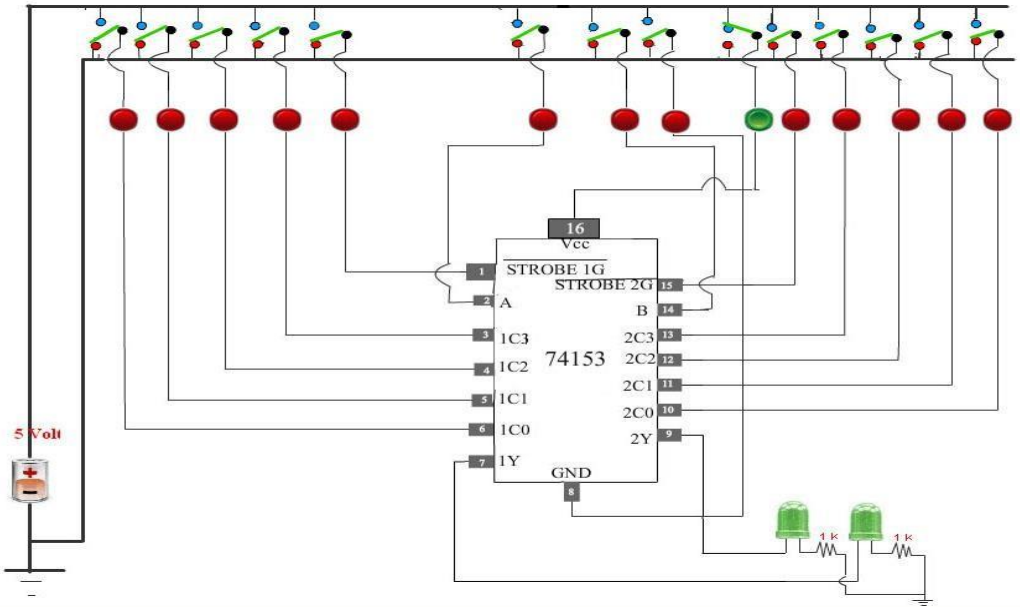
- The principal objective of this experiment(Part 1) is to fully understand the functionality of a Dual 4 line to 1 line Multiplexer(IC 74153) and to show the multiplex function of 74153 in terms of select lines. Note that each of the on-chip multiplexers act independently from the other, while sharing the same select lines
- The principal objective of this experiment(Part 2) is to fully understand the functionality of a Quad 2 line to 1 line Multiplexer(IC 74157) and to show the multiplex function of 74157 in terms of select line. Note that each of the on-chip multiplexers act independently from the other, while sharing the same select line.

Procedure:

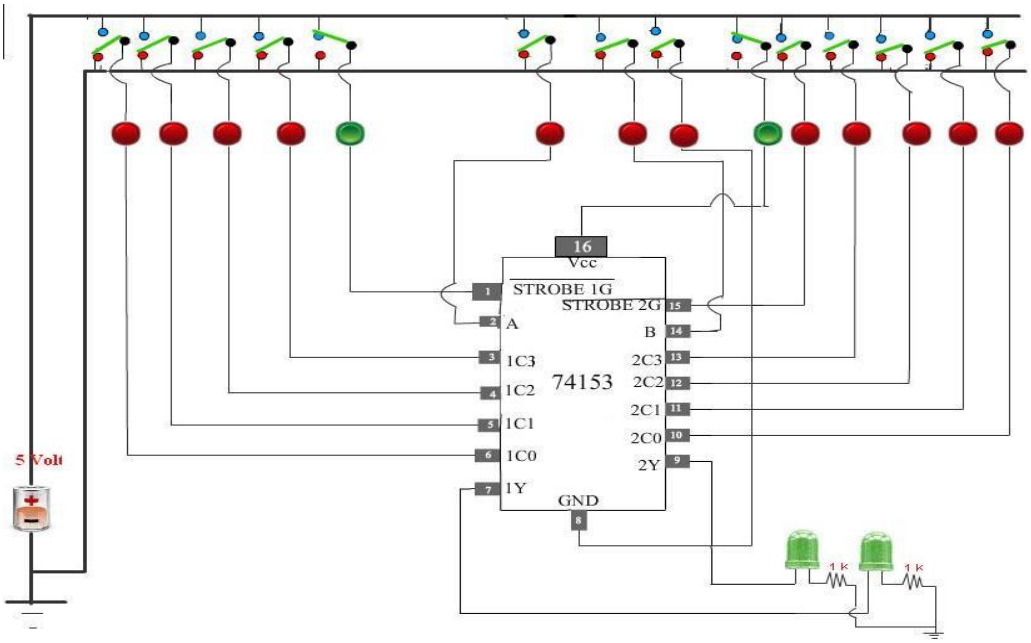
Please follow these steps to do the experiment.

Part 1:

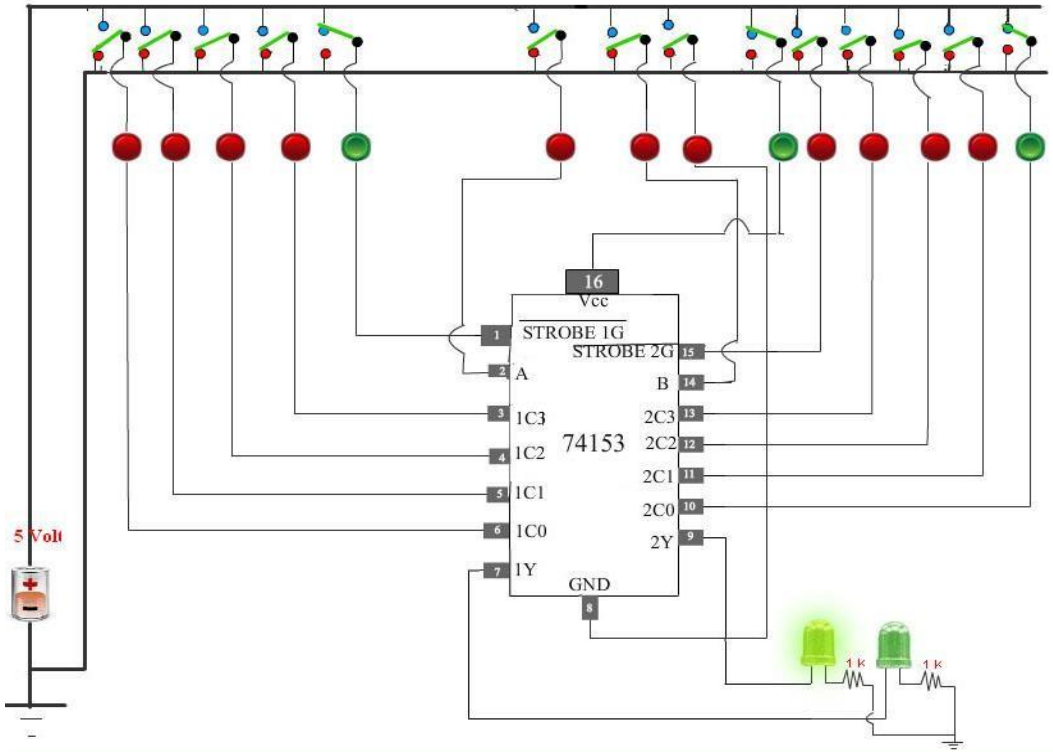
1. At first go through the structure of 74153.Then apply high level voltage to Vcc and low level voltage to GND. If Vcc and ground are not connected properly then error message will be shown and no output will be generated.



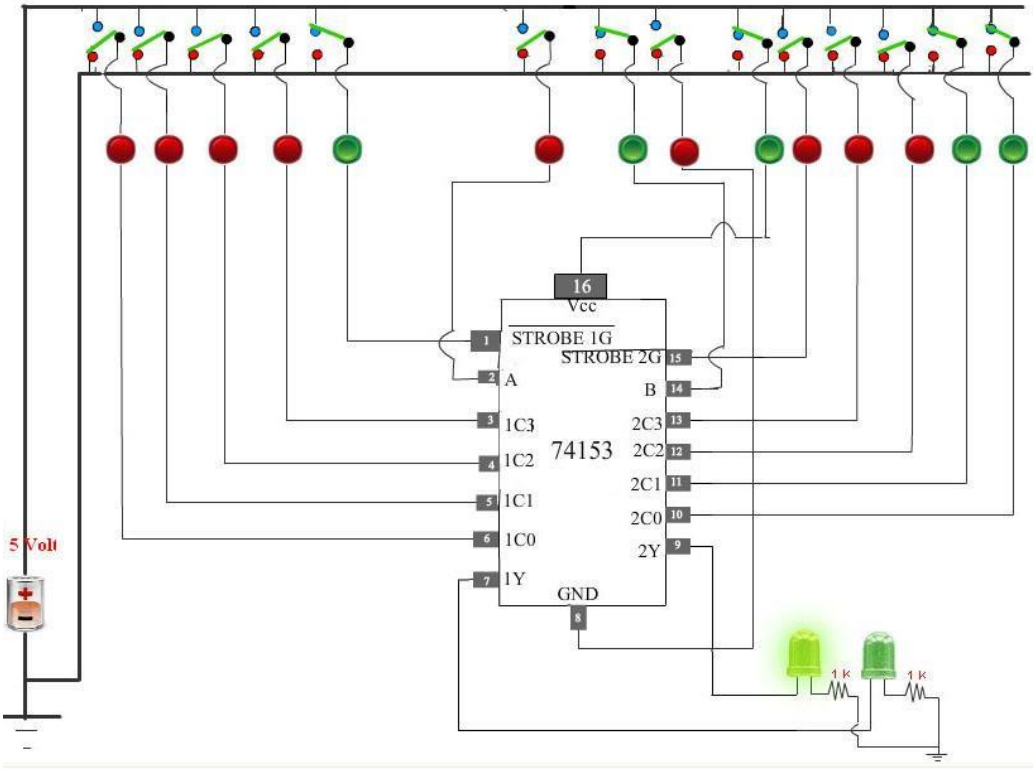
2. Next, apply high level voltage to Strobe1G or strobe 2G. If STROBE 1G is high 2nd Multiplexer is activated . If STROBE 2G is high then 1st Multiplexer is activated.



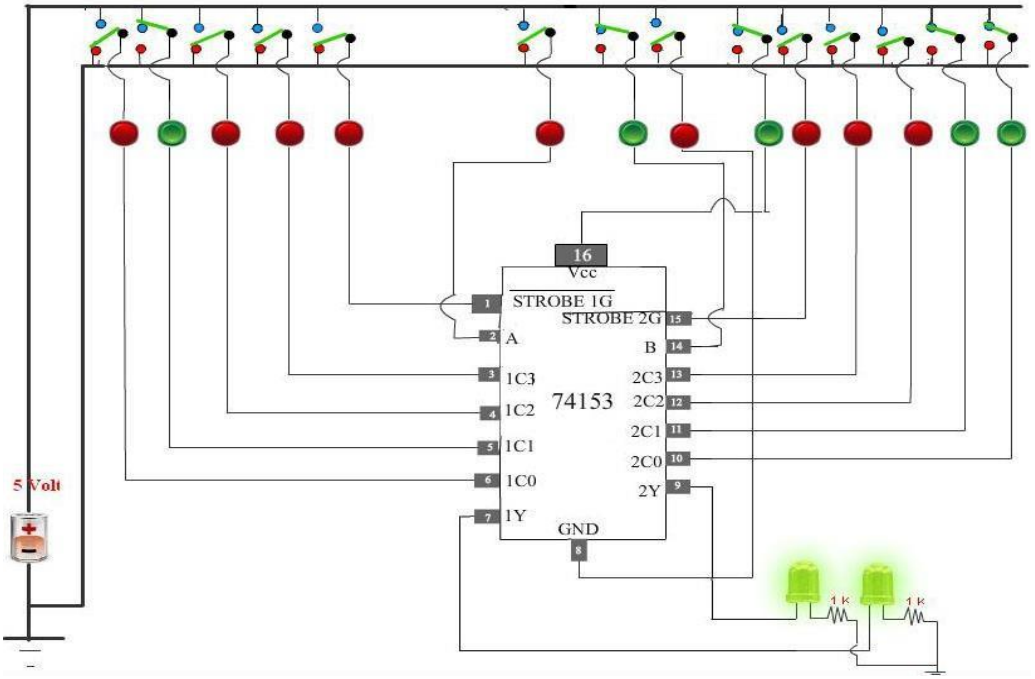
3. Next, apply low level voltage to the select inputs A and B (A Most significant Bit,B Less significant bit). Then apply a high level voltage to 2C0. Now check that how Dual 4 Line to 1 Line Multiplexer select the particular input to be multiplexed and to be applied to the output IY{1 = 1, 2}.



4. For all the combinations of the select inputs A,B verify that both the LEDs are glowing or not glowing. If the LED glows, it indicates that the corresponding output has reached logic 1 level. Similarly a dark LED indicates low level output voltage.

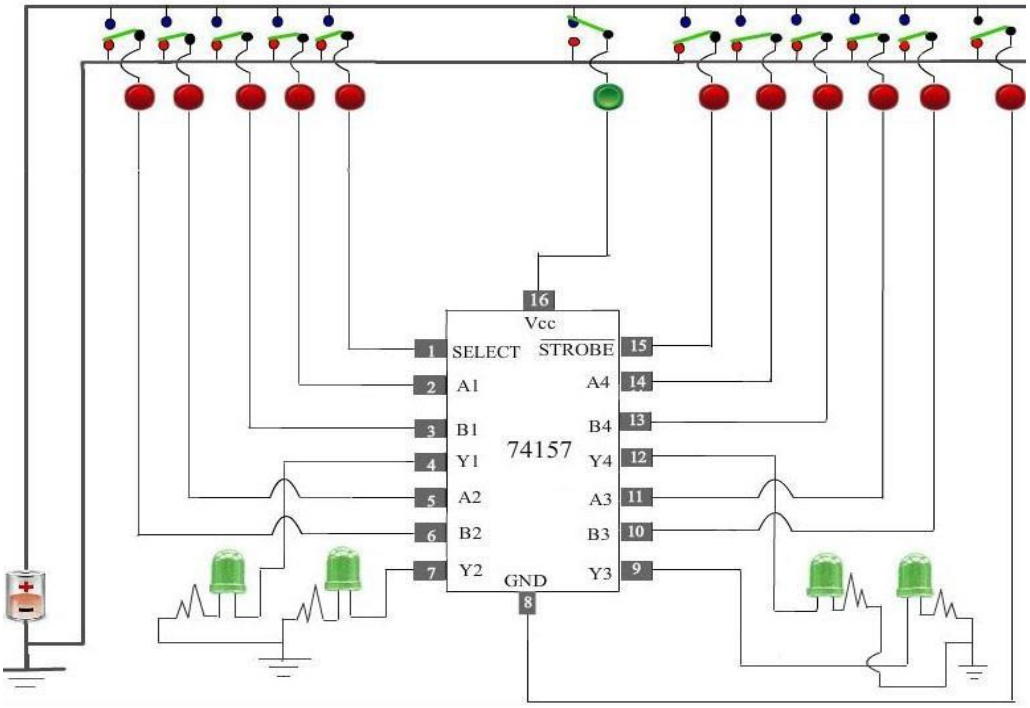


5. If both the Strobe inputs are low then both Multiplexers are activated.

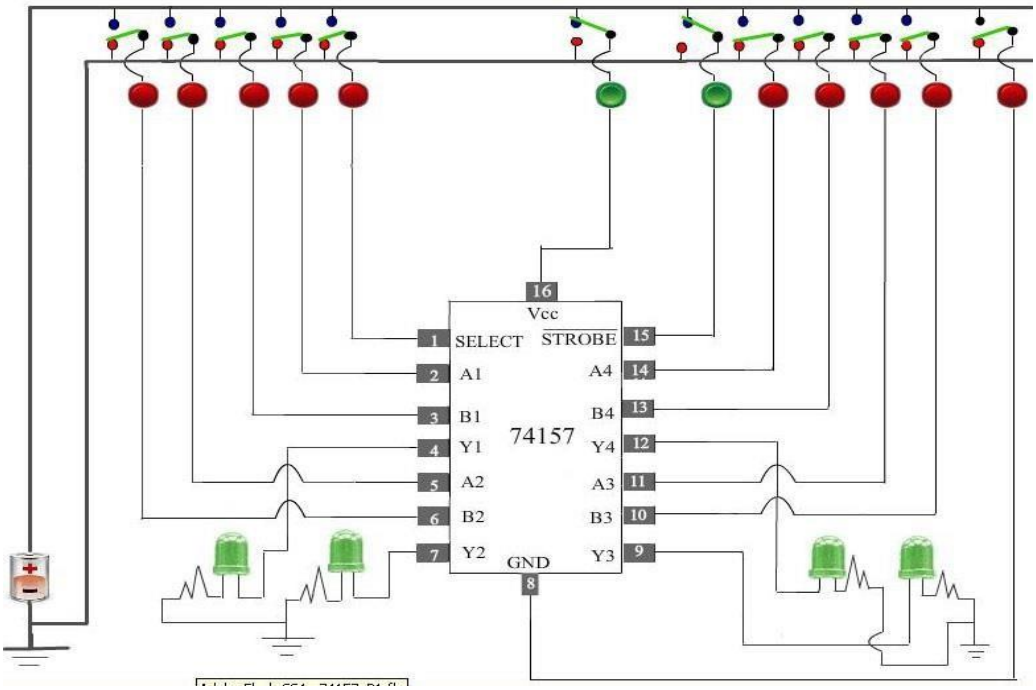


Part 2:

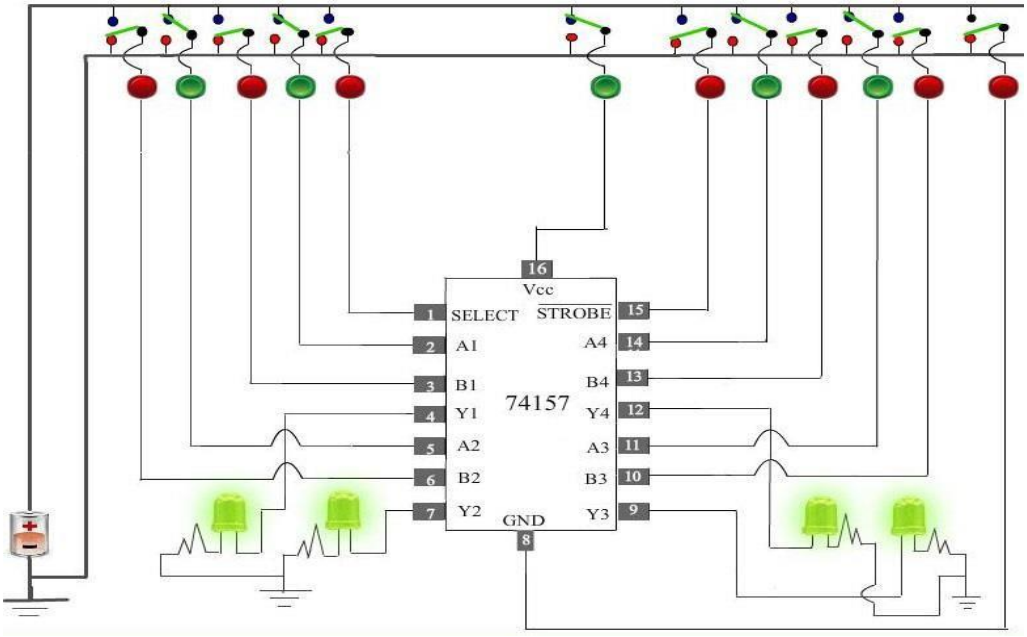
1. At first go through the structure of 74157.Then apply high level voltage to Vcc and low level voltage to GND. If Vcc and ground are not connected properly then error message will be shown and no output will be generated.



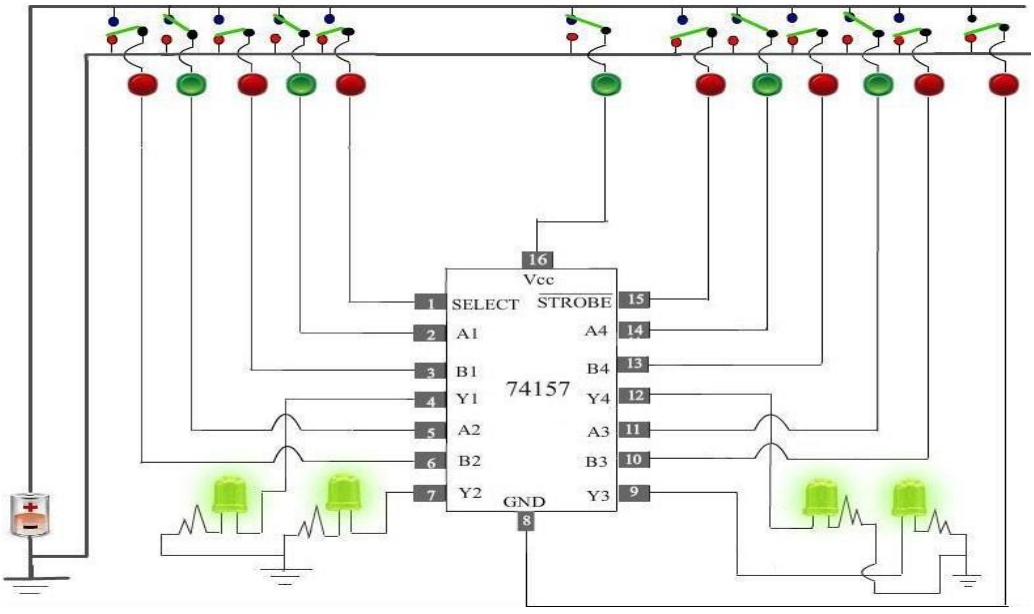
2. Next, apply high level voltage to Strobe input.Now check that all the LEDs are not glowing.Soall the outputs are at low state



3. Next, apply low level voltage to the Strobe input and also apply low level voltage to the Select input. Then apply high level voltage to A1,A2,A3,A4.Now check that all the LEDs are glowing.Because inputs are properly multiplexed to the outputs of the four multiplexers according to the voltage applied to the select input.



4. Next, apply low level voltage to the Strobe input and apply high level voltage to the Select input. Then apply high level voltage to B1,B2,B3,B4. Now check that all the LEDs are glowing.Because inputs are properly multiplexed to the outputs of the four multiplexers according to the voltage applied to the select input.



5. If the LED glows, it indicates that the corresponding output has reached logic 1 level. Similarly a dark LED indicates low level output voltage.

Url: https://dec-iitkgp.vlabs.ac.in/exp/functions-using-multiplexers/index.html

Conclusion:

The experiment demonstrates that complex logic functions can be efficiently implemented using multiplexers.The synthesized circuit produces the correct output for all input combinations, validating the use of multiplexers in logic function realization.

1. To verify De-Morgan's Theorems

Aim of the experiment:

To verify De-Morgan's theorems.

Procedure:

1. Under Simulation, click Theorem 1 or Theorem 2.

Familiarise with components

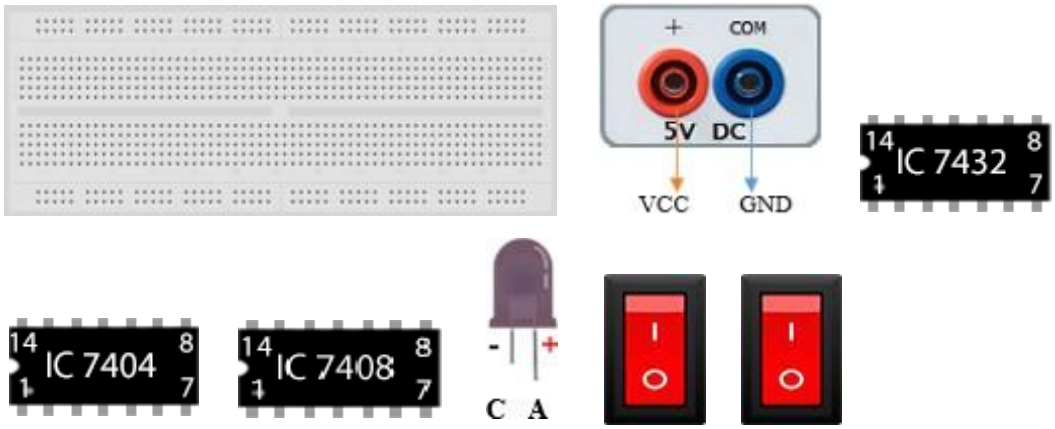


Figure 1: Components

1st Theorem :-

L.H.S. term for De-Morgan's 1st theorem, i.e (A + B)' :–

1. Click on the Component button to place components on the table.
2. Make connections as per the circuit diagram and pin diagrams of ICs or according to connection table.

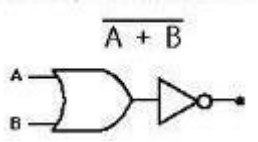
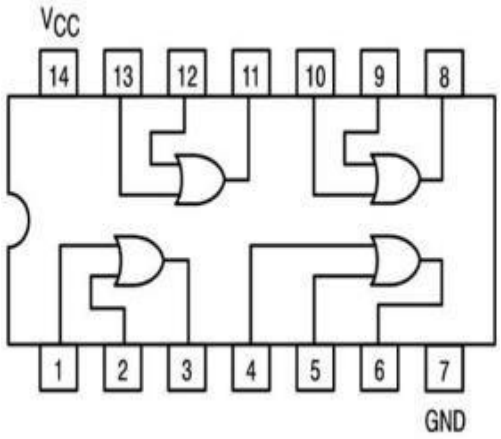


Figure 2 Circuit diagram of (A + B)'



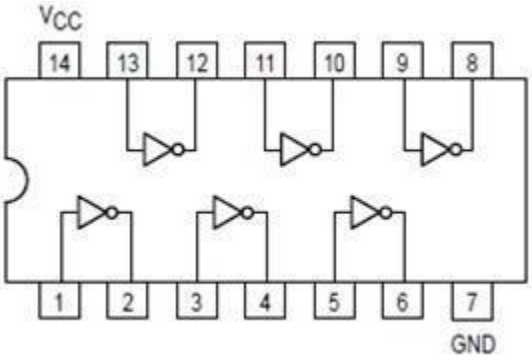


Figure 3 Pin diagram of IC 7432

Figure 4 Pin diagram of IC 7404

Table 1: Connection table for (A + B)'

S. No.	Source	Target
1	IC7432 _ Pin 14	Supply VCC
2	IC7432 _ Pin 07	Supply GND
3	IC7404 _ Pin 14	Supply VCC
4	IC7404 _ Pin 07	Supply GND
5	Led _ terminal C	Supply GND
6	Switch _ terminal A	IC7432 _ Pin 13
7	Switch _ terminal B	IC7432 _ Pin 12
8	IC7432 _ Pin 11	IC7404 _ Pin 13
9	IC7404 _ Pin 12	Led _ terminal A

3. Click on Check Connections button. If connections are right, click on 'OK', then Simulation will become active.
4. Provide the input by clicking toggle switches A and B.
5. Fill the observed values in the Truth Table.
6. Verify Truth Table by clicking on Check button, if outputs are correct then click on OK.
7. Click on the Result button provided below the table.

R.H.S. term for De-Morgan's 1st theorem, i.e A'B' :–

1. Click on the Component button to place components on the table.
2. Make connections as per the circuit diagram and pin diagrams of ICs or according to connection table.

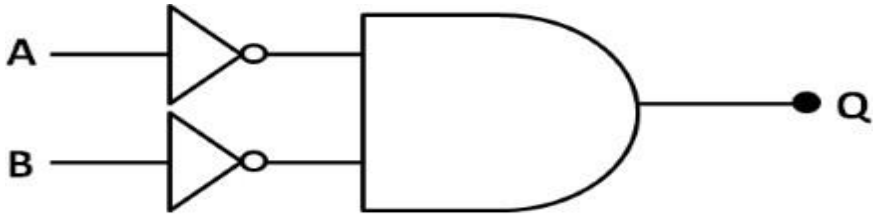


Figure 5: Circuit diagram of A'.B'

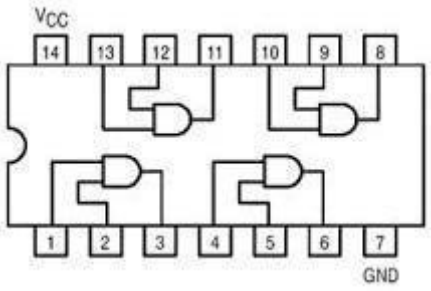


Figure 6: Pin diagram of IC 7408

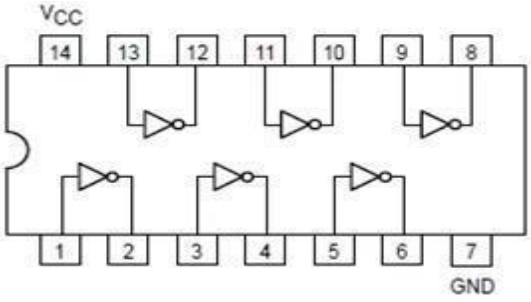


Figure 7 Pin diagram of IC 7404

Table 2: Connection table for A'.B'

S. No.	Source	Target
1	IC7404 _ Pin 14	Supply VCC
2	IC7404 _ Pin 07	Supply GND
3	IC7408 _ Pin 14	Supply VCC
4	IC7408 _ Pin 07	Supply GND
5	Led _ terminal C	Supply GND
6	Switch _ terminal A	IC7404 _ Pin 13
7	Switch _ terminal B	IC7404 _ Pin 11
8	IC7404 _ Pin 12	IC7408 _ Pin 13
9	IC7404 _ Pin 10	IC7408 _ Pin 12
10	IC7408 _ Pin 11	Led _ terminal A

3. Click on Check Connections button. If connections are right, click on 'OK', then Simulation will become active.
4. Provide the input by clicking toggle switches A and B.
5. Fill the observed values in the Truth Table.
6. Verify Truth Table by clicking on Check button, if outputs are correct then click on OK.
7. Click on the Result button provided below the table.

2nd Theorem :-

L.H.S. term for De-Morgan's 2nd theorem, i.e (A.B)' :–

1. Click on the Component button to place components on the table.
2. Make connections as per the circuit diagram and pin diagrams of ICs or according to connection table.

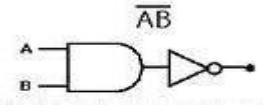


Figure 8: Circuit diagram of (A.B)'

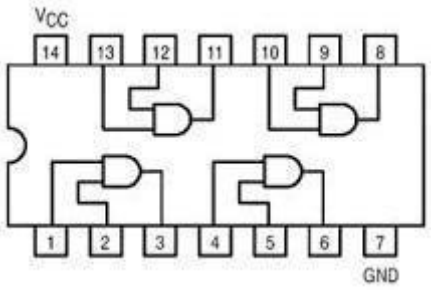
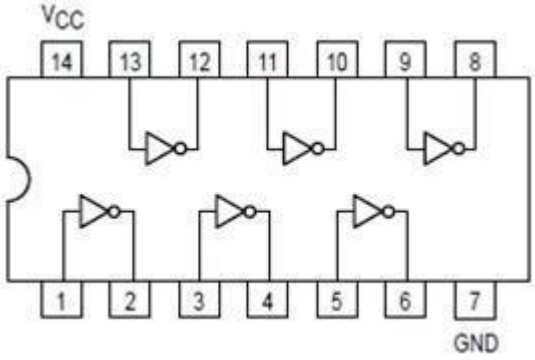


Figure 9: Pin diagram of IC 7408



Figue 10: Pin diagram of IC

7404 Table 3: Connection table for (A.B)'

S. No.	Source	Target
1	IC7408 _ Pin 14	Supply VCC
2	IC7408 _ Pin 07	Supply GND
3	IC7404 _ Pin 14	Supply VCC
4	IC7404 _ Pin 07	Supply GND
5	Led _ terminal C	Supply GND
6	Switch _ terminal A	IC7408 _ Pin 13
7	Switch _ terminal B	IC7408 _ Pin 12
8	IC7408 _ Pin 11	IC7404 _ Pin 13
9	IC7404 _ Pin 12	Led _ terminal A

3. Click on Check Connections button. If connections are right, click on 'OK', then Simulation will become active.
4. Provide the input by clicking toggle switches A and B.
5. Fill the observed values in the Truth Table.
6. Verify Truth Table by clicking on Check button, if outputs are correct then click on OK.
7. Click on the Result button provided below the table.

R.H.S. term for De-Morgan's 2nd theorem, i.e (A' + B') :–

1. Click on the Component button to place components on the table.
2. Make connections as per the circuit diagram and pin diagrams of ICs or according to connection table.

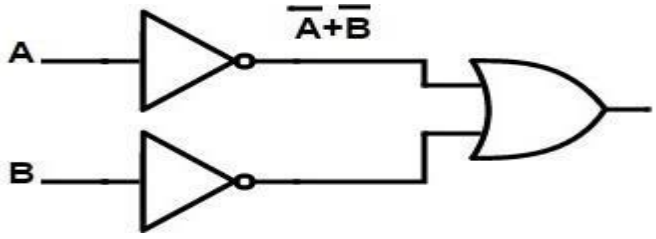


Figure 11: Circuit diagram of (A' + B')

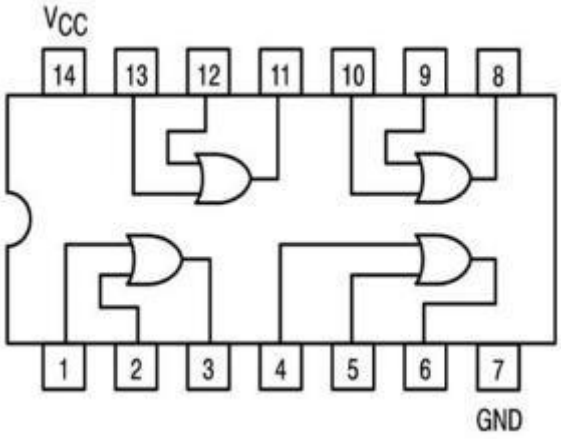


Figure 12: Pin diagram of IC 7432

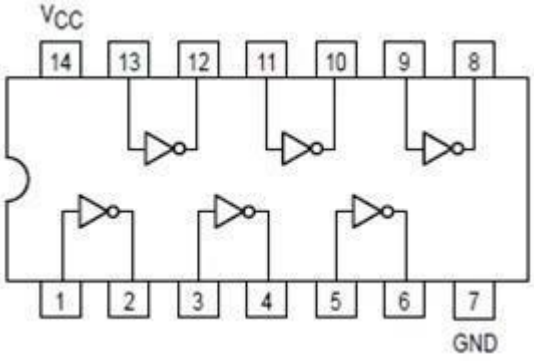


Figure 13: Pin diagram of IC 7404

Table 4: Connection table for (A' + B')

S. No.	Source	Target
1	IC7404 _ Pin 14	Supply VCC
2	IC7404 _ Pin 07	Supply GND
3	IC7432 _ Pin 14	Supply VCC
4	IC7432 _ Pin 07	Supply GND
5	Led _ terminal C	Supply GND
6	Switch _ terminal A	IC7404 _ Pin 13
7	Switch _ terminal B	IC7404 _ Pin 11
8	IC7404_ Pin 12	IC7432 _ Pin 13
9	IC7404_ Pin 10	IC7432 _ Pin 12
10	IC7432 _ Pin 11	Led _ terminal A

3. Click on Check Connections button. If connections are right, click on 'OK', then Simulation will become active.
4. Provide the input by clicking toggle switches A and B.
5. Fill the observed values in the Truth Table.
6. Verify Truth Table by clicking on Check button, if outputs are correct then click on OK.
7. Click on the Result button provided below the table.

Url: **https://ade-iitr.vlabs.ac.in/exp/de-morgans-theorems/**

Conclusion:

The experiment verifies De Morgan's theorems by showing the equivalence between complemented sums and products.

Thus, logical expressions can be simplified and implemented using alternative gate configurations with the same functionality.

PART-B

Experiment No: 1

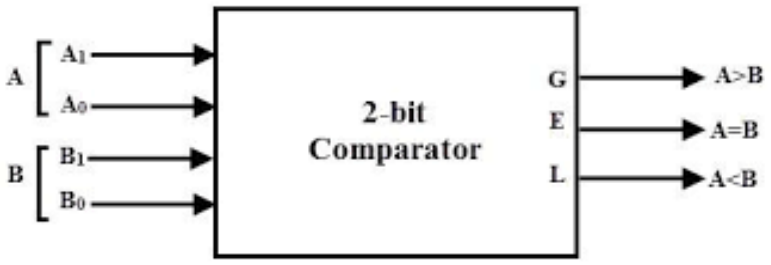
COMPARATOR

AIM: To design and simulate a 2-bit comparator using dataflow modelling; extend it to 4-bit using structural modelling.

TOOL: Cadence ISE Simulator (Verilog).

THEORY:

A magnitude digital Comparator is a combinational circuit that compares two digital or binary numbers to find out whether one binary number is equal, less than, or greater than the other binary number. We logically design a circuit for which we will have two inputs one for A and the other for B and have three output terminals, one for A > B condition, one for A = B condition, and one for A < B condition. The circuit works by comparing the bits of the two numbers starting from the most significant bit (MSB) and moving toward the least significant bit (LSB). At each bit position, the two corresponding bits of the numbers are compared. If the bit in the first number is greater than the corresponding bit in the second number, the A>B output is set to 1, and the circuit immediately determines that the first number is greater than the second. Similarly, if the bit in the second number is greater than the corresponding bit in the first number, the A<B output is set to 1, and the circuit immediately determines that the first number is less than the second.



Let us now derive the Boolean expression for the outputs L, E, and G.

Case 1: A = B

The comparator produces an output A = B which is E, if $A_0 = B_0$ and $A_1 = B_1$. Therefore, the Boolean expression for the output E will be,

$$E = (A_0 \odot B_0)(A_1 \odot B_1)$$

Case 2: A < B

The comparator produces an output A < B which is L, if

- $A_1 = 0$ and $B_1 = 1$, OR
- $A_1 = B_1$ and $A_0 = 0$ and $B_0 = 1$.

From these statements, we can write the Boolean expression for the output L as follows –

$$L = \overline{A_1}\, B_1 + (A_1 \odot B_1)\, \overline{A_0}\, B_0$$

Case 3: A > B

The output of the comparator will be A > B i.e., G, if

- $A_1 = 1$ and $B_1 = 0$, OR
- $A_1 = B_1$ and $A_0 = 1$ and $B_0 = 0$.

From these statements, the Boolean expression for the output G will be,

$$G = A_1 \overline{B_1} + (A_1 \odot B_1)\, A_0 \overline{B_0}$$

The following figure shows the logic circuit diagram of the 2-bit magnitude comparator –

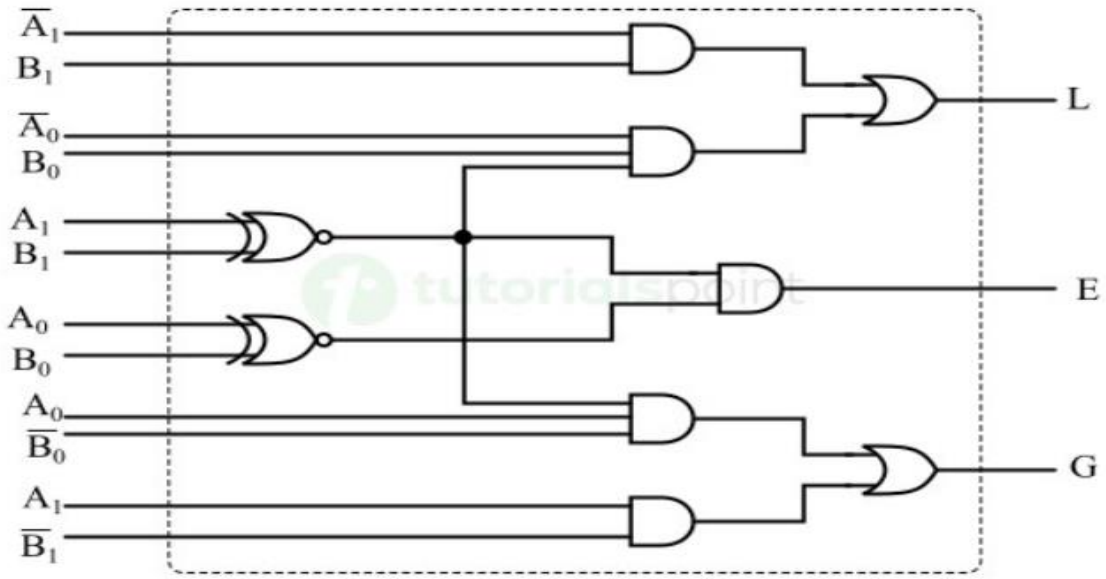


Fig: 2-bit Comparator circuit diagram

PROGRAM: 2-Bit Comparator (Dataflow Modeling)

```
module comparator_2bit (
input [1:0] a, b,
output greater, equal, lower);
assign greater = (a > b);
assign equal   = (a == b);
assign lower   = (a < b);
endmodule
```

Simulation Testbench (2-bit):

```
module tb_comparator_2bit;
reg [1:0] a, b;
wire greater, equal, lower;
comparator_2bit uut (.a(a), .b(b), .greater(greater), .equal(equal), .lower(lower));
initial begin
   $monitor("Time=%0t | a=%b b=%b | gt=%b eq=%b lt=%b", $time, a, b, greater, equal, lower);
   a = 2'b00; b = 2'b00; #10;
   a = 2'b01; b = 2'b10; #10;
   a = 2'b11; b = 2'b01; #10;
   a = 2'b10; b = 2'b10; #10;
   $finish;
end
endmodule
```

4-Bit Comparator (Structural Modeling)

```
module comparator_4bit_struct (
input [3:0] a, b,
output g, e, l);
wire g_h, e_h, l_h; // High bits (3,2)
wire g_l, e_l, l_l; // Low bits (1,0)

// Instantiate High 2-bit comparator (MSB)
 comparator_2bit high_comp (.a(a[3:2]), .b(b[3:2]), .greater(g_h), .equal(e_h), .lower(l_h));

// Instantiate Low 2-bit comparator (LSB)
 comparator_2bit low_comp (.a(a[1:0]), .b(b[1:0]), .greater(g_l), .equal(e_l), .lower(l_l));
// Structural logic to combine outputs
assign g = g_h | (e_h & g_l); // A > B if MSB > or (MSB equal and LSB >)
assign l = l_h | (e_h & l_l); // A < B if MSB < or (MSB equal and LSB <)
assign e = e_h & e_l;       // A = B only if both segments are equal
endmodule
```

Simulation Testbench (4-bit):

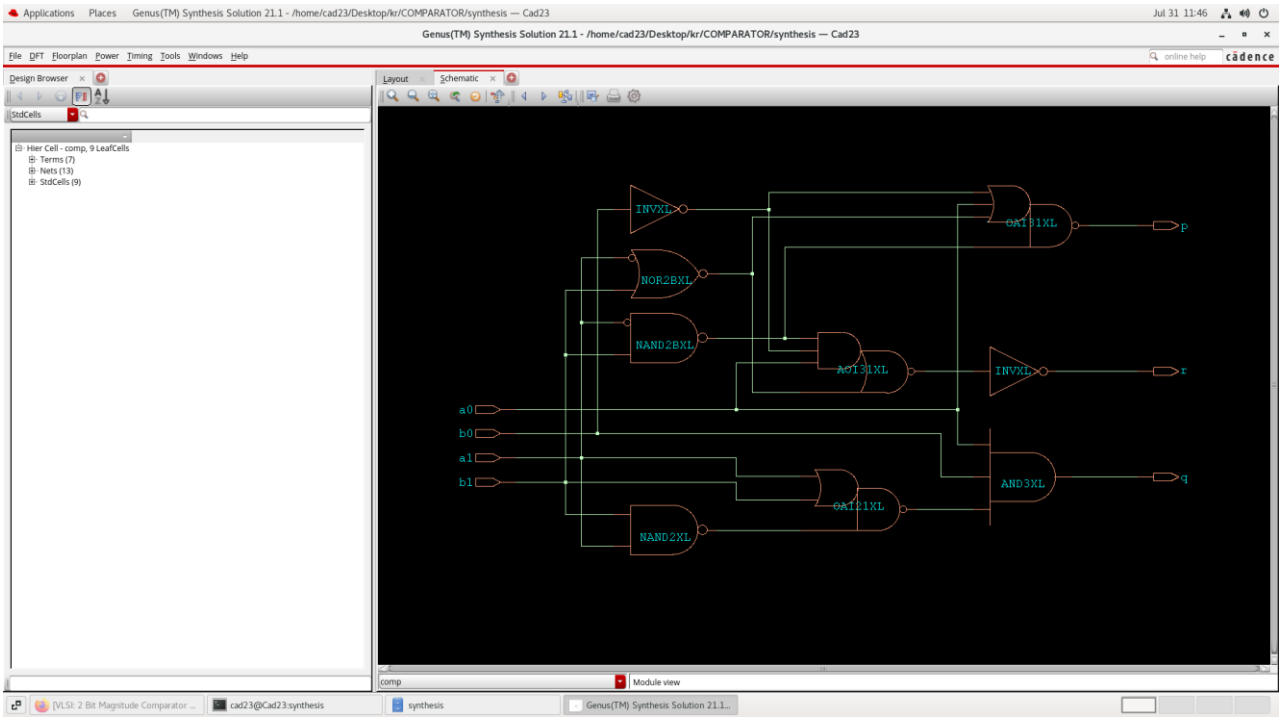
```
module tb_comparator_4bit;
 reg [3:0] a, b;
 wire g, e, l;
 // Instantiate UUT
 comparator_4bit_struct uut (.a(a), .b(b), .g(g), .e(e), .l(l));
 initial begin
    $monitor("Time=%0t | a=%d b=%d | gt=%b eq=%b lt=%b", $time, a, b, g, e, l);
    a = 4'b0000; b = 4'b0000; #10; // 0=0
    a = 4'b1010; b = 4'b1000; #10; // 10>8
    a = 4'b0101; b = 4'b1100; #10; // 5<12
    a = 4'b1111; b = 4'b1111; #10; // 15=15

 $finish;
 end
 endmodule
```

TRUTH TABLE :

INPUT				OUTPUT		
A1	A0	B1	B0	A<B	A=B	A>B
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

RTL SCHEMATIC:



SIMULATION RESULT:

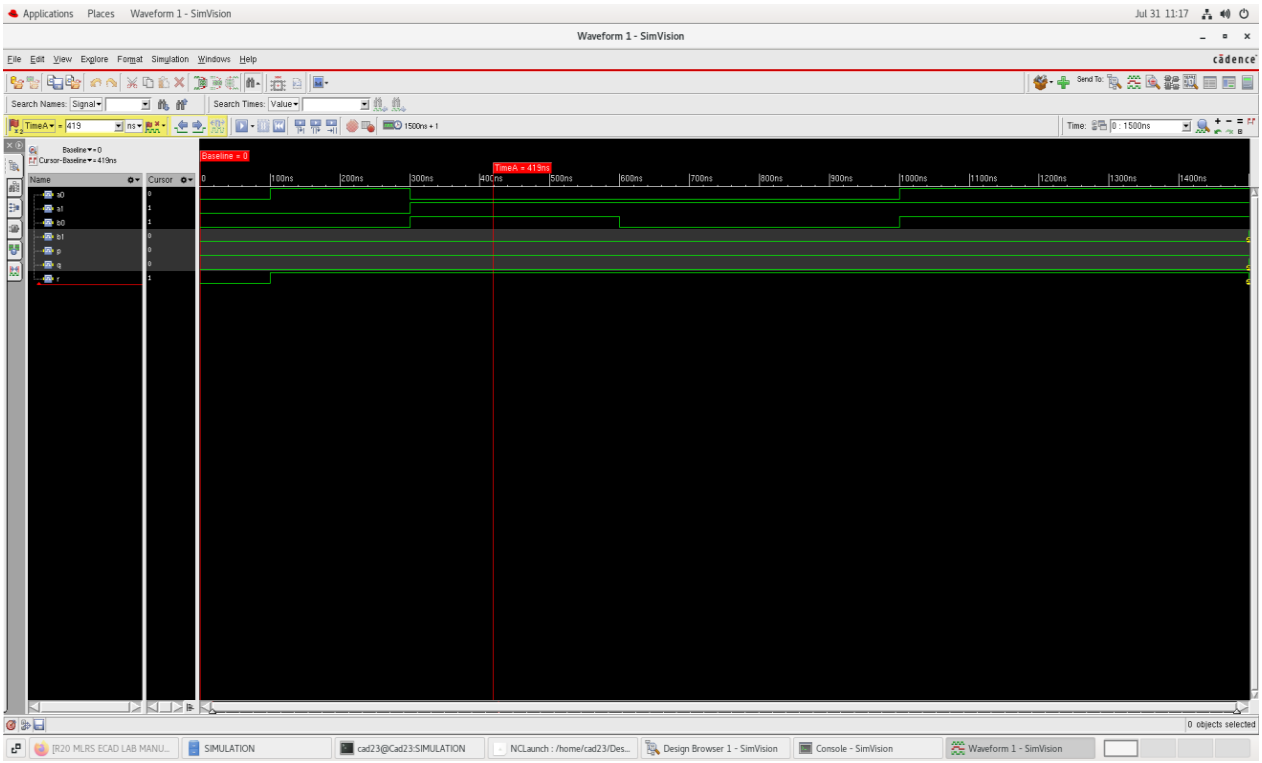


Fig: Output waveform of 2-bit comparator

RESULT:

Hence 2- Bit & 4-bit Comparator has been synthesized and simulated using Cadence ISE Simulator.

CONCLUSION:

End of the Experiment student's can able to realize 2-bit & 4-bit binary Comparator using Verilog HDL in Xilinx Vivado / Cadence tool.

VIVA QUESTIONS

S.No	Question	CO	Bloom's Taxonomy
1	What is a comparator in digital electronics?	CO1	Remember
2	What are the outputs of a comparator?	CO1	Remember
3	What is the function of a 2-bit comparator?	CO1	Remember
4	What is the function of a 4-bit comparator?	CO1	Remember
5	What are the possible comparison conditions between two binary numbers?	CO2	Understand
6	How do you design a 2-bit comparator using logic expressions?	CO3	Apply
7	How do you extend a 2-bit comparator to a 4-bit comparator?	CO3	Apply
8	What is the role of Verilog HDL in comparator design?	CO1	Remember
9	What is behavioral modeling in Verilog?	CO2	Understand
10	Which operators are used for comparison in Verilog?	CO2	Understand
11	How do you implement equality comparison in Verilog?	CO3	Apply
12	What is the purpose of simulation in Verilog design?	CO2	Understand
13	What is synthesis in FPGA design flow?	CO2	Understand
14	What is place and route in FPGA implementation?	CO2	Understand
15	What is the role of testbench in simulation?	CO3	Apply
16	How do you verify the correctness of a comparator?	CO3	Apply
17	What happens if inputs to comparator are equal?	CO2	Understand
18	What are the advantages of using HDL for comparator design?	CO2	Understand
19	What are the hardware resources used in FPGA for comparators?	CO4	Analyze
20	How can you optimize a comparator design?	CO5	Evaluate

Experiment No: 2A
2X1 MUX (data flow model)

AIM: To design and simulate the Multiplexer 2x1 data flow model using Verilog HDL.

TOOL: Cadence ISE Simulator (Verilog).

THEORY:

A multiplexer is a logic circuit that receives binary information from several inputs and transmits information to a single output. The selected input is controlled by a set of select inputs. In order to select one out of N input lines for connection to the output the number of select lines required is M such that 2^M = N. Alternately Multiplexer is a data selection device which selects one input line out of 2^n input lines depending on N number of selection lines.

PROGRAM:

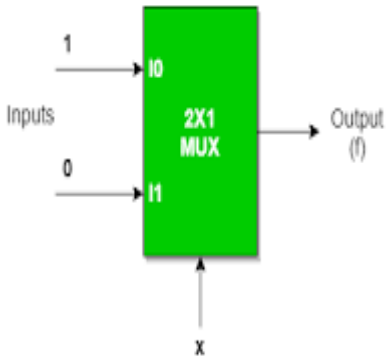
```
module mux2to1 (
input wire I0,   // Input 0
input wire I1,   // Input 1
input wire S,    // Select signal
output wire Y    // Output
 );
// Dataflow modeling using continuous assignment and ternary operator
assign Y = (S) ? I1 : I0;
endmodule
```

Simulation Testbench:

```
module tb_mux2to1;
   // Inputs
   reg I0, I1, S;
   // Outputs
   wire Y;
   // Instantiate the Unit Under Test (UUT)
   mux2to1 uut (
      .I0(I0),
      .I1(I1),
      .S(S),
      .Y(Y)
   );
   initial begin
      // Monitor changes
      $monitor("Time=%0t | S=%b | I1=%b I0=%b | Y=%b", $time, S, I1, I0, Y);
```

```
// Test Cases
    I0 = 0; I1 = 1; S = 0; #10; // S=0, Y should be I0 (0)
    I0 = 0; I1 = 1; S = 1; #10; // S=1, Y should be I1 (1)
    I0 = 1; I1 = 0; S = 0; #10; // S=0, Y should be I0 (1)
    I0 = 1; I1 = 0; S = 1; #10; // S=1, Y should be I1 (0)
    $finish;
 end
endmodule
```

2X1 Mux block diagram and Truth table:



Truth Table

x	f
0	1
1	0

RTL SCHEMATIC:

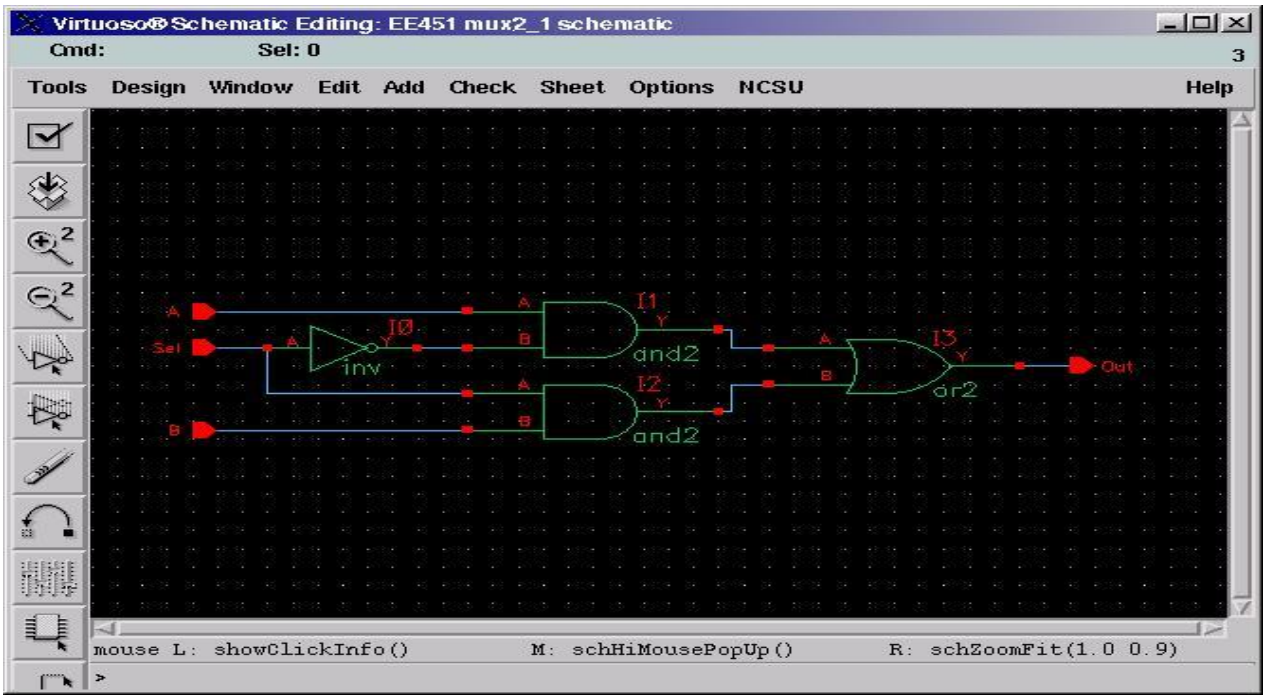


Fig: 2X1 MUX RTL schematic diagram

OUTPUT WAVEFORM:

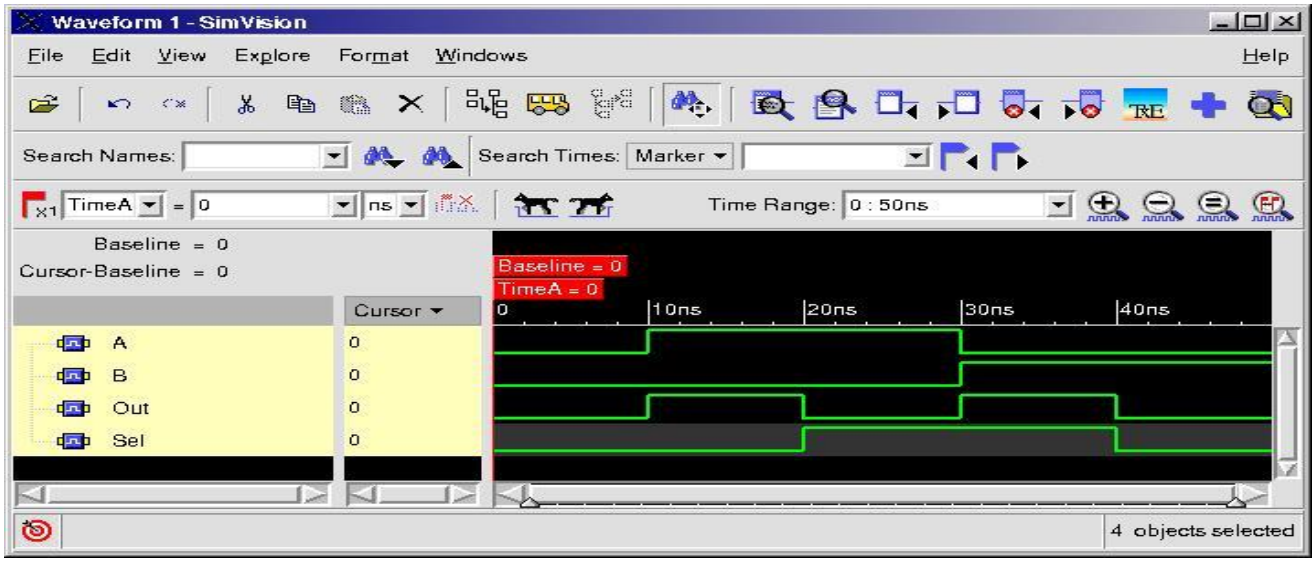


Fig: 2X1 MUX output waveform

RESULT:

Hence 2x1 Mux has been synthesized and simulated using Cadence ISE Simulator.

CONCLUSION:

End of the Experiment student's can able to realize and simulate any type of demultiplexer using Cadence / Xilinx vivado.

Experiment No: 2B
8X1 MUX (structural model)

AIM: To design and simulate the Multiplexer 8x1 Structural model using Verilog HDL.

TOOL: Cadence ISE Simulator (Verilog).

THEORY:

A multiplexer is a logic circuit that receives binary information from several inputs and transmits information to a single output. The selected input is controlled by a set of select inputs. In order to select one out of N input lines for connection to the output the number of select lines required is M such that 2^M = N. Alternately Multiplexer is a data selection device which selects one input line out of 2^n input lines depending on N number of selection lines.

PROGRAM:

```
module mux8to1(
input [7:0] d,
input [2:0] sel,
output y );
// Intermediate wires connecting stages
wire w1, w2, w3, w4, w5, w6;
// Stage 1: Four 2x1 Muxes
// Select S0 drives the first stage
mux2to1 m1(d[0], d[1], sel[0], w1);
mux2to1 m2(d[2], d[3], sel[0], w2);
mux2to1 m3(d[4], d[5], sel[0], w3);
mux2to1 m4(d[6], d[7], sel[0], w4);
// Stage 2: Two 2x1 Muxes
// Select S1 drives the second stage
mux2to1 m5(w1, w2, sel[1], w5);
mux2to1 m6(w3, w4, sel[1], w6);
// Stage 3: Final 2x1 Mux
// Select S2 drives the final stage
mux2to1 m7(w5, w6, sel[2], y);
endmodule
```

Simulation Testbench

```
module mux8to1_tb;
 reg [7:0] d;
 reg [2:0] sel;
 wire y;
 // Instantiate the Unit Under Test (UUT)
 mux8to1 uut (
    .d(d),
    .sel(sel),
    .y(y)
 );
 initial begin
    // Monitor outputs
    $monitor("Sel=%b, Data=%b, Output=%b", sel, d, y);
    // Apply test cases
    d = 8'hAA; // 10101010
    sel = 3'b000; #10; // Expected y = d[0] = 0
    sel = 3'b001; #10; // Expected y = d[1] = 1
    sel = 3'b010; #10; // Expected y = d[2] = 0
    sel = 3'b011; #10; // Expected y = d[3] = 1
    sel = 3'b100; #10; // Expected y = d[4] = 0
    sel = 3'b101; #10; // Expected y = d[5] = 1
    sel = 3'b110; #10; // Expected y = d[6] = 0
    sel = 3'b111; #10; // Expected y = d[7] = 1

    $finish;
  end
  endmodule
```

CIRCUIT DIAGRAM:

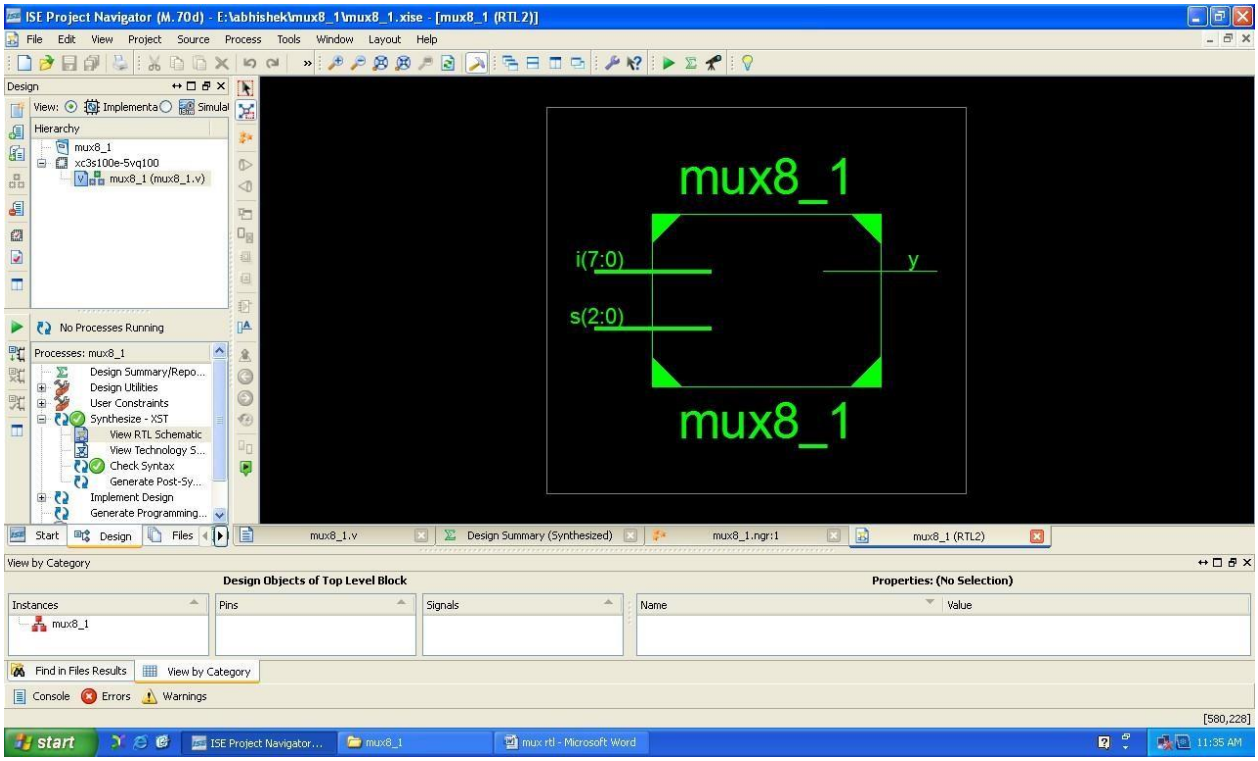


Fig: Circuit diagram of 8*1 Multiplexer

TRUTH TABLE:

S2	S1	S0	OUT
0	0	0	I(0)
0	0	1	I(1)
0	1	0	I (2)
0	1	1	I(3)
1	0	0	I(4)
1	0	1	I(5)
1	1	0	I(6)
1	1	1	I(7)

RTL SCHEMATIC:

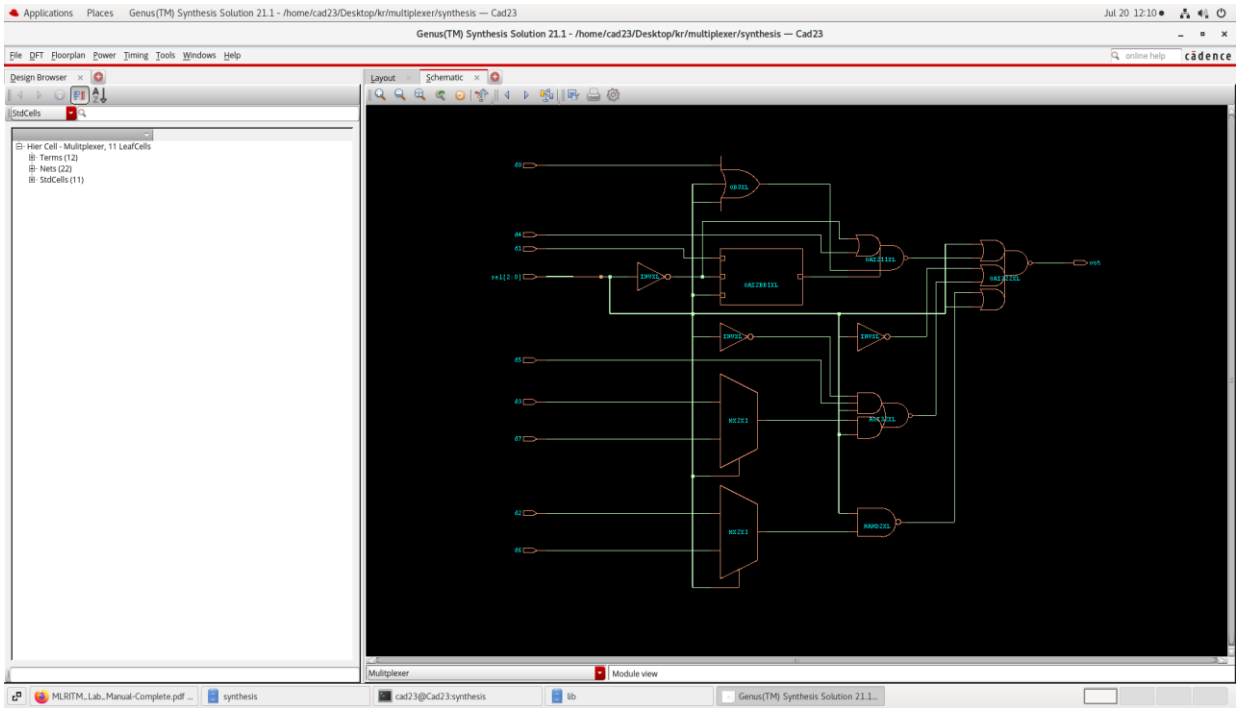


Fig: RTL schematic of 8*1 Multiplexer

TECHOLOGICALSCHEMATIC:

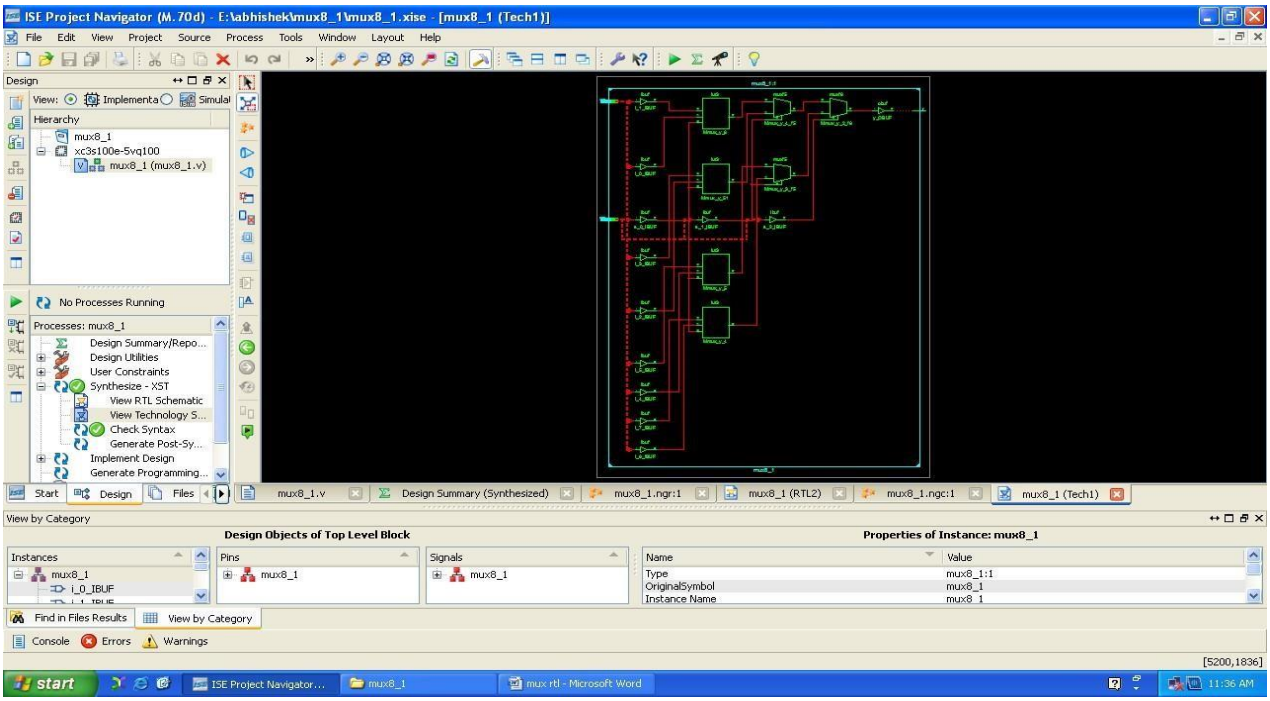


Fig: Technological schematic of 8*1 Multiplexer

OUTPUT WAVEFORM:

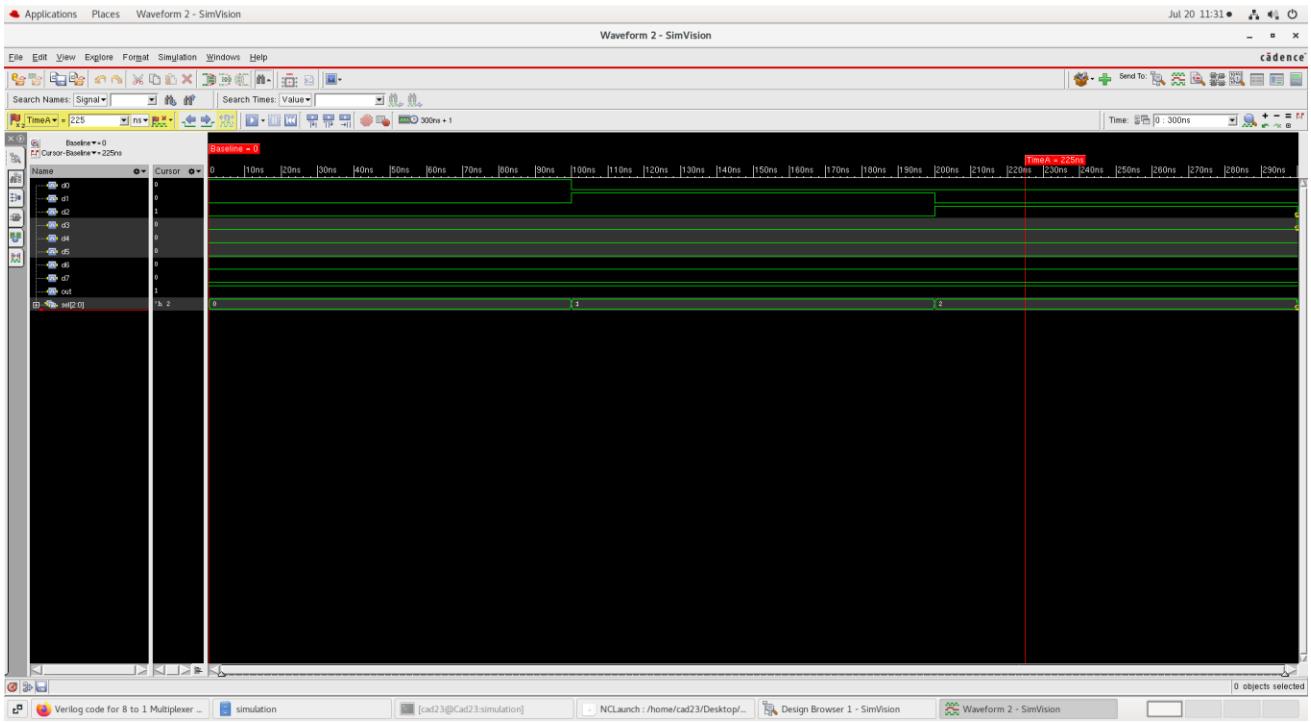


Fig: Output waveform of 8*1 Multiplexer

RESULT:

Hence 8x1mux have been synthesized and simulated using Cadence ISE Simulator.

CONCLUSION:

End of the Experiment student's can able to realize and simulate any type of multiplexer and demultiplexer using Cadence / Xilinx vivado.

VIVA QUESTIONS

S.No	Question	CO	Bloom's Taxonomy
1	What is a multiplexer (MUX)?	CO1	Remember
2	What is the function of a 2:1 multiplexer?	CO1	Remember
3	How many inputs and outputs does a 2:1 MUX have?	CO1	Remember
4	What is the role of the select line in a multiplexer?	CO2	Understand
5	Write the Boolean expression for a 2:1 MUX.	CO3	Apply
6	What is dataflow modelling in Verilog?	CO1	Remember
7	Which statement is used in dataflow modelling?	CO1	Remember
8	How do you implement a 2:1 MUX using assign statement?	CO3	Apply
9	What is structural modelling in Verilog?	CO1	Remember
10	How is structural modelling different from dataflow modelling?	CO2	Understand
11	How many select lines are required for an 8:1 MUX?	CO3	Apply
12	How can an 8:1 MUX be constructed using 2:1 MUXs?	CO3	Apply
13	How many 2:1 MUXs are required to build an 8:1 MUX?	CO3	Apply
14	What is the role of module instantiation in structural modelling?	CO2	Understand
15	How are signals connected between modules in structural design?	CO2	Understand
16	What is the purpose of a testbench in MUX design?	CO3	Apply
17	What happens if multiple inputs are high in a MUX?	CO4	Analyze
18	What are the applications of multiplexers?	CO2	Understand
19	What are the advantages of using MUX in digital systems?	CO2	Understand
20	How can you optimize the design of a multiplexer?	CO5	Evaluate

Experiment No: 3A

2 to 4 DECODER (data flow model)

AIM: To design and simulate Module 2 to 4 Decoder using dataflow model in Verilog HDL.

TOOL: Cadence ISE Simulator (Verilog).

THEORY:

Decoder is a combinational logic device, which converts N number of inputs into 2^N number of outputs.

In its implementation it requires 2^N number of AND GATES.

PROGRAM:

```
module decoder2to4 (
input a, b,
output y0, y1, y2, y3);
// Dataflow modeling using continuous assignments
assign y0 = (~a) & (~b); // 00
assign y1 = (~a) & b;    // 01
assign y2 = a & (~b);    // 10
assign y3 = a & b;       // 11
endmodule
```

Simulation Testbench (Verilog)

```
module tb_decoder2to4;
  // Inputs
  reg a, b;
  // Outputs
  wire y0, y1, y2, y3;
  // Instantiate the Unit Under Test (UUT)
  decoder2to4 uut (
    .a(a),
    .b(b),
```

```
 .y0(y0),
 .y1(y1),
 .y2(y2),
 .y3(y3));
 initial begin
 // Monitor outputs
 $monitor("Time=%0t | a=%b b=%b | y3 y2 y1 y0 = %b%b%b%b",
        $time, a, b, y3, y2, y1, y0);
 // Apply test cases
 a=0; b=0; #10; // y0 activated
 a=0; b=1; #10; // y1 activated
 a=1; b=0; #10; // y2 activated
 a=1; b=1; #10; // y3 activated
 $finish; // End simulation
end
endmodule
```

CIRCUIT DIAGRAM:

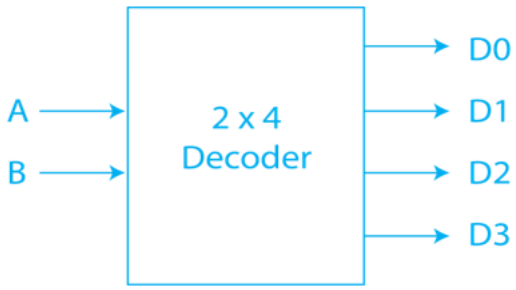
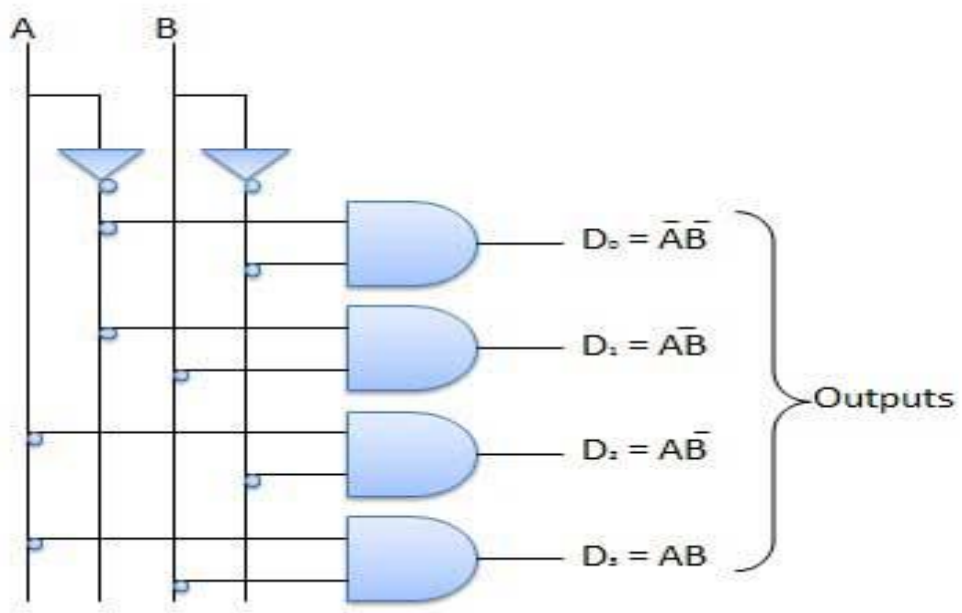


Fig: Block diagram of 2 to 4 Decoder

TRUTH TABLE:

A	B	Y0	Y1	Y2	Y3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

2X4 circuit diagram:



RTL SCHEMATIC:

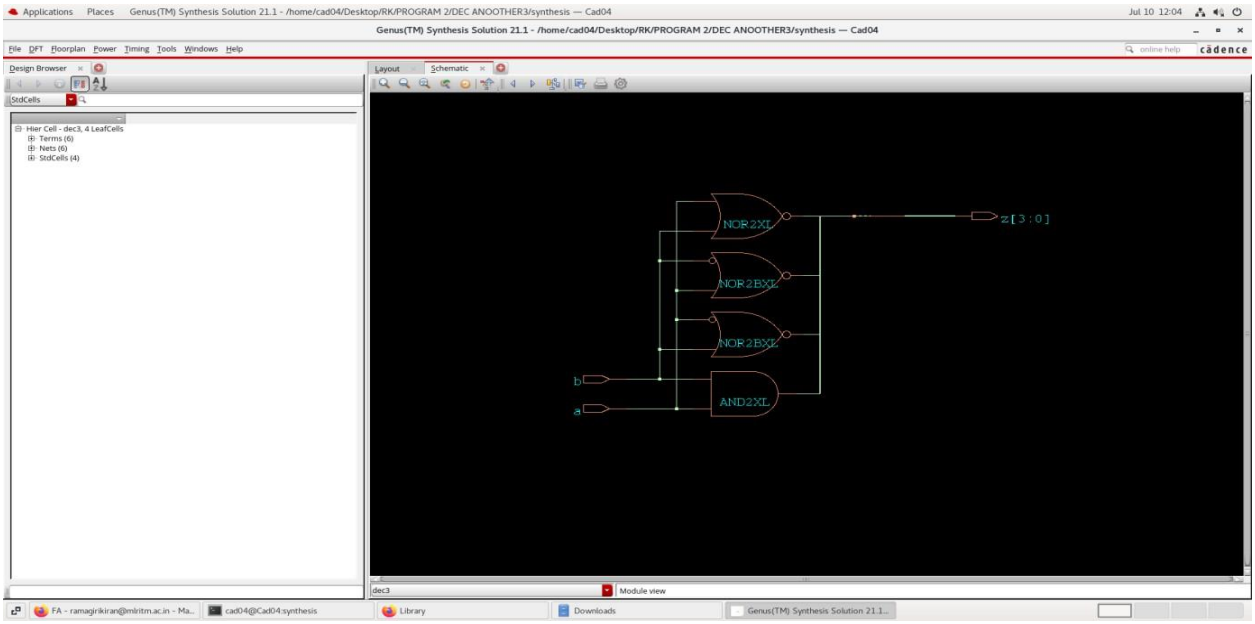


Fig: RTL Schematic of 2 to 4 Decoder

TECHNOLOGICAL SCHEMATIC:

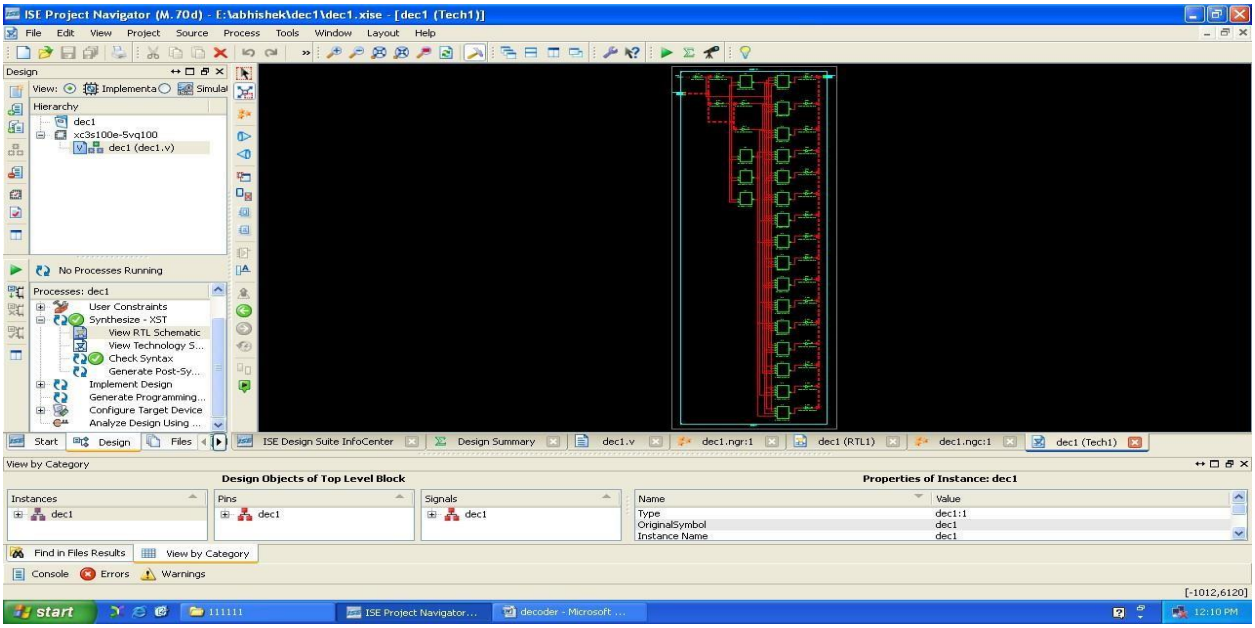


Fig: Technological schematic of 2 to 4 Decoder

OUTPUT WAVE FORM:

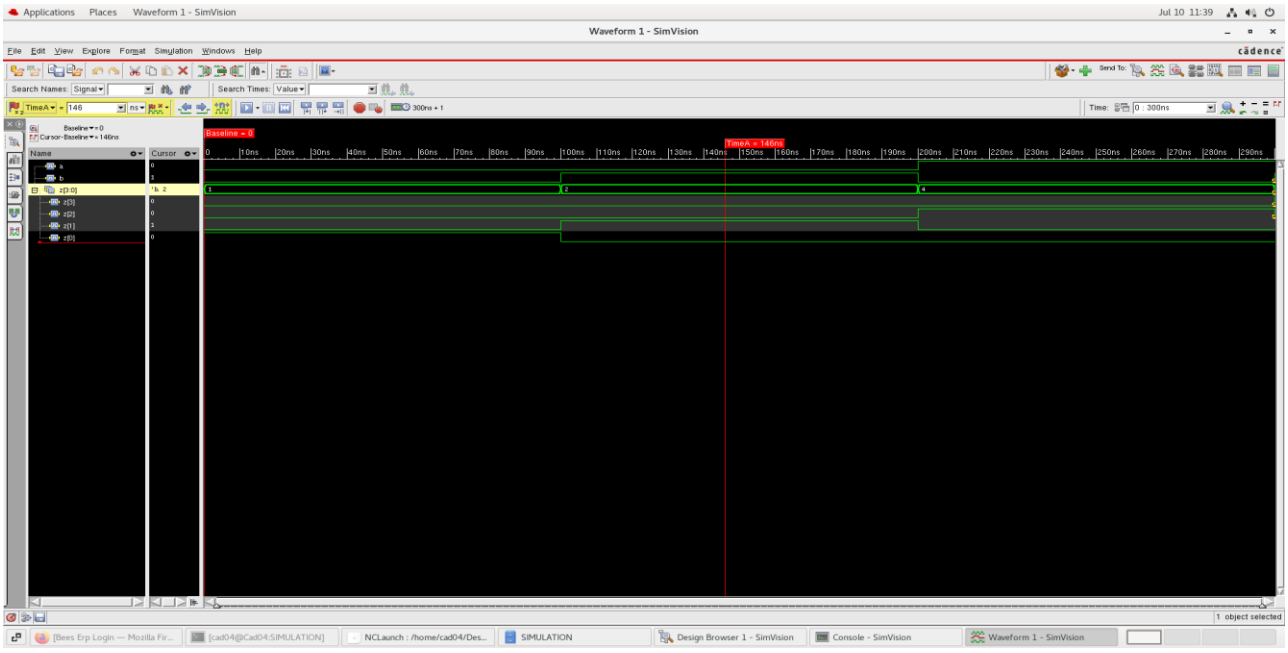


Fig: Output waveform of 2 to 4 Decoder

RESULT:

Hence 2 to 4 decoder has been synthesized and simulated using Cadence ISE Simulator.

CONCLUSION:

End of the Experiment student's can able to design and simulate 8-to-3 encoder (without and with priority) and 4-to-16 decoder using Xilinx Vivado/CADENCE.

Experiment No: 3B

3x8 DECODER(Structural model)

AIM: To design and simulate Decoder 3x8 using structural model using Verilog HDL.

TOOL: Cadence ISE Simulator (Verilog).

THEORY:

A decoder is a combinational circuit that performs the reverse operation of an encoder, converting n input lines into 2^n output lines, with only one output active for each input combination.

PROGRAM:

3-to-8 Decoder Verilog Code (Structural)

```
module decoder_3_8 (
 input [2:0] i,
 output [7:0] y);
 wire a_n, b_n, c_n;
 // Inverters
 not n1(a_n, i[2]);
 not n2(b_n, i[1]);
 not n3(c_n, i[0]);
 // AND Gates (Active High Output)
 and g0(y[0], a_n, b_n, c_n); // 000
 and g1(y[1], a_n, b_n, i[0]); // 001
 and g2(y[2], a_n, i[1], c_n); // 010
 and g3(y[3], a_n, i[1], i[0]); // 011
 and g4(y[4], i[2], b_n, c_n); // 100
 and g5(y[5], i[2], b_n, i[0]); // 101
 and g6(y[6], i[2], i[1], c_n); // 110
 and g7(y[7], i[2], i[1], i[0]); // 111
 endmodule
```

Simulation Testbench

```
module tb_decoder3x8;
    reg [2:0] A;
    wire [7:0] Y;
 // Instantiate the Unit Under Test (UUT)
    decoder3x8 uut (.Y(Y), .A(A));
  initial begin
        $monitor("Input: %b | Output: %b", A, Y);
        // Apply all input combinations
        A = 3'b000; #10;
        A = 3'b001; #10;
        A = 3'b010; #10;
        A = 3'b011; #10;
        A = 3'b100; #10;
        A = 3'b101; #10;
        A = 3'b110; #10;
        A = 3'b111; #10;
        $finish;
    end
    endmodule
```

CIRCUIT DIAGRAM:

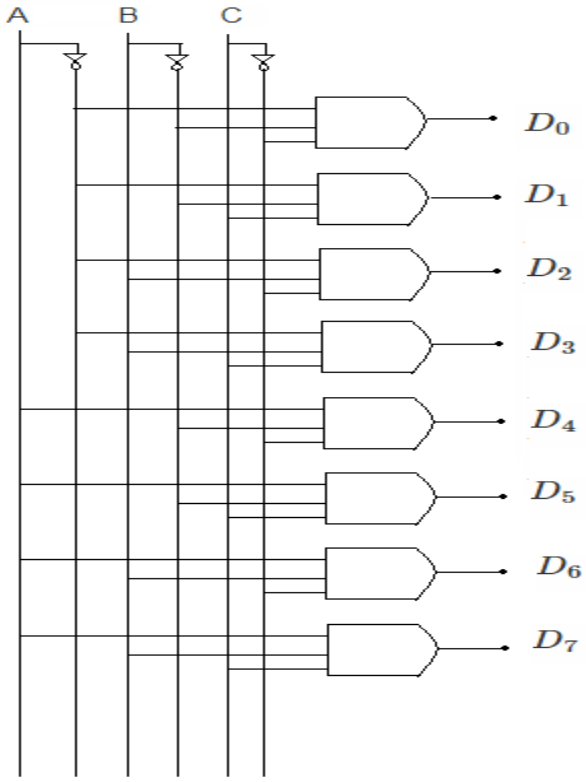


Fig: Circuit diagram of 3 to 8 Decoder

TRUTH TABLE:

A_2	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

RTL SCHEMATIC:

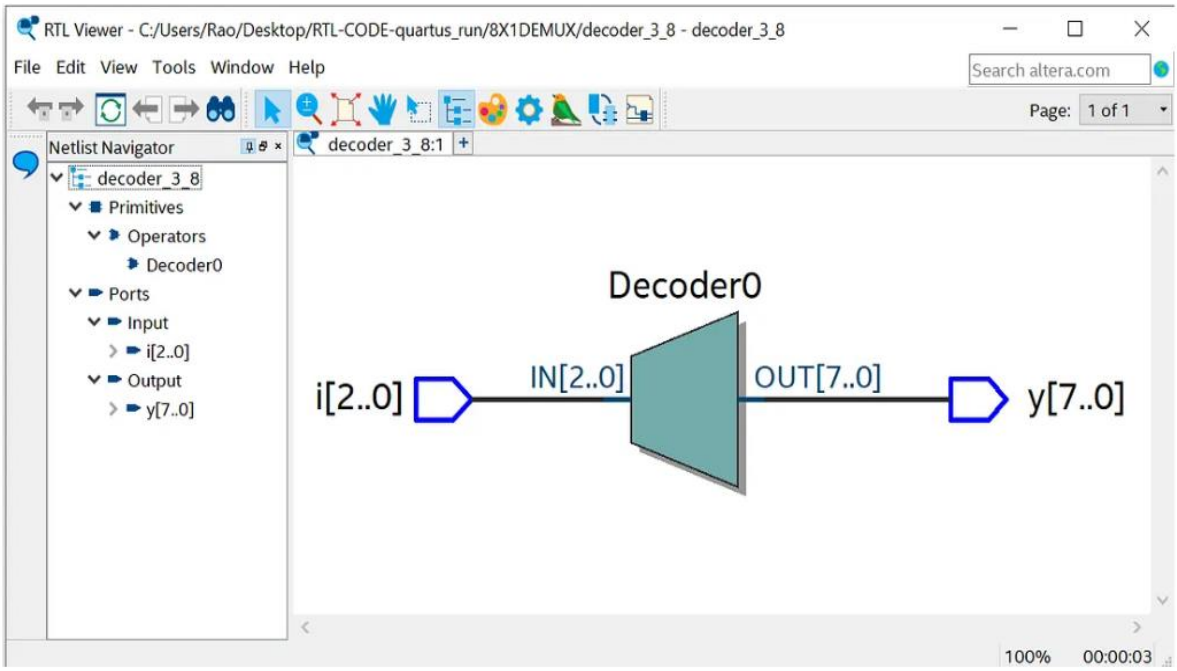


Fig : RTL schematic of 3 to 8 decoder

OUTPUT WAVEFORM:

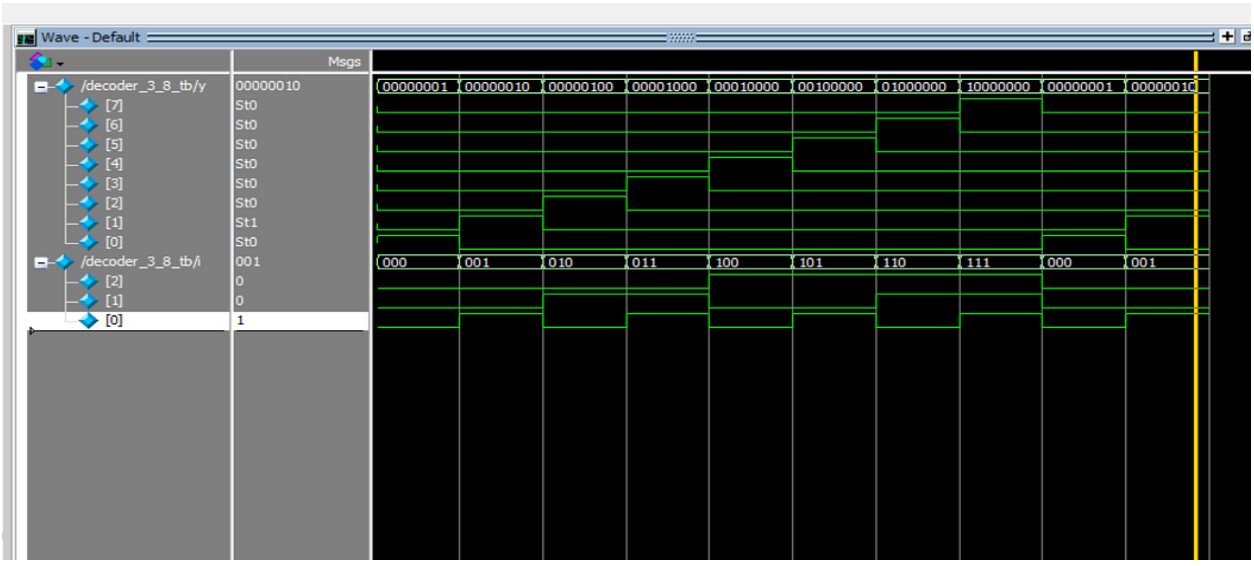


Fig : Output waveform of 3 to 8 Decoder

RESULT:

Hence 3 to 8 decoder has been synthesized and simulated using Cadence ISE Simulator.

CONCLUSION:

End of the Experiment student's can able to design and simulate 3-to-8 decoder (without and with priority) and 2-to-4 decoder using Cadence / Xilinx Vivado.

VIVA QUESTIONS

S.No	Question	CO	Bloom's Taxonomy
1	What is a decoder?	CO1	Remember
2	What is the function of a 3×8 decoder?	CO1	Remember
3	How many inputs and outputs does a 3×8 decoder have?	CO1	Remember
4	What is the relation between number of inputs and outputs in a decoder?	CO2	Understand
5	What is meant by minterms in a decoder?	CO2	Understand
6	What is structural modelling in Verilog?	CO1	Remember
7	How is structural modelling different from behavioural modelling?	CO2	Understand
8	Which basic gates are used to implement a 3×8 decoder?	CO1	Remember
9	How many AND gates are required for a 3×8 decoder?	CO2	Understand
10	Why are NOT gates used in decoder design?	CO2	Understand
11	What is the role of enable input in a decoder?	CO2	Understand
12	How do you write a structural model in Verilog?	CO3	Apply
13	How are modules interconnected in structural modelling?	CO3	Apply
14	What is the purpose of wire in Verilog structural design?	CO2	Understand
15	How do you verify the decoder functionality?	CO3	Apply
16	What happens when multiple inputs are high in a decoder?	CO4	Analyze
17	What is the difference between decoder and encoder?	CO2	Understand
18	How can a decoder be used to implement Boolean functions?	CO3	Apply
19	What are the applications of a decoder?	CO2	Understand
20	How can you extend a 3×8 decoder to higher-order decoders?	CO5	Evaluate

Experiment No: 4

BOOLEAN FUNCTION USING A DECODER

AIM: Implement a given Boolean function using a decoder-based approach in behavioural modelling.

TOOL: Cadence ISE Simulator (Verilog).

THEORY:

Implementing a Boolean function using a decoder in behavioural modelling involves using a case statement or always block to define the decoder's functionality (mapping inputs to minterms) and subsequently ORing the specific minterms required by the Boolean function.

$$\text{Example: } F(A, B, C) = \sum m(1, 3, 5, 6)$$

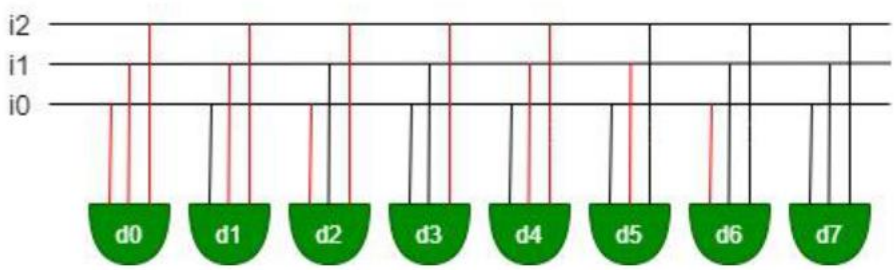
Behavioral Model of a 3-to-8 Decoder:

```
module boolean_decoder_beh(
  input A, B, C, // 3 variables
  output F );      // Output function
  wire [7:0] dec_out;
 always @(*) begin
  case ({A,B,C})
     3'b000: dec_out = 8'b0000_0001;
     3'b001: dec_out = 8'b0000_0010; // minterm 1
     3'b010: dec_out = 8'b0000_0100;
     3'b011: dec_out = 8'b0000_1000; // minterm 3
     3'b100: dec_out = 8'b0001_0000;
     3'b101: dec_out = 8'b0010_0000; // minterm 5
     3'b110: dec_out = 8'b0100_0000; // minterm 6
     3'b111: dec_out = 8'b1000_0000;
     default: dec_out = 8'b0000_0000;
  endcase
  end
 endmodule
```

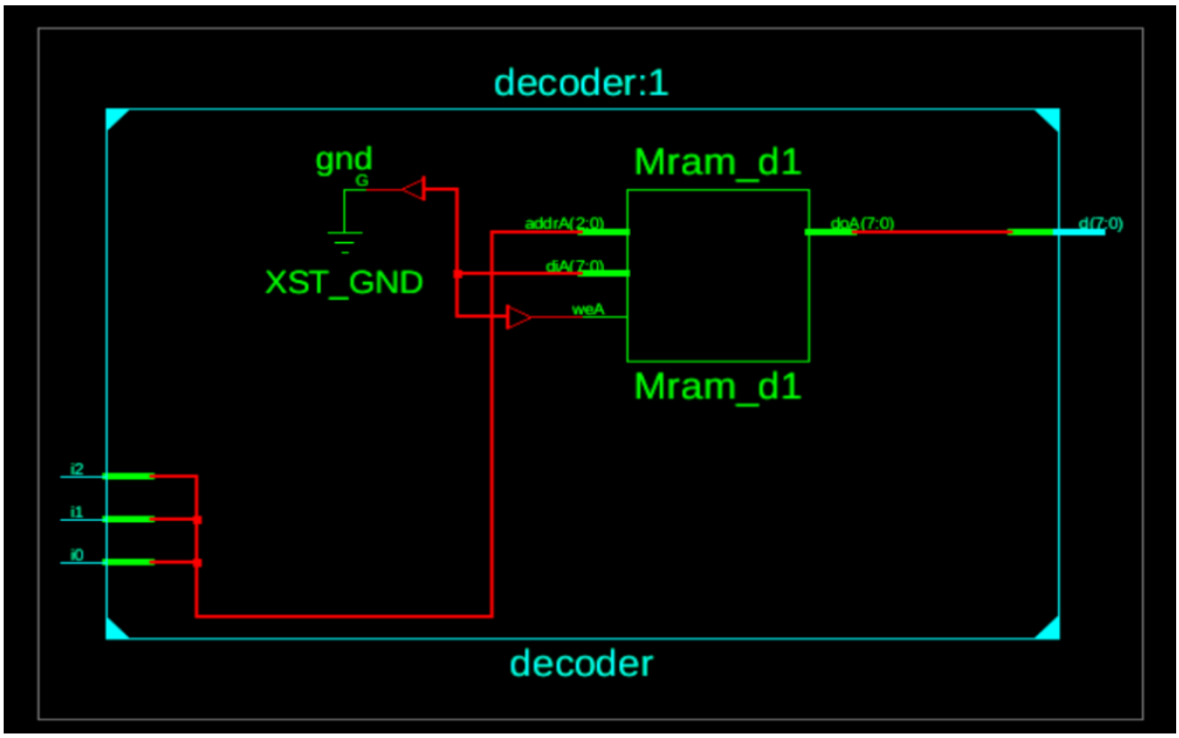
Simulation Testbench

```
module testbench_boolean_function;
reg A, B, C;
 wire F_out;
// Instantiate the top-level module
 boolean_function_decoder DUT_function (
   .A(A),
   .B(B),
   .C(C),
   .F_out(F_out));
 // Stimulus generation
  initial begin
  // Initialize inputs
  A = 0; B = 0; C = 0;
  // Apply all 8 input combinations with delays
  #10 A = 0; B = 0; C = 1; // m1
  #10 A = 0; B = 1; C = 0; // m2
  #10 A = 0; B = 1; C = 1; // m3
  #10 A = 1; B = 0; C = 0; // m4
  #10 A = 1; B = 0; C = 1; // m5
  #10 A = 1; B = 1; C = 0; // m6
  #10 A = 1; B = 1; C = 1; // m7
  #10 $finish; // Terminate simulation
    end
// Monitor outputs
 initial begin
  $monitor("Time=%t | A=%b, B=%b, C=%b | F_out=%b", $time, A, B, C, F_out);
  $dumpfile("waveform.vcd");
  $dumpvars(0, testbench_boolean_function);
  end
 endmodule
```

CIRCUIT DIAGRAM:



RTL SCHEMATIC:



OUTPUT WAVEFORM:

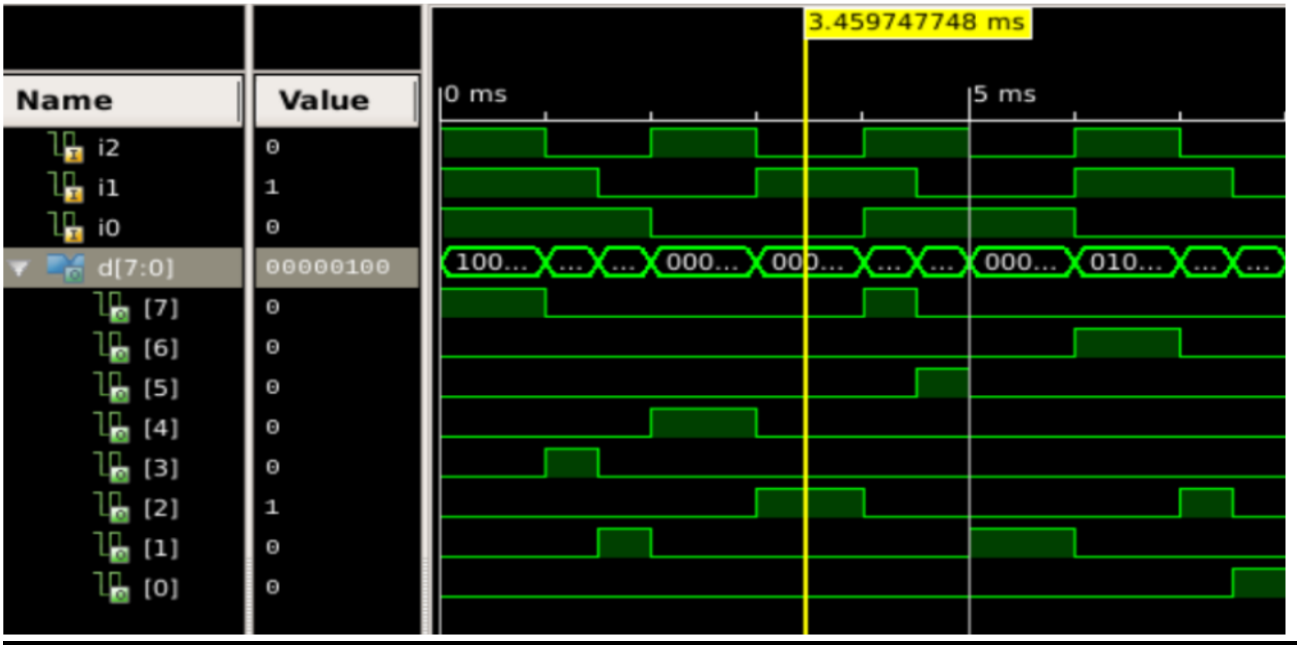


Fig: simulation output wave form of decoder

RESULT:

Hence the Boolean function using a decoder has been successfully designed, simulated, and synthesized using ISE Simulator.

CONCLUSION:

End of the Experiment student's can able to design and simulate Boolean function using a decoder using Cadence / Xilinx Vivado.

Viva questions

S.No	Question	CO	Bloom's Taxonomy
1	What is a decoder in digital electronics?	CO1	Remember
2	What is the purpose of using a decoder in Boolean function implementation?	CO2	Understand
3	What is meant by minterms in Boolean algebra?	CO1	Remember
4	How does a decoder generate minterms?	CO2	Understand
5	What is Sum of Minterms (SOP) form?	CO2	Understand
6	How do you implement a Boolean function using a decoder?	CO3	Apply
7	What is behavioural modelling in Verilog?	CO1	Remember
8	Which constructs are used in behavioural modelling?	CO2	Understand
9	How do you represent a decoder in behavioural Verilog?	CO3	Apply
10	What is the role of the case statement in decoder design?	CO2	Understand
11	How do you select required minterms from decoder outputs?	CO3	Apply
12	What is the purpose of OR operation in decoder-based implementation?	CO2	Understand
13	How many outputs does an n-input decoder have?	CO1	Remember
14	What is the advantage of using a decoder for logic implementation?	CO2	Understand
15	What are the limitations of decoder-based design?	CO4	Analyze
16	How do you verify the Boolean function using simulation?	CO3	Apply
17	What happens if an incorrect minterm is selected?	CO4	Analyze
18	How can a decoder be used to implement multiple functions?	CO3	Apply
19	What is the difference between combinational logic design and decoder-based design?	CO4	Analyze
20	How can you optimize a decoder-based implementation?	CO5	Evaluate

Experiment No: 5

UNIVERSAL N-BIT SHIFT REGISTER

AIM: To design and simulate a universal n-bit shift register (left, right, hold, parallel load) using behavioral modelling.

TOOL: Cadence ISE Simulator (Verilog).

THEORY:

A shift register is a sequential logic circuit used for storing and shifting data. It consists of flip-flops connected in series, where the output of one flip-flop is connected to the input of the next.A Universal Shift Register can perform multiple operations:

- Hold (No change)
- Shift Left
- Shift Right
- Parallel Load

The operation is controlled using select lines. For an n-bit register, it can store n bits of data and perform operations synchronously with the clock signal.

PROGRAM:

```
module universal_shift #(parameter n = 4) (
input clk,
input rst,
input [1:0] sel, // 00-Hold, 01-Shift Right, 10-Shift Left, 11-Parallel Load
input [n-1:0] d, // Parallel input
input sin_left, // Serial input for left shift
input sin_right, // Serial input for right shift
output reg [n-1:0] q);
always @(posedge clk or posedge rst)
    begin
    if (rst)
    q <= 0;
    else
```

```
  begin
  case (sel)
  2'b00: q <= q; // Hold
  2'b01: q <= {sin_right, q[n-1:1]}; // Shift Right
  2'b10: q <= {q[n-2:0], sin_left}; // Shift Left
  2'b11: q <= d; // Parallel Load
  endcase
  end
  end
  endmodule
```

TESTBENCH CODE:

```
module tb_universal_shift;
  parameter n = 4;
  reg clk, rst;
  reg [1:0] sel;
  reg [n-1:0] d;
  reg sin_left, sin_right;
  wire [n-1:0] q;
  // Instantiate DUT
  universal_shift #(n) uut (
  .clk(clk),
  .rst(rst),
  .sel(sel),
  .d(d),
  .sin_left(sin_left),
  .sin_right(sin_right),
  .q(q)
  );
  // Clock generation
  always #5 clk = ~clk;
  initial
  begin
  clk = 0;
  rst = 1;
  sel = 2'b00;
  d = 4'b0000;
```

```
sin_left = 0;
sin_right = 0;
#10 rst = 0;
// Parallel Load
sel = 2'b11; d = 4'b1010;
#10;
// Hold
sel = 2'b00;
#10;
// Shift Right
sel = 2'b01; sin_right = 1;
#20;
// Shift Left
sel = 2'b10; sin_left = 1;
#20;
// Another Parallel Load
sel = 2'b11; d = 4'b1100;
#10;
$stop;
end
endmodule
```

TRUTH TABLE:

Mode Control		Register operation
S1	S0	
0	0	No change
0	1	Shift left
1	0	Shift right
1	1	Parallel load

CIRCUIT DIAGRAM:

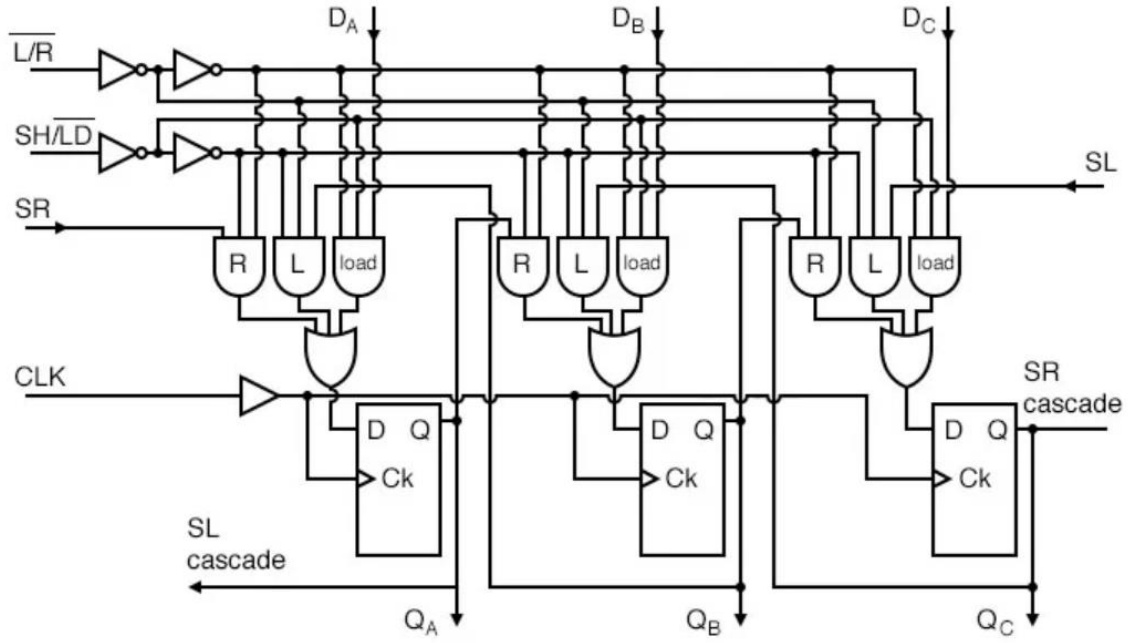


Fig : Circuit Diagram of universal shift left/right/parallel load

RTL SCHEMATIC:

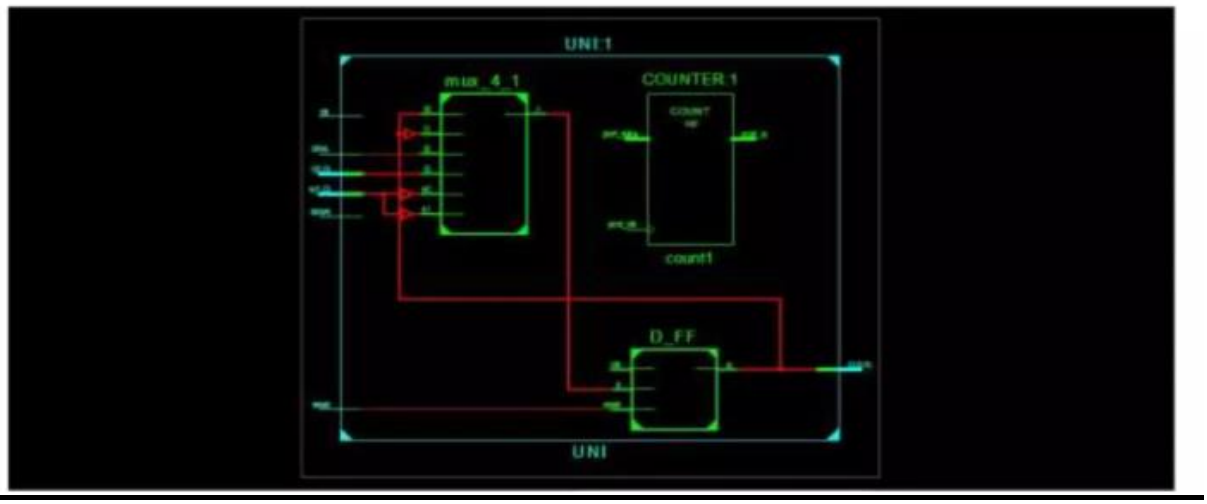


Fig : RTL schematic of universal shift left/right/parallel load

OUTPUT WAVEFORM:

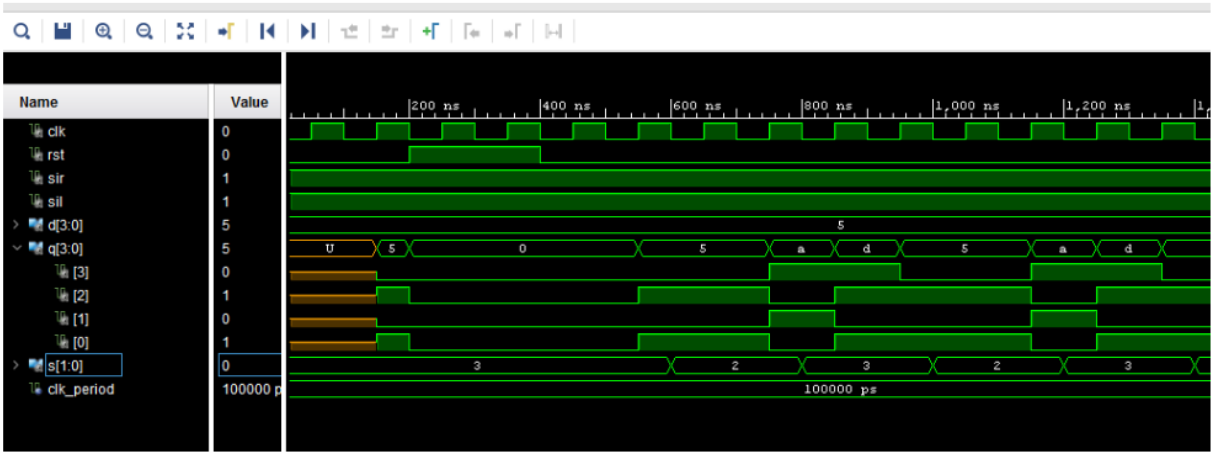


Fig: Output Waveform of Universal n-bit Shift Register

RESULT:

Hence the Universal n-bit Shift Register has been successfully designed, simulated, and synthesized using ISE Simulator.

CONCLUSION:

End of the Experiment student's can able to design and simulate Universal n-bit Shift Register using Cadence / Xilinx Vivado.

VIVA QUESTIONS

S.No	Question	CO	Bloom's Taxonomy
1	What is a shift register?	CO1	Remember
2	What is meant by a universal shift register?	CO1	Remember
3	What are the different modes of operation in a universal shift register?	CO2	Understand
4	What is left shift operation?	CO1	Remember
5	What is right shift operation?	CO1	Remember
6	What is meant by hold condition in a shift register?	CO2	Understand
7	What is parallel loading in a shift register?	CO2	Understand
8	How many control signals are required for a universal shift register?	CO3	Apply
9	What is the role of clock in shift registers?	CO1	Understand
10	What is behavioural modelling in Verilog?	CO1	Remember
11	Which Verilog construct is used to implement shift registers?	CO2	Understand
12	How do you implement left shift in Verilog?	CO3	Apply
13	How do you implement right shift in Verilog?	CO3	Apply
14	How is parallel load implemented in Verilog?	CO3	Apply
15	What happens if no operation is selected?	CO4	Analyze
16	What is the difference between serial input and parallel input?	CO2	Understand
17	How do you design an n-bit shift register?	CO3	Apply
18	What are the applications of shift registers?	CO2	Understand
19	How do you verify a shift register using simulation?	CO3	Apply
20	How can you optimize a shift register design?	CO5	Evaluate

Experiment No: 6
MOD-n COUNTER

AIM: To design and a synchronous MOD-n counter using behavioral modelling with D or JK flip-flops.

TOOL: Cadence ISE Simulator (Verilog).

THEORY:

Binary a counter's are most important and widely used digital circuits. A counter is circuit that counts the number of occurrences of an input. Each count a binary number is called a state of the counter. Hence a counter counting in terms of N bits as 2^N different states, the number of different states of a counter is also known as modulus of the counter. Thus, an N bit counter is a modulo 2^N counter. Counter circuits are primarily constituted of Flip Flops.

PROGRAM:

```
module mod6_counter (
input clk, input rst,
output reg [2:0] q // 3 bits required for 6 states (2^3 = 8));
always @(posedge clk or posedge rst) begin
      if (rst) begin
         q <= 3'b000; // Asynchronous Reset
      end else begin
         if (q == 3'b101) // If count is 5 (MOD-6), reset
            q <= 3'b000;
         else
            q <= q + 1'b1; // Increment
      end
   end
endmodule
```

Simulation Testbench

```
module tb_mod6_counter;
   reg clk;
   reg rst;
   wire [2:0] q;
   // Instantiate the Unit Under Test (UUT)
   mod6_counter uut (
      .clk(clk),
      .rst(rst),
      .q(q));
```

```
    // Clock generation (10ns period)
    always #5 clk = ~clk;

    initial begin
        // Initialize Inputs
        clk = 0;
        rst = 1;

        // Wait for global reset
        #15;
        rst = 0;
        // Run simulation for 100ns
        #100;

        $finish;
      end
    initial begin
        $monitor("Time=%0t | Reset=%b | Count=%d", $time, rst, q);
    end
endmodule
```

CIRCUIT DIAGRAM:

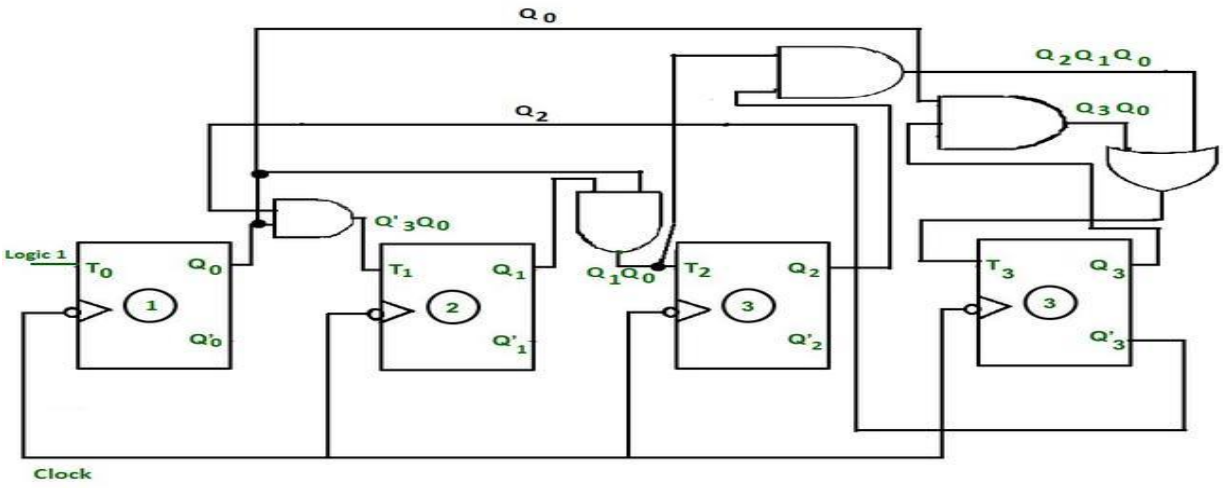


Fig : Circuit diagram of MOD-n Up-Down Counter

OUTPUT WAVEFORM:

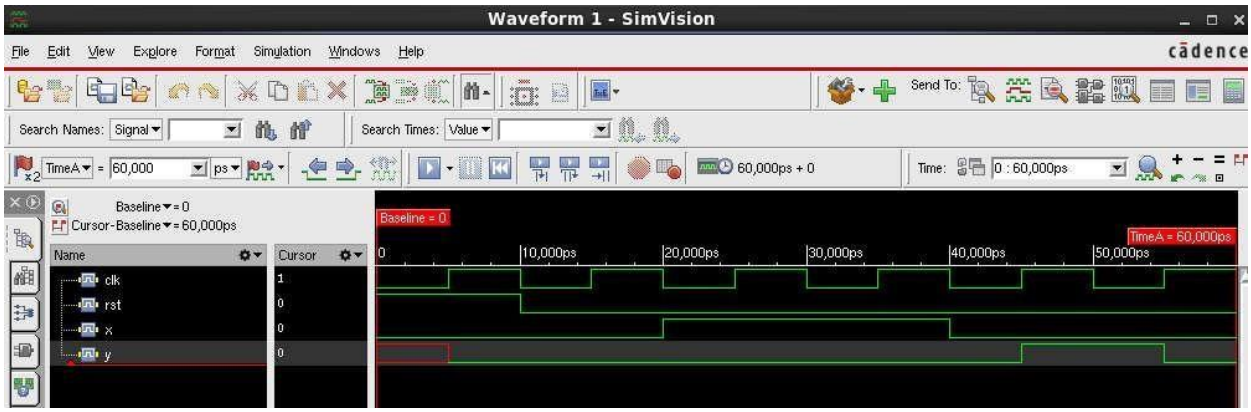


Fig : Output waveform of MOD-n Up-Down Counter

RTL SCHEMATIC:

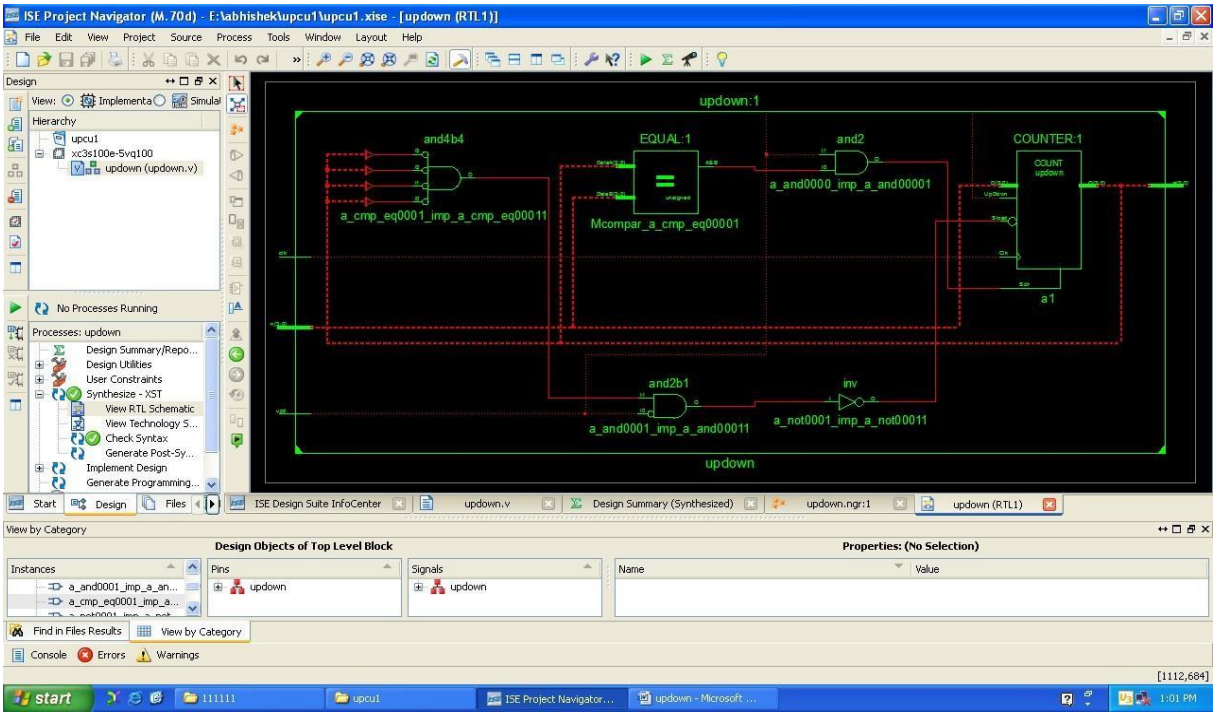


Fig: RTL schematic of MOD-n Up-Down Counter

RESULT:

Hence all the counter modules have been synthesized and simulated using ISE Simulator.

CONCLUSION:

End of the Experiment student's can able to realize the source code for MOD-n counters (Synchronous/Asynchronous reset)

VIVA QUESTIONS

S.No	Question	CO	Bloom's Taxonomy
1	What is a synchronous counter?	CO1	Remember
2	What is meant by MOD-n counter?	CO1	Remember
3	How is a synchronous counter different from an asynchronous counter?	CO2	Understand
4	Why are synchronous counters faster than ripple counters?	CO2	Understand
5	What determines the value of 'n' in a MOD-n counter?	CO2	Understand
6	How many flip-flops are required for a MOD-n counter?	CO3	Apply
7	What is the formula to calculate number of flip-flops?	CO3	Apply
8	What is behavioural modelling in Verilog?	CO1	Remember
9	Which construct is used to model synchronous counters in Verilog?	CO2	Understand
10	What is the role of clock in synchronous counters?	CO1	Understand
11	What is the difference between D and JK flip-flops?	CO2	Understand
12	Why are D flip-flops commonly used in synchronous counters?	CO2	Understand
13	How do you design a MOD-10 counter using D flip-flops?	CO3	Apply
14	What is state transition in counters?	CO2	Understand
15	What is the purpose of reset in a counter?	CO2	Understand
16	What is synchronous reset vs asynchronous reset?	CO2	Understand
17	How do you handle unused states in a MOD-n counter?	CO4	Analyze
18	What happens if the counter enters an invalid state?	CO4	Analyze
19	How do you verify a synchronous counter design?	CO3	Apply
20	How can you optimize a MOD-n counter design?	CO5	Evaluate

Experiment No: 7
RIPPLE COUNTER(Asynchronous)

AIM: To design and simulate an asynchronous (ripple) counter for a custom sequence using structural modelling.

TOOL: Cadence ISE Simulator (Verilog).

THEORY:

A ripple counter is an asynchronous counter made by cascading flip-flops, where the output of one flip-flop drives the clock of the next. Only the first flip-flop receives the external clock; others are triggered by the previous stage's output, causing a "ripple" effect.

- Asynchronous operation.
- Flip-flops operate in toggle mode.
- Only one flip-flop receives the external clock (LSB).
- Can be configured as up, down, or up/down counter.

An n-bit ripple counter has 2^n states and is also called a MOD-n counter. Counting sequences repeat after reaching the maximum or minimum count:

- Up counter: 000 → 001 → ... → 111 → 000...
- Down counter: 111 → 110 → ... → 000 → 111...

CIRCUIT DIAGRAM:

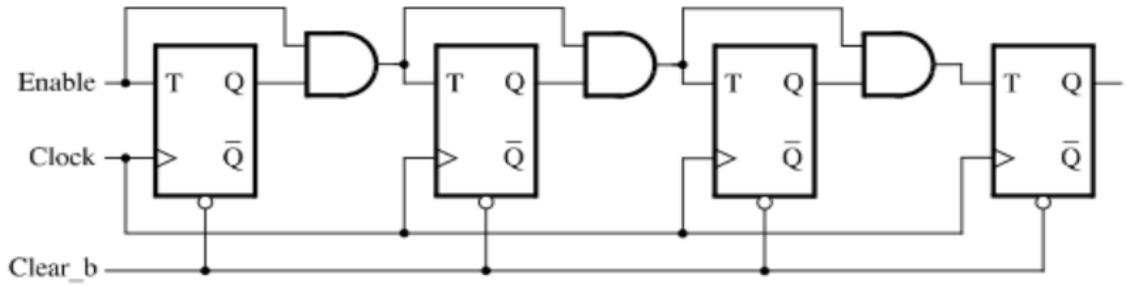


Fig: Asynchronous ripple counter circuit diagram

Program:

1. T Flip-Flop Module (Structural component)

```
module tff (
    input t,
    input clk,
    input reset,
    output reg q);
    always @(posedge clk or posedge reset) begin
        if (reset)
            q <= 1'b0;
        else if (t)
            q <= ~q;
    end
endmodule
```

2. Custom Sequence Counter (Top Module)

```
module custom_async_counter (
    input clk,
    input reset,
    output [2:0] q);

    wire t_const = 1'b1; // T is always high for toggling

    assign q[0] = 1'b0;

    tff tff1 (t_const, clk, reset, q[1]);

    tff tff2 (t_const, q[1], reset, q[2]);
 endmodule
```

Simulation Testbench

```
`timescale 1ns / 1ps
module tb_custom_async_counter;
    reg clk;
    reg reset;
    wire [2:0] q;
    // Instantiate the Unit Under Test (UUT)
    custom_async_counter uut (
        .clk(clk),
        .reset(reset),
        .q(q));
```

```
    // Clock generation
    initial begin
        clk = 0;
        forever #5 clk = ~clk; // 10ns period
    end
    // Test sequence
    initial begin
        // Initialize
        reset = 1;
        #10;
        reset = 0;
        // Run for 100 ns to observe sequence 0-2-4-6-0
        #100;
        $finish;
    end
    initial begin
        $monitor("Time=%0t | Reset=%b | Q=%b (Decimal: %d)", $time, reset, q, q);
    end
endmodule
```

TRUTH TABLE:

Clock	Q_C	Q_B	Q_A	Output of Reset Logic Y
0	0	0	0	1
1	0	0	1	1
2	0	1	0	1
3	0	1	1	1
4	1	0	0	1
5	1	0	1	1
6	1	1	0	0
7	1	1	1	0

TECHNOLOGICL SCHEMATIC:

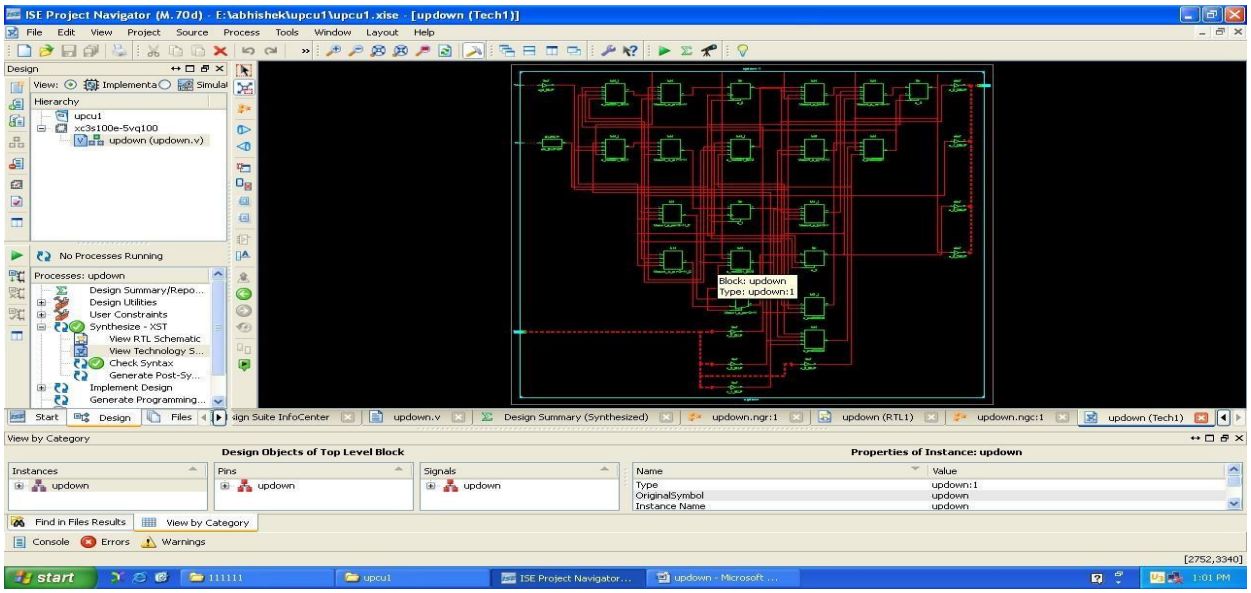


Fig: Technological schematic of ripple Counter

OUTPUT WAVEFORM:

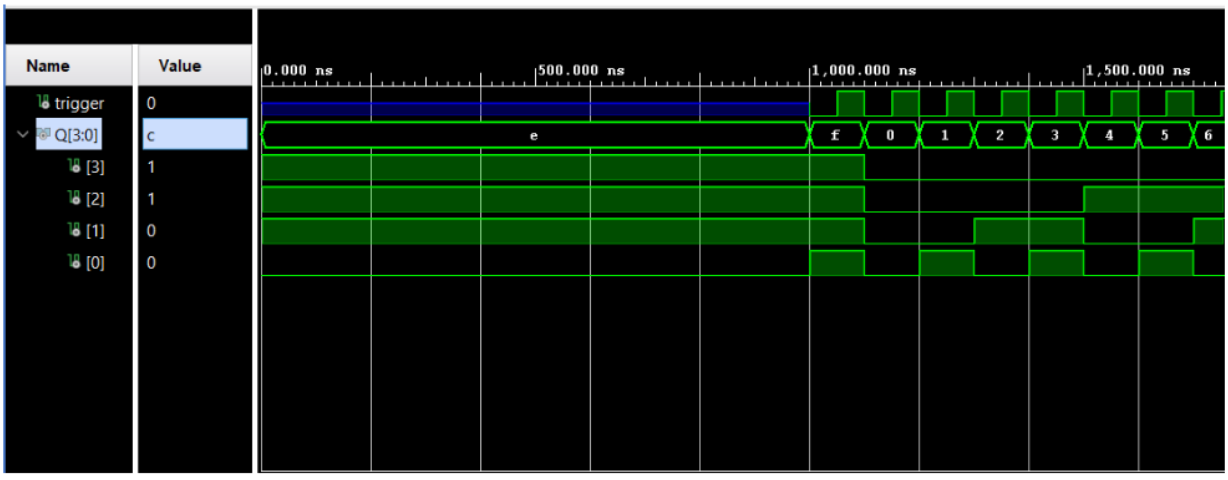


Fig: Output waveform of ripple Counter

RESULT:

Hence all the counter modules have been synthesized and simulated using ISE Simulator.

CONCLUSION:

End of the Experiment student's can able to realize the source code for ripple counters (Asynchronous reset).

VIVA QUESTIONS

S.No	Question	CO	Blooms Taxonomy
1	What is an asynchronous (ripple) counter?	CO1	Remember
2	Why is it called a ripple counter?	CO2	Understand
3	What is the difference between synchronous and asynchronous counters?	CO2	Understand
4	What are the main components used in a ripple counter?	CO1	Remember
5	What is structural modeling in Verilog?	CO1	Remember
6	How is structural modeling different from behavioral modeling?	CO2	Understand
7	What type of flip-flops are commonly used in ripple counters?	CO1	Remember
8	How does the clock propagate in an asynchronous counter?	CO2	Understand
9	What is propagation delay in ripple counters?	CO2	Understand
10	Why are ripple counters slower than synchronous counters?	CO3	Apply
11	What is meant by a custom sequence counter?	CO1	Remember
12	How do you design a counter for a non-binary sequence?	CO3	Apply
13	What is the role of reset logic in custom counters?	CO2	Understand
14	How do you detect invalid states in a custom sequence?	CO4	Analyze
15	What happens if invalid states are not handled?	CO4	Analyze
16	How do you implement reset logic using gates?	CO3	Apply
17	How are flip-flops interconnected in a ripple counter?	CO2	Understand
18	How do you verify the counter using simulation?	CO3	Apply
19	What are the advantages of ripple counters?	CO2	Understand
20	What are the limitations of ripple counters?	CO4	Analyze

Experiment No:8

FINITE STATE MACHINE

AIM: To design and Implement a sequence detector for a given binary pattern using FSM (Moore/Mealy) in behavioural modelling.

TOOL: Cadence ISE Simulator (Verilog).

THEORY:

The Finite State Machine is an abstract mathematical model of a sequential logic function. It has finite inputs, outputs and number of states. FSMs are implemented in real-life circuits through the use of Flip Flops. The implementation procedure needs a specific order of steps (algorithm), in order to be carried out. A Sequential Logic function has a "memory" feature and takes into account past inputs in order to decide on the output. The Finite State Machine is an abstract mathematical model of a sequential logic function. It has finite inputs, outputs and number of states. FSMs are implemented in real-life circuits through the use of Flip Flops. The implementation procedure needs a specific order of steps (algorithm), in order to be carried out.

PROGRAM:

```
module Sequence_Detector_MOORE_Verilog(sequence_in,clock,reset,detector_out );
input clock; // clock signal
input reset; // reset input
input sequence_in; // binary input
output reg detector_out; // output of the sequence detector
parameter Zero=3'b000, // "Zero"
  State One=3'b001, // "One"
  State OneZero=3'b011, //
  "OneZero" State
  OneZeroOne=3'b010, // "OnceZeroOne"
  State OneZeroOneOne=3'b110;//
  "OneZeroOneOne" State
reg [2:0] current_state, next_state; // current state and next state
// sequential memory of the
Moore FSM always @(posedge
```

```
 clock, posedge reset) begin
  if(reset==1)
  current_state <= Zero;// when reset=1, reset the state of the FSM to "Zero" State
  else
   current_state <= next_state; // otherwise, next state
 // combinational logic of the Moore FSM
 // to determine next state
always @(current_state,sequence_in)
  begin
case(current_state)
Zero:begin
if(sequence_in==1)
 next_state = One;
else
next_state = Zero;
end
One:begin
if(sequence_in==0)
 next_state = OneZero;
   else
next_state = One;
end
 OneZero:begin
if(sequence_in==0)
 next_state = Zero;
else
next_state = OneZeroOne;
end
OneZeroOne:begin
 if(sequence_in==0)
next_state = OneZero;
else
next_state = OneZeroOneOne;
OneZeroOneOne:begin
if(sequence_in==0) next_state = OneZero; else
next_state = One;
end
default:next_state = Zero;
  endcase
  end
 // combinational logic to determine the output
 // of the Moore FSM, output only depends on current state
```

```
 always @(current_state)

 begin
case(current_state)
Zero: detector_out = 0;
One:  detector_out = 0;
OneZero: detector_out = 0;
OneZeroOne: detector_out = 0;
OneZeroOneOne: detector_out = 1;
default: detector_out = 0;
endcase
end
endmodule
```

CIRCUIT DIAGRAM:

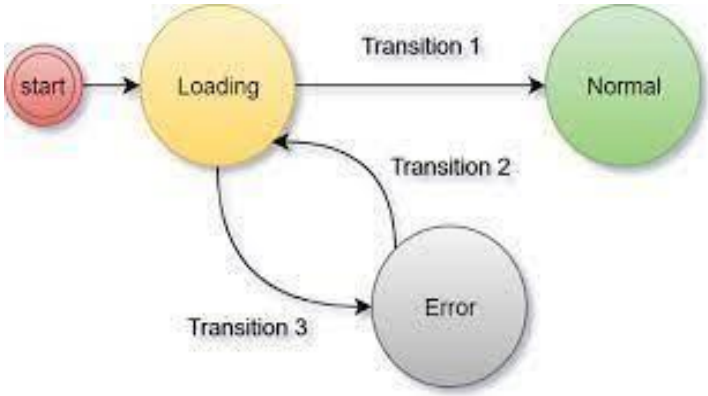


Fig : State diagram of a FSM

TRUTH TABLE:

Current State		Input	Next State		Outputs
A	B	I	A_{next}	B_{next}	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	0	1
1	0	0	0	0	0
1	0	1	1	0	0
1	1	0	X	X	X
1	1	1	X	X	X

OUTPUT WAVEFORM:

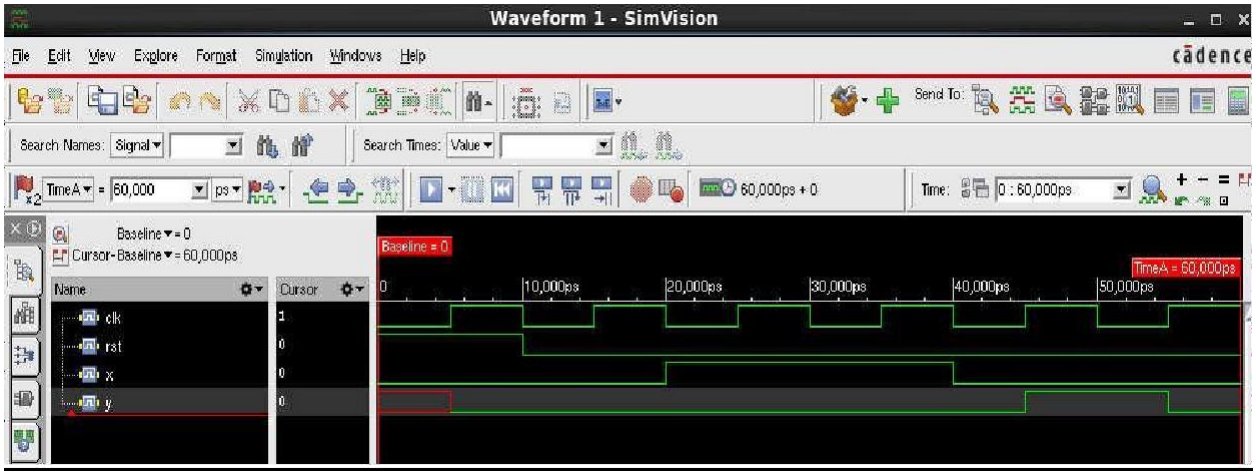


Fig: Output waveform of FSM

RESULT:

Hence finite state Machine has been synthesized and simulated using ISE Simulator.

CONCLUSION:

End of the Experiment student's can able to design and simulate finite state Machine using Cadence / Xilinx Vivado.

VIVA QUESTIONS

S.No	Question	CO	Bloom's Level
1	What is a sequence detector?	CO1	Remember
2	What is a Finite State Machine (FSM)?	CO1	Remember
3	What are the types of FSM?	CO1	Remember
4	Define Moore machine.	CO1	Remember
5	Define Mealy machine.	CO1	Remember
6	What is the main difference between Moore and Mealy FSM?	CO2	Understand
7	Which FSM gives faster output response and why?	CO2	Understand
8	What is a state in FSM?	CO1	Remember
9	What is a state transition diagram?	CO2	Understand
10	What is a state table?	CO2	Understand
11	What is meant by overlapping sequence detection?	CO3	Apply
12	What is non-overlapping sequence detection?	CO3	Apply
13	How do you design a sequence detector for pattern "101"?	CO3	Apply
14	How many states are required for detecting sequence "1101"?	CO3	Apply
15	Why does Moore machine require more states than Mealy?	CO4	Analyze
16	What is state encoding?	CO2	Understand
17	What are different state encoding techniques?	CO2	Understand
18	What is behavioural modelling in Verilog?	CO1	Remember
19	Which Verilog construct is used for FSM implementation?	CO2	Understand
20	What is the role of always block in FSM design?	CO2	Understand