# SRM VALLIAMMAI ENGINEERING COLLEGE

## (An Autonomous Institution)

SRM Nagar, Kattankulathur-603203

## DEPARTMENT OF INFORMATION TECHNOLOGY

ACADEMIC YEAR: 2020-2021

ODD SEMESTER

# LAB MANUAL

**(REGULATION - 2019)**

# 1908306-DIGITAL SYSTEMS AND COMMUNICATION LABORATORY

THIRD SEMESTER

B.Tech - Information Technology

**Prepared by**

Ms.S.Shenbagavadivu

Mr.R.Sankaranarayanan

# PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

1. To afford the necessary background in the field of Information Technology to deal with engineering problems to excel as engineering professionals in industries.
2. To improve the qualities like creativity, leadership, teamwork and skill thus contributing towards the growth and development of society.
3. To develop ability among students towards innovation and entrepreneurship that caters to the needs of Industry and society.
4. To inculcate and attitude for life-long learning process through the use of information technology sources.
5. To prepare then to be innovative and ethical leaders, both in their chosen profession and in other activities.

# PROGRAMME OUTCOMES (POs)

After going through the four years of study, Information Technology Graduates will exhibit ability to:

| PO# | Graduate Attribute | Programme Outcome |
|---|---|---|
| 1 | Engineering knowledge | Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization for the solution of complex engineering problems. |
| 2 | Problem analysis | Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences. |
| 3 | Design/development of solutions | Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for public health and safety, and cultural, societal, and environmental considerations. |
| 4 | Conduct investigations of complex problems | Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to |

| | | |
|---|---|---|
| | | provide valid conclusions |
| 5 | Modern tool usage | Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools, including prediction and modeling to complex engineering activities, with an understanding of the limitations. |
| 6 | The engineer and society | Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities relevant to the professional engineering practice |
| 7 | Environment and sustainability | Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development. |
| 8 | Ethics | Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice |
| 9 | Individual and team work | Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings |
| 10 | Communication | Communicate effectively on complex engineering activities with the engineering community and with the society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions |
| 11 | Project management and finance | Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments |
| 12 | Life-long learning | Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change |

# PROGRAM SPECIFIC OUTCOMES (PSOs)

By the completion of Information Technology program the student will have following Program specific outcomes

1. Design secured database applications involving planning, development and maintenance using state of the art methodologies based on ethical values.
2. Design and develop solutions for modern business environments coherent with the advanced technologies and tools.
3. Design, plan and setting up the network that is helpful for contemporary business environments using latest hardware components.
4. Planning and defining test activities by preparing test cases that can predict and correct errors ensuring a socially transformed product catering all technological needs.

## 908306 DIGITAL SYSTEMS AND COMMUNICATION LABORATORY

L T P C
0 0 2 1

**OBJECTIVES:**

- Understanding the concept of basic gates.
- Develop a circuit for combinational and sequential circuits.
- Get a deep understanding on analog modulation technques.
- Learn the concept of digital modulation techniques.
- Understand the working principle of HDL.
- Design and develop basic adigital and communication circuits.

**LIST OF EXPERIMENTS**

1. Verification of Boolean Theorems using basic gates.
2. Design and implement Half/Full Adder and Subtractor.
3. Design and implement combinational circuits using MSI devices- Parity generator / checker
4. Design and implement shift-registers.
5. Design and implement synchronous counters.
6. Amplitude modulation and demodulation
7. Frequency modulation and demodulation
8. Pulse code modulation and demodulation.
9. FSK, PSK and DPSK schemes (Simulation)
10. Coding combinational circuits using HDL.
11. Coding sequential circuits using HDL.

**TOTAL: 45 PERIODS**

## COURSE OUTCOMES

| | |
|---|---|
| 1908306.1 | Design a basic combinational circuit applications. |
| 1908306.2 | Design a basic analog modulation circuits AM,FM. |
| 1908306.3 | Simukate a FSK,PSK circuits in HDL. |
| 1908306.4 | Construct a pulse code modulation circuits. |
| 1908306.5 | Code and Simulate a combinational and sequential circuits |

## CO-PO MATRIX

| CO | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1908306.1 | 3 | 2 | 3 | - | 2 | - | - | - | - | - | - | - |
| 1908306.2 | 2 | 3 | 3 | - | 2 | - | - | - | - | - | - | - |
| 1908306.3 | 3 | 3 | 3 | - | 2 | - | - | - | - | - | - | - |
| 1908306.4 | 3 | 3 | 3 | - | 2 | - | - | - | - | - | - | - |
| 1908306.5 | 2 | 2 | 2 | - | 2 | - | - | - | - | - | - | - |
| 1908306 | 3 | 3 | 3 | - | 2 | - | - | - | - | - | - | - |

## CO-PSO MATRIX

| COURSE | PSO 1 |
|---|---|
| 1908306.1 | **3** |
| 1908306.2 | **3** |
| 1908306.3 | **3** |
| 1908306.4 | **2** |
| 1908306.5 | **3** |
| 1908306 | **3** |

## MODE OF ASSEMENT

## EVALUATION PROCEDURE FOR EACH EXPERIMENTS

| S.No | Description | Mark |
|---|---|---|
| 1. | Aim & Pre-Lab discussion | 20 |
| 2. | Observation | 20 |
| 3. | Conduction and Execution | 30 |
| 4. | Output & Result | 10 |
| 5. | Viva | 20 |
| **Total** | | **100** |

## INTERNAL ASSESSMENT FOR LABORATORY

| S.No | Description | Mark |
|---|---|---|
| 1. | Conduction & Execution of Experiment | 25 |
| 2. | Record | 10 |
| 3. | Model Test | 15 |
| **Total** | | **50** |

# INTRODUCTION/ DESCRIPTION OF MAJOR SOFTWARE & HARDWARE INVOLVED IN LAB

## LIST OF EQUIPMENT FOR A BATCH OF 30 STUDENTS

### HARDWARE

- ✓ Digital trainer kits – 30 Nos.
- ✓ Kits for AM, FM and PCM
- ✓ Signal generator / Function generators / Power Supply / CRO / Bread Board each -15 Nos.
- ✓ Digital ICs required for the experiments in sufficient numbers.

### SOFTWARE

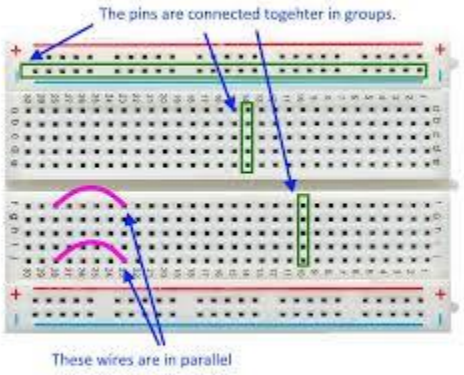
- ✓ HDL simulator.
- ✓ MATLAB / SCILAB for simulation experiments PCs - 10 Nos.

## OUTCOME

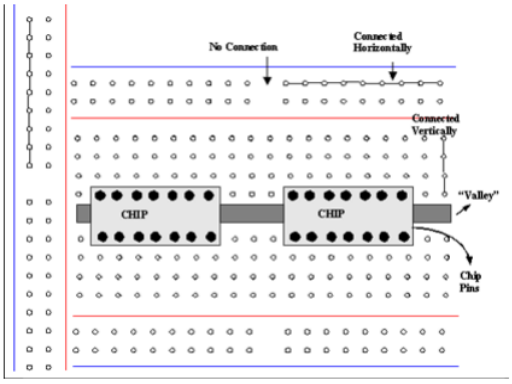
- ✓ Design a basic combinational circuit applications.
- ✓ Design a basic analog modulation circuits AM,FM.
- ✓ Simulate a FSK,PSK circuits in HDL.
- ✓ Construct a pulse code modulation circuits.
- ✓ Code and Simulate a combinational circuits and sequental circuits.

## BREADBOARD

An electronics breadboard is actually referring to a solder less breadboard. These are great units for making temporary circuits and prototyping, and they require absolutely no soldering. Prototyping is the process of testing out an idea by creating a preliminary model from which other forms are developed or copied, and it is one of the most common uses for breadboards

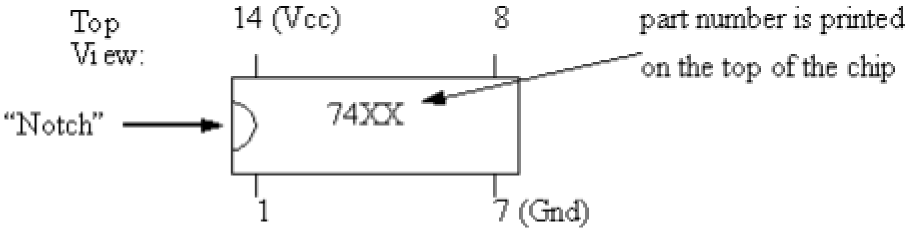


The pins are connected togehter in groups.
These wires are in parallel



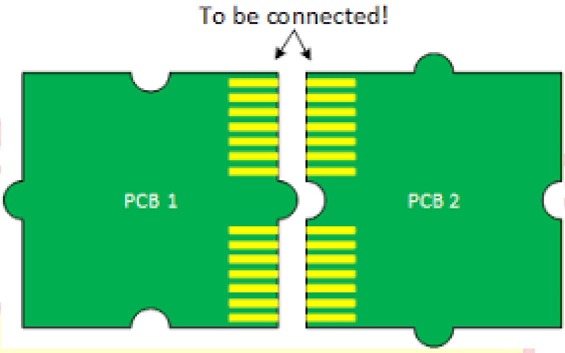
## Transistor-Transistor Logic (TTL)

TTL refers to the nature of the circuit of the digital logic gate; All of the TTL chips you will use are "Dual In-line Packages" or DIPs. Most of the packages are 14 pins, and the pins are number from looking at the chip from the top: Below the "notch" is pin 1 to pin 7, and above the notch is pin 14 down to 8. NOTE that Pin 14 must always be connected to VCC (+5V) and pin 7 to ground (0V)



## PRINTED CIRCUIT BOARD

PCBs can be single-sided (one copper layer), double-sided (two copper layers on both sides of one substrate layer), or multi-layer (outer and inner layers of copper, alternating with layers of substrate). A basic PCB consists of a flat sheet of insulating material and a layer of copper foil, laminated to the substrate. Chemical etching divides the copper into separate conducting lines called tracks or *circuit traces*, pads for connections, vias to pass connections between layers of copper, and features such as solid conductive areas for EM shielding or other purposes. The tracks function as wires fixed in place, and are insulated from each other by air and the board substrate material. The surface of a PCB may have a coating that protects the copper from corrosion and reduces the chances of solder shorts between traces or undesired electrical contact with stray bare wires. For its function in helping to prevent solder shorts, the coating is called solder resist.



## MODELSIM HDL SIMULATOR:

The original Modeltech (VHDL) simulator was the first mixed-language simulator capable of simulating VHDL and Verilog design entities together. In 2003, ModelSim 5.8 was the first simulator to begin supporting features of the Accellera SystemVerilog 3.0 standard.[1] In 2005 Mentor introduced Questa to provide high performance Verilog and SystemVerilog simulation and expand Verification capabilities to more advanced methodologies such as Assertion Based Verification and Functional Coverage. Today Questa is the leading high performance SystemVerilog and Mixed simulator supporting a full suite of methodologies including industry standard OVM and UVM. ModelSim is still the leading simulator for FPGA design.

## LANGUAGE SUPPORT FOR MODELSIM:

- VHDL
- Verilog
- Verilog 2001
- SystemVerilog
- PSL
- SystemC

## STUDY AND VERIFICATION OF BOOLEAN LOGIC GATES

**AIM:**

To draw a pin diagram and truth table for the basic logic gates and also verify its truth table

**THEORY:**

Circuit that takes the logical decision and the process are called logic gates. Each gate has one or more input and only one output. In digital system processing the Boolean function by using basic gates (such as OR, AND, NOT gates). All Basic gates form these universal gates. NAND, NOR and X-OR are known as universal gates.

- ❖ **OR GATE:** The OR gate performs a logical addition commonly known as OR function. The output is high when any one of the inputs is high. The output is low level when both the inputs are low.
- ❖ **AND GATE:** The AND gate performs a logical multiplication commonly known as AND function. The output is high when both the inputs are high. The output is low level when any one of the inputs is low.
- ❖ **NOT GATE:** The NOT gate is called an inverter. The output is high when the input is low. The output is low when the input is high.
- ❖ **NAND GATE:** The NAND gate is a contraction of AND-NOT. The output is high when both inputs are low and any one of the input is low .The output is low level when both inputs are high.
- ❖ **NOR GATE:** The NOR gate is a contraction of OR-NOT. The output is high when both inputs are low. The output is low when one or both inputs are high.
- ❖ **X-OR GATE:** The output is high when any one of the inputs is high. The output is low when both the inputs are low and both the inputs are high.

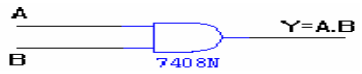
**APPARATUS REQUIRED:**

| SL No. | COMPONENT | SPECIFICATION | QTY |
|---|---|---|---|
| 1. | AND GATE | IC 7408 | 1 |
| 2. | OR GATE | IC 7432 | 1 |
| 3. | NOT GATE | IC 7404 | 1 |
| 4. | NAND GATE 2 I/P | IC 7400 | 1 |
| 5. | NOR GATE | IC 7402 | 1 |
| 6. | X-OR GATE | IC 7486 | 1 |
| 7. | NAND GATE 3 I/P | IC 7410 | 1 |
| 8. | IC TRAINER KIT | - | 1 |
| 9. | PATCH CORD | - | 14 |

## CIRCUIT & TRUTH TABLE:

## AND GATE

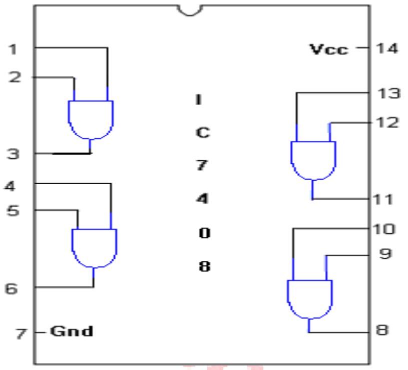
SYMBOL:



TRUTH TABLE

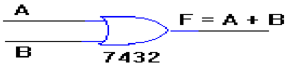
| A | B | A.B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

PIN DIAGRAM:



## OR GATE

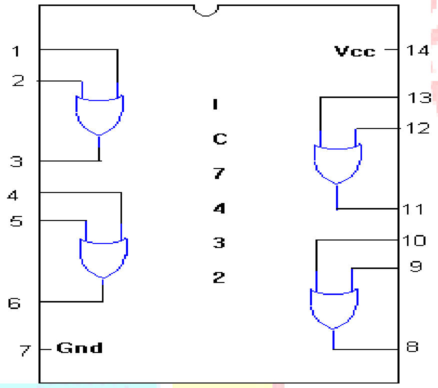
SYMBOL :



TRUTH TABLE

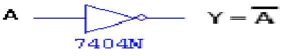
| A | B | A+B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

PIN DIAGRAM :



## NOT GATE

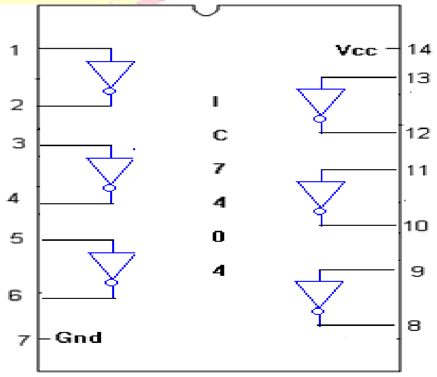
SYMBOL:



TRUTH TABLE :

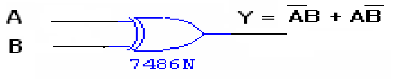
| A | $\overline{A}$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

PIN DIAGRAM:

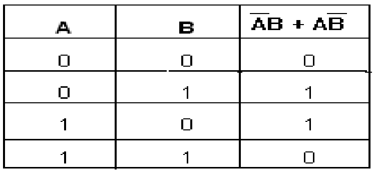


## XOR GATE

SYMBOL :

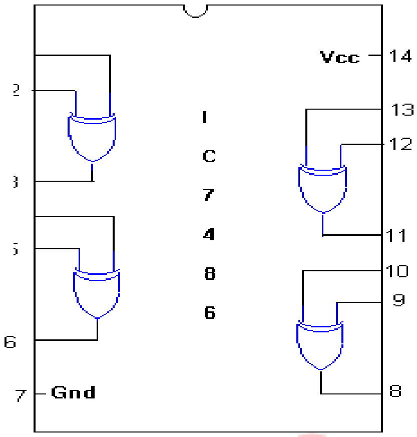


TRUTH TABLE :



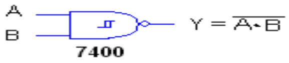
| A | B | $\overline{A}B + A\overline{B}$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

PIN DIAGRAM :

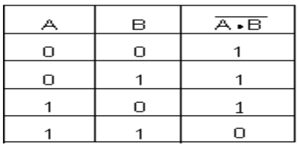


## 2INPUT NAND GATE

SYMBOL:

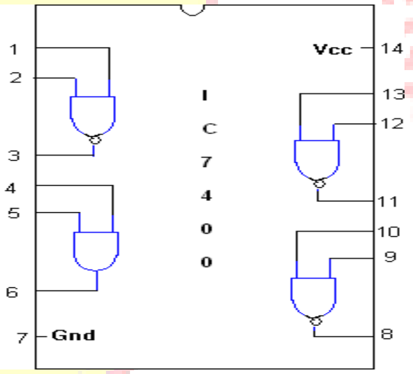


TRUTH TABLE



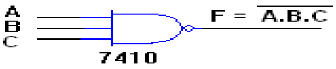
| A | B | $\overline{A.B}$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

PIN DIAGRAM:

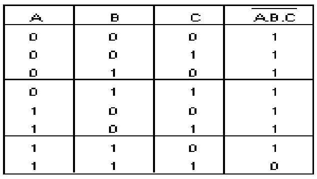


## 3INPUT NAND GATE

SYMBOL :

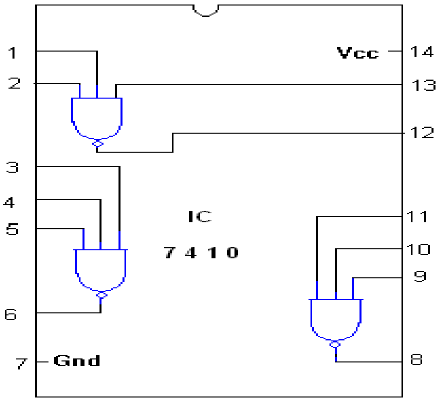


TRUTH TABLE



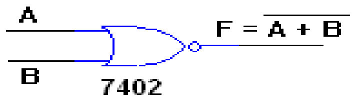
| A | B | C | $\overline{A.B.C}$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

PIN DIAGRAM :

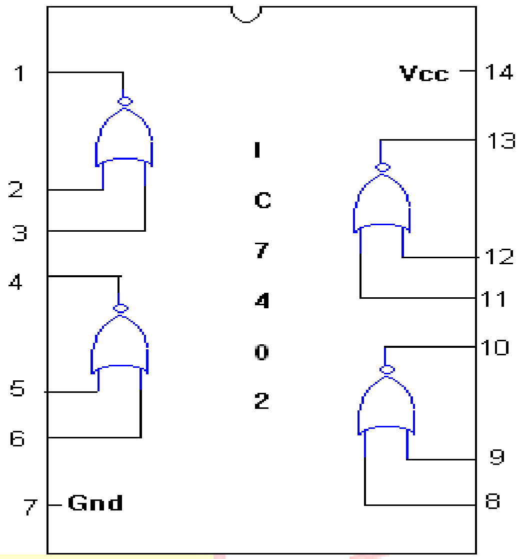


# NOR GATE

## SYMBOL :



## PIN DIAGRAM :



## TRUTH TABLE

| A | B | $\overline{A+B}$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**PROCEDURE:**

(i) Connections are given as per pin diagram along with GND & VCC

(ii) Logical inputs are given as per circuit diagram.

(iii) Observe the output and verify the truth table for each gate

**RESULT:**

Thus the logic gates and it's the truth table is verified successfully.

## EX.NO 1: Verification of Boolean Theorems using basic gates

### AIM

To verify the Boolean theorem using logic gates and it's the truth table

### THEORY

A set of rules or Laws of Boolean Algebra expressions have been invented to help reduce the number of logic gates needed to perform a particular logic operation resulting in a list of functions or theorems known commonly as the Laws of Boolean Algebra. Boolean Algebra is the mathematics we use to analyse digital gates and circuits.

### BOOLEAN LAWS

- ❖ The basic Laws of Boolean Algebra that relate to the Commutative Law allowing a change in position for addition and multiplication, the Associative Law allowing the removal of brackets for addition and multiplication, as well as the Distributive Law allowing the factoring of an expression, are the same as in ordinary algebra.
- ❖ The variables used in Boolean Algebra only have one of two possible values, a logic "0" and a logic "1" but an expression can have an infinite number of variables all labelled individually to represent inputs to the expression
- ❖ Boolean algebra is a mathematical system consisting of a set of two or more distinct elements, two binary operators denoted by the symbols (+) and (.) and one unary operator denoted by the symbol either bar (-) or prime ('). They satisfy the commutative, associative, distributive and absorption properties of the Boolean algebra.

**Commutative Property**

- ✓ Boolean addition is commutative, given by A+B=B+A
- ✓ Boolean algebra is also commutative over multiplication, given by A.B=B.A
- ✓ The associative property of addition is given by A+ (B+C) = (A+B) +C
- ✓ The associative law of multiplication is given by A. (B.C) = (A.B).C

Distributive Property

- ✓ The Boolean addition is distributive over Boolean multiplication,
  given by A+BC = (A+B) (A+C)
- ✓ Boolean multiplication is also distributive over Boolean addition
  given by A. (B+C) = A.B+A.C

**De Morgan's Therem**

- ✓ **First Theorem**

It states that the complement of a product is equal to the sum of the complements.

$(AB)' = A' + B'$

- ✓ Second Theorem

It states that the complement of a sum is equal to the product of the complements. $(A+B)' = A'.B'$

**Idempotent Law**

$X + X = X \; X \bullet X = X$

**Involution Law**

$(X')' = X$

**Absorption Law**

$X + XY = X \; X(X + Y) = X$

**Consensus theorem**

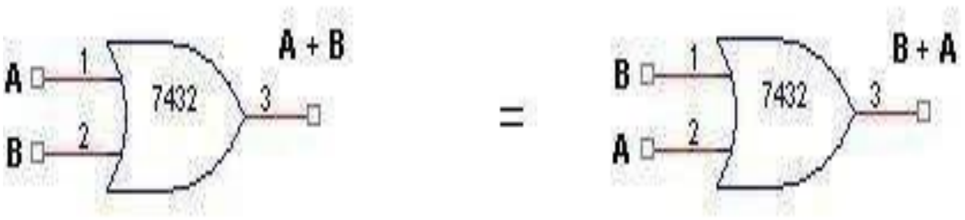
$XY + X'Z + YZ = XY + X'Z$

## APPARATUS REQUIRED:

| S.NO | COMPONENTS | RANGE | QUANTITY |
|---|---|---|---|
| 1 | Digital IC trainer kit | - | 1 |
| 2 | IC | 7400 | 1 |
| | | 7402 | 1 |
| | | 7404 | 1 |
| | | 7408 | 1 |
| | | 7432 | 1 |
| | | 7486 | 1 |
| 3 | Bread board | - | 1 |
| 4 | Connecting wires | - | As required |

## CIRCUIT DIAGRAM & TRUTH TABLE:

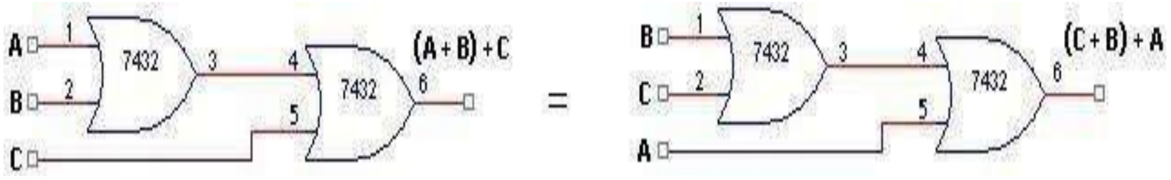
### a)Commutative Law



Truth Table

| Input | | Output | |
|---|---|---|---|
| A | B | A+B | B+A |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 |

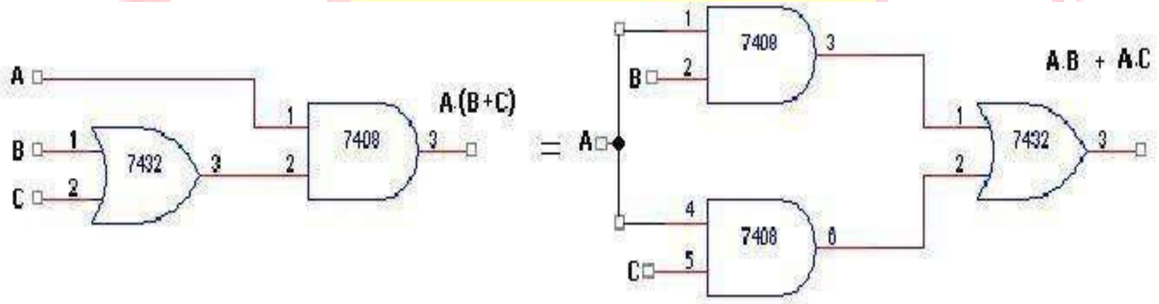
## b)Associative Law



### Truth Table

| Input | | | Output | | | |
|---|---|---|---|---|---|---|
| A | B | C | A+B | (A+B)+C | B+C | A+(B+C) |
| 0 | 0 | 0 | 0 | **0** | 0 | **0** |
| 0 | 0 | 1 | 0 | **1** | 1 | **1** |
| 0 | 1 | 0 | 1 | **1** | 1 | **1** |
| 0 | 1 | 1 | 1 | **1** | 1 | **1** |
| 1 | 0 | 0 | 1 | **1** | 0 | **1** |
| 1 | 0 | 1 | 1 | **1** | 1 | **1** |
| 1 | 1 | 0 | 1 | **1** | 1 | **1** |
| 1 | 1 | 1 | 1 | **1** | 1 | **1** |

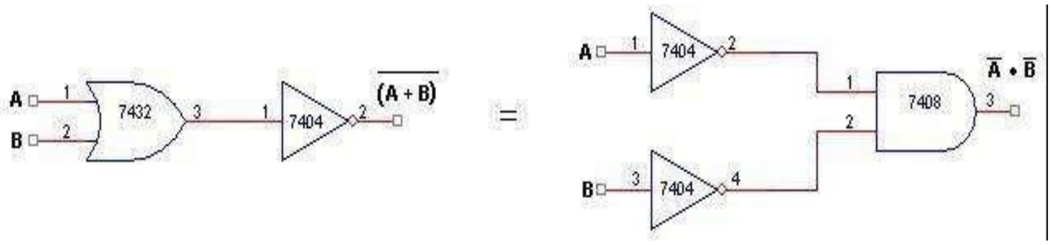
## c)Distributive Law



### Truth Table

| Input | | | Output | | | | |
|---|---|---|---|---|---|---|---|
| A | B | C | B+C | A.(B+C) | A.B | A.C | A.B+A. |
| 0 | 0 | 0 | 0 | **0** | 0 | 0 | **0** |
| 0 | 0 | 1 | 1 | **0** | 0 | 0 | **0** |
| 0 | 1 | 0 | 1 | **0** | 0 | 0 | **0** |
| 0 | 1 | 1 | 1 | **0** | 0 | 0 | **0** |
| 1 | 0 | 0 | 0 | **0** | 0 | 0 | **0** |
| 1 | 0 | 1 | 1 | **1** | 0 | 1 | **1** |
| 1 | 1 | 0 | 1 | **1** | 1 | 0 | **1** |
| 1 | 1 | 1 | 1 | **1** | 1 | 1 | **1** |

### d)De-Morgan's Theorem 1:



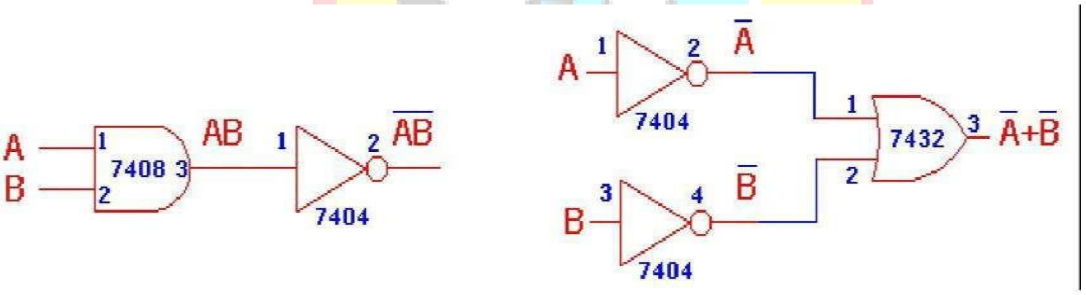
**Truth Table:**

| Input | | Output | |
|---|---|---|---|
| A | B | (A+B)' | A'. B' |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 |

1. Observation table for First Theorem of Demorgan

| Input | | Output | |
|---|---|---|---|
| A | B | LHS = $\overline{A + B}$ | RHS = $\overline{A}\,.\,\overline{B}$ |
| 0 | 0 | | |
| 0 | 1 | | |
| 1 | 0 | | |
| 1 | 1 | | |

### e) De-Morgan's Theorem: 2



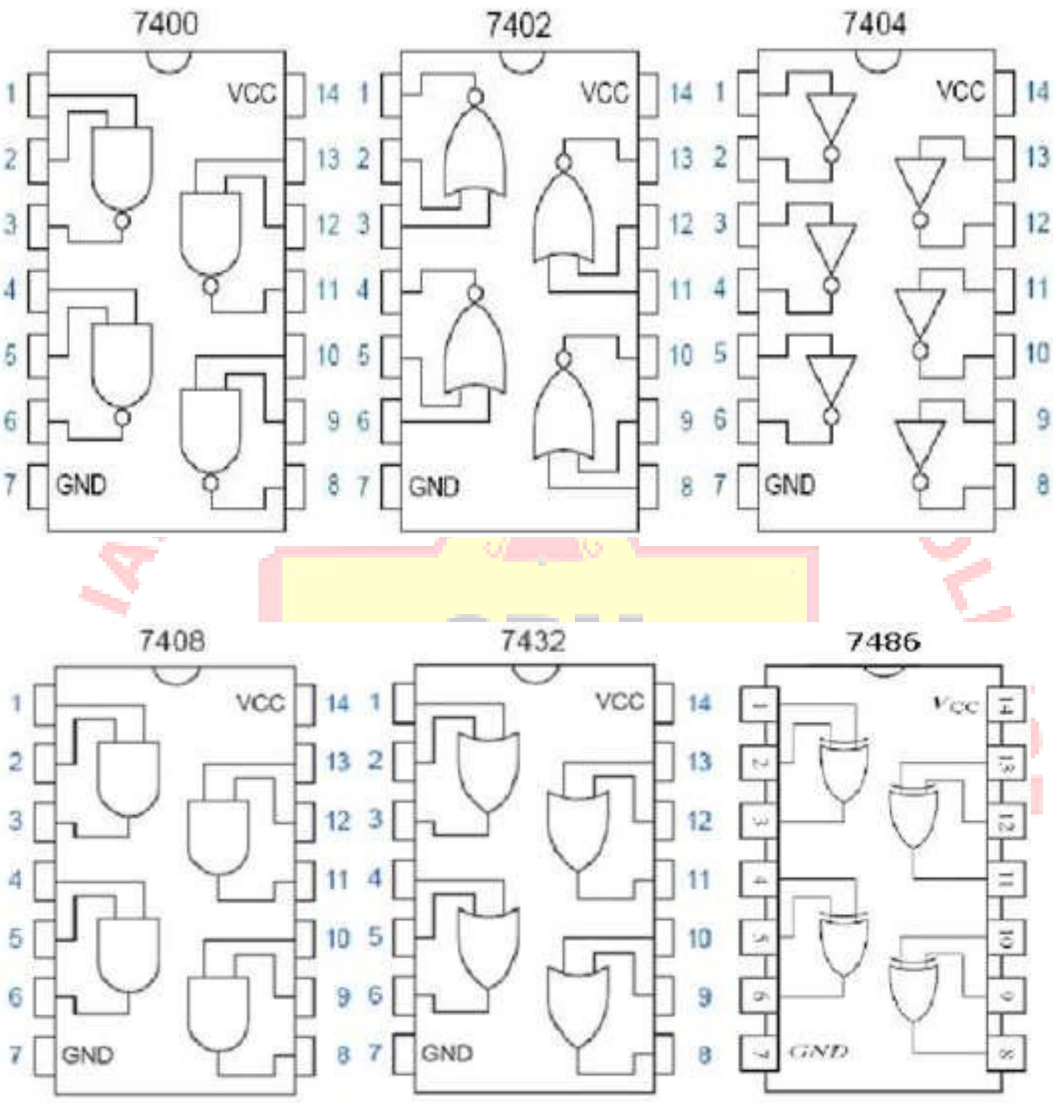
Truth Table

| Input | | Output | |
|---|---|---|---|
| A | B | (A.B)' | A' + B' |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |

2. Observation table for Second Theorm of Demorgan

| Input | | Output | |
|---|---|---|---|
| A | B | LHS = (A.B)' | RHS =A' + B' |
| 0 | 0 | | |
| 0 | 1 | | |
| 1 | 0 | | |
| 1 | 1 | | |

**<u>Pin diagram for reference</u>**



PROCEDURE

1. Test the individual ICs with its specified verification table for proper working.
2. Connections are made as per the circuit/logic diagram.
3. Make sure that the ICs are enabled by giving the suitable Vcc and ground connections.
4. Apply the logic inputs to the appropriate terminals of the ICs.
5. Observe the logic output for the inputs applied.
6. Verify the observed logic output with the verification/truth table given.

**PRE LAB:**

Basic Theorems:

a) A+0 = A b) A*1 = A

a) A+1 = 1 b) A*0 = 0

a) $A+A = A$ b) $A*A = A$

a) $A+\bar{A} = 1$ b) $A*\bar{A} = 0$

**Universal Gate:**

NAND and NOR are called universal gate, using this we can implement all other basic gates like AND, OR, NOT. One of the main disdvantages of using the complete sets of AND, OR and NOT gates is that to produce any equivalent logic gate or function we require two (or more) different types of logic gate, AND and NOT, or OR and NOT, or all three as shown above. However, we can realise all of the other Boolean functions and gates by using just one single type of universal logic gate, the NAND (NOT AND) or the NOR (NOT OR) gate, thereby reducing the number of different types of logic gates required, and also the cost.

**POST LAB:**

1. What is the Truth table for function F= A+B’C?
2. How to implement AND using NAND?
3. How to implement AND using NOR?

**RESULT**:

Thus verified the Boolean theorem using logic gates and it's the truth table

## Ex.No: 2 Design and implement Half/Full Adder and Subtractor

### AIM:

To design and construct half adder, full adder, half subtractor and full subtractor circuits and verify the truth table using logic gates.

### THEORY:

### HALF ADDER:

A half adder has two inputs for the two bits to be added and two outputs one from the sum ' S' and other from the carry ' c' into the higher adder position. Above circuit is called as a carry signal from the addition of the less significant bits sum from the X-OR Gate the carry out from the AND gate.

### FULL ADDER:

A full adder is a combinational circuit that forms the arithmetic sum of input; it consists of three inputs and two outputs. A full adder is useful to add three bits at a time but a half adder cannot do so. In full adder sum output will be taken from X-OR Gate, carry output will be taken from OR Gate.

### HALF SUBTRACTOR:

The half subtractor is constructed using X-OR and AND Gate. The half subtractor has two input and two outputs. The outputs are difference and borrow. The difference can be applied using X-OR Gate, borrow output can be implemented using an AND Gate and an inverter.

### FULL SUBTRACTOR:

The full subtractor is a combination of X-OR, AND, OR, NOT Gates. In a full subtractor the logic circuit should have three inputs and two outputs. The two half subtractor put together gives a full subtractor .The first half subtractor will be C and A B. The output will be difference output of full subtractor. The expression AB assembles the borrow output of the half subtractor and the second term is the inverted difference output of first X-OR.

### PRELAB:

### K-MAP METHOD:

The **Karnaugh map** (**KM** or **K-map**) is a method of simplifying Boolean algebra expressions The map method provides a simple, straight forward procedure for

minimizing Boolean functions. This method may be regarded as a pictorial form of truth table. This method is known as Karnaugh map or k-map. in Karnaugh maps, the cells are ordered in Gray code,[6][4] and each cell position represents one combination of input conditions, while each cell value represents the corresponding output value

**APPLICATIONS:**

Karnaugh maps are used to simplify real-world logic requirements so that they can be implemented using a minimum number of physical logic gates. A sum-of-products expression can always be implemented using AND gates feeding into an OR gate, and a product-of-sums expression leads to OR gates feeding an AND gate. Karnaugh maps can also be used to simplify logic expressions in software design

**APPARATUS REQUIRED:**

| Sl.No. | COMPONENT | SPECIFICATION | QTY. |
|---|---|---|---|
| 1. | AND GATE | IC 7408 | 1 |
| 2. | X-OR GATE | IC 7486 | 1 |
| 3. | NOT GATE | IC 7404 | 1 |
| 4. | OR GATE | IC 7432 | 1 |
| 3. | IC TRAINER KIT | - | 1 |
| 4. | PATCH CORDS | - | 23 |

**CIRCUIT DIAGRAM & TRUTH TABLE:**

a) HALF ADDER

**TRUTH TABLE**

| **A** | **B** | **CARRY** | **SUM** |
|---|---|---|---|
| **0** | **0** | **0** | **0** |
| **0** | **1** | **0** | **1** |
| **1** | **0** | **0** | **1** |
| **1** | **1** | **1** | **0** |

**K-Map for SUM**

| A \ B | 00 | 01 |
|---|---|---|
| 00 |  | 1 |
| 01 | 1 |  |

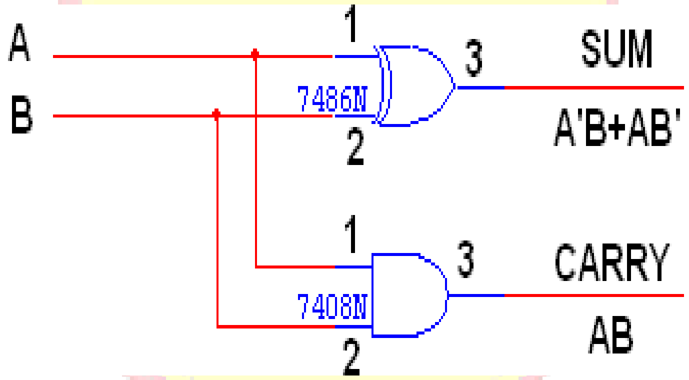
**K-Map for CARRY**

| A \ B | 00 | 01 |
|---|---|---|
| 00 |  |  |
| 01 |  | 1 |

**SUM = A’B + AB’**

**CARRY = AB**

### LOGIC DIAGRAM



## b) FULL ADDER

### TRUTH TABLE

| A | B | C | CARR Y | SU M |
|---|---|---|---|---|
| 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

| A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 |  | 1 |  | 1 |
| 1 | 1 |  | 1 |  |

SUM = A’B’C + A’BC’ + ABC’ + ABC

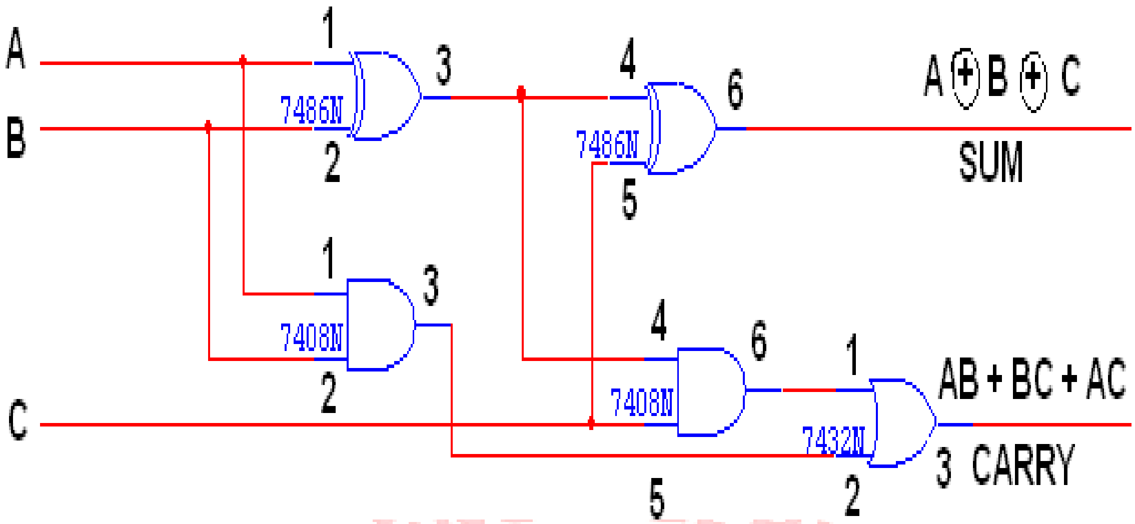
### K-Map for CARRY

| A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 |  |  | 1 |  |
| 1 |  | 1 | 1 | 1 |

CARRY = AB + BC + AC

## LOGIC DIAGRAM

### FULL ADDER USING TWO HALF ADDER



## HALF SUBTRACTOR

**TRUTH TABLE:**

| A | B | BORROW | DIFFERENCE |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 |

**K-Map for DIFFERENCE**

| A \ B | 00 | 01 |
|---|---|---|
| 00 |  | 1 |
| 01 | 1 |  |

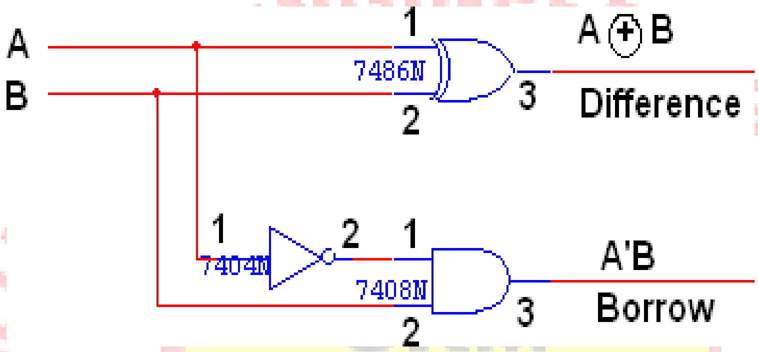
**DIFFERENCE = A’B + AB’**

**K-Map for BORROW**

| A \ B | 00 | 01 |
|---|---|---|
| 00 | | 1 |
| 01 | | |

BORROW = A’B

LOGIC DIAGRAM:



## FULL SUBTRACTOR

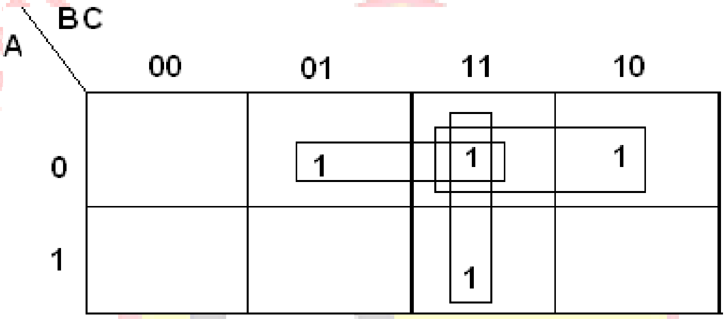
TRUTH TABLE

| A | B | C | BORROW | DIFFERENCE |
|---|---|---|---|---|
| 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

| A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 |  | 1 |  | 1 |
| 1 | 1 |  | 1 |  |

Difference = A’B’C + A’BC’ + AB’C’ + ABC

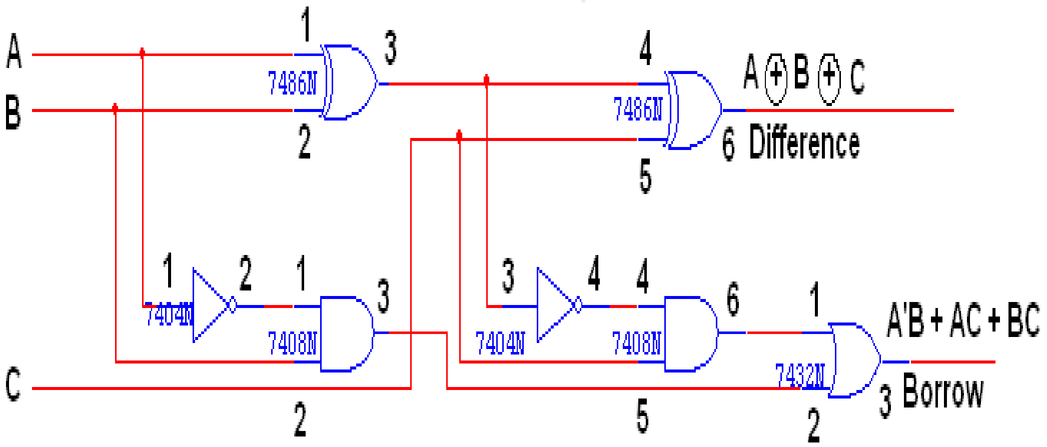


| A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 |  | 1 | 1 | 1 |
| 1 |  |  | 1 |  |

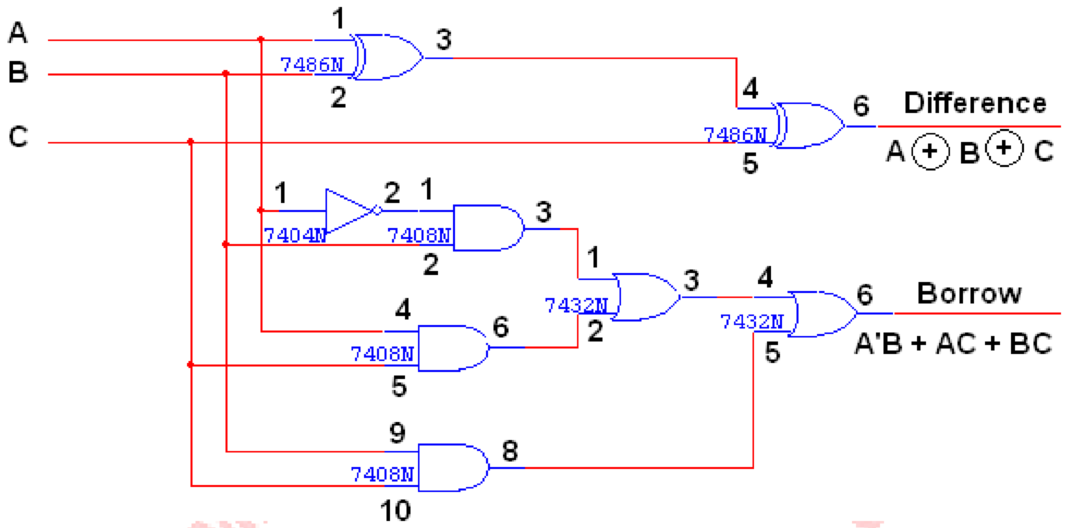
### K-Map for Borrow

Borrow = A’B + BC + A’C

### FULL SUBTRACTOR USING TWO HALF SUBTRACTOR:



## FULL SUBTRACTOR



## PROCEDURE

1. Test the individual ICs with its specified verification table for proper working.
2. Connections are made as per the circuit/logic diagram.
3. Make sure that the ICs are enabled by giving the suitable Vcc and ground connections.
4. Apply the logic inputs to the appropriate terminals of the ICs.
5. Observe the logic output for the inputs applied.

## POSTLAB:

1.Derive the equation for Full Adder using two Half Adder?

2. Derive the equation for Full Subtractor using two Half Subtractor?

3. Simplify the Boolean function x (x’+y)

## Result

Thus design and implement of half/full adder and subtractor is done

## EX.NO 3: Design and implement combinational circuits using MSI devices- Parity generator / checker

### AIM:

To design and verify the truth table of a three bit Odd Parity generator and checker.

### APPARATUS REQUIRED:

| S.No | Name of the Apparatus | Range | Quantity |
|---|---|---|---|
| 1. | Digital IC trainer kit | | 1 |
| 2. | EX-OR gate | IC 7486 | |
| 3. | NOT gate | IC 7404 | |
| 4. | Connecting wires | | As required |

### THEORY:

A parity bit is used for the purpose of detecting errors during transmission of binary information. A parity bit is an extra bit included with a binary message to make the number of 1’s either odd or even. The message including the parity bit is transmitted and then checked at the receiving end for errors. An error is detected if the checked parity does not correspond with the one transmitted. The circuit that generates the parity bit in the transmitter is called a parity generator and the circuit that checks the parity in the receiver is called a parity checker.In even parity the added parity bit will make the total number of 1’s an even amount and in odd parity the added parity bit will make the total number of 1’s an odd amount.In a three bit odd parity generator the three bits in the message together with the parity bit are transmitted to their destination, where they are applied to the parity checker circuit. The parity checker circuit checks for possible errors in the transmission. Since the information was transmitted with odd parity the four bits received must have an odd number of 1’s. An error occurs during the transmission if the four bits received have an even number of 1’s, indicating that one bit has changed during transmission. The output of the parity checker is denoted by PEC (parity error check) and it will be equal to 1 if an error occurs, i.e., if the four bits received has an even number of 1’s.

### PRELAB:

### Advantages of Parity

The advantages of parity are

- Simplicity

- Easy to use

**Applications of Parity**

The applications of parity are

- In digital systems and many hardware applications, this parity is used
- The parity bit is also used in Small Computer System Interface (SCSI) and also in Peripheral Component Interconnect (PCI) to detect the errors

## ODD PARITY GENERATOR:

**TRUTH TABLE:**

| S.No | INPUT ( Three bit message) | | | OUTPUT ( Odd Parity bit) |
|---|---|---|---|---|
| | A | B | C | P |
| 1. | 0 | 0 | 0 | 1 |
| 2. | 0 | 0 | 1 | 0 |
| 3. | 0 | 1 | 0 | 0 |
| 4. | 0 | 1 | 1 | 1 |
| 5. | 1 | 0 | 0 | 0 |
| 6. | 1 | 0 | 1 | 1 |
| 7. | 1 | 1 | 0 | 1 |
| 8. | 1 | 1 | 1 | 0 |

From the truth table the expression for the output parity bit is,

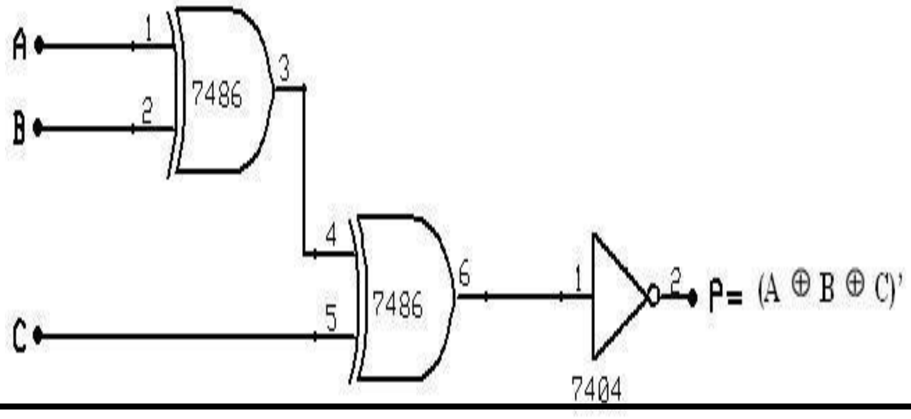
$$P( A, B, C) = \Sigma (0, 3, 5, 6)$$

Also written as,

$$P = A'B'C' + A'BC + AB'C + ABC' = (A \quad B \quad C)'$$

**CIRCUIT DIAGRAM:**

**ODD PARITY GENERATOR**



## ODD PARITY CHECKER

**TRUTH TABLE:**

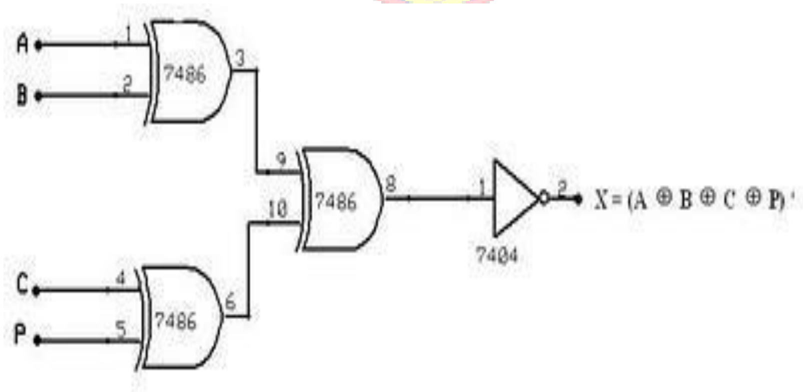
| S.No | INPUT<br>( four bit message Received ) | | | | OUTPUT<br>(Parity error check) |
|---|---|---|---|---|---|
| | A | B | C | P | X |
| 1. | 0 | 0 | 0 | 0 | 1 |
| 2. | 0 | 0 | 0 | 1 | 0 |
| 3. | 0 | 0 | 1 | 0 | 0 |
| 4. | 0 | 0 | 1 | 1 | 1 |
| 5. | 0 | 1 | 0 | 0 | 0 |
| 6. | 0 | 1 | 0 | 1 | 1 |
| 7. | 0 | 1 | 1 | 0 | 1 |
| 8. | 0 | 1 | 1 | 1 | 0 |
| 9. | 1 | 0 | 0 | 0 | 0 |
| 10. | 1 | 0 | 0 | 1 | 1 |
| 11. | 1 | 0 | 1 | 0 | 1 |
| 12. | 1 | 0 | 1 | 1 | 0 |
| 13. | 1 | 1 | 0 | 0 | 1 |
| 14. | 1 | 1 | 0 | 1 | 0 |
| 15. | 1 | 1 | 1 | 0 | 0 |
| 16. | 1 | 1 | 1 | 1 | 1 |

From the truth table the expression for the output parity checker bit is,
X (A, B, C, P) = Σ (0, 3, 5, 6, 9, 10, 12, 15)

The above expression is reduced as,
X = (A B C P) ‘

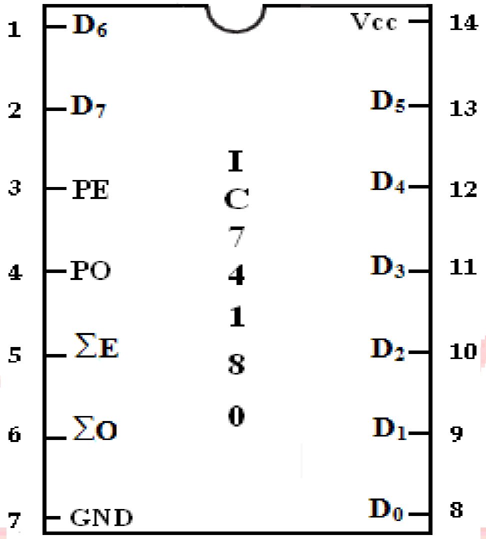
**CIRCUIT DIAGRAM:**

**ODD PARITY CHECKER**



## 8- Bit ODD/EVEN PARITY GENERATOR/ CHECKER:

### PIN DIAGRAM:



### FUNCTION TABLE:

| INPUTS | | | OUTPUTS | |
|---|---|---|---|---|
| Number of Data Inputs<br>(D$_0$ - D$_7$) | PE | PO | ∑E | ∑O |
| EVEN | 1 | 0 | 1 | 0 |
| ODD | 1 | 0 | 0 | 1 |
| EVEN | 0 | 1 | 0 | 1 |
| ODD | 0 | 1 | 1 | 0 |
| X | 1 | 1 | 0 | 0 |
| X | 0 | 0 | 1 | 1 |

**POSTLAB:**

1. Which OSI Layer takes care of error detection and correction?
2. Name the other two techniques to perform error detection and correction.
3. Discuss about generator and checker.

**RESULT**

The design of the three bit odd Parity generator and checker circuits was done and their truth tables were verified.

**EX.NO.4** **Implementation of Shift Register**

## AIM:

To implement and verify the truth table of Serial in serial Out(SISO), Serial in parallel Out(SIPO), Parallel in serial Out(PISO) & Parallel in parallel Out(PIPO)

## APPARATUS REQUIRED:

| S.No | Name of the Apparatus | Range | Quantity |
|---|---|---|---|
| 1. | Digital IC trainer kit | | 1 |
| 2. | D Flip Flop | IC 7474 | 2 |
| 3. | Connecting wires | | As required |

## THEORY:

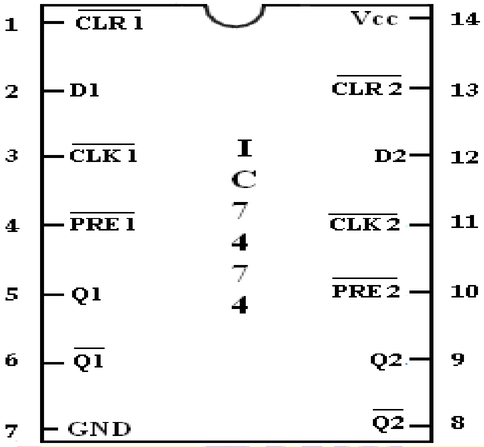
A register capable of shifting its binary information either to the left or to the right is called a shift register. The logical configuration of a shift register consists of a chain of flip flops connected in cascade with the output of one flip flop connected to the input of the next flip flop. All the flip flops receive a common clock pulse which causes the shift from one stage to the next. The Q output of a D flip flop is connected to the D input of the flip flop to the left. Each clock pulse shifts the contents of the register one-bit position to the right. The serial input determines, what goes into the right most flip flop during the shift. The serial output is taken from the output of the left most flip flop prior to the application of a pulse. Although this register shifts its contents to its left, if we turn the page upside down we find that the register shifts its contents to the right. Thus a unidirectional shift register can function either as a shift right or a shift left register.

## PRELAB:

1. Flip flops can be used to store a single bit of binary data (1or 0). However, in order to store multiple bits of data, we need multiple flip flops. N flip flops are to be connected in an order to store n bits of data. A **Register** is a device which is used to store such information. It is a group of flip flops connected in series used to store multiple bits of data.
2. The information stored within these registers can be transferred with the help of **shift registers**. Shift Register is a group of flip flops used to store multiple bits of data.
3. Applications of shift register.
   - ❖ The shift registers are used for temporary data storage.
   - ❖ The shift registers are also used for data transfer and data manipulation.
   - ❖ The serial-in serial-out and parallel-in parallel-out shift registers are used to produce time delay to digital circuits.

- The serial-in parallel-out shift register is used to convert serial data into parallel data thus they are used in communication lines where demultiplexing of a data line into several parallel line is required.
- A Parallel in Serial out shift register us used to convert parallel data to serial data.

**PIN DIAGRAM OF IC 7474:**



**Function Table:**

| Inputs | | | | Outputs | |
|---|---|---|---|---|---|
| Preset | Clear | Clock | D | Q | Q' |
| 0 | 1 | X | X | 1 | 0 |
| 1 | 0 | X | X | 0 | 1 |
| 0 | 0 | X | X | 1 | 1 |
| 1 | 1 | ↑ | 0 | 0 | 1 |
| 1 | 1 | ↑ | 1 | 1 | 0 |
| 1 | 1 | 0 | X | No Change | |

**SERIAL IN SERIAL OUT:**

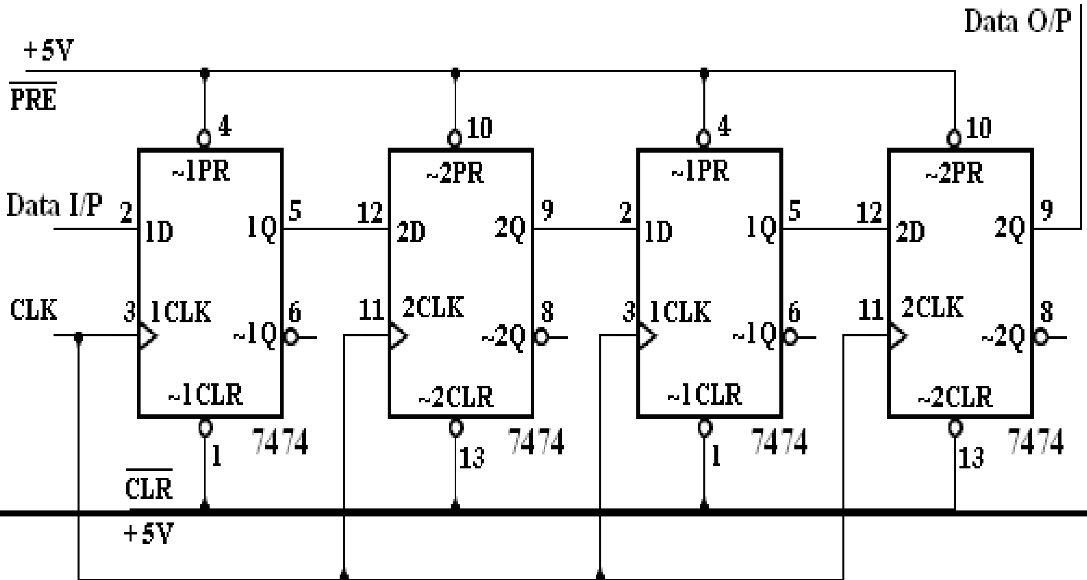


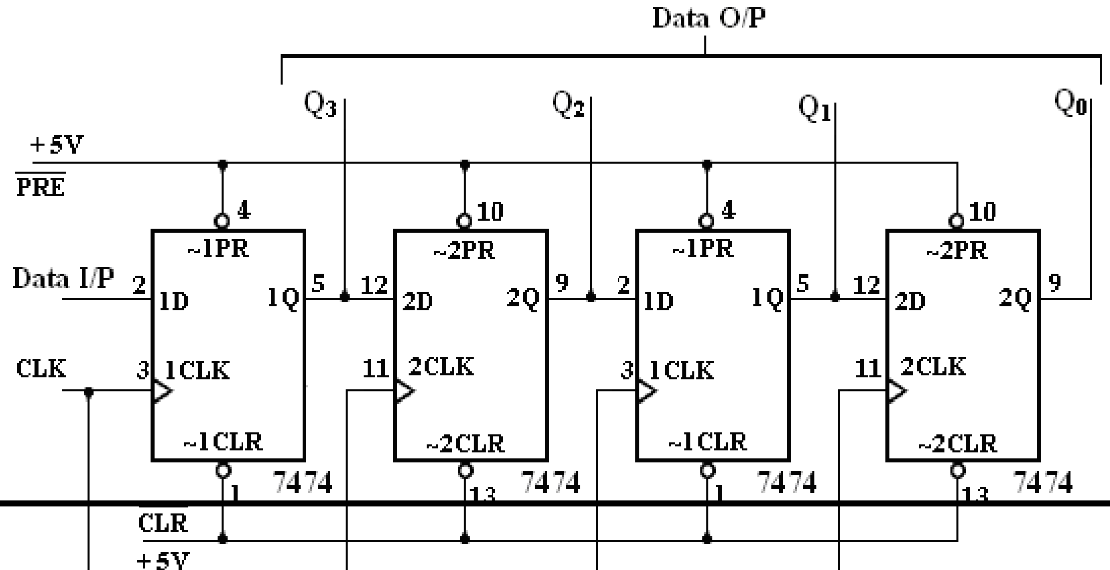
**TABLE:**

| CLK | Serial IN | Serial OUT |
|---|---|---|
| 1 | 1 | 0 |
| 2 | 1 | 0 |
| 3 | 1 | 0 |
| 4 | 1 | 1 |
| 5 | 0 | 1 |
| 6 | 0 | 1 |
| 7 | 0 | 1 |
| 8 | 0 | 0 |

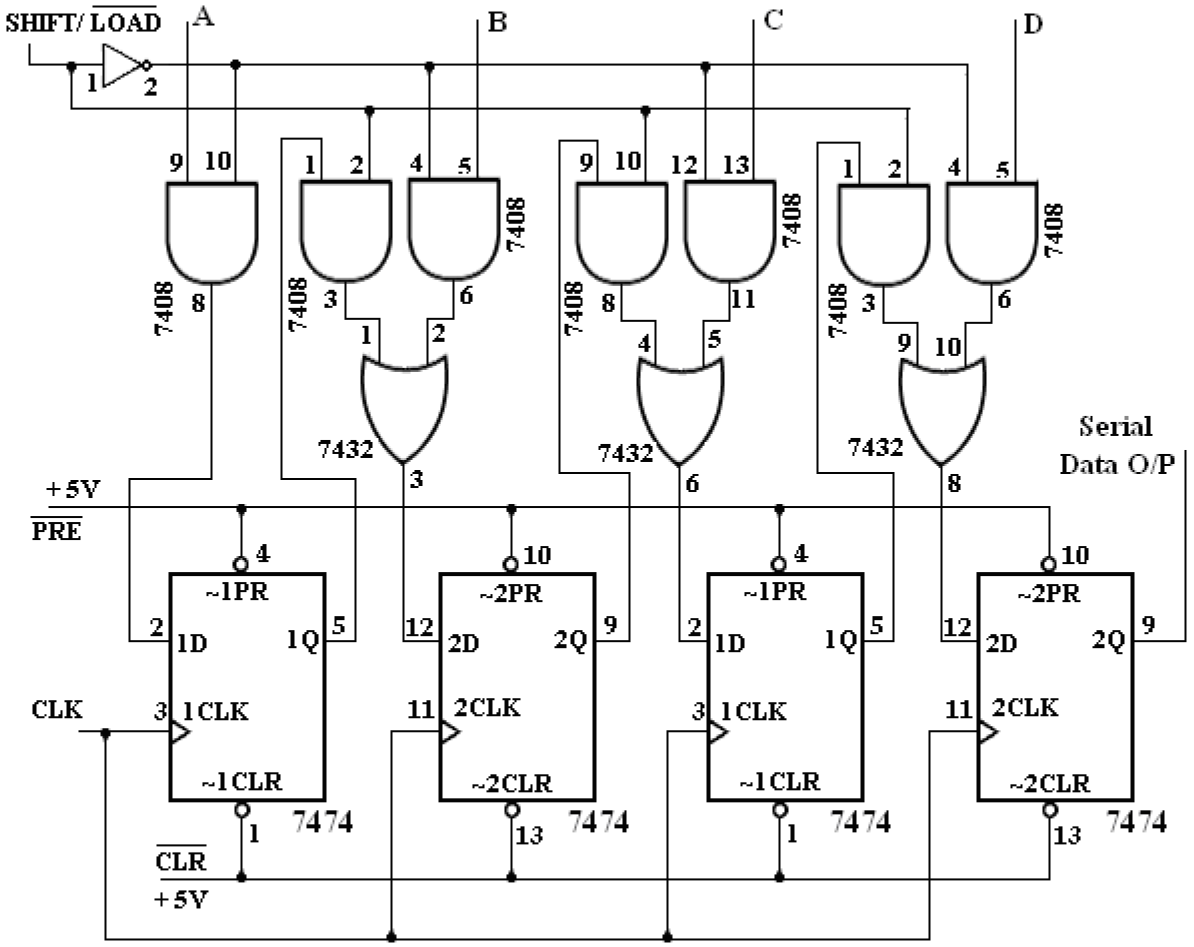
**TRUTH TABLE:**

| CLK | DATA | OUTPUT | | | |
|---|---|---|---|---|---|
| | | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ |
| 1 | 1 | 1 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 0 | 1 | 0 |
| 4 | 1 | 1 | 0 | 0 | 1 |

**Parallel IN Serial OUT:**



## LOGIC DIAGRAM:



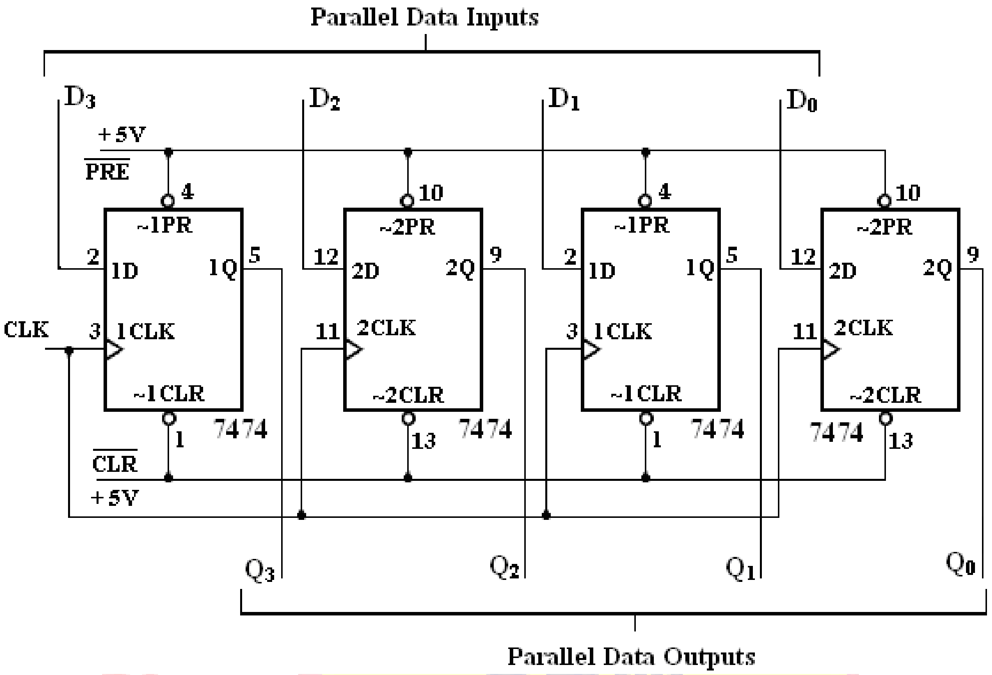
## TRUTH TABLE:

| S/ L' | CLK | INPUTS | | | | OUTPUT |
|---|---|---|---|---|---|---|
| | | A | B | C | D | Q |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 2 | 1 | 0 | 0 | 1 | 0 |
| 1 | 3 | 1 | 0 | 0 | 1 | 1 |

## Parallel IN Parallel OUT:

### Logic Diagram:

### TRUTH TABLE:



| CLK | DATA INPUTS | | | | OUTPUT | | | |
|---|---|---|---|---|---|---|---|---|
| | $D_3$ | $D_2$ | $D_1$ | $D_0$ | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 2 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |

## PROCEDURE:

1. Connections are given as per the logic diagram.
2. Logic inputs are given as per the logic diagram.
3. Observe the logic output and verify with the truth tables.

## POSTLAB:

1. Classify the types of Shift register.
2. Types of Flip flops and their truth table
3. Realize JK Flipflop using DFlip Flop.

## RESULT:

Thus the design and implementation of Serial in serial Out (SISO), Serial in parallel Out (SIPO), Parallel in serial Out (PISO), Parallel in parallel Out (PIPO) were done successfully.

## Ex.No:05 Design and implementation of synchronous counters

**AIM:**

To design and implement synchronous counter.

2bit asynchronous counter

3 bit asynchronous counter

4 bit asynchronous counter

**THEORY:**

A register that goes through the prescribed sequence of states upon the application of input pulse is called counter. A counter that follows a binary number sequence is binary counter. A n bit counter consists of n flip flop.

There are two types of counter

i) Synchronous counter

ii) Asynchronous counter

In synchronous counter common clock pulse trigger all flip flops at same time.

In synchronous counter clock pulse applied to all function flip flop. The decision whether output of flip flop is complemented is determined by J & K if J = K = 0 then no change in output if J = K = 1 then complemented.

The least significant bit always complemented Q with every pulse. A flip flop in any other positive complemented when LSB goes to 1. For (eg.) in 4 bit counter let the input $A_3A_2A_1A_0$ = 0111, then output will be 1000. Because A0 is always complemented. $A_1$ complemented since $A_1A_0$ is 11 & $A_2$ is complemented since $A_2A_1A_0$ is 111. So next output is 1000.

**PRELAB:**

**1. Shift Register Vs Counter**

A register is a memory device that can be used to store more than one bit of information and usually realized as several flip-flops with common control signals that control the movement of data to and from the register. The registers can hold data to be used for temporary storage or it can participate in arithmetic or logical operations.

A counter is a special case of a register. A counter is a register capable of incrementing and/or decrementing its contents, in other words a counter is a register that goes through a predetermined sequence of states.

### 2. Types of Counter

- Synchronous counter
- Asynchronous Counter or Ripple Counter
- Up/Down Counter
- Decade Counter
- Ring counter
- Cascaded counter
- Johnson counter
- Modulus counter

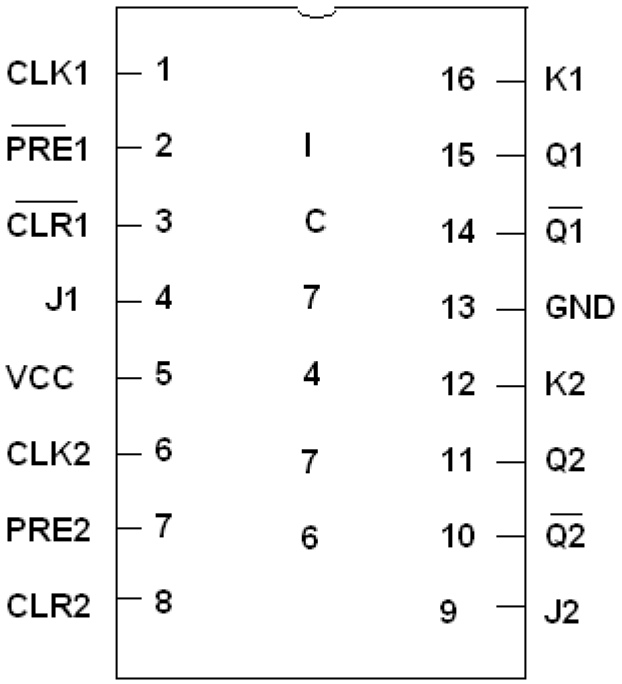
### 3. Synchronous Vs Asynchronous counter

- ❖ Asynchronous (ripple) counter – changing state bits are used as clocks to subsequent state flip-flops
- ❖ Synchronous counter – all state bits change under control of a single clock

**APPARATUS REQUIRED:**

| S.NO | COMPONENET | SPECIFICATION | QUANTITY |
|---|---|---|---|
| 1. | JK- Flip Flop | IC 7476 | 2 |
| 2. | AND GATE | IC7408 | 1 |
| 3. | IC TRAINER KIT | - | 1 |
| 4. | BATCH CARDS | - | 34 |

**PIN CONFIGURATION OF IC 7476 (JK FLIP FLOP):**

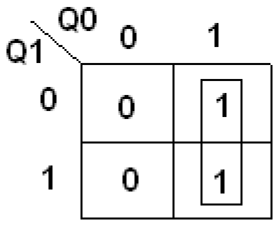


**2-BIT SYNCHRONOUS COUNTER:**

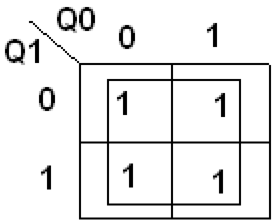
**STATE TABLE:**

| Present State | | Next State | | Inputs | |
|---|---|---|---|---|---|
| **Q1** | **Q0** | **Q1** | **Q0** | **I1** | **I0** |
| **0** | **0** | **0** | **1** | **0** | **1** |
| **0** | **1** | **1** | **0** | **1** | **1** |
| **1** | **0** | **1** | **1** | **0** | **1** |
| **1** | **1** | **0** | **0** | **1** | **1** |

**K-MAP:**



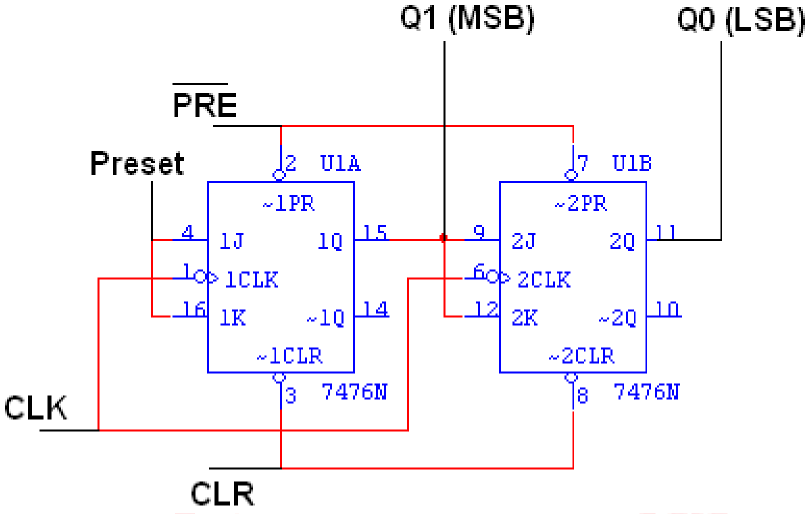
I1 = Q0



I0 = 1

**LOGIC DIAGRAM:**

**3-BIT**



**SYNCHRONOUS COUNTER:**

**STATE TABLE:**

| Present State | | | Next State | | | Inputs | | |
|---|---|---|---|---|---|---|---|---|
| Q2 | Q1 | Q0 | Q2 | Q1 | Q0 | T2 | T1 | T0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |

**K-MAP:**

| Q2 \ Q1 Q0 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | | | 1 | |
| 1 | | | 1 | |

T2 = Q1 Q0

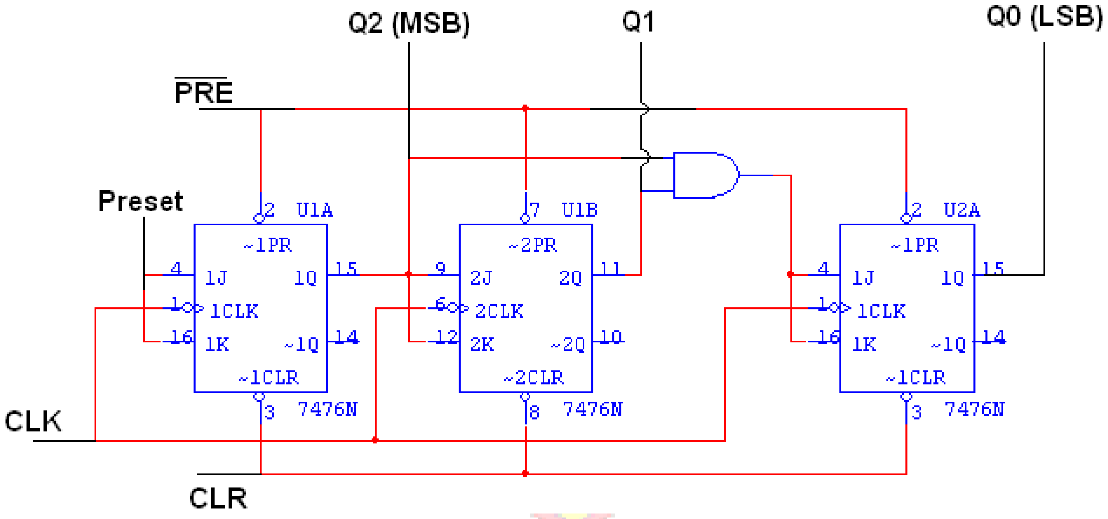
| Q2 \ Q1 Q0 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | | 1 | 1 | |
| 1 | | 1 | 1 | |

T1 = Q0

| Q2 \ Q1 Q0 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |

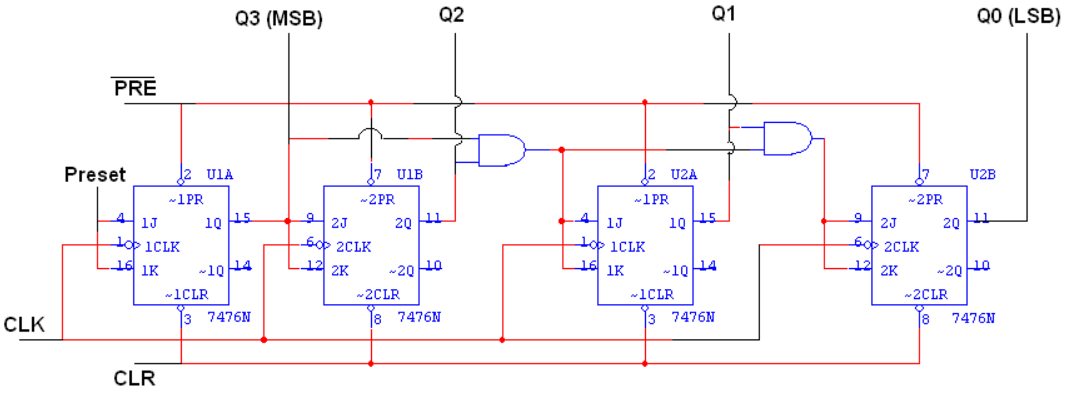
T0 = 1

**LOGIC DIAGRAM:**



**4-BIT SYNCHRONOUS COUNTER:**

**LOGIC DIAGRAM:**



## K-MAP:

| Q3 Q2 \ Q1 Q0 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | | | | |
| 01 | | | 1 | |
| 11 | | | 1 | |
| 10 | | | | |

T3 = Q2 Q1 Q0

| Q3 Q2 \ Q1 Q0 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | | | 1 | |
| 01 | | | 1 | |
| 11 | | | 1 | |
| 10 | | | 1 | |

T2 = Q1 Q0

| Q3 Q2 \ Q1 Q0 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | | 1 | 1 | |
| 01 | | 1 | 1 | |
| 11 | | 1 | 1 | |
| 10 | | 1 | 1 | |

T1= Q0

| Q3 Q2 \ Q1 Q0 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | 1 | 1 |
| 01 | 1 | 1 | 1 | 1 |
| 11 | 1 | 1 | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 |

T0 = 1

## STATE TABLE:

| Present State | Next State | Inputs |
|---|---|---|
| Q3 Q2 Q1 Q0 | Q3 Q2 Q1 Q0 | T3 T2 T1 T0 |

|  |  |  |
|---|---|---|
| 0 0 0 0 | 0 0 0 1 | 0 0 0 1 |
| 0 0 0 1 | 0 0 1 0 | 0 0 1 1 |
| 0 0 1 0 | 0 0 1 1 | 0 0 0 1 |
| 0 0 1 1 | 0 1 0 0 | 0 1 1 1 |
| 0 1 0 0 | 0 1 0 1 | 0 0 0 1 |
| 0 1 0 1 | 0 1 1 0 | 0 0 1 1 |
| 0 1 1 0 | 0 1 1 1 | 0 0 0 1 |
| 0 1 1 1 | 1 0 0 0 | 1 1 1 1 |
| 1 0 0 0 | 1 0 0 1 | 0 0 0 1 |
| 1 0 0 1 | 1 0 1 0 | 0 0 1 1 |
| 1 0 1 0 | 1 0 1 1 | 0 0 0 1 |
| 1 0 1 1 | 1 1 0 0 | 0 1 1 1 |
| 1 1 0 0 | 1 1 0 1 | 0 0 0 1 |
| 1 1 0 1 | 1 1 1 0 | 0 0 1 1 |
| 1 1 1 0 | 1 1 1 1 | 0 0 0 1 |
| 1 1 1 1 | 0 0 0 0 | 1 1 1 1 |

**PROCEDURE:**

(i) Connections are given as per the circuit diagram.

(ii) Give the logic input as per the truth table.

(iii) Observe the logical outputs and verify the truth table.

**POSTLAB:**

1. How many Flip Flops are required to design MOD 35 Counter?
2. How many states are there in a 3bit ring counter?
3. Depict the excitation table of the following flip flop
    - JK FlipFlop
    - DFlipFlop
    - T Flip Flop

4.What is the use of characteristics table and excitation table of flip flops?

**RESULT:**

Thus the design and implement of synchronous counter was designed and output was verified.

## EX.NO6 Amplitude Modulation and Demodulation

**Aim:**

1. To generate amplitude modulated wave and determine the percentage modulation. \
2. To Demodulate the modulated wave using envelope detector.

**Apparatus Required:**

1. Amplitude Modulation and Demodulation Trainer
2. Function Generator
3. Oscilloscope
4. Connecting Wires

**Theory:**

Modulation is defined as the process by which some characteristics of a carrier signal is varied in accordance with a modulating signal. The base band signal is referred to as the modulating signal and the output of the modulation process is called as the modulation signal.

Amplitude modulation is defined as the process in which is the amplitude of the carrier wave is varied about a means values linearly with the base band signal. The envelope of the modulating wave has the same shape as the base band signal provided the following two requirements are satisfied

(1). the carrier frequency fc must be much greater then the highest frequency components fm of the message signal m (t)

i.e. fc >>fm

(2) The modulation index must be less than unity. if the modulation index is greater than unity, the carrier wave becomes over modulated

**PRELAB:**

1. Need for Modulation in the communication systems
   - Reduction in the height of antenna
   - Avoids mixing of signals
   - Increases the range of communication
   - Multiplexing is possible
   - Improves quality of reception
2. Frequency range of human voice signal range becomes

   300Hz to 3000Hz

3. AM spectrum consists of Carrier frequency with both upper and lower sideband

**Procedure:**

1 Switch on the trainer and check the O/P of carrier generator with (RF generator) with oscilloscope and observe the values

2. Connect 1KHz with 2 Volts A.F signal at AF I/P to the modulator circuit with function generator and Observe the value.

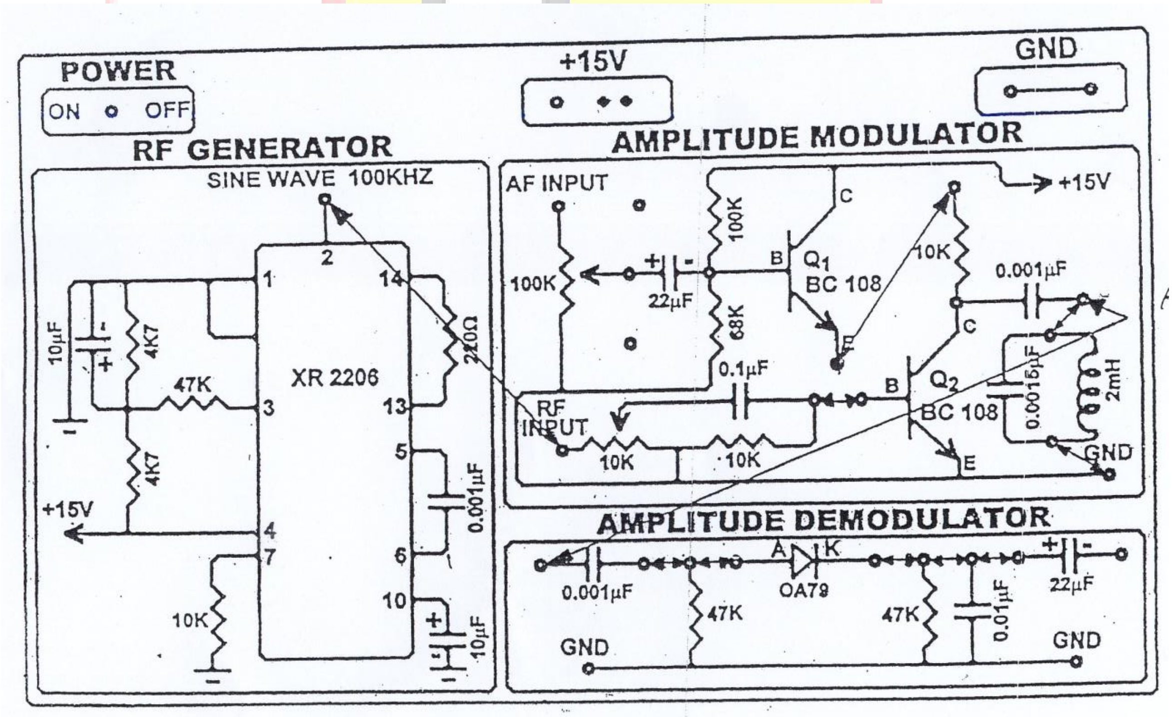
3. Connect the carrier signal(RF) at carrier I/P(RF) of modulator circuit.

4. Observe the modulator output signal at AM O/P observe the values.

5. Vary the modulating frequency and amplitude and observe the effects on the modulated waveform. Observe the critical modulation ,over modulation and under modulation forms.

6. The depth of modulation can be varied using the variable knob (potentiometer) provided at A.F. input.

7. The percentage of modulation or modulation factor can be calculated using the following formulas. ‘



10. Connect the AM wave to the demodulator circuit and observe the output

11. Note down frequency and amplitude of the demodulated output waveform.

12. Draw the demodulated wave form., m=1

Tabulation 1:

| Waveform | Amplitude | Time Period | Frequency |
|---|---|---|---|
| Unit | V | ms | Hz |
| Message Signal | | | |
| Carrier Signal | | | |
| Modulated Signal | | | |
| Demodulation | | | |

Tabulation 2:

| Modulation | Vm | Vc | Vmax | Vmin | $M=\frac{Vmax-Vmin}{Vmax+Vmin}$ |
|---|---|---|---|---|---|
| Under | | | | | |
| Over | | | | | |
| Critical | | | | | |

**Model graph**

$$\% \text{ of Modulation} = \frac{V_{max} - V_{min}}{V_{max} + V_{min}} \text{X } 100$$

$$\text{or Modulation factor} = \frac{V_{max} - V_{min}}{V_{max} + V_{min}}$$

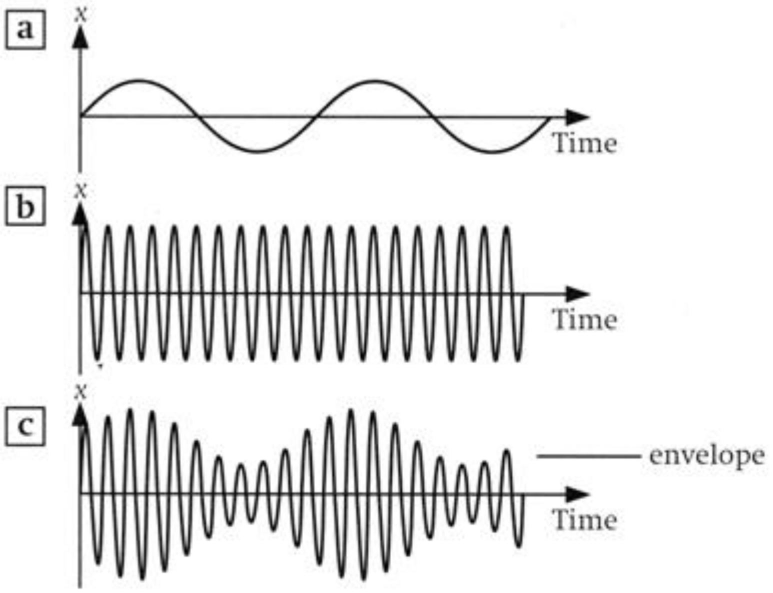
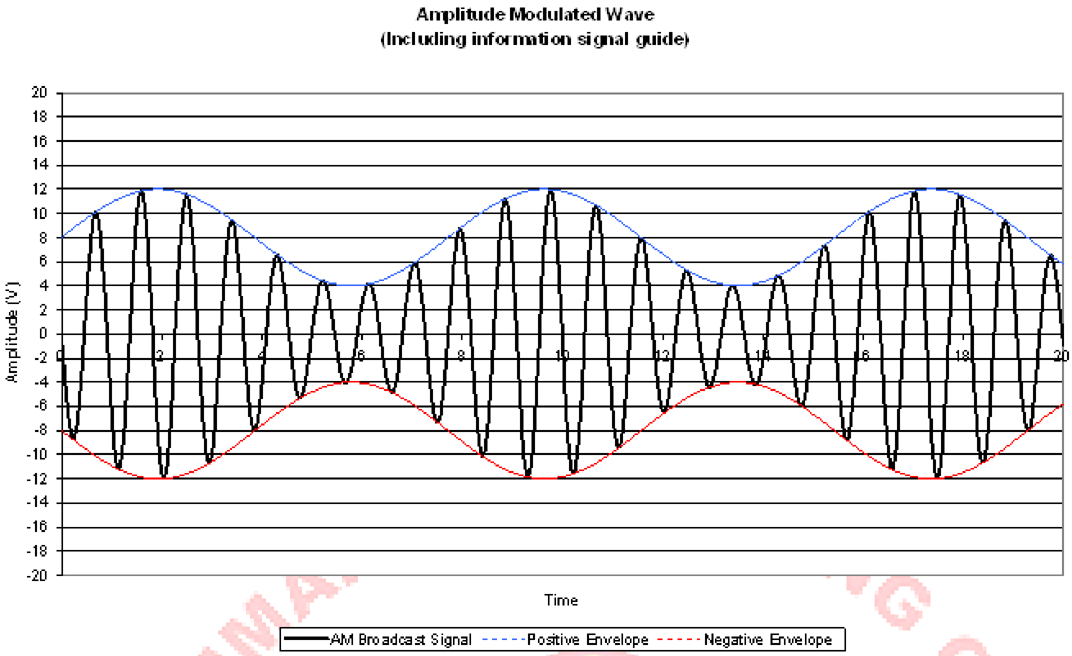


Diagram ‘a’ shows the information signal.

Diagram ‘b’ shows the unmodulated carrier signal.

Diagram ‘c’ shows the modulated carrier signal.



**POSTLAB:**

1. List out the applications of Amplitude Modulation.
2. Advantages of Amplitude Modulation
3. Disadvantages of Amplitude Modulation
4. Say the types of Amplitude Modulation
5. Carrier wave of frequency f = 1mHz with pack voltage of 20V used to modulated a signal of frequency 1kHz with pack voltage of 10v. Find out the following
   - i) μ?
   - ii) Frequencies of modulated wave?
   - iii) Bandwidth

**RESULT:**

Thus, the amplitude modulation and demodulation was constructed and percentage of modulation was calculated

## Ex.No: 7: Frequency Modulation and Demodulation

**AIM:**

- To generate frequency modulated signal and determine the modulation index and bandwidth for various values of amplitude and frequency of modulating signal.
- To demodulate a Frequency Modulated signal using FM detector.

**APPARATUS REQUIRED:**

1. Amplitude Modulation and Demodulation Trainer
2. Function Generator
3. Oscilloscope
4. Connecting Wires

**THEORY:**

The process, in which the frequency of the carrier is varied in accordance with the instantaneous amplitude of the modulating signal, is called "Frequency Modulation". The FM signal is expressed as Where*Ac* is amplitude of the carrier signal,*f c* is the carrier frequency β is the modulation index of the FM wave

**MODULATION INDEX:**

The modulation index is equal to the ratio of the frequency deviation to the modulating frequency. The modulation index will vary according to the frequency that is modulating the transmitted carrier and the amount of deviation. However when designing a system it is important to know the maximum permissible values. This is given by the deviation ratio and is obtained by inserting the maximum values into the formula for the modulation index.

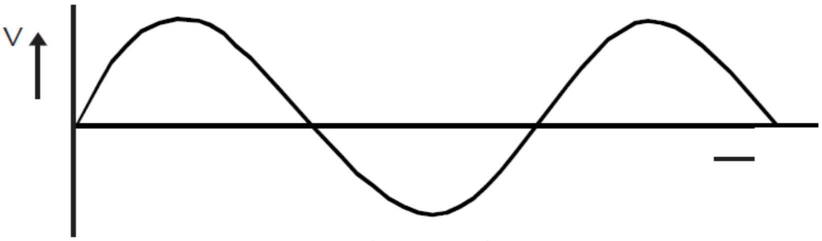
D = (Max deviation frequency) / (Max modulation frequency)

This may also be expressed as:

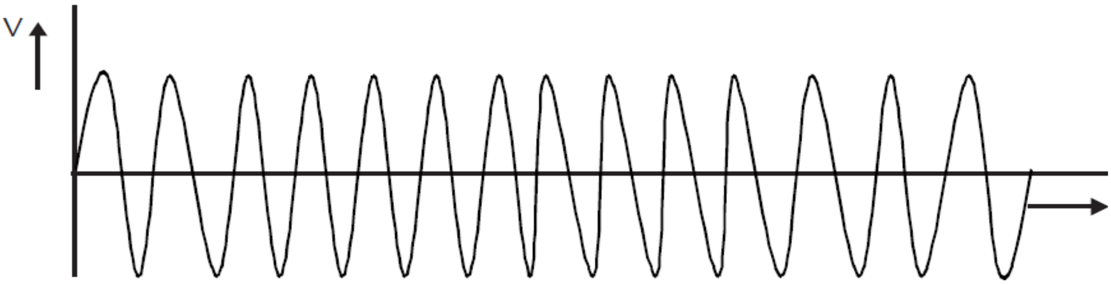
$$\beta = \Delta\omega / \omega m$$

For a VHF FM sound broadcast transmitter the maximum deviation is 75 kHz and the maximum modulation frequency is 15 kHz giving a deviation ratio of 5.

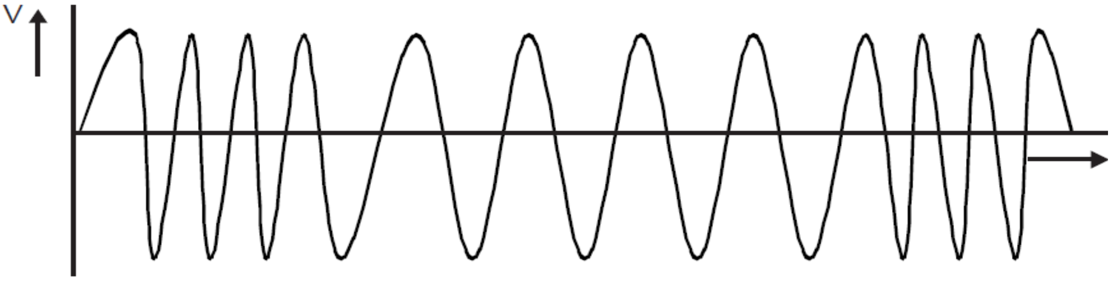
**Expected Waveforms :**



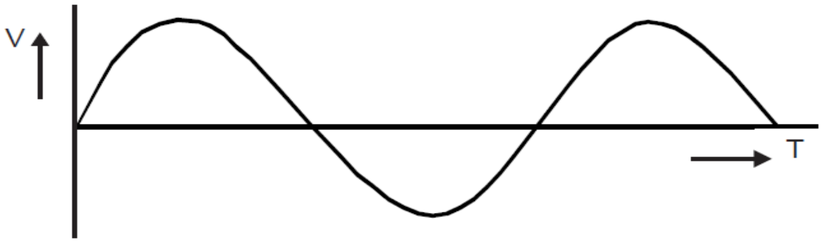
Input Modulation Signal



Carrier Wave



Frequency Modulated Signal



Demodulated signal

CALCULATING FM MODULATION BANDWIDTH

A formula is used to determine the RF Bandwidth or occupancy for an FM signal. If you are designing an FM system on microwave or satellite, you will need to take care that your signal does not cross-talk into other signals on the system.

Frequency Modulation creates modulation sidebands that theoretically extend to infinite bandwidth. These sidebands consist of Bessel Functions of any order. From a practical standpoint the band occupancy of an FM modulated carrier only needs to count the Bessel

Function sidebands of significant amplitude. The formula that calculates this bandwidth is called

CARSON'S RULE.

**CARSON'S RULE** requires knowing the modulating frequency and the maximum frequency deviation of the transmitted carrier. As an example, a monaural RF band modulator will have a peak deviation of 75KHz and the highest audio frequency is 15KHz. To calculate the CARSON'S RULE bandwidth occupancy of this signal, add the highest audio frequency to the peak deviation (15KHz + 75KHz = 90KHz), then multiply by two to include both the upper and lower sideband (90KHz X 2 = 180KHz). The CARSON'S BANDWIDTH for this signal is 180KHz. Since there are many Bessel Function sidebands beyond 180KHz, FM channels must be spaced considerably farther apart than 180KHz. The FCC has determined that a spacing of 400KHz provides sufficient "Guard Band" to effectively prevent inter-channel cross-talk, but that 180KHz is sufficient bandwidth to receive the original modulation with less than 1% distortion. The distortion is due to a failure to receive all of the modulation energy.

Similar CARSON RULE calculations can be used for other modulation bandwidths and peak deviations, with similar considerations for Guard Bands between channels.

Amplitude Modulation bandwidth can be considered exactly two times the highest frequency of modulation, while Frequency Modulation bandwidth is described by Bessel Functions that extend much higher than those of Amplitude Modulation. In fact the "FM Advantage" in signal-to-noise ratio stems exactly from spreading the modulation over a greater bandwidth than Amplitude Modulation.

CARSON'S RULE

BANDWIDTH = 2 X (PEAK DEVIATION + HIGHEST MODULATING FREQUENCY)

**PRELAB:**

**Difference between AM and FM** include the following.

1. Equation for FM:
   $V= A \sin [ wct +\Delta f / fm \sin wmt ] = A \sin [ wct + mf \sin wmt ]$
2. Equation for
   $AM = Vc ( 1 + m \sin \omega mt ) \sin \omega ct$ where m is given by $m = Vm / Vc$
3. In FM, the Modulation Index can have any value greater than 1 or less than one
4. In AM, the Modulation Index will be between 0 and 1
5. In FM, carrier amplitude is constant.
6. Therefore transmitted power is constant.
7. Transmitted power does not depend on the modulation index

8. Transmitted power depends on the modulation index
9. The number of significant sidebands in FM is large.
10. Only two sidebands in AM
11. A bandwidth of FM depends on the modulation index of FM
12. Bandwidth does not depend on the modulation index of AM. Always 2 sidebands. BW of AM is 2 fm
13. FM has better noise immunity.FM is rugged/robust against noise. The quality of FM will be good even in the presence of noise.
14. In AM, quality is affected seriously by noise
15. The bandwidth required by FM is quite high.FM bandwidth = 2 [$\Delta$f + fm].
16. The bandwidth required by AM is less (2 fm)
17. Circuits for FM transmitter and receiver are very complex and very expensive.
18. Circuits for AM transmitter and receiver are simple and less expensive

**PROCEDURE:**

1. Switch on the FM experimental board.

2. Connect Oscilloscope at the FM O/P and observe the carrier frequency without connect the A.F. input.

3. Connect around 7KHz sine wave (A.F. signal) from function generator to the input of the frequency modulator (At sine input). Observe the value.

4. Now observe the frequency modulated output on the 1st channel of the CRO and adjust the amplitude of the AF signal frequency modulated wave form with measurable deviation observe the values. Measure the frequency deviation.

5. Vary the modulating frequency (A.F Signal) and amplitude and observe the effects on the modulated waveform.

6. Connect the FM o/p to the FM i/p of De-modulator.

7. Vary the potentiometer (VCO) provided in the demodulator section.

8. Observe the output at demodulation o/p on second channel of CRO and compare the input signal.

9. Draw the demodulated wave form

**PRECAUTIONS:**

1. Connect the circuit properly.

2. Apply the required voltages wherever needed.

3. Do not apply stress on the components.

Tabulation :

| Waveform | Amplitude | Time Period |
|---|---|---|
| Unit | V | Ms |
| Message  Signal | | |
| Carrier Signal | | |
| Modulated Signal | | |
| Demodulated Signal | | |

Frequency deviation $=1/T_{max}-1/T_{min}$

**POSTLAB:**

1. List out the Applications of FM?
2. Discuss about advantages of FM modulation?
3. Define Digital Modulation?
4. List out the advantages of Digital modulation over analog modulation.

**Result:**

Thus, the frequency modulation and demodulation was constructed and percentage of modulation was calculated

## Ex.no 8: Pulse Code Modulation and Demodulation

**AIM:**

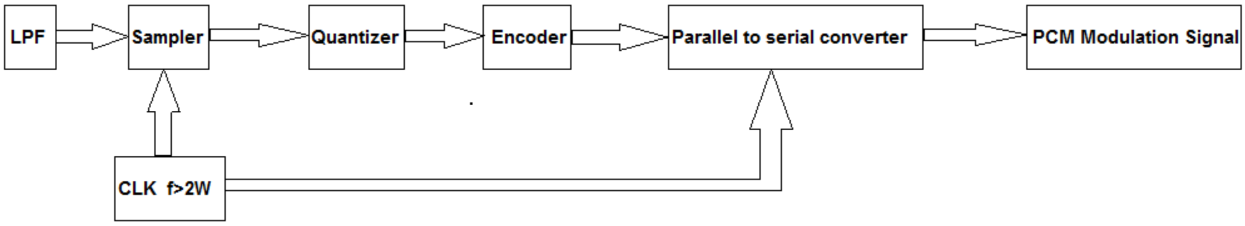
To study the circuit of pulse code modulation and demodulation.

**APPARATUS REQUIRED:**

1. Pulse code modulation and demodulation trainer
2. CRO
3. Connecting Wires

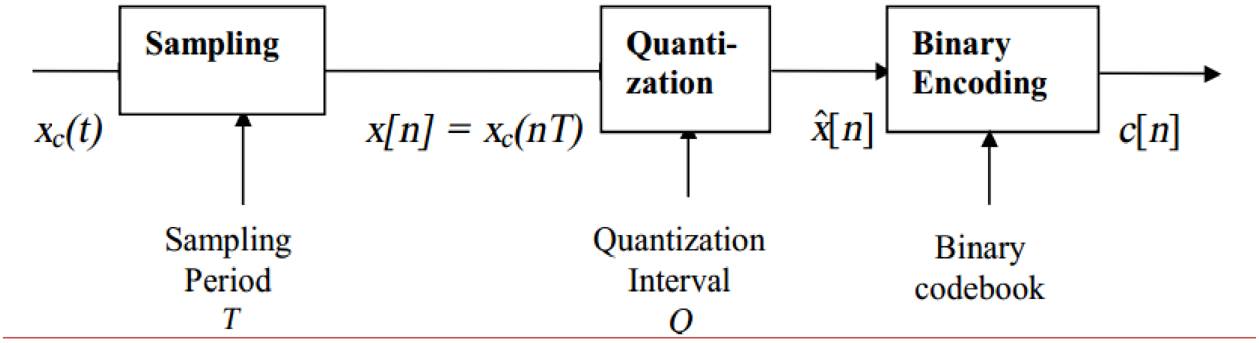
**THEORY:**

In pulse code modulation (PCM) a message signal is represented by a sequence of coded pulses, which is accomplished by representing the signal in discrete form in both time and amplitude. The basic elements of a PCM system are shown below.



The basic operations performed in the transmitter of a PCM system are sampling, quantizing and encoding. The low pass filter prior to sampling is included to prevent aliasing of the message signal. The incoming message signal is sampled with a train of narrow rectangular pulses so as to closely approximate the instantaneous sampling process. To ensure perfect reconstruction of the message signal at the receiver, the sampling rate must be greater than twice the highest frequency component W of the message signal in accordance with the sampling theorem.

The quantizing and encoding operations are usually performed in the same circuit, which is called an analog-to-digital converter. The same circuit, which is called and analog-to-digital converter. The sampled version of the message signal is then quantized, thereby providing a new representation of the signal that is discrete in both time and amplitude.



In combining the process of sampling and quantization, the specification of a continuous message (baseband) signal becomes limited to a discrete set of values, but not in the form best suited to transmission. To exploit the advantages of sampling and quantizing for the purpose of making the transmitted signal more robust to noise, interference and other channel impairments, we require the use of an encoding process to translate the discrete set of sample values to a more appropriate form of signal.

**Regeneration:**

The most important feature of PCM system lies in the ability to control the effects of distortion and noise produced by transmitting a PCM signal through a channel. This capability is accomplished by reconstructing the PCM signal by means of a chain of regenerative repeaters located at sufficiently closed spacing along the transmission route. As illustrated in figure below three basic functions are performed by a regenerative repeater: equalization timing and decision making

**Regeneration Repeater**

The equalizer shapes the received pulses so as to compensate for the effects of amplitude and phase distortion produced by the non ideal transmission characteristics of the channel.

The timing circuit provides a periodic pulse trainer, derived from the received pulses, for sampling the equalized pulses at the instants of time where the signal to noise ratio is maximum.

Each sample so extracted is compared to predetermined threshold in the decision making device. In each bit interval, a decision is then made whether the symbol is 1 or 0 on the basis of whether the threshold is exceeded or not. If the threshold is exceeded, a pulse representing symbol '1' is transmitted. In the way, the accumulation of distortion and noise in a repeater span is completely removed.

The basic operations in the receiver are regeneration of impaired signals, decoding and reconstruction of the train of quantized samples. The first operation in the receiver is to regenerate (i.e., reshape and cleanup) the received pulses one last time. These clean pulses are then regrouped in to code words and decoded into a quantized PAM signal. The decoding process involves generating a pulse, the amplitude of which is the linear sum of all the pulses in the code word. The final operation in the receiver is to recover the message signal by passing the decoder output through a low pass reconstruction filter whose cut-off frequency is equal to the message band width W. Assuming that the transmission path is error free, the recovered signal includes no noise with the exception of the initial distortion introduced by the quantization process.

**PRELAB:**

1. Types of Pulse Code Modulation

- Linear PCM (LPCM) is PCM with linear quantization.[32]
- Differential PCM (DPCM) encodes the PCM values as differences between the current and the predicted value. An algorithm predicts the next sample based on the previous samples, and the encoder stores only the difference between this prediction and the actual value. If the prediction is reasonable, fewer bits can be used to represent the same information. For audio, this type of encoding reduces the number of bits required per sample by about 25% compared to PCM.
- Adaptive DPCM (ADPCM) is a variant of DPCM that varies the size of the quantization step, to allow further reduction of the required bandwidth for a given signal-to-noise ratio.
- Delta modulation is a form of DPCM that uses one bit per sample to indicate whether the signal is increasing or decreasing compared to the previous sample.

2. Sampling

Sampling is a process of measuring the amplitude of a continuous-time signal at discrete instants, converts the continuous signal into a discrete signal. For example, conversion of a sound wave to a sequence of samples.

**PROCEDURE:**

1. Switch ON Pulse Code Modulation and Demodulation Trainer Kit.

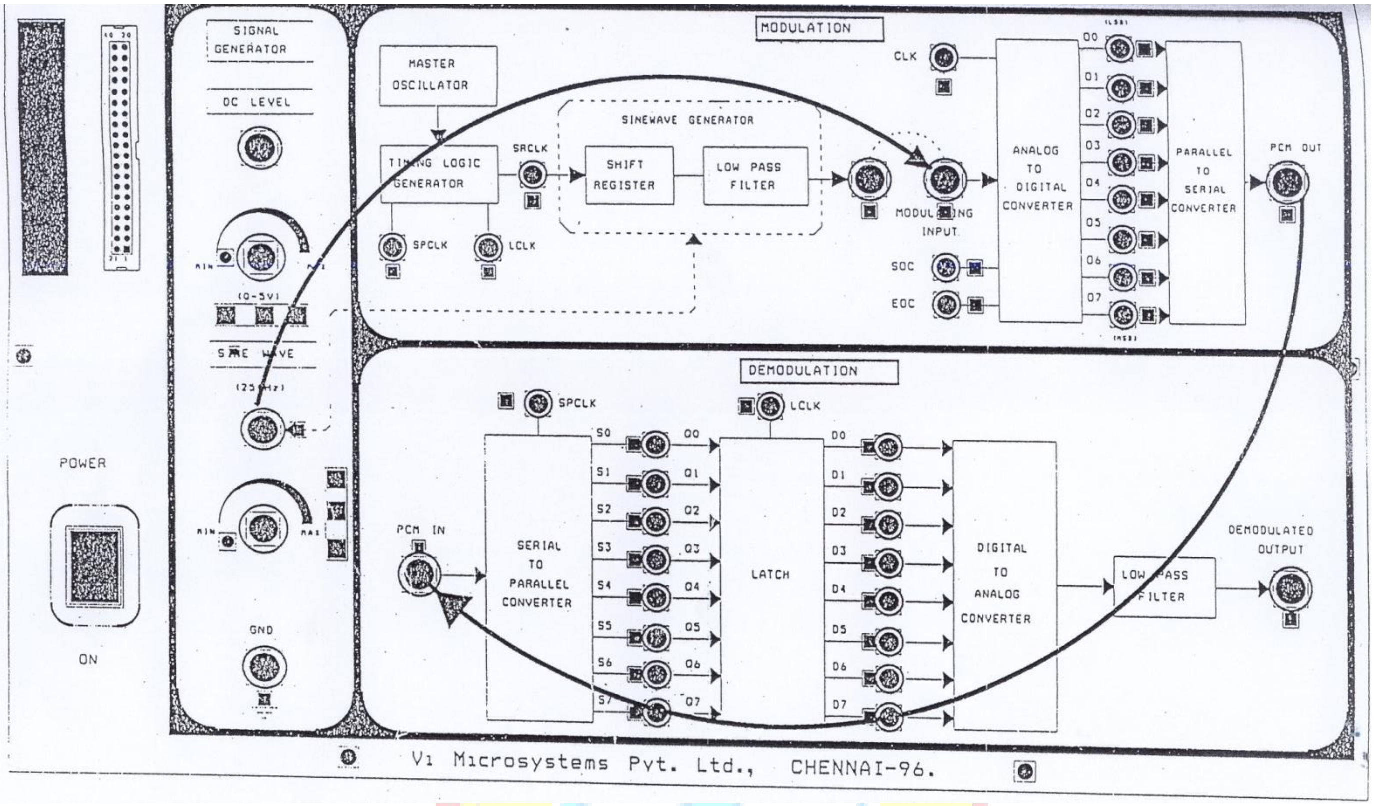
2. Connect the AF to the Analog I/P of modulation section. Observe the value.

3. Connect the clock O/P of bit clock generator to the clock I/P of modulation section. Observe the value.

4. By varying the variable AF observe the PCM O/P on CRO with EOC

5. Connect the PCM O/P to PCM I/P of demodulation section.

7. Observe the DAC O/P at channel 1 of CRO and observe the demodulated O/P at channel 2 of CRO with SOC.

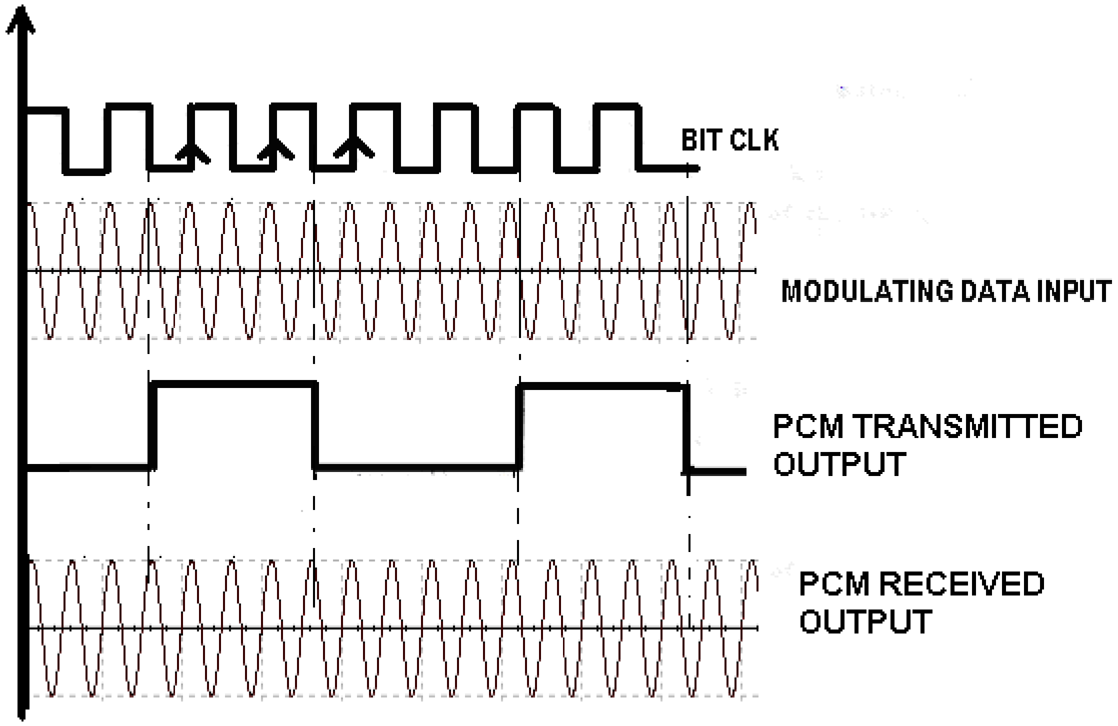


**Tabulation:**

| Signal | Amplitude | Time Period |
|---|---|---|
| Unit | V | ms |
| Modulating signal input | | |
| Clock Pulse Code(Bit clock) | | |
| PCM Output | | |
| Demodulated output | | |

| | | |
|---|---|---|
| | | |

**Model graph**



**POSTLAB:**

1. List out the advantages of Pulse Code Modulation
2. Define Quantization.
3. Say the disadvantage of PCM?
4. Discuss about DPCM Vs ADPCM

**RESULT:**

Thus, the pulse code modulation and demodulation studied.

**Ex.no: 9 FSK, PSK and DPSK schemes (Simulation)**

**AIM:**

To generate and demodulate frequency shift keyed (FSK) signal using MATLAB

**THEORY:**

**Generation of FSK:**

Frequency-shift keying (FSK) is a frequency modulation scheme in which digital information is transmitted through discrete frequency changes of a carrier wave. The simplest FSK is binary FSK (BFSK). BFSK uses a pair of discrete frequencies to transmit binary (0s and 1s) information. With this scheme, the "1" is called the mark frequency and the "0" is called the space frequency.

In binary FSK system, symbol 1 & 0 are distinguished from each other by transmitting one of the two sinusoidal waves that differ in frequency by a fixed amount.

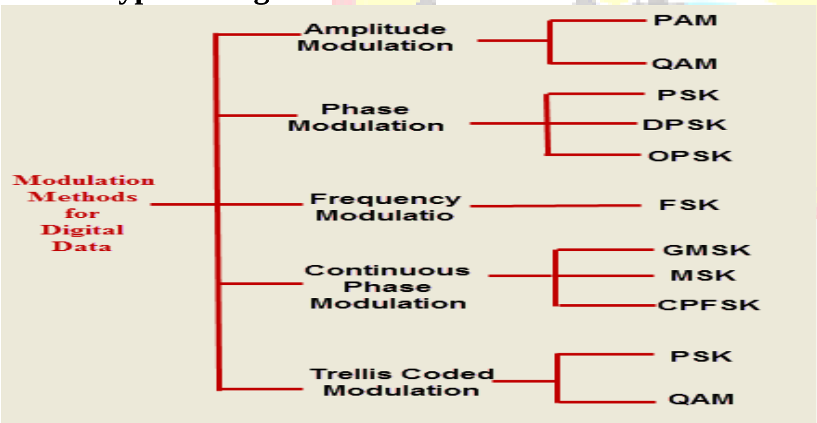
**PRELAB:**

1. **Diference Between Analog modulation and Digital Modulation**

| Analog Modulation | Digital Modulation |
|---|---|
| An AM signal can signify any value in a range. | A DM signal can only signify with a set of discrete values. |
| In analog modulation (AM), the input must be in the form of analog | In digital modulation (DM), the input must be the data in the form of digital |
| In AM, the value between the max & min is considered to be applicable. | In DM, only two binary numbers are considered applicable such as 1 and 0. |

| | |
|---|---|
| Most of the signals that we transmit in nature are analog like voice signal, and it is much simpler to complete analog modulation compare with digital | But using digital modulation, you require transmitting through an ADC converter before transmission & a DAC at the end of the receiver for recovering the unique signal. The extra phases required to transmit DM (digital modulation) enhances both the price and difficulty of the transmitter as well as receiver. |
| The AM can generate a signal to carry the frequently changing data. | The DM generates a signal whose rate changes at particular time intervals. |
| In AM, it is not easy to disconnect the signal from noise. | In DM, the signal can simply disconnect from noise |

1. **Types of Digital Modulation**



**PROGRAM 1**
**FSK Modulation**

```
clc;
clear all;
close all;
%GENERATE CARRIER SIGNAL
Tb=1; fc1=2;fc2=5;
t=0:(Tb/100):Tb;
c1=sqrt(2/Tb)*sin(2*pi*fc1*t);
```

```
c2=sqrt(2/Tb)*sin(2*pi*fc2*t);
%generate message signal
N=8;
m=rand(1,N);
t1=0;t2=Tb
for i=1:N
t=[t1:(Tb/100):t2]
if m(i)>0.5
m(i)=1;
m_s=ones(1,length(t));
invm_s=zeros(1,length(t));
else
m(i)=0;
m_s=zeros(1,length(t));
invm_s=ones(1,length(t));
end
message(i,:)=m_s;
%Multiplier
fsk_sig1(i,:)=c1.*m_s;

fsk_sig2(i,:)=c2.*invm_s;

fsk=fsk_sig1+fsk_sig2;




 %plotting the message signal and the modulated signal

 subplot(3,2,2);axis([0 N -2 2]);plot(t,message(i,:),'r');

title('message signal');xlabel('t---->');ylabel('m(t)');grid on;hold on;

subplot(3,2,5);plot(t,fsk(i,:));

title('FSK signal');xlabel('t---->');ylabel('s(t)');grid on;hold on;

 t1=t1+(Tb+.01); t2=t2+(Tb+.01);

end

hold off %

Plotting binary data bits and carrier signal

ubplot(3,2,1);stem(m);

title('binary data');xlabel('n---->'); ylabel('b(n)');grid on;

subplot(3,2,3);plot(t,c1);

title('carrier signal-1');xlabel('t---->');ylabel('c1(t)');grid on;

subplot(3,2,4);plot(t,c2);

title('carrier signal-2');xlabel('t---->');ylabel('c2(t)');grid on;
```

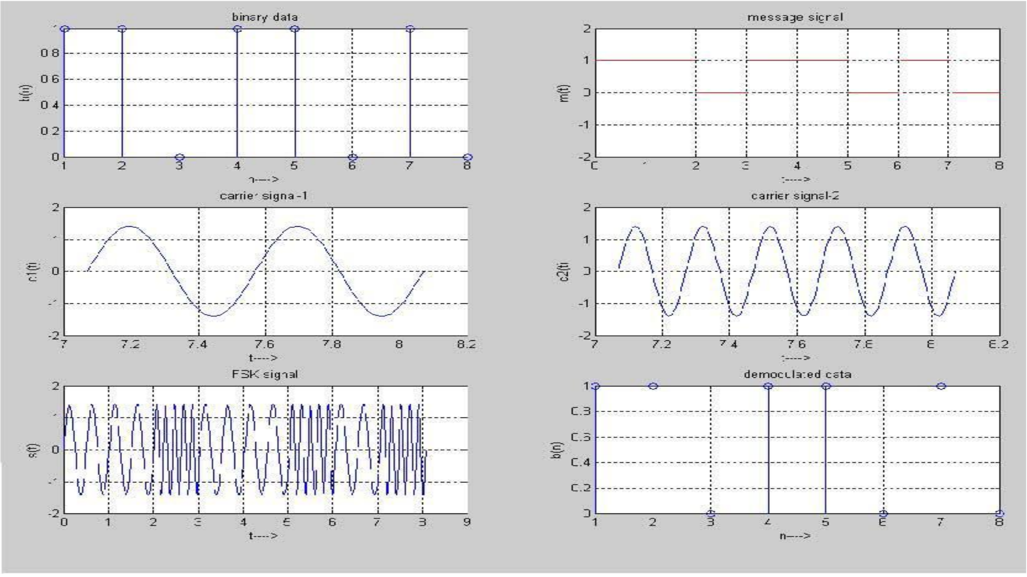
**% FSK Demodulation**

```
t1=0;t2=Tb
for i=1:N
 t=[t1:(Tb/100):t2]
%correlator
x1=sum(c1.*fsk_sig1(i,:));
x2=sum(c2.*fsk_sig2(i,:));
 x=x1-x2;
 %decision device
if x>0
demod(i)=1;
else
demod(i)=0;
end
t1=t1+(Tb+.01);
t2=t2+(Tb+.01);
end
%Plotting the demodulated data bits
subplot(3,2,6);stem(demod);
title(' demodulated data');xlabel('n---->');ylabel('b(n)'); grid on;
```

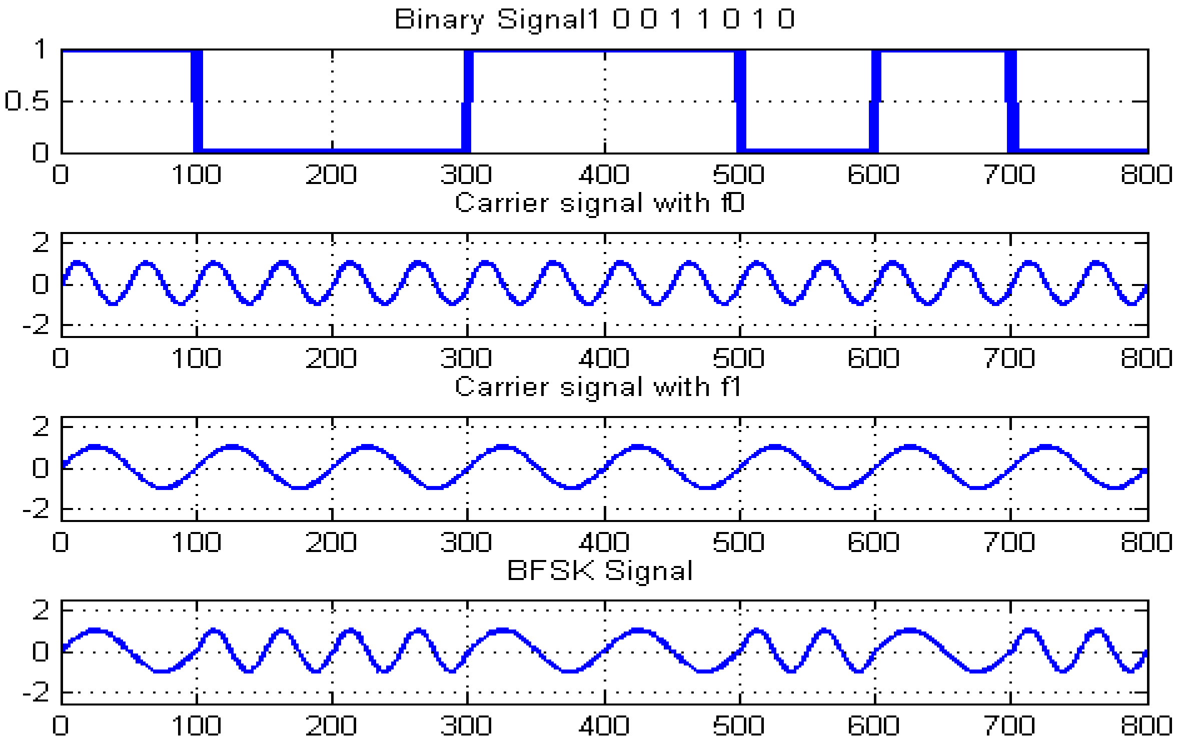
**Model graph**

**PROGRAM 2**

% Program for Digital Modulation FREQUENCY SHIFT KEYING (FSK)



```
clc;closeall;clear all;
g=[1 0 0 1 1 0 1 0];
f0=2; f1=1; n=1000;
t1 = 0:0.01:n-0.01;
c1=sin(2*pi*f0*t1);
c2=sin(2*pi*f1*t1);
t=0:2*pi/99:2*pi;
cp=[];sp=[];
mod=[];mod1=[];bit=[];
for n=1:length(g);
if g(n)==0;
die=ones(1,100);
c=sin(f0*t);
se=zeros(1,100);
else g(n)==1;
die=ones(1,100); c=sin(f1*t);
se=ones(1,100);
end
cp=[cp die];
mod=[mod c];
bit=[bit se];
end
fsk=cp.*mod;
subplot(4,1,1);plot(bit,'LineWidth',3.5);grid on;
title('Binary Signal1 0 0 1 1 0 1 0 ');
subplot(4,1,2);plot(c1,'LineWidth',1.5);grid on;
title('Carrier signal with f0');
axis([0 100*length(g) -2.5 2.5]);
subplot(4,1,3);plot(c2,'LineWidth',1.5);grid on;
title('Carrier signal with f1');
axis([0 100*length(g) -2.5 2.5]);
subplot(4,1,4);plot(fsk,'LineWidth',1.5);grid on;
title('BFSK Signal');
axis([0 100*length(g) -2.5 2.5]);
```



**Result:**

The program for FSK modulation and demodulation has been simulated in MATLAB and necessary graphs are plotted.

**Phase Shift Keying:**

**Generation of PSK signal:**

PSK is a digital modulation scheme that conveys data by changing, or modulating, the phase of a reference signal (the carrier wave). PSK uses a finite number of phases, each assigned a unique pattern of binary digits. Usually, each phase encodes an equal number of bits. Each pattern of bits forms the symbol that is represented by the particular phase. The demodulator, which is designed specifically for the symbol-set used by the modulator, determines the phase of the received signal and maps it back to the symbol it represents, thus recovering the original data.

In a coherent binary PSK system, the pair of signal S1(t) and S2 (t) used to represent binary symbols 1 & 0 are defined by

S1 (t) = $\sqrt{}$2Eb/ Tb Cos 2$\pi$fct

S2 (t) =$\sqrt{}$2Eb/Tb (2$\pi$fct+$\pi$) = - $\sqrt{}$ 2Eb/Tb Cos 2$\pi$fct where 0 ≤ t<Tb and

Eb = Transmitted signed energy for bit

The carrier frequency fc =n/Tb for some fixed integer n.

**Antipodal Signal:**

The pair of sinusoidal waves that differ only in a relative phase shift of 180° are called antipodal signals

**Algorithm**

Initialization commands

PSK modulation

1. Generate carrier signal.
2. Start FOR loop
3. Generate binary data, message signal in polar form
4. Generate PSK modulated signal.
5. Plot message signal and PSK modulated signal.
6. End FOR loop.
7. Plot the binary data and carrier.

PSK demodulation

1. Start FOR loop
2. Perform correlation of PSK signal with carrier to get decision variable
3. Make decision to get demodulated binary data. If x>0, choose ‘1’ else choose ‘0’
4. Plot the demodulated binary data.

**PROGRAM 1**

**% PSK modulation**

```
clc;
clear all;
close all;
%GENERATE CARRIER SIGNAL
Tb=1;
t=0:Tb/100:Tb;
fc=2;
c=sqrt(2/Tb)*sin(2*pi*fc*t);
%generate message signal
```

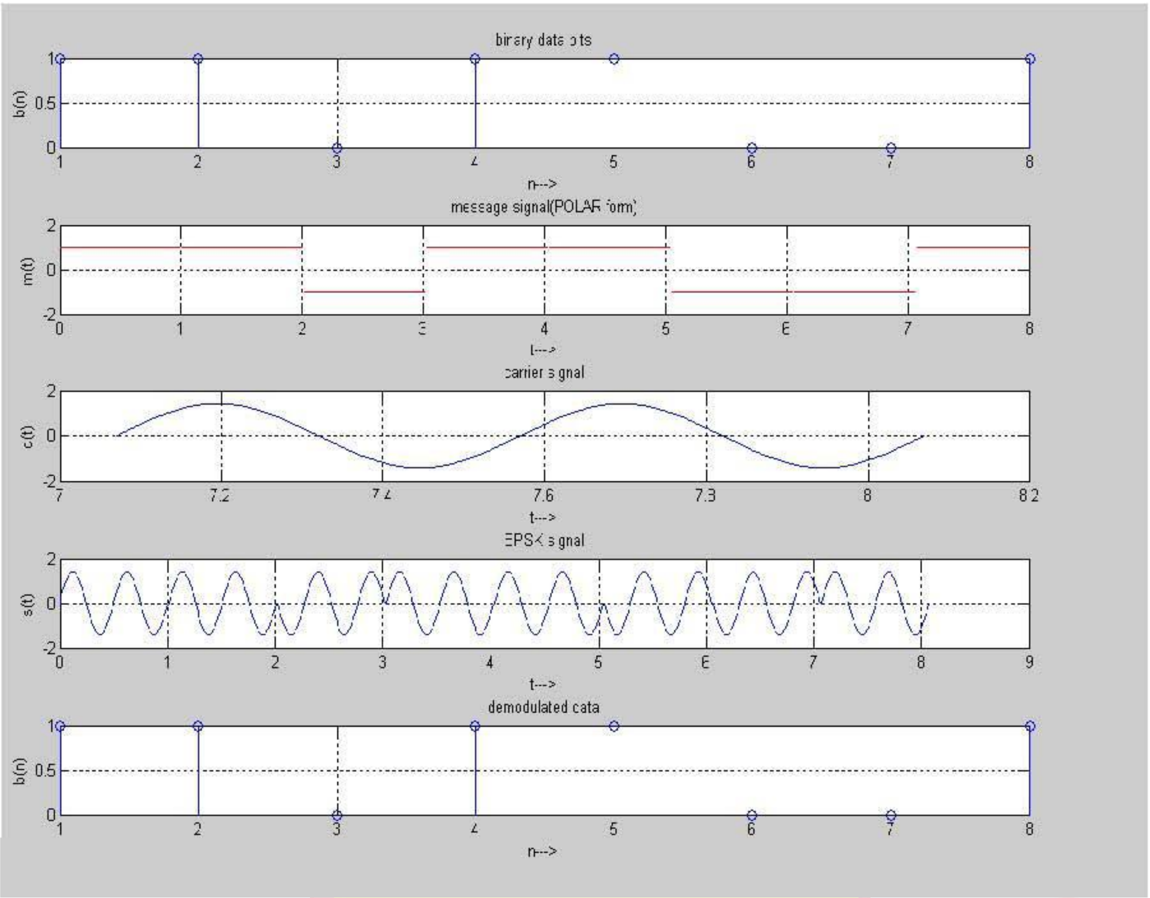
```
N=8;
m=rand(1,N);
t1=0;t2=Tb
for i=1:N
t=[t1:.01:t2]
if m(i)>0.5
m(i)=1;
m_s=ones(1,length(t));
else
m(i)=0;
m_s=-1*ones(1,length(t));
end
message(i,:)=m_s;
%product of carrier and message signal
bpsk_sig(i,:)=c.*m_s;
%Plot the message and BPSK modulated signal
subplot(5,1,2);axis([0 N -2 2]);plot(t,message(i,:),'r');
title('message signal(POLAR form)');xlabel('t--->');ylabel('m(t)');
grid on; hold on;
subplot(5,1,4);plot(t,bpsk_sig(i,:));
title('BPSK signal');xlabel('t--->');ylabel('s(t)');
grid on; hold on;
t1=t1+1.01; t2=t2+1.01;
end
hold off
%plot the input binary data and carrier signal
subplot(5,1,1);stem(m);
```

**% PSK Demodulation**

```
t1=0;t2=Tb
for i=1:N
t=[t1:.01:t2]
%correlator
x=sum(c.*bpsk_sig(i,:));
%decision device
```

```
if x>0
demod(i)=1;
else
demod(i)=0;
end
t1=t1+1.01;
t2=t2+1.01;
end
%plot the demodulated data bits
subplot(5,1,5);stem(demod);
title('demodulated data');xlabel('n--->');ylabel('b(n)');
grid on
title('binary data bits');xlabel('n--->');ylabel('b(n)');
grid on;
subplot(5,1,3);plot(t,c);
title('carrier signal');xlabel('t--->');ylabel('c(t)');
grid on;
```

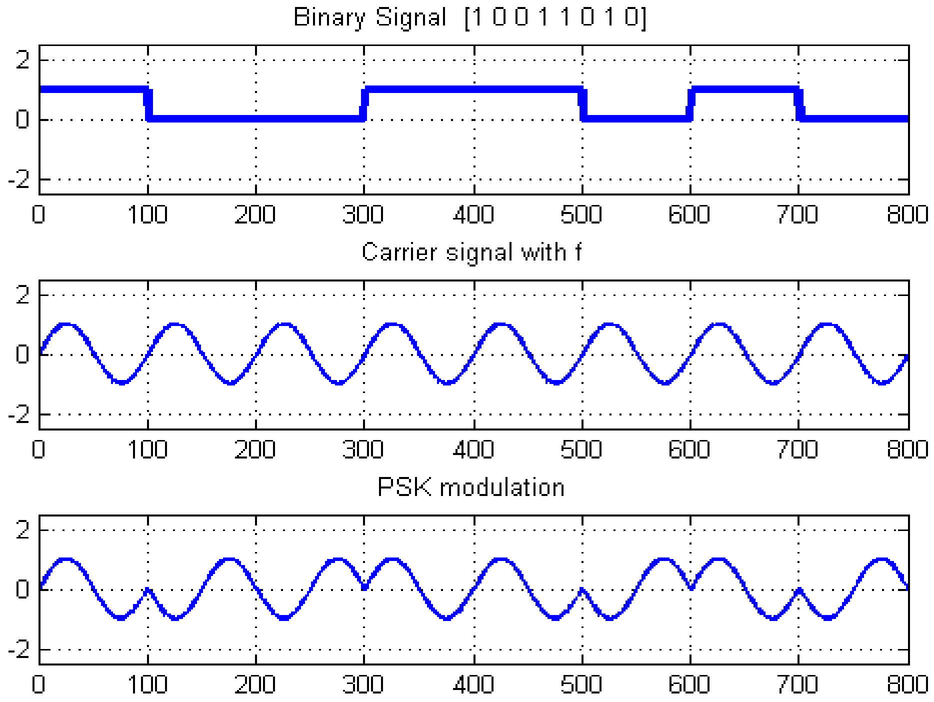
**Model graph**



**PROGRAM 2**

```
% Program for Digital Modulation PHASE SHIFT KEYING (PSK)
clc;

closeall;clear all;

f=1;

g=[1 0 0 1 1 0 1 0];

n=1000;

t1 = 0:0.01:n-0.01;

t=0:2*pi/99:2*pi;

c1=sin(2*pi*f*t1);

cp=[];sp=[];

mod=[];mod1=[];bit=[];

for n=1:length(g);

if g(n)==0;

die=-ones(1,100);

se=zeros(1,100);
```

```
else g(n)==1;
die=ones(1,100);
se=ones(1,100);
end
c=sin(f*t);
cp=[cp die];
mod=[mod c];
bit=[bit se];
end
bpsk=cp.*mod;
subplot(3,1,1);plot(bit,'LineWidth',3.5);grid on;
title('Bi        nary Signal [1 0 0 1 1 0 1 0]');
axis([0 100*length(g) -2.5 2.5]);
subplot(3,1,2);plot(c1,'LineWidth',1.5);grid on;
title('Carrier signal with f');
axis([0 100*length(g) -2.5 2.5]);
subplot(3,1,3);plot(bpsk,'LineWidth',1.5);grid on;
title('PSK modulation');
axis([0 100*length(g) -2.5 2.5]);
```



**POSTLAB:**

1. List out the Advantages of Digital Modulation over Analog Modulation
2. Define QPSK and its applications
3. Discuss about BPSK Vs QPSK
4. Depict the wave form of QPSK

**RESULT**

The program for PSK modulation and demodulation has been simulated in MATLAB and necessary graphs are plotted

## Ex.No:10 Simulation Of Combinational Circuits Using HDL

**AIM:**

To write a verilog code for half adder, full adder and multiplexer.

**TOOLS REQUIRED**:

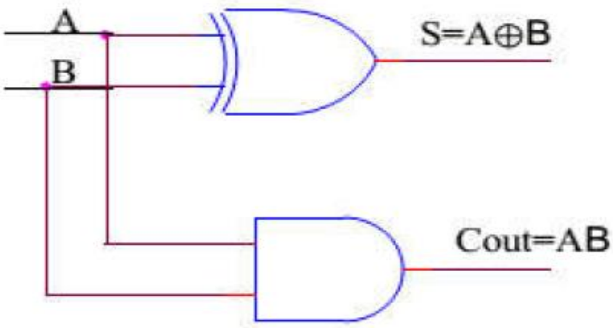
Modalism

**PRELAB:**

**VHDL**: VHDL stands for very high-speed integrated circuit hardware description language. It is a programming language used to model a digital system by dataflow, behavioral and structural style of modeling. This language was first introduced in 1981 for the department of Defense (DoD) under the VHSIC program. **levels of verilog HDL.**

- **Behavioral Level: Module** can be implemented in terms of the desired design algorithm without concern for the hardware implementation details. Very similar to C programming
- **Dataflow Level**: Module designed by specifying dataflow. The designer is aware of how data flows between hardware registers and how the data is processed in the design
- **Gate Level:** Module implemented in terms of logic gates like (and ,or) and interconnection between gates
- **Switch Level:** Module implemented with switches and interconnects. Lowest level of Abstraction

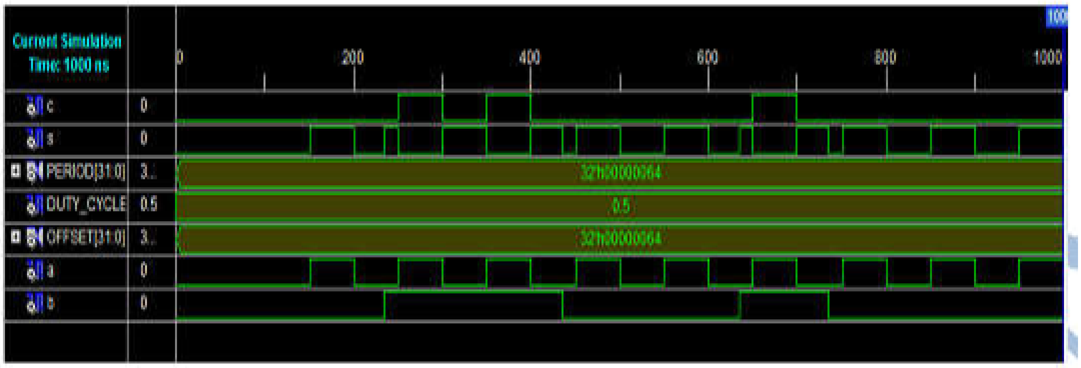
**HALF ADDER**



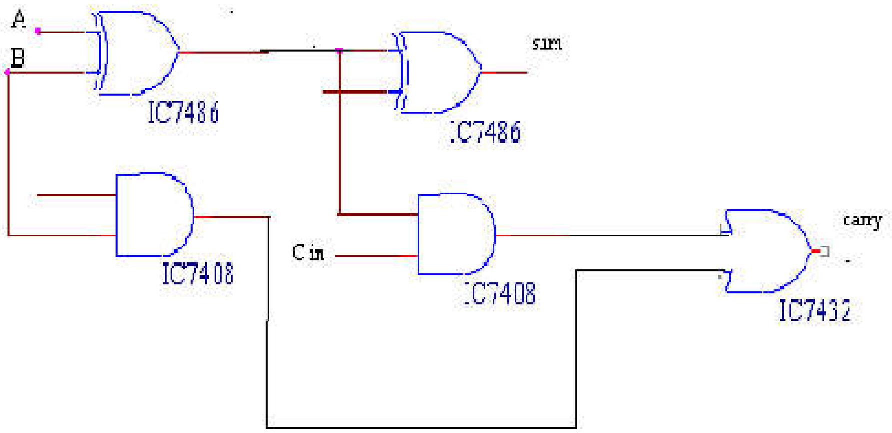
**PROGRAM**

```
module halfadder (sum, carry, A, B);
        input A, B;
        output sum, carry;
        xor g1(sum,A,B);
        and g2(carry,A,B);
endmodule
```

**SIMULATED OUTPUT:**

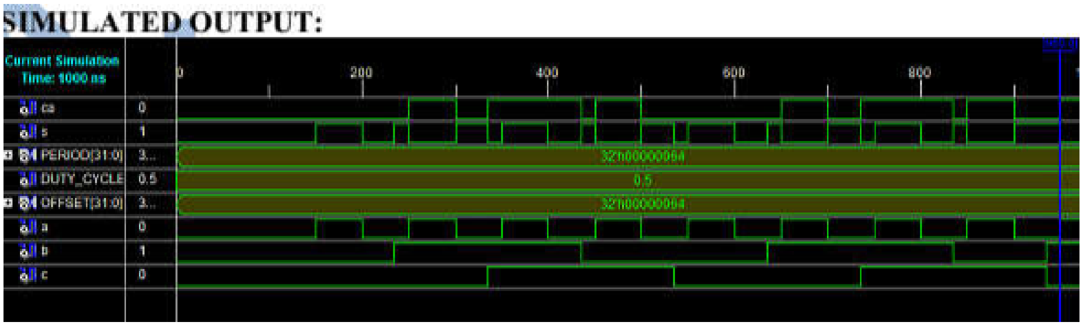


## FULL ADDER:



```
module fulladd(a, b, c, s, ca);
    input a;
    input b;
    input c;
    output s;
    output ca;
    wire  s1,c1,c2;
    xor g1(s1,a,b);
    and g2(c1,a,b);
    xor g3 (s,s1,c);
    and g4(c2,s1,c);
    or g5(ca,c1,c2);
endmodule
```

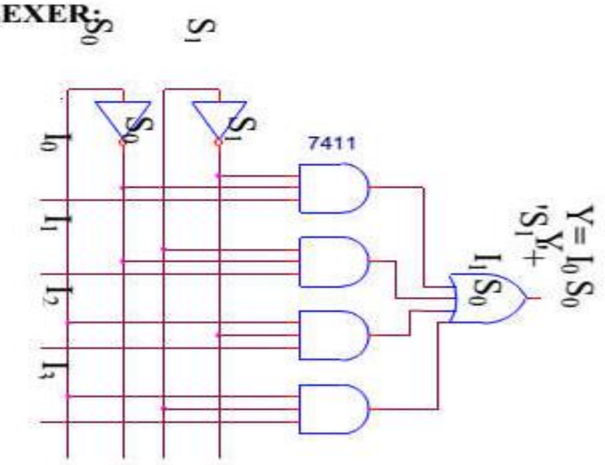
## SIMULATED OUTPUT:

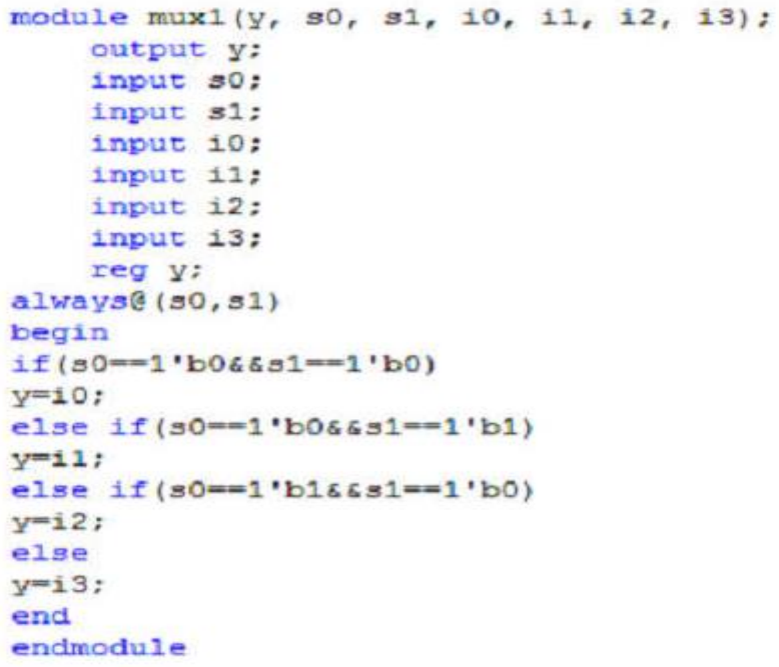


Current Simulation Time: 1000 ns

```
ca              0
s               1
PERIOD[31:0]    3...    32'h00000064
DUTY_CYCLE      0.5     0.5
OFFSET[31:0]    3...    32'h00000064
a               0
b               1
c               0
```

**MULTIPLEXER:**





```
module mux1(y, s0, s1, i0, i1, i2, i3);
     output y;
     input s0;
     input s1;
     input i0;
     input i1;
     input i2;
     input i3;
     reg y;
always@(s0,s1)
begin
if(s0==1'b0&&s1==1'b0)
y=i0;
else if(s0==1'b0&&s1==1'b1)
y=i1;
else if(s0==1'b1&&s1==1'b0)
y=i2;
else
y=i3;
end
endmodule
```

**POSTLAB:**

1. Discuss about combinational circuit vs Sequential circuit
2. Define Moore's Law
3. List out the applications of Multiplexer

**RESULT:**

Thus the verilog code for half adder, full adder and multiplexer were simulated and verified.

## Ex.No:11 SIMULATIONS OF SEQUENTIAL CIRCUITS USING HDL

**AIM**:

To write a verilog code for RS, D, JK flip flop and up counter

**TOOLS REQUIRED:**

Modalism Software

**PRELAB**:

1. Discuss the Truth table of SR Flip Flops. SR Flip Flop Truth Table

| S | R | Q(t+1) |
|---|---|---|
| 0 | 0 | Q(t) |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | Invalid inputs |

2. Discuss the Truth table of JK Flip Flops.

JK Flip Flop

Truth Table

| J | K | Q(t+1) |
|---|---|---|
| 0 | 0 | Q(t) |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | Q'(t) |

3. Discuss the Truth table of D Flip Flops.

D Flip Flop

Truth Table

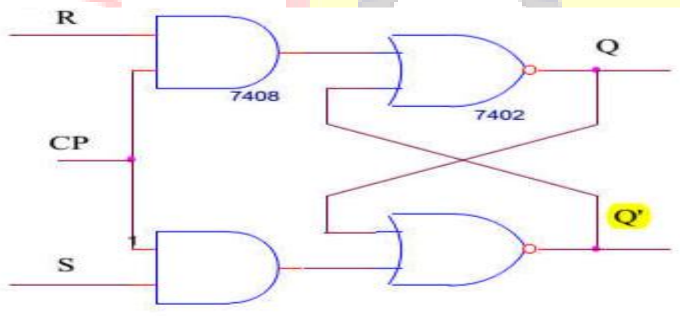
| D | Q(t+1) |
|---|---|
| 0 | 0 |
| 1 | 1 |

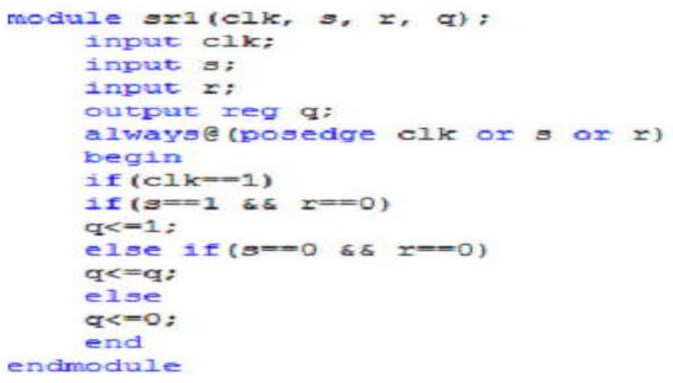
4. Discuss the Truth table of T Flip Flops.

| T | Q(t+1) |
|---|---|
| 0 | Q(t) |
| 1 | Q'(t) |

PROGRAM:

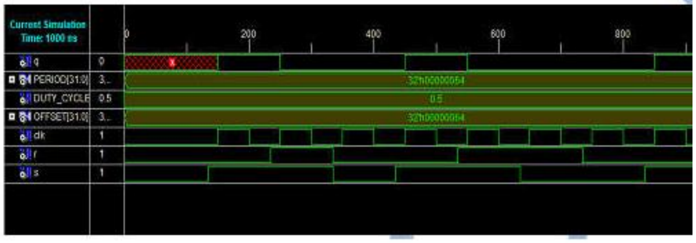
RS FLIP FLOP:



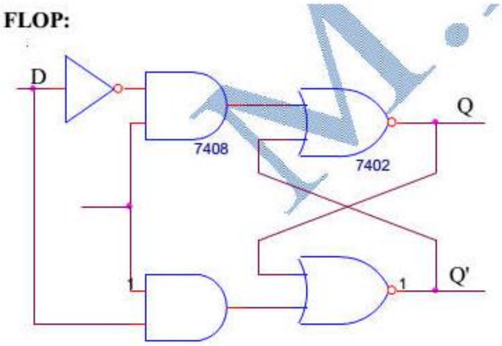


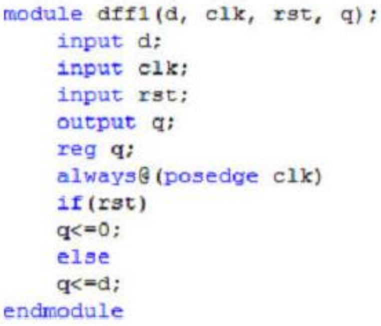
```
module sr1(clk, s, r, q);
    input clk;
    input s;
    input r;
    output reg q;
    always@(posedge clk or s or r)
    begin
    if(clk==1)
    if(s==1 && r==0)
    q<=1;
    else if(s==0 && r==0)
    q<=q;
    else
    q<=0;
    end
endmodule
```

**SIMULATED OUTPUT:**



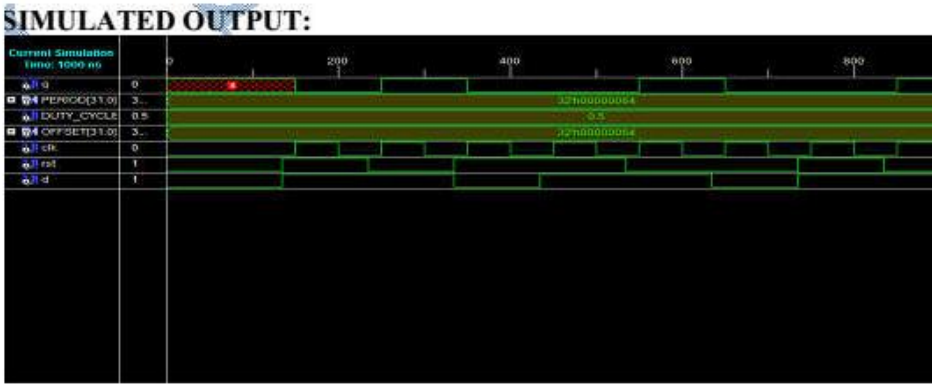
**D FLIP FLOP:**



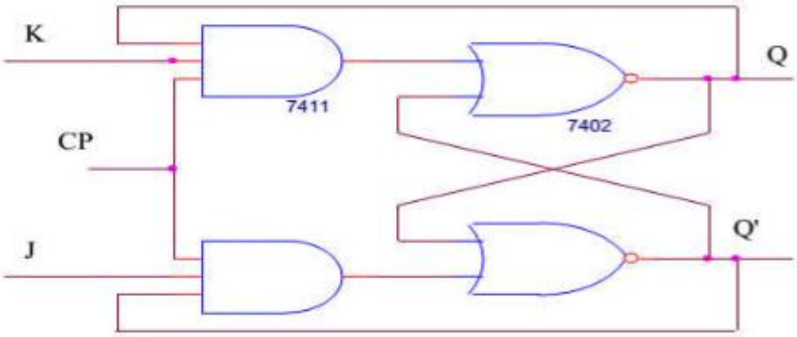


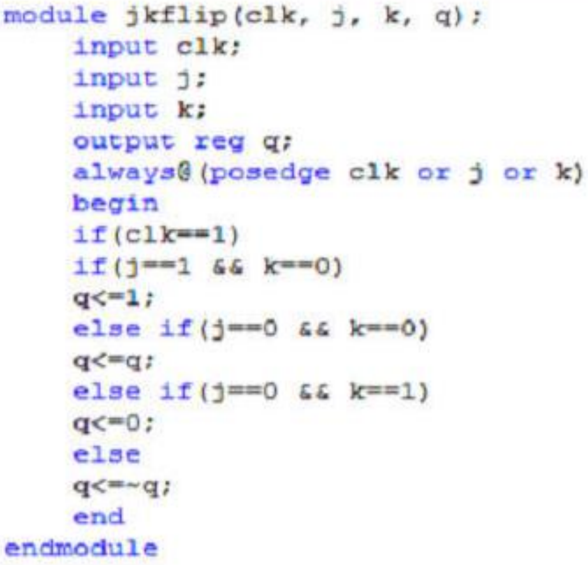
```
module dff1(d, clk, rst, q);
    input d;
    input clk;
    input rst;
    output q;
    reg q;
    always@(posedge clk)
    if(rst)
    q<=0;
    else
    q<=d;
endmodule
```

**SIMULATED OUTPUT:**



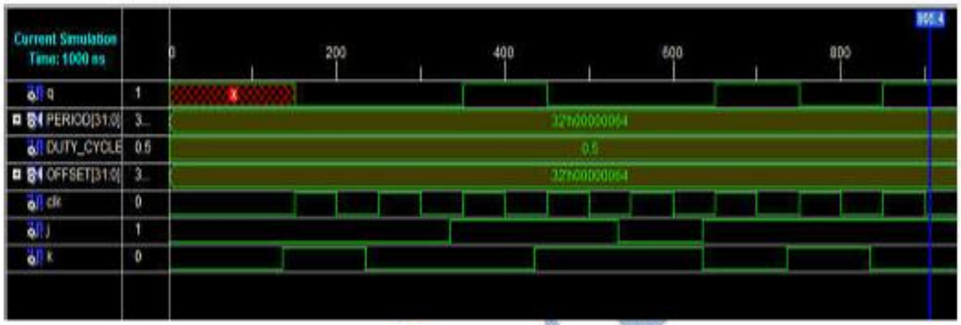
**JK FLIP FLOP:**





```
module jkflip(clk, j, k, q);
     input clk;
     input j;
     input k;
     output reg q;
     always@(posedge clk or j or k)
     begin
     if(clk==1)
     if(j==1 && k==0)
     q<=1;
     else if(j==0 && k==0)
     q<=q;
     else if(j==0 && k==1)
     q<=0;
     else
     q<=~q;
     end
endmodule
```

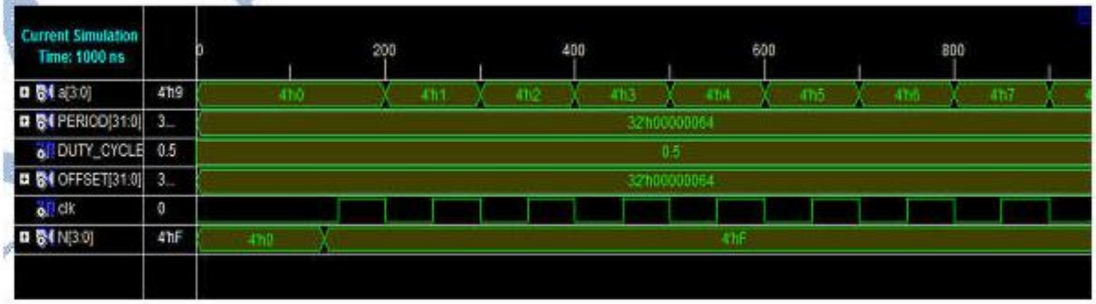
## SIMULATED OUTPUT:



## UP COUNTER:

```
module upcount(clk, N, a);
    input clk;
    input [3:0] N;
    output reg [3:0] a;
    initial a=4'b0000;
    always @(negedge clk)
    a=(a==N)?4'b0000: a+1'b1;
endmodule
```

## SIMULATED OUTPUT:



**POSTLAB**:

1. Say some other simulation tools
2. Depict the up counter and down counter output waveform diagram
3. Illustrate the construction 3-bit counter using TFlipFlop
4. Difference between synchronous vs Asynchronous counter

**RESULT**:

Thus the Verilog code for RS Flip Flop,JK,D Flip Flop and upcounter were simulated and verified.

## EX.NO. 12 Internet of Things (TBS)

### Internet of Things (IoT)

IOT is the networking of physical objects that contain electronics embedded within their architecture in order to communicate and sense interactions amongst each other or with respect to the external environment. In the upcoming years, IoT-based technology will offer advanced levels of services and practically change the way people lead their daily lives. Advancements in medicine, power, gene therapies, agriculture, smart cities, and smart homes are just a very few of the categorical examples where IoT is strongly established.

Over 9 billion ‘Things’ (physical objects) are currently connected to the Internet, as of now. In the near future, this number is expected to rise to a whopping 20 billion.

There are four main components used in IoT:

1. **Low-power embedded systems –**
   Less battery consumption, high performance are the inverse factors play a significant role during the design of electronic systems.
2. **Cloud computing –**
   Data collected through IoT devices is massive and this data has to be stored on a reliable storage server. This is where cloud computing comes into play. The data is processed and learned, giving more room for us to discover where things like electrical faults/errors are within the system.
3. **Availability of big data –**
   We know that IoT relies heavily on sensors, especially real-time. As these electronic devices spread throughout every field, their usage is going to trigger a massive flux of big data.
4. **Networking connection –**
   In order to communicate, internet connectivity is a must where each physical object is represented by an IP address. However, there are only a limited number of addresses available according to the IP naming. Due to the growing number of devices, this naming system will not be feasible anymore. Therefore, researchers are looking for another alternative naming system to represent each physical object.

**There are two ways of building IoT:**

1. Form a separate internetwork including only physical objects.
2. Make the Internet ever more expansive, but this requires hard-core technologies such as rigorous cloud computing and rapid big data storage (expensive)

**Characteristics of IoT:**

- Massively scalable and efficient
- IP-based addressing will no longer be suitable in the upcoming future.
- An abundance of physical objects is present that does not use IP, so IoT is made possible.
- Devices typically consume less power. When not in use, they should be automatically programmed to sleep.
- A device that is connected to another device right now may not be connected in another instant of time.
- Intermittent connectivity – IoT devices aren't always connected. In order to save bandwidth and battery consumption, devices will be powered off periodically when not in use. Otherwise, connections might turn unreliable and thus prove to be inefficient.

As a quick note, IoT incorporates trillions of sensors, billions of smart systems, and millions of applications.

**Modern Applications:**

1. Smart Grids
2. Smart cities
3. Smart homes
4. Healthcare
5. Earthquake detection
6. Radiation detection/hazardous gas detection
7. Smartphone detection
8. Water flow monitoring