CONTENTS

EXPERIMENT NO:1

FAMILIARISATION OF IC TRAINER KIT AND DIGITAL ICs

AIM

To familiarize with the digital IC trainer kit and logic gate IC packages. Also to verify the truth tables of the logic gates.

COMPONENTS REQUIRED

1) IC Trainer kit
2) IC- 7400
3) IC-7402
4) IC-7404
5) IC-7408
6) IC-7432
7) IC-7486

THEORY

IC trainer kit:

IC trainer kit is laboratory equipment mainly used to set up and test digital circuits while doing experiments. A typical trainer kit consists of a breadboard, IC sockets, a series of input switches, a series of output LED's, a section to provide clock signals and a voltage source. ICs can be fitted in the sockets or breadboard and they can be powered from the voltage source. The frequency of the clock can be selected by turning the knob into different directions. In order to feed mono pulses manually, a debouncer switch is also provided. The number – select switches in the input provide 0 or 1 state voltage for digital inputs. LED's in the output glow to visualize the high states in the digital circuits.

Logic gates:

In digital electronics, a gate is a logic circuit with one output and one or more inputs. Logic gates are available as ICs.

AND gate:

The AND gate performs logical multiplication, more commonly known as AND operation. The AND gate output will be in high state only when all the inputs are in high state. 7408 is a digital IC in the TTL family and contains four AND gates. For this reason, it is called quad two input AND gate. Every AND gate has two inputs in this Dual-in-Line Package (DIP). 14 is the supply pin. For

standard TTL devices to work properly, the supply voltage level must be +4.75V and +5.25V. This is why +5V is the nominal supply voltage specified for all TTL devices. Pin 7 is the common ground for the chip. The other pins are inputs and outputs.

OR gate:

It performs logical addition. Its output will become high if any of the inputs is in logic high. 7432 is a quad two input OR gate.

NOT gate:

It performs a basic logic function called inversion or complementation. The purpose of the inverter is to change one logic level to opposite level. IC 7404 is a hex inverter.

NAND gate:

A NOT gate following an AND gate is called NOT-AND or NAND gate. Its output will be low if all the inputs are in high state. 7400 IC is a quad two input NAND gate.

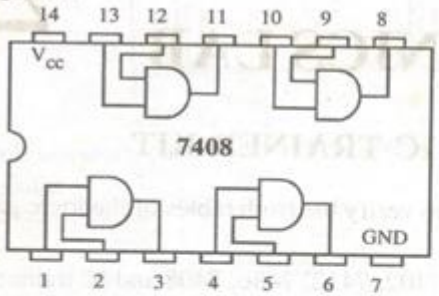
NOR gate:

A NOT gate following and OR gate is called NOT-OR or NOR gate. Its output will be in low state if any of its inputs is in high state. 7402 is a quad two input NOR gate.

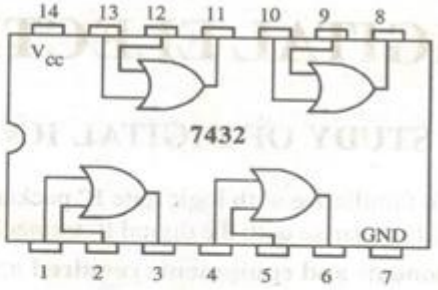
XOR gate:

Its output will be high if and only if one input is in high state. 7486 is a quad two input XOR gate.

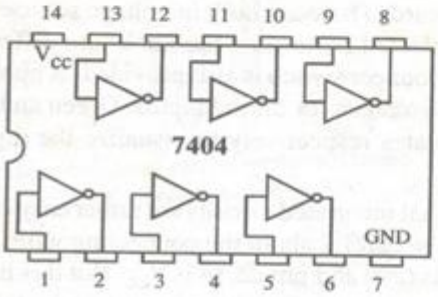
DIP pin out



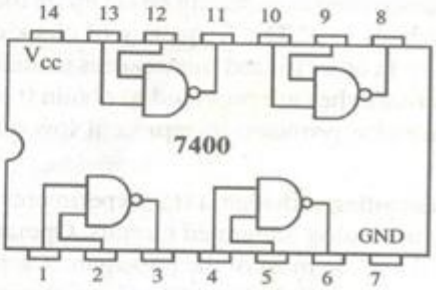
Quad two input AND gate



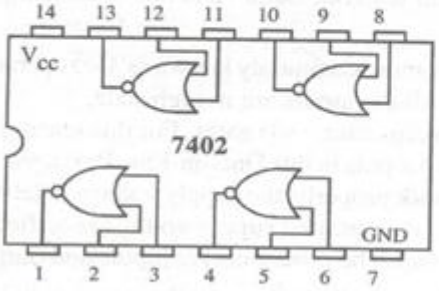
Quad two input OR gate



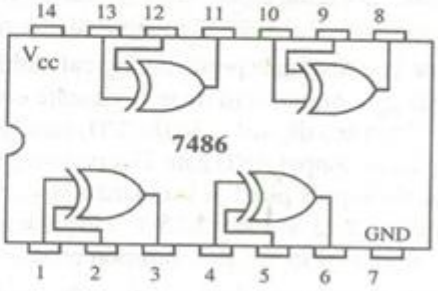
HEX inverter gates



Quad two input NAND gates



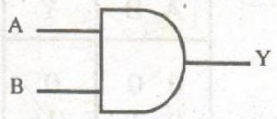
Quad two input NOR gates



Quad two input XOR gates

Symbols and truth tables

AND gate

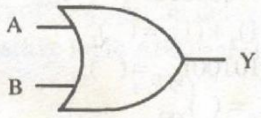


Logic diagram

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Truth table

OR gate

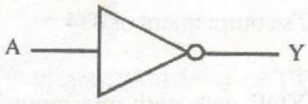


Logic diagram

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Truth table

NOT gate

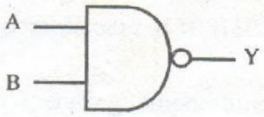


Logic diagram

A	Y
0	1
1	0

Truth table

NAND gate

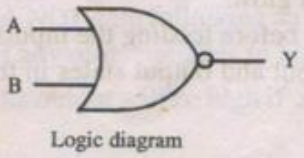


Logic diagram

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Truth table

NOR gate



Logic diagram

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

XOR gate

Logic diagram

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Truth table

PROCEDURE

1. Test all the components and IC packages
2. Verify the pinouts of the ICs before feeding the inputs.
3. Set up the circuit. Enter the input through toggle switches in the trainer kit and observe the LED outputs.
4. Enter the output states in truth table corresponding to the input conditions.

RESULT

Famialirised with the digital IC trainer kit and logic IC packages.Also Verified the truth tables of the logic gates.

EXPERIMENT NO:2

REALIZATION OF FUNCTION USING BASIC & UNIVERSAL GATES

AIM

To design and realize functions using basic gates and universal gates in SOP and POS forms.

COMPONENTS REQUIRED

IC Trainer kit

1) IC-7400

2) IC-7408

3) IC-7404

4)IC- 7432

THEORY

Boolean Functions can be represented easily in SOP (sum of products) form and POS (product of sums) form. To represent these standardized equations logically, we use the logic gates.Any Boolean function can be represented by using a number of logic gates by properly interconnecting them. Logic gates implementation or logic representation of Boolean functions is very simple and easy form.

SOP Boolean Function Implementation using Basic Gates:

The sum of product or SOP form is represented by using basic logic gates: AND gate and OR gate. The SOP form implementation will have AND gates at its input side and as the output of the function is the sum of all product terms, it has an OR gate at its output side.An important to remember is that we use NOT gate to represent the inverse or complement of the variables.

$$\mathbf{F = AB + A\,\overline{B}}$$

POS Boolean Function Implementation using Basic Gates:

The product of sums or POS form can be represented by using basic logic gates like AND gate and OR gates. The POS form implementation will have the OR gate at its input side and as the output of the function is product of all sum terms, it has AND gate at its output side. In POS form implementation, we use NOT gate to represent the inverse or complement of the variables.

$$\mathbf{F = (A + B) \,.\, (A + \overline{B})}$$

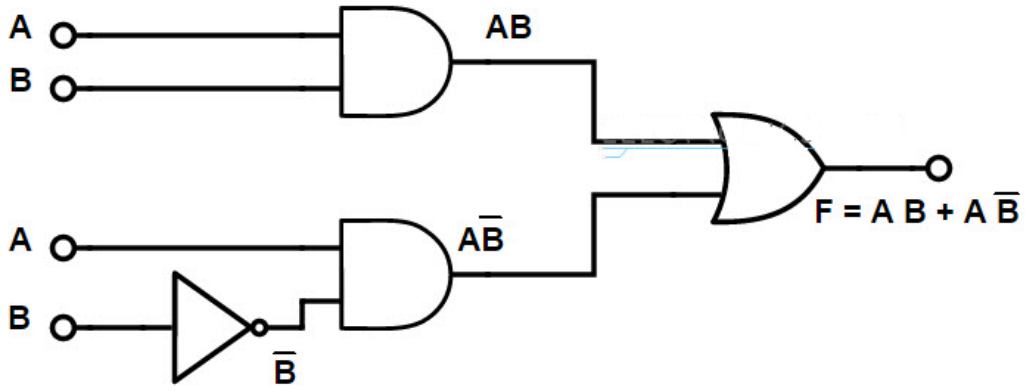
Implementation of Boolean functions using Universal Gates:

NAND gate and NOR gate are considered to be 'Universal Logic Gates'. The reason behind this is, NAND gate and NOR gate can perform (or can function like) all the 3 basic gates, such as AND gate, OR gate and NOT gate. We can design any basic logic gate by using NAND gate or NOR gate. This is why they are called as "Universal Gates".

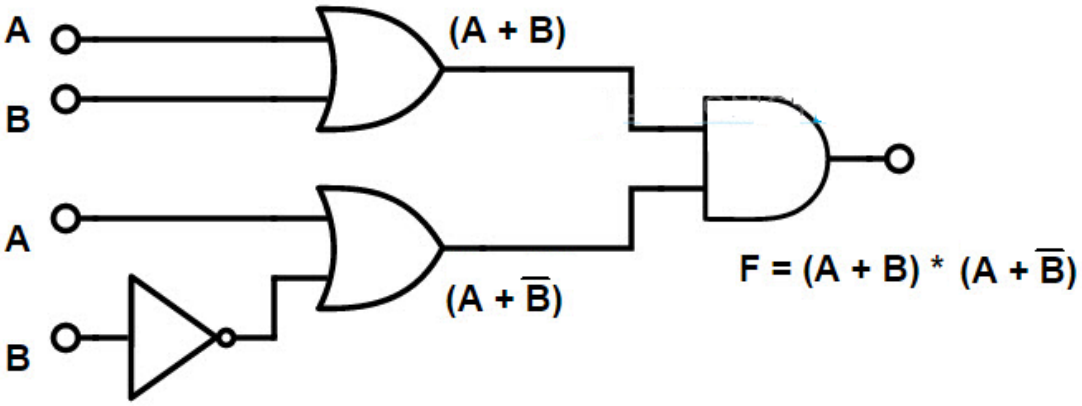
In NAND gate implementation, we use NAND gates at both input and output side.NAND gate is a logical combination of AND gate and NOT gate and this can function like AND gate, OR gate and NOT gate. So, we use NAND gates to implement the Boolean function.The important thing to remember about NAND gate is this is the inverse of basic AND gate. This means the output of the NAND gate is equal to the complement of the output of the AND gate.

$$\mathbf{F = A + (\overline{B} + C) \,.\, (\overline{D} + B\overline{E})}$$

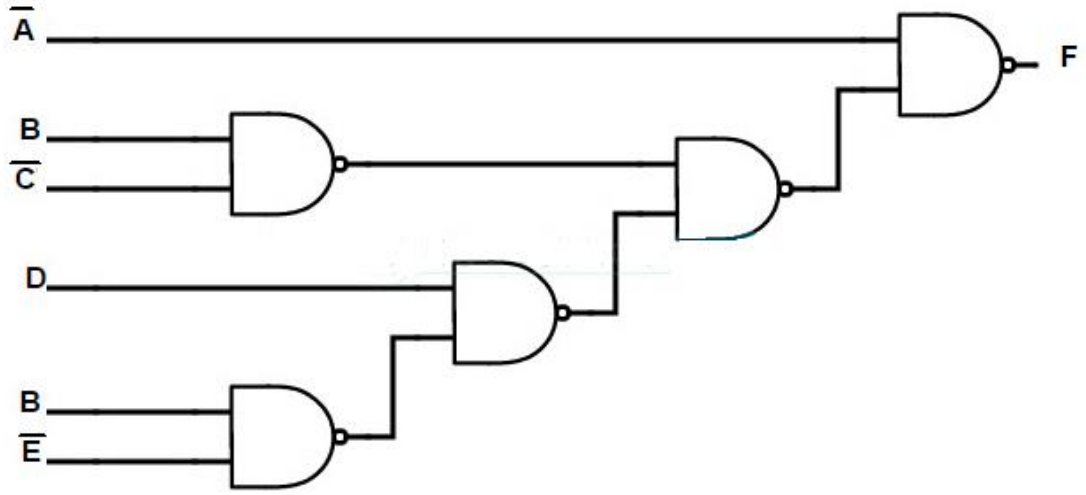
CIRCUIT DIAGRAM



SOP Boolean Function Implementation using Basic Gates



POS Boolean Function Implementation using Basic Gates



Implementation of Boolean function using NAND Gates only

PROCEDURE

1. Test all the components and IC packages
2. Set up the circuit. one by one .
3. Enter the input through toggle switches in the trainer kit and observe the LED outputs.
4. Enter the output states in truth table corresponding to the input conditions.

RESULT

Designed the circuit for the given boolean functions and realized functions using basic gates and universal gates in SOP and POS forms.

EXPERIMENT NO:3
ADDERS AND SUBTRACTORS

AIM

To design and realize half adder /full adder and half subtractor and full subtractor using basic gates and universal gates.

COMPONENTS REQUIRED

IC Trainer kit

1) IC-7400
2) IC-7486
3) IC-7408
3) IC-7404

THEORY

Adders:

The simplest binary adder is called a half adder. Half adder has two input bits and two output bits. One output bit is the sum and the other is carry. They are represented by S and C respectively in the logic symbol.

A half adder has no provision to add a carry from the lower order bits when binary numbers are added. When two input bits and a carry are to be added, the number of input bits becomes three and the input combinations increases to eight. For this, a full adder is used. Like half adder, it also has a sum bit and a carry bit. The new carry generated is represented by Cn and carry generated from the previous addition is represented by $Cn_{-1.}$

Subtractors:

The simplest binary subtractor is called a half subtractor. Half subtractor has two input bits and two output bits. One output bit is the difference and the other is borrow. They are represented by D and B respectively in the logic symbol.

A half subtractor has no provision to borrow from the higher order bits when binary numbers are subtracted. In such a condition, the number of input bits becomes three and the input combinations increases to eight. For this, a full subtractor is used. Like half subtractor, it also has a difference bit and a borrow bit.

CIRCUIT DIAGRAM

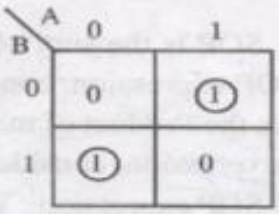
ADDERS

Half Adder

Input		Output	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

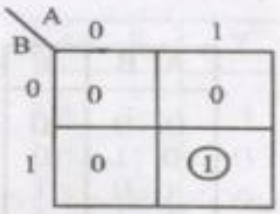
Truth table

Realization of sum and carry using Karnaugh map

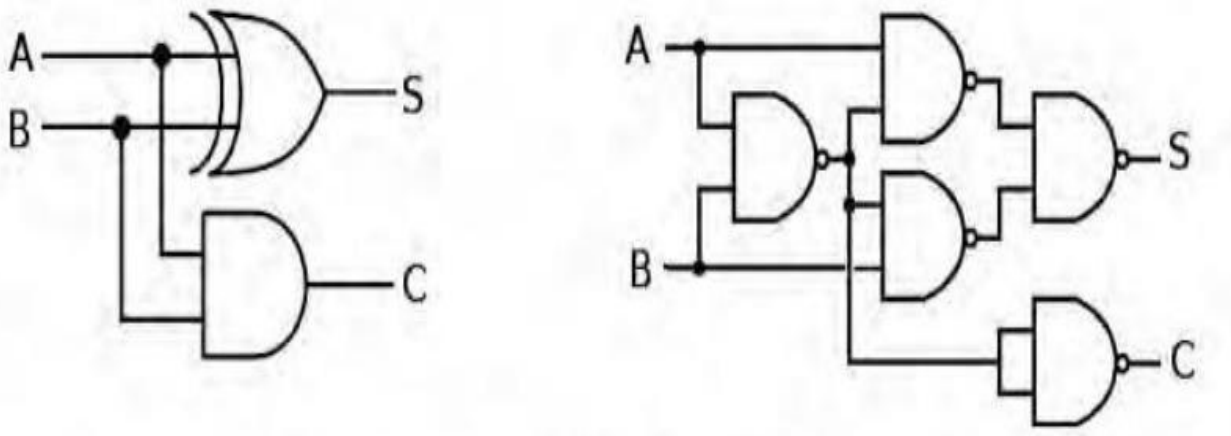


$S = A\bar{B} + \bar{A}B$

$= A \oplus B$

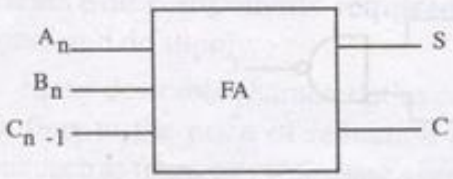


carry = A.B



Logic circuit of Half adder and Haff adder using NAND gate only

Full adder

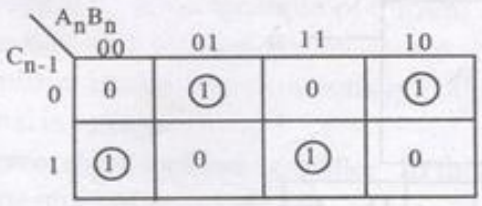


Logic symbol

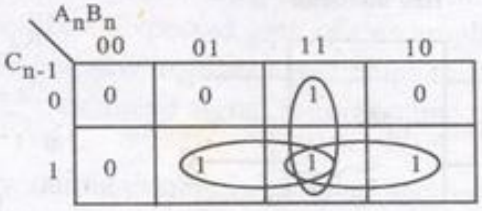
Input			Output	
A_n	B_n	C_{n-1}	S_n	C_n
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Truth table

Realisation of sum and carry using Karnaugh map

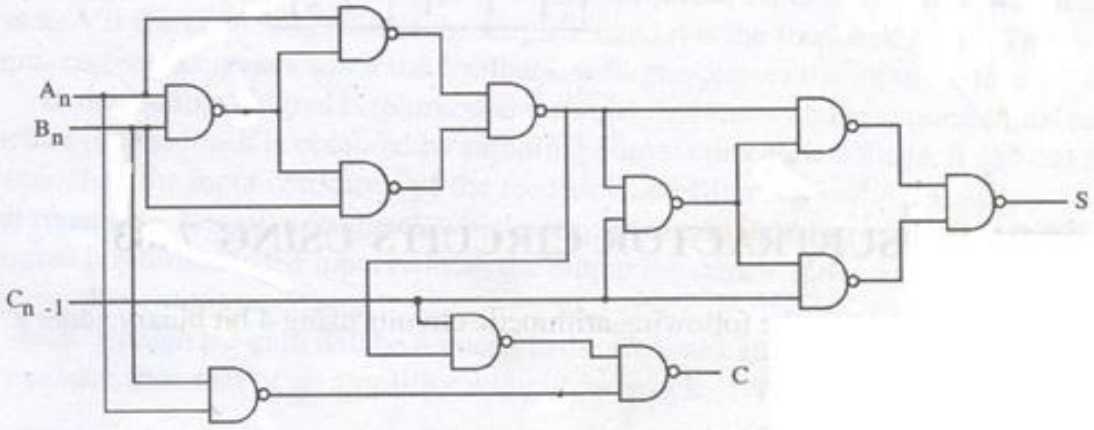


$$S_n = \overline{A}_n\overline{B}_nC_{n-1} + \overline{A}_nB_n\overline{C}_{n-1} + A_nB_nC_{n-1} + A_n\overline{B}_n\overline{C}_{n-1}$$
$$= A_n \oplus B_n \oplus C_{n-1}$$

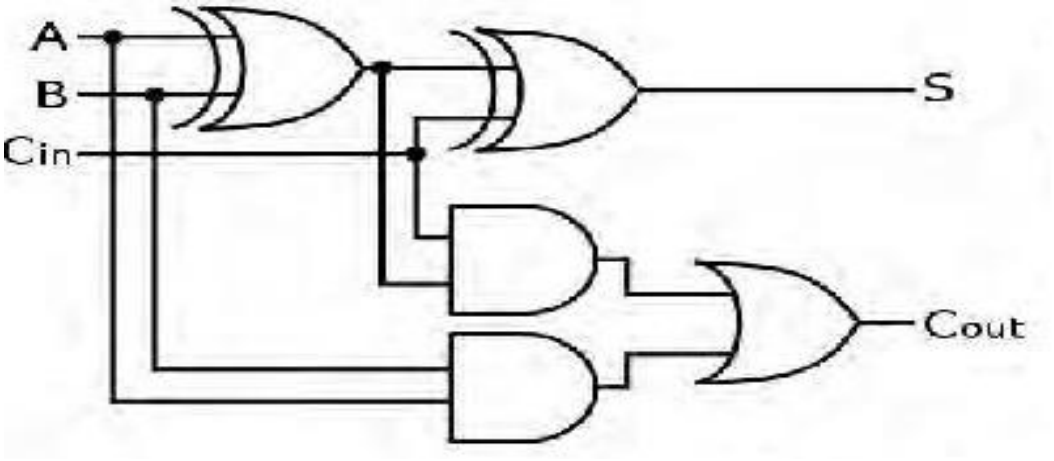


$$C_n = A_nB_n + B_nC_{n-1} + A_nC_{n-1} \text{ or}$$
$$= A_nB_n + A_n\overline{B}_nC_{n-1} + \overline{A}_nB_nC_{n-1} \text{ when horizontal grouping is avoided.}$$
$$= A_nB_n + C_{n-1}\cdot(A_n \oplus B_n)$$

Realisation using logic gates



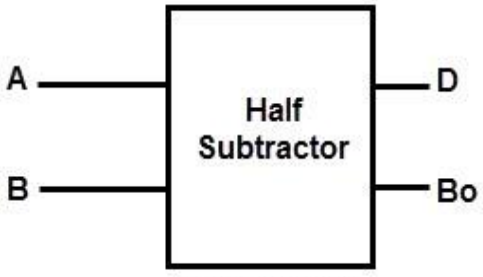
Full adder using NAND gates only



Logic circuit of Full Adder

SUBTRACTOR

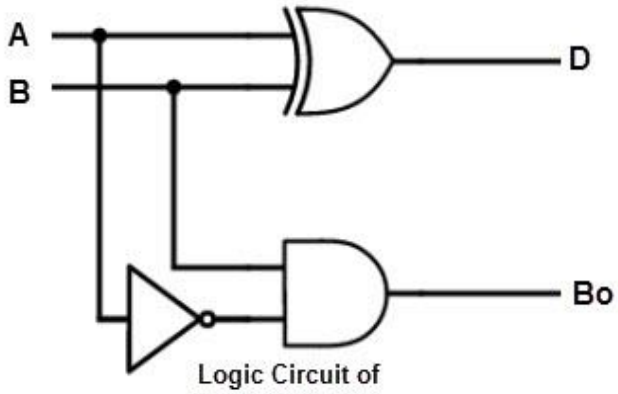
Half Subtractor



Block Diagram

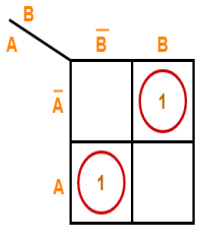
A	B	D	B_o
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Truth Table



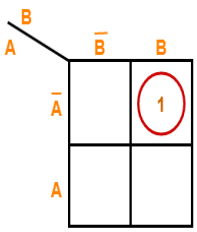
Logic Circuit of
Half Subtractor

For D:



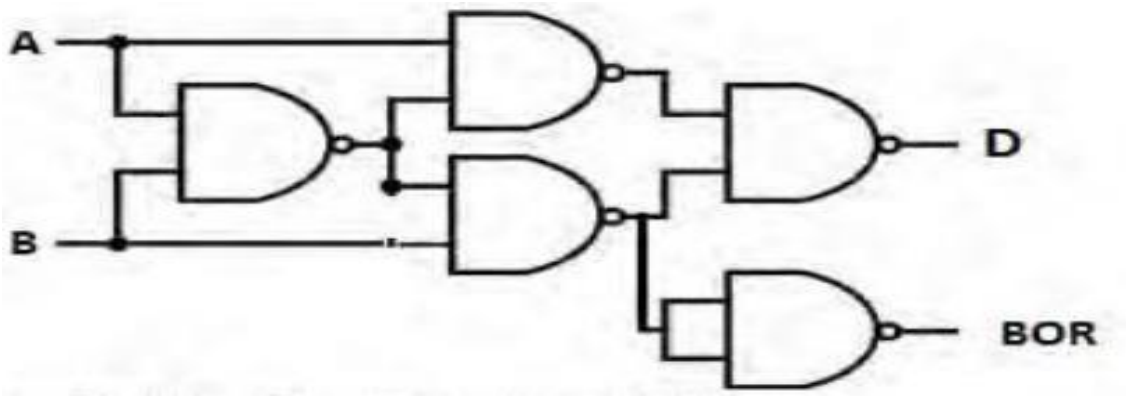
D = A ⊕ B

For b:



b = Ā.B

K Maps



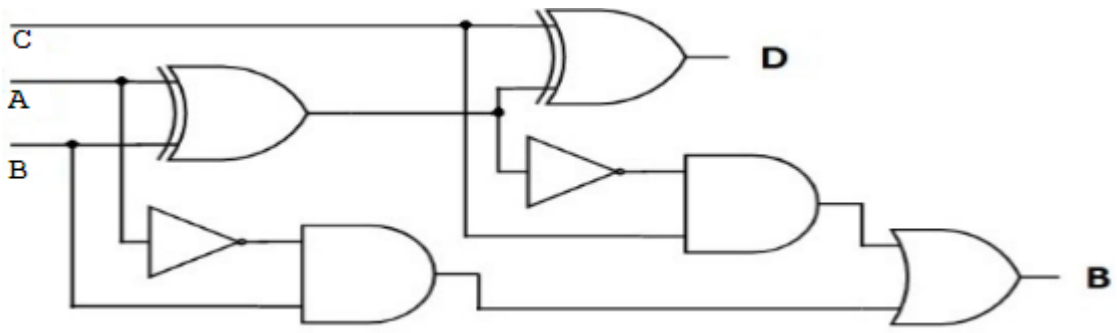
Half Subtractor using NAND Gates

Full subtractor

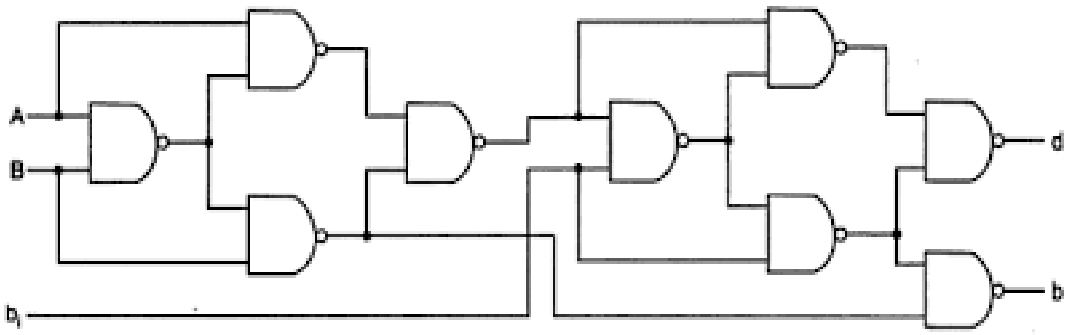
Block Diagram of Full Subtractor

A	B	B_{in}	D	B_o
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Truth Table



Full subtractor using logic gates



Logic diagram of a full-subtractor using only 2-input NAND gates.

PROCEDURE

1. Test all the components and IC packages
2. Set up the half adder circuit and feed the input bit combinations.
3. Observe the output corresponding to input combinations on LEDs and enter it in the truth table.
4. Repeat the above steps for the full adder, half subtractor and full subtractor circuits.

RESULT

Designed and set up half adder, full adder, half subtractor and full subtractor using basic gates and universal gates.Checked and verified their outputs.

EXPERIMENT NO:4

CODE CONVERTERS

AIM To design and set up the following circuits:

1. A four bit binary to Gray code converter.
2. A four bit Gray to binary code converter.
3. BCD to Excess 3 code converter

COMPONENTS REQUIRED

1) IC-7486
2) IC-7432
3) IC-7404
4) IC-7408
5) IC trainer kit.

THEORY

Binary code to gray code converter

To convert a binary number to corresponding Gray code, the following rules are applied:

1. The MSB in the Gray code is the same as the corresponding bit in a binary number.
2. Going from left to right, add each adjacent pair of binary digits to get the next Gray code digit. Disregard carries.

As the first step to design a binary to Gray code converter, set up a truth table with binary numbers $B_3B_2B_1B_0$ and corresponding Gray code numbers $G_3G_2G_1G_0$. Set up a circuit realizing the simplified logic expressions obtained using K maps for Gs as the functions of Bs.

Gray code to binary code converter

To convert from Gray code to binary, the following rules are applied:

1. The most significant digit in the binary number is the same as the corresponding digit in the Gray

code.

2. Add each binary digit generated to the Gray code digit in the next adjacent position. Disregard carries.

To design the Gray to binary code converter, set up the truth table and get simplified expressions using K maps for each binary bit as a function of Gray code bits. Each Gray code number differs from the preceding number by a single bit.

BCD to EXCESS 3 Code Converter

The Excess-3code for a decimal digit is the binary combination corresponding to the decimal digit plus 3.

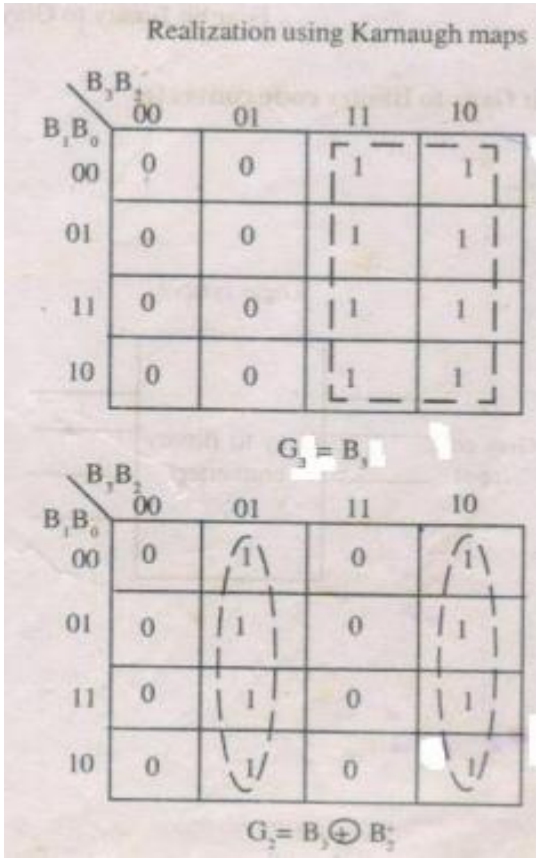
For example, the excess-3 code for decimal digit 5 is the binary combination for 5+3=8, which is 1000.Each BCD digit four bits with the bits with the bits, from most significant to least significant, labeled A,B,C,D. As the same for excess-3

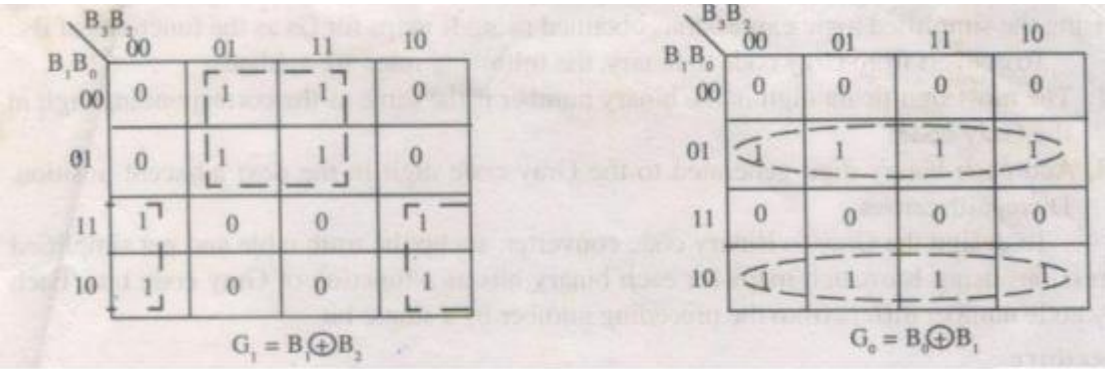
CIRCUIT DIAGRAM

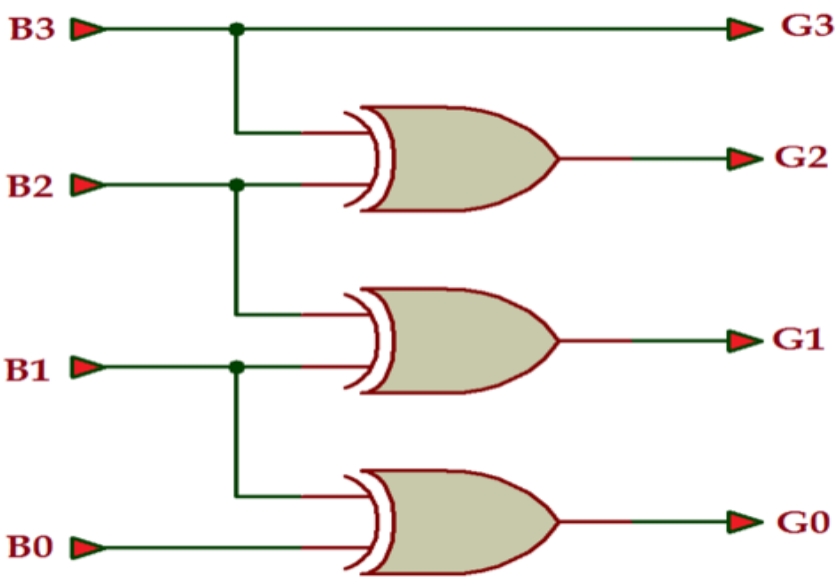
Binary to gray code converter

BINARY INPUT				GRAY CODE OUTPUT			
B3	B2	B1	B0	G3	G2	G1	G0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

Truthtable




B3B2
B1B0
00 01 11 10
00 0 1 1 0
01 0 1 1 0
11 1 0 0 1
10 1 0 0 1
G1 = B1⊕B2
B3B2
B1B0
00 01 11 10
00 0 0 0 0
01 1 1 1 1
11 0 0 0 0
10 1 1 1 1
G0 = B0⊕B1


B3
B2
B1
B0
G3
G2
G1
G0

Gray to Binary code converter

GRAY CODE INPUT				BINARY OUTPUT			
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

Truthtable

Realisation using K-maps

G_1G_0 \ G_3G_2	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	0	0	1	1
10	0	0	1	1

$B_3 = G_3$

G_1G_0 \ G_3G_2	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	1	0	1	0
10	1	0	1	0

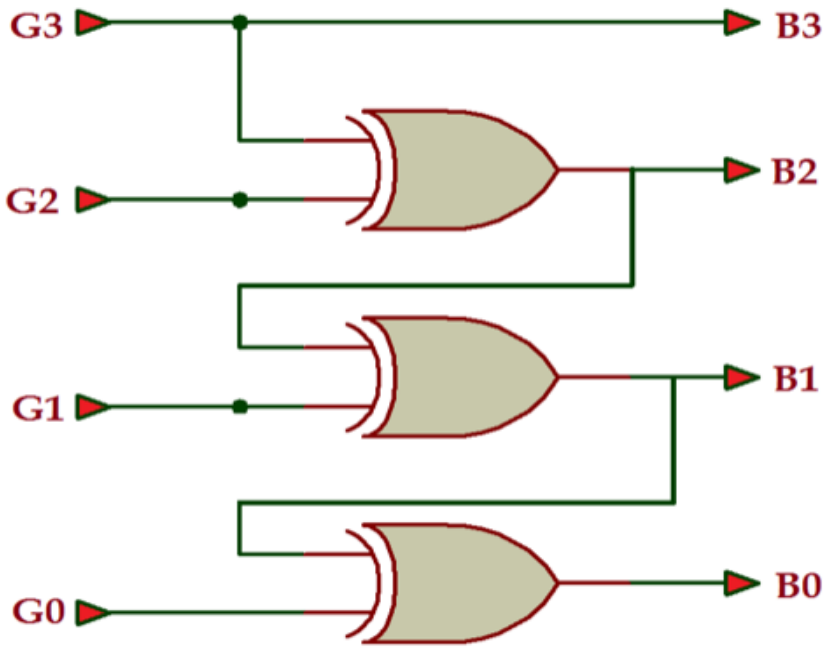
$B_1 = G_1 \oplus G_2 \oplus G_3$

G_1G_0 \ G_3G_2	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

$B_2 = G_3 \oplus G_2$

G_1G_0 \ G_3G_2	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

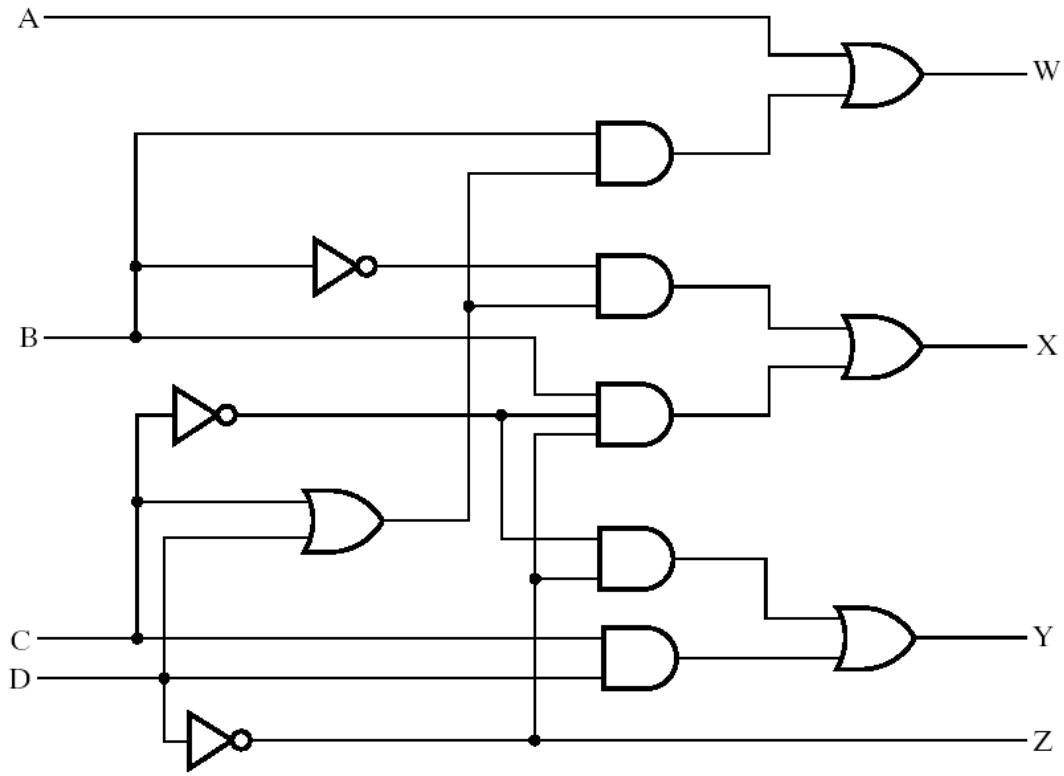
$B_0 = G_1 \oplus G_2 \oplus G_3 \oplus G_0$



BCD to EXCESS 3 Code Converter

Truth Table - BCD to Excess-3 code

Decimal Digit	Input BCD				Output Excess-3			
	A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0



PROCEDURE

For Binary code to gray code converter

1. Test all the components and IC packages
2. Set up the circuit of binary code to gray code converter.
3. Connect the outputs in sequence to LEDs.
4. Feed various 4 bit binary inputs to the circuit through toggle switches on the trainer kit.
5. Observe and verify the corresponding gray code outputs

For Gray code to Binary code converter

1. Modify the circuit to set up a gray code to binary code converter.
2. Feed gray codes to the inputs of the circuit.
3. Observe and verify the corresponding binary code outputs.

For BCD to EXCESS 3 Code Converter

1. Set up the circuit of BCD to EXCESS 3 Code Converter
2. Feed various 4 bit binary inputs to the circuit through toggle switches on the trainer kit.
3. Observe and verify the corresponding outputs

RESULT

Studied and verified the working of a binary to gray code converter,gray to binary code converter and BCD to Excess 3 code converter.

EXPERIMENT NO:5

FOUR BIT ADDER/ SUBTRACTOR and BCD ADDER USING IC 7483

AIM

To design and implement the following circuits using IC 7483.

1. 4 bit adder/ subtractor circuit.
2. BCD Adder

COMPONENTS REQUIRED

1) IC-7483
2) IC-7486
3) IC-7410
4) IC-7400
5) IC trainer kit.

THEORY

The 7483 is a TTL IC with four full adders in it. This means that it can add nibbles. In a four bit binary adder, $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$ are inputs and $S_3S_2S_1S_0$ is the output. CARRY IN pin is grounded. This circuit is also called a nibble adder.

In a **4 bit binary adder/ subtract circuit**, to add the nibbles SUB is to be made 0. To subtract $B_3B_2B_1B_0$ from $A_3A_2A_1A_0$, SUB is to be made 1. EXOR gates function as controlled inverters. When SUB = 1, $B_3B_2B_1B_0$ is complemented. Now $A_3A_2A_1A_0$, complemented version of $B_3B_2B_1B_0$, and 1 at Cin pin are added together. Cout is ignored. Thus, the 2's complement of subtrahend is added with minuend. If minuend is less than subtrahend, the obtained output will be the 2's complement of difference. For example,

10 – (minuend)	Binary equivalent of 10	1010 +
9 (subtrahend)	2's complement of 9	0111
1 (result)	Result	1 0001 omit carry

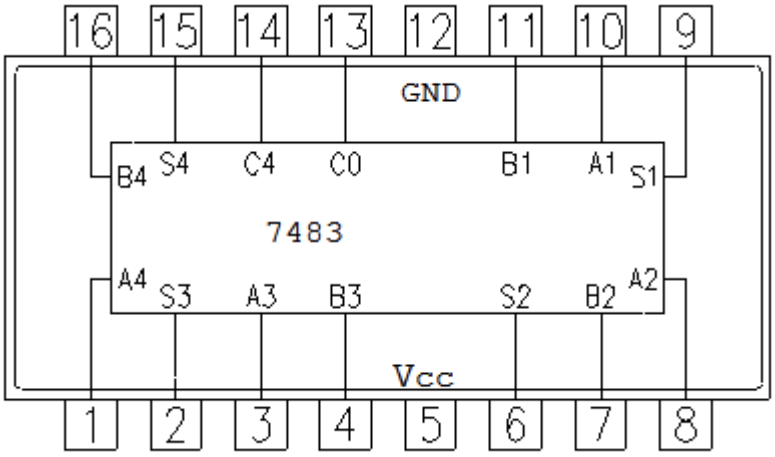
When minuend < subtrahend:

9 –	Binary equivalent of 9	1001 +
10	2's complement of 10	0110
- 1 (result)	Result	1111

In **BCD addition**, if the sum exceeds 9(i.e., 1001), 6(i.e., 0110) must be added to the result to convert it into BCD number. For this purpose, two 7483 ICs are required. One for binary addition and the other for the addition of 6. A combinational circuit is set up so that its output Cout makes $A_3A_2A_1A_0$ = 0110 when $S_3S_2S_1S_0$ > 1001 or Cout = 1.

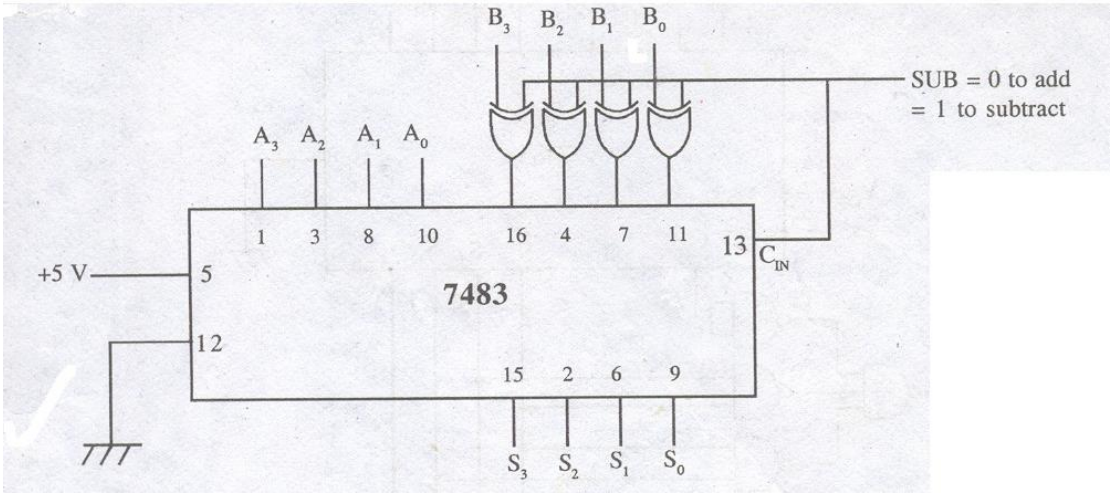
Since the maximum BCD sum is 18, most significant BCD digit of the sum output needs only one bit and it is taken from C as shown in figure.

CIRCUIT DIAGRAM

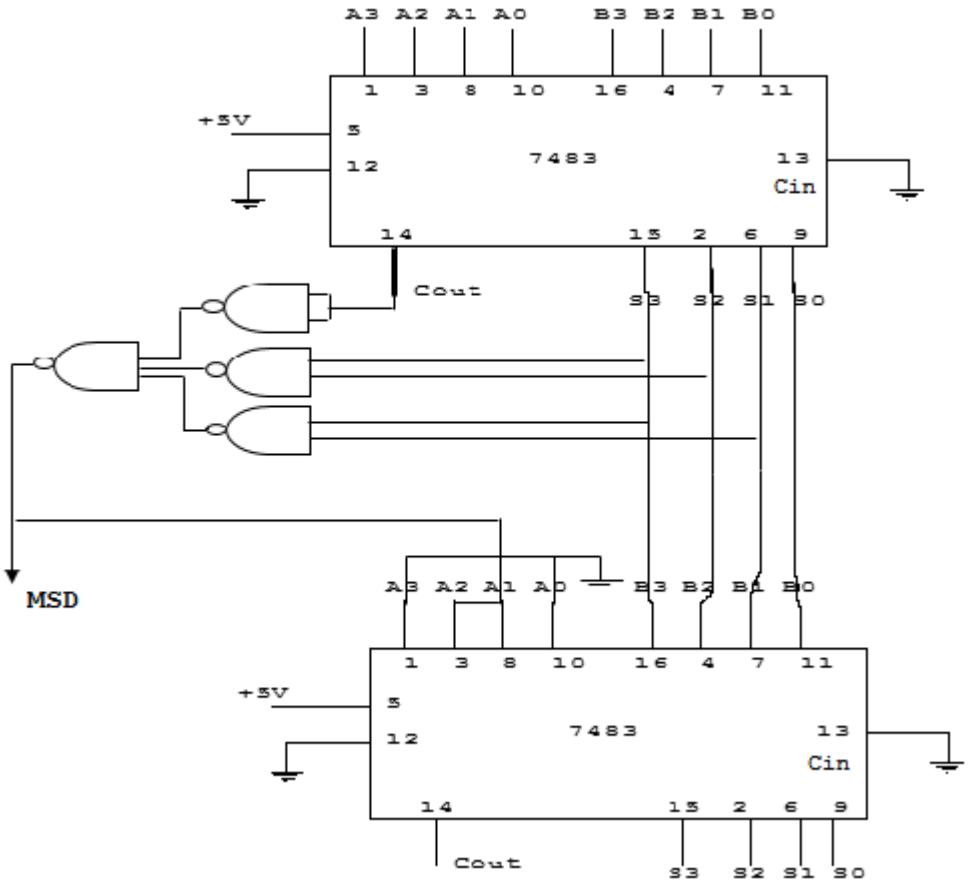


Pinout of IC 7483

4 bit add/substract circuit



BCD ADDER



PROCEDURE

For 4 bit adder/ subtractor

1. Test all the components and IC packages

2. Set up the adder/ subtract circuit as per the diagram.

3. Make SUB = 0 and verify whether it works as a nibble adder.

4. To function as a subtractor, make SUB = 1, $A_3A_2A_1A_0$ = 1010 and $B_3B_2B_1B_0$ = 1001. Verify whether the $S_3S_2S_1S_0$ = 0001.

5. Keeping SUB = 1 reverse the values of nibbles and verify the output = 1111 = 2's complement of 0001.Repeat with other nibbles.

For BCD Adder

1. Set up the BCD adder circuit as shown in figure.

2. Verify the working of the circuit by trying different numbers.

RESULT

Designed and set up a 4 bit binary adder/subtractor circuit and BCD Adder using IC-7483 and verified their working.

EXPERIMENT NO: 6

STUDY OF FLIP - FLOPS

AIM

To implement and study about the following flip flops:

a. SR flip flop
b. D flip flop
c. T flip flop
d. JK flip flop
e. MS JK flip flop

COMPONENTS REQUIRED

1) IC-7410
2) IC-7400
3) IC trainer kit

THEORY

Flip flop is the basic building block in any memory system since its output will remain in its state until it is forced to change it by some means.

Clocked SRFlip flop:

'S' and 'R' stand for set and reset. There are four input combinations possible at the inputs. But S = R = 1 is forbidden since the output will be indeterminate. When the FF is switched ON, its output state will be uncertain. When an initial state is to be assigned, two separate inputs called preset and clear are used. They are active low inputs.

D Flip flop:

It has only one input referred to as D input or data input. The input data is transferred to the output after a clock pulse applied. DFF can be derived from JKFF by usig JK input as the D input and J is inverted and fed to K input.

T Flip flop:

T stands for toggle. The output toggles when a clock pulse is applied. That is, the output of the flip flop changes state for an input pulse. TFF can be derived from JK flip flop by shorting J and K inputs.

JK Flip flop:

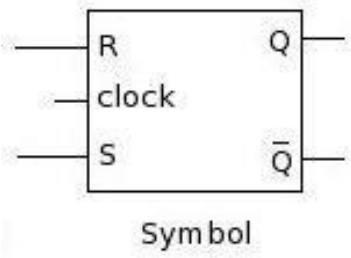
The indeterminate output state of SR FF when S = R = 1 is avoided by converting it to a JK FF.

Master Slave JK Flip flop:

The race around condition of JK FF is rectified in master slave JK FF. racing is the toggling of the output more than once during a positive clock edge. KSJKFF is created by cascading two JKFFs. The clock fed to the first stage (master) is inverted and fed to the second stage (slave). This ensures that the slave follows the master and eliminates the chance of racing.

CIRCUIT DIAGRAM

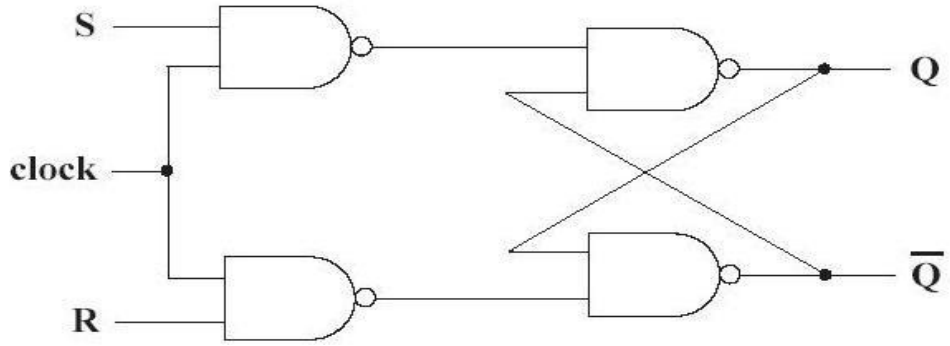
Clocked SR flip flop



Logic Symbol

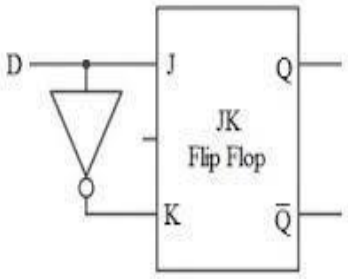
Input		Output
S	R	Q
0	0	Q_n
0	1	0
1	0	1
1	1	X

Truth Table

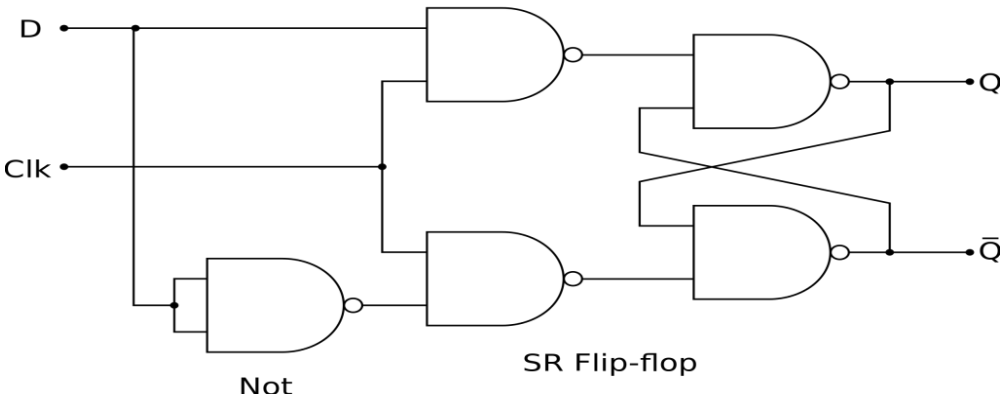


Input	Output
D	Q
0	0
1	1

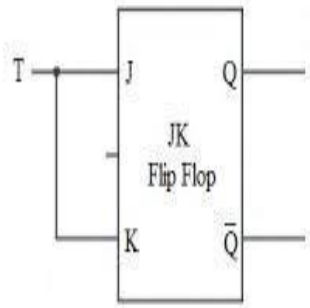
D Flip flop



Logic Symbol **Truth Table**



T Flip flop

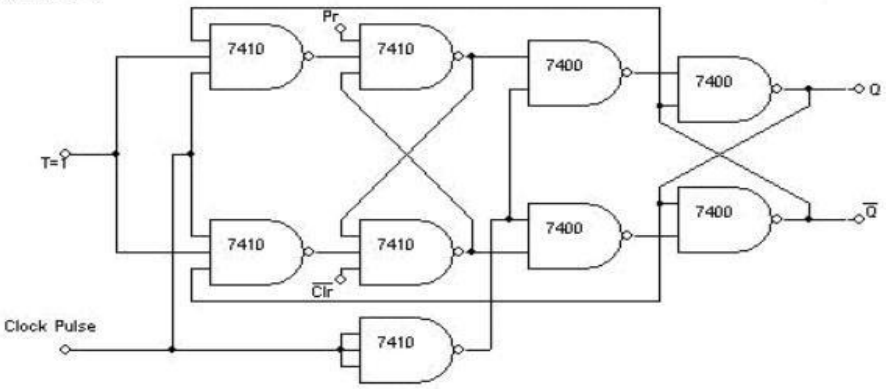


Logic Symbol

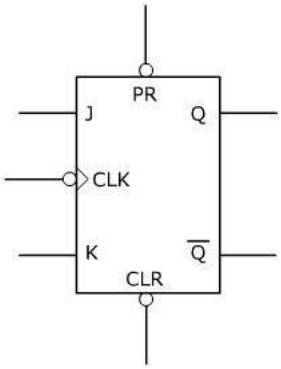
$\bar{T}$	Q_{n+1}
0	Q_n
1	Q_n'

Truth Table

T Flip-Flop:-



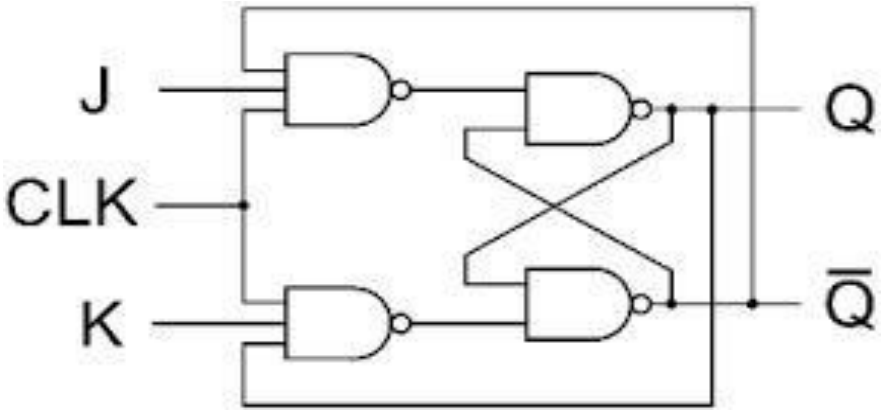
JK Flip flop



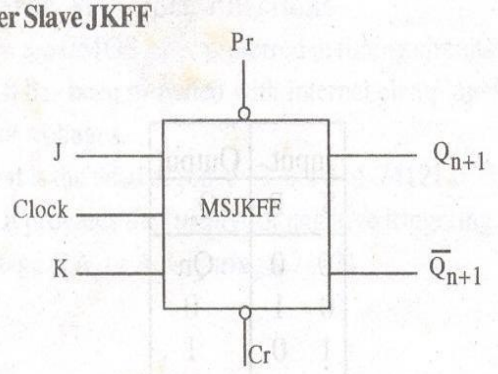
Logic Symbol

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	Qn'

Truth Table



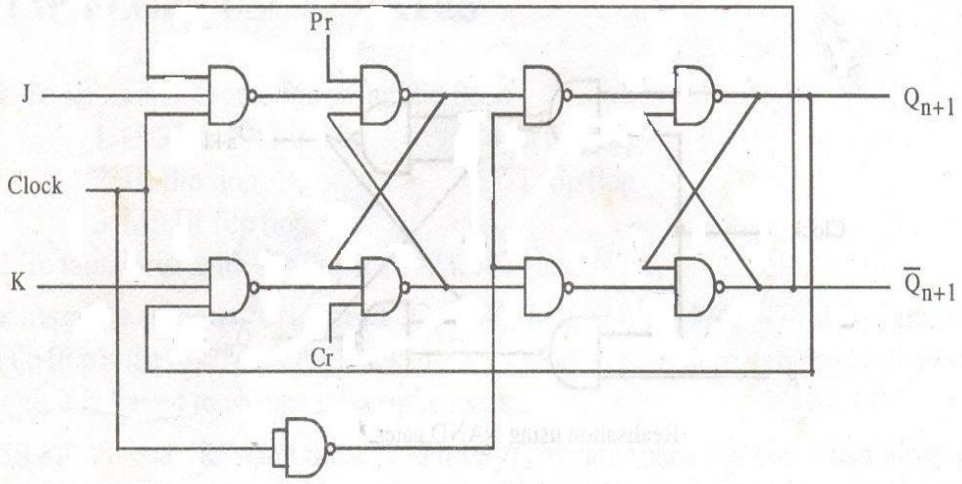
Master Slave JKFF



Logic symbol

Input		Output
J	K	Q_{n+1}
0	0	Qn
0	1	0
1	0	1
1	1	$\overline{Qn}$

Truth table



Realisation using NAND gates

PROCEDURE

1. Test all the components and IC packages.
2. Set up the flip flop as per circuit diagram using gates.
3. Give +5V supply to the pin 14 and ground to the pin 7 to all ICs.
4. Give inputs and CLK to the respective pins of the ICs and observe the output for each clock pulse on LEDs.
5. Repeat the steps 2,3 and 4 for all the flipflops.
6. Prepare truth table, observe and verify it.

RESULT

Implemented and studied the following flip flops using NAND gate

a) Clocked SR flip flop
b) D flip flop
c) T flip flop.
d) JK flip flop
e) MS JK flip flop

EXPERIMENT NO: 7

ASYNCHRONOUS COUNTERS

AIM

To set up the following asynchronous counters using JKFFs and study their working.

1. 4 bit binary up counter (mod 16)
2. 4 bit binary down counter
3. Decade up counter

COMPONENTS REQUIRED

1) IC- 7476

2) IC-7400

3) IC trainer kit

THEORY

A counter is a circuit that produces a set of unique output combinations in relation to the number of applied input pulses. The number of unique outputs of a counter is known as its modulus or mod number.

In asynchronous counters, the flip flops are not given the clock simultaneously. Therefore the propagation delay increases with the number of flip flops used. Four JK flip flops must be used in toggle mode to count 16 states (JKFF can be converted to TFF by shorting J & K inputs). 7476 is a dual JK Master Slave flip flop with preset and clear.

4 bit binary up counter (ripple counter):

In this circuit, all flip flops are clocked by the Q output of the preceding flip flop. JK inputs of all the flip flops are connected to a high state. A ripple counter comprising of n flip flops can be used to count upto 2^n pulses. A circuit with four flip flops gives a maximum count of $2^4 = 16$. The counter gives a natural binary count from 0 to 15 and resets to initial condition on 16^{th} pulse.

With the application of the first clock pulse Q_0 changes from 0 to 1. Q_1, Q_2 and Q_3 remain unaffected. With the second pulse, Q_0 becomes 0 and Q1 becomes 1. At the 16th clock pulse all Q outputs reset and the cycle repeats.

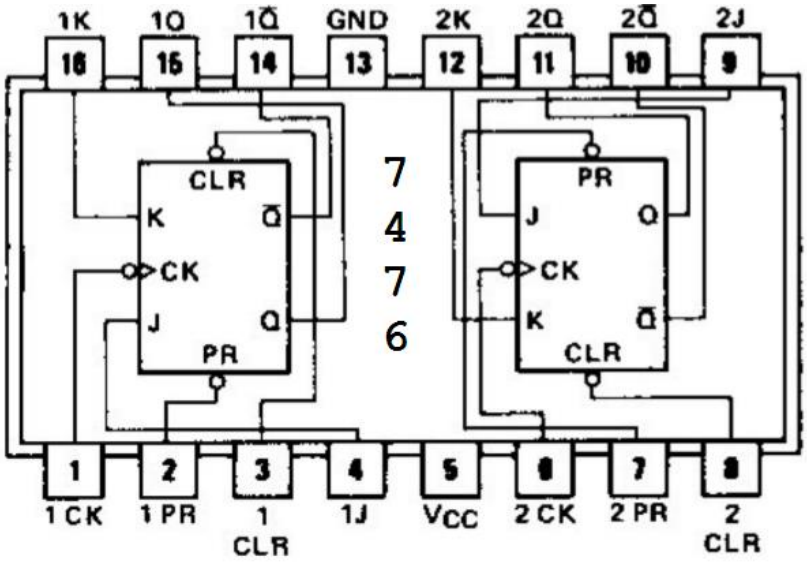
4 bit binary down counter :

In this circuit,the succeeding flipflops are clocked by th $\overline{Q}$ output of preceeding flipflops.The outputs are taken from Q outputs . Initially all Q outputs are set.At the arrival of 16th clock pulse all Q outputs become reset and cycle continues.

Decade up counter:

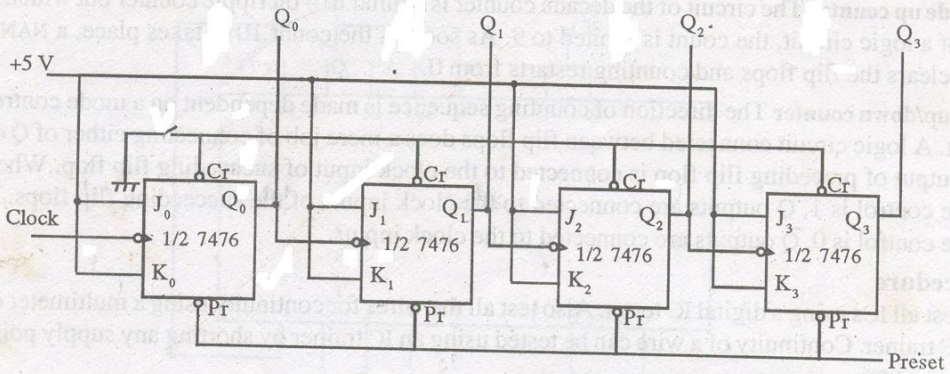
The circuit of the decade counter is similar to 4 bit ripple counter. But with the aid of a logic circuit, the count is limited to 9. As soon as the count 1010 takes place, a NAND gate clears the flip flops and counting restarts from 0.

CIRCUIT DIAGRAM

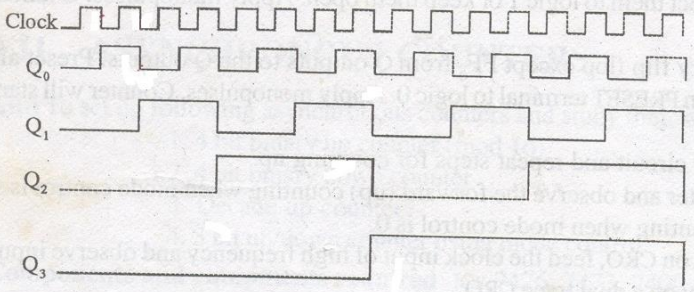


Pinout of IC 7476

4-Bit binary Up Counter



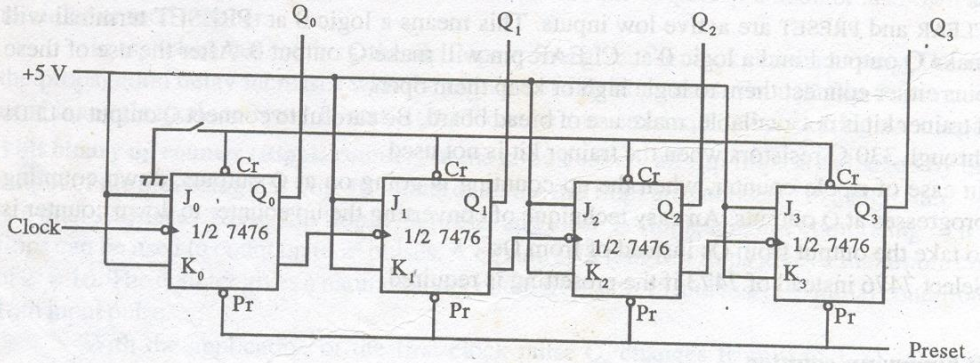
Waveforms for 4 bit binary up counter



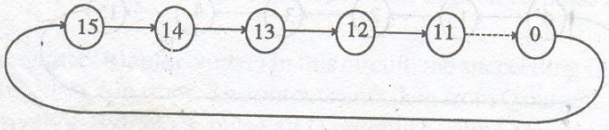
Truth table

Clock	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

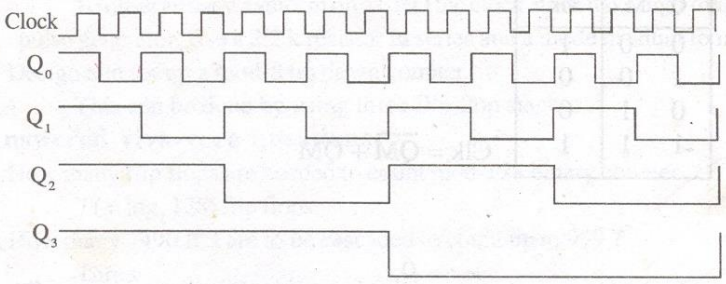
4 bit binary down counter



State diagram



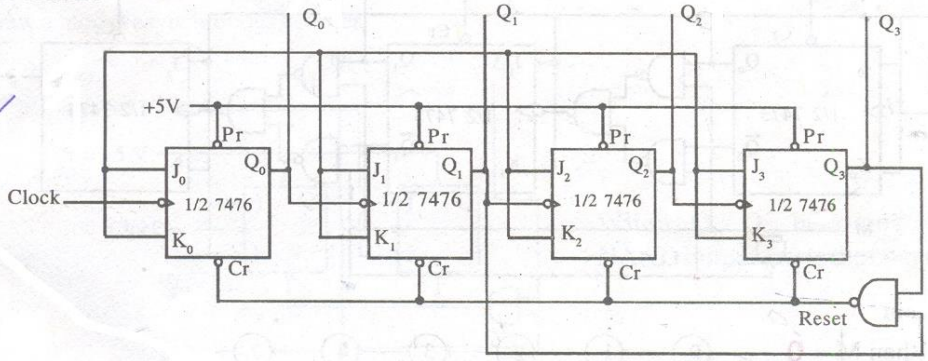
Waveforms for 4 bit binary down counter



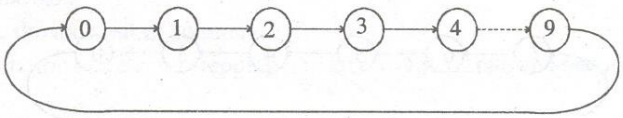
Truth table

Clock	Q_3	Q_2	Q_1	Q_0
0	1	1	1	1
1	1	1	1	0
2	1	1	0	1
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
13	0	0	1	0
14	0	0	0	1
15	0	0	0	0

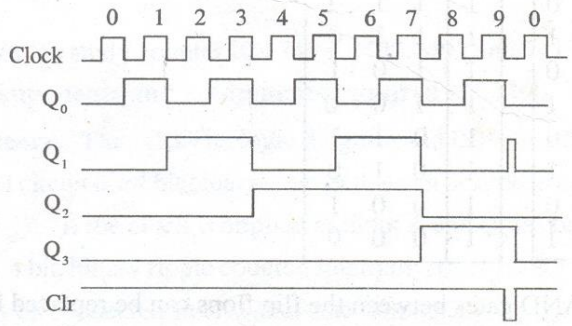
Decade counter



State diagram



Waveforms and truth table for decade counter



Clock	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

PROCEDURE

For 4 bit Binary Up counter

1. Test all the components and IC packages
2. Set up the circuit for a 4 bit ripple up counter.
3. Connect all the PRESET pins to +5V or logic 1 to disable it or keep them open.
4. Clear all the flip flop outputs initially connecting common CLEAR pins to logic 0. After the usage of CLEAR pins, disable it(ie. logic 1).
5. Apply monopulses or give very low frequency from the automatic clock to the CLK input.
6. Observe the output counting up from 0000 to 1111.

For 4 bit Binary Down Counter

1. Set up the circuit for a 4 bit down counter.
2. Repeat the steps 3,4 and 5 of 4 bit binary up counter .
3. Observe the output counting down from 1111 to 0000.

For Decade Counter

1. Set up the circuit for decade counter
2. Reset the outputs using CLEAR input.
3. Apply monopulses or give very low frequency from the automatic clock to the CLK input.
4. Observe the output counting down from 0000 to 1001 .

RESULT

Set up and verified the working of asynchronous 4 bit binary up counter (mod16),4 bit binary down counter and decade counter using JKFFs.

EXPERIMENT NO:8

SYNCHRONOUS COUNTERS

AIM

To design, set up and verify the working of following synchronous counters using JKFFs:

1. 4 bit binary up counter
2. Sequence generator for a sequence 0, 2, 4, 5, 0, 2, 4, 5,……

COMPONENTS REQUIRED

1) IC- 7476

2) IC-7408

3) IC trainer kit

THEORY

Synchronous and asynchronous counters provide the same outputs. The difference is that in the synchronous counters all flip flops work in synchronism with the input clock pulse. That means the outputs of all the flip flops in the counter change state at the same instant. Therefore the propagation delay occurring in asynchronous counter is eliminated in synchronous counters. Synchronous counters for any given count sequence or modulus can be designed and set by the following procedure.

1. Find the number of flip flops using the relation $M = 2^N$ where M is the modulus of the counter and N is minimum number of flip flops required. $N = \log_2 M$.
2. Write down the count sequence (FF outputs) in a tabular form.
3. Determine the flip flop inputs which must be present for the desired next state using excitation table of flip flops.
4. Prepare K-maps for each FF input in terms of FF outputs as the input variables and Obtain the minimized expressions from k-maps.
5. Set up the circuit using FFs and other gates.

4 Bit Up counter

A 4 Bit Up counter up is capable of progressing the counting upwards from 0000 to 1111. An external CLK signal is given to all four flip-flops in parallel

Sequence generators

While the counting progresses, there is a chance of the counter falling to an unused or undesired state. If it happens, the next state will be unknown to counter and it might not progress as desired. To avoid this kind of lock out, a logic circuit is designed to make the counter start from the initial state if the counter falls to an undesired state.

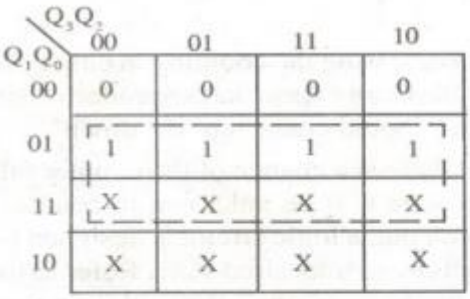
CIRCUIT DIAGRAM

4 bit binary up counter

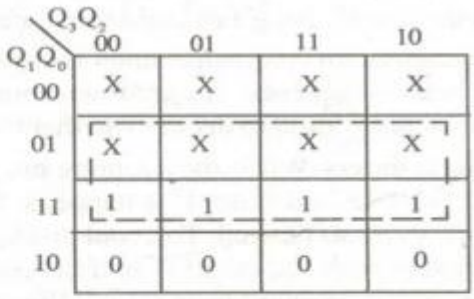
State table

Counter states								JK FF inputs							
Present state				Next state											
Q_3	Q_2	Q_1	Q_0	Q_3	Q_2	Q_1	Q_0	J_0	K_0	J_1	K_1	J_2	K_2	J_3	K_3
0	0	0	0	0	0	0	1	1	X	0	X	0	X	0	X
0	0	0	1	0	0	1	0	X	1	1	X	0	X	0	X
0	0	1	0	0	0	1	1	1	X	X	0	0	X	0	X
0	0	1	1	0	1	0	0	X	1	X	1	1	X	0	X
0	1	0	0	0	1	0	1	1	X	0	X	X	0	0	X
0	1	0	1	0	1	1	0	X	1	1	X	X	0	0	X
0	1	1	0	0	1	1	1	1	X	X	0	X	0	0	X
0	1	1	1	1	0	0	0	X	1	X	1	X	1	1	X
1	0	0	0	1	0	0	1	1	X	0	X	0	X	X	0
1	0	0	1	1	0	1	0	X	1	1	X	0	X	X	0
1	0	1	0	1	0	1	1	1	X	X	0	0	X	X	0
1	0	1	1	1	1	0	0	X	1	X	1	1	X	X	0
1	1	0	0	1	1	0	1	1	X	0	X	X	0	X	0
1	1	0	1	1	1	1	0	X	1	1	X	X	0	X	0
1	1	1	0	1	1	1	1	1	X	X	0	X	0	X	0
1	1	1	1	0	0	0	0	X	1	X	1	X	1	X	1

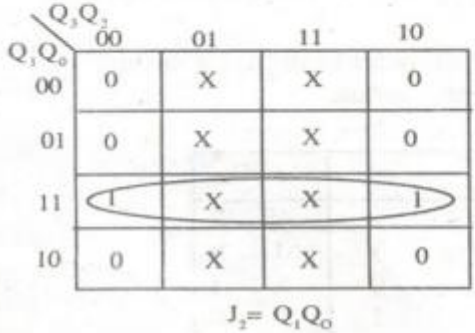
Realisation using Karnaugh map



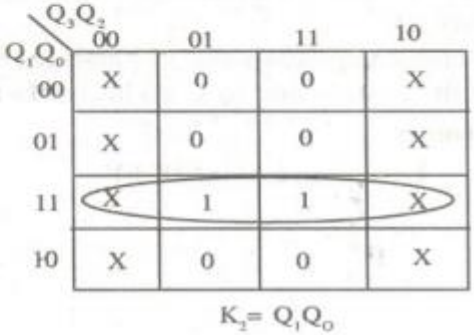
$J_1 = Q_0$



$K_1 = Q_0$



$J_2 = Q_1Q_0$



$K_2 = Q_1Q_0$

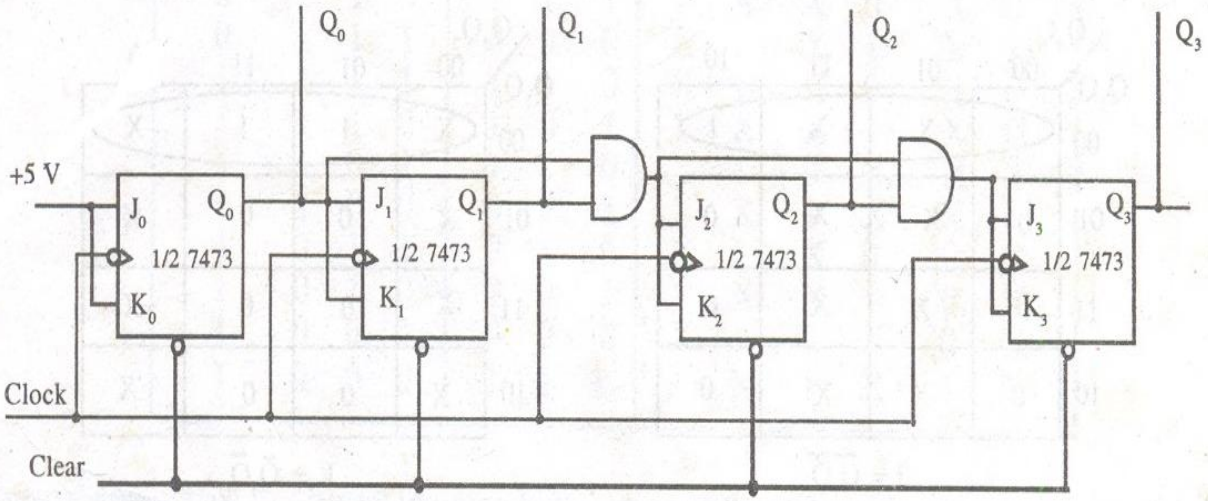
Q_1Q_0 \ Q_3Q_2	00	01	11	10
00	0	0	X	X
01	0	0	X	X
11	0	1	X	X
10	0	0	X	X

$J_3 = Q_2Q_1Q_0$

Q_1Q_0 \ Q_3Q_2	00	01	11	10
00	X	X	0	0
01	X	X	0	0
11	X	X	1	0
10	X	X	0	0

$K_3 = Q_2Q_1Q_0$

Circuit diagram

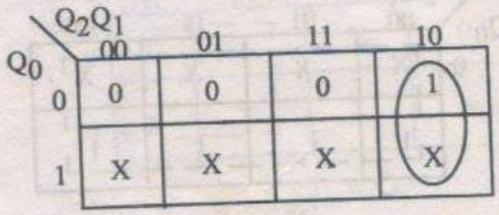


Sequence generator for a sequence 0, 2, 4, 5, 0, 2, 4, 5,……

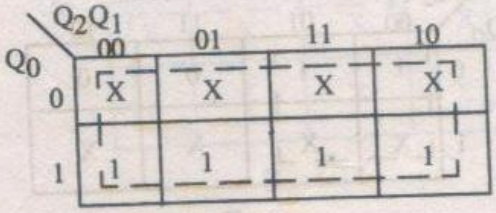
Self starting sequence counter for a sequence 0, 2, 4, 5, 0, 2, 4, 5.....

	Present state			Next state			JK inputs					
	Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	J_0	K_0	J_1	K_1	J_2	K_2
	0	0	0	0	1	0	0	X	1	X	0	X
	0	1	0	1	0	0	0	X	X	1	1	X
	1	0	0	1	0	1	1	X	0	X	X	0
	1	0	1	0	0	0	X	1	0	X	X	1
1→	0	0	1	0	0	0	X	1	0	X	0	X
3→	0	1	1	0	0	0	X	1	X	1	0	X
6→	1	1	0	0	0	0	0	X	X	1	X	1
7→	1	1	1	0	0	0	X	1	X	1	X	1

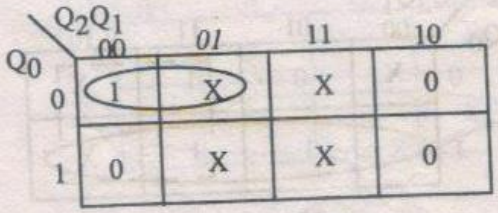
Karnaugh map simplification



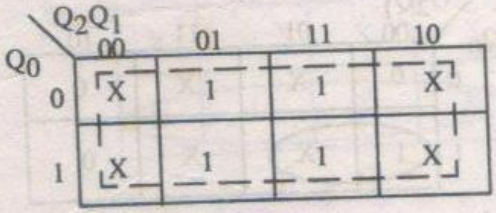
$J_0 = Q_2\overline{Q}_1$



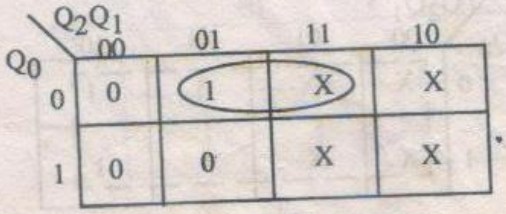
$K_0 = 1$



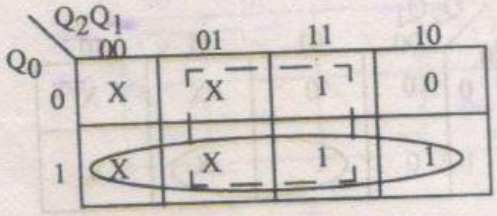
$J_1 = \overline{Q}_0\overline{Q}_2$



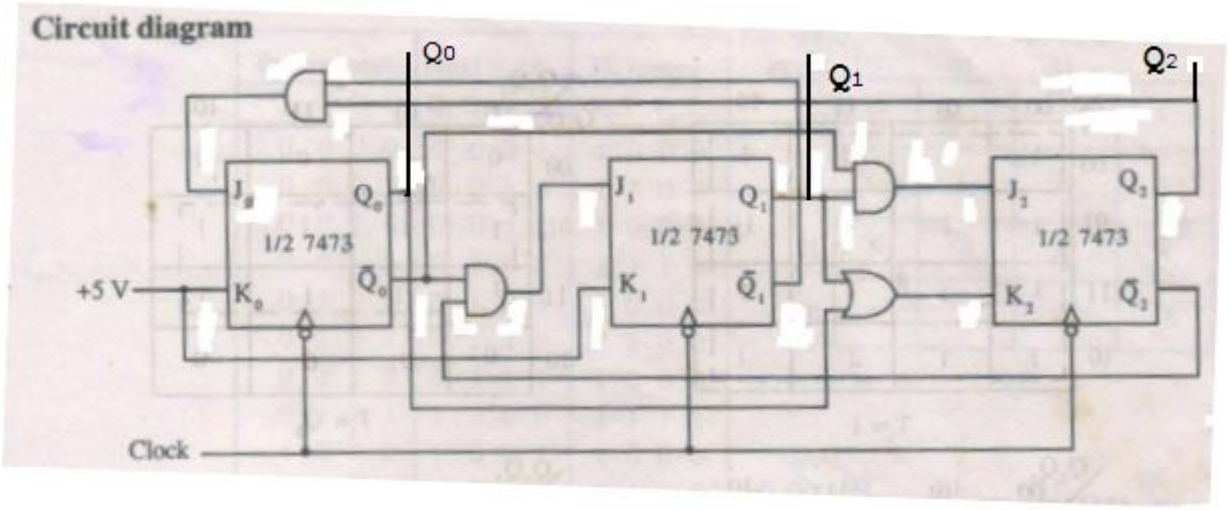
$K_1 = 1$



$J_2 = \overline{Q}_0Q_1$



$K_2 = Q_1 + Q_0$



PROCEDURE

For 4 bit binary Up Counter

1. Test all the components and IC packages.
2. Set up the circuit for synchronous 4 bit binary up counter.
3. Connect J and K inputs of the first flip flop to +5V.
4. Connect the PRESET pins to logic 1 to disable them.
5. Clear all the flip flop outputs initially connecting common CLEAR terminal to logic 0. After the usage of CLEAR pins connect them to logic 1 or keep them open.
6. Apply monopulses or give very low frequency from the automatic clock to the CLK input. Counter starts counting.
7. Observe r the counting is done as per the correct sequence.

For Sequence Generator

1. Set up the circuit for sequence generator as per circuit diagram.

2. Repeat steps 4, 5 and 6.

3. Verify whether the sequence generated is as desired.

RESULT

Designed, set up and verified the working of synchronous 4 bit binary up counter and sequence generator using JKFFs.

EXPERIMENT NO:9

SHIFT REGISTERS

AIM

To design and implement

(i) Serial in serial out Right Shift Register

(ii) Ring Counter

(iii) Johnson Counter

COMPONENTS REQUIRED

IC 7476

IC trainer kit.

THEORY

A register is capable of shifting its binary information in one or both directions is known as shift register. The logical configuration of shift register consist of a Flip flop cascaded with output of one flip flop connected to input of next flip flop. All flip flops receive common clock pulses which causes the shift in the output of the flip flop. The simplest possible shift register is one that uses only one flip flop.

Serial in serial out Right Shift Register

The output of a given flip flop is connected to the input of next flip flop of the register. Each clock pulse shifts the content of register one bit position to right.

Ring counter

It is constructed using JK flip flops by connecting Q and $\bar{Q}$ outputs from one flip flop to the J and K inputs of the next flip flop. The outputs of the final flip flop are connected to the inputs of the first flip flop. To

start the counter, first FF is set using preset facility and the remaining flip flops are reset using reset input. When click signal arrives, this set condition continues to shift around the ring.

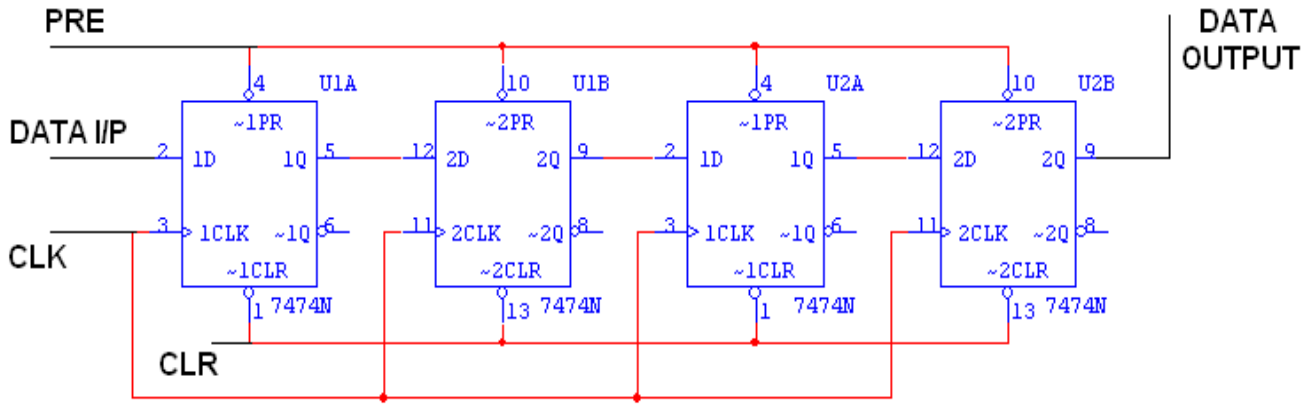
As it can be seen from the truth table, there are four unique output states for this counter, rendering a mode 4 ring counter. Ring counter is called a divide by N counter where N is the number of flip flops.

Johnson counter

A ring counter can be converted to a Johnson counter by connecting the $\bar{Q}$ and Q outputs of the last flip flop ar to the J and K inputs of the first flip flop respectively. The mod number of the counter is double of that of the ring counter.Johnson counter is also called twisted ring counter or divide by 2N counter.

CIRCUIT DIAGRAM

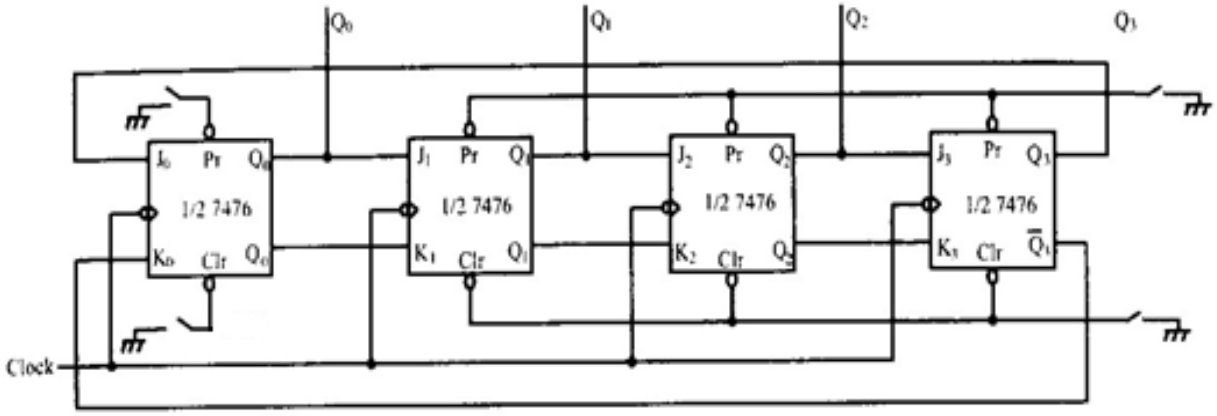
SERIAL IN SERIAL OUT RIGHT SHIFT REGISTER



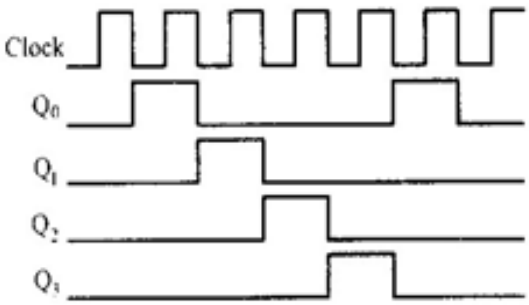
CLK	Serial in	Serial out
1	1	0
2	0	0
3	0	0
4	1	1
5	X	0
6	X	0
7	X	1

TRUTH TABLE

Ring Counter

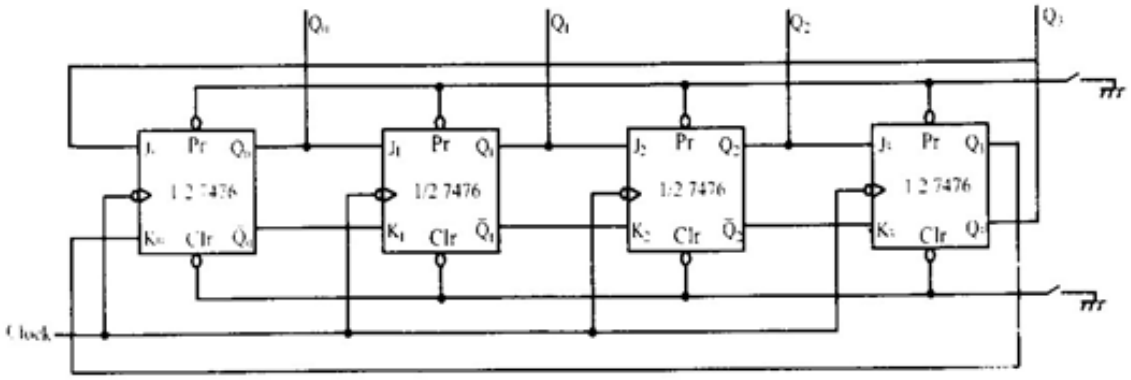


Waveforms for ring counter

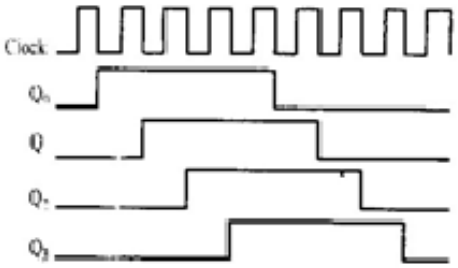


Clk	Q_0 Q_1 Q_2 Q_3
1	1 0 0 0
2	0 1 0 0
3	0 0 1 0
4	0 0 0 1

Johnson counter



Waveforms for Johnson counter



Clk	Q_0 Q_1 Q_2 Q_3
0	0 0 0 0
1	1 0 0 0
2	1 1 0 0
3	1 1 1 0
4	1 1 1 1
5	0 1 1 1
6	0 0 1 1
7	0 0 0 1

PROCEDURE

For Shift Register

1. Test all the components and IC packages.
2. Setup the shift register as per circuit diagram.
3. Clear all the flipflops using CLEAR pins.Feed a four bit binary number one bit at a time .Apply clock pulse after feeding each bit to the input.
4. Observe the shifting of bits. By applying CLOCK.

For Ring Counter

1. Setup the ring counter as per circuit diagram.
2. Ensure that CLEAR and PRESET pins are in inactive mode. (ie. Logic 1)
3. Clear all flipflops by applying logic 0 to all CLEAR pins.After clearing the outputs,inactivate all CLEAR pins by applying logic 1.
4. Set the output of first flipflop to logic 1, by applying logic 0 to its PRESET pin.After presetting the output,inactivate that PRESET pin by switching it to logic 1.
5. Apply monopulses using the debouncer switch or give very low frequency pulse from the automatic clock to the CLOCK input and observe the output.

For Johnson Counter

1. Setup the johnson counter by interchanging the J and K connections of the first flip flop as shown in figure to make a Johnson counter.
2. Repeat the steps from 2 to 5 to verify the output of Johnson counter.

RESULT

Set up and verified shift register , ring counter and johnson counter.

EXPERIMENT NO:10

MULTIPLEXERS AND DEMULTIPLEXERS USING BASIC GATES

AIM

To design and implement

1. 4 line to 1 line multiplexer using gates.
2. 1 line to 4 line demultiplexer using gates

COMPONENTS REQUIRED

1. IC trainer kit
2. IC-7404
3. IC-7411
4. IC-7432

THEORY

Multiplexer:

Multiplexer is a combinational circuit which can select any one of the inputs and route it to the output. A multiplexer has data input lines, data select lines and output.

The logic symbol of a 4 line to 1 line multiplexer is shown in figure. According to the two bit binary code on the data select inputs, corresponding data input line will be selected and routed to the output. For example, if S_1S_0 is 00, D_0 will be selected, if S_1S_0 is 01, D_1 will be selected and so on

From the truth table it can be seen that the output

$$Y = D_0\overline{S_1}\overline{S_0} + D_1\overline{S}_1S_0 + D_2S_1\overline{S}_0 + D_3S_1S_0$$

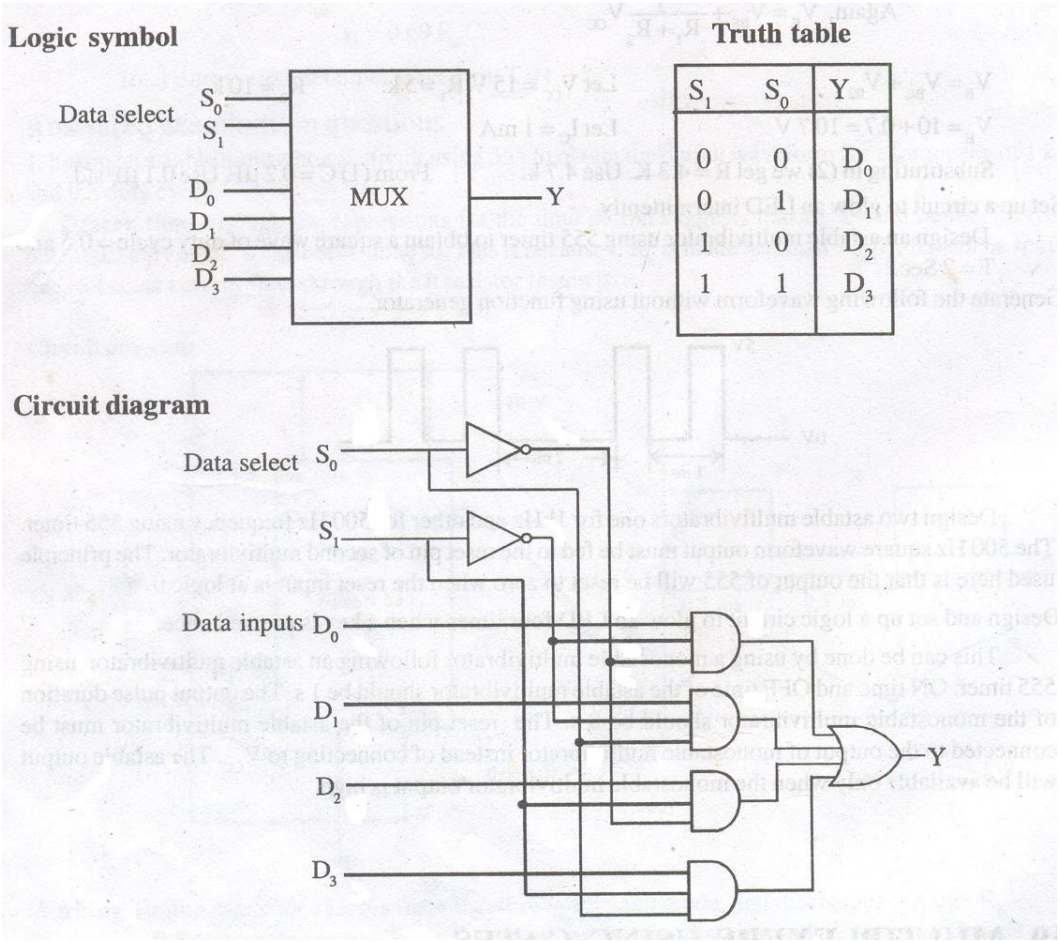
This Boolean expression can be realized using gates.

Demultiplexer:

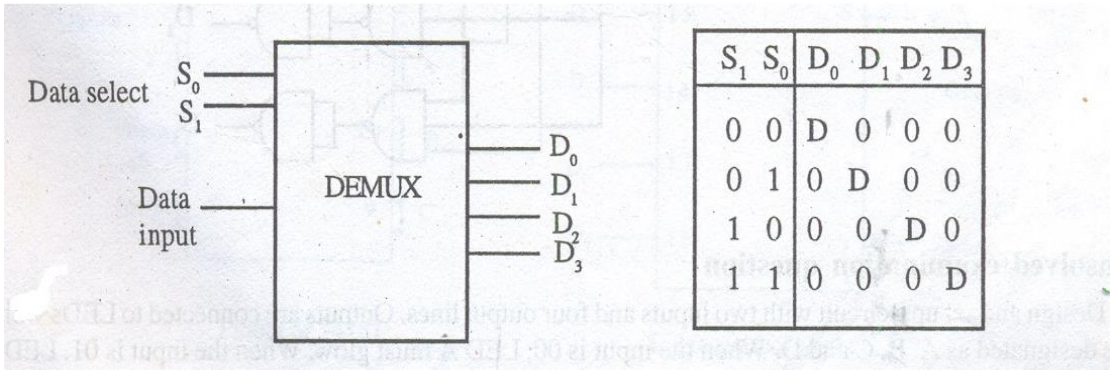
Demultiplexer does the reverse operation of multiplexer. The data on the line is distributed to that output line whose binary code is given on the data select lines, ie S_1S_0. Each output line has its own unique binary code. The logic symbol of a 1 line to 4 line demultiplexer is shown in figure. When the input on data select inputs S_1S_0 is 00, data on the data line will be available on D_0 output, if the input on data select inputs is S_1S_0 is 01, data on the data line will be available on D_1 output and so on

CIRCUIT DIAGRAM

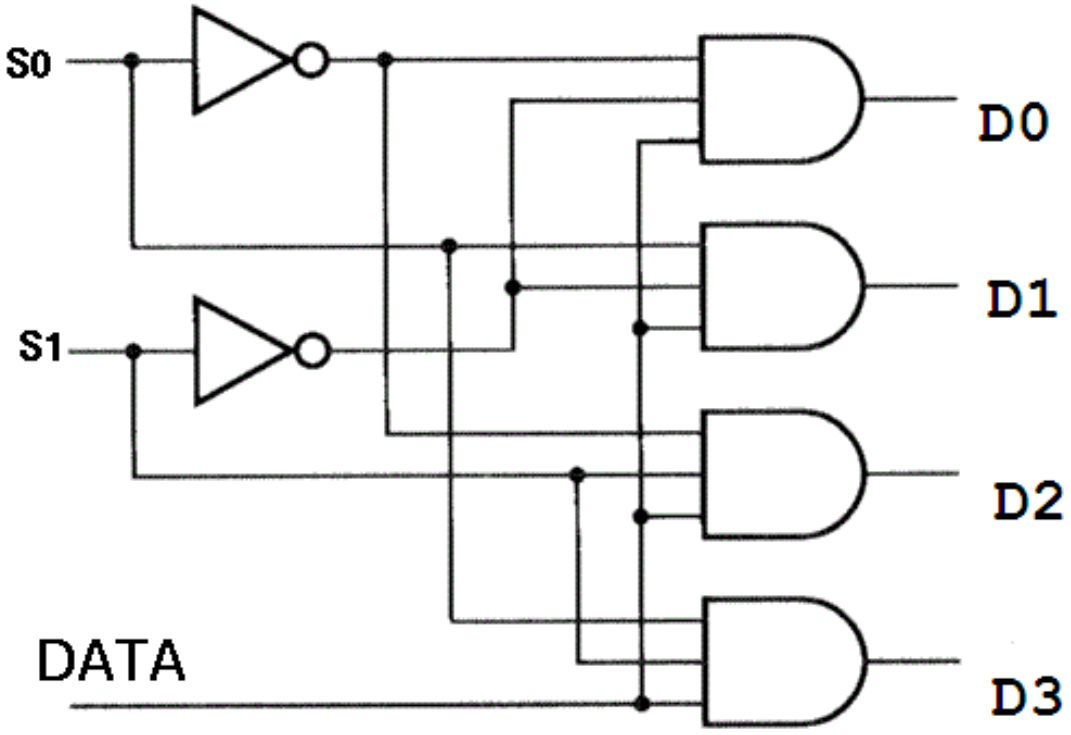
4:1 Multiplexer



1:4 Demultiplexer



S_1	S_0	D_0	D_1	D_2	D_3
0	0	D	0	0	0
0	1	0	D	0	0
1	0	0	0	D	0
1	1	0	0	0	D



PROCEDURE

For 4:1 Multiplexer

1. Test all the components and IC packages.
2. Set up the circuit for multiplexer as per the diagram.
3. Give any random binary combinations at D_0 through D_3.
4. Feed all four combinations at S_1S_0 and observe the corresponding output.
5. Verify that the circuit functions as a multiplexer.

For 1:4 Demultiplexer

1. Set up the circuit for demultiplexer as per the diagram.
2. Give logic 1 on data(D) input line.
3. Feed all four combinations one by one at S_1S_0 and observe the data at the corresponding output line.
4. Verify that the circuit functions as a demultiplexer.

RESULT

Studied and verified the working of a 4 line to 1 line multiplexer and a 1 line to 4 line demultiplexer using gates.

EXPERIMENT NO:11

COMBINATIONAL LOGIC DESIGN USING MULTIPLEXER AND DEMULTIPLEXER ICs

AIM

To study the combinational logic design using multiplexers and demultiplexer ICs.

COMPONENTS REQUIRED

1) IC-74154
2) IC-7400
3) IC-7408
4) IC trainer kit.
5) IC-74151

THEORY

Multiplexers can be used to realize logic circuits. A multiplexer with n number of select lines can be used to realize an n variable Boolean expression very easily. With the additional logic gates or circuits an n + 1 variable expression can be realized with the same multiplexer.

Multiplexer IC 74151

IC 74151 is an 8: 1 multiplexer with 16 pin IC package which provides two complementary outputs Y & $\bar{Y}$. The o/p Y is same as the selected i/p & $\bar{Y}$ is its complement. The n: 1 multiplexer can be used to realize a m variable function. (2m= n, m is no. of select inputs)

An 8 line to 1 line multiplexer can be used to implement a three variable logic function and a 16 line to 1 line multiplexer to implement a four variable Boolean expression without any additional logic gates. Suppose the Boolean expression to be implemented is

$$F = \overline{ABC} + \bar{A}\,B\bar{C} + ABC$$

The minterms in the above expression corresponds to 000, 010 and 111. This expression can be implemented using 74151 by merely applying logic 1 inputs to D_0, D_2 and D_7 and logic 0 to other inputs.

If we have a Boolean function of n + 1 variable, we can take n of these variables and connect them to the selection lines of the MUX. The remaining single variable of the function is used for the input of the MUX. If A is the remaining single variable the input of the MUX are chosen to be A or A or 1 or 0. A general procedure for implementing any Boolean function of n- variables with 2^{n-1} x MUX is as follows:

Step-1: List the inputs of the MUX and under them list all minterms in two rows. The first row lists on those minterms where A is complemented.

Step-2: Circle all the minterms of the function and inspect each coloumn separately. If the two minterms in a coloumn are circled apply 1 to the MUX input, if both are not circled apply 0 to the MUX input, if only bottom one is circled apply A to the corresponding to the MUX input and if the top minterm is circled apply A to the MUX input.

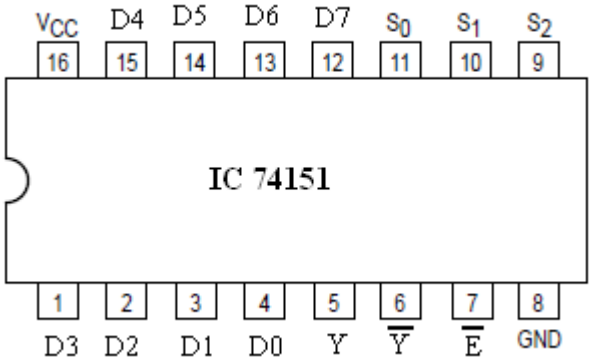
Demultiplexer IC 74154

It is a 1:16 demultiplexer in 24 -pin DIP.It has 4 select lines to select one out of 16 outputs Y0 throughY15and 2 active low strobe inputs G1 and G2.One of the strobe inputs should be used as the data input and other held at logic 0.If either of these strobe input is at logic1,all outputs will be at logic 1 irrespective of the select inputs.It can be used as 4:16 decoder by connecting G1 and G2 to logic 0and using select lines as the input lines.Suppose the Boolean expression to be implemented using the decoder is

$$F = \bar{A}\bar{B}C\bar{D}+\bar{A}B\bar{C}\bar{D}+\bar{A}BCD+A\bar{B}\bar{C}\bar{D}+AB\bar{C}\bar{D}$$

It means that the minterms are corresponding to the numbers 2, 4, 7, 8 and 12. If the binary numbers corresponding to these numbers are applied at the select inputs, corresponding data output will be 0. The outputs Y_2, Y_4, Y_7, Y_8 AND Y_{12} must be inverted and ORed to obtain F. According to De Morgan's theorem inverting and ORing is equivalent to NANDing.

CIRCUIT DIAGRAM



Pinout of IC74151

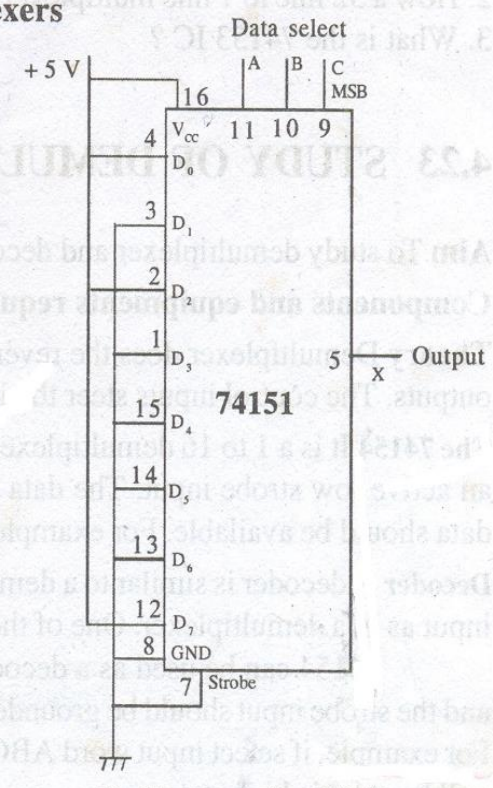
Realisation of Boolean expressions using multiplexers

The truth table to be implemented using a MUX

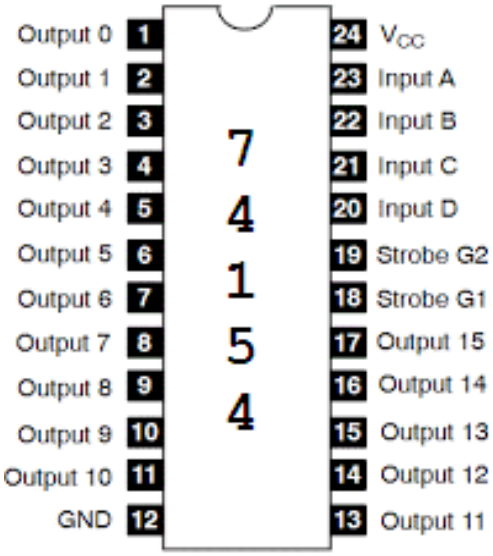
$F = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + ABC$

Inputs			Output
C	B	A	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Note: Pin diagram of 74151 is given in previous experiment



Realization using 74151



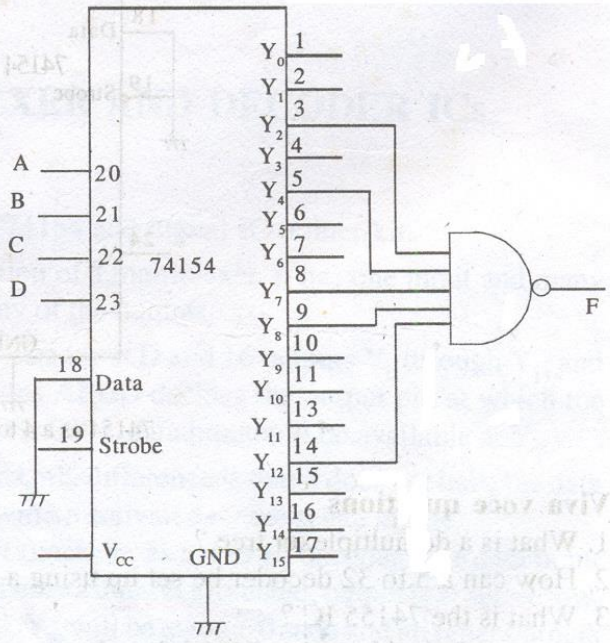
Pinout of IC74154

Implementation of boolean expression using a decoder

The truth table to be implemented using a MUX

$F = \bar{A}\bar{B}C\bar{D} + \bar{A}B\bar{C}\bar{D} + \bar{A}BCD + A\bar{B}\bar{C}\bar{D} + AB\bar{C}\bar{D}$

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



Realisation using decoder

PROCEDURE

1. Set up the logic circuits one by one as shown in figure.

2. Verify their truth tables by feeding necessary inputs at the data select inputs.

RESULT

Designed ,implemented and studied the combinational logic design using MUX and DEMUX IC.

EXPERIMENT NO: 12

REALIZATION OF VERILOG MODULES FOR BASIC GATES & FAMILIARISATION OF VERILOG

AIM

Familiarisation of verilog and Study and Develop verilog modules for basic gates, synthesis and implementation.

THEORY

About Xilinx

Xilinx is an American technology company that is primarily a supplier of programmable logic devices. The company invented the field-programmable gate array (FPGA). It is the semiconductor company that created the first fabless manufacturing model. Xilinx sells a broad range of FPGAs, complex programmable logic devices (CPLDs), design tools, intellectual property and reference designs. Xilinx customers represent just over half of the entire programmable logic market, at 51%. Altera (now Intel) is Xilinx's strongest competitor with 34% of the market. Other key players in this market are Actel (now Microsemi), and Lattice Semiconductor. We are using the Xilinx ISE Design Suite for the simulation of the Verilog modules.

Logic gates:

In digital electronics, a gate is a logic circuit with one output and one or more

inputs. Logic gates are available as ICs.

AND gate:

The AND gate performs logical multiplication, more commonly known as AND operation.

The AND gate output will be in high state only when all the inputs are in high state. 7408 is

a digital IC in the TTL family and contains four AND gates. For this reason, it is called quad

two input AND gate. Every AND gate has two inputs in this Dual-in-Line Package (DIP). 14

is the supply pin. For standard TTL devices to work properly, the supply voltage level must

be +4.75V and +5.25V. This is why +5V is the nominal supply voltage specified for all TTL devices. Pin 7 is the common ground for the chip. The other pins are inputs and outputs.

OR gate:

It performs logical addition. Its output will become high if any of the inputs is in logic high. 7432 is a quad two input OR gate.

NOT gate:

It performs a basic logic function called inversion or complementation. The purpose of the inverter is to change one logic level to opposite level. IC 7404 is a hex inverter.

NAND gate:

A NOT gate following an AND gate is called NOT-AND or NAND gate. Its output will be low if all the inputs are in high state. 7400 IC is a quad two input NAND gate.

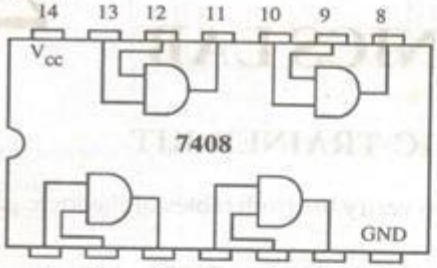
NOR gate:

A NOT gate following and OR gate is called NOT-OR or NOR gate. Its output will be in low state if any of its inputs is in high state. 7402 is a quad two input NOR gate.

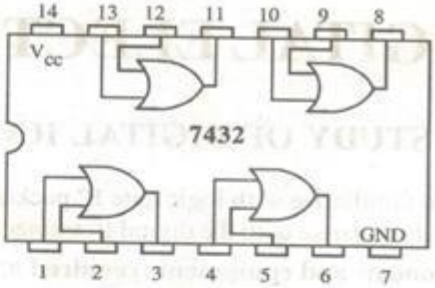
XOR gate:

Its output will be high if and only if one input is in high state. 7486 is a quad two input XOR gate.

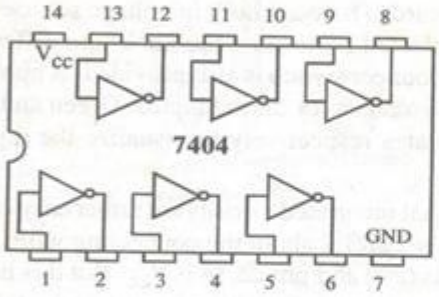
DIP pin out



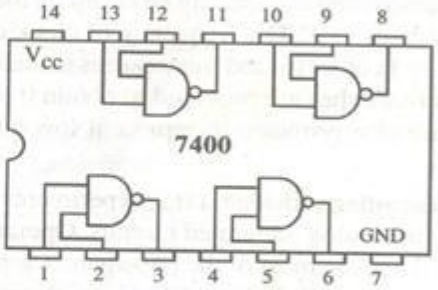
Quad two input AND gate



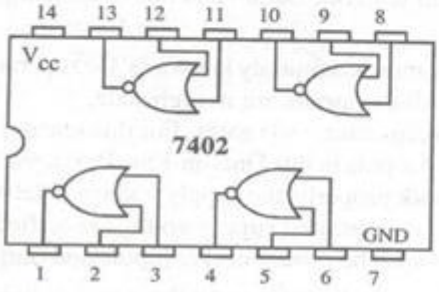
Quad two input OR gate



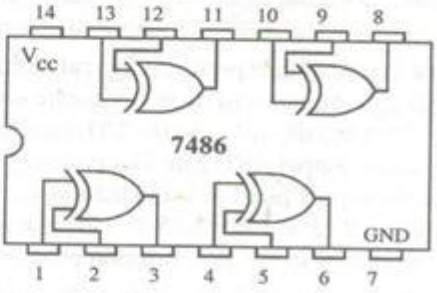
HEX inverter gates



Quad two input NAND gates



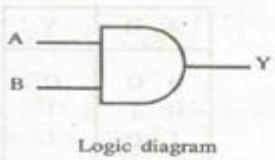
Quad two input NOR gates



Quad two input XOR gates

Symbols and truth tables

AND gate



Logic diagram

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Truth table

OR gate

Logic diagram

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Truth table

NOT gate

Logic diagram

A	Y
0	1
1	0

Truth table

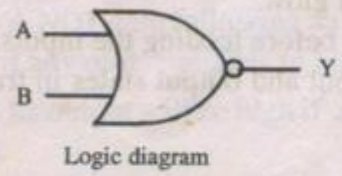
NAND gate

Logic diagram

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Truth table

NOR gate



Logic diagram

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

XOR gate

Logic diagram

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Truth table

VERILOG PROGRAMS

AND GATE

```
module and_logic(a,b,c);
input a;
input b;
output c;
assign c=a&b;
endmodule
```

OR GATE

```
module or_logic(a,b,c);
input a;
input b;
output c;
```

```
assign c=a|b;
endmodule
```

NOT GATE

```
module not_logic(a,b);
input a;
output b;

assign b=~a;
endmodule
```

NAND GATE

```
module nand_logic(a,b,c);
input a;
input b;
output c;
assign c=~(a&b);
endmodule
```

NOR GATE

```
module nor_logic(a,b,c);
input a;
input b;
output c;
assign c=~(a|b);
```

```
endmodule
```

XOR GATE

```
module xor_logic(a,b,c);
input a;
input b;
output c;
assign c=a^b;
endmodule
```

XNOR GATE

```
module xnor_logic(a,b,c);
input a;
input b;
output c;
assign c=~(a^b);
endmodule
```

RESULT

Developed and implemented verilog modules for logic gates. Also studied and verified the truth tables of the logic gates.

EXPERIMENT NO: 13

VERILOG MODULES FOR HALF ADDER AND FULL ADDER

AIM

Study and Develop verilog modules for half adder and full adder in 3 modelling styles .

THEORY

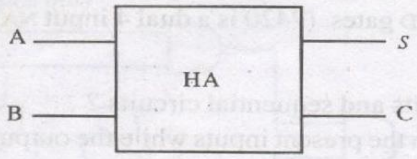
Adders:

The simplest binary adder is called a half adder. Half adder has two input bits and two output bits. One output bit is the sum and the other is carry. They are represented by S and C respectively in the logic symbol.

A half adder has no provision to add a carry from the lower order bits when binary numbers are added. When two input bits and a carry are to be added, the number of input bits becomes three and the input combinations increases to eight. For this, a full adder is used. Like half adder, it also has a sum bit and a carry bit. The new carry generated is represented by Cn and carry generated from the previous addition is represented by Cn-1.

CIRCUIT DIAGRAM

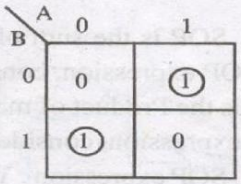
Half Adder



Input		Output	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

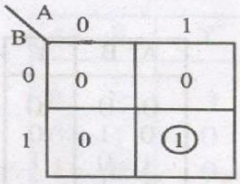
Truth table

Realization of sum and carry using Karnaugh map



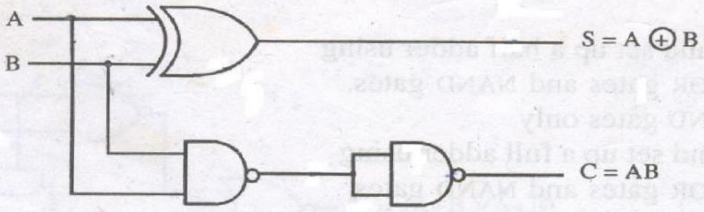
$S = A\overline{B} + \overline{A}B$

$= A \oplus B$

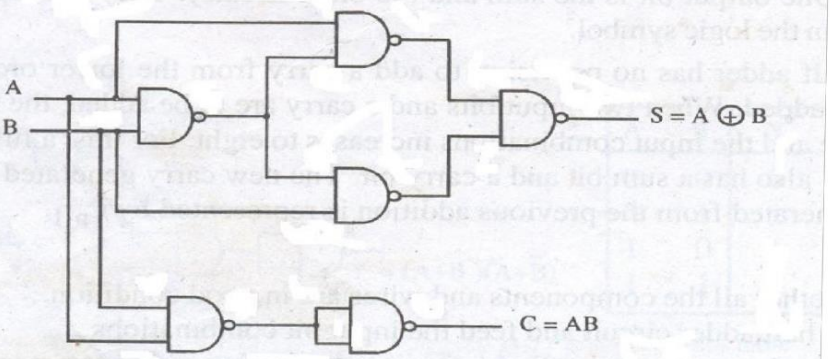


carry = A.B

Implementation using logic gates

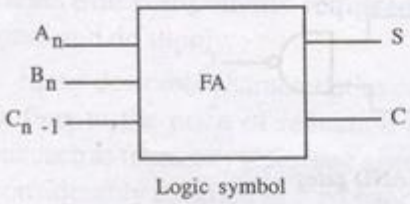


Half adder using EXOR and NAND gates



Half adder using NAND gates only

Full adder

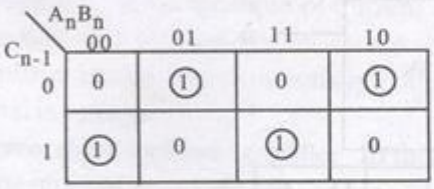


Logic symbol

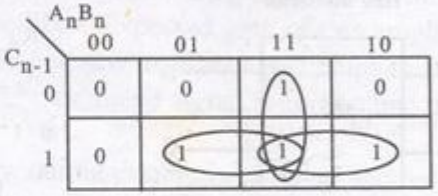
Input			Output	
A_n	B_n	C_{n-1}	S_n	C_n
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Truth table

Realisation of sum and carry using Karnaugh map

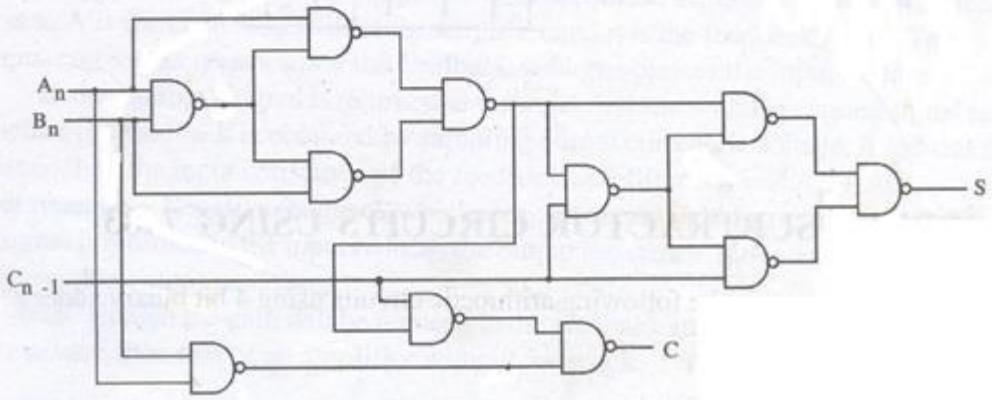


$$S_n = \overline{A}_n\overline{B}_nC_{n-1} + \overline{A}_nB_n\overline{C}_{n-1} + A_nB_nC_{n-1} + A_n\overline{B}_n\overline{C}_{n-1}$$
$$= A_n \oplus B_n \oplus C_{n-1}$$

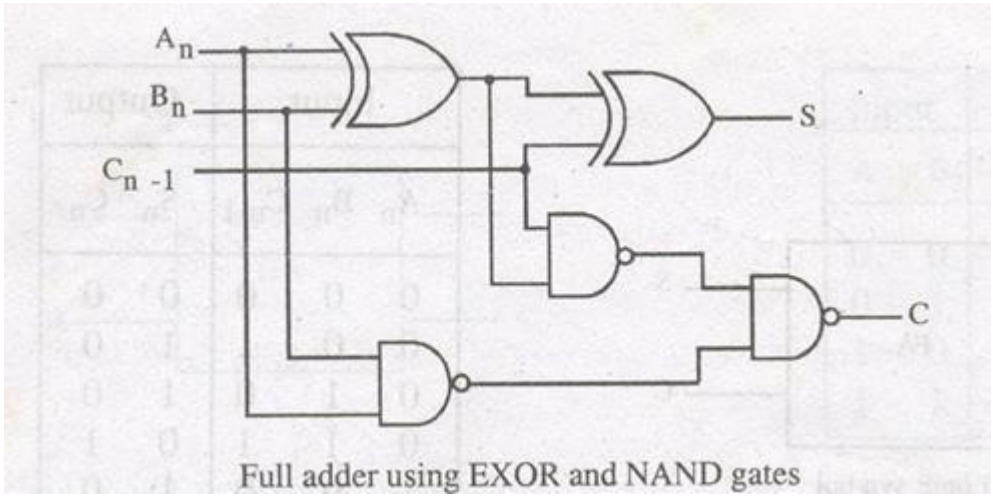


$$C_n = A_nB_n + B_nC_{n-1} + A_nC_{n-1} \text{ or}$$
$$= A_nB_n + A_n\overline{B}_nC_{n-1} + \overline{A}_nB_nC_{n-1} \text{ when horizontal grouping is avoided.}$$
$$= A_nB_n + C_{n-1}.(A_n \oplus B_n)$$

Realisation using logic gates



Full adder using NAND gates only



Full adder using EXOR and NAND gates

VERILOG PROGRAMS

HALF ADDER

Dataflow Modeling

```
module half_adder(a,b,s,c);
input a;
input b;
output s;
output c;
assign s=a^b;
assign c=a&b;
endmodule
```

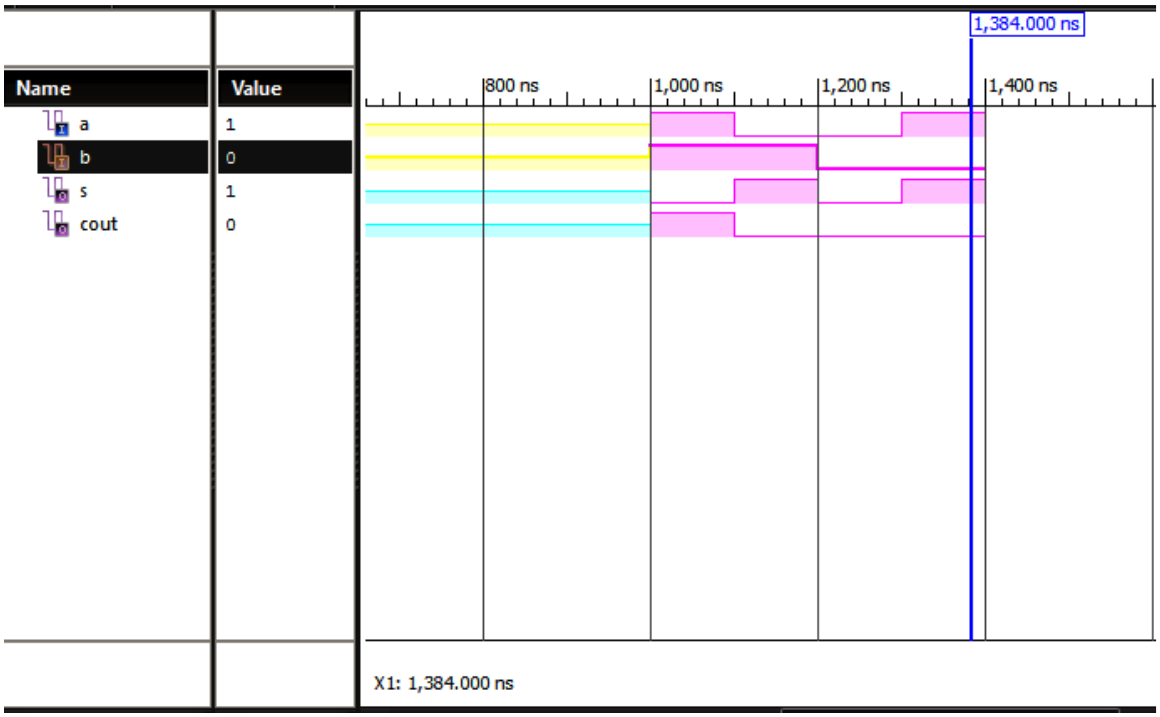
Behavioral Modeling

```
module halfadder(
   a,b,
    s,cout
   );
input a,b;
```

```
output reg s,cout;
always @(a,b)
begin
s=a^b;
cout=(a&b);
End
Endmodule
```

Gate level /Structural modeling

```
module ha_gatelevel(
input a,
input b,
output sum,
output carry
);
xor (sum, a,b);
and (carry,a,b);
endmodule
```



FULL ADDER

Dataflow Modeling

```
module fa_dataflow(
 input a,
 input b,
 input c,
 output sum,
 output carry
 );
assign sum=a^b^c;
assign carry= (a&b)|(b&c)|(a&c);
endmodule
```

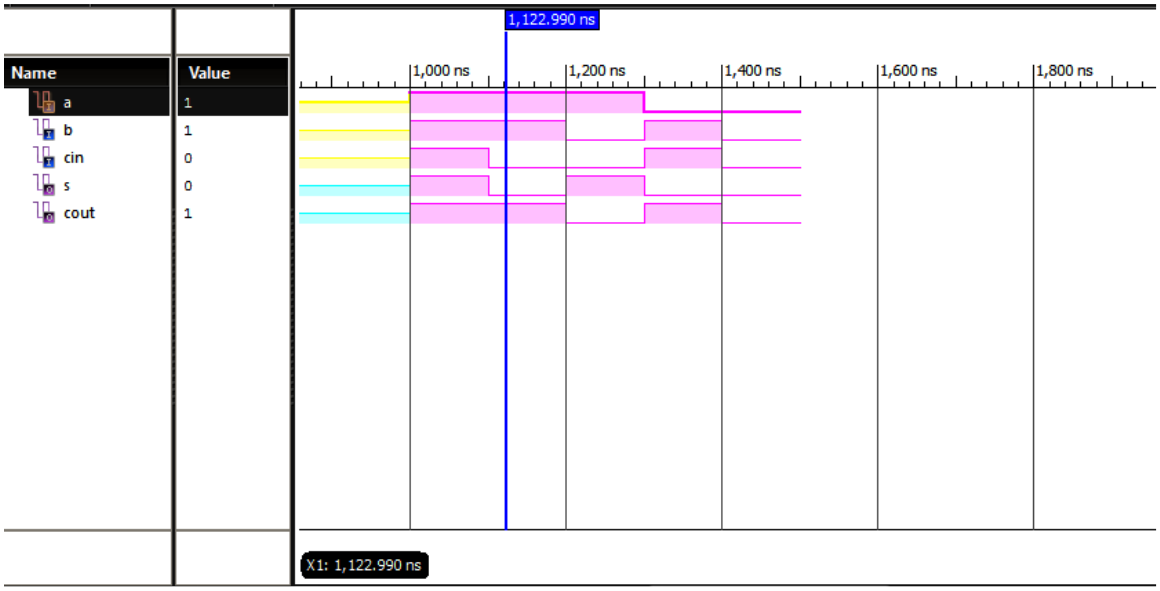
Behavioral Modeling

```
module fulladder(
    a,b,cin,
     s,cout
```

```
   );
input a,b,cin;
output reg s,cout;
always @(a,b,cin)
begin
s=a^b^cin;
cout=(a&b)| (b&cin)|(a&cin);
end
Endmodule
```

Gate level /Structural modeling

```
module fa_gate_level(
 input a,
 input b,
 input c,
 output s,
 output co
 );
wire n1,n2,n3,n4;
xor(n4,a,b);
xor(s,n4,c);
and(n1,a,b);
and(n2,b,c);
and(n3,a,c);
or(co,n1,n2,n3);
endmodule
```



RESULT

Developed and implemented verilog modules for half adder and full adder. Also studied and verified its truth table.

EXPERIMENT NO: 14

VERILOG MODULE FOR CODE CONVERTERS

AIM

Study and Develop verilog modules for

1.4 -bit Binary to gray

2.4-bit Gray to Binary .

THEORY

Binary code to gray code converter

To convert a binary number to corresponding Gray code, the following rules are applied:

3. The MSB in the Gray code is the same as the corresponding bit in a binary number.
4. Going from left to right, add each adjacent pair of binary digits to get the next Gray code digit. Disregard carries.

As the first step to design a binary to Gray code converter, set up a truth table with binary numbers $B_3B_2B_1B_0$ and corresponding Gray code numbers $G_3G_2G_1G_0$. Set up a circuit realizing the simplified logic expressions obtained using K maps for Gs as the functions of Bs.

Gray code to binary code converter

To convert from Gray code to binary, the following rules are applied:

3. The most significant digit in the binary number is the same as the corresponding digit in the Gray code.
4. Add each binary digit generated to the Gray code digit in the next adjacent position. Disregard carries.

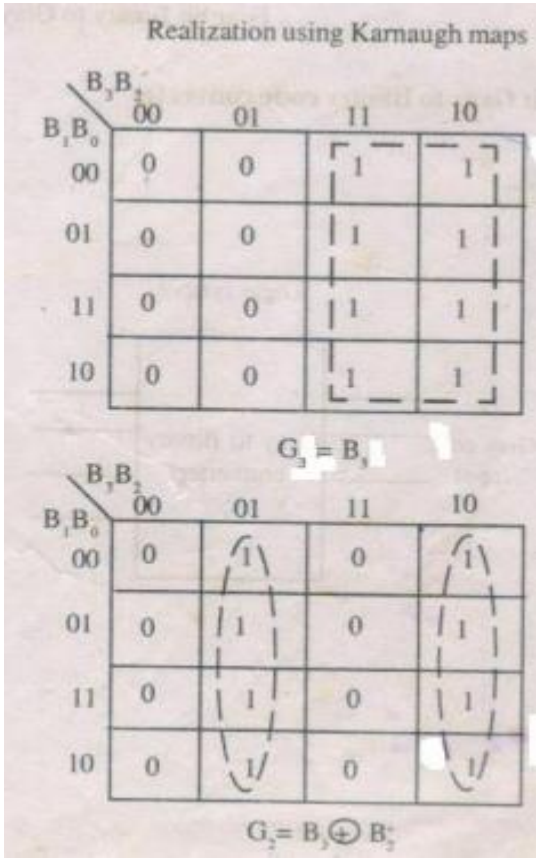
To design the Gray to binary code converter, set up the truth table and get simplified expressions using K maps for each binary bit as a function of Gray code bits. Each Gray code number differs from the preceding number by a single bit.

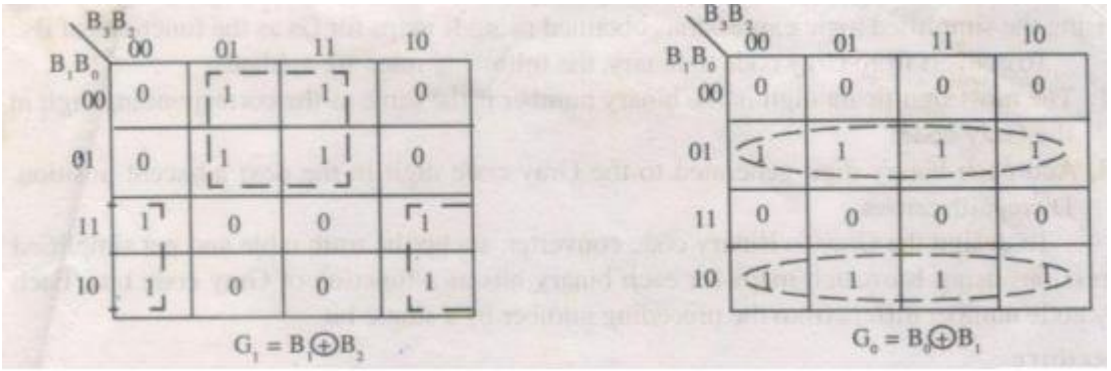
CIRCUIT DIAGRAM

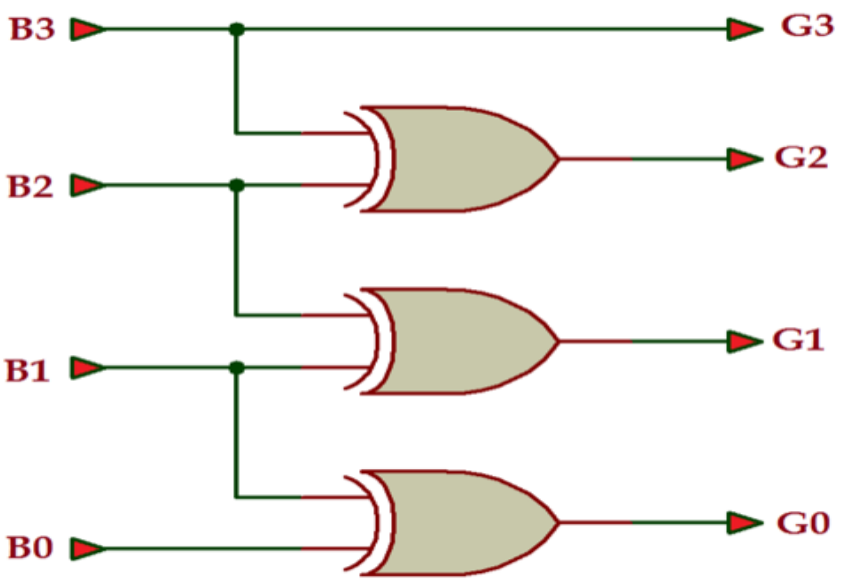
Binary to gray code converter

BINARY INPUT				GRAY CODE OUTPUT			
B3	B2	B1	B0	G3	G2	G1	G0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

Truthtable




B3B2
B1B0
00
01
11
10
G1 = B1⊕B2
G0 = B0⊕B1

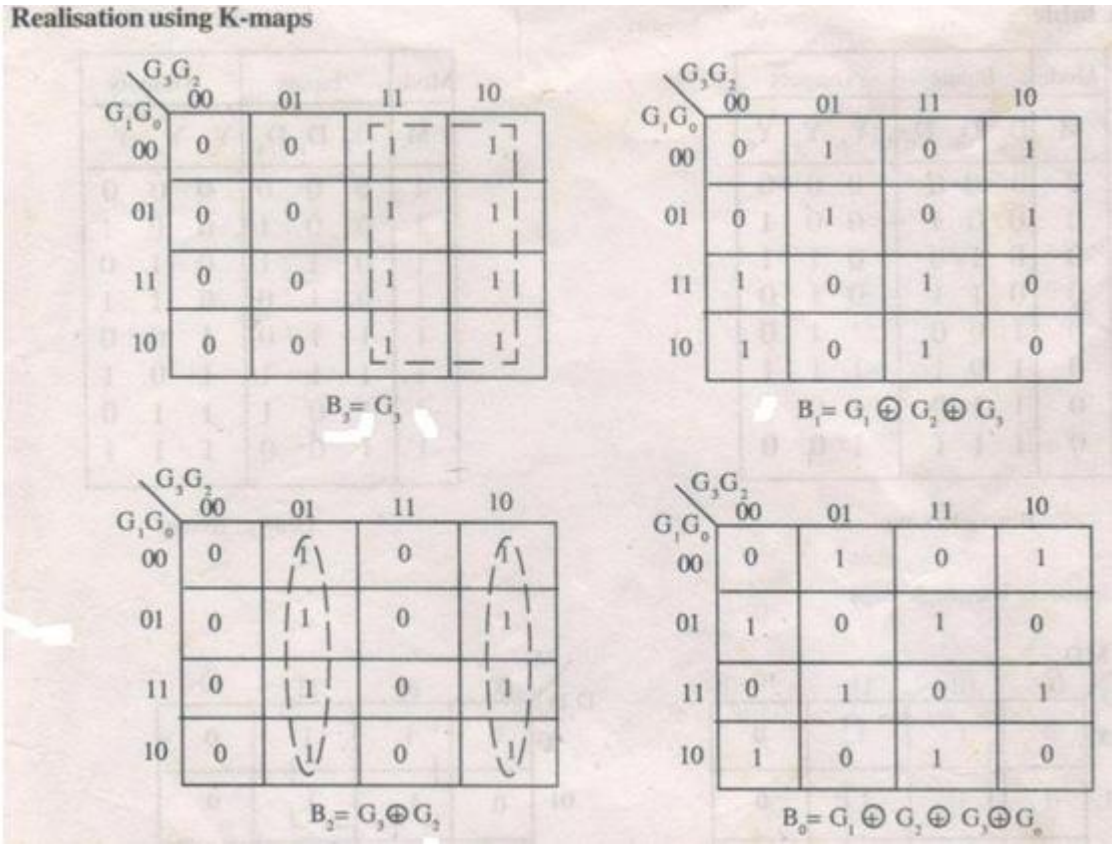

B3
B2
B1
B0
G3
G2
G1
G0

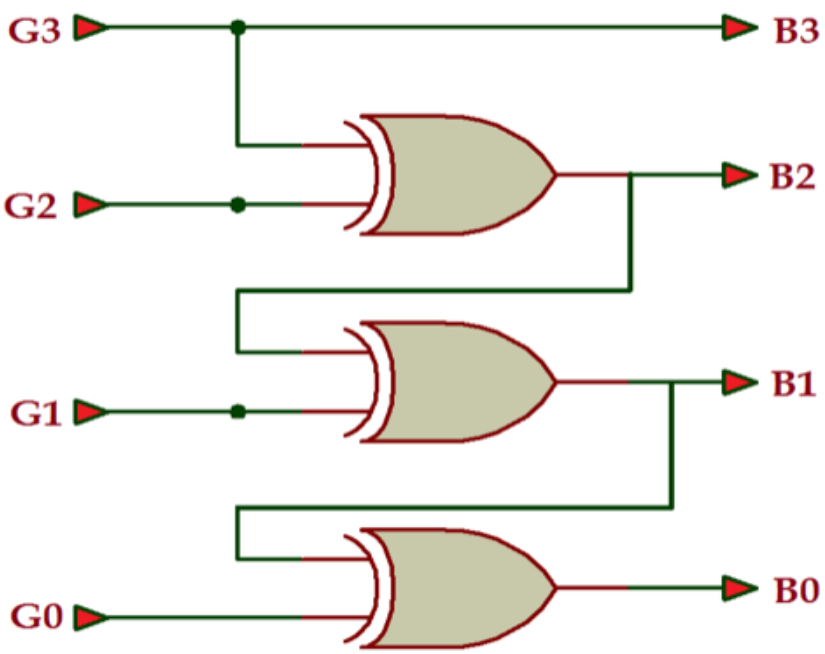
Gray to Binary code converter

GRAY CODE INPUT				BINARY OUTPUT			
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

Truthtable

Realisation using K-maps

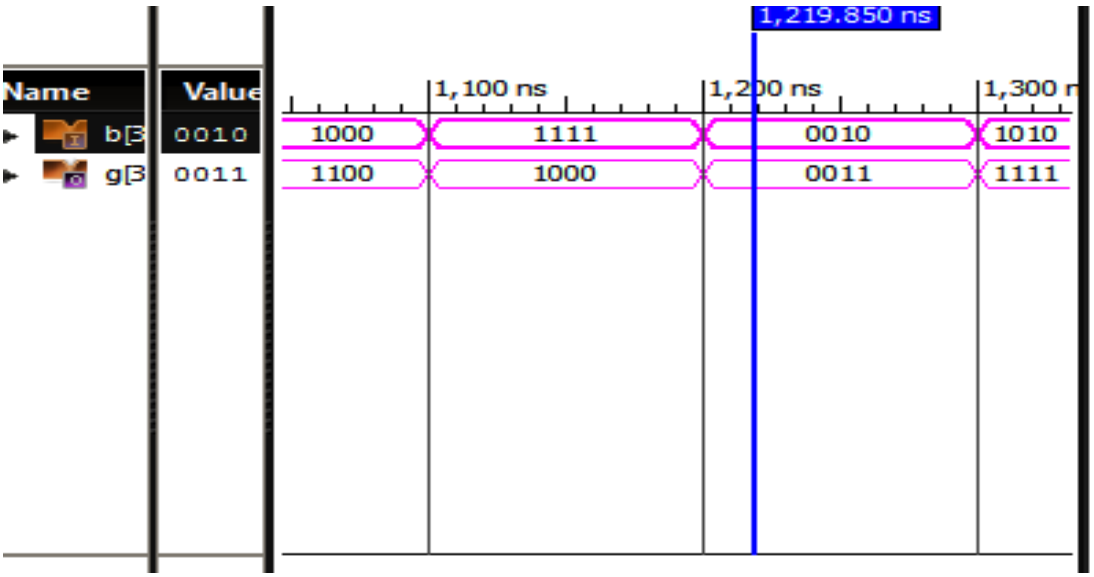




VERILOG PROGRAMS

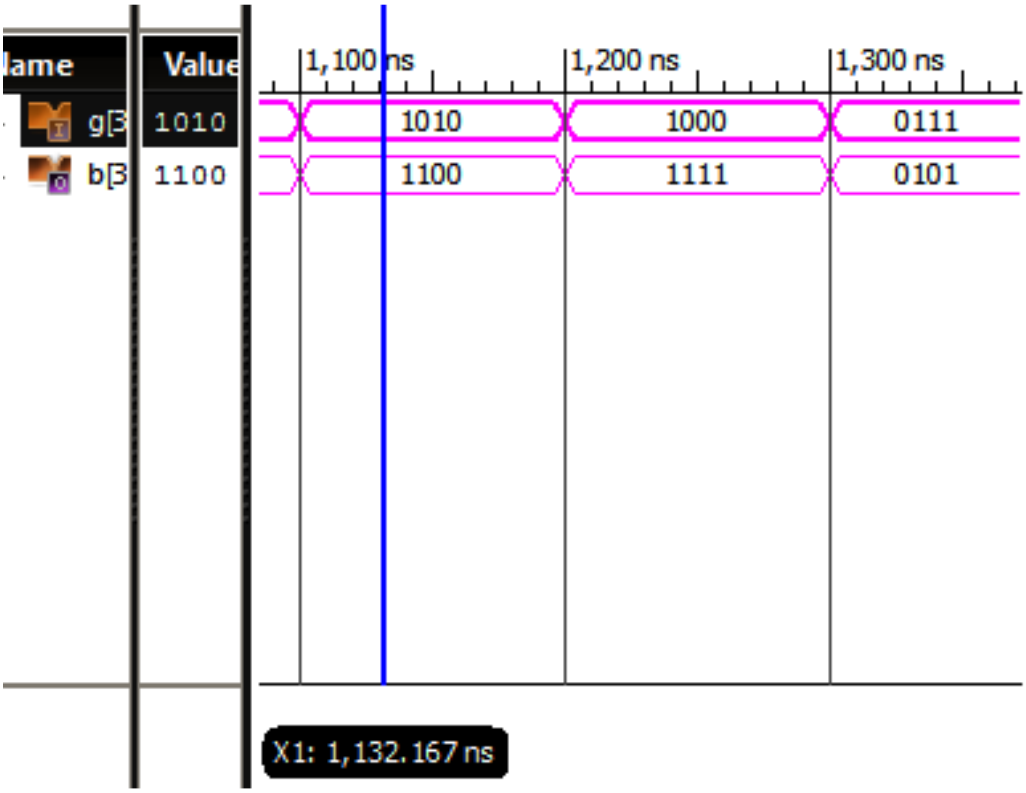
BINARY TO GRAY CODE CONVERTER

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



GRAY TO BINARY CODE CONVERTER

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



RESULT

Developed and implemented verilog module for

1.4 -bit Binary to gray

2.4-bit Gray to Binary .

EXPERIMENT NO: 15

VERILOG MODULE FOR MULTIPLEXER AND DEMULTIPLEXER

AIM

Study and Develop verilog module for a

1. 4*1 MUX.

2. 1*4 DEMUX.

THEORY

Multiplexer is a combinational circuit which can select any one of the inputs and route it to the output. A multiplexer has data input lines, data select lines and output.

The logic symbol of a 4 line to 1 line multiplexer is shown in figure. According to the two bit binary code on the data select inputs, corresponding data input line will be selected and routed to the output. For example, if S1S0 is 00, D0 will be selected, if S1S0 is 01, D1 will be selected and so on.

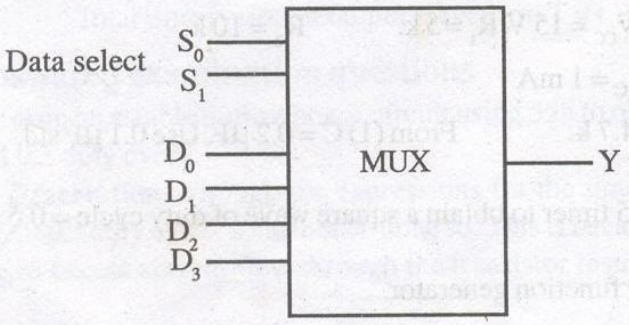
From the truth table it can be seen that the output

$$Y = D_0\overline{S_1}\overline{S_0} + D_1\overline{S}_1S_0 + D_2S_1\overline{S}_0 + D_3S_1S_0$$

This Boolean expression can be realized using gates

CIRCUIT DIAGRAM

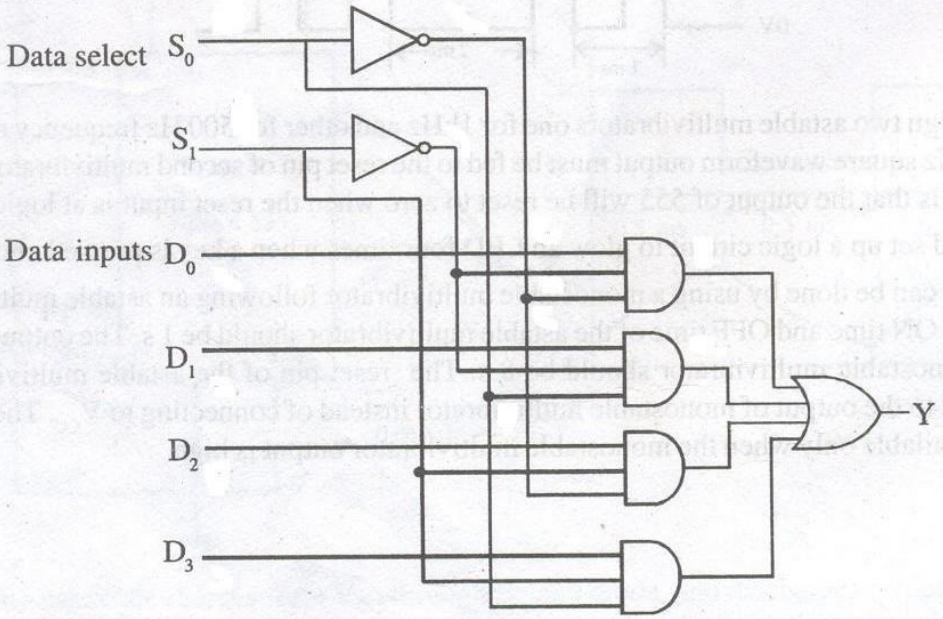
Logic symbol



Truth table

S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

Circuit diagram



VERILOG PROGRAMS

MUX 4x1

Gate level /Structural modeling

```
module mux4x1_gate_level(  input a,
 input b,
 input c,
 input d,
 input s0,
 input s1,
 output y
 );
wire n1,n2,n3,n4,n5,n6;
not (n1,s1);
not (n2,s0);
and (n3,a,n1,n2);
and(n4,b,n1,s0);
and (n5,c,s1,n2);
and(n6,d,s1,s0);
or(y,n3,n4,n5,n6);
endmodule
```

Dataflow modeling

```
module mux4x1(
 input s0,
 input s1,
 input a,
```

```
 input b,
 input c,
 input d,
 output y
 );
assign y=(a&(~s1)&(~s0))|(b&(~s1)&s0)|(c&s1&(~s0))|(d&s1&s0);
 endmodule
```

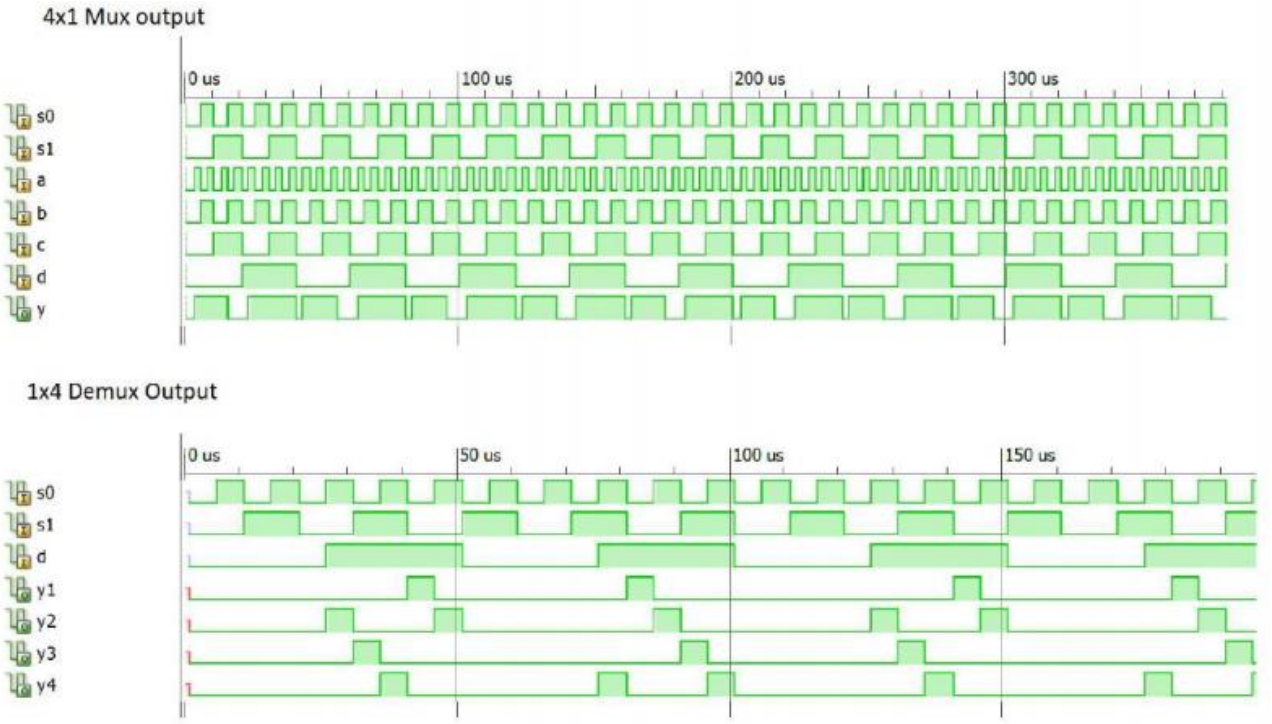
DEMUX 1x4

Gate level /Structural modeling

```
module demux1x4(
 input s1,
 input s0,
 input d,
 output y1,
 output y2,
 output y3,
 output y4
 );
not (n1,s1);
not (n2,s0);
and (y1, d,n1,n2);
and( y2,d,n1,s0);
and(y3,d,s1,n2);
and(y4,d,s1,s0);
endmodule
```

Dataflow modeling

```
module demux1x4_dataflow(
 input s0,
 input s1,
 input d,
 output y1,
 output y2,
 output y3,
 output y4
);
assign y1=(d&(~s1)&(~s0)); assign y2=(d&(~s1)&s0);
assign y3=(d&s1&(~s0)); assign y4=(d&s1&s0);
endmodule
```



RESULT

Developed and implemented verilog module for

1.4*1 MUX.

2.1*4 DEMUX.

VIVA QUESTIONS

Gates

1)Electronic circuits that operate on one or more input signals to produce standard output _____

a)Series circuits b) Parallel circuits c) Logic signals d) Logic gates

Ans: d

2)A _____ gate gives the output as 1 only if all the inputs signals are 1.

a) AND b) OR c) NOR d) Ex-OR

Ans:a

3) The Boolean expression of an OR gate is _____

a)A.B b) A'B+AB' c) A+B d) A'B'

Ans: c

4)The gate which is used to reverse the output obtained is _____

a)NOR b)NAND c)EX-OR d) NOT

Ans:d

5)Which of the following gate will returns a 1 only if both the inputs are 0?

a)AND b)OR c)NAND d)EX-OR

Ans: c

6)The universal gate is ……

a)AND gate b)Or gate c)NAND gate d)None of the above

Ans: c

7)The inputs of a NAND gate are connected together. The resulting circuit is …….

a)Or gate b)AND gate c)NOT gate d)None of the above

Ans: c

8)The NOR gate is OR gate followed by ……

a)AND gate b)NAND gate c)NOT gate d)None of the above

Ans: c

9)Digital circuits can be made by the repeated use of ……
a)OR gate b)NOT gate c)NAND gate d)None of the above

Ans: c

10)A single transistor can be used to build …….. gates .
a)OR b) NOT c)AND d) NAND

Ans: b

Adders and Subtractors

1)In parts of the processor, adders are used to calculate _____
a)Addresses b)Table indices c)Increment and decrement operators
d)All of the Mentioned

Ans: d

2)Total number of inputs in a half adder is _____
a)2 b)3 c)4 d)1

Ans: a

3)In which operation carry is obtained?
a)Subtraction b)Addition c)Multiplication d) Both addition and subtraction

Ans: a

4)If A and B are the inputs of a half adder, the sum is given by _____
a)A AND B b)A OR B c)A EX-OR B d)A EX-NOR B

Ans: c

5)If A and B are the inputs of a half adder, the carry is given by _____
a)A AND B b) A OR B c)A EX-OR B d)A EX-NOR B

Ans: a

6)Half subtractor is used to perform subtraction of _____
a)2 bits b)3 bits c)4 bits d)5bits

Ans: a

7)For subtracting 1 from 0, we use to take a _____ from neighbouring bits.
a)Carry b)Borrow c)Input d)Output

Ans: b

8)How many outputs are required for the implementation of a subtractor?
a)1 b)2 c)3 d)4

Ans: b

9)Let the input of a subtractor is A and B then what the output will be if A = B?
a)0 b)1 c)A d)B

Ans: a

10)Let A and B is the input of a subtractor then the difference output will be _____
a) A EX-OR B b)A AND B c)A OR B d)A EX-NOR

Ans: a

Flip-Flops (FFs)

1)The output of latches will remain in set/reset until _____
a)The trigger pulse is given to change the state
b)Any pulse given to go into previous state
c)They don't get any pulse more
d)The pulse is edge-triggered

Ans: a

2)What is a trigger pulse?
a)A pulse that starts a cycle of operation
b)A pulse that reverses the cycle of operation
c)A pulse that prevents a cycle of operation
d)A pulse that enhances a cycle of operation

Ans: a

3)If Q = 1, the output is said to be _____
a)Set b)Reset c)Previous state d)Current state

Ans: a

4)The sequential circuit is also called _____
a)Flip-Flop b)Latch c)Strobe d)Adder

Ans: a

5)The basic latch consists of _____
a)Two inverters b)Two comparators c)Two amplifiers d)Two adders

Ans: a

MUX and DEMUXs

1)BCD to seven segment conversion is a _____
a)Decoding process b)Encoding process c)Comparing process
d) None of the mentioned

Ans: a

2)Invalid BCD can be made to valid BCD by adding with _____
a)0101 b)0110 c)0111 d)1001

Ans: b

3)Device which converts an input device state into a binary representation of ones or zeros is termed as
a)Encoder b)Decoder c)Multiplexer d)Data Selector

Ans: a

4)A circuit that changes a code into a set of signals is called
a)Encoder b)Decoder c)Multiplexer d)Data Selector

Ans: b

5)Modulo 6 counter can be built using a three-element
a)Shift Register b)Bus c)Flip-Flop d)Trigger

Ans: a

6)A demultiplexer accepts ..... inputs.
a)Single b)Multiple c)Two d)Three

Ans: b

7)In a multiplexer, the selection of a particular input line is controlled by _____
a)Data controller b)Selected lines c)Logic gates d)Both data controller and selected lines

Ans: b

8)If the number of n selected input lines is equal to 2^m then it requires _____ select lines.
a)2 b)m c)n d)2^n

Ans: b

9)Which of the following circuit can be used as parallel to serial converter?
a)Multiplexer b)Demultiplexer c)Decoder d)Digital counter

Ans: a

10)How many select lines would be required for an 8-line-to-1-line multiplexer?
a)2 b)4 c)8 d)3

Ans: d

Counters

1)In digital logic, a counter is a device which _____
a)Counts the number of outputs
b)Stores the number of times a particular event or process has occurred
c)Stores the number of times a clock pulse rises and falls
d)Counts the number of inputs

Ans: b

2)A counter circuit is usually constructed of _____
a)A number of latches connected in cascade form
b)A number of NAND gates connected in cascade form
c)A number of flip-flops connected in cascade
d)A number of NOR gates connected in cascade form

Ans: c

3)Ripple counters are also called _____ .
a)SSI counters b)Synchronous counters c)VLSI counters d)Asynchronous counters

Ans: d

4)The parallel outputs of a counter circuit represent the _____ .
a)Parallel data word b)Clock frequency c)Clock count
d)Counter modulus

Ans: c

5)What is the maximum possible range of bit-count specifically in n-bit binary counter consisting of 'n' number of flip-flops?
a)0 to 2^n b)0 to 2^(n+1) c)0 to 2^(n-1) d)0 to 2^(n+1/2)

Ans: c