BSc. Electronics 3rd Sem.

Digital Electronics & VHDL Lab

Lab Course Code: PS/ELEC/C- 302L

List of experiments

1. (A) Introduction to digital laboratory equipment and ICs.
   (B) To verify the truth table of AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR, logic gates.
2. To design and verify AND, OR, NOT, NOR and XOR gates using NAND/NOR gate only.
3. To convert a Boolean expression into logic gate circuit and assemble it using logic gate IC's.
4. To design and verify Half Adder, Full Adder.
5. To design and Verify Half subtractor and Full subtractor.
6. To design a seven segment display driver.
7. To design & implement 4 bit Adder & Subtractor using Full Adder IC 7483.
8. To design Decoder and Encoders using logic gates.
9. To build a Flip-Flop Circuits using elementary gates (RS, Clocked RS, JK, D-type).
10. To design and verify Multiplexer and Demultiplexer using logic gates.
11. To design a shift register and study serial and parallel shifting of data.
12. Design a counter using D/T/JK Flip-Flop.

List of Experiment

B.Sc. III Semester (Electronic hons) Lab 2

(Digital electronic & Verilog/VHDL Lab)

a.	Introduction to digital laboratory equipment's & IC's.	1A
b.	To verify truth table of AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR logic gates.	1B
.	To design and verify AND, OR, NOT, NOR and XOR gates using NAND / NOR gate only.	2
.	To convert a Boolean expression into logic gate circuit and assemble it using logic gate IC's.	3
.	To Design and verify Half Adder, Full Adder.	4
.	To Design and verify Half Subtractor,Full Subtractor.	5
.	To Design a seven segment display driver.	6
.	To Design & implement 4 bit Adder & Subtractor using Full Adder IC 7483.	7
.	To Design Decoder & Encoders using logic gates.	8
.	To build a Flip- Flop Circuits using elementary gates. (RS, Clocked RS,JK, D-type).	9
0.	To Design and verify Multiplexer and Demultiplexer using logic gates.	10
1.	Design a shift register and study Serial and parallel shifting of data.	11
2.	Design a counter using D/T/JK Flip-Flop.	12

Experiment No: 01A

AIM: Introduction to Digital Laboratory Equipments & IC's

The Breadboard

The breadboard consists of two terminal strips and two bus strips (often broken in the centre). Each bus strip has two rows of contacts. Each of the two rows of contacts are a node. That is, each contact along a row on a bus strip is connected together (inside the breadboard). Bus strips are used primarily for power supply connections, but are also used for any node requiring a large number of connections. Each terminal strip has 60 rows and 5 columns of contacts on each side of the centre gap. Each row of 5 contacts is a node.

You will build your circuits on the terminal strips by inserting the leads of circuit components into the contact receptacles and making connections with 22-26 gauge wire. There are wire cutter/strippers and a spool of wire in the lab. It is a good practice to wire +5V and 0V power supply connections to separate bus strips.

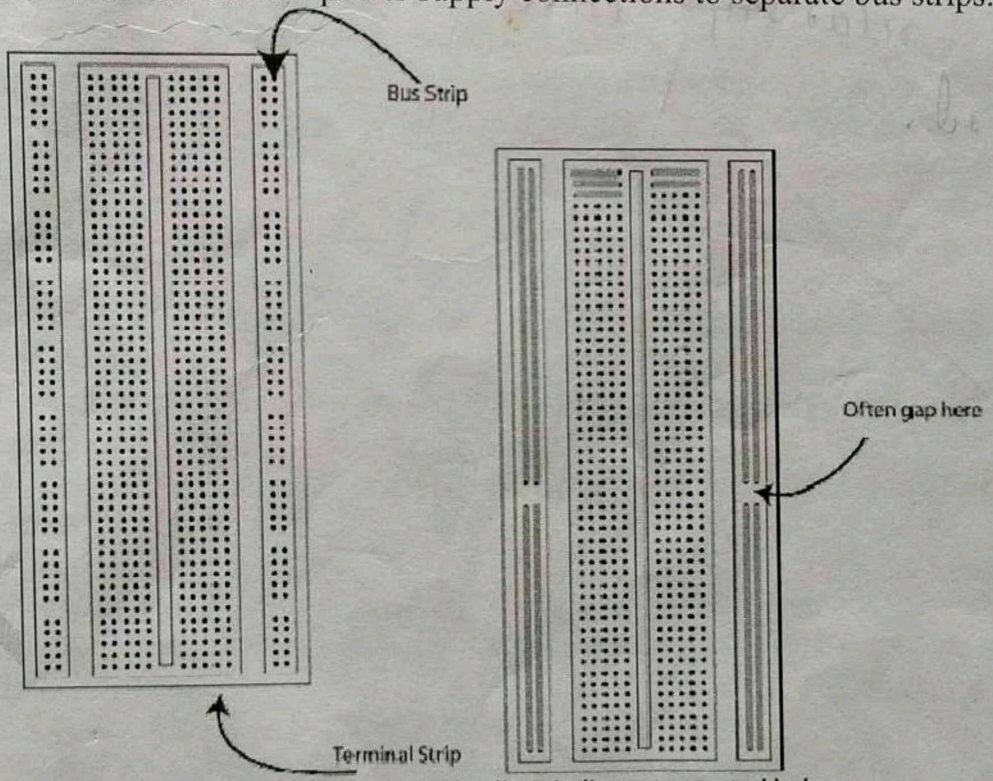


Fig 1. The breadboard. The lines indicate connected holes.

The 5V supply **MUST NOT BE EXCEEDED** since this will damage the ICs (Integrated circuits) used during the experiments. Incorrect connection of power to the ICs could result in them exploding or becoming very hot - with the **possible**

serious injury occurring to the people working on the experiment! Ensure that the power supply polarity and all components and connections are correct before switching on power.

Building the Circuit:

Throughout these experiments we will use TTL chips to build circuits. The steps for wiring a circuit should be completed in the order described below:

1 Turn the power (Trainer Kit) off before you build anything!
2 Make sure the power is off before you build anything!
3 Connect the +5V and ground (GND) leads of the power supply to the power and ground bus strips on your breadboard.
4 Plug the chips you will be using into the breadboard. Point all the chips in the same direction with pin 1 at the upper-left corner. (Pin 1 is often identified by a dot or a notch next to it on the chip package)
5 Connect +5V and GND pins of each chip to the power and ground bus strips on the breadboard.
6 Select a connection on your schematic and place a piece of hook-up wire between corresponding pins of the chips on your breadboard. It is better to make the short connections before the longer ones. Mark each connection on your schematic as you go, so as not to try to make the same connection again at a later stage.
7 Get one of your group members to check the connections, **before you turn the power on.**
8 If an error is made and is not spotted before you turn the power on. Turn the power off immediately before you begin to rewire the circuit.
9 At the end of the laboratory session, collect you hook-up wires, chips and all equipment and return them to the demonstrator.
10. Tidy the area that you were working in and leave it in the same condition as it was before you started.

Common Causes of Problems:

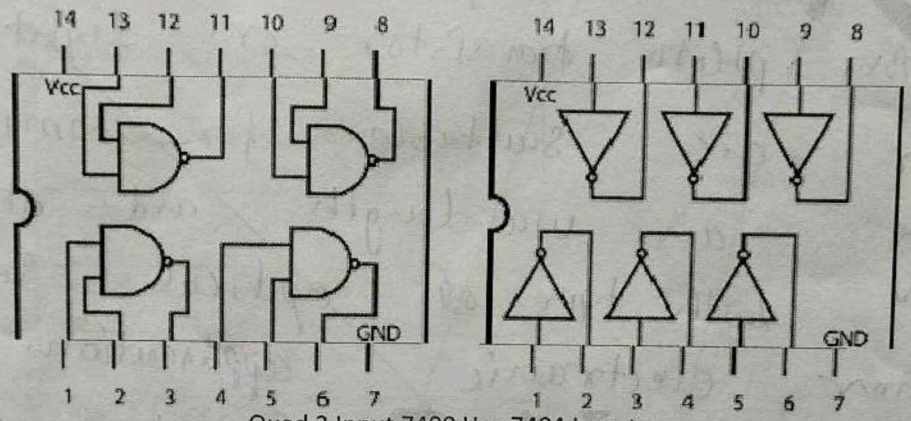
1 Not connecting the ground and/or power pins for all chips.
2 Not turning on the power supply before checking the operation of the circuit.
3 Leaving out wires.
4 Plugging wires into the wrong holes.
5 Driving a single gate input with the outputs of two or more gates
6 Modifying the circuit with the power on.

In all experiments, you will be expected to obtain all instruments, leads, components at the start of the experiment and return them to their proper place

after you have finished the experiment. Please inform the demonstrator or technician if you locate faulty equipment. If you damage a chip, inform a demonstrator, don't put it back in the box of chips for somebody else to use.

Example Implementation of a Logic Circuit:

Build a circuit to implement the Boolean function F = /(/A./B), please note that the notation /A refers to $\overline{A}$. You should use that notation during the write-up of your laboratory experiments.



Quad 2 Input 7400 Hex 7404 Inverter

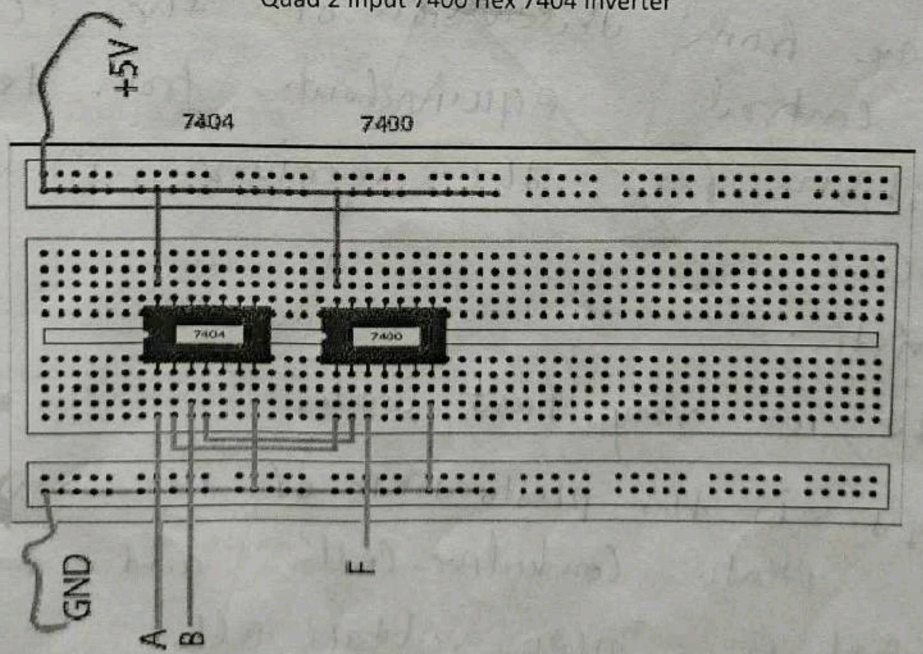


Fig 2. The complete designed and connected circuit

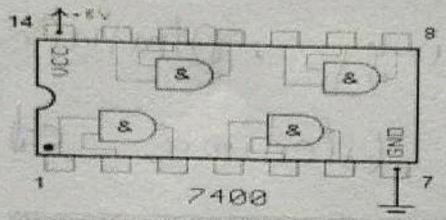
Sometimes the chip manufacturer may denote the first pin by a small indented circle above the first pin of the chip. Place your chips in the same direction, to save confusion at a later stage. Remember that you must connect power to the chips to

get them to work.

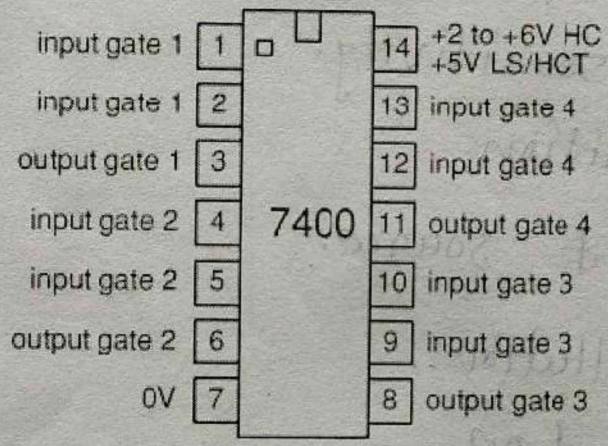
Useful IC Pin details

IC NUMBER	Description of IC
7400	Quad2inputNANDGATE
7401	Quad2inputNANDGate(opencollector)
7402	Quad 2 input NOR Gate
7403	Quad2inputNORGates(opencollector)
7404	Hex Inverts
7421	Dual 4 input AND Gates
7430	8 input NAND Gate
7432	Quad 2 input OR Gates
7486	Quad 2 input EX-OR Gate
74107	Dual j-k Flip Flop
74109	Dual j-k Flip Flop
74174	Hex D Flip Flop
74173	Quad D Flip Flop
7473	Dual j-k Flip Flop
7474	Dual D Flip Flop
7475	Quad Bi-stable latch

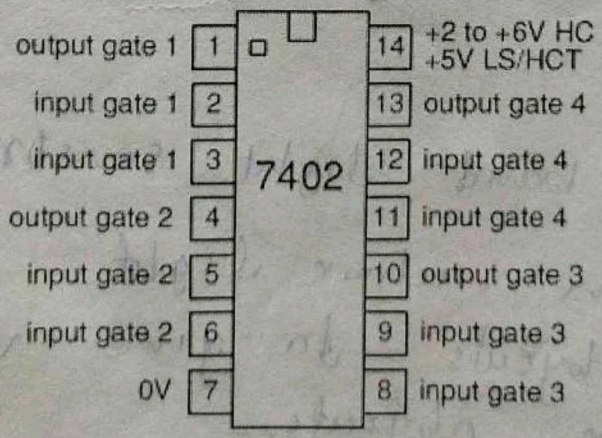
7476	Dual j-k Flip Flop



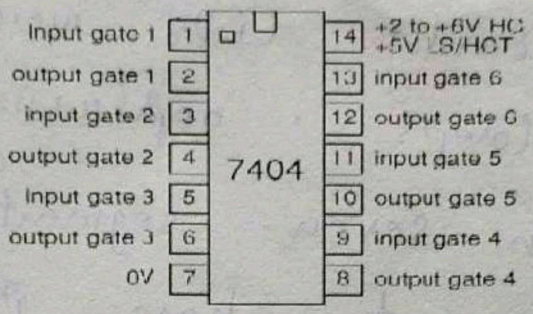
7400(NAND)



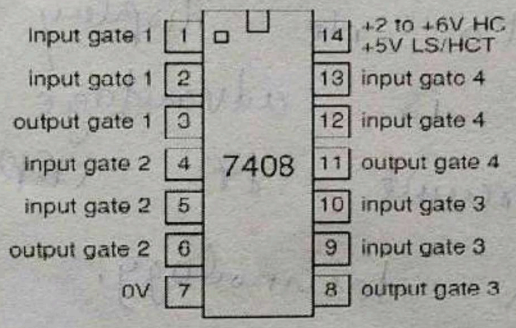
7402(NOR)



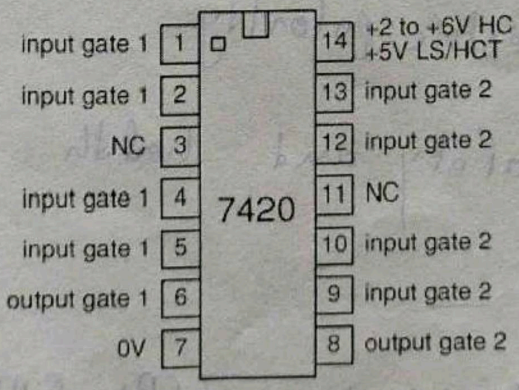
7404(NOT)



7408(AND)

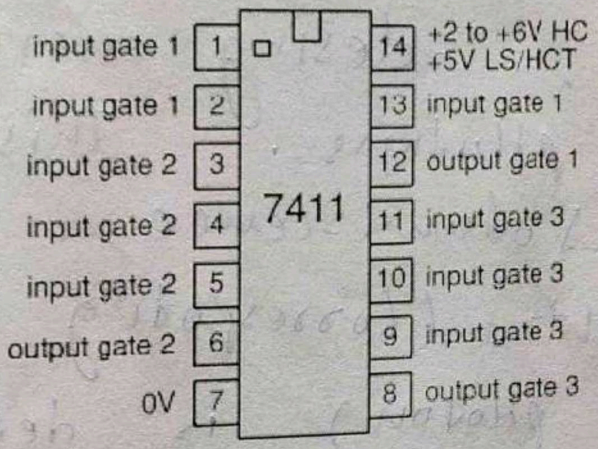


7420(4-i/pNAND)

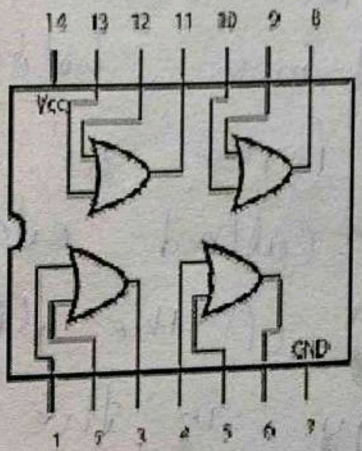


8

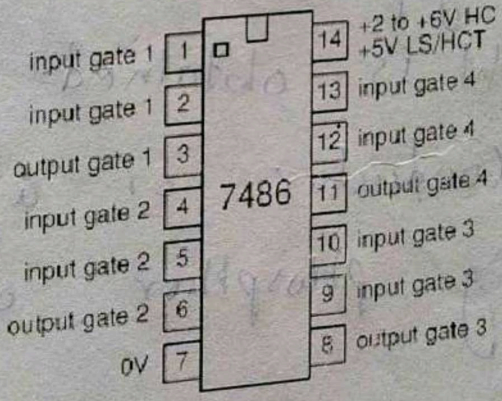
7411(3-i/p AND)

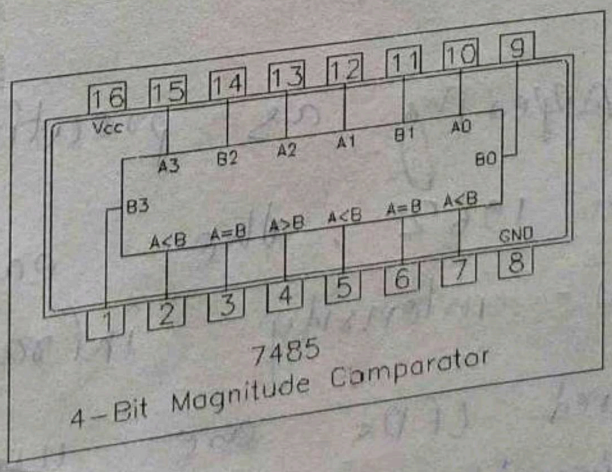


7432(OR)



7486(EX- R)





1/8

Experiment No.1 (1B)

Aim: To study and verify the Truth Tables of AND, OR, NOT, NAND, NOR, EXOR logic gates

Components: IC 7400, 7402, 7404,7408,7432,7486
Apparatus: Prototyping board (breadboard), DC Power Supply, Connecting Wires

Theory:

AND, OR and NOT gates are **basic gates**. XOR and XNOR are **universal gates**. Basically logic gates are electronic circuits because they are made up of number of electronic devices and components. Inputs and outputs of logic gates can occur only in two levels. These two levels are term HIGH and LOW, or TRUE and FALSE, or ON AND off, OR SIMPLY 1 AND 0. A table which lists all possible combinations of input variables and the corresponding outputs is called a 'truth table'. It shows how the logic circuit's output responds to various combinations of logic levels at the inputs.

AND GATE:-

An AND gate has two or more inputs but only one output. The output assumes the logic 1 state only when each one of its inputs is at logic 1 state. The output assumes logic 0 state even if one of its input is at logic 0 state. AND gate is also called an 'all or nothing' gate.

The logic symbol & truth table of two input AND gate are shown in figure 1.a & 1.b respectively. The symbol for AND operation is '.'.

With input variables A & B the Boolean expression for output can be written as;

$$X = A.B$$

Logic symbol:

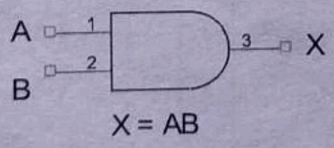


Fig.1.a

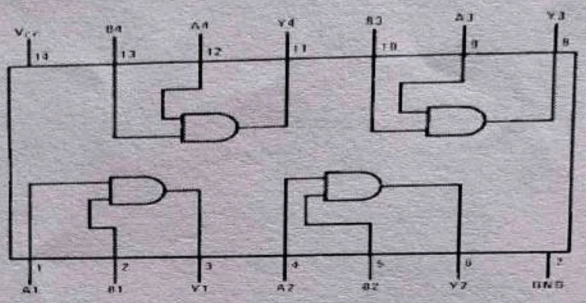
Truth table:

Input		Output
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

Fig.1.b

1a

Pin diagram of IC**74LS08**:



OR GATE

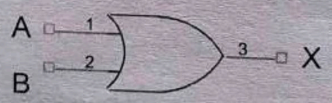
Like an AND gate, an OR gate may have two or more inputs but only one output. The output assumes the logic 1 state, even if one of its inputs is in logic 1 state. Its output assumes logic 0 state, only when each one of its inputs is in logic 0 state. OR gate is also called an 'any or all' gate. It can also be called an inclusive OR gate because it includes the condition 'both the input can be present'.

The logic symbol & truth table of two input OR gate are shown in figure 1.c & 1.d respectively. The symbol for OR operation is '+'.

With input variables A & B the Boolean expression for output can be written as;

$$X = A + B$$

Logic symbol:



X = A + B

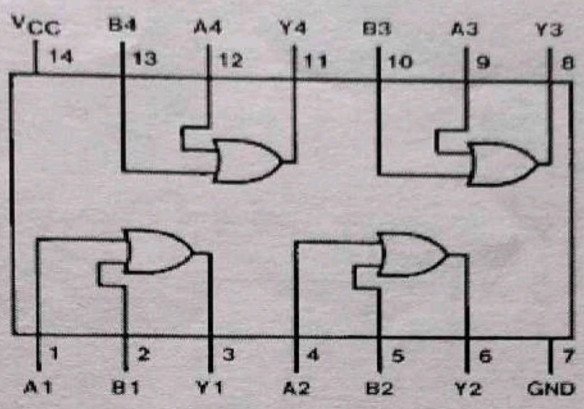
Fig.1.c

Truth table:

Input		Output
A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Fig.1.d

Pin diagram of IC**74LS32**:



NOT GATE

A NOT gate is also known an inverter, has only one input and only one output. It is a device whose output is always the complement of its input. That is the output of a not gate assumes the logic 1 state when its input is in logic 0 state and assumes the logic 0 state when its input is in logic 1 state.

The logic symbol & truth table of NOT gate are shown in figure 1.e & 1.f respectively. The symbol for NOT operation is '-'-'(bar).

With input variable A the Boolean expression for output can be written as;

$$X = \overline{A}$$

This is read as "X is equal to a bar".

Logic symbol:

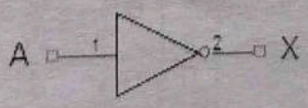


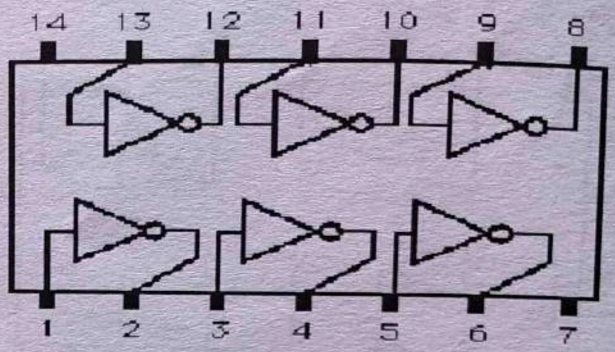
Fig.1.e

Truth table:

Input	Output
A	X
0	1
1	0

Fig.1.f

Pin diagram for IC**74LS04**:



NAND GATE

NAND gate is universal gate. It can perform all the basic logic function. NAND means NOT AND that is, AND output is NOTed.so NAND gate is combination of an AND gate and a NOT gate. The output is logic 0 level, only when each of its inputs assumes a logic 1 level. For any other combination of inputs, the output is logic 1 level. NAND gate is equivalent to a bubbled OR gate.

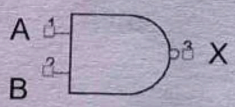
The logic symbol & truth table of two input NAND gate are shown in figure 1.g & 1.h respectively.

With input variables A & B the Boolean expression for

output can be written as;

$$X = \overline{A.B}$$

Logic symbol:



$X = \overline{AB}$

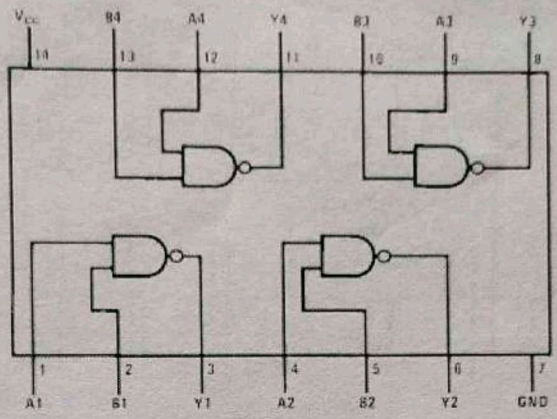
Fig.1.g

Truth table:

Input		Output
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

Fig.1.h

Pin diagram for IC**74LS00**:

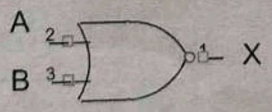


NOR GATE

NOR gate is universal gate. It can perform all the basic logic function. NOR means NOT OR that is, OR output is NOTed.so NOR gate is combination of an OR gate and a NOT gate. The output is logic 1 level, only when each of its inputs assumes a logic 0 level. For any other combination of inputs, the output is logic 0 level. NOR gate is equivalent to a bubbled AND gate. The logic symbol & truth table of two inputs NOR gate are shown in figure 1.i& 1.j respectively. With input variables A & B the Boolean expression for output can be written as;

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

Logic symbol:



$X = \overline{A + B}$

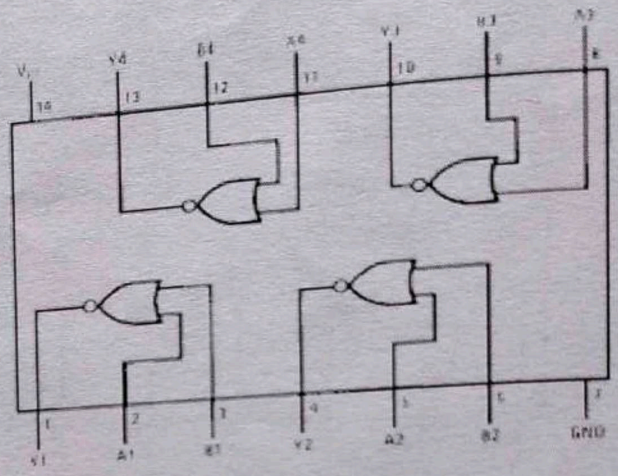
Fig.1.i

Truth table:

Input		Output
A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

Fig.1.j

Pin diagram for IC74LS02:

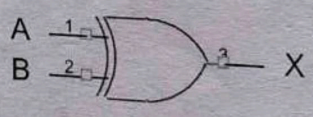


EXCLUSIVE-OR (X-OR) GATE

An X-OR gate is a two input, one output logic circuit, whose output assumes a logic 1 state when one and only one of its two inputs assumes a logic 1 state. Under the condition when both the inputs are same either 0 or 1, the output assumes a logic 0 state. Since an X-OR gate produces an output 1 only when the inputs are not equal, it is called as an anti-coincidence gate or inequality detector. The output of an X-OR gate is the modulo sum of its two inputs. The logic symbol & truth table of two input X-OR gate are shown in figure 1.k & 1.l respectively. The symbol for X-OR operation is '⊕'. With input variables A & B the Boolean expression for output can be written as;

$$X = A \oplus B$$

Logic symbol:



$X = A \oplus B$

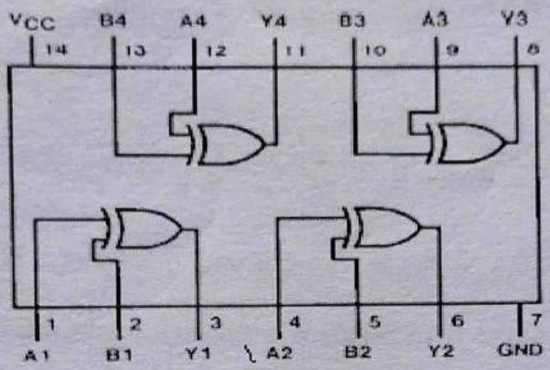
Fig.1.k

Truth table:

Input		Output
A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

Fig.1.l

Pin diagram for IC**74LS86**:



Procedure:-

- Set up the breadboard.
- Test all required ICs.
- Give various combinations of inputs and check the output.
- Repeat the procedure for each IC.

Observation Table: LED ON (RED light): Logic 1
LED OFF (Green Light): Logic 0
Input variables: A, B
Output variable: Y

Sr. No	Input(A) LED	Input(B) LED	Output (OR) $\overline{Y} = A$	Output (AND) Y=AB	Output (OR) Y=A+B	Output (NAND) $\overline{Y} = AB$	Output (NOR) $\overline{Y= A+B}$	Output (XOR) Y=A⊕B
1								
2								
3								
4								

Results and Analysis:
NOT Gate: When logic 1 is applied to one of NOT gate of 7404 IC, then output becomes zero. When input LED is ON (RED), the output LED become OFF (Green) vice versa.

OR Gate: The output of an OR gate is a 1 if one or the other or both of the inputs are 1, but a 0 if both inputs are 0. When One or the other or Both of the input LEDS are ON (RED Light), then output LED is ON(RED) otherwise Output LED is OFF(Green Light)

AND Gate: The output of an AND gate is only 1 if both its inputs are 1. For all other possible inputs the output is 0.When both the LEDS are On, then output LED is ON (RED Light) otherwise Output LED is OFF.

NOR Gate: The output of the NOR gate is a 1 if both inputs are 0 but a 0 if one or the other or both the inputs are 1.

NAND Gate: The output of the NAND gate is a 0 if both inputs are 1 but a 1 if one or the other or both the inputs are 0.

EXOR gate: The output of the XOR gate is a 1 if either but not both inputs are 1 and a 0 if the inputs are both 0 and both 1.

Conclusion: Any Boolean expression can be realized using NOT, AND, OR, NAND,NOR, EXOR gates.

(V7)

Experiment No.2 (2)

Aim: - realization of logic gates using NAND and NOR gates.

Components: - IC 7400, 7402

Apparatus: - Digital trainer kit, wires, probes, etc.

Theory:-

Universal gates: -

The Nand and Nor gates are called as universal gates, because it is possible to implement any Boolean expression with the help of only Nand or only Nor gates.

Hence a user can build any combinational circuit with the help of only Nand gates or only Nor gates.

The NAND & NOR gates are called as 'Universal gates'. Because it is possible to implement any Boolean expression with the help of only NAND or only NOR gate. We can construct AND, OR, NOT, X-OR & X-NOR gates.

The Boolean expression for NAND gate is,

$$X = \overline{AB}$$

The Boolean expression for NOR gate is,

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

This is a great advantage because a user will have to make a stock of only Nand or Nor gates.

All gates using Nand Gate:-

1) Not using Nand:-

The Boolean expression for NOT gate is $A = \overline{A}$ Fig. shows the realization of a NOT gate using a two i/p NAND gate. As both i/p's are connected together we can write i/p A=B=A

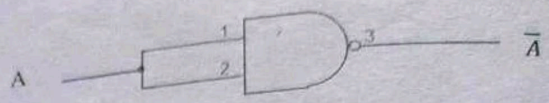
So o/p is given as

$$Y = \overline{A.B}$$

$Y = \overline{A.A}$ $\because A = B$

But A.A=A $\because$ by AND law

$Y = \overline{A}$



2) AND using NAND:-

The Boolean expression for an AND gate is Y=A.B

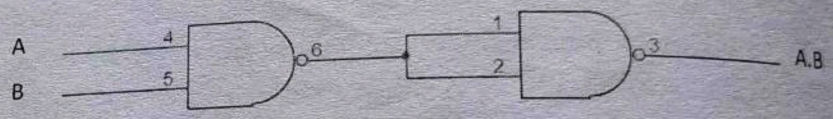
Taking double inversion,

$$Y = \overline{\overline{A.B}}$$

But $\overline{\overline{A}} = A$

Y=A.B

This equation can be realized using only NAND gate as shown in fig.



3) OR using NAND:-

The Boolean expression for an OR gate is Y=A + B

Taking double inversion,

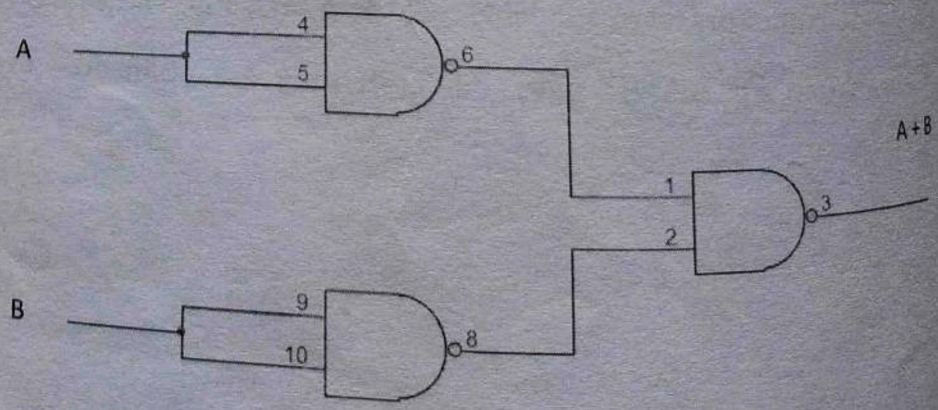
$$Y = \overline{\overline{A + B}}$$

But by DE-Morgan's theorem

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

$$Y = \overline{\overline{A}.\overline{B}}$$

This is required expression for OR gate



4) NOR using NAND:-

The Boolean expression for an NOR gate is $Y = \overline{A + B}$

But by DE-Morgan's theorem

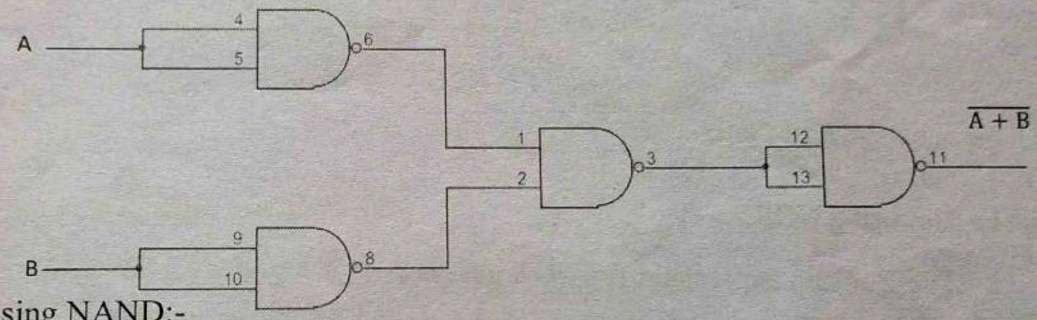
$$\overline{A + B} = \overline{A}.\overline{B}$$

$$Y = \overline{A}.\overline{B}$$

Taking double inversion,

$$Y = \overline{\overline{A}.\overline{B}}$$

This is required expression for NOR gate



5) Ex-OR using NAND:-

The expression for Ex- OR gate is

$$Y = A \oplus B$$

$$Y = \overline{A}.B + A.\overline{B}$$

Taking double inversion

$$Y = \overline{\overline{\overline{A}.B + A.\overline{B}}}$$

Let $\overline{A}.B = X$ and $A.\overline{B} = Z$

$$Y = \overline{\overline{X + Z}}$$

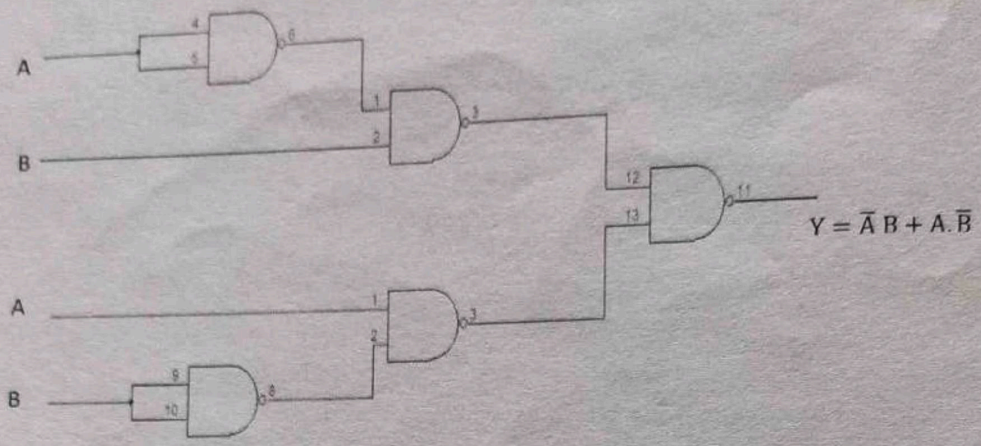
using De – Morgan'stheorem

$$\overline{X + Z} = \overline{X}.\overline{Z}$$

$$Y = \overline{\overline{X}.\overline{Z}}$$

$$Y = \overline{\overline{(\overline{A}B)}.\overline{(A.\overline{B})}}$$

This is required expression for Ex-OR gate using NAND gate.



All gates using NOR Gate:-

1) Not using NOR:-

The Boolean expression for NOT gate is A = $\bar{A}$ Fig. shows the realization of a NOT gate using a two i/p NOR gate. As both i/p's are connected together we can write i/p A=B=A

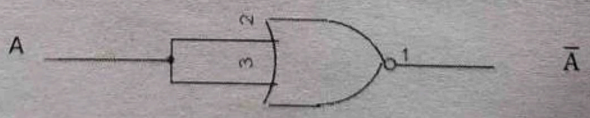
So o/p is given as

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

$$Y = \overline{A + A} \qquad \because A = B$$

But A+A=A $\qquad \because$ by OR law

$$Y = \bar{A}$$



2) OR using NOR:-

The Boolean expression for an OR gate is Y=A+B

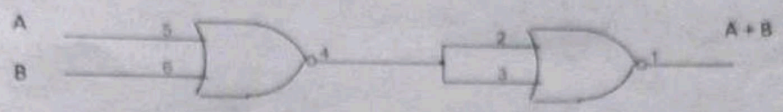
Taking double inversion,

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

But $\bar{\bar{A}} = A$

Y=A+B

This equation can be realized using only NAND gate as shown in fig.



3) AND using NOR:-

The Boolean expression for an AND gate is Y=A.B

Taking double inversion,

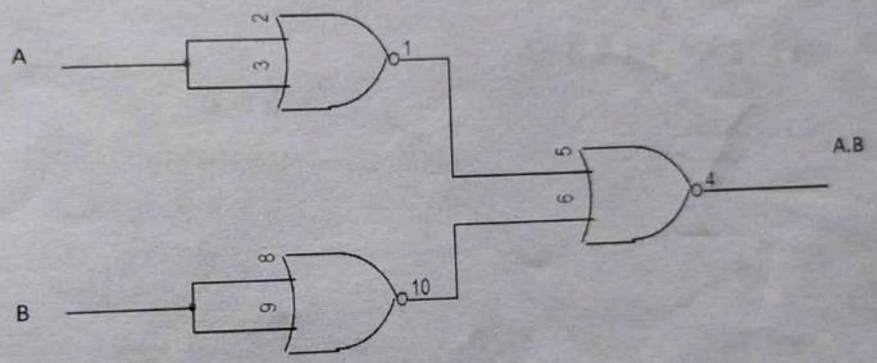
$$Y = \overline{\overline{A.B}}$$

But by DE-Morgan's theorem

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

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

This is required expression for AND gate



4) NAND using NOR:-

The Boolean expression for an NOR gate is $Y = \overline{A.B}$

But by DE-Morgan's theorem

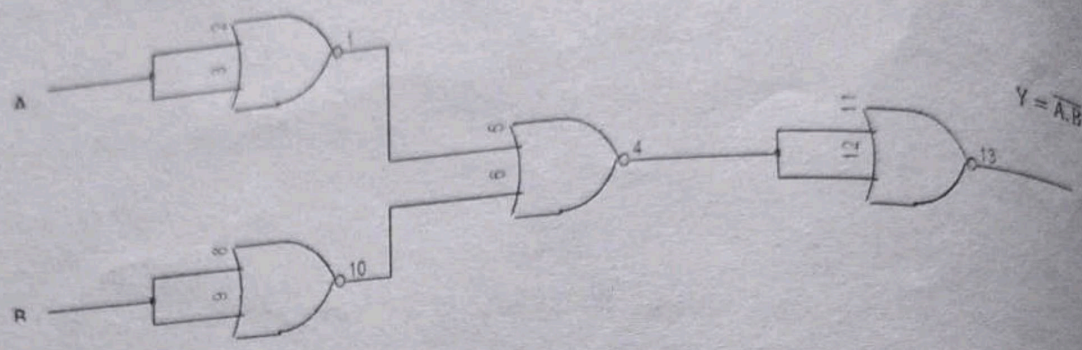
$$\overline{A.B} = \overline{A} + \overline{B}$$

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

Taking double inversion,

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

This is required expression for NAND gate



5)EX-OR using NOR:-

The expression for EX- OR gate is

$$Y = A \oplus B$$

$$Y = \bar{A}.B + A.\bar{B}$$

Taking double inversion

$$Y = \overline{\overline{\bar{A}.B + A.\bar{B}}}$$

Let $\bar{A}.B = X$ and $A.\bar{B} = Z$

$$Y = \overline{\overline{X + Z}}$$

using De – Morgan'stheorem

$$\overline{X + Z} = \bar{X}.\bar{Z}$$

$$Y = \overline{\bar{X}.\bar{Z}}$$

$$Y = \overline{\overline{(\bar{A}.B)}.\overline{(A.\bar{B})}}$$

But $\overline{\bar{A}.B} = A + \bar{B}$ and $A.\overline{\bar{B}} = \bar{A} + B$

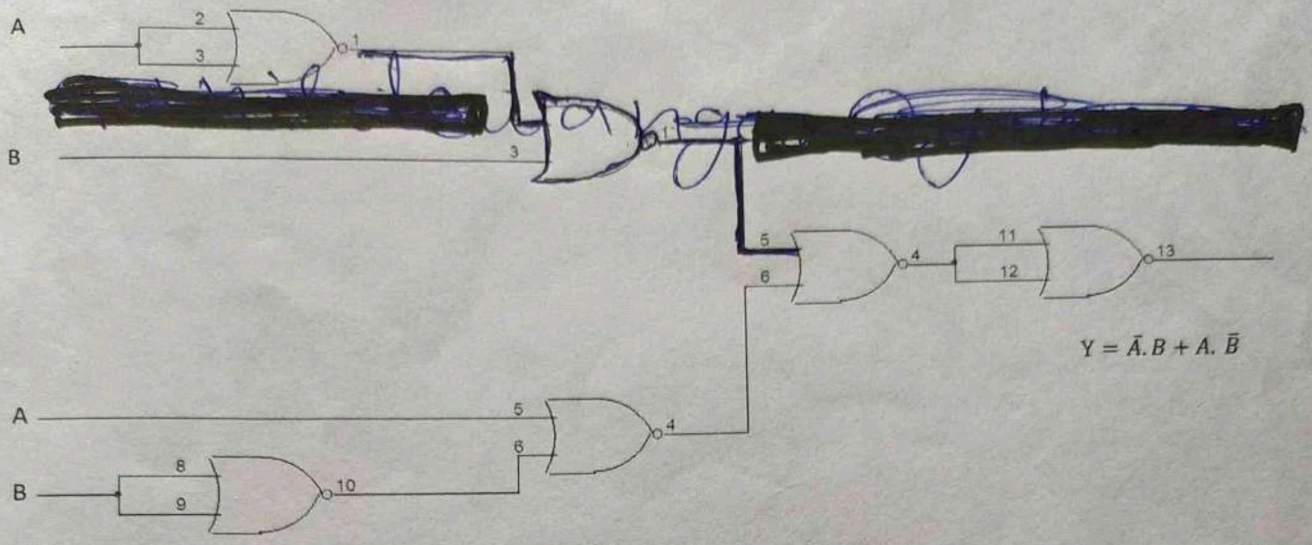
$$Y = \overline{(A + \bar{B}).(\bar{A} + B)}$$

$$Y = \overline{(A + \bar{B})} + \overline{(\bar{A} + B)}$$

Taking double inversion, we get

$$Y = \overline{\overline{(A + \bar{B})} + \overline{(\bar{A} + B)}}$$

This is required expression for EX-OR gate using NOR gate.



Conclusion:-

All the gates are realized using NAND and NOR gates and truth tables are verified.

Experiment No: 03

AIM:- To convert a Boolean expression into logic gate circuit and assemble it using logic gate IC's.

APPARATUS:- LED, IC"s , Wires , 5 volt DC supply, Bread Board/ trainer kit etc.

Theory:

Rules of Boolean Algebra

Table 4-1 lists 12 basic rules that are useful in manipulating and simplifying Boolean expressions. Rules 1 through 9 will be viewed in terms of their application to logic gates. Rules 10 through 12 will be derived in terms of the simpler rules and the laws previously discussed.

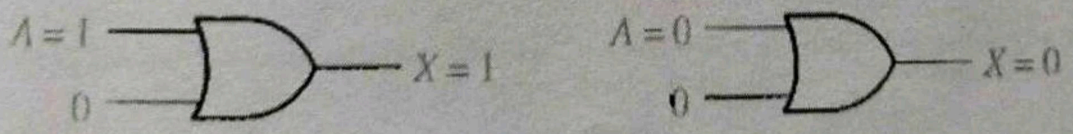
Table 4-1 Basic rules of Boolean algebra.

1. $A + 0 = A$	7. $A \cdot A = A$
2. $A + 1 = 1$	8. $A \cdot \overline{A} = 0$
3. $A \cdot 0 = 0$	9. $\overline{\overline{A}} = A$
4. $A \cdot 1 = A$	10. $A + AB = A$
5. $A + A = A$	11. $A + \overline{A}B = A + B$
6. $A + \overline{A} = 1$	12. $(A + B)(A + C) = A + BC$

A, B, or *C* can represent a single variable or a combination of variables.

Rule 1. A + 0 = A

A variable ORed with 0 is always equal to the variable. If the input variable A is 1, the output variable X is 1, which is equal to A. If A is 0, the output is 0, which is also equal to A. This rule is illustrated in Fig.(4-6), where the lower input is fixed at 0.



A+0=A

Fig.(4-6)

Rule 2. A + 1 = 1

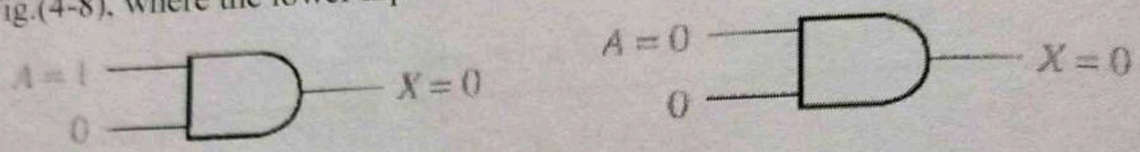
A variable ORed with 1 is always equal to 1. A 1 on an input to an OR gate produces a 1 on the output, regardless of the value of the variable on the other input. This rule is illustrated in Fig.(4-7), where the lower input is fixed at 1.

X=A+1=1

Fig.(4-7)

Rule 3. A . 0 = 0

A variable ANDed with 0 is always equal to 0. Any time one input to an AND gate is 0, the output is 0, regardless of the value of the variable on the other input. This rule is illustrated in Fig.(4-8), where the lower input is fixed at 0.

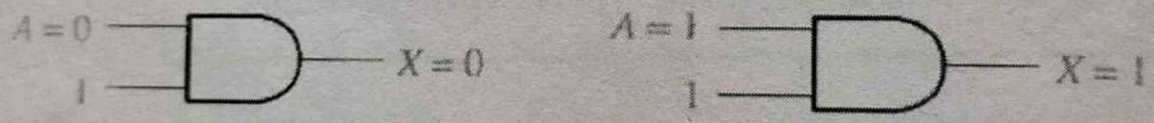


X= A.0 = 0

Fig.(4-8)

Rule 4. A . 1 = A

A variable ANDed with 1 is always equal to the variable. If A is 0 the output of the AND gate is 0. If A is 1, the output of the AND gate is 1 because both inputs are now 1s. This rule is shown in Fig.(4-9), where the lower input is fixed at 1.

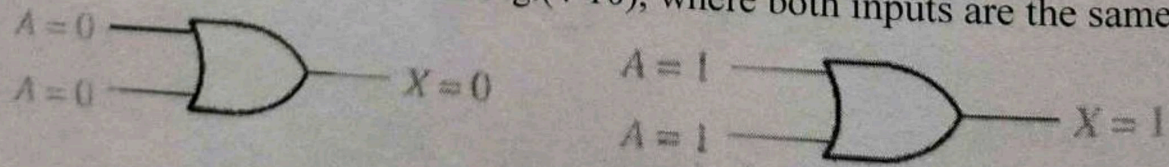


X = A . 1 = A

Fig.(4-9)

Rule 5. A + A = A

A variable ORed with itself is always equal to the variable. If A is 0, then 0 + 0 = 0; and if A is 1, then 1 + 1 = 1. This is shown in Fig.(4-10), where both inputs are the same variable

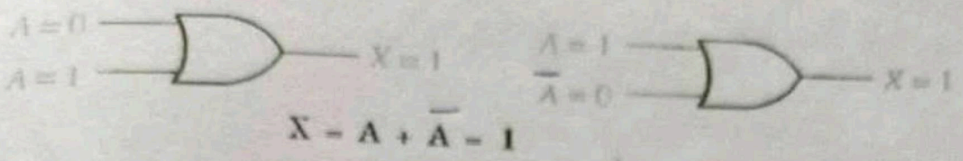


X = A + A = A

Fig.(4-10)

Rule 6. $A + \overline{A} = 1$

A variable ORed with its complement is always equal to 1. If A is 0, then 0 + 0 = 0 + 1 = 1. If A is l, then 1 + 1 = 1+ 0 = 1. See Fig.(4-11), where one input is the complement of the other.

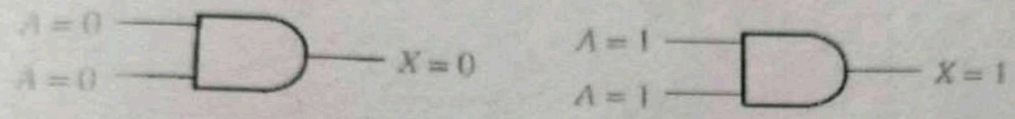


X = A + Ā = 1

Fig.(4-11)

Rule 7. A . A = A

A variable ANDed with itself is always equal to the variable. If A = 0, then 0.0 = 0; and if A = 1, then 1.1 = 1. Fig.(4-12) illustrates this rule.

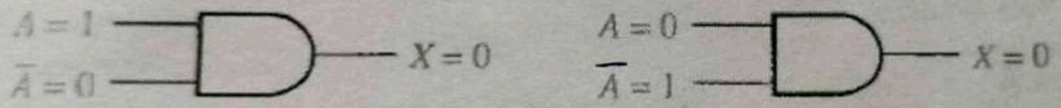


A . A = A

Fig.(4-12)

Rule 8. A . Ā = 0

A variable ANDed with its complement is always equal to 0. Either A or A will always be 0: and when a 0 is applied to the input of an AND gate. The output will be 0 also. Fig.(4-13) illustrates this rule

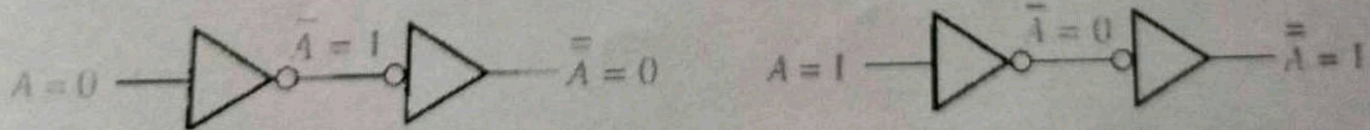


A . A = 0

Fig.(4-13)

Rule 9 $A = \bar{\bar{A}}$

The double complement of a variable is always equal to the variable. If you start with the ariable A and complement (invert) it once, you get A. If you then take A and complement (invert) it, you get A, which is the original variable. This rule is shown in Fig.(4-14) using inverters.



A = A

Fig.(4-14)

Rule 10. A + AB = A

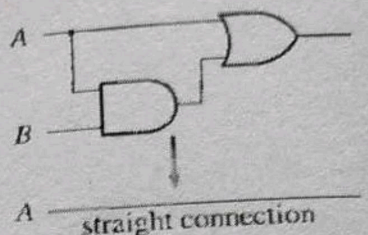
This rule can be proved by applying the distributive law, rule 2, and rule 4 as follows:

$A + AB = A(1 + B)$ Factoring (distributive law)
$= A \cdot 1$ Rule 2: $(1 + B) = 1$
$= A$ Rule 4: $A \cdot 1 = A$
The proof is shown in Table 4-2, which shows the truth table and the resulting logic circuit simplification.

Table 4-2

A	B	AB	A + AB
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

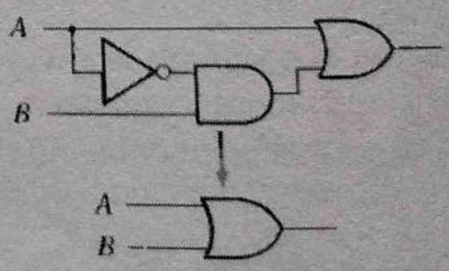
equal



Rule 11. $A + \bar{A}B = A + B$
This rule can be proved as follows:
$A + \bar{A}B = (A + AB) + \bar{A}B$ Rule 10: $A = A + AB$
$= (AA + AB) + \bar{A}B$ Rule 7: $A = AA$
$= AA + AB + A\bar{A} + \bar{A}B$ Rule 8: adding $A\bar{A} = 0$
$= (A + \bar{A})(A + B)$ Factoring
$= 1.(A + B)$ Rule 6: $A + \bar{A} = 1$
$= A + B$ Rule 4: drop the 1
The proof is shown in Table 4-3, which shows the truth table and the resulting logic circuit simplification.
Table 4-3

A	B	$\bar{A}B$	$A + \bar{A}B$	A + B
0	0	0	0	0
0	1	1	1	1
1	0	0	1	1
1	1	0	1	1

equal

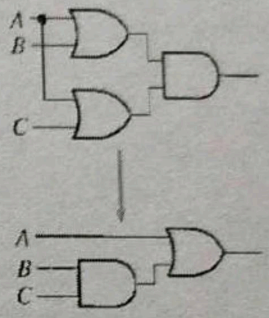


Rule 12. $(A + B)(A + C) = A + BC$
This rule can be proved as follows:
$(A + B)(A + C) = AA + AC + AB + BC$ Distributive law
$= A + AC + AB + BC$ Rule 7: $AA = A$
$= A(1 + C) + AB + BC$ Rule 2: $1 + C = 1$
$= A.1 + AB + BC$ Factoring (distributive law)

= A(1 + B) + BC Rule 2: 1 + B = 1
= A. 1 + BC Rule 4: A . 1 = A
= A + BC
The proof is shown in Table 4-4, which shows the truth table and the resulting logic circuit
simplification.
Table 4-4

A	B	C	A + B	A + C	(A + B)(A + C)	BC	A + BC
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1
1	0	1	1	1	1	0	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

equal



Result :- *Corresponding truth tables of logic circuit are verified.*

PRECAUTIONS:-

1. *Supply should not exceed 5v.*
2. *Connections should be tight and easy to inspect.*
3. *Use L.E.D. with proper sign convention and check it before conneting in circuit.*
4. *Switch off power supply after observation*

Experiment No: 04

Aim :- To Design and verify Half Adder, Full Adder.

APPARATUS:- LED, IC"s(7486,7404,7408,7432) , Wires , 5 volt DC supply, Bread Board etc.

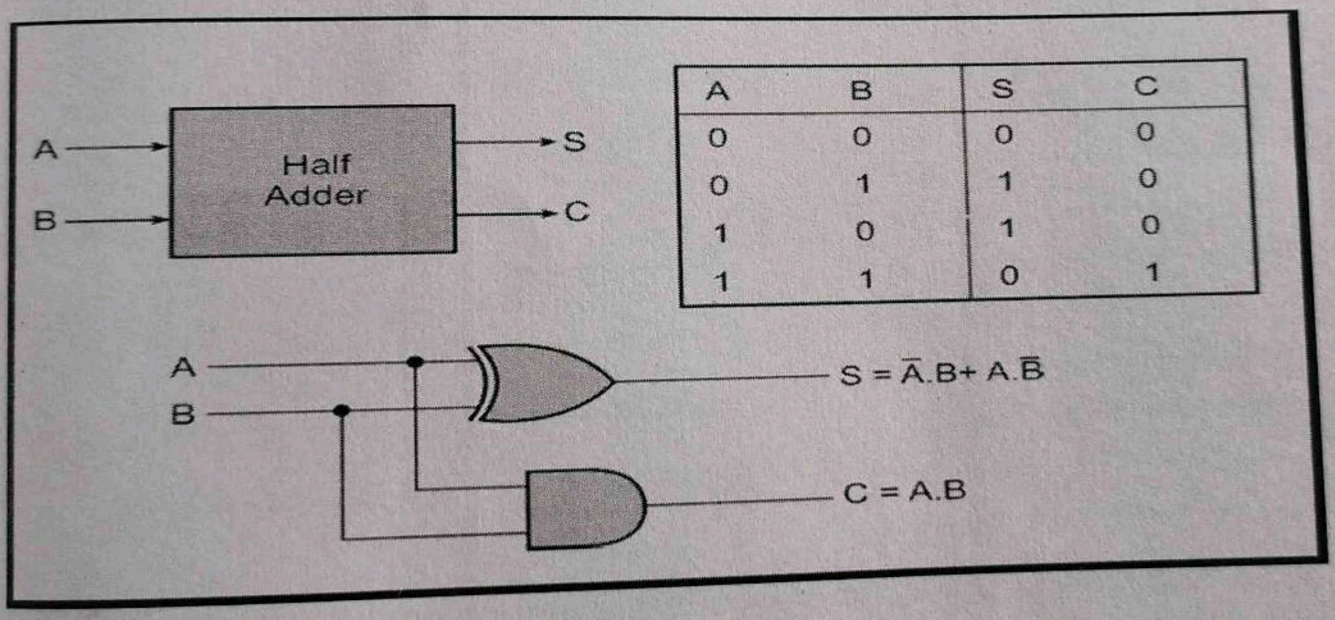
THEORY:

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

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

Full-Adder: The half-adder does not take the carry bit from its previous stage into account. This carry bit from its previous stage is called carry-in bit. A combinational logic circuit that adds two data bits, A and B, and a carry-in bit, Cin, is called a full-adder. The Boolean functions describing the full-adder are:

$S = (A \oplus B) \oplus Cin$ $\qquad$ $C = AB + Cin\,(A \oplus B)$



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

For Carry

A \ B	0	1
0	0	0
1	0	1

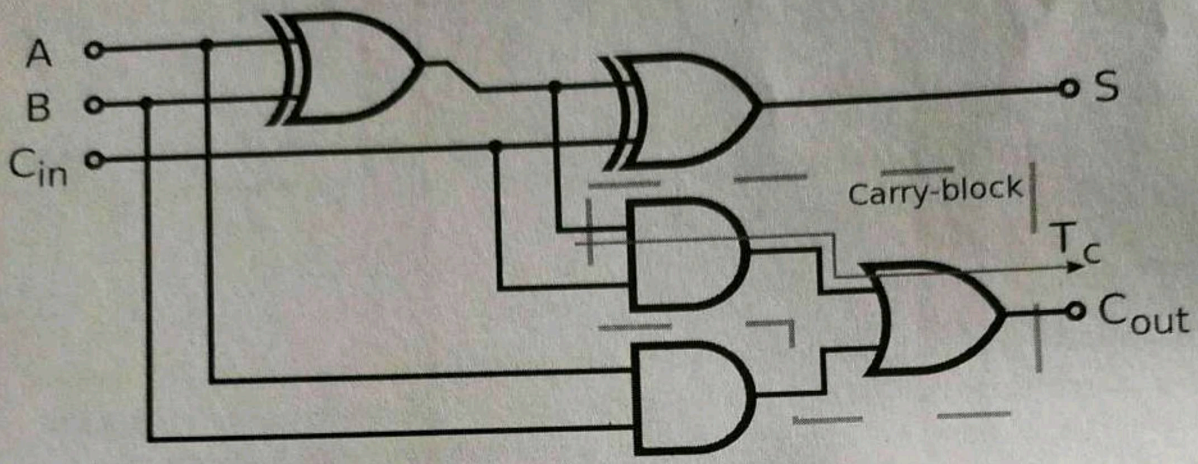
Carry = AB

For Sum

A \ B	0	1
0	0	1
1	1	0

Sum = $A\bar{B}+\bar{A}B$
$= A \oplus B$

Full adder:-



Truth table of full adder :-

Input			Output	
A	B	Cin	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

K-map simplification for carry and sum

For Carry (C_{out})

A \ BC_{in}	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$C_{out} = AB + A C_{in} + B C_{in}$

For Sum

A \ BC_{in}	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$Sum = \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + A\bar{B}\bar{C}_{in} + ABC_{in}$

Result :- *Corresponding truth tables of logic circuit for Half adder ,full adder are verified.*

PRECAUTIONS:-

1. *Supply should not exceed 5v.*
2. *Connections should be tight and easy to inspect.*
3. *Use L.E.D. with proper sign convention and check it before conneting in circuit.*
4. *Switch off power supply after observation.*
5. *Don't remove IC's on bead board unnecessarily.*

Experiment No: 05

Aim :- **To Design and verify Half subtractor ,full subtractor .**

APPARATUS:- LED, IC"s(7486,7404,7408,7432) , Wires , 5 volt DC supply, Bread Board etc.

THEORY:

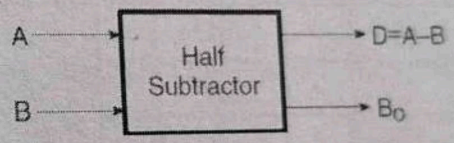
Half Subtractor: Subtracting a single-bit binary value B from another A (i.e. A -B) produces a difference bit D and a borrow out bit B-out. This operation is called half subtraction and the circuit to realize it is called a half subtractor. The Boolean functions describing the halfSubtractor are:

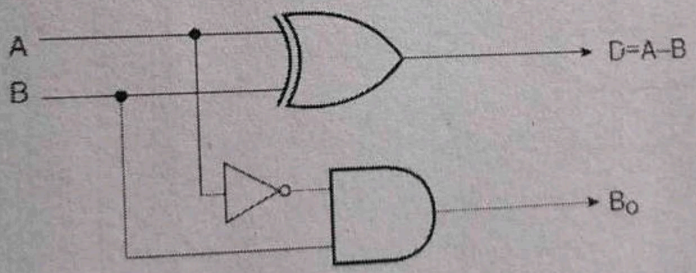
$$S = A \oplus B \qquad Br = A' B$$

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

$$D = (A \oplus B) \oplus Cin \qquad Br = A'B + A' (Cin) + B (Cin)$$

A	B	D	B_O
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0





Half subtractor circuit

For Difference

A \ B	0	1
0	0	(1)
1	(1)	0

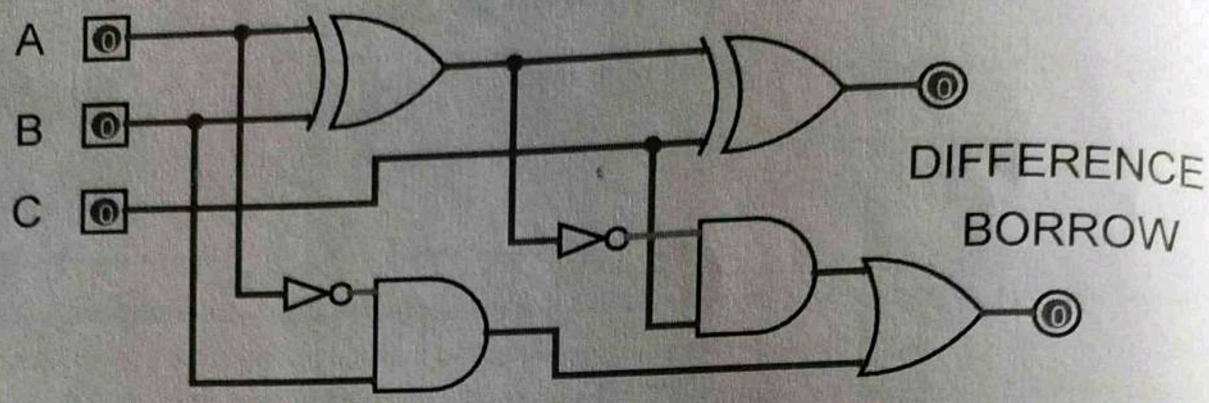
Difference = $A\bar{B} + \bar{A}B$
= $A \oplus B$

For Borrow

A \ B	0	1
0	0	(1)
1	0	0

Borrow = $\bar{A}B$

Fig. 3.19

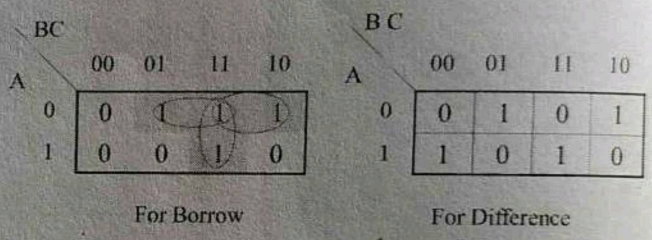


Full subtractor circuit

Truth table of Full subtractor :-

Input			Output	
A	B	C	Difference	Borrow
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

K map simplification for full subtractor



Result :- *Corresponding truth tables of logic circuit for Half* substractor , Full substractor *are verified.*

PRECAUTIONS:-

1. *Supply should not exceed 5v.*
2. *Connections should be tight and easy to inspect.*
3. *Use L.E.D. with proper sign convention and check it before conneting in circuit.*
4. *Switch off power supply after observation.*
5. *Don't remove IC's on bead board unnecessarily.*

Exp. - 06

EXPERIMENT: 06 BCD TO 7-SEGMENT DECODER/DRIVER

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

LEARNING OBJECTIVE:

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

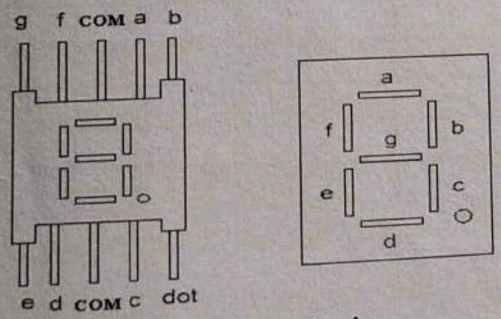
COMPONENTS REQUIRED:

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

THEORY:

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

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



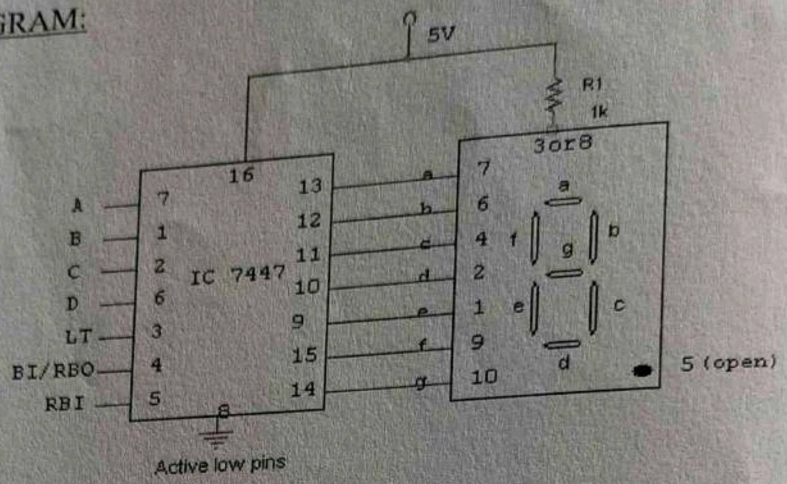
Seven-Segment Display

LED's are basically of two types-

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

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

CIRCUIT DIAGRAM:



TRUTH TABLE:

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

PROCEDURE:

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

VIVA QUESTIONS:

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

Exp. - 07

EXPERIMENT NO. 7
DESIGN OF 4-BIT ADDER AND SUBTRACTOR

AIM:

To design and implement 4-bit adder and subtractor using IC 7483.

APPARATUS REQUIRED:

Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	IC	IC 7483	1
2.	EX-OR GATE	IC 7486	1
3.	NOT GATE	IC 7404	1
3.	IC TRAINER KIT	-	1
4.	PATCH CORDS	-	40

THEORY:

4 BIT BINARY ADDER:

A binary adder is a digital circuit that produces the arithmetic sum of two binary numbers. It can be constructed with full adders connected in cascade, with the output carry from each full adder connected to the input carry of next full adder in chain. The augends bits of 'A' and the addend bits of 'B' are designated by subscript numbers from right to left, with subscript 0 denoting the least significant bits. The carries are connected in chain through the full adder. The input carry to the adder is C_0 and it ripples through the full adder to the output carry C_4.

4 BIT BINARY SUBTRACTOR:

The circuit for subtracting A-B consists of an adder with inverters, placed between each data input 'B' and the corresponding input of full adder. The input carry C_0 must be equal to 1 when performing subtraction.

4 BIT BINARY ADDER/SUBTRACTOR:

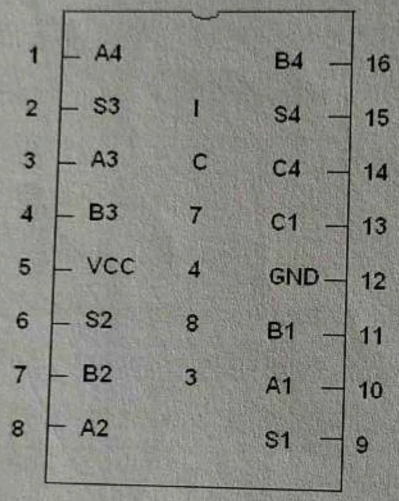
The addition and subtraction operation can be combined into one circuit with one common binary adder. The mode input M controls the operation. When M=0, the circuit is adder circuit. When M=1, it becomes subtractor.

4 BIT BCD ADDER:

Consider the arithmetic addition of two decimal digits in BCD, together with an input carry from a previous stage. Since each input digit does not exceed 9, the output sum cannot be greater than 19, the 1 in the sum being an input carry. The output of two decimal digits must be represented in BCD and should appear in the form listed in the columns.

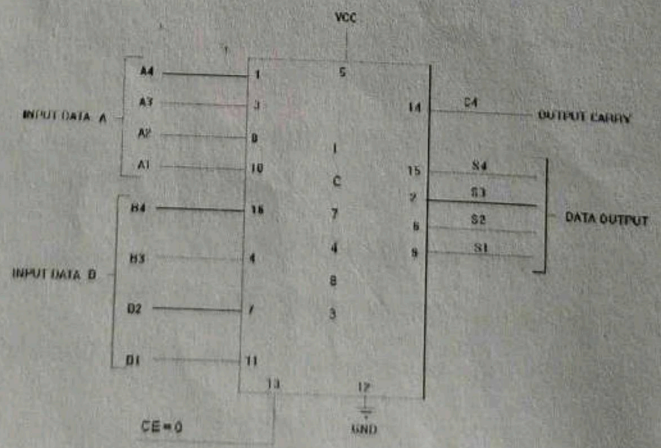
ABCD adder that adds 2 BCD digits and produce a sum digit in BCD. The 2 decimal digits, together with the input carry, are first added in the top 4 bit adder to produce the binary sum.

PIN DIAGRAM FOR IC 7483:



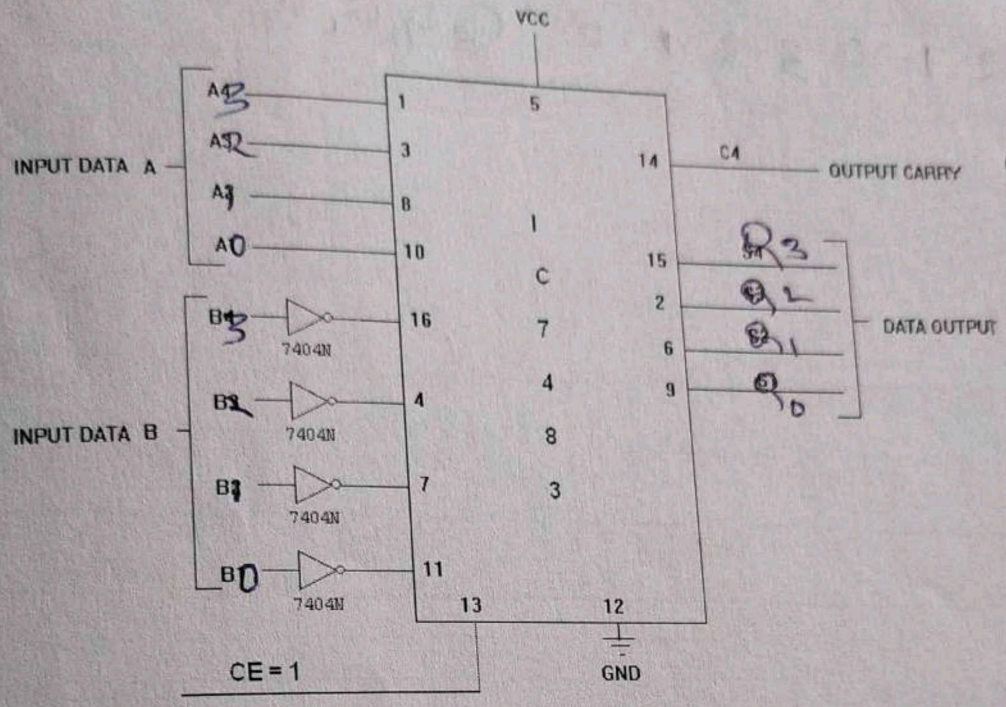
LOGIC DIAGRAM:

4-BIT BINARY ADDER



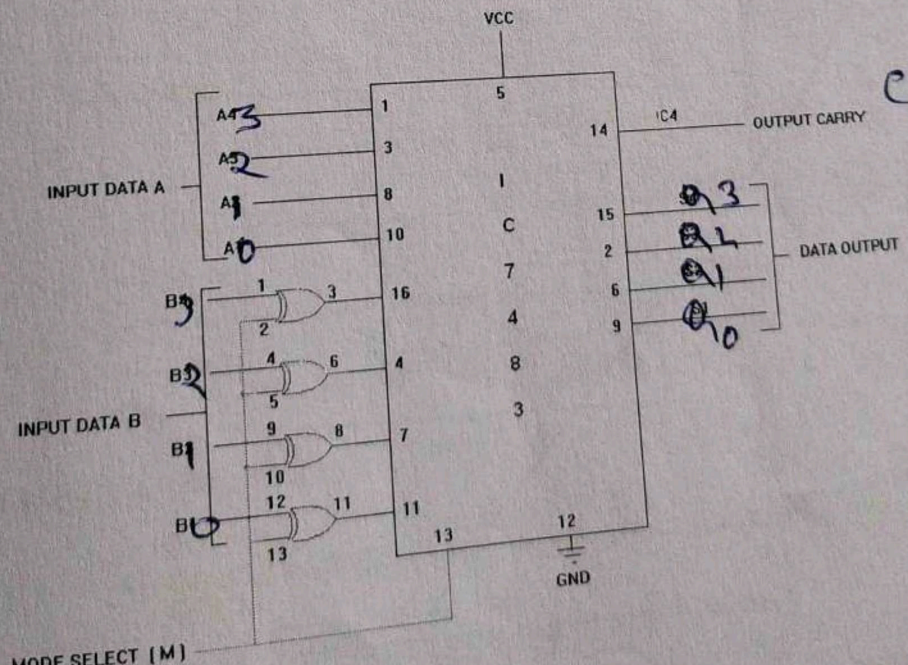
LOGIC DIAGRAM:

4-BIT BINARY SUBTRACTOR



LOGIC DIAGRAM:

4-BIT BINARY ADDER/SUBTRACTOR

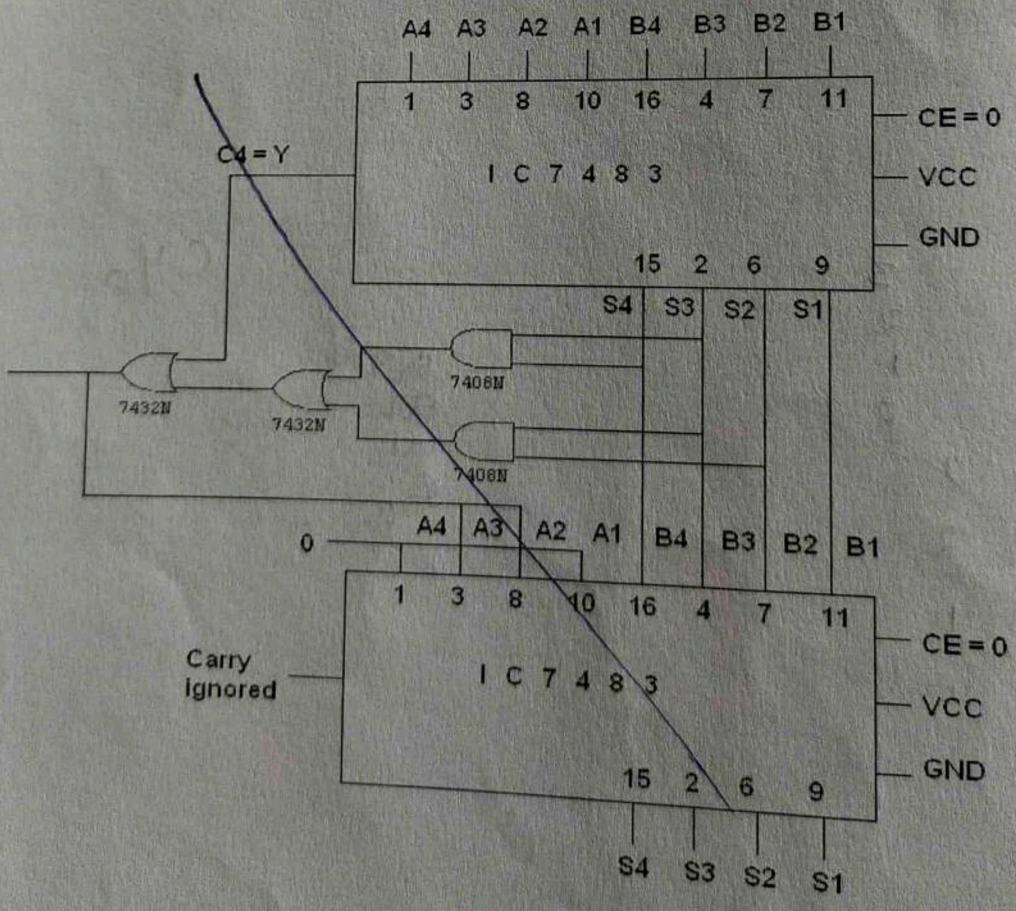


TRUTH TABLE:

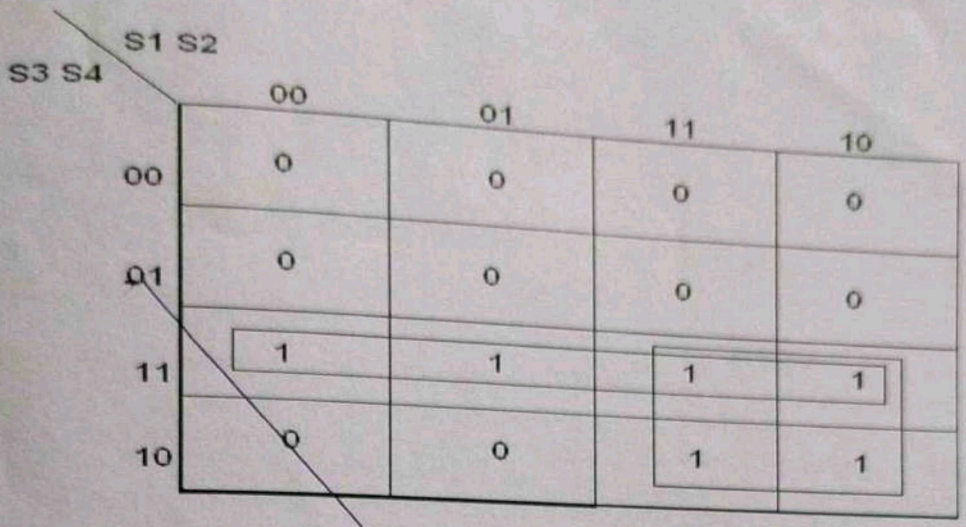
Input Data A				Input Data B				Addition					Subtraction				
A4	A3	A2	A1	B4	B3	B2	B1	C	S4	S3	S2	S1	B	D4	D3	D2	D1
1	0	0	0	0	0	1	0	0	1	0	1	0	1	0	1	1	0
1	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0
0	0	1	0	1	0	0	0	0	1	0	1	0	0	1	0	1	0
0	0	0	1	0	1	1	1	0	1	0	0	0	0	1	0	1	0
1	0	1	0	1	0	1	1	1	0	0	1	0	0	1	1	1	1
1	1	1	0	1	1	1	1	1	1	0	1	0	0	1	1	1	1
1	0	1	0	1	1	0	1	1	0	1	1	1	0	1	1	0	1

LOGIC DIAGRAM:

BCD ADDER



K MAP



$Y = S4\ (S3 + S2)$

TRUTH TABLE:

BCD SUM				CARRY
S4	S3	S2	S1	C
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

PROCEDURE:

(i) Connections were given as per circuit diagram.

(ii) Logical inputs were given as per truth table

(iii) Observe the logical output and verify with the truth tables.

EXPERIMENT: 8 DECODERS

AIM: To realize a decoder circuit using basic gates and to verify, ~~IC 74LS139~~

LEARNING OBJECTIVE:

To learn about working principle of decoder

To learn and understand the working of IC 74LS139

To realize using basic gates as well as universal gates

COMPONENTS REQUIRED:

IC74LS139, IC 7400, IC 7408, IC 7432, IC 7404, IC 7410, Patch chords, & IC Trainer Kit

THEORY:

A decoder is a combinational circuit that connects the binary information from 'n' input lines to a maximum of 2^n unique output lines. Decoder is also called a min-term generator/max-term generator. A min-term generator is constructed using AND and NOT gates. The appropriate output is indicated by logic 1 (positive logic). Max-term generator is constructed using NAND gates. The appropriate output is indicated by logic 0 (Negative logic).

The IC 74139 accepts two binary inputs and when enable provides 4 individual active low outputs. The device has 2 enable inputs (Two active low).

CIRCUIT DIAGRAM:

2:4 DECODER (MIN TERM GENERATOR):

TRUTH TABLE:

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

BOOLAEN EXPRESSIONS:

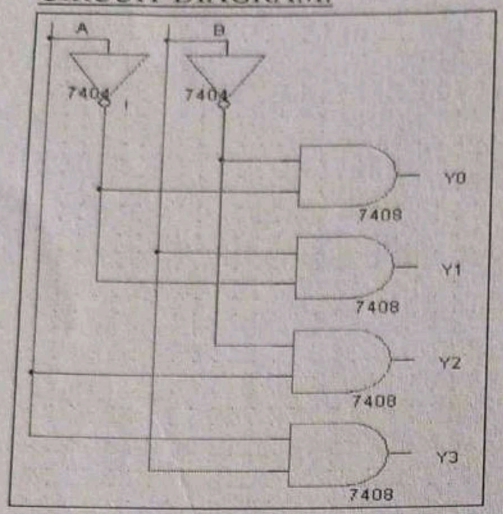
$Y0 = \overline{AB}$

$Y1 = \overline{A}B$

$Y2 = A\overline{B}$

$Y3 = AB$

CIRCUIT DIAGRAM:

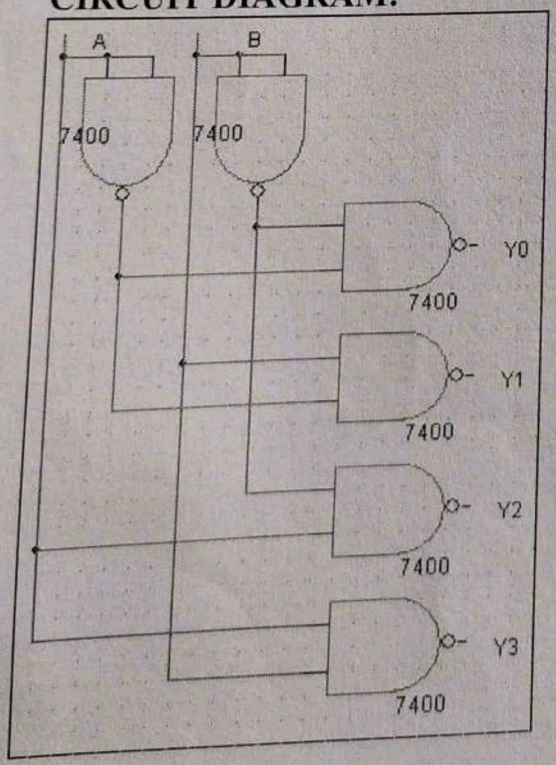


2:4 DECODER (MAX TERM GENERATOR):

TRUTH TABLE:

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

CIRCUIT DIAGRAM:



PROCEDURE:

1. Make the connections as per the circuit diagram.
2. Change the values of G1, G2A, G2B, A, B, and C, using switches.
3. Observe status of Y0, to Y7 on LED's.
4. Verify the truth table.

RESULT: Verified the Operation of 3 to 8 Decoder

VIVA QUESTIONS:

1. What are the applications of decoder?
2. What is the difference between decoder & encoder?
3. For n- 2^n decoder how many i/p lines & how many o/p lines?
4. What are the different codes & their applications?
5. What are code converters?
6. Using 3:8 decoder and associated logic, implement a full adder?
7. Implement a full subtractor using IC 74138?
8. What is the difference between decoder and de-mux?

aim:→ To design basic flip flop using gates & verify their truth Table.

Flip Flops

A digital computer needs devices which can store information. A flip flop is a binary storage device. It can store binary bit either 0 or 1. It has two stable states HIGH and LOW i.e. 1 and 0. It has the property to remain in one state indefinitely until it is directed by an input signal to switch over to the other state. It is also called bistable multivibrator.

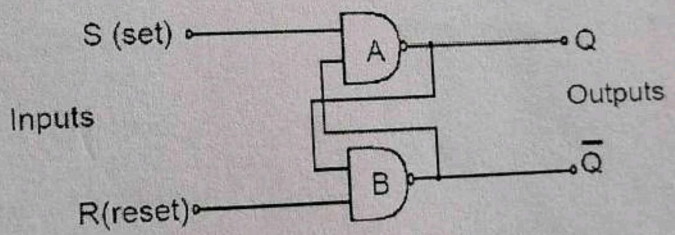
The basic formation of flip flop is to store data. They can be used to keep a record or what value of variable (input, output or intermediate). Flip flop are also used to exercise control over the functionality of a digital circuit i.e. change the operation of a circuit depending on the state of one or more flip flops. These devices are mainly used in situations which require one or more of these three.

Operations, storage and sequencing.

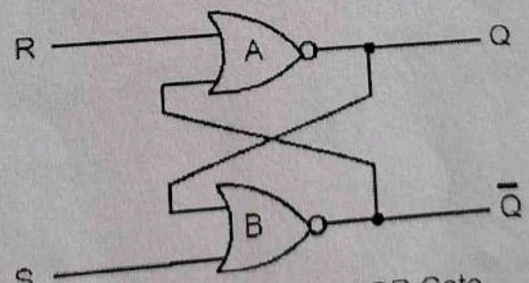
Latch Flip Flop

The R-S (Reset Set) flip flop is the simplest flip flop of all and easiest to understand. It is basically a device which has two outputs one output being the inverse or complement of the other, and two inputs. A pulse on one of the inputs to take on a particular logical state. The outputs will then remain in this state until a similar pulse is applied to the other input. The two inputs are called the Set and Reset input (sometimes called the preset and clear inputs).

Such flip flop can be made simply by cross coupling two inverting gates either NAND or NOR gate could be used Figure 1(a) shows on RS flip flop using NAND gate and Figure 1(b) shows the same circuit using NOR gate.



(a) Latch Flip Flop NAND Gate



(b) RS Latch Flip Flop NOR Gate

Figure 1: Latch R-S Flip Flop Using NAND and NOR Gates

To describe the circuit of Figure 1(a), assume that initially both R and S are at the logic 1 state and that output is at the logic
state.

Now, if Q = 0 and R = 1, then these are the states of inputs of gate B, therefore the outputs of gate B is at 1 (regardless of Q i.e. 0). The output of gate B is connected to an input of gate A so if S = 1, both inputs of gate A are at the 1 state which means that the output of gate A must be 0 (as was originally specified). In other words, the 0 state at Q is continuously enabling gate B so that any change in R has no effect. Also the 1 state at $\overline{Q}$ is continuously enabling gate A so that any change in S is transmitted through to Q. The above conditions constitute one of the stable states of the device referred to as the Reset state since Q = 0.

Now suppose that the R-S flip flop in the Reset state, the S input goes to 0. The output of gate A i.e. Q will go to 1 and R = 1, the output of gates B ($\overline{Q}$) will go to 0 with $\overline{Q}$ now 0 gate A is disabled keeping Q at 1. Consequently, with the 1 state it has no effect on the flip flop whereas a change in R will cause a change in the output of gate B. These conditions constitute the other stable state of the device, called the Set state since Q = 1. Note that the change of S from 1 to 0 has caused the flip flop to change from the Reset state to the Set state.

There is another input condition which has not yet been considered. That is when both the R and S inputs are taken to state 0. When this happens both Q and $\overline{Q}$ will be forced to 1 and will remain so far as long as R and S are kept at 0. But when both inputs return to 1 there is no way of knowing whether the flip flop will latch in the Reset state or the Set state. This condition is said to be indeterminate because of this indeterminate state great care must be taken when using R-S flip flops to ensure that both inputs are not instructed simultaneously.

Table 1: The truth table for the NAND R-S flip flop

Initial Conditions	Inputs (Pulsed)		Final Output	
Q	S	R	Q	
1	0	0	indeterminate	
1	0	1	1	0
1	1	0	0	1
1	1	1	1	0
0	0	0	indeterminate	
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1

or more simply shown in Table 2

Table 2: Simple NAND R-S Flip Flop Truth Table

S	R	Q
0	0	indeterminate
0	1	Set (1)
1	0	Reset(0)
1	1	No Change

When NOR gate are used the R and S inputs are transposed compared with the NAND version. Also the stable states are when S are both 0. A change of state is effected by pulsing the appropriate input to the 1 state. The indeterminate state is when both R and S are simultaneously at logic 1. Table 3 shows this operation

S	R	Q
0	0	No Change
0	1	Reset (0)
0	0	Set (1)
1	1	Indeterminate
1		

Clocked RS Flip Flop

The RS latch flip flop required the direct input but no clock. It is very use full to add clock to control precisely the time at which the flip flop changes the state of its output.

In the clocked R-S flip flop the appropriate levels applied to their inputs are blocked till the receipt of a pulse from an other source called clock. The flip flop changes state only when clock pulse is applied depending upon the inputs. The basic circuit is shown in Figure 2. This circuit is formed by adding two AND gates at inputs to the R-S flip flop. In addition to control inputs Set (S) and Reset (R), there is a clock input (C) also.

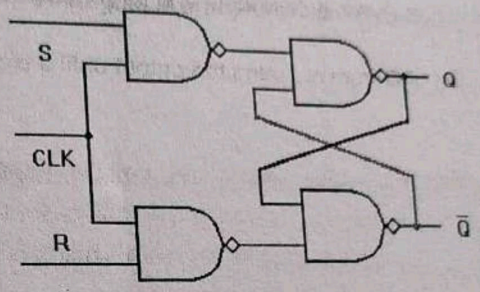


Figure 2: Clocked RS Flip Flop

Table 4: The truth table for the Clocked R-S flip flop

Initial Conditions	Inputs (Pulsed)		Final Output	Comment
Q	S	R	Q (t + 1)	No Change
0	0	0	0	No Change
0	0	1	0	Clear Q
0	1	0	1	Set Q
0	1	1	???	indeterminate
1	0	0	1	No Change
1	0	1	0	Clear Q
1	1	0	1	Set Q
1	1	1	???	indeterminate

The excitation table for R-S flip flop is very simply derived as given below

Table 5: Excitation table for R-S Flip Flop

S		

S	R	Q
1	0	Set (1)
1	1	Indeterminate

D Flip Flop

A D type (Data or delay flip flop) has a single data input in addition to the clock input as shown in Figure 3

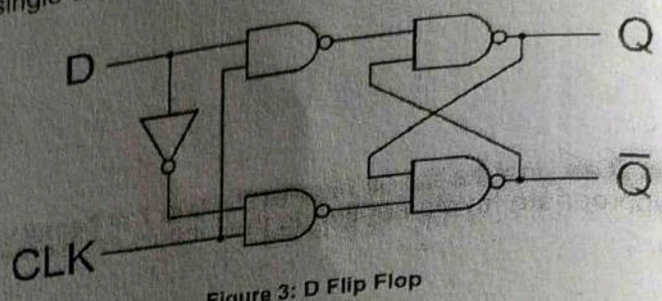


Figure 3: D Flip Flop

Basically, such type of flip flop is a modification of clocked RS flip flop gates from a basic Latch flip flop and it in to a clock RS flip flop. The D input goes directly to S input and its complement through NOT gate, is applie

This kind of flip flop prevents the value of D from reaching the output until a clock pulse occurs. The action of forward as follows.

When the clock is low, both AND gates are disabled, there fore D can change values with out affecting the va other hand, when the clock is high, both AND gates are enabled. In this case, Q is forced equal to D when the low, Q retains or stores the last value of D. The truth table for such a flip flop is as given below in table 6.

Table 6: Truth table for D Flip Flop

S	R	Q(t + 1)
0	0	0
0	1	1
1	0	0
1	1	1

The excitation table for D flip flop is very simply derived given as under.

Table 7: Excitation table for D Flip Flop

S	Q
0	0
1	1

JK Flip Flop

One of the most useful and versatile flip flop is the JK flip flop the unique features of a JK flip flop are:

1. If the J and K input are both at 1 and the clock pulse is

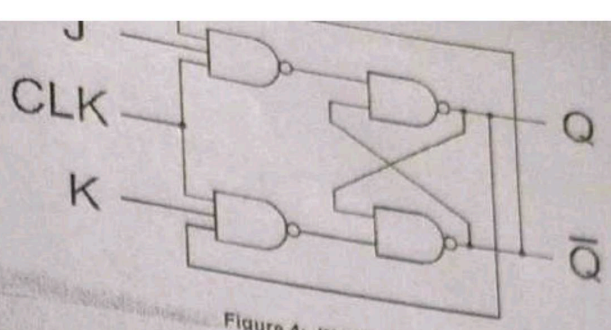


Figure 4: JK Flip Flop

When J = 0 and K = 0

These J and K inputs disable the NAND gates, therefore clock pulse have no effect on the flip flop. In other words, Q returns it last value.

When J = 0 and K = 1,

The upper NAND gate is disabled the lower NAND gate is enabled if Q is 1 therefore, flip flop will be reset (Q = 0 , =1)if not already in that state.

When J = 1 and K = 0

The lower NAND gate is disabled and the upper NAND gate is enabled if is at 1, As a result we will be able to set the flip flop (Q = 1, = 0) if not already set

When J = 1 and K = 1

If Q = 0 the lower NAND gate is disabled the upper NAND gate is enabled. This will set the flip flop and hence Q will be 1. On the other hand if Q = 1, the lower NAND gate is enabled and flip flop will be reset and hence Q will be 0. In other words , when J and K are both high, the clock pulses cause the JK flip flop to toggle. Truth table for JK flip flop is shown in table 8.

Table 8: The truth table for the JK flip flop

Initial Conditions	Inputs (Pulsed)		Final Output
Q	S	R	Q (t + 1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

The ... very simply derived as given in table 9.

Exp. - 10

Experiment No: 10

AIM: To design and implement Multiplexer and Demultiplexer using logic gates and study of IC 74150 and IC 74154.

APPARATUS REQUIRED:

Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	3 I/P AND GATE	IC 7411	2
2.	OR GATE	IC 7432	1
3.	NOT GATE	IC 7404	1
4.	IC TRAINER KIT	-	1
5.	PATCH CORDS	-	-

THEORY:

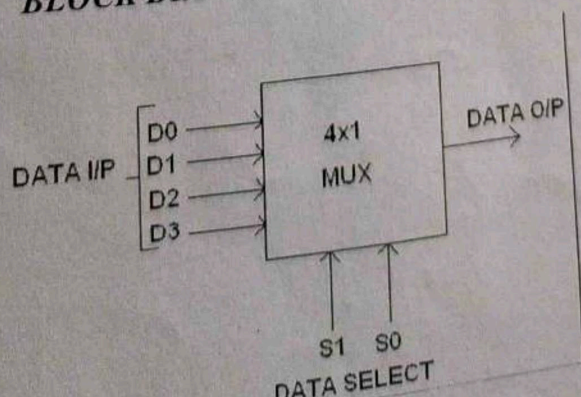
MULTIPLEXER:

Multiplexer means transmitting a large number of information units over a smaller number of channels or lines. A digital multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line. The selection of a particular input line is controlled by a set of selection lines. Normally there are 2n input line and n selection lines whose bit combination determine which input is selected.

DEMULTIPLEXER:

The function of Demultiplexer is in contrast to multiplexer function. It takes information from one line and distributes it to a given number of output lines. For this reason, the demultiplexer is also known as a data distributor. Decoder can also be used as demultiplexer. In the 1: 4 demultiplexer circuit, the data input line goes to all of the AND gates. The data select lines enable only one gate at a time and the data on the data input line will pass through the selected gate to the associated data output line.

BLOCK DIAGRAM FOR 4:1 MULTIPLEXER:

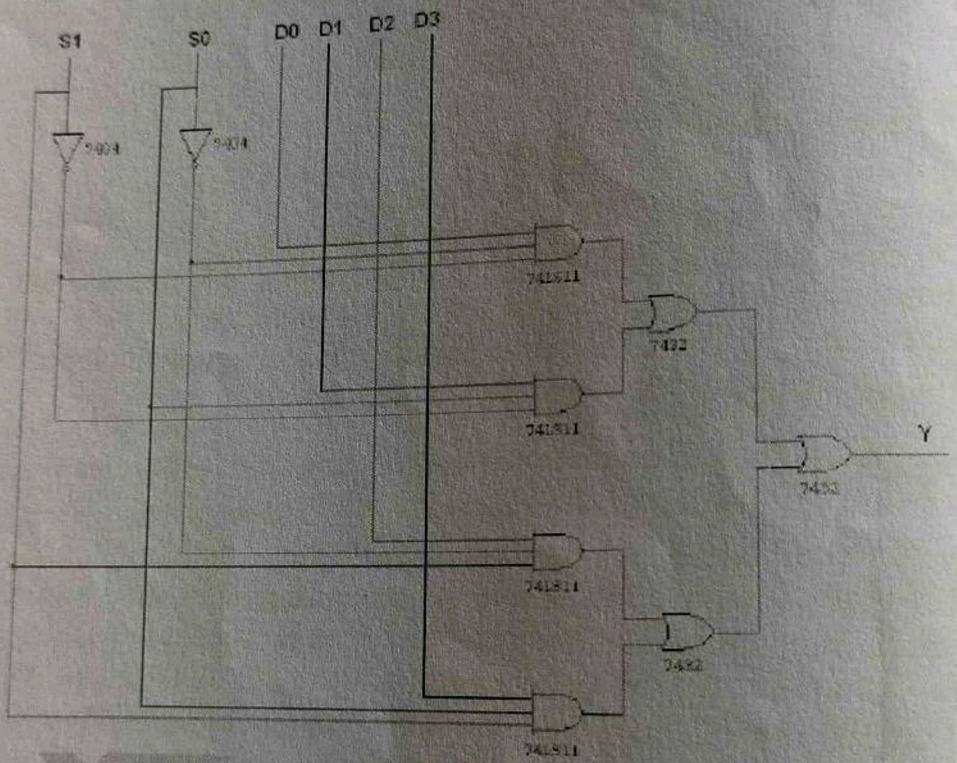


FUNCTION TABLE:

S1	S0	INPUTS Y
0	0	D0 — D0 S1' S0'
0	1	D1 — D1 S1' S0
1	0	D2 — D2 S1 S0'
1	1	D3 — D3 S1 S0

$$Y = D0\ S1'\ S0' + D1\ S1'\ S0 + D2\ S1\ S0' + D3\ S1\ S0$$

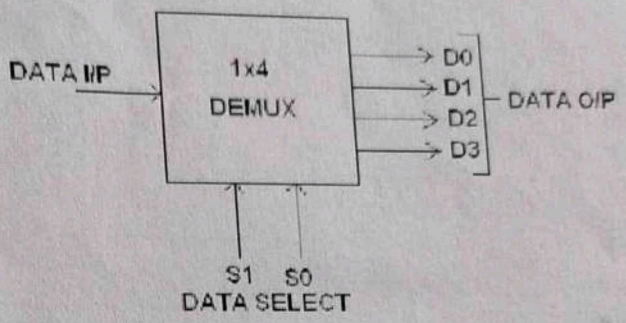
CIRCUIT DIAGRAM FOR MULTIPLEXER:



TRUTH TABLE:

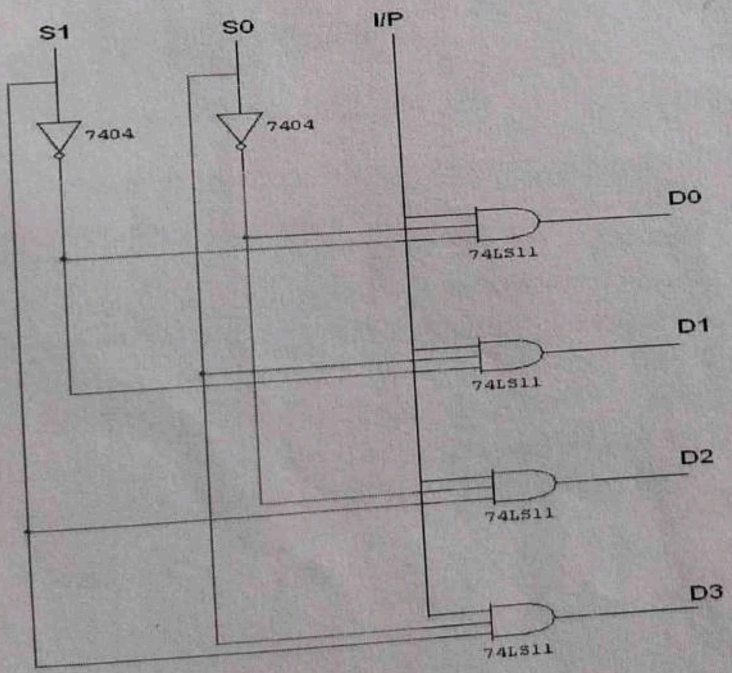
S1	S0	Y = OUTPUT
0	0	D0
0	1	D1
1	0	D2
1	1	D3

BLOCK DIAGRAM FOR 1:4 DEMULTIPLEXER:



S1	S0	INPUT
0	0	X → D0 = X S1' S0'
0	1	X → D1 = X S1' S0
1	0	X → D2 = X S1 S0'
1	1	X → D3 = X S1 S0

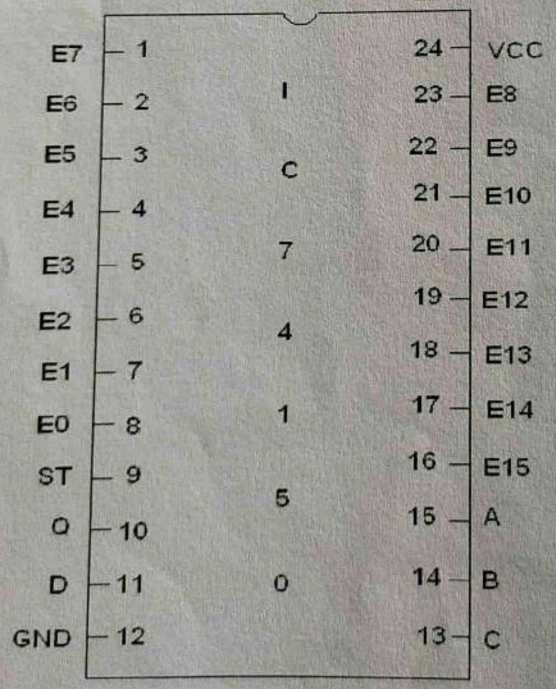
Y = X S1' S0' + X S1' S0 + X S1 S0' + X S1 S0



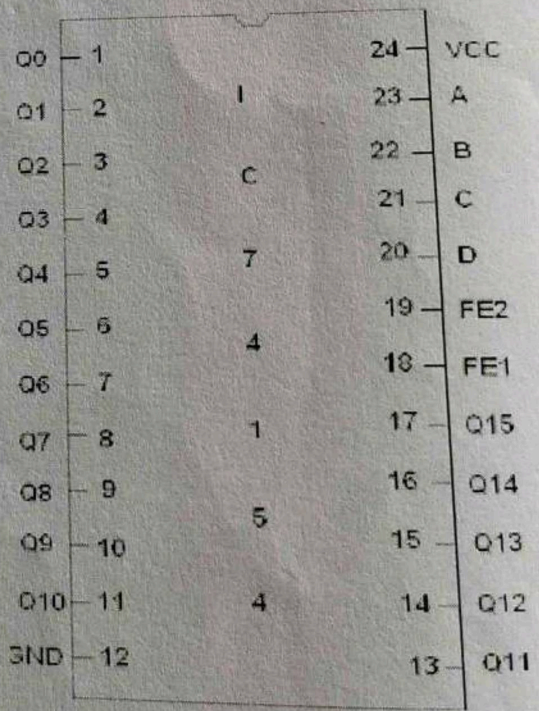
TRUTH TABLE:

INPUT			OUTPUT			
S1	S0	I/P	D0	D1	D2	D3
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	0	0
1	0	1	0	0	1	0
1	1	0	0	0	0	0
1	1	1	0	0	0	1

PIN DIAGRAM FOR IC 74150:



PIN DIAGRAM FOR IC 74154:



PROCEDURE:

(i) Connections are given as per circuit diagram.

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

Exp. - 11

Lab 11: Design of Shift Registers and Shift Register Counters

Data Serial & Parallel,

11.1 Aim

The purpose of this experiment is to introduce the design of Shift Registers. The student should also be able to design n-bit shift register.

11.2 Hardware Requirement

a. Equipments - Digital IC Trainer Kit

b. Discrete Components - IC7474 Dual D Flip-flop / CD 4013

11.3 Theory

Shift registers are a type of sequential logic circuit, mainly for storage of digital data. They are a group of flip-flops connected in a chain so that the output from one flip-flop becomes the input of the next flip-flop. Most of the registers possess no characteristic internal sequence of states. All the flip-flops are driven by a common clock, and all are set or reset simultaneously. The basic types of shift registers are

1. Serial In - Serial Out
2. Serial In - Parallel Out
3. Parallel In - Serial Out
4. Parallel In - Parallel Out
5. Bidirectional shift registers

11.3.1 Serial In - Serial Out Shift Registers

A basic four-bit shift register can be constructed using four D flip-flops, as shown below. The operation of the circuit is as follows. The register is first cleared, forcing all four outputs to zero. The input data is then applied sequentially to the D input of the first flip-flop on the left (FF0). During each clock pulse, one bit is transmitted from left to right. Assume a data word to be 1001. The least significant bit of the data has to be shifted through the register from FF0 to FF3. Assume a data word to be 1001. The least significant bit of the data has to be shifted through the register from FF0 to FF3.

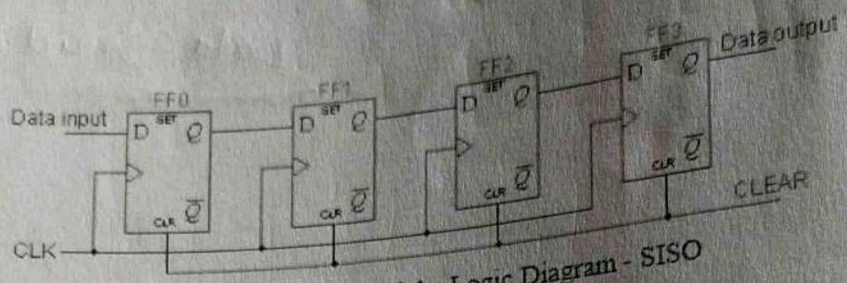


Figure 9.1 Logic Diagram - SISO

11.3.2 Parallel In - Parallel Out Shift Registers

For parallel in - parallel out shift registers, all data bits appear on the parallel outputs immediately following the simultaneous entry of the data bits. The following circuit is a four-bit parallel in - parallel out shift register constructed by D flip-flops.

The D's are the parallel inputs and the Q's are the parallel outputs. Once the register is clocked, all the data at the D inputs appear at the corresponding Q outputs simultaneously.

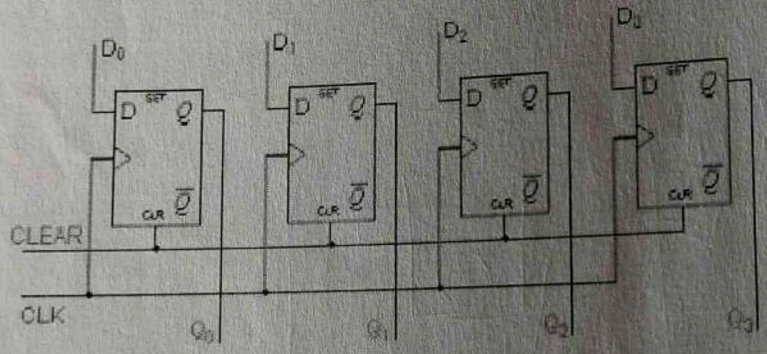


Figure 11.2 Logic Diagram – PIPO

Shift Register Counter

Two of the most common types of shift register counters are Ring counter and the Johnson counter. They are basically shift registers with the serial outputs connected back to the serial inputs in order to produce particular sequences. These registers are classified as counters because they exhibit a specified sequence of states.

Johnson counters are a variation of standard ring counters, with the inverted output of the last stage fed back to the input of the first stage. They are also known as twisted ring counters. An n-stage Johnson counter yields a count sequence of length 2n, so it may be considered to be a mod-2n counter. The circuit above shows a 4-bit Johnson counter. The state sequence for the counter is given in the table as well as the animation on the left.

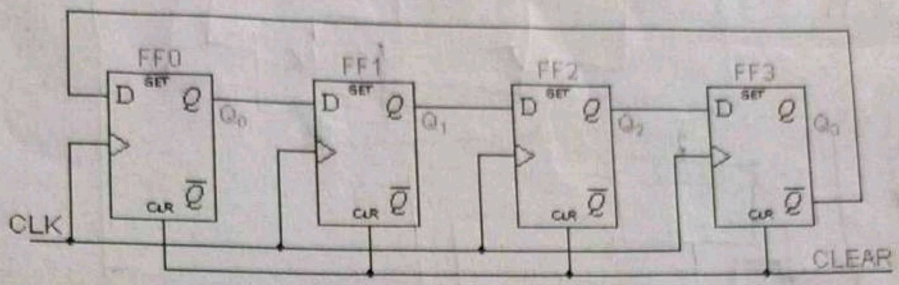


Figure 11.3 Logic Diagram with Truth Table

Clock Pulse	Q3	Q2	Q1	Q0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	1	1	1
4	1	1	1	1
5	1	1	1	0
6	1	1	0	0
7	1	0	0	0

Figure 11.4 Truth Table

11.4 Prelab questions

1. Redraw the above logic diagrams using JK flip-flops.
2. How many flip-flops are required to store the data 1001?
3. How many clock pulses are required to enter a byte of data serially into an 8-bit shift register?
4. Define shift register counters.
5. What is bidirectional and unidirectional shift register?
6. Explain the function of SHIFT/LOAD input in PISO shift register.

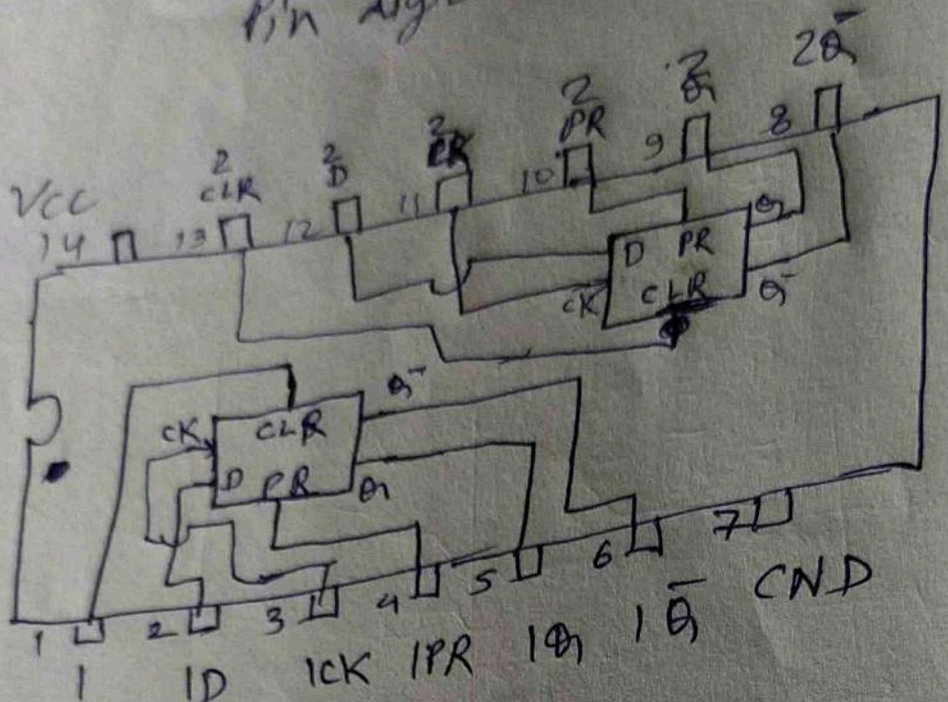
11.5 Result

Thus, design and verification of shift registers are done successfully.

11.6 PostLab questions

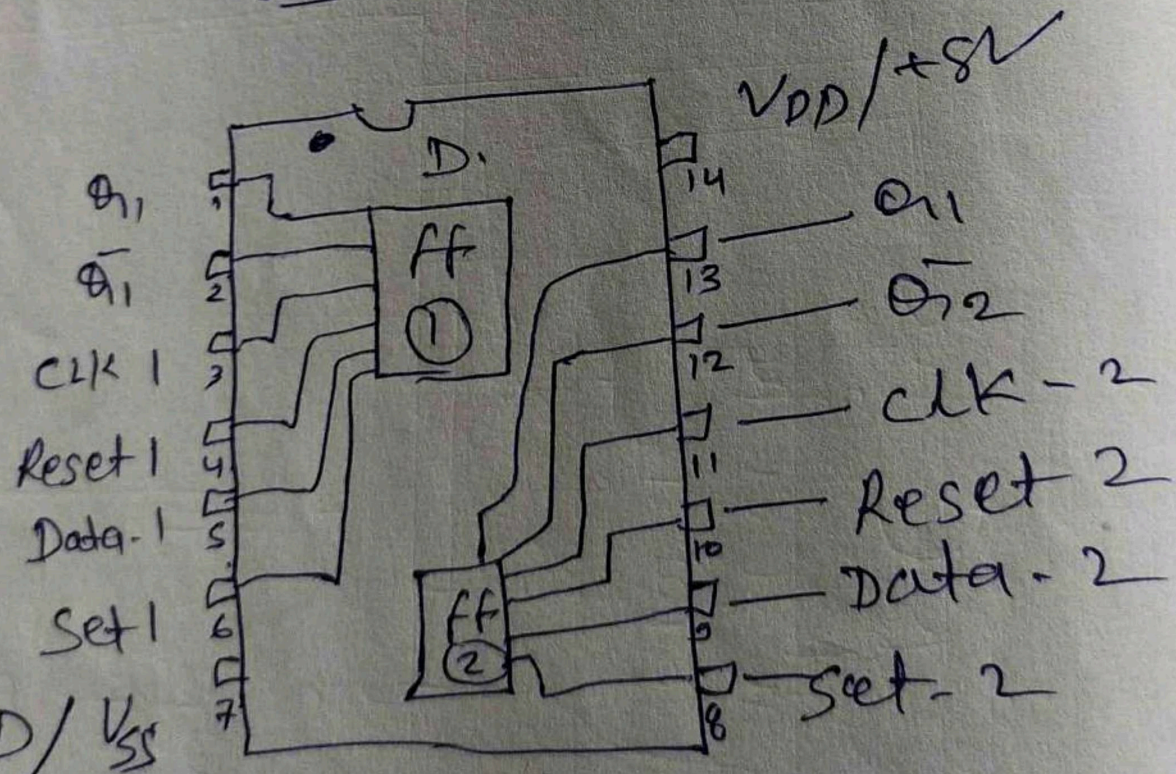
1. Draw the Logic diagram of Universal shift register.
2. Write the applications of Shift registers.
3. Draw the State Table and Block diagram of a Serial Adder.

Pin digram 7474



and

CD 4013 (D flip-flop.)



circuit of four bit Shift Register