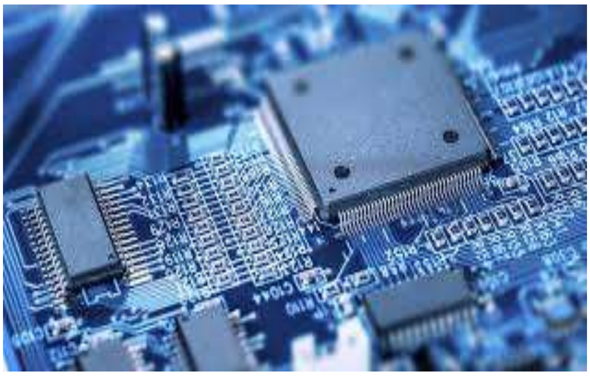
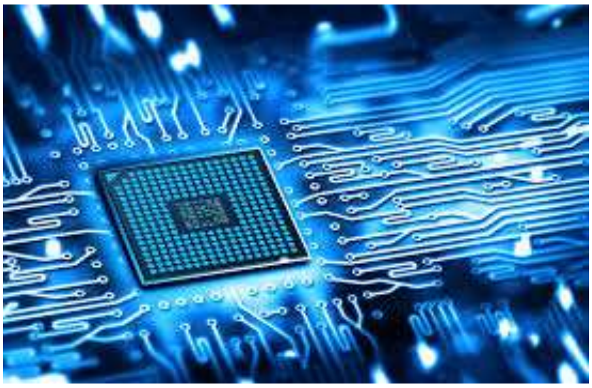
LABORATORY MANUAL

Digital Electronics





Department of Electrical Engineering

JORHAT ENGINEERING COLLEGE

Assam-785007

Course Code	Course Title	Hours per week L-T-P	Credit C
EI181315	**Digital Electronics Lab**	**0-0-2**	**1**

Course outcomes:

At the end of the course, the students will be able to:

CO1:
Verify and analyze the outputs of combinational logic circuits and relate theoretical concepts with experimental analysis.

CO2:
Verify and analyze the outputs of sequential logic circuits and relate theoretical concepts with experimental analysis.

CO3:
Organize and write an engineering report after performing an experiment on digital circuits.

LIST OF EXPERIMENTS

Experiment No.	Title of the Experiment
1	Realization of basic gates by using universal gates
2	Realization of XOR gate
3	Combinational Logic Design using 74xx ICs
4	Arithmetic Circuit- construction and testing using 74xxICs: Half/Full Adder
5	Construction of 1- bit comparator using 74xxICs.
6	code converters – Binary to Gray & Gray to binary.
7	Verification of Truth Table of SR Flip-Flop
8	Verification of Truth Tables of JK, D, T Flip-Flops
9	Decade Counter design

Text Books:

1. Digital Design – M. Marris Mano.

2. Logic Design Theory – NN Biswas

3. Digital Fundamental – TL Floyd

4. Digital Electronics- R. P. Jain.

Experiment 1: Realization of logic gates with the help of universal gates.

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

Theory

Introduction

Logic gates are electronic circuits which perform logical functions on one or more inputs to produce one output. There are seven logic gates. When all the input combinations of a logic gate are written in a series and their corresponding outputs written along them, then this input/ output combination is called Truth Table.

1)Nand gate as Universal gate

NAND gate is actually a combination of two logic gates i.e. AND gate followed by NOT gate. So its output is complement of the output of an AND gate.This gate can have minimum two inputs, output is always one. By using only NAND gates, we can realize all logic functions: AND, OR, NOT, X-OR, X-NOR, NOR. So this gate is also called as universal gate.

1.1)NAND gates as NOT gate

A NOT produces complement of the input. It can have only one input, tie the inputs of a NAND gate together. Now it will work as a NOT gate. Its output is

Y = (A.A)'

Y = (A)'

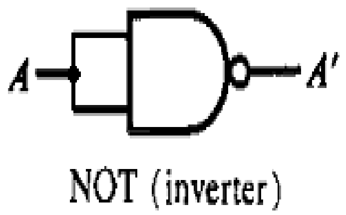


Figure-1: NAND gates as NOT gate

Input	Output
A	A’
0 1	1 0

Figure-2: Truth table of NOT

1.2)NAND gates as AND gate

A NAND produces complement of AND gate. So, if the output of a NAND gate is inverted, overall output will be that of an AND gate.

Y = ((A.B)’)’

Y = (A.B)

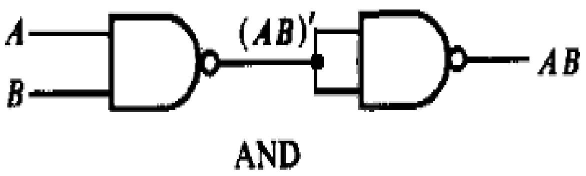


Figure-3:NAND gates as AND gate

Input		Output
A	B	F = A.B
0	0	0
0	1	0
1	0	0
1	1	1

Figure-4: Truth table of AND

1.3)NAND gates as OR gate

From DeMorgan's theorems:

(A.B)' = A' + B'

(A'.B')' = A'' + B'' = A + B

So, give the inverted inputs to a NAND gate, obtain OR operation at output.

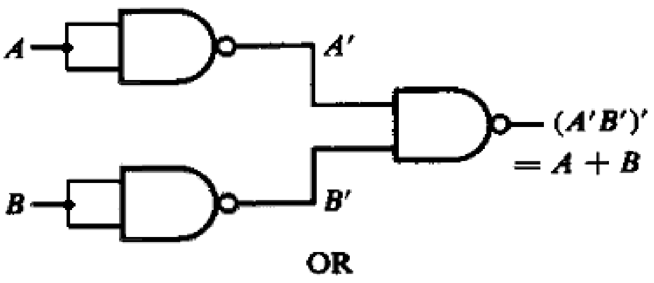


Figure-5: NAND gates as OR gate

A	B	X = A+B
0	0	0
0	1	1
1	0	1
1	1	1

Figure-6: Truth table of OR

1.4) NAND gates as Ex-OR gate

The output of a two input Ex-OR gate is shown by: Y = A'B + AB'. This can be achieved with the logic diagram shown in the left side.

Gate No.	Inputs	Output
1	A, B	(AB)'
2	A, (AB)'	(A (AB)')'
3	(AB)', B	(B (AB)')'
4	(A (AB)')', (B (AB)')'	A'B + AB'

Now the ouput from gate no. 4 is the overall output of the configuration.

$$
\begin{aligned}
Y &= ((A\,(AB)')'\,(B\,(AB)')')' \\
&= (A(AB)')'' + (B(AB)')'' \\
&= (A(AB)') + (B(AB)') \\
&= (A(A' + B)') + (B(A' + B')) \\
&= (AA' + AB') + (BA' + BB') \\
&= (\,0 + AB' + BA' + 0\,) \\
&= AB' + BA'
\end{aligned}
$$

$$\Rightarrow \quad Y = AB' + A'B$$

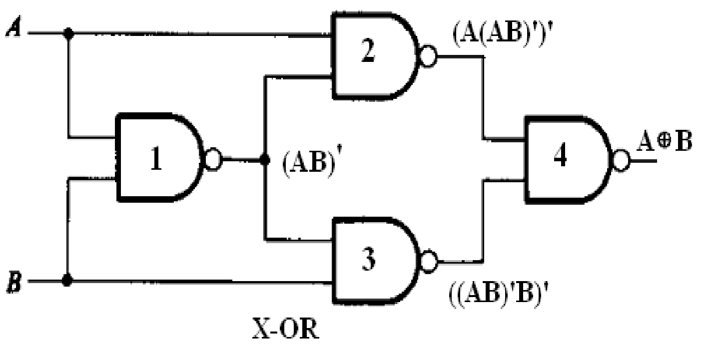


Figure-7: NAND gates as Ex-OR gate

A	B	A **XOR** B
0	0	0
0	1	1
1	0	1
1	1	0

Figure-8: Truth table of Ex-OR

1.5) NAND gates as Ex-NOR gate

Ex-NOR gate is actually Ex-OR gate followed by NOT gate. So give the output of Ex-OR gate to a NOT gate, overall output is that of an Ex-NOR gate.

Y = AB+ A'B'

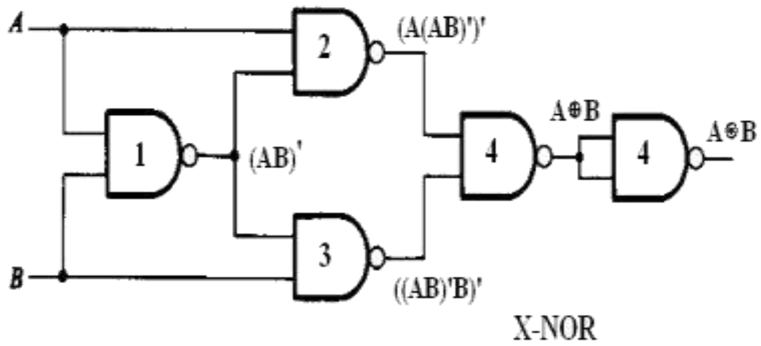


Figure-9: NAND gates as Ex-NOR gate

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

Figure-10: Truth table of Ex-NOR

>1.6) Implementing the simplified function with NAND gates only

We can now start constructing the circuit. First note that the entire expression is inverted and we have three terms ANDed. This means that we must use a 3-input NAND gate. Each of the three terms is, itself, a NAND expression. Finally, negated single terms can be generates with a 2-input NAND gate acting as an inverted. Figure 8 illustrates a circuit using NAND gates only.

F=((C'.B.A)'(D'.C.A)'(C.B'.A)')'

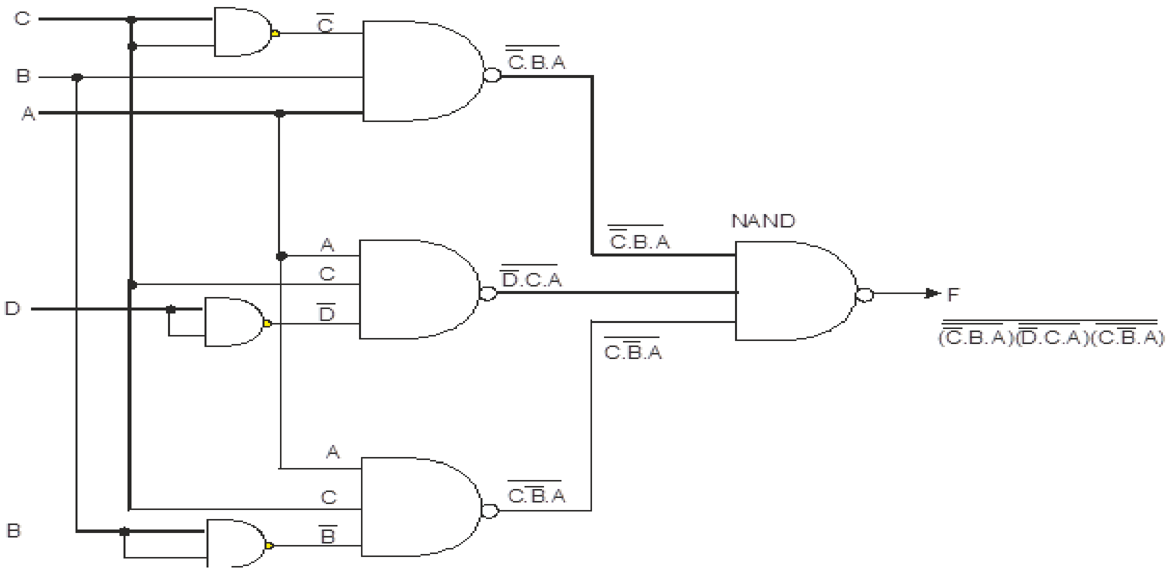


Figure-11: Implementing the simplified function with NAND gates only

2) NOR gate as Universal Gate

NOR gate is actually a combination of two logic gates: OR gate followed by NOT gate. So its output is complement of the output of an OR gate. This gate can have minimum two inputs, output is always one. By using only NOR gates, we can realize all logic functions: AND, OR, NOT, Ex-OR, Ex-NOR, NAND. So this gate is also called universal gate.

2.1)NOR gates as NOT gate

A NOT produces complement of the input. It can have only one input, tie the inputs of a NOR gate together. Now it will work as a NOT gate. Its output is

Y = (A+A)’

Y = (A)’

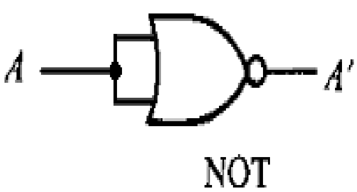


Figure-12: NOR gates as NOT gate

Input	Output
A	A’
0 1	1 0

Figure-13: Truth table of NOT

2.2)NOR gates as OR gate

A NOR produces complement of OR gate. So, if the output of a NOR gate is inverted, overall output will be that of an OR gate.

Y = ((A+B)’)’

Y = (A+B)

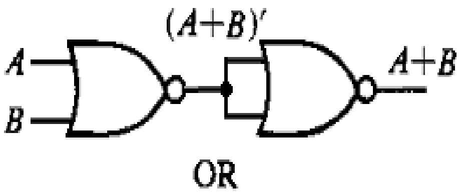


Figure-14: NOR gates as OR gate

A	B	X = A+B
0	0	0
0	1	1
1	0	1
1	1	1

Figure-15: Truth table of OR

2.3) NOR gates as AND gate

From DeMorgan's theorems:
(A+B)' = A'B'
(A'+B')' = A''B'' = AB
So, give the inverted inputs to a NOR gate, obtain AND operation at output.

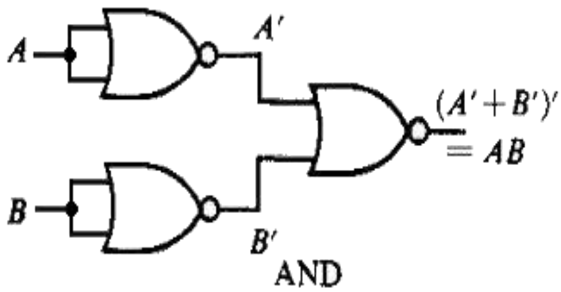


Figure-16:NOR gates as AND gate

Input		Output
A	B	F = A.B
0	0	0
0	1	0
1	0	0
1	1	1

Figure-17: Truth table of AND

2.4)NOR gates as Ex-NOR gate

The output of a two input Ex-NOR gate is shown by: Y = AB + A'B'. This can be achieved with the logic diagram shown in the left side.

Gate No.	Inputs	Output
1	A, B	(A + B)’
2	A, (A + B)’	(A + (A+B)’)’
3	(A + B)’, B	(B + (A+B)’)’
4	(A + (A + B)’)’, (B + (A+B)’)’	AB + A’B’

Now the ouput from gate no. 4is the overall output of the configuration.

Y = ((A + (A+B)’)’ (B +(A+B)’)’)’
= (A+(A+B)’)’’.(B+(A+B)’)’’
= (A+(A+B)’).(B+(A+B)’)
= (A+A’B’).(B+A’B’)
= (A + A’).(A + B’).(B+A’)(B+B’)
= 1.(A+B’).(B+A’).1
= (A+B’).(B+A’)
= A.(B + A’) +B’.(B+A’)
= AB + AA’ +B’B+B’A’
= AB + 0 + 0 + B’A’
= AB + B’A’
=> Y = AB + A’B’

Figure-18: NOR gates as Ex-NOR gate

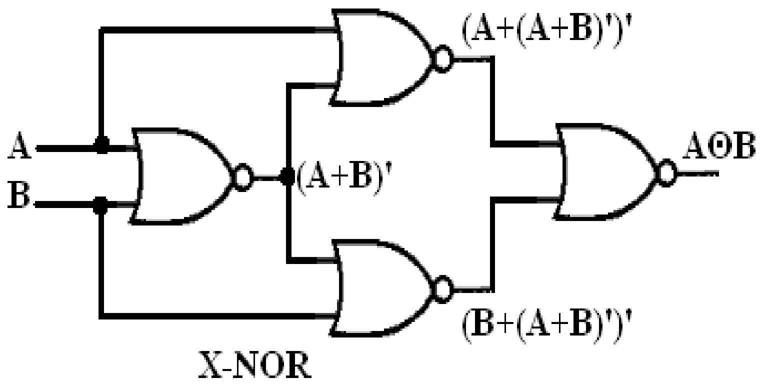


Figure-19: Truth table of Ex-NOR

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

2.5) NOR gates as Ex-OR gate

Ex-OR gate is actually Ex-NOR gate followed by NOT gate. So give the output of Ex-NOR gate to a NOT gate, overall output is that of an Ex-OR gate.
Y = A'B+ AB'

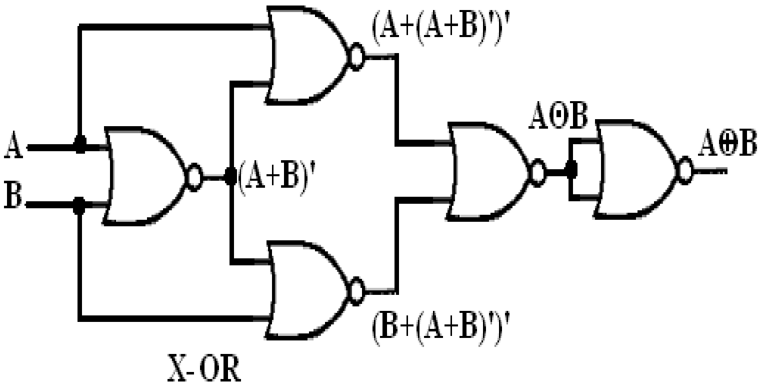


Figure-20: NOR gates as Ex-OR gate

A	B	A **XOR** B
0	0	0
0	1	1
1	0	1
1	1	0

Figure-21: Truth table of Ex-OR

2.6) Constructing a circuit with NOR gates only

Designing a circuit with NOR gates only uses the same basic techniques as designing a circuit with NAND gates; that is, the application of deMorgan's theorem. The only difference between NOR gate design and NAND gate design is that the former must eliminate product terms and the later must eliminate sum terms.
F=(((C.B'.A)+(D.C'.A)+(C.B'.A))')'

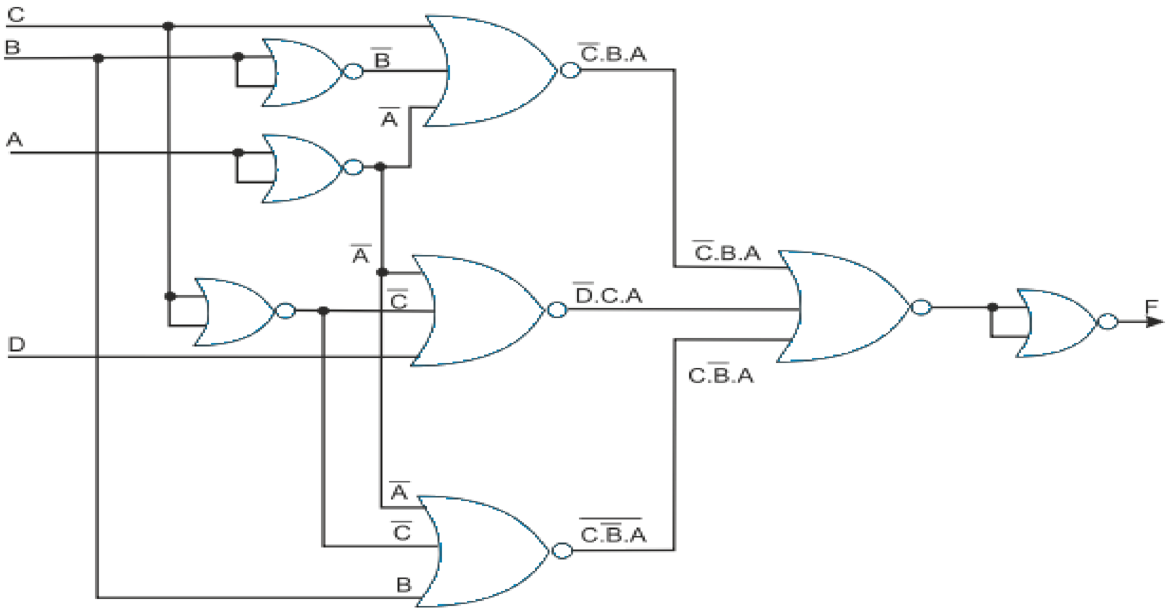


Figure-22: Implementing the simplified function with NOR gates only

<u>Procedure</u>

<u>Results and Discussion:</u>

<u>EXPERIMENT 2:</u> Realization of XOR gate

<u>Aim:</u> To verify and interpret the logic and truth table for Ex-OR gates using RTL (Resistor Transistor Logic), DTL (Diode Transistor Logic) and TTL (Transistor Transistor Logic) logics in simulator 1 and verify the truth table for Ex-OR gates in simulator 2.

Theory

Ex-OR gate

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

Y= A⊕B

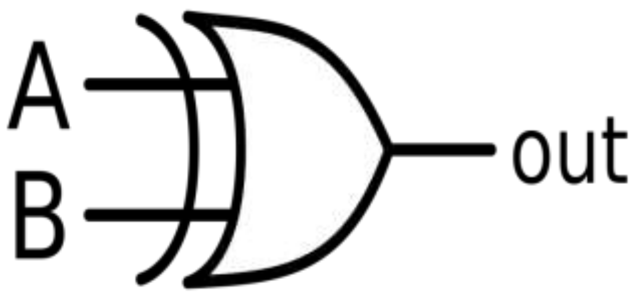


Figure-16: Logic Symbol of Ex-OR gate

A	B	A **XOR** B
0	0	0
0	1	1
1	0	1
1	1	0

Figure-17: Truth Table of Ex-OR gate

Ex-OR gate is created from AND, NAND and OR gates. The output is high only when both the inputs are different.

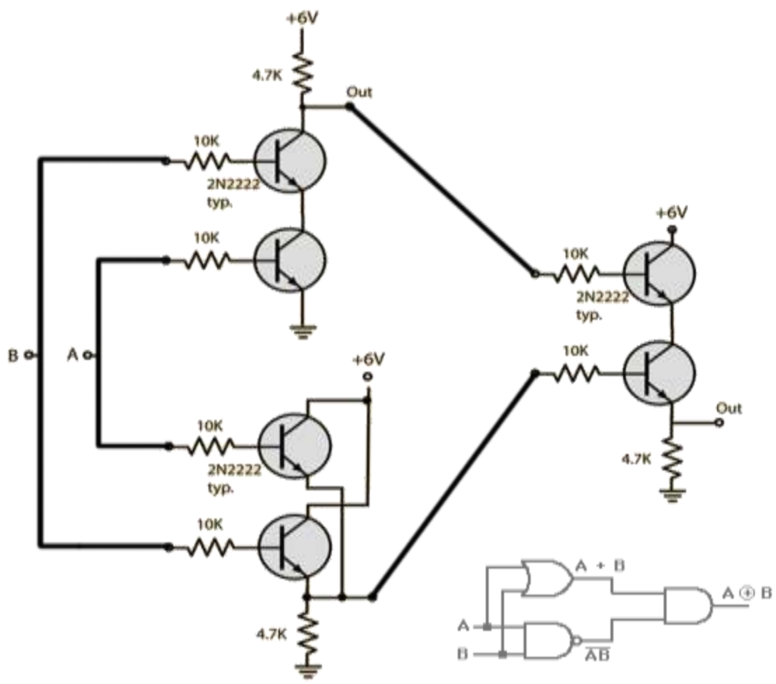


Figure-18: Ex-OR gate through RTL Logic.

Procedure

Results and Discussion:

EXPERIMENT 3: **Combinational Logic Design using 74xx ICs**

Aim: **To analyse the truth table of 4 * 2 decoder/de-multiplexer using NOT (7404) and AND (7408) logic gate ICs and 2 * 4 encoder using OR (7403) logic gate IC and to understand the working of 4 * 2 decoder and 2 * 4 encoder circuit with the help of LEDs display.**

Theory

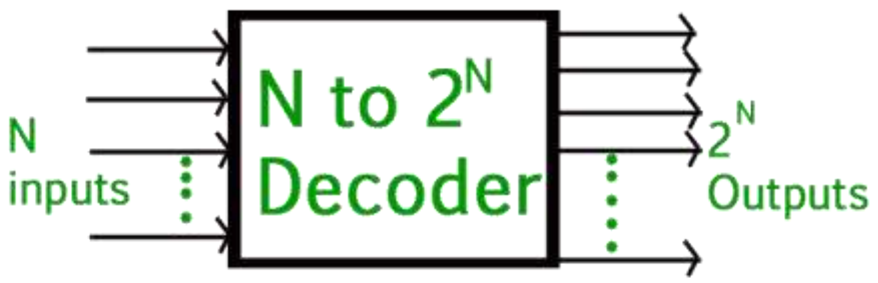
Introduction

Binary code of N digits can be used to store 2^N distinct elements of coded information. This is what encoders and decoders are used for. Encoders convert 2^N lines of input into a code of N bits and Decoders decode the N bits into 2^N lines.

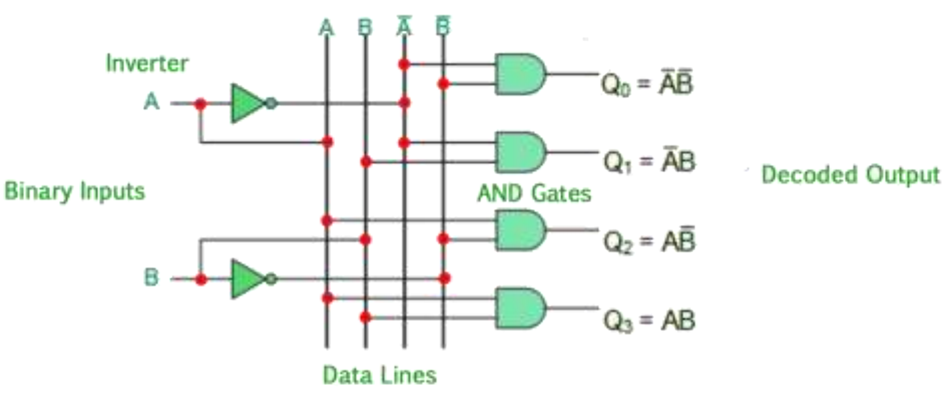
1) 2x4 Decoder / De-multiplexer

The name "Decoder" means to translate or decode coded information from one format into another, so a digital decoder transforms a set of digital input signals into an equivalent decimal code at its output

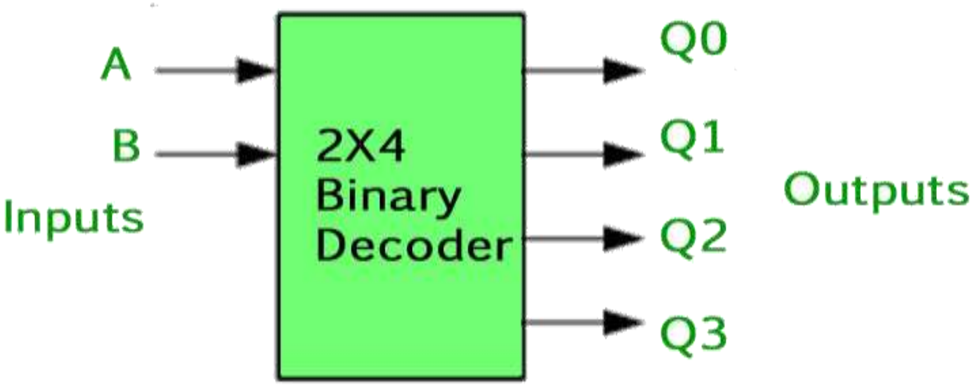
A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines.



1.1) 2-to-4 Binary Decoder



The 2-to-4 line binary decoder depicted above consists of an array of four AND gates. The 2 binary inputs labelled A and B are decoded into one of 4 outputs, hence the description of 2-to-4 binary decoder. Each output represents one of the minterms of the 2 input variables, (each output = a minterm).



A	B	Q0	Q1	Q2	Q3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Fig: Logic Diagram and TRUTH TABLE of DECODER

The binary inputs A and B determine which output line from Q0 to Q3 is “HIGH” at logic level “1” while the remaining outputs are held “LOW” at logic “0” so only one output can be active (HIGH) at any one time.

Therefore, whichever output line is “HIGH” identifies the binary code present at the input, in other words it “decodes” the binary input.Some binary decoders have an additional input pin labelled “Enable” that controls the outputs from the device.

This extra input allows the decoders outputs to be turned “ON” or “OFF” as required. Output is only generated when the Enable input has value 1; otherwise, all outputs are 0. Only a small change in the implementation is required: the Enable input is fed into the AND gates which produce the outputs.

If Enable is 0, all AND gates are supplied with one of the inputs as 0 and hence no output is produced. When Enable is 1, the AND gates get one of the inputs as 1, and now the output depends upon the remaining inputs. Hence the output of the decoder is dependent on whether the Enable is high or low.

2) Encoder

An Encoder is a combinational circuit that performs the reverse operation of Decoder.It has maximum of 2^n input lines and 'n' output lines, hence it encodes the information from 2^n inputs into an n-bit code. It will produce a binary code equivalent to the input, which is active High. Therefore, the encoder encodes 2^n input lines with 'n' bits.

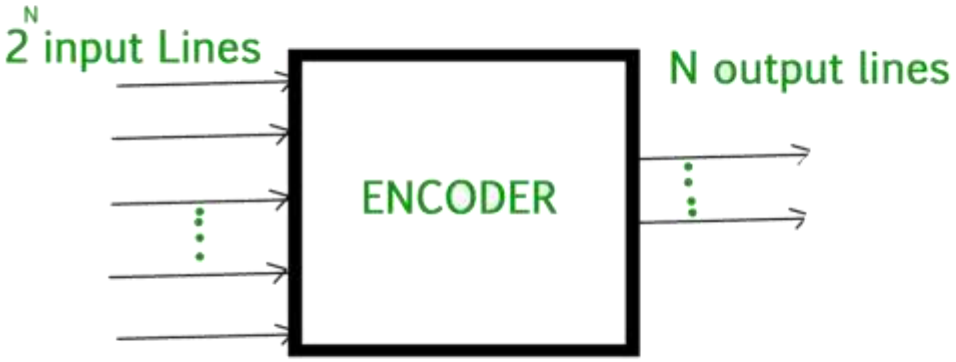


Fig: Logic Diagram of ENCODER

2.1)4 : 2 Encoder

The 4 to 2 Encoder consists of four inputs Y3, Y2, Y1 & Y0 and two outputs A1 & A0. At any time, only one of these 4 inputs can be '1' in order to get the respective binary code at the output. The figure below shows the logic symbol of 4 to 2 encoder :

Fig: Logic Diagram of 4 : 2 Encoder
The Truth table of 4 to 2 encoder is as follows :

INPUTS				OUTPUTS	
Y3	Y2	Y1	Y0	A1	A0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Fig: TRUTH TABLE of 4 : 2 Encoder

Procedure

<u>Results and Discussion:</u>

<u>EXPERIMENT 4: </u>Arithmetic Circuit- construction and testing using 74xxICs: Half/Full Adder

<u>Aim:</u>To verify the truth table of half adder and full adder by using XOR and NAND gates respectively and analyse the working of half adder and full adder circuit with the help of LEDs in simulator 1 and verify the truth table only of half adder and full adder in simulator 2.

<u>Theory</u>

Introduction

Adders are digital circuits that carry out addition of numbers. Adders are a key component of arithmetic logic unit. Adders can be constructed for most of the numerical representations like Binary Coded Decimal (BDC), Excess – 3, Gray code, Binary etc. out of these, binary addition is the most frequently performed task by most common adders. Apart from addition, adders are also used in certain digital applications like table index calculation, address decoding etc.

Binary addition is similar to that of decimal addition. Some basic binary additions are shown below.

0	0	1	1
<u>+0</u>	<u>+1</u>	<u>+0</u>	<u>+1</u>
0	1	1	(carry) 1 0

Schematic representation of half adder

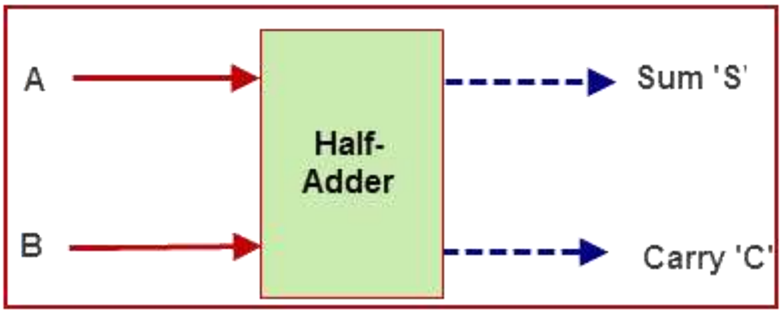
1) Half Adder

Half adder is a combinational circuit that performs simple addition of two binary numbers. The block diagram of a half adder is shown below.

1.1) Half Adder Truth Table

If we assume A and B as the two bits whose addition is to be performed, a truth table for half adder with A, B as inputs and Sum, Carry as outputs can be tabulated as follows.

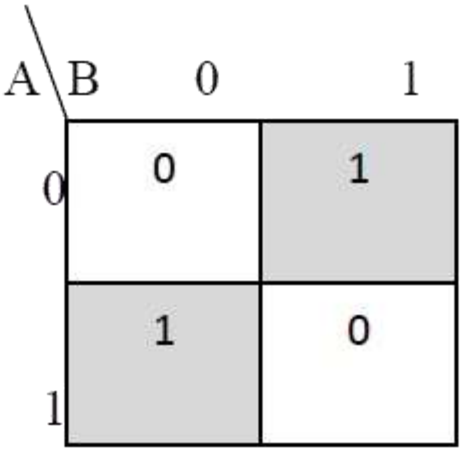
The sum output of the binary addition carried out above is similar to that of an Ex-OR operation while the carry output is similar to that of an AND operation. The same can be verified with help of Karnaugh Map.

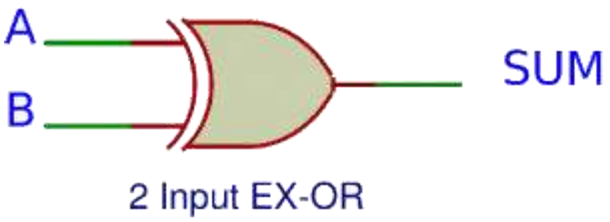


Truth Table			
Input		Output	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

The truth table and K Map simplification for sum output is shown below.

Truth Table		
A	B	SUM
0	0	0
0	1	1
1	0	1
1	1	0

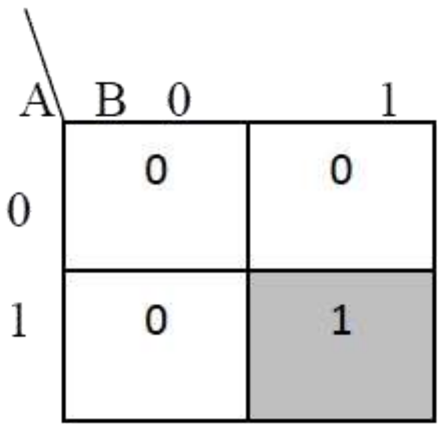


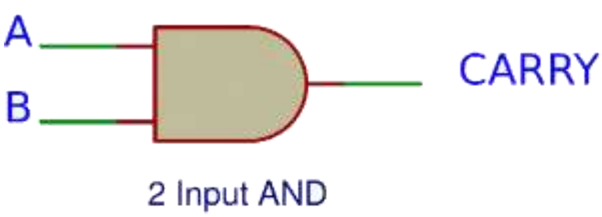


Sum = A B' + A' B

The truth table and K Map simplification for carry is shown below.

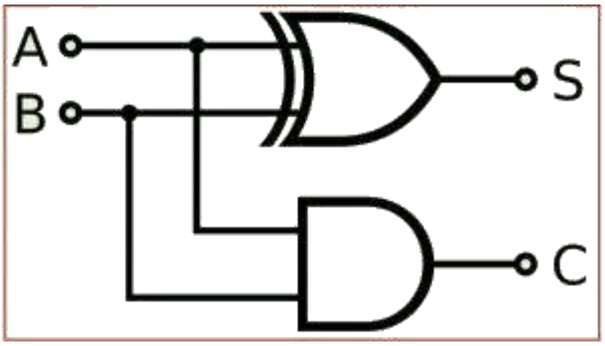
Truth Table		
A	B	Carry
0	0	0
0	1	0
1	0	0
1	1	1





Carry = AB

If A and B are binary inputs to the half adder, then the logic function to calculate sum S is Ex – OR of A and B and logic function to calculate carry C is AND of A and B. Combining these two, the logical circuit to implement the combinational circuit of half adder is shown below.

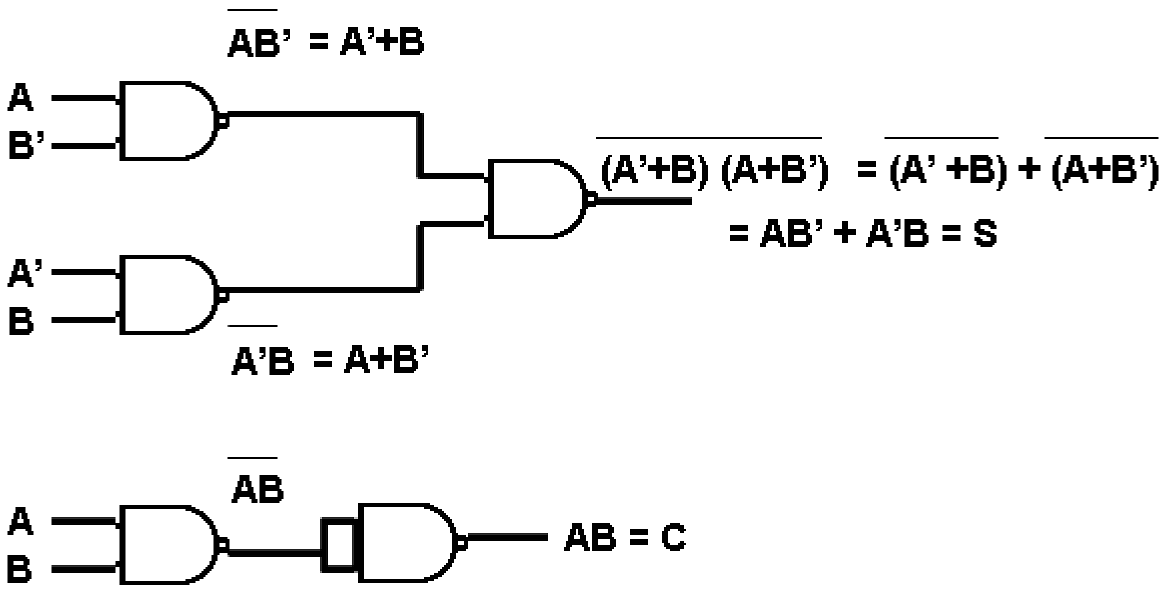


Half Adder Logic Diagram

As we know that NAND and NOR are called universal gates as any logic system can be implemented using these two, the half adder circuit can also be implemented using them. We know that a half adder circuit has one Ex – OR gate and one AND gate.

1.2) Half Adder using NAND gates

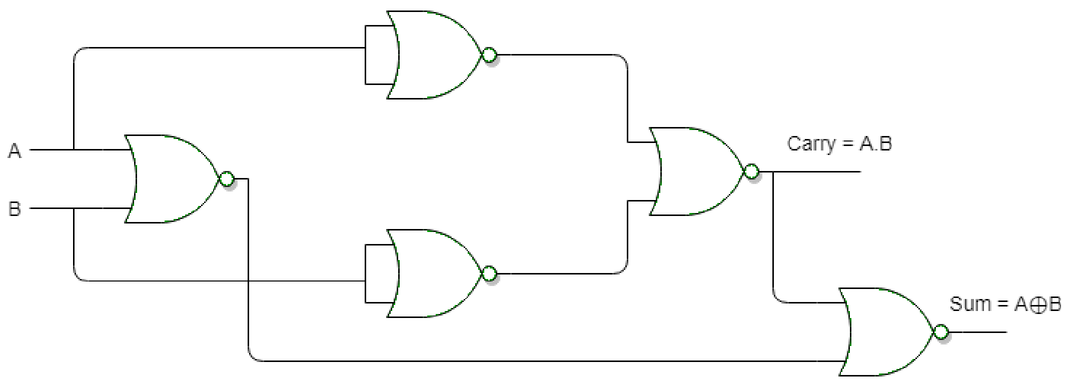
Five NAND gates are required in order to design a half adder. The circuit to realize half adder using NAND gates is shown below.



Realization of half adder using NAND gates

1.3) Half Adder using NOR gates

Five NOR gates are required in order to design a half adder. The circuit to realize half adder using NOR gates is shown below.



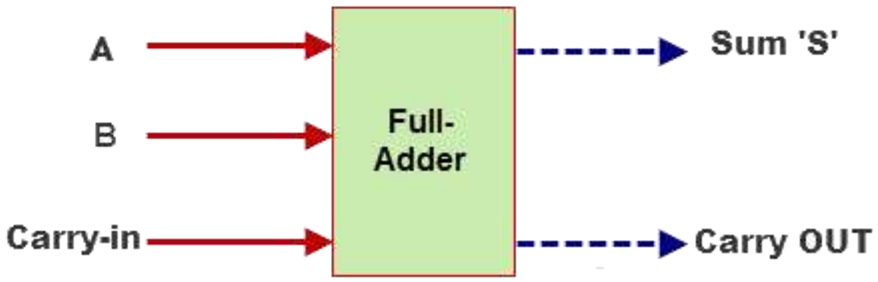
Realization of half adder using NOR Gates

2) Full Adder

Full adder is a digital circuit used to calculate the sum of three binary bits which is the main difference between full adder and half adder. Full adders are complex and difficult to implement when compared to half adders. Two of the three bits are same as before which are A, the augend bit and B, the addend bit. The additional third bit is carry bit from the previous stage and is called

'Carry' – in generally represented by CIN. It calculates the sum of three bits along with the carry. The output carry is called Carry – out and is represented by COUT.

The block diagram of a full adder with A, B and CIN as inputs and S, COUT as outputs is shown below.



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

Full Adder Block Diagram and Truth Table

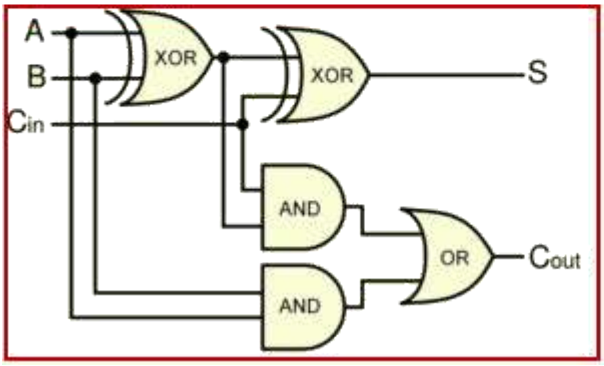
Based on the truth table, the Boolean functions for Sum (S) and Carry – out (COUT) can be derived using K – Map.

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

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

The simplified equation for sum is S = A'B'Cin + A'BCin' + ABCin
The simplified equation for COUT is COUT = AB + ACIN + BCIN

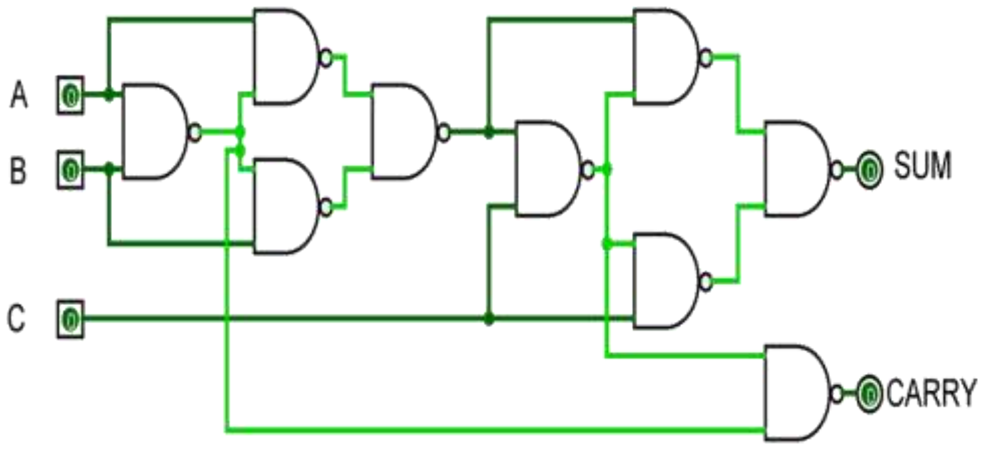
In order to implement a combinational circuit for full adder, it is clear from the equations derived above, that we need four 3-input AND gates and one 4-input OR gates for Sum and three 2-input AND gates and one 3-input OR gate for Carry – out.



Full Adder Logic Diagram

2.1) Full Adder using NAND gates

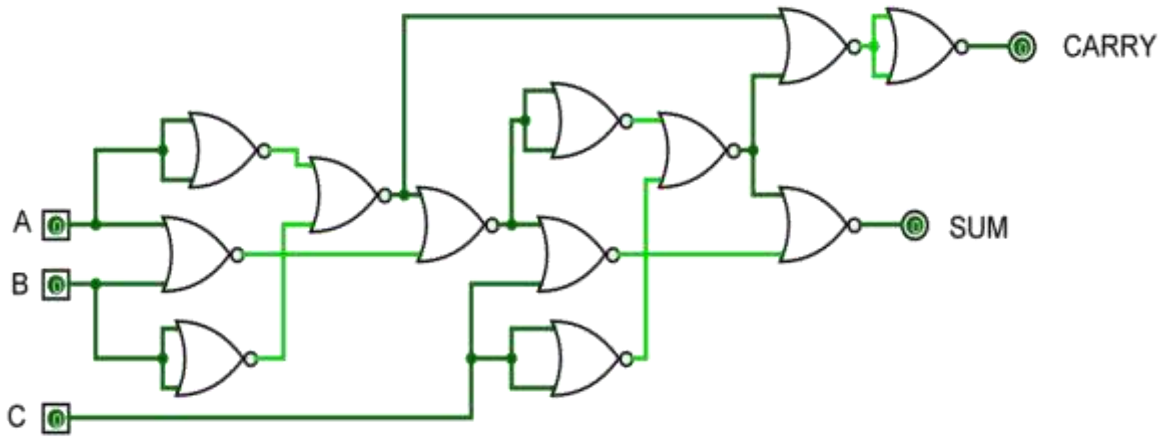
As mentioned earlier, a NAND gate is one of the universal gates and can be used to implement any logic design. The circuit of full adder using only NAND gates is shown below.



Full Adder using NAND gates

2.2)Full Adder using NOR gates

As mentioned earlier, a NOR gate is one of the universal gates and can be used to implement any logic design. The circuit of full adder using only NOR gates is shown below.



Full Adder using NOR gates

Procedure

Results and Discussion:

<u>EXPERIMENT 5:</u> Construction of 1- bit comparator using 74xxICs.

<u>Aim:</u> To analyse the truth table of 1-bit comparator by using NOT, AND and NOR logic gate ICs and to understand the working of 1-bit comparator with the help of LEDs display.

<u>Theory</u>

Introduction

A magnitude digital comparator is a combinational circuit that compares two digital or binary numbers in order to find out whether one binary number is equal, less than or greater than the other binary number. We logically design a circuit for which we will have two inputs one for A and other for B and have three output terminals, one for A > B condition, one for A = B condition and one for A < B condition.

Figure-1: Block Diagram of Comparator

1) 1-Bit Magnitude Comparator :

A comparator used to compare two bits is called a single bit comparator. It consists of two inputs each for two single bit numbers and three outputs to generate less than, equal to and greater than between two binary numbers. The truth table for a 1-bit comparator is given below :

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

Figure-2: Truth Table of 1-Bit Comparator

From the above truth table logical expressions for each output can be expressed as follows:

A > B : AB'

A < B : A'B

A = B : A'B' + AB

By using these Boolean expressions, we can implement a logic circuit for this comparator as given below :

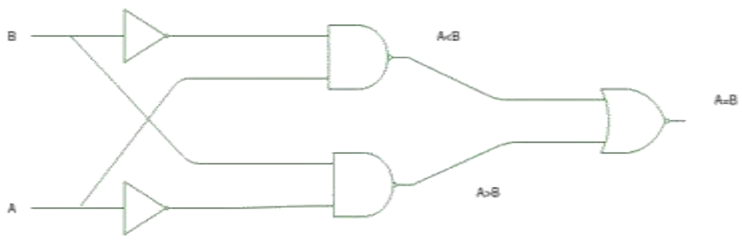


Figure-3: Logic Circuit of 1-Bit Comparator

Applications of Comparators :

1. Comparators are used in central processing units (CPUs) and microcontrollers (MCUs).
2. These are used in control applications in which the binary numbers representing physical variables such as temperature, position, etc. are compared with a reference value.
3. Comparators are also used as process controllers and for Servo motor control.
4. Used in password verification and biometric applications.

Procedure

Results and Discussion:

EXPERIMENT 6: code converters – Binary to Gray & Gray to binary

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

Theory

Introduction

Binary Numbers is default way to store numbers, but in many applications binary numbers are difficult to use and a variation of binary numbers is needed. This is where Gray codes are very useful.

Gray code has property that two successive numbers differ in only one bit because of this property gray code does the cycling through various states with minimal effort and used in K-maps, error correction, communication etc.

In computer science many a times we need to convert binary code to gray code and vice versa. This conversion can be done by applying following rules :

1) Binary to Gray conversion :

1. The Most Significant Bit (MSB) of the gray code is always equal to the MSB of the given binary code.
2. Other bits of the output gray code can be obtained by Ex-ORing binary code bit at that index and previous index.
   There are four inputs and four outputs. The input variable are defined as B_3, B_2, B_1, B_0 and the output variables are defined as G_3, G_2, G_1, G_0. From the truth table, combinational circuit is designed.The logical expressions are defined as :

$\mathbf{B_3 = G_3}$

$\mathbf{B_2 \oplus B_3 = G_2}$

$\mathbf{B_1 \oplus B_2 = G_1}$

$\mathbf{B_0 \oplus B_1 = G_0}$

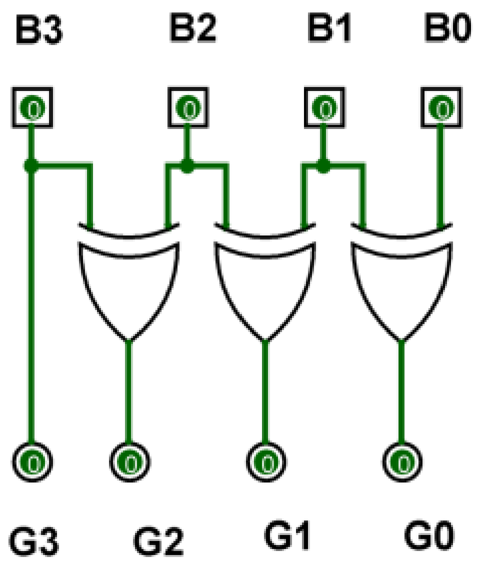


Figure-1: Binary to Gray Code Converter Circuit

Natural-binary code				Gray code			
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

Figure-2: Binary to Gray Code Converter Truth Table

2) Gray to binary conversion :

1.The Most Significant Bit (MSB) of the binary code is always equal to the MSB of the given binary number.

2.Other bits of the output binary code can be obtained by checking gray code bit at that index. If current gray code bit is 0, then copy previous binary code bit, else copy invert of previous binary code bit.

There are four inputs and four outputs. The input variable are defined as G_3, G_2, G_1, G_0 and the output variables are defined as B_3, B_2, B_1, B_0. From the truth table, combinational circuit is designed.The logical expressions are defined as :

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

$G_1 \oplus G_2 \oplus G_3 = B_1$

$G_2 \oplus G_3 = B_2$

$G_3 = B_3$

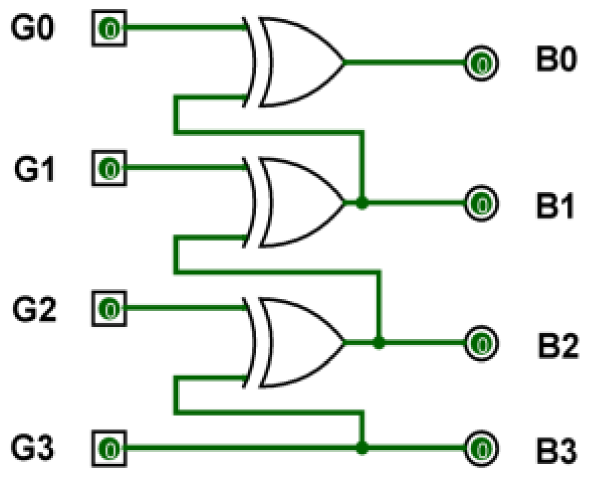


Figure-3: Gray to Binary Code Converter Circuit

Gray code				Natural-binary code			
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	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
1	0	0	0	1	1	1	1
1	0	0	1	1	1	1	0
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	0	1	0	1	1
1	1	1	1	1	0	1	0

Figure-4: Gray to Binary Code Converter Truth Table

Procedure

Results and Discussion:

EXPERIMENT 7: Verification of Truth Table of SR Flip-Flop

Aim: To verify the truth table and timing diagram of SR, JK flip-flop by using NAND & NOR gates ICs and analyse the circuit of SR flip-flop with the help of LEDs display.

Theory

Introduction

A flip flop is an electronic circuit with two stable states that can be used to store binary data. The stored data can be changed by applying varying inputs. Flip-flops and latches are fundamental building blocks of digital electronics systems used in computers, communications, and many other types of systems.

1) R-S flip flop
2) D flip flop
3) J-K flip flop
4) T flip flop

The basic NAND gate RS flip flop circuit is used to store the data and thus provides feedback from both of its outputs again back to its inputs. The RS flip flop actually has three inputs, SET, RESET and its current output Q relating to its current state as shown in figure below.

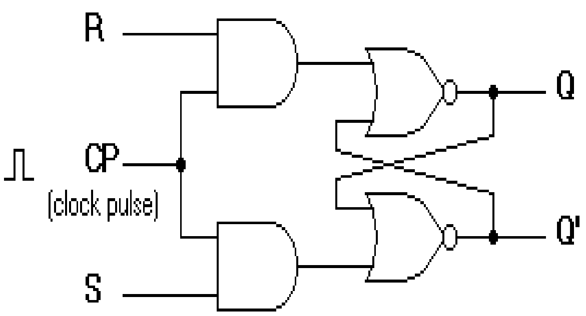


Figure-1:S-R flip flop circuit diagram

INPUTS			OUTPUT	STATE
CLK	S	R	Q	
X	0	0	No Change	Previous
↑	0	1	0	Reset
↑	1	0	1	Set
↑	1	1	-	Forbidden

Figure-2:Characteristics table of S-R flip flop

Procedure

Results and Discussion:

EXPERIMENT 8: JK, T and D flip-flops by using NAND & NOR gates ICs and analyse the circuit of RS, JK, T and D flip-flops with the help of LEDs display.

Theory

Introduction

A flip flop is an electronic circuit with two stable states that can be used to store binary data. The stored data can be changed by applying varying inputs. Flip-flops and latches are fundamental building blocks of digital electronics systems used in computers, communications, and many other types of systems.

1) R-S flip flop
2) D flip flop
3) J-K flip flop
4) T flip flop

1) D flip flop

A D flip flop has a single data input. This type of flip flop is obtained from the SR flip flop by connecting the R input through an inverter, and the S input is connected directly to data input. The modified clocked SR flip-flop is known as D-flip-flop and is shown below. From the truth table of SR flip-flop we see that the output of the SR flip-flop is in unpredictable state when the inputs are same and high. In many practical applications, these input conditions are not required. These input conditions can be avoided by making them complement of each other.

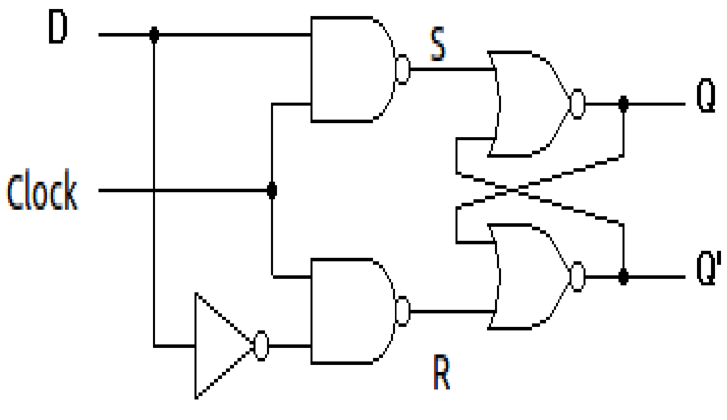


Figure-3: Circuit diagram of D flip flop

Input			Output	
D	reset	clock	Q	Q’
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	1	0
1	1	0	0	1
1	1	1	0	1

Figure-4: Characteristics table of D flip flop

2) J-K flip flop

In a RS flip-flop the input R=S=1 leads to an indeterminate output. The RS flip-flop circuit may be re-joined if both inputs are 1 than also the outputs are complement of each other as shown in characteristics table below.

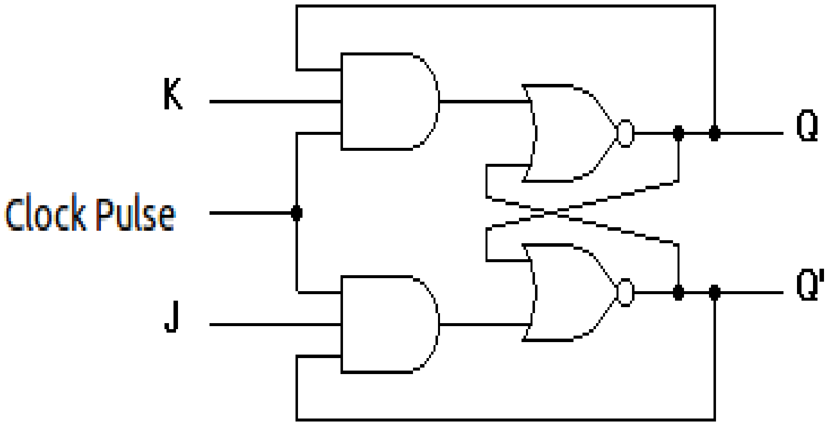


Figure-5: Circuit diagram of J-K flip flop

Trigger	Inputs		Output				Inference
			Present State		Next State		
CLK	J	K	Q	Q'	Q	Q'	
	x	x	-		-		Latched
	0	0	0	1	0	1	No Change
			1	0	1	0	
	0	1	0	1	0	1	Reset
			1	0	0	1	
	1	0	0	1	1	0	Set
			1	0	1	0	
	1	1	0	1	1	0	Toggles
			1	0	0	1	

Figure-6:Characteristics table of J-K flip flop

3) T flip flop

T flip-flop is known as toggle flip-flop. The T flip-flop is modification of the J-K flip-flop. Both the JK inputs of the JK flip – flop are held at logic 1 and the clock signal continuous to change as shown in table below.

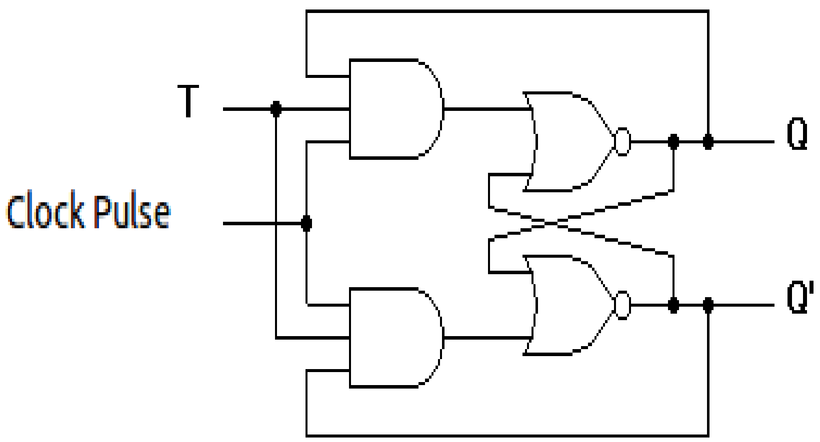


Figure-7: Circuit diagram of T flip flop

T flip-flop

T	Clock	Q	Q'
0	↑	Q	Q'
1	↑	Q'	Q
X	↓	Q	Q'

Figure-8: Characteristics table of T flip flop

Procedure

Results and Discussion:

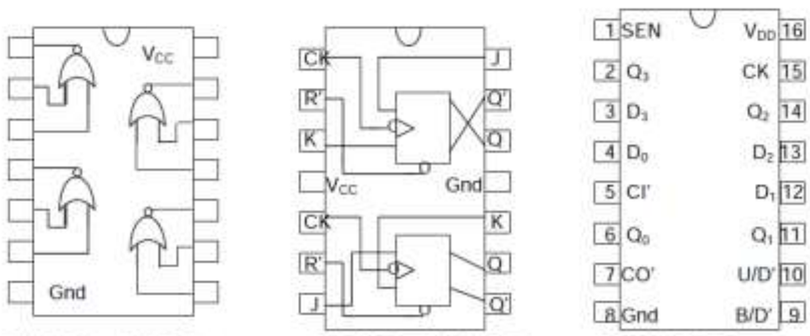
EXPERIMENT 9: **Decade counter design**

Aim: To verify the truth table and timing diagram of Decade counter and analyse the circuit of Decade counter .

Theory

Usually, counter circuits are digital in nature, and count in natural binary. Many types of counter circuits are available as digital building blocks, for example a number of chips in the 4000 series implement different counters. Occasionally there are advantages to using a counting sequence other than the natural binary sequence such as the binary coded decimal counter, a linear feedback shift register counter, or a Gray-code counter. Counters are useful for digital clocks and timers, and in oven timers, VCR clocks, etc.
The counters will be assembled using two 74LS73 dual J-K flip-flop chips and a 74LS02 quad NOR chip. Note that each flip-flop has an asynchronous Reset (R’) input besides the synchronous J-K inputs. This enables one to reset any of the flip-flops by making R’ = 0 irrespective of the status of the clock (CK) input. The asynchronous R’ input will be utilised in this experiment to initialise the flip-flop outputs as well as to obtain counters having cycle length N is less than 16.

Pin connections:



74LS02 Quad NOR Gate   74LS73 Dual J-K Flip-flop   CD 4029 4-bit Counter

Binary Ripple Counter

1. Make J = K = 1 for all the flip-flops, thereby converting the J-K flip-flops to T flip-flops. Connect all R’ inputs together to an Input Switch, and the outputs Q0,Q1,Q2,Q3 to four LED Displays.
2. Set up an Up-counting Binary Ripple Counter by making clock connections as follows: CK0 = Manual Clock (CLK-M), CK1 = Q0, CK2 = Q1, CK3 = Q2.
3. Using the Input Switch connected to the common R’ input, initialise the counter to the stat
4. Apply Manual Clock pulses and tabulate the state sequence for the entire cycle.
5. Now change the clock input connections to CK1 = Q0’, CK2 = Q1’, CK3 = Q2’, to obtain a Down-counting Binary Ripple counter.

Asynchronous (ripple) counter:

An asynchronous (ripple) counter is a single JK-type flip-flop, with its J (data) input fed from its own inverted output. This circuit can store one bit, and hence can count from zero to one before it overflows (starts over from 0). This counter will increment once for every clock cycle and takes two clock cycles to overflow, so every cycle it will alternate between a transition from 0 to 1 and a transition from 1 to 0. Notice that this creates a new clock with a 50% duty cycle at exactly half the frequency of the input clock. If this output is then used as the clock signal for a similarly arranged D flip-flop (remembering to invert the output to the input), you will get another 1 bit counter that counts half as fast. Putting them together yields a two-bit counter:

Cycle	Q1	Q2	(Q1:Q0)dec
0	0	0	0
1	0	1	1
2	1	0	2
3	1	1	3
4	0	0	0

Synchronous counter:

A simple way of implementing the logic for each bit of an ascending counter (which is what is depicted in the image to the right) is for each bit to toggle when all of the less significant bits are at a logic high state. For example, bit 1 toggles when bit 0 is logic high; bit 2 toggles when both bit 1 and bit 0 are logic high; bit 3 toggles when bit 2, bit 1 and bit 0 are all high; and so on. Synchronous counters can also be implemented with hardware finite state machines, which are more complex but allow for smoother, more stable transitions. Hardware-based counters are of this type.

Decade counter:

A decade counter is one that counts in decimal digits, rather than binary. A decade counter may have each digit binary encoded (that is, it may count in binary-coded decimal, as the 7490 integrated circuit did) or other binary encodings (such as the bi-quinary encoding of the 7490 integrated circuit). Alternatively, it may have a "fully decoded" or one-hot output code in which each output goes high in turn (the 4017 is such a circuit). The latter type of circuit finds applications in multiplexers and demultiplexers, or wherever a scanning type of behavior is useful. Similar counters with different numbers of outputs are also common. The decade counter is also known as a mod-counter when it counts to ten (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). A Mod Counter that counts to 64 stops at 63 because 0 counts as a valid digit.
A decade counter has the count sequence 0 → 1 → 2 →.....→ 8 → 9 → 0.., which can be achieved by making R’ = (Q3 •Q1)’ for all the flip-flops in a 4-bit binary counter. This forces the counter to go to the state 0000 as soon as the counter makes the transition from the state 1001 representing count 9 to the next state 1010 according to the normal up counting sequence.

Decade synchronous counter

The logic for the J-K inputs required for a Decade Synchronous Counter is as follows: J0 = K0 = 1; J1 = Q0•Q3’, K1 = Q0; J2 = K2 = Q0•Q1; J3 = Q0•Q1•Q2, K3 = Q0.

Multipurpose 4-bit Synchronous Counter

CD4029 is a multipurpose 4-bit counter capable of operating in all the four combinations of Binary/BCD and Up/Down modes, depending on the values of the control inputs B/D' and U/D'. In addition, the 4-bit output Q3Q2Q1Q0 of the counter can be preset to any value by applying the desired bits to the direct inputs D3D2D1D0 and making the Set ENable control SEN

B/D′ (Binary/BCD)	U/D′ (Up/Down)	Mode of Operation
1	1	Binary Up Counter (0000→0001→…→1111→0000…)
1	0	Binary Down Counter (1111→1110→…→0000→1111…)
0	1	BCD Up Counter (0000→0001→…→1001→0000…)
0	0	BCD Down Counter (1001→1000→…→0000→1001…)

=

Procedure

Results and Discussion: