Index

No.	Experiment Name
01	Realization of logic gates using discrete components.
02	Realization of logic gates using universal building blocks.
03	To realize Half/ Full adder using Ex-OR and basic gates.
04	To realize Half/ full subtructor using Ex-OR and basic gates.
05	To design and contruct BCD code to excess-3 code converter.
06	(a) Realization of logic expression using K-map for $y = f(x, y, z) = \leq (0, 2, 4, 5, 6)$. (b) Implementation of parity generator and checker circuit.

PABNA UNIVERSITY OF SCIENCE AND TECHNOLOGY

Department Of Information & Communication Engineering

Faculty Of Engineering & Technology

Practical Lab Report

Course Code: ICE-2102

Course Title: Digital Electronics Sessional

No of the Experiment: 01

Name of the Experiment: Realization of logic gates using discrete components.

Submitted By-

Nuzhat Tabassum

Roll: 150631
2nd year 1st Semester
Session: 2014-2015
Date of experiment: 11,04,16
Date of submission: 02,05,16

Submitted To-

Sohag Sarker

Assistant Professor
Department of Information & Communication Engineering
Pabna University of Science & Technology

Signature: 09.05.2016

Name of the Experiment:

Realization of logic gates using discrete components.

Objective:

To construct logic gates OR, AND, NOT, NAND gates using discrete components and verify their truth tables.

Theory:

OR:

This operation is represented by a plus sign. For example: $x+y=z$ is read "x OR y is equal to z". Meaning that $z=1$, if $x=1$ or if $y=1$ or if both $x=y=1$. if both $x=y=0$ then $z=0$.

AND:

The operation is represented by a dot or by the absence of an operation. For example: $x.y=z$ or $xy=z$ is read "x AND y is equal to z". The logical operation AND is interpreted to mean that $z=1$ if and only if $x=1$ and $y=1$; otherwise $z=0$.

NOT:

This operation is represented by a prime or a bar. for example: $x' = z$ or $\bar{x} = z$ is read "x not, is equal to z". meaning that z is what x isn't, if $x = 1$ that $z = 0$ but if $x = 0$ then $z = 1$.

NOR:

The NOR function is the component of OR function and its name is an abbreviation of NOT-OR. Uses an 'OR graphic symbol followed by a small circuit.

NAND:

The NAND function is the component of the AND function as indicated by a graphic symbol consists of an AND graphic symbol followed by small circuits.

Appartus Required:

(i) Resistor.
(ii) Diode.
(iii) Transistor.
(iv) Bread-board.
(v) A.V0 meter.
(vi) Connecting wires.
(vii) power supply.

Circuit diagram :

OR Gate :

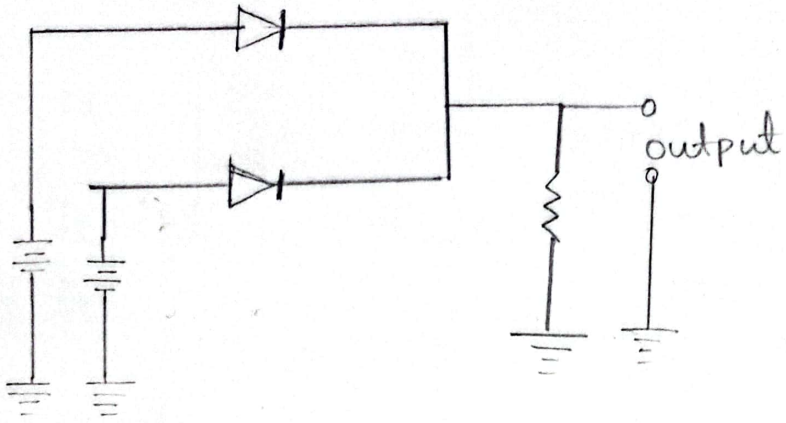


fig 1 : OR gate.

AND Gate :

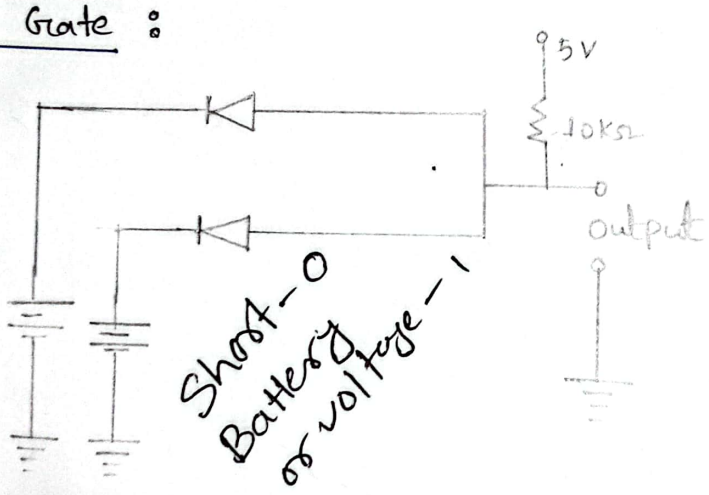


fig 2 : AND gate.

NOT Gate :

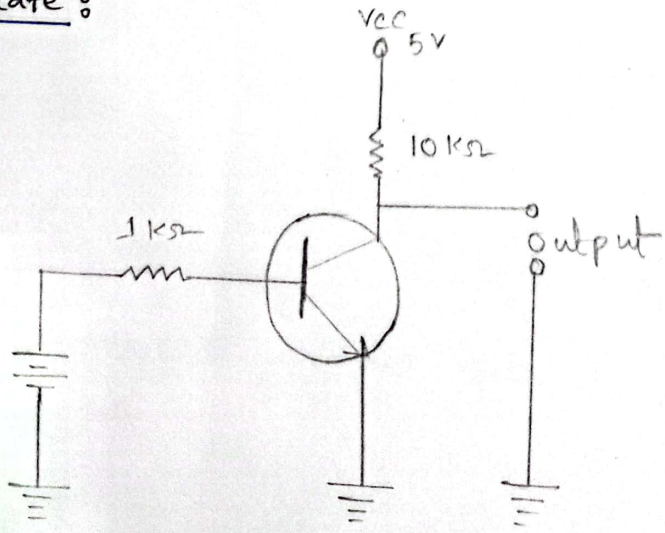


fig 3 : NOT gate

NOR Gate :

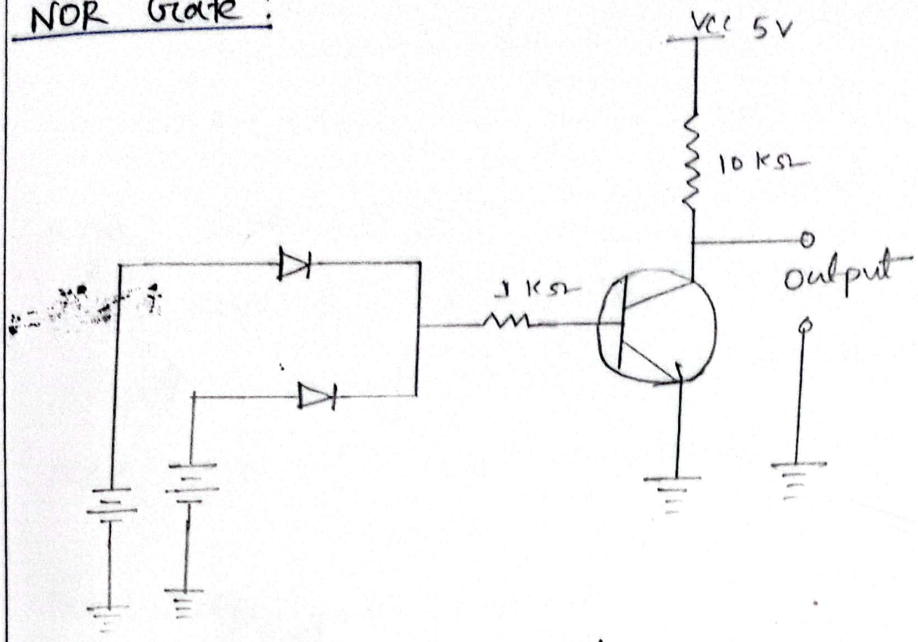


fig 4 : NOR gate.

NAND Gate :

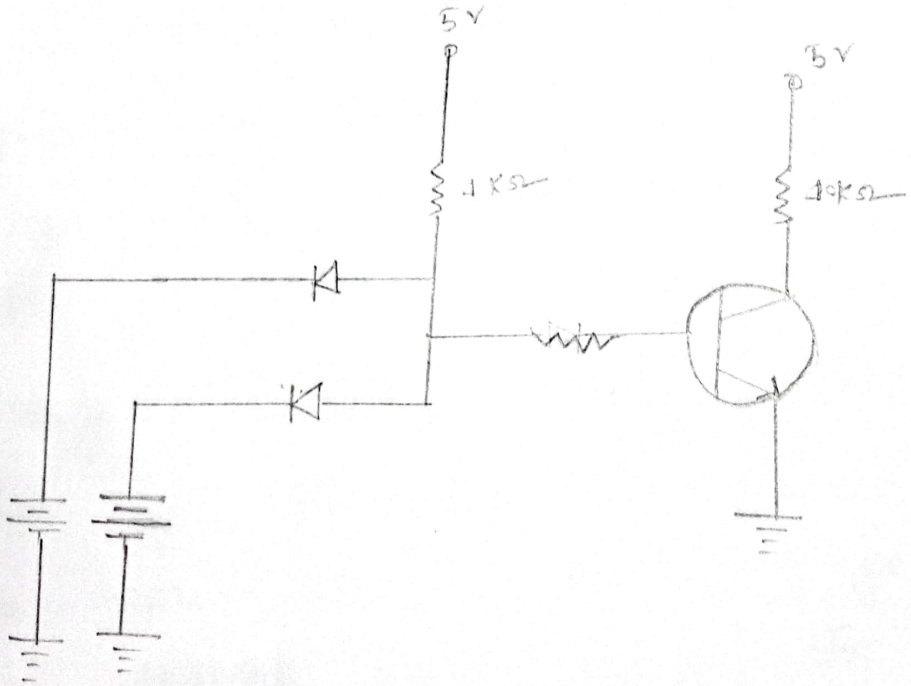


fig 5 : NAND gate.

Description:

We used diodes, resistors, transistor for drawing their circuit. For OR gate, two diodes and one resistor were used. In the circuit we applied 5V input voltage and measured output voltage from one terminal of resistor and ground. We measure voltage four time.

For AND gate circuit we used diodes and one resistor on bread board. Then applied voltage through two diodes and measured voltage from resistor and ground.

For NOT gate we used two resistors and one transistor. We applied 5V, in output we got 0V and in input 0V applied then we got 5V in the output.

Similarly, for NOR and NAND gates we used resistors, diodes and transistor for circuit set up. And then applied 5V for input voltage and measured desired output voltage for each gates. This output voltage verify their truth tables for each gates.

Experimental data Table:

Table 1: Data table for OR gate (using logic level).

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

Table-2: Data table for AND gate (using logic level).

A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

Table-3: Data table for NOT gate (using logic level)

A	A'
0	1
1	0

Table-4: Data table for NOR gate (using logic level)

A	B	Z
0	0	1
0	1	0
1	0	0
1	1	0

Table-5: Data Table for NAND gate (using logic level)

A	B	Z
0	0	1
0	1	1
1	0	1
1	1	0

Experimental data Table:

Table-1: Data table for OR gate (Using logic level).

Truth Table:

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

Table-2: Data table for AND gate (Using logic level).

Truth Table.

A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

Table-3: Data table for NOT gate (Using logic level)

Truth table:

A	A'
0	1
1	0

Table - 4: Data table for NOR gate (using logic level)

Truth table:

A	B	Z
0	0	1
0	1	0
1	0	0
1	1	0

Table - 5: Data table for NAND gate (using logic level)

Truth table:

A	B	Z
0	0	1
0	1	1
1	0	1
1	1	0

Result and Discussion:

From the truth table of experimental data, we can see that OR, AND, NAND, NOT and NOR all gates, their values have satisfied their conditions.

From OR table we have seen that we got same output voltage for the input voltage. We got 0V when the input voltage is zero. From AND table we have seen that if two input voltages are true then it drop voltage.

NOT, NOR and NAND gate we got desired output voltage that satisfied their truth tables.

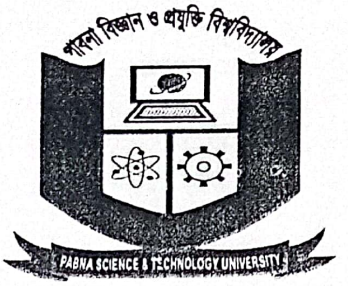
precaution:

(i) Resistors, Diodes, transistor and all other components were connected properly and tightly.

(ii) Applied voltage ~~and~~ were measured very carefully.

(iii) Measured output voltage very carefully.

PABNA UNIVERSITY OF SCIENCE AND TECHNOLOGY



Department Of Information & Communication Engineering

Faculty Of Engineering & Technology

Practical Lab Report

Course Code: *ICE-2102*

Course Title: *Digital Electronics Sessional*

No of the Experiment: 03

Name of the Experiment: To realize Half / Full adder Using Ex-OR and basic gates.

Submitted By-

Nuzhat Tabassum

Roll: 150631
2nd year 1st Semester
Session: 2014-2015
Date of experiment: 2,5,16
Date of submission: 21,8,16

Submitted To-

Sohag Sarker

Assistant Professor
Department of Information & Communication Engineering
Pabna University of Science & Technology

Signature:

Name of the Experiment: To realize Half / full adder using Ex-OR and basic gates.

Objective:
To construct Half/ full adder using Ex-OR gate and basic gates and verify their thruth tables.

Theory:
Digital computers perform a variety of information processing tasks. Among the basic functions encounted are the various arithmatic operations. The most basic arithmatic operation is the addition of two binary digits. The higher significant bit of this result is called a carry.

Adder:
A combinational circuit that performs the addition of two / three bits is called adder. There are two types of adder. Half adder and Full adder. The two adder circuits are the first combinational circuits we shall design.

Half-Adder:

A combinational circuit that performs the addition of two bits is called a half adder. we find that this circuit needs two binary inputs and two binary outputs. The input variables designed the augend and addend bits. The output variables produce the sum and carry. we orbitarily assign symbols x and y to the two inputs and S (for sum) and C (for carry) to the outputs. From the truth table we can see that,

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

The simplified Boolean functions for the two outputs can be obtained directly from the truth table. The simplified sum of products expressions are:

$$S = x'y + xy'$$
$$C = xy$$

The logic diagram of a half-adder circuit is given below:

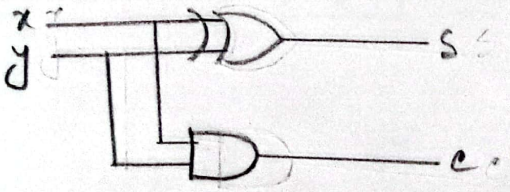


fig 1: Half-adder circuit.

from the circuit we can write an expression for an half adder circuit as $C = xy$ and $S = x \oplus y$.

Full - Adder : A combinational circuit that performs the addition of three bits is a full-adder. This circuit forms the arithmatic sum of three input bits. It consists of three input and two output. Two of the input variables denoted by x and y represent the two significant bits to be added. The third input z, represent the carry from the previous lower significant position. The two outputs are designed by the symbols S for sum and C for carry and the truth table of the full adder is given below:

x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

The logic diagram of a full-adder circuit is given below:

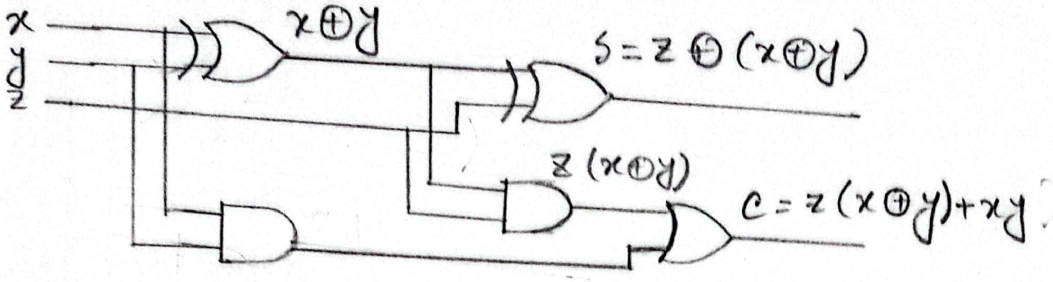


fig 2: Logic diagram of a full-adder.

From the circuit we can write an expression for full-adder circuit i.e.

$$S = z \oplus (x \oplus y)$$
$$= z'(xy' + x'y) + z(xy + x'y')$$
$$= z'(xy' + x'y) + z(xy + x'y')$$
$$= x'y'z' + xyz' + xyz + x'y'z$$

and $C = z(xy' + x'y) + xy$

$$= xy'z + x'yz + xy$$

Apparatus Required:

(i) Bread-board.
(ii) Resistors.
(iii) Transistors.
(iv) Diodes.
(v) IC-7486.
(vi) IC-7404
(vii) IC-7408
(viii) IC-7432
(ix) Avo-meter.

Circuit diagram:

(i) Half-Adder:

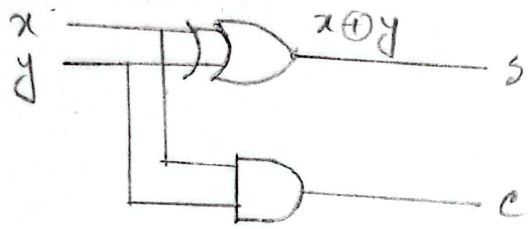


fig 1: A Half-adder circuit.

(ii) Full-Adder:

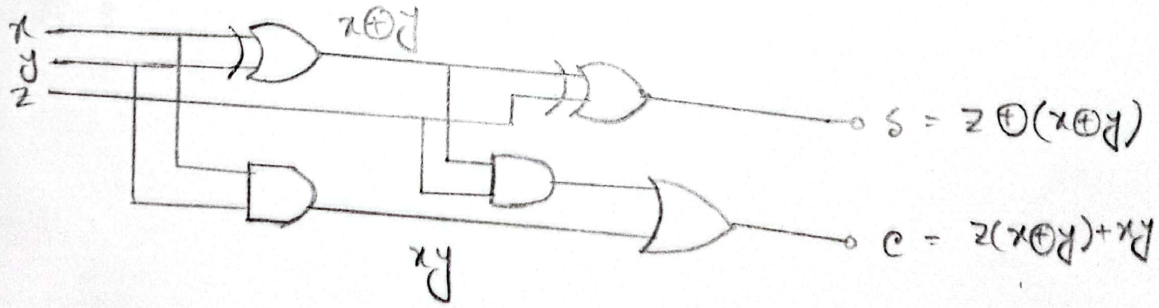


fig 2: A full-adder circuit.

procedure:

(i) We used two types of IC for drawing an half adder circuit on the bread board like fig-1.

(ii) We supplied current from the ~~switch~~ switch on the bread-board.

(iii) For half adder we used IC-7486 and IC-7408 and after completing circuit set-up we verified the truth table.

(iv) For a full-adder circuit we used three types IC i.e IC-7486, IC-7408 and IC-7432 for drowing the circuit on the bread board like fig-2. As a same process we supplied the power and verified the truth table.

Experimental Data Table:

Table-1: Data table for an half-adder circuit.

input		output	
x	y	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Table-2: Data table for full-adder circuit.

input			output	
x	y	z	c	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Experimental data table:

Table-1: Data table for Half adder circuit.

Input		output	
x	y	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Table-2: Data table for full adder circuit.

Input			output	
x	y	z	c	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Result and Discussion:

From the data table of an half-adder circuit, we see when input 0,0 then output is 0(c), 0(s). When input is 0, 1, then output is 0(c), 1(s). Then we supplied 1,0 and got output 0(c), 1(s) and we supplied 1,1 and got output 1(c), 0(s). Thus verified the truth table.

From the data table of an full-adder circuit we use the input variable and position 1,2,4,7 then their sum is 1 and the light has on and otherwise 0 and the light has off. When output carry 1 and the light has on otherwise 0 and light has off. Thats verified the desired results.

precaution:

(i) The connecting wires and all the required components connected tightly.

(ii) Voltage were measured carefully.

PABNA UNIVERSITY OF SCIENCE AND TECHNOLOGY

Department Of Information & Communication Engineering

Faculty Of Engineering & Technology

Practical Lab Report

Course Code: *ICE-2102*

Course Title: *Digital Electronics Sessional*

No of the Experiment: 04

Name of the Experiment: To realize Half/Full subtractor using ex-OR and basic gates.

Submitted By-

Nuzhat Tabassum

Roll: 150631

2nd year 1st Semester

Session: 2014-2015

Date of experiment: 2,05,16

Date of submission: 21,08,16

Submitted To-

Sohag Sarker

Assistant Professor

Department of Information & Communication Engineering

Pabna University of Science & Technology

Signature:

Name of the Experiment: To realize Half/ Full subtractor using Ex-OR and basic gates.

Objective: To construct half/ full subtractor using Ex-OR and basic gates and to verify the truth table.

Theory:

The subtraction of two binary numbers may be accomplished by taking the complement of the subtracted and adding it to be minued. By this method, the subtruction operation becomes an addition operation requiring full-adders for its machine implementation. It is possible to implementation subtraction with logic gates/circuits in a direct manner. By this method, each subtacted method bit of the number is subtracted form a difference bit to its corresponding significant minued bit. If the minued bit is smaller than the subtracted bit, a 1 is borrowed from the next significant position. That is called borrow.

Two types of subtractor circuit is used. Half subtractor and full subtractor.

Half Subtractor: A half subtractor is a combinational circuit that subtracts two bits and produces their difference. It also has an output to specify if a 1 has been borrowed. Designate the minued bit by x and the subtracted bit by y. To perform $x-y$. The half subtractor needs two output. One is difference symbol D and the other is borrow symbol B. From the truth table we can see the relationship between input and output for a half subtraction.

x	y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

The output borrow B is a 0 as long as $x \geq y$. It is a 1 for $x=0$ and $y=1$. The D output is the result of the arithmatic operation $2B+x-y$.

The Boolean functions for the two outputs of the half-subtractor are derived directly from the truth table.

$$D = x'y + xy'$$

and $B = x'y$.

Full-Subtractor: A full-subtractor is a combinational circuit that performs a subtraction between two bits, taking into account that a 1 may have been borrowed by a lower significant stage. This circuit has three inputs and two outputs. The three inputs x, y and z, denote the minued, subtrahend and previous borrow respectively. The two outputs D and B represent the difference and output borrow respectively. The truth table for the circuit is given below:

x	y	z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

The simplified sum of products output functions are give below:

$$D = x'y'z + x'yz' + xy'z' + xyz$$

$$B = x'y + x'z + yz$$

Appartus Required:

(i) Bread-board.
(ii) IC-7408
(iii) IC-7404
(iv) IC-7432
(v) IC-7486
(vi) Resistors and Diodes.
(v) connecting wires.
(vi) AVO meter.

Circuit diagram:

(i) A Half-subtractor:

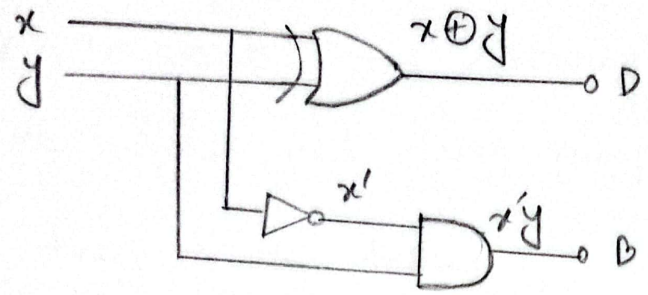


fig 1: Half subtractor circuit diagram.

(ii) A full-subtractor:

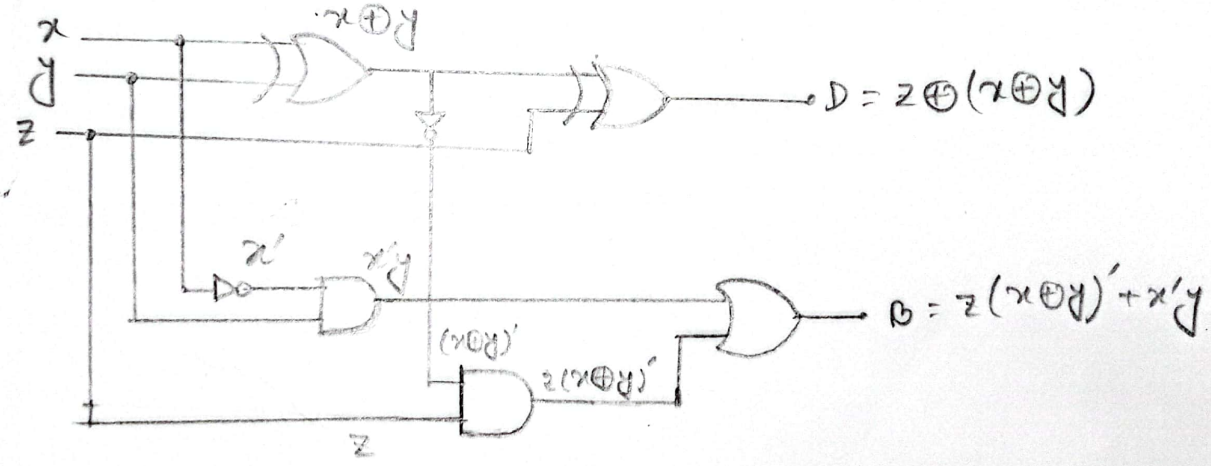


fig 2: Full Subtractor circuit diagram.

procedure:

(i) For a half subtractor to set up a the circuit design we used three types of IC's i.e IC-7486, IC-7404, IC-7408 and drowing the circuit on the bread board like fig-1. And then supplied the current and verified the truth table.

(ii) For a full subtractor we used four types of IC's i.e IC-7486, IC-7404, IC-7408, IC-7432 and drawing the circuit on the bread board. Then the same process the supplied the current and verified the truth table for full-subtractor.

Experimental data table:

Table-1: Data table for a half subtractor.

input		output	
x	y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

Table-2: Data table for a full subtractor.

input			output	
x	y	z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Experimental Data Table:

Table-1: Data table for a half subtractor.

input		output	
x	y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

Table-2: Data table for a full subtractor.

input			output	
x	y	z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Result and Discussion:

From the data table of a half subtractor we can see that when the input 0,0 the output is 0(B), 0(D). When input is 0, 1 then the output- 1(B) and 1(D). when we ⁶ input 1,0 the the output shows 0(B) and 1(D). And when we input 1, 1 then the output is 0,0. Thus the data table verified the truth table.

From the data table of a full subtractor we can see that the input three variable are implemented and Eight combinations accurs and four times borrow is 1 and the four time the light is off and other times is ON. Thus this data can verify the truth table.

precaution:

(i) The connecting wires and other necessary components were connected tightly on the bread board.

(ii) Voltage mearsured very carefully.

PABNA UNIVERSITY OF SCIENCE AND TECHNOLOGY

Department Of Information & Communication Engineering

Faculty Of Engineering & Technology

Practical Lab Report

Course Code: *ICE-2102*

Course Title: *Digital Electronics Sessional*

No of the Experiment: 05

Name of the Experiment: To design and construct BCD code to excess-3 code converter.

Submitted By-

Nuzhat Tabassum

Roll: 150631

2nd year 1st Semester

Session: 2014-2015

Date of experiment: 09, 05, 16

Date of submission: 21, 08, 16

Submitted To-

Sohag Sarker

Assistant Professor

Department of Information & Communication Engineering

Pabna University of Science & Technology

Signature:

Name of the Experiment: To design and construct BCD code to excess-3 code converter.

Objective:

To construct BCD to excess-3 code converter and verify the truth table.

Theory:

BCD Codes: The term BCD refers to representing the 10 decimal digits in binary terms. Which simply means to count in binary. Numeric codes used to represent decimal digits are called Binary Coded Decimal (BCD). Codes. A BCD codes is one which the digits of a decimal number are encoded. One at a time into group of four binary digits. There are a large number of BCD codes in order to represent decimal digits 0, 1, 2.....9 it is necessary to use a sequence of at least four binary digits. Such a sequence of binary digits which represent a decimal digit is called code word.

Excess-3-Code: It is a non-weighted code. It is also a self-complementing BCD code used in decimal number arithmatic units. The excess-3 code for the decimal number is performed in the same manner as BCD expect that decimal number 3 is added to the each decimal unit before encoding it to binary.

Code Converters:

The availability of a large variety of codes for the same discrete elements of information results in the use of different codes by different digital systems. It is sometimes necessary to use the output of one system as the input to another. A Conversion circuit must be inserted between the two systems if each uses different codes for the same information. Thus, a code converter is a circuit that makes the two systems compatible even though each uses a different binary code.

Appartus Required:

(i) Bread-board.
(ii) IC-7408
(iii) IC-7404
(iv) IC-7432
(v) Connecting wires.

Circuit Diagram :

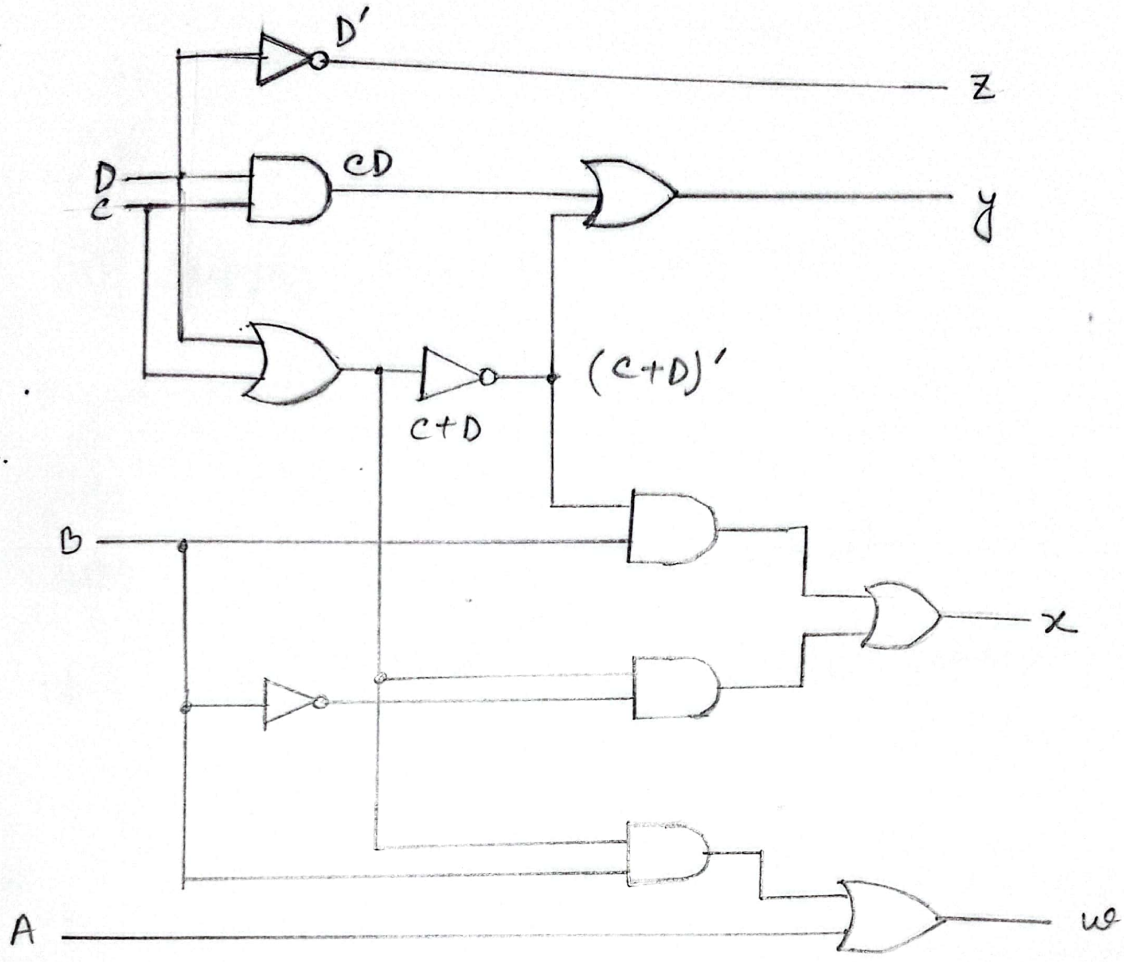


fig 1: Logic diagram for BCD-to-3-excess code Converter.

Procedure:

(i) Construct the BCD to Excess 3 converter circuit on the bread board as shown in fig.

(ii) Supply current from the switch on the bread-board.

(iii) Then Observe the result and verify it with corresponding truth table.

Experimental Data table:

Table-1: Truth table for code conversion from BCD to excess-3 code.

Input BCD				Output Excess-3 code			
A	B	C	D	w	x	y	z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

Experimental Data table :

Table-1 : Truth table for code conversion from BCD to excess -3 code.

Input BCD				Output Excess-3 code			
A	B	C	D	w	x	y	z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

Result and Discussion:

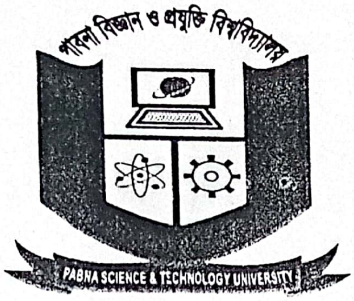
In the truth table we can see that input and output variables are shown. The bit combinations for the inputs and their corresponding output are obtained directly. We note that four binary variables may have 16-bit combinations, only 10 of which are listed in the truth table. The six bit Combinations aren't listed for the input variables. Since they will never accur we can assign to the output variables either a 10 or a 0, whenever gives a simpler circuit. So we can overlook them.

precaution:

(i) All the components are connected tightly on the bread-board.

(ii) Measure the outputs very carefully.

PABNA UNIVERSITY OF SCIENCE AND TECHNOLOGY



Department Of Information & Communication Engineering

Faculty Of Engineering & Technology

Practical Lab Report

Course Code: *ICE-2102*

Course Title: *Digital Electronics Sessional*

No of the Experiment: 06

Name of the Experiment: (a) Realization of logic expression using K-map for $y = f(x, y, z) = \sum(0, 2, 4, 5, 6)$.
(b) Implementation of parity generator and checker circuit using Ex-OR and Ex-NOR gates.

Submitted By-

Nuzhat Tabassum

Roll: 150631
2nd year 1st Semester
Session: 2014-2015
Date of experiment: 30,05,16
Date of submission: 21,08,16

Submitted To-

Sohag Sarker

Assistant Professor
Department of Information & Communication Engineering
Pabna University of Science & Technology

Signature:

Name of the Experiment:

(a) Realization of logic expression using K-map for the expression,

$$y = f(x, y, z) = \sum(0, 2, 4, 5, 6)$$

(b) Implementation of parity Generator and Checker circuit using Ex-OR and Ex-Nor gates.

Theory:

The map-method: The map method provides a simple straightforward procedure for minimizing Boolean functions. This method may be regarded either as a pritorial form of a thruth table or as an extension of the veen diagram.

The map is a diagram made up of squares. Each square represents one minterm, since any Boolean function can be expressed as a sum of minterms. it follows that a Boolean function is recognized graphically in the map from the area enclosed by those squares whose minterms are included in the function. In fact the map represents a visual diagram of all possible ways a function may be expressed in a standard form. By recognizing various patterns the user can derive alternative algebric expressions for the some function from which we can select the simplest one.

we shall assume that the simplest algebric expression is any one in a sum of products or product of sums that has a minimum number of literals.

parity Generator and parity checker:

A parity bit is an extra bit included with a binary message to make the number of 1's either or even. The message age, including the parity bit. its from transmitted and then checked at the recieveing end for errors. An error is detected if the checked correspond to the one transmitted. The circuit that generates the parity bit in the transmittor is called a parity generator, the circuit that checks the parity in the reciever is called a parity cheeker.

Consider a three bit message to the transmitted with an odd parity bit for the parity generator. The parity bit is the output. For odd parity the bit P is generated so as to make the total number of 1's odd (including P). The function for P can be expressed as following.

$$P = z \odot (x \oplus y)$$

The three bit message and the parity bit are transmitted to their destination where they are applied to a parity checker circuit. An error accurs during transmission if the parity at the four bits recieved be even, since the binary information

transmitted as originally odd. The output c of the parity checker should be a 1 when an error occurs. i.e when the number of 1's in the four inputs is even. The function of 1's in the four inputs are even. The function of parity checker C can be expressed as follows -

$$C = x \odot y \odot z \odot p$$

Appartus Required:

(i) Bread board.
(ii) IC-7404
(iii) IC-7486
(iv) IC-7408
(v) IC-7432 and connecting wires.

circuit diagram:

(i)

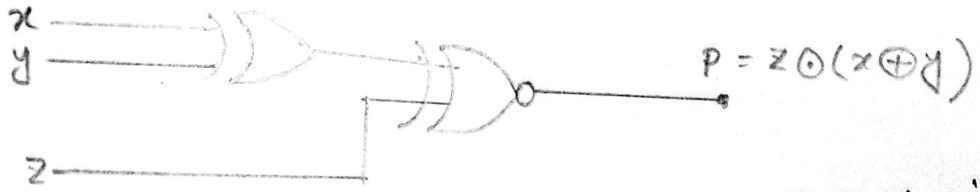


fig 1: 3-bit odd parity generator circuit.

(ii)

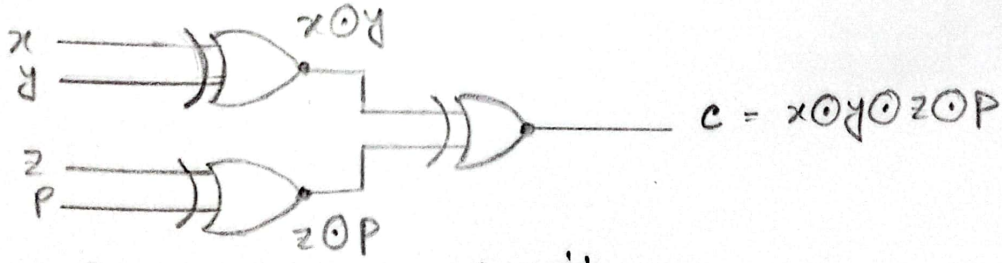


fig 2: checker circuit.

(iii)

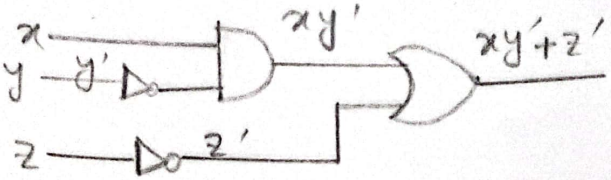


fig-3: K-mapping.

Experimental procedure:

(i) We used two Ic for drawing an odd parity generator on the bread board like figure-1. We used IC-7404 and IC-7486 as NOT-gate and Ex-OR gate respectively. We transmit three bit as input and verify the thruth table for odd parity generator.

(ii) Then draw an odd parity checker circuit like figure-2 by using IC-7486 and IC-7404. Here we transmitted 4 bit (3 bit message and the parity bit) and get the output C. We verify the truth table for the checker circuit.

(iii) Then we draw a circuit for K-map for $y = f(x, y, z) = \Sigma(0, 2, 4, 5, 6)$ and verify the truth table.

Experimental data table:

Data table for realization of logic expression using k-map for the expression and implementation of parity generator check Ex-OR and Ex-NOR gates.

Table-1: Truth table for $y = f(x,y,z) = \Sigma(0,2,4,5,6)$ using k-map.

x	y	z	z'	xy'	z'+xy'
0	0	0	1	0	1
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	0	0	0

Table-2: Truth table for odd parity generator.

x	y	z	P
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Table-3: Truth table for odd parity check.

x	y	z	P	parity check c
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Experimental data Table: Table for realization of logic expression using K-map for the expression and Implementation of parity generator. Check Ex-OR and Ex-NOR gates.

Table-1: Thruth table for $y = f(x,y,z) = \leq(0,2,4,5,6)$ using K-map.

x	y	z	z'	xy'	xy'+z'
0	0	0	1	0	1
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	0	0	0

Table-2: Truth table for odd parity generator.

x	y	z	P
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Table-3: Truth table for odd parity check.

x	y	z	P	parity check r
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Result and Discussion:

From the truth table - 1 we see that the expression $y = f(x, y, z) = \leq (0, 2, 4, 5, 6)$ can be simplified as $y = z' + xy'$. From the truth table-2 we see that $P = 1$ when the number of 1's is in x, y and z is even.

Again from the truth table - B, we see that $c = 1$ when the number of 1's in the four inputs is even. The output c is 1 when an error occurs.

precaution :

(i) Using all gates properly.

(ii) All the conections should be made properly.

(iii) Ic should not be ~~rese~~ reversed.