<u>Department of Electrical Engineering</u>

Faculty Member:Miss Rafia **Date: 3-4-25**

Group No.:

Semester:2nd **Section: bsds-2a**

EE-221: Digital Logic Design

Lab 6: Binary to Gray and Gray to Binary Code Conversion

Name	Reg. No	Viva / Lab Performance 5 Marks	Analysis of data in Lab Report 5 Marks	Modern Tool Usage 5 Marks	Ethics and Safety 5 Marks	Individual and Team Work 5 Marks	Total marks Obtained 25 Marks
AILYA ZAINAB	523506						
IMAN NAEEM	525378						
LAIBA NASIR	510419						
LUQMAN SHEHZAD	507599						

Lab6: Binary to Gray and Gray to Binary Code Conversion

This Lab has been divided into two parts:

In first part you are required to design and implement a binary to gray and gray to binary code converter. You will be cascading these two converters thus implementing a binary to gray coder and decoder (gray to binary).

The next part is the Verilog Modeling and Simulation of the Circuit you implemented in you first lab.

Objectives:

- Understand steps involved in design of combinational circuits
- Understand binary codes for decimals and their hardware realization
- Write code for combinational circuits using Verilog Gate Level Modeling
- Design a circuit in Verilog by calling different modules

Lab Instructions

- This lab activity comprises three parts, namely Pre-lab, Lab tasks, and Post-Lab Viva session.
- The lab report will be uploaded on LMS three days before scheduled lab date. The students will get hard copy of lab report, complete the Pre-lab task before coming to the lab and deposit it with teacher/lab engineer for necessary evaluation. Alternately each group to upload completed lab report on LMS for grading.
- The students will start lab task and demonstrate design steps separately for step-wise evaluation(course instructor/lab engineer will sign each step after ascertaining functional verification)
- Remember that a neat logic diagram with pins numbered coupled with nicely patched circuit will simplify trouble-shooting process.
- After the lab, students are expected to unwire the circuit and deposit back components before leaving.
- The students will complete lab task and submit complete report to Lab Engineer before leaving lab.
- There are related questions at the end of this activity. Give complete answers.

Pre-Lab Tasks: (To be done before coming to the lab) (2 marks)

- What is a self-complementing code? Name any two of them; show their complementing nature with examples and describe advantages.

- In the lab you would be implementing a gray to binary and binary to gray code converter. Make a truth table for both the codes by filling in the following tables and Simplify the expressions for W,X,Y,Z in terms of A,B,C,D and vice versa. Also give some applications in which gray code could be used.

  HINT:

  Our inputs and outputs are of 4-bit each. You will have to make 4 K-Maps (Consider W as independent function of A,B,C,D, Make K-Map and simplify it). Arrive at the simplest expression for each output.

Binary to Gray Converter

Dec	Binary	Gray

	A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

Simplified Expressions

W	A	A
X	A'.B + A.B'	A ⊕ B
Y	B'.C + B.C'	B ⊕ C
Z	C'. D + C.D'	C ⊕ D

Gray to Binary Converter

Dec	Gray				Binary			
	W	X	Y	Z	A	B	C	D
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	1	0	0	1	0
3	0	0	1	0	0	0	1	1
4	0	1	1	0	0	1	0	0
5	0	1	1	1	0	1	0	1
6	0	1	0	1	0	1	1	0
7	0	1	0	0	0	1	1	1
8	1	1	0	0	1	0	0	0
9	1	1	0	1	1	0	0	1

10	1	1	1	1	1	0	1	0
11	1	1	1	0	1	0	1	1
12	1	0	1	0	1	1	0	0
13	1	0	1	1	1	1	0	1
14	1	0	0	1	1	1	1	0
15	1	0	0	0	1	1	1	1

Simplified Expressions

A	W
B	$W \oplus X$
C	$W \oplus X \oplus Y$
D	$W \oplus X \oplus Y \oplus Z$

Applications of Gray Code:

- Gray codes are used in linear and rotary position encoders in preference to weighted binary encoding. This avoids the possibility that, when multiple bits change in the binary representation of a position, a misread will result from some of the bits changing before others.
- Gray codes are also used in labelling the axes of Karnaugh maps since 1953 as well as in Händler circle graphs since 1958, both graphical methods for logic circuit minimization.
- Digital logic designers use gray codes extensively for passing multi-bit count information between synchronous logic that operates at different clock frequencies. The logic is considered operating in different "clock domains". It is fundamental to the design of large chips that operate with many different clocking frequencies.
- In modern digital communications, gray codes play an important role in error correction.

HINT:

Our inputs and outputs are of 4-bit each. You will have to make 4 K-Maps

(Consider A as independent function of W,X,Y,Z. Make K-Map and simplify it).

Arrive at the simplest expression for each output.

Dec	Gray				Binary			
	W	X	Y	Z	A	B	C	D

A=

B=

C=

D=

Draw the logic diagram for the Binary-to-Gray and Gray-to-Binary code converters using Exclusive-OR gates in the space provided below.

Binary

Gray

Binary

W
X
Y
Z
A
B
C
D
A
B
C
D

Only the following gates are available to you for lab tasks.

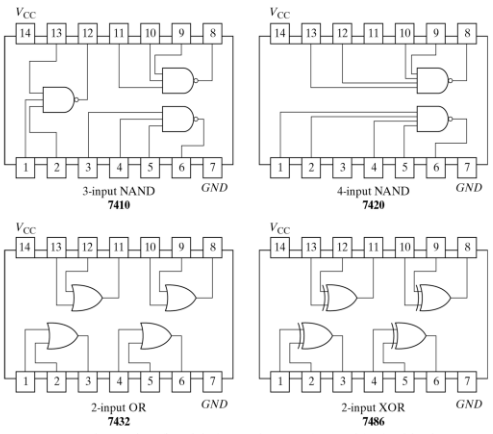


Fig. 11-1(cond) Digital Gates in IC Packages with Identification Numbers and Pin Assignments

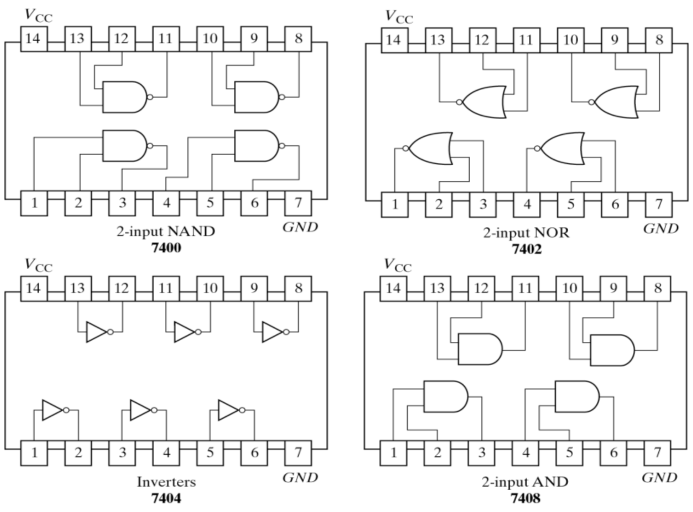


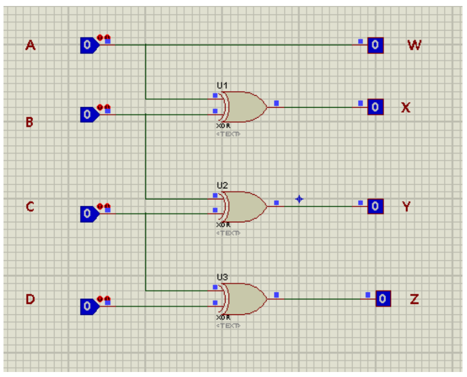
Fig. 11-1 Digital Gates in IC Packages with Identification Numbers and Pin Assignments

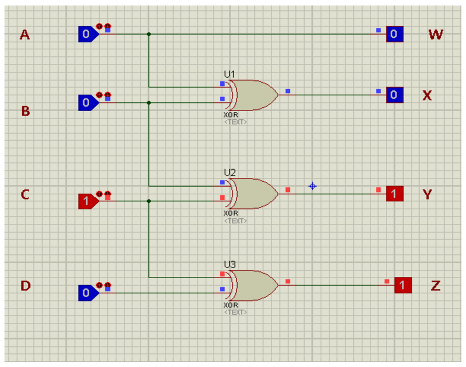
Lab Tasks:

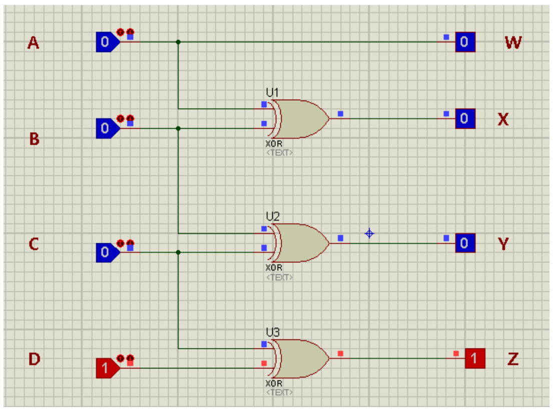
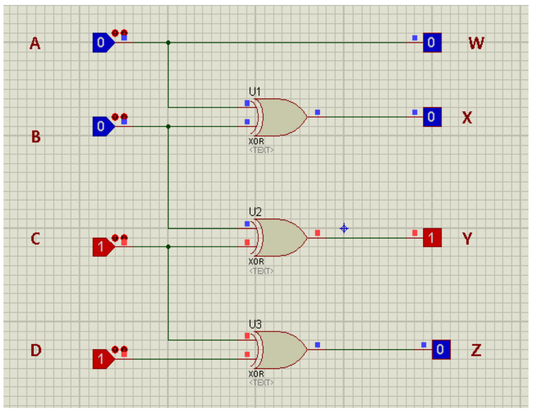
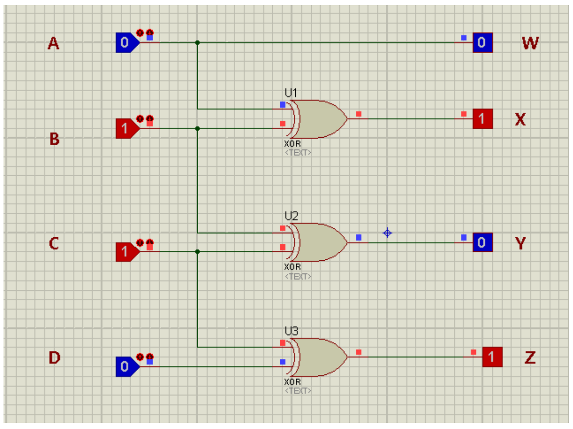
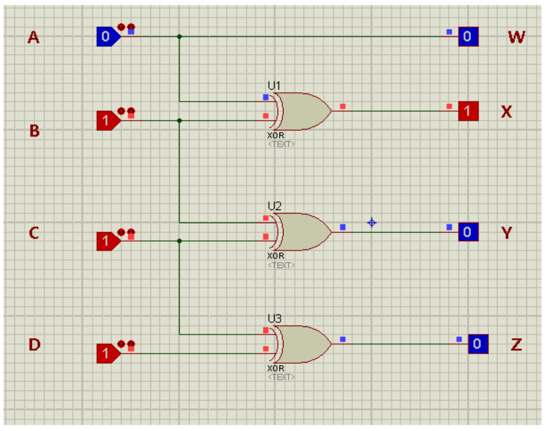
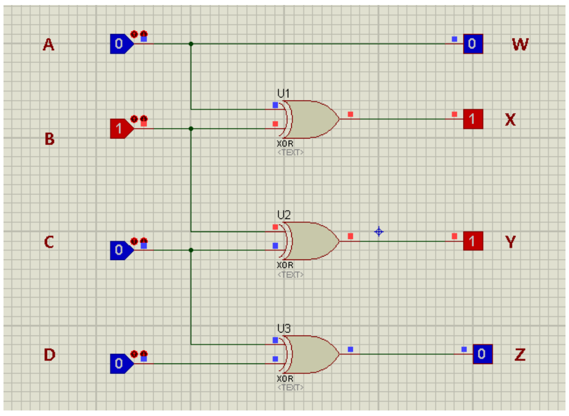
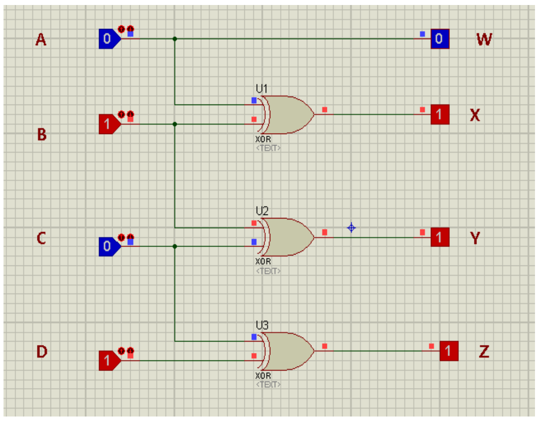
Lab Task 1: (2)

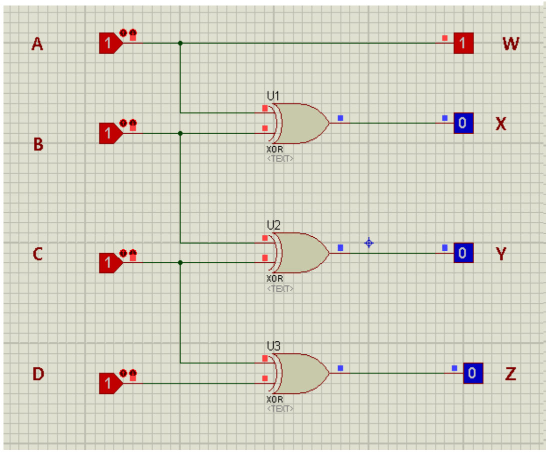
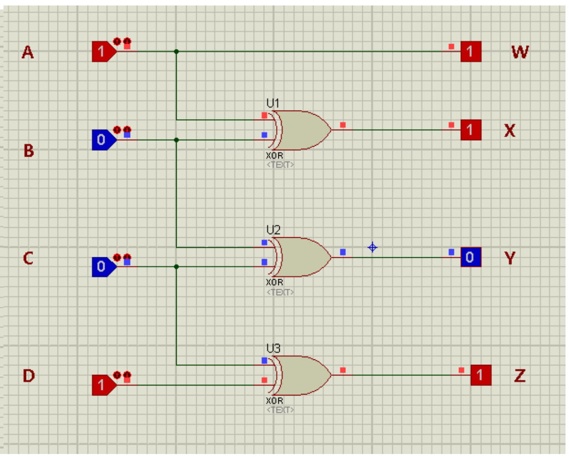
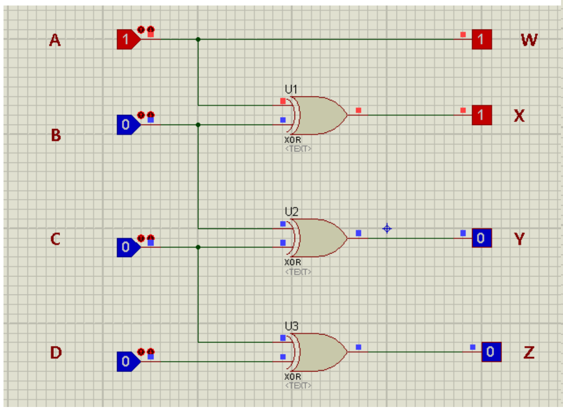
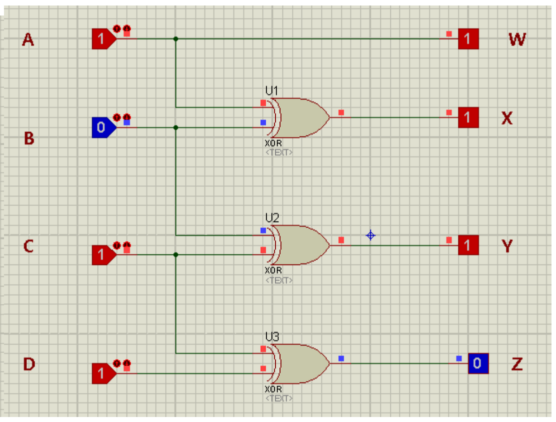
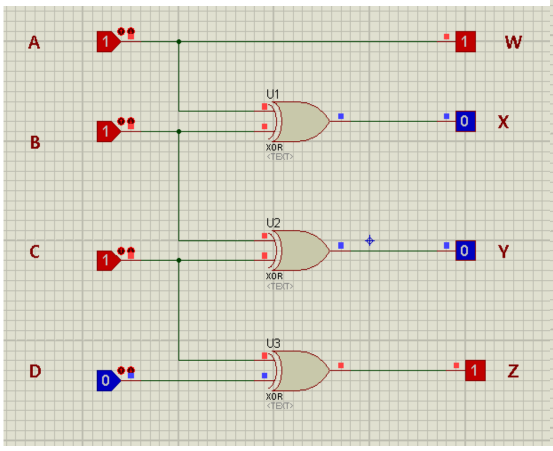
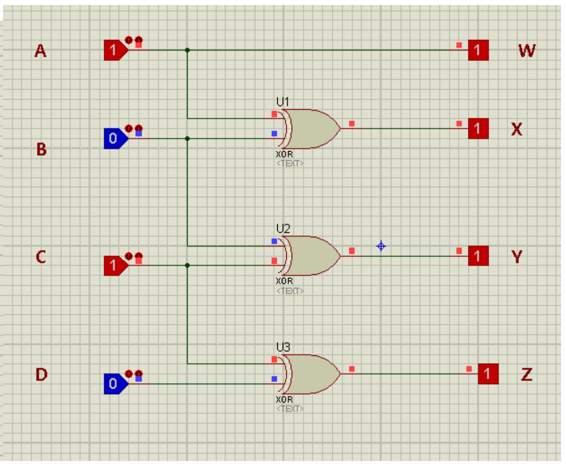
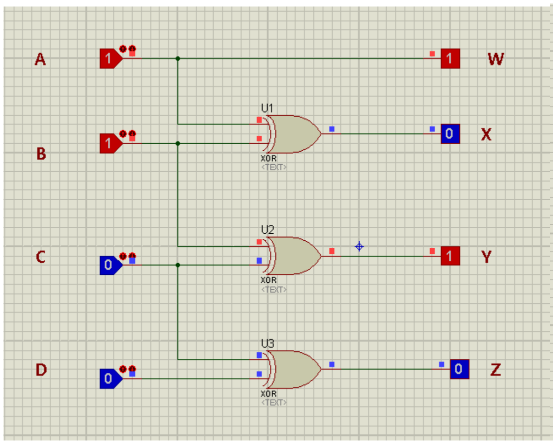
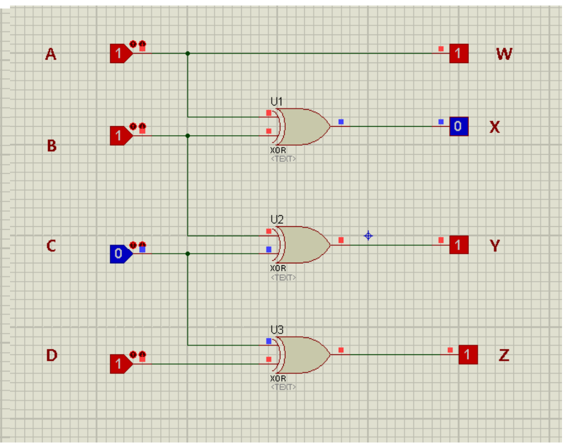
Implement the Binary to Gray Code Converter using logic gates. Simulate in Proteus. Show the results to your Teacher/Lab Engr. What and how many gates did you use? **Do not dispatch your hardware. You will need it in lab task 3.**

BINARY TO GRAY CODE IN PROTEUS

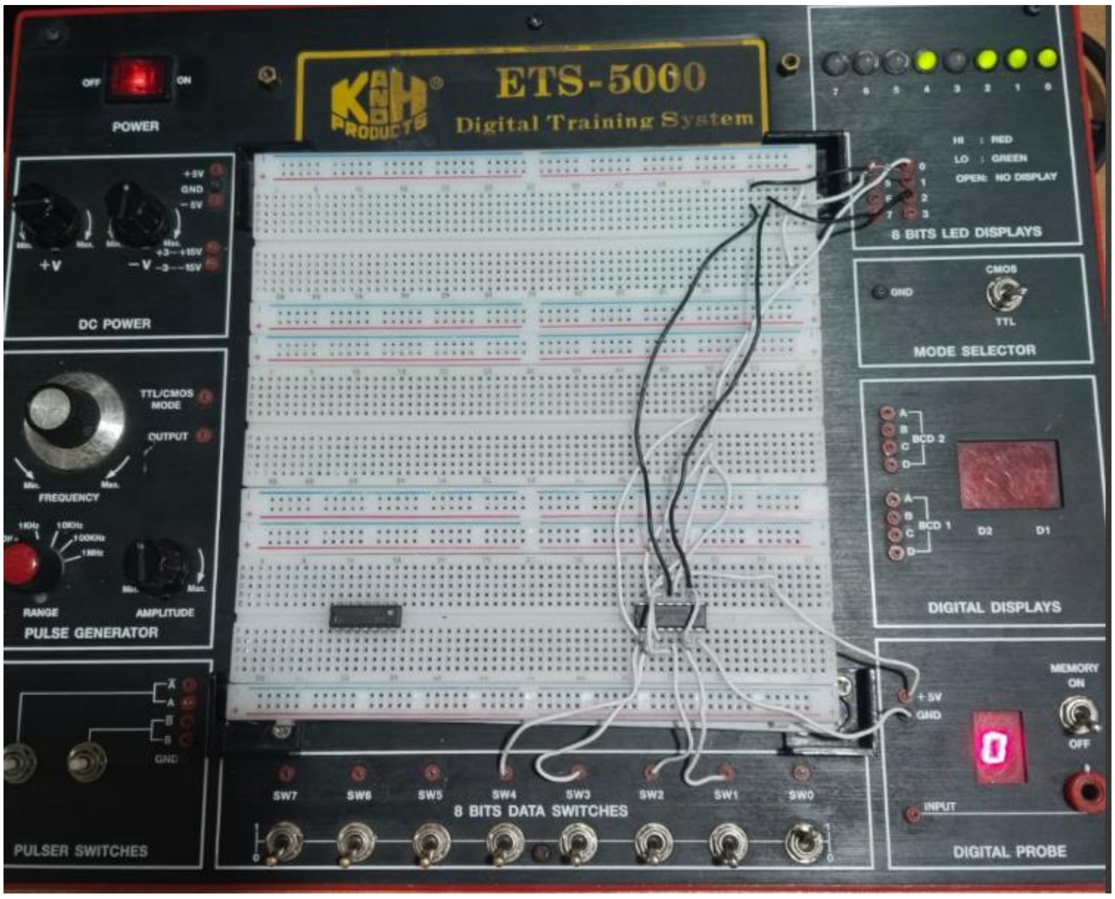





A
0
B
0
C
0
D
1
U1
U2
U3
XOR
W
0
X
0
Y
0
Z
1

A
0
B
0
C
1
D
1
U1
U2
U3
XOR
W
0
X
0
Y
1
Z
0

A
0
B
1
C
1
D
0
U1
U2
U3
XOR
W
0
X
1
Y
0
Z
1

A
0
B
1
C
1
D
1
U1
U2
U3
XOR
W
0
X
1
Y
0
Z
0

A
0
B
1
C
0
D
0
U1
U2
U3
XOR
W
0
X
1
Y
1
Z
0

A
0
B
1
C
0
D
1
U1
U2
U3
XOR
W
0
X
1
Y
1
Z
1

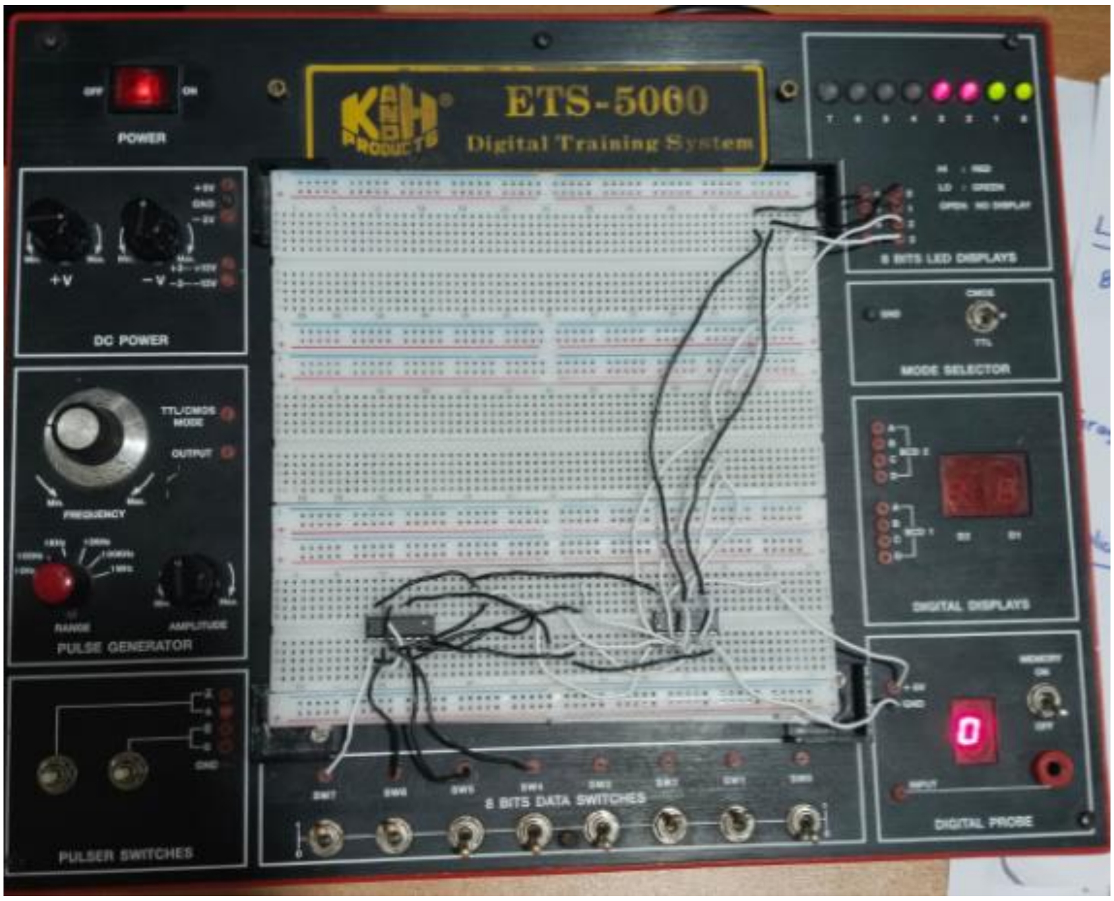

A 1
B 1
C 1
D 1
U1 XOR
U2 XOR
U3 XOR
1 W
0 X
0 Y
0 Z

A 1
B 0
C 0
D 1
U1 XOR
U2 XOR
U3 XOR
1 W
1 X
0 Y
1 Z

A 1
B 0
C 0
D 0
U1 XOR
U2 XOR
U3 XOR
1 W
1 X
0 Y
0 Z

A 1
B 0
C 1
D 1
U1 XOR
U2 XOR
U3 XOR
1 W
1 X
1 Y
0 Z

A 1
B 1
C 1
D 0
U1 XOR
U2 XOR
U3 XOR
1 W
0 X
0 Y
1 Z

A 1
B 0
C 1
D 0
U1 XOR
U2 XOR
U3 XOR
1 W
1 X
1 Y
1 Z

A 1
B 1
C 0
D 0
U1 XOR
U2 XOR
U3 XOR
1 W
0 X
1 Y
0 Z

A 1
B 1
C 0
D 1
U1 XOR
U2 XOR
U3 XOR
1 W
0 X
1 Y
1 Z

HARDWARE:



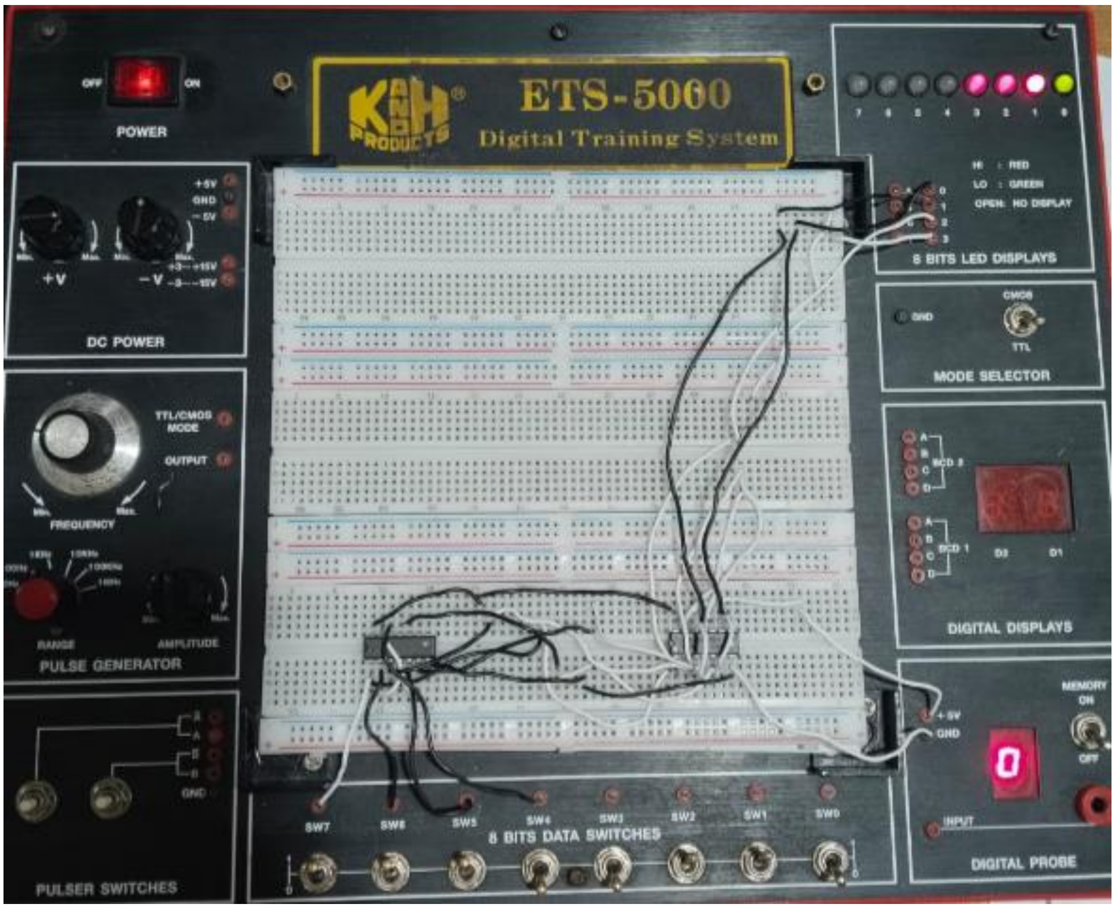
INPUT: 0000

OUTPUT: 0000



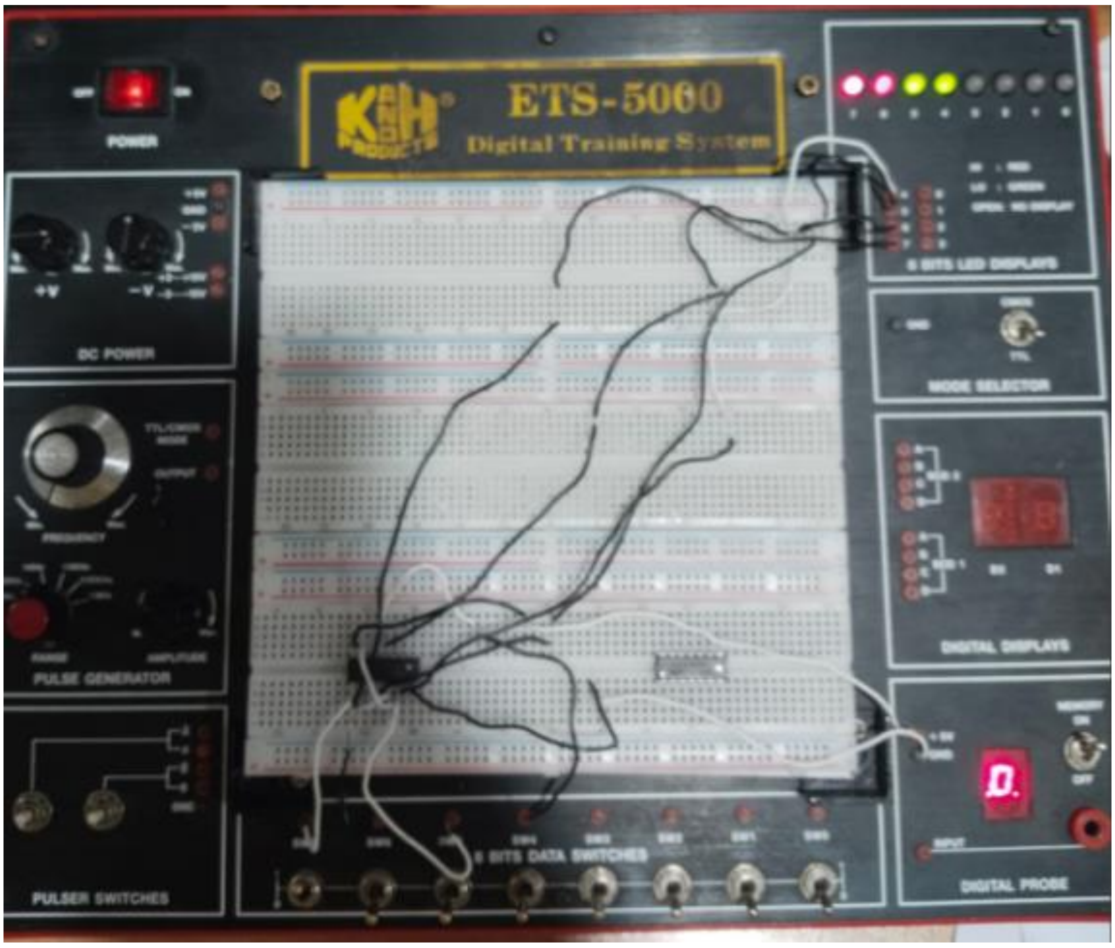
INPUT: 1100

OUTPUT: 1010



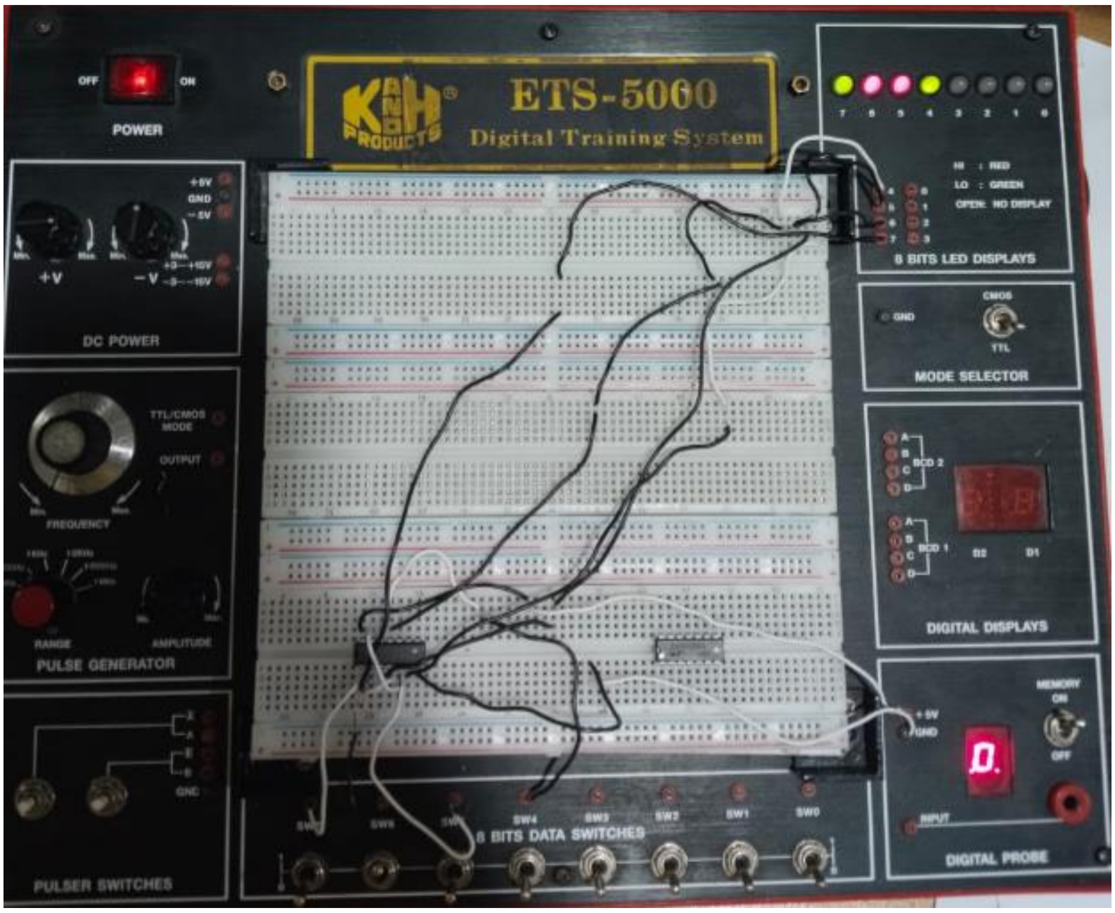
INPUT: 1110

OUTPUT: 1001



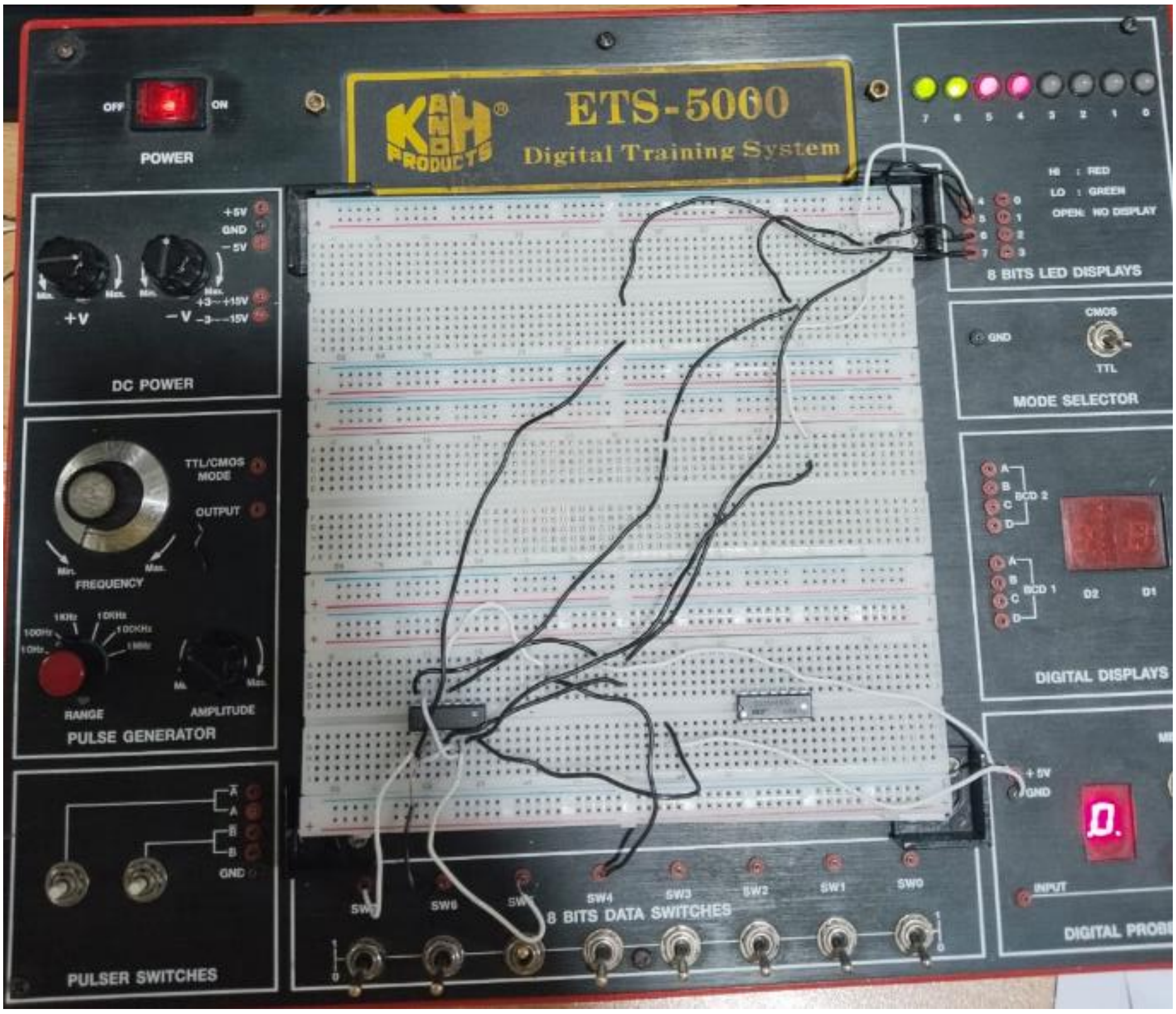
INPUT: 1000

OUTPUT: 1100



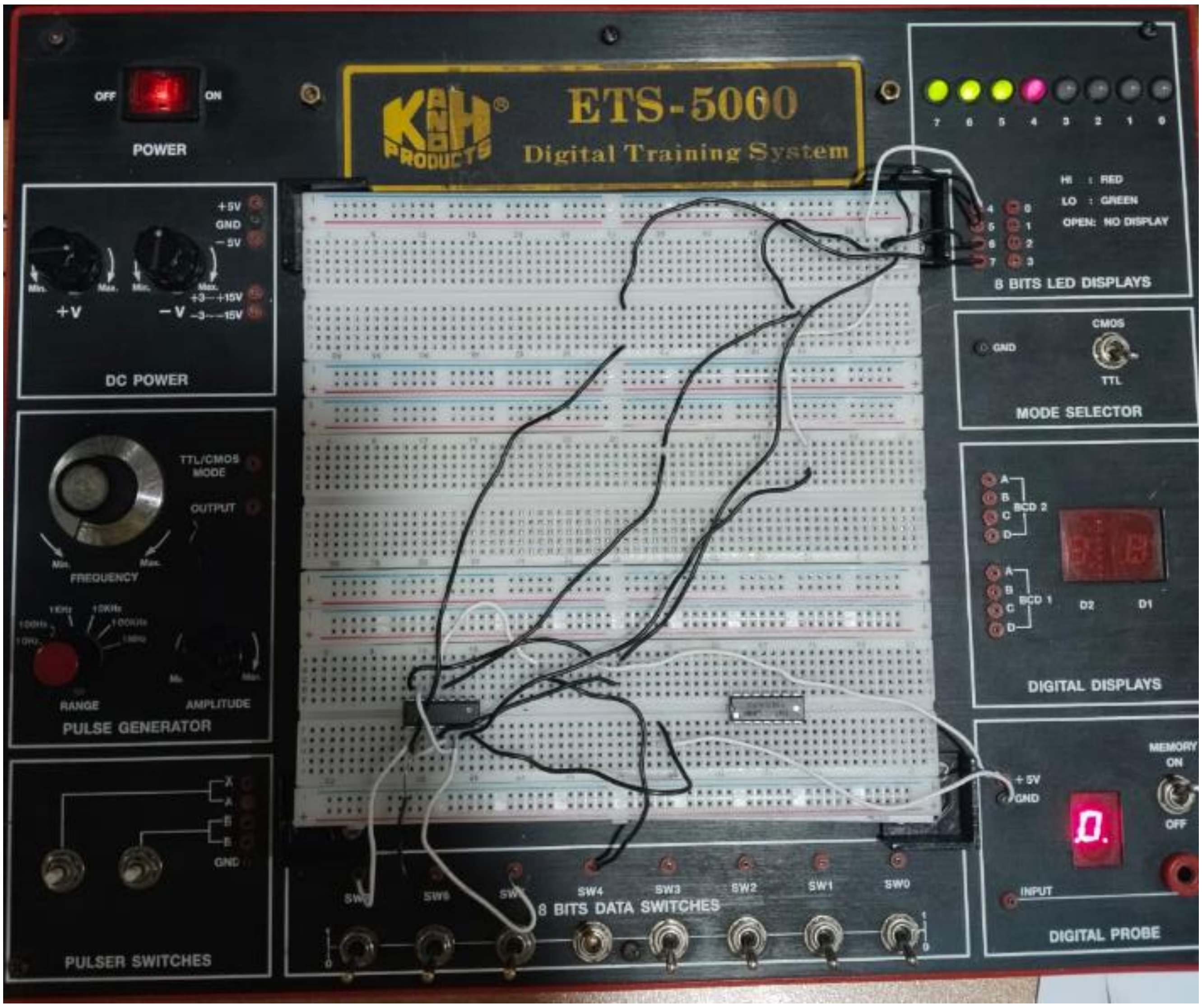
INPUT: 0100

OUTPUT: 0110



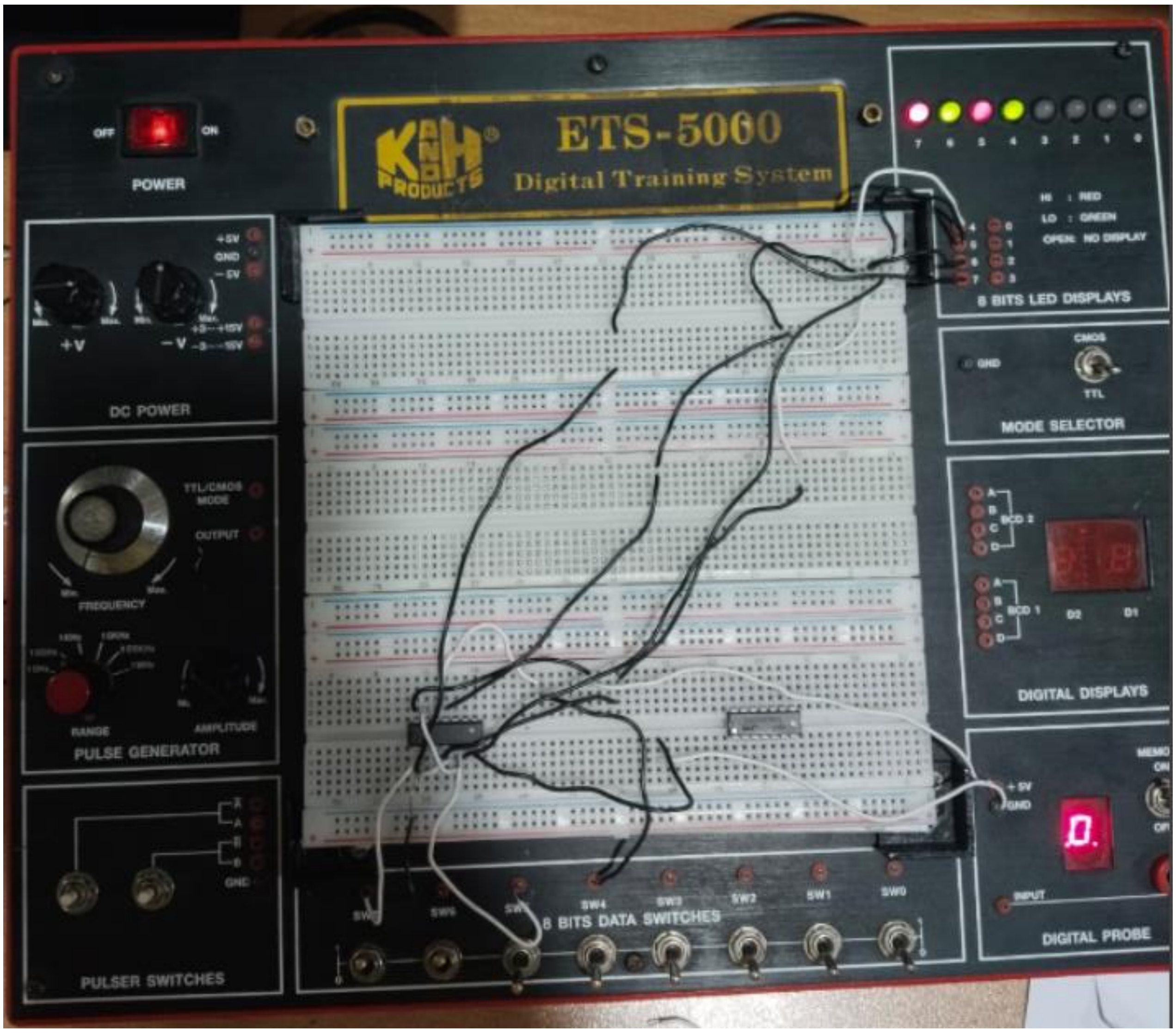
INPUT: 0010

OUTPUT: 0011



INPUT: 0001

OUTPUT: 0001



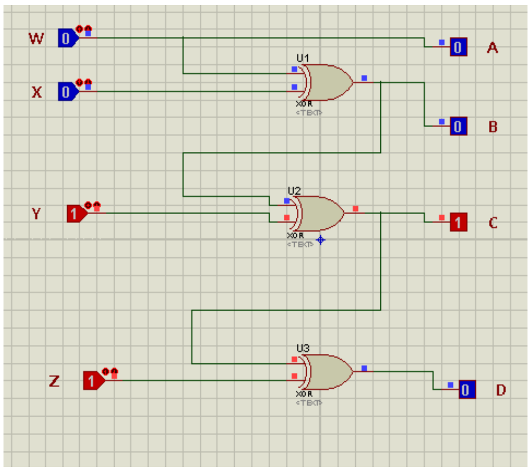
INPUT: 1100

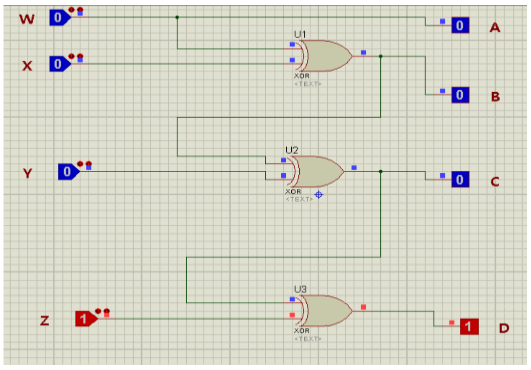
OUTPUT: 1010

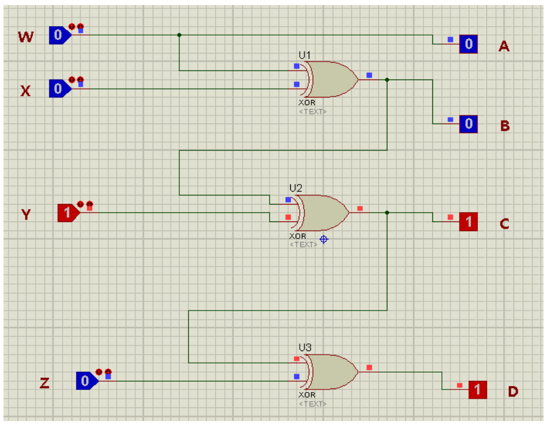
Lab Task 2: (1)

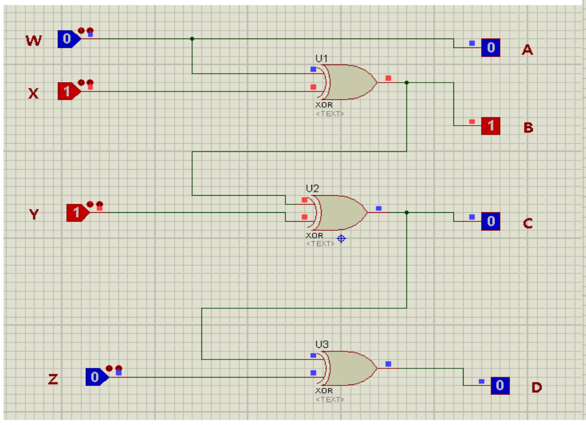
Realize the Gray to Binary Code Converter using exclusive-OR gates. Simulate in Proteus. Show the results to your Teacher/ Lab Engr. What and how many gates did you use? **Do not dispatch your hardware. You will need it in lab task 3.**

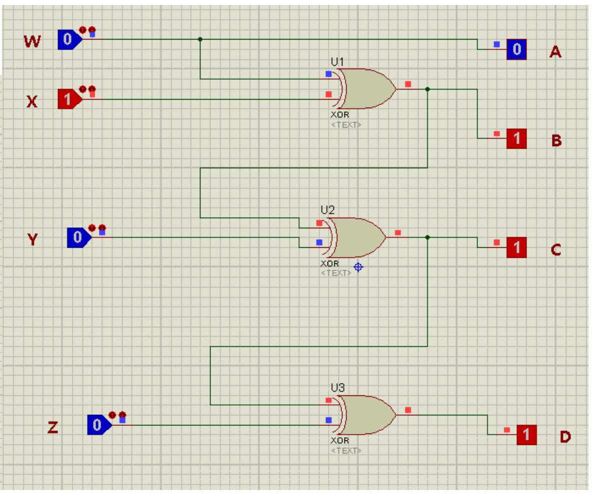
GRAY CODE TO BINARY IN PROTEUS

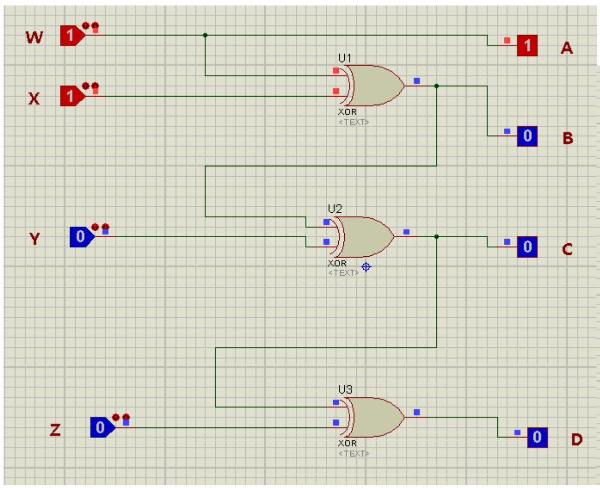


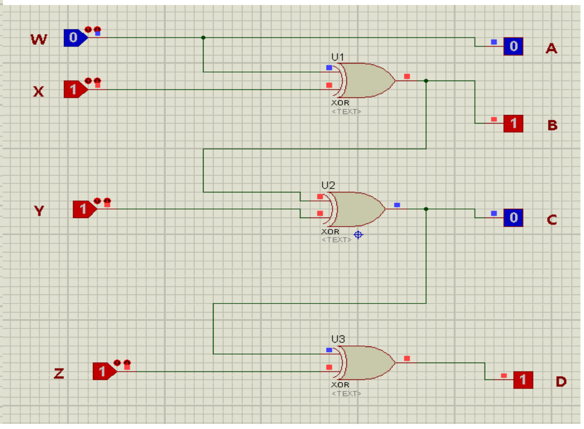


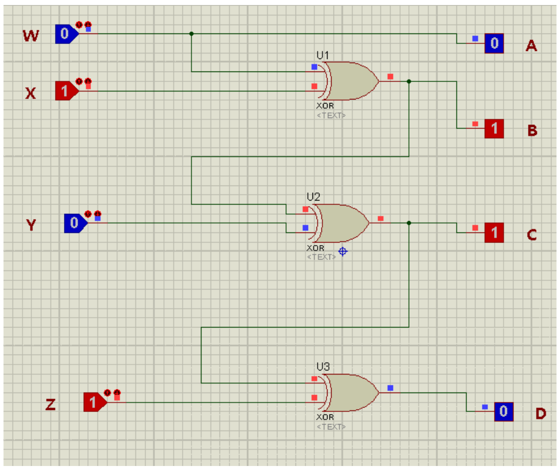


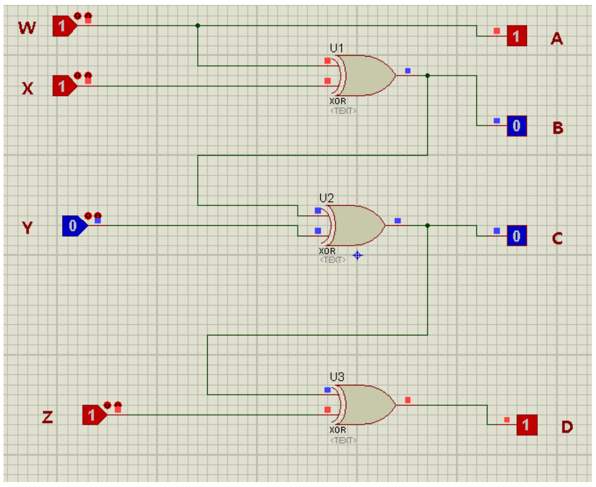

W 0
X 1
Y 1
Z 0
U1
U2
U3
XOR
<TEXT>
A 0
B 1
C 0
D 0

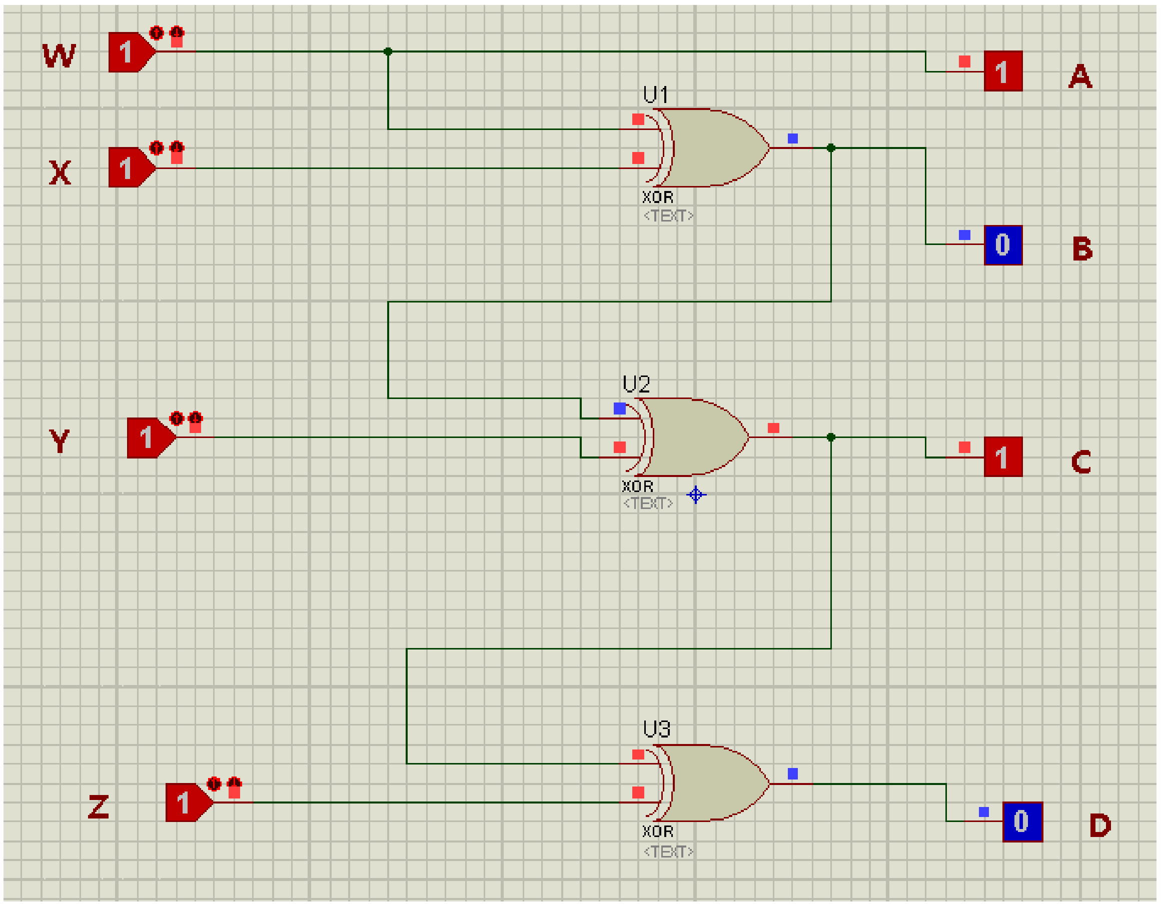

W 0
X 1
Y 0
Z 0
U1
U2
U3
XOR
<TEXT>
A 0
B 1
C 1
D 1


W 1
X 1
Y 0
Z 0
U1
U2
U3
XOR
<TEXT>
A 1
B 0
C 0
D 0

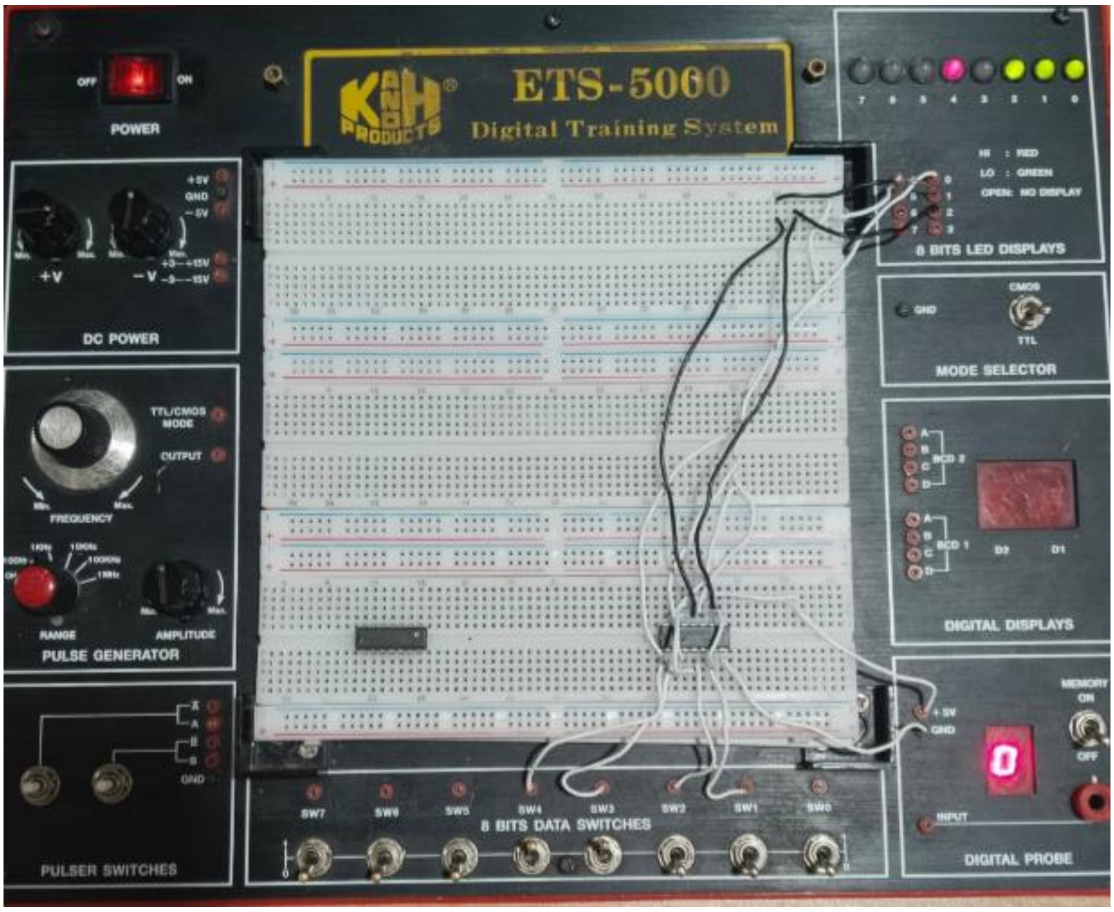

W 0
X 1
Y 1
Z 1
U1
U2
U3
XOR
<TEXT>
A 0
B 1
C 0
D 1


W 0
X 1
Y 0
Z 1
U1
U2
U3
XOR
<TEXT>
A 0
B 1
C 1
D 0


W
1
X
1
Y
0
Z
1
U1
XOR
<TEXT>
U2
XOR
<TEXT>
U3
XOR
<TEXT>
1
A
0
B
0
C
1
D

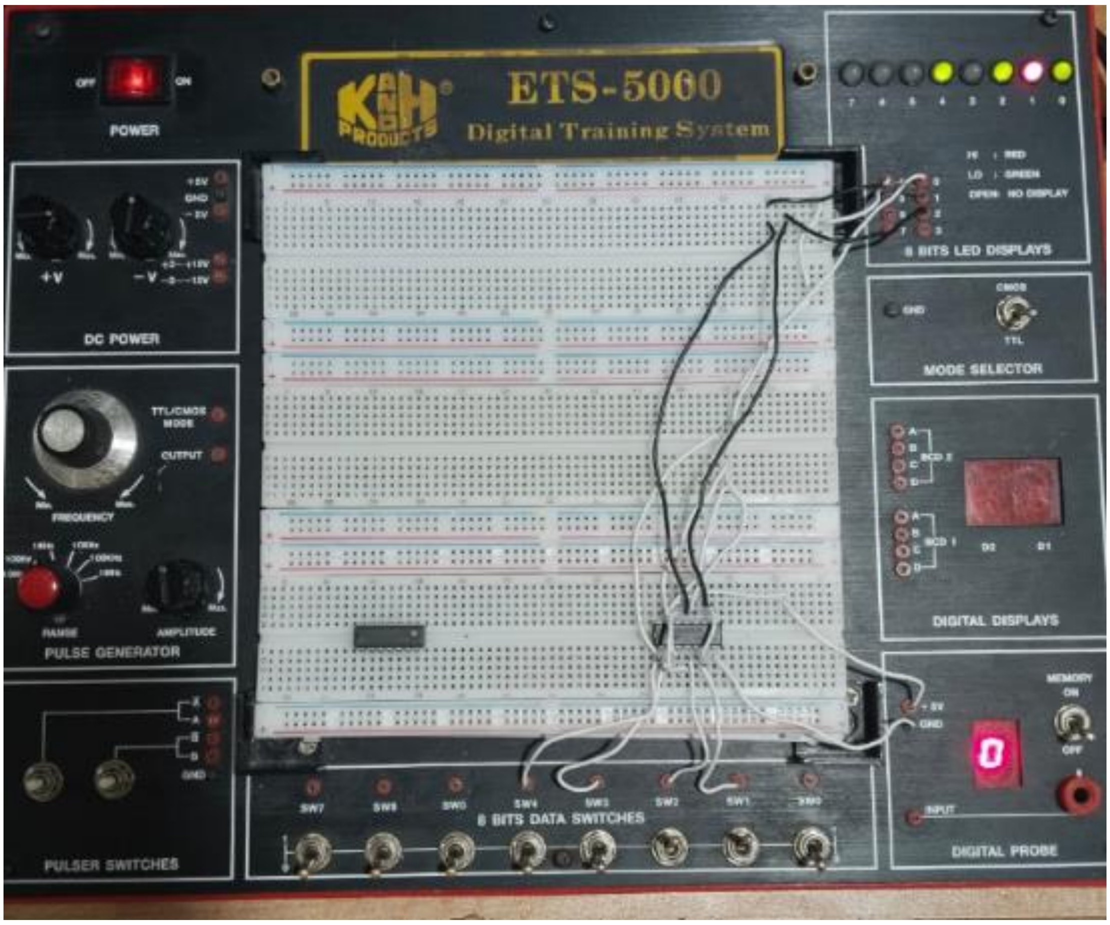


HARDWARE:



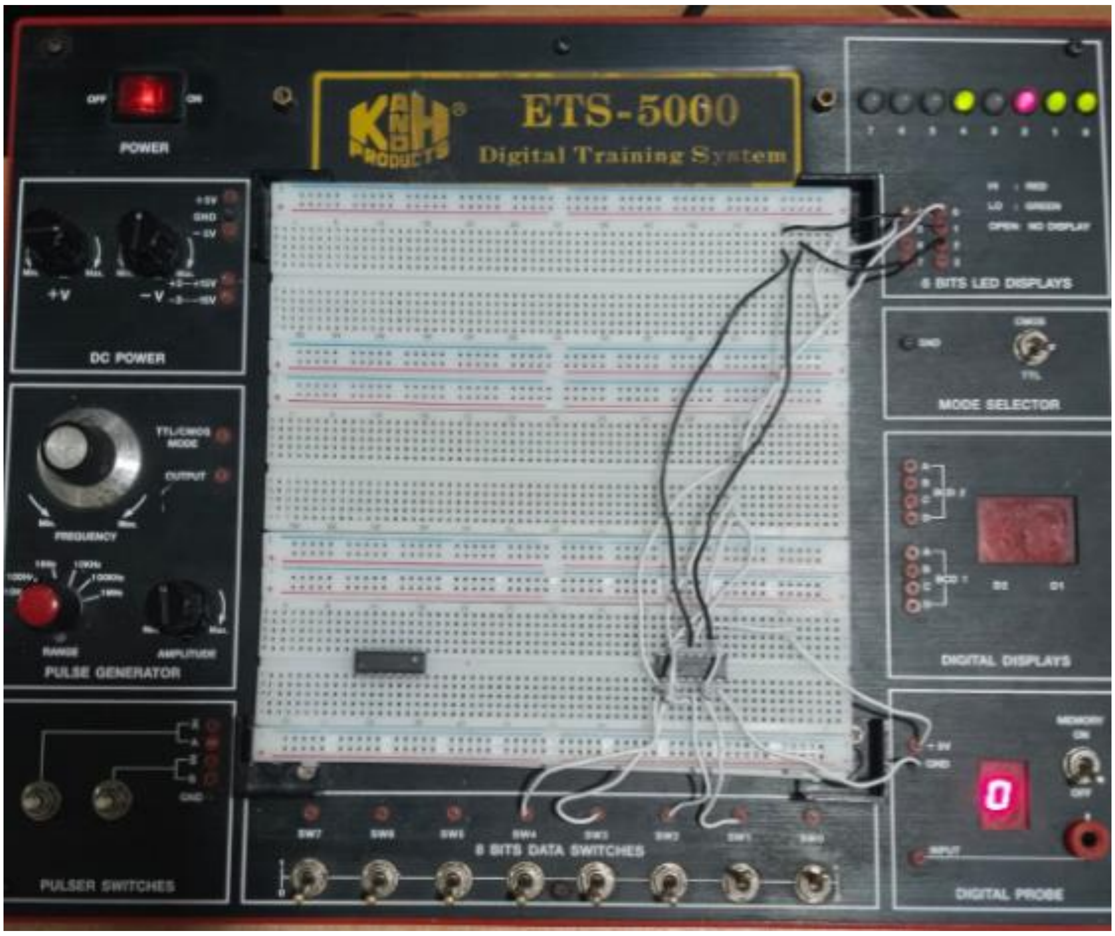
INPUT: 1100

OUTPUT: 1000



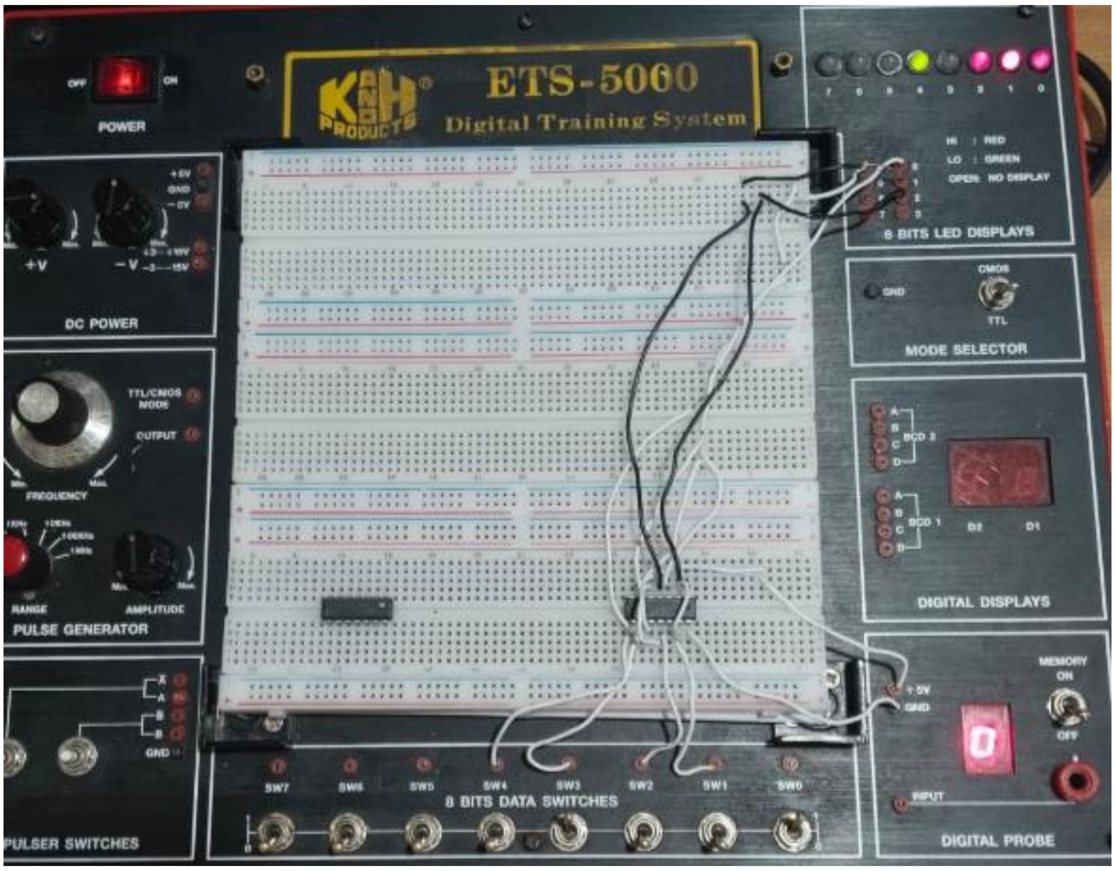
INPUT: 0011

OUTPUT: 0010



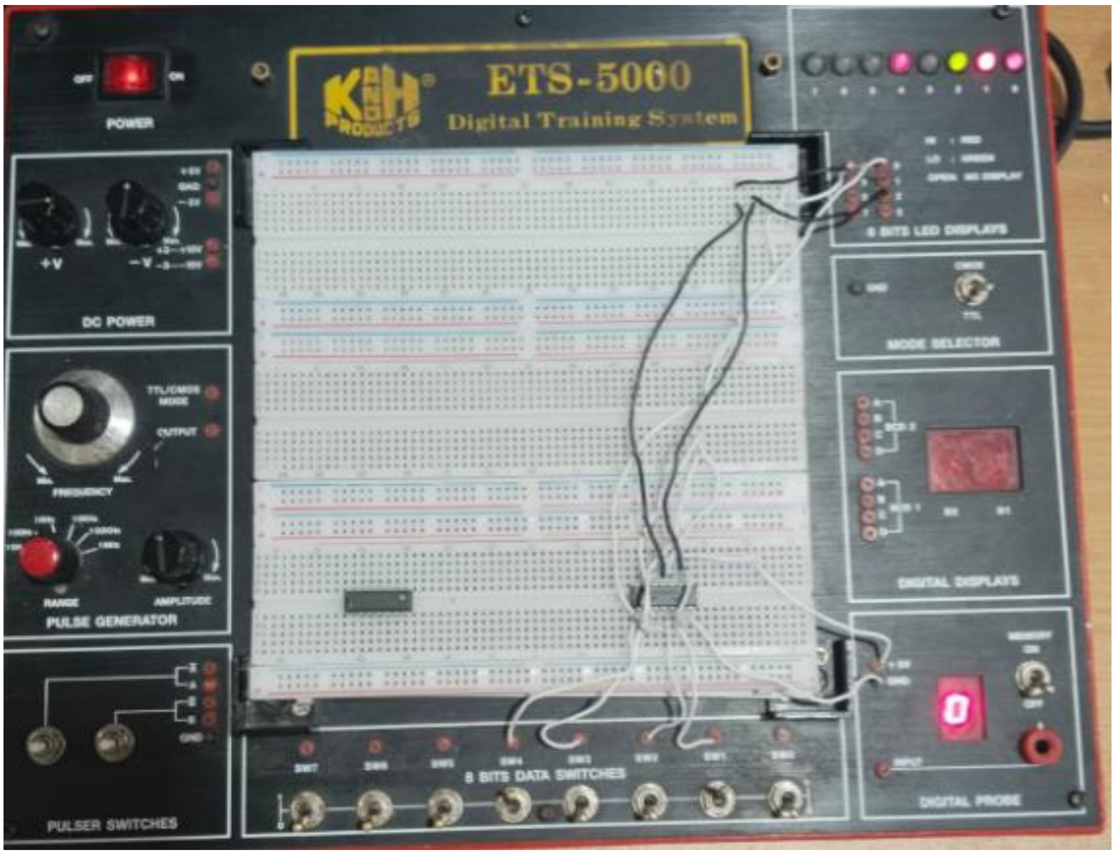
INPUT: 0001

OUTPUT: 0001



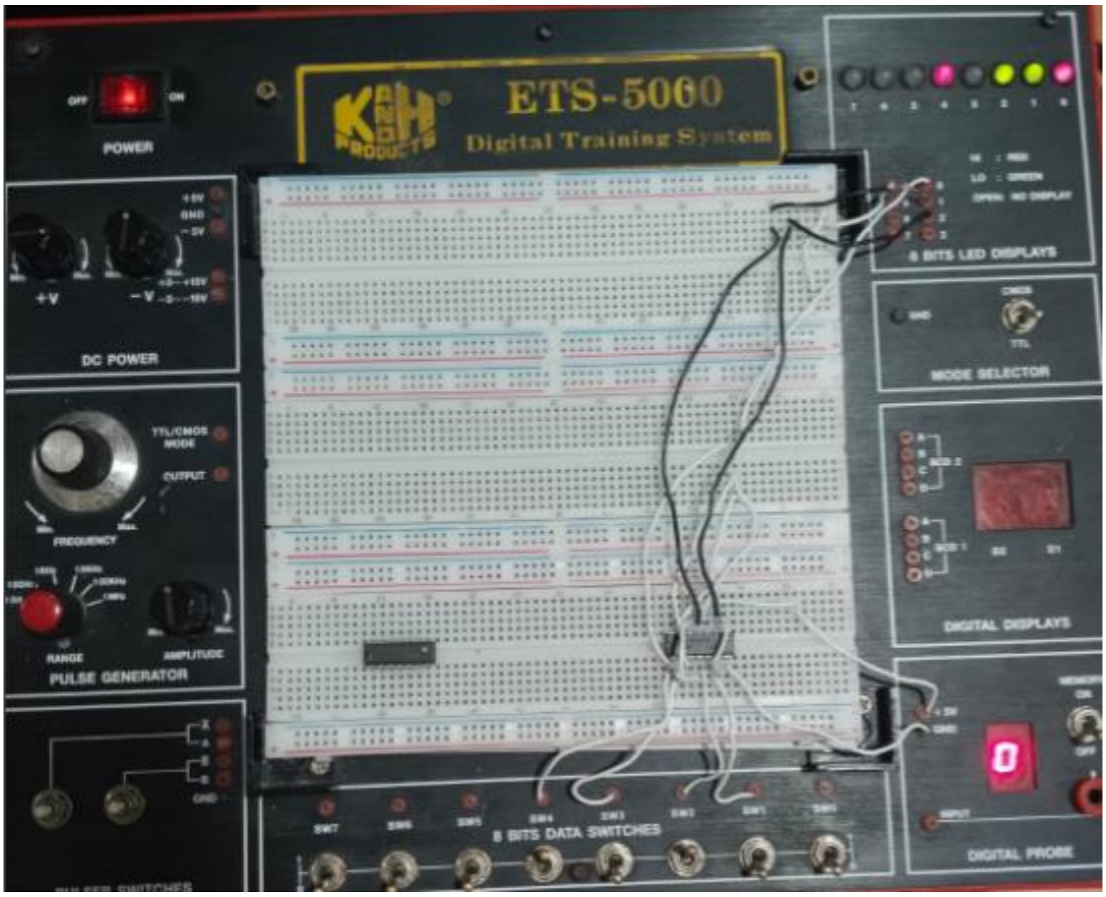
INPUT: 0100

OUTPUT: 0111



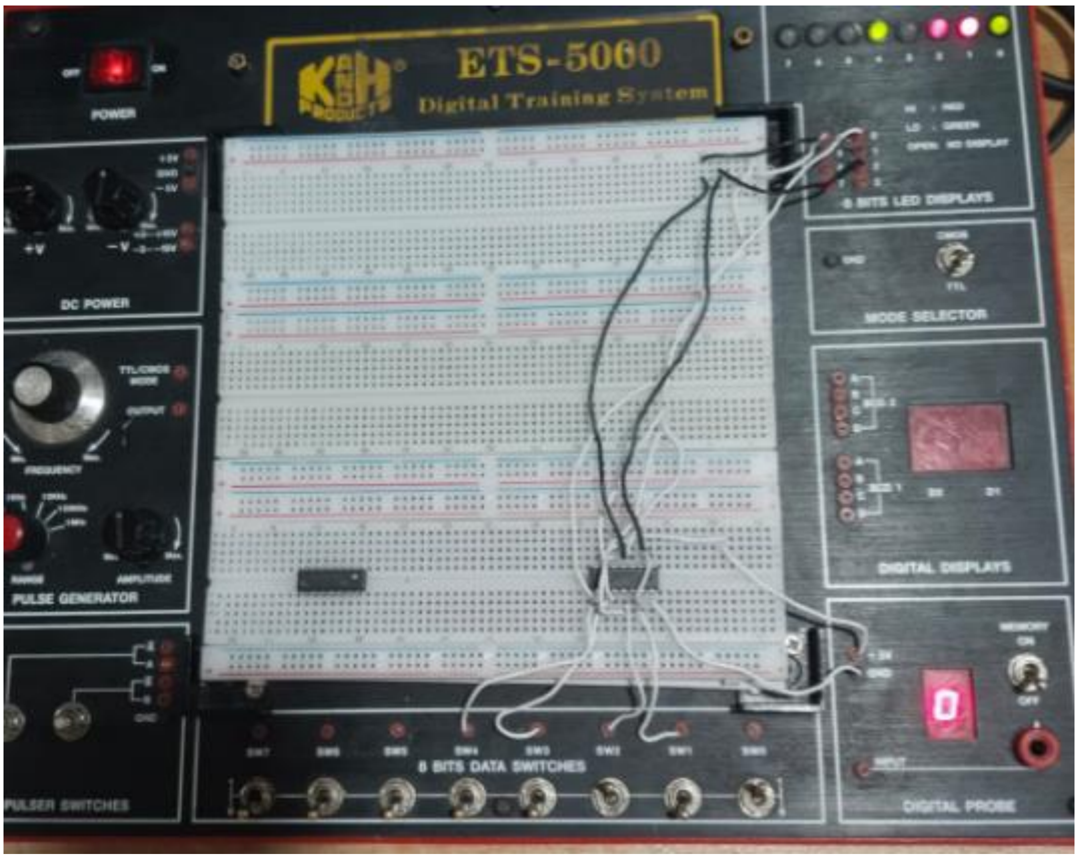
INPUT: 1001

OUTPUT: 1110



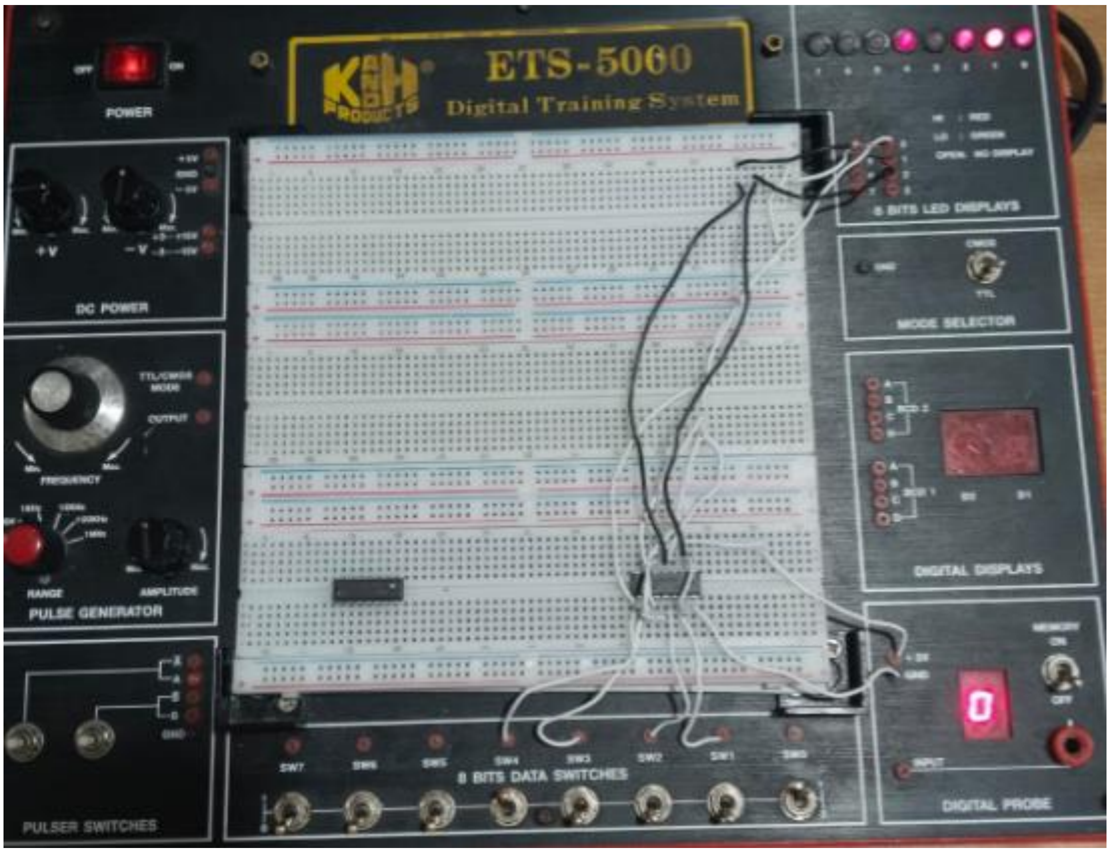
INPUT: 1010

OUTPUT: 1100



INPUT: 0010

OUTPUT: 0011



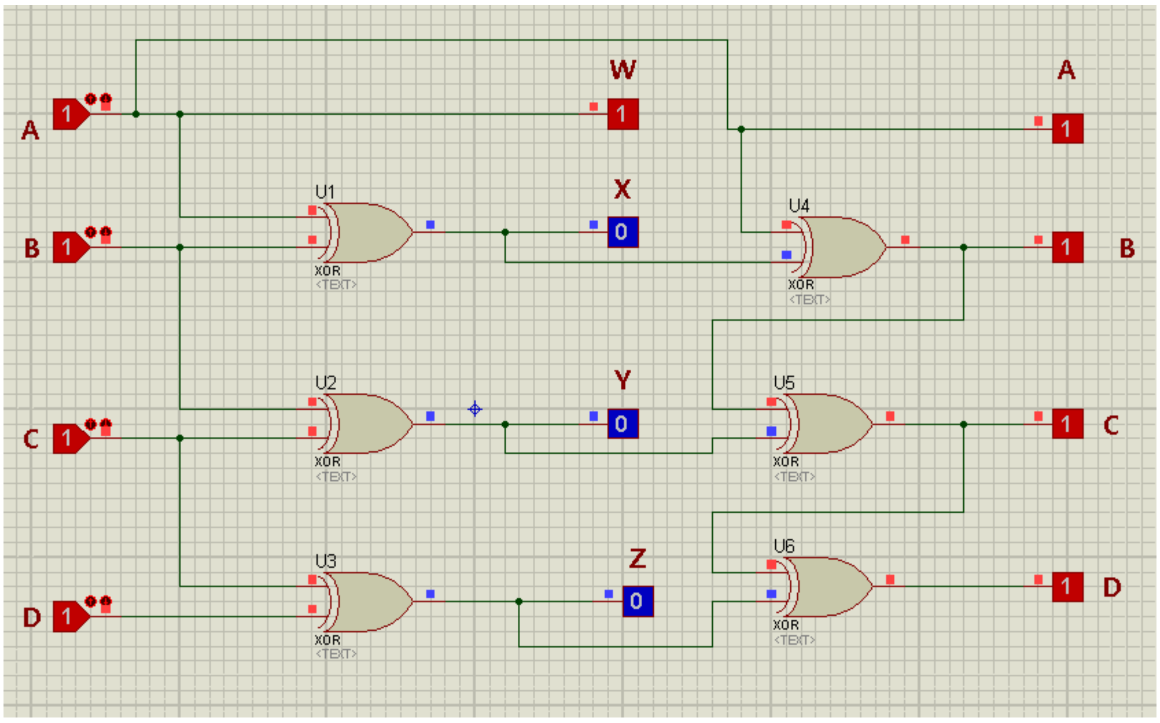
INPUT: 1000

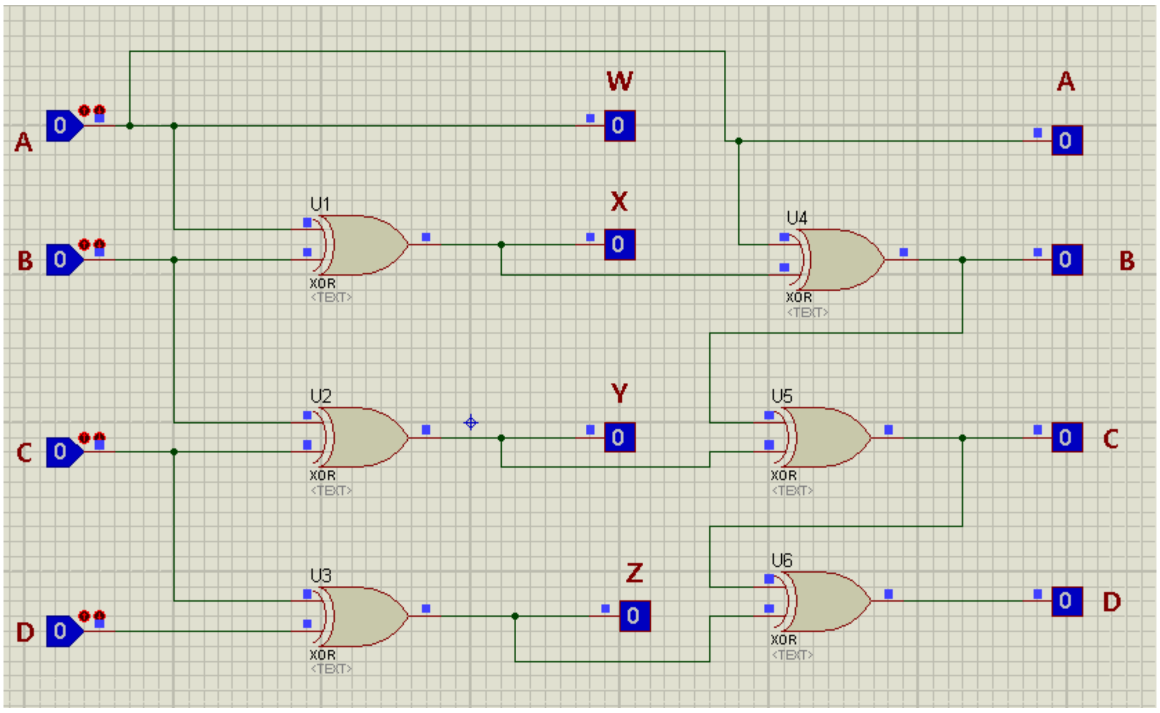
OUTPUT: 1111

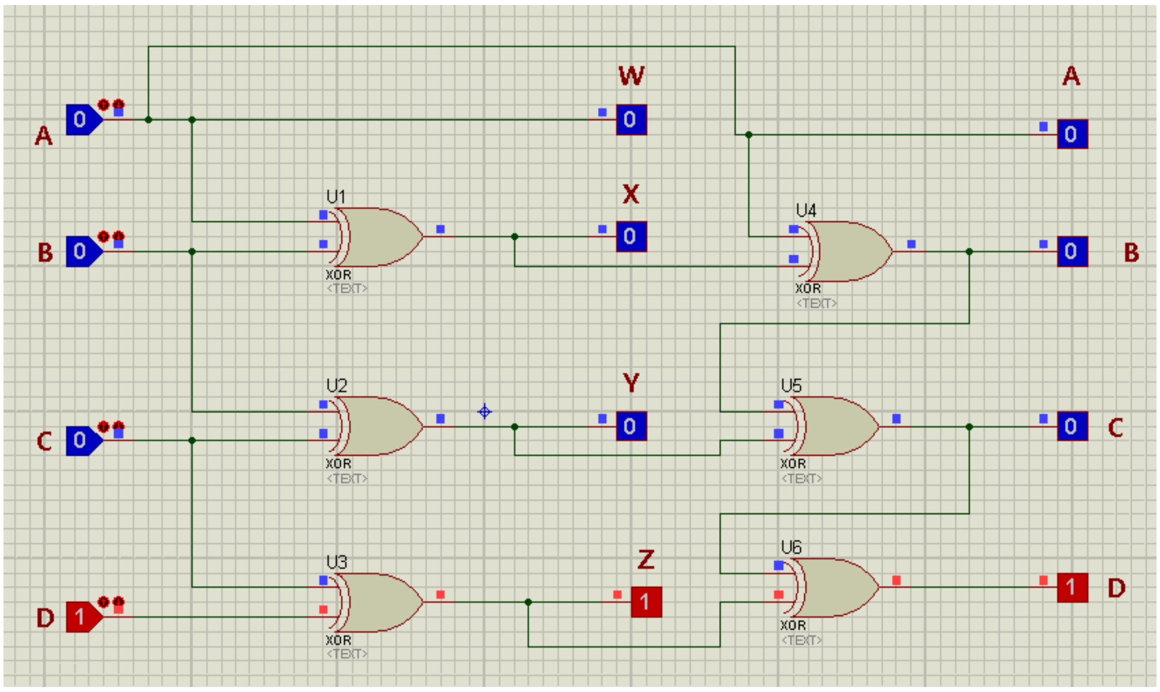
Lab Task 3: (2.5)

Now cascade the two circuits in series by connecting the outputs of binary-to-gray converter to the inputs of the gray-to-binary converter. You should be able to get the binary input at output as well. Show the results to your Teacher/Lab Engr. Use LEDs to show input-output relationship.

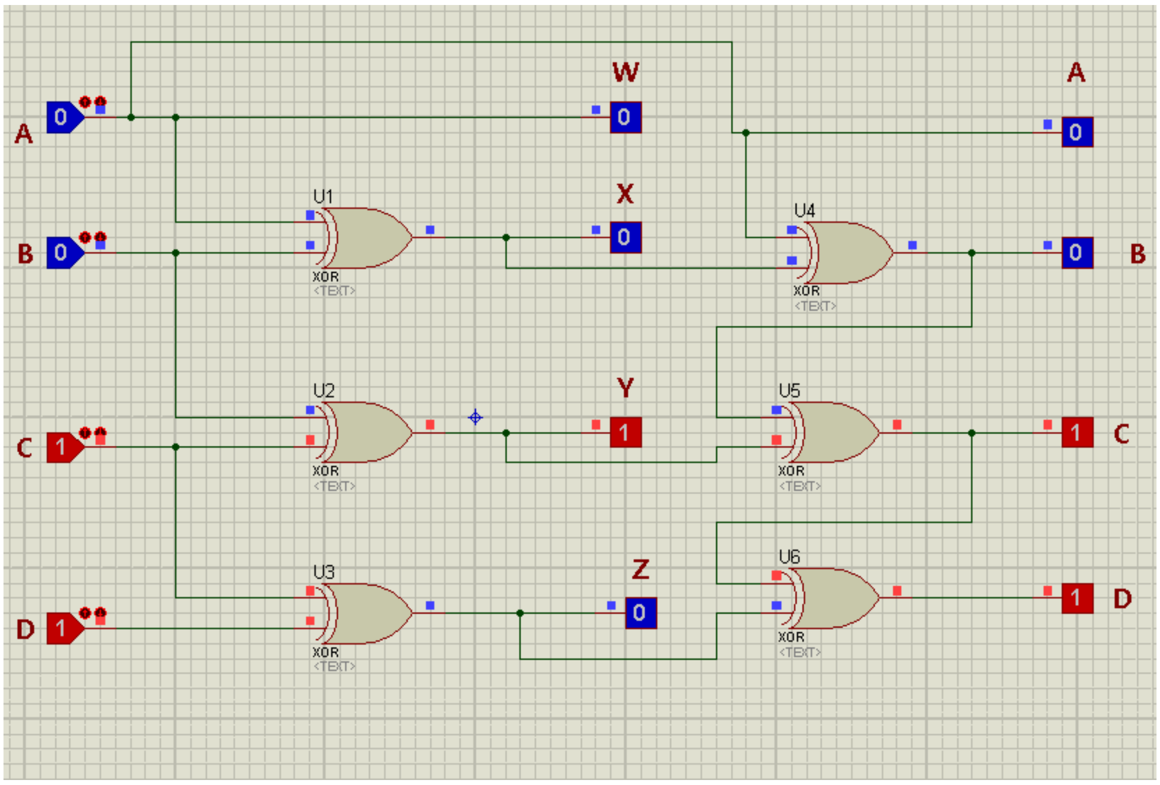
COMBINED CIRCUIT IN PROTEUS



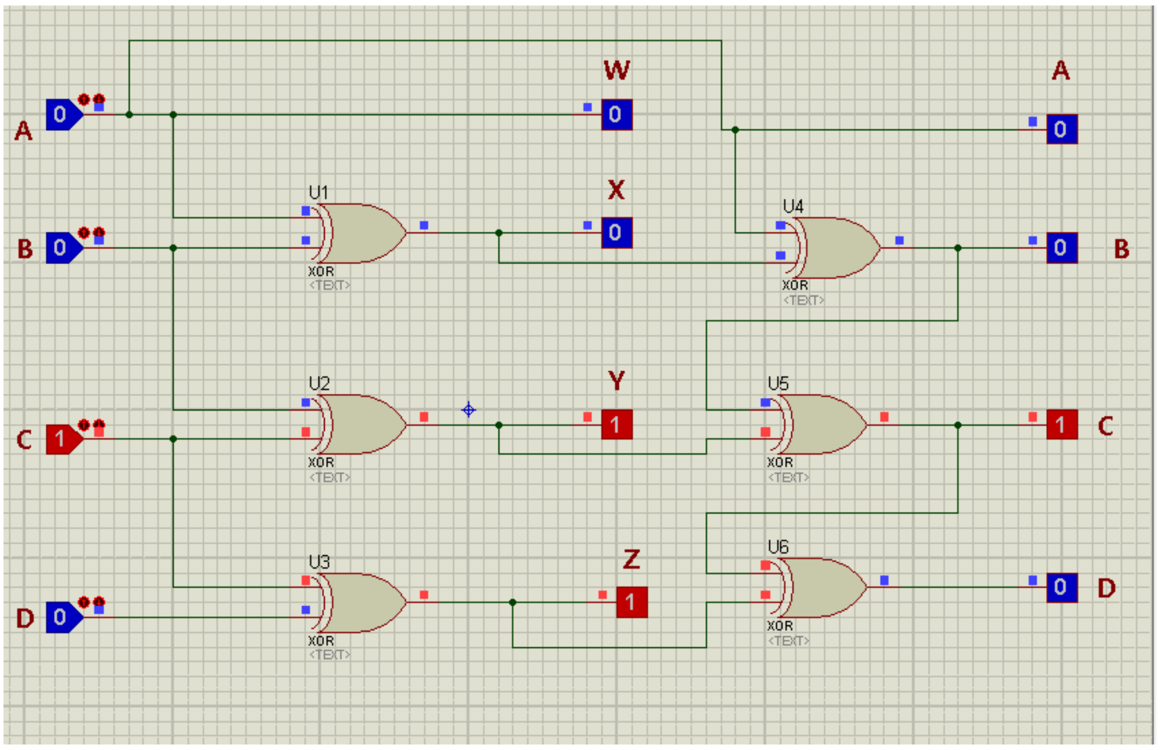


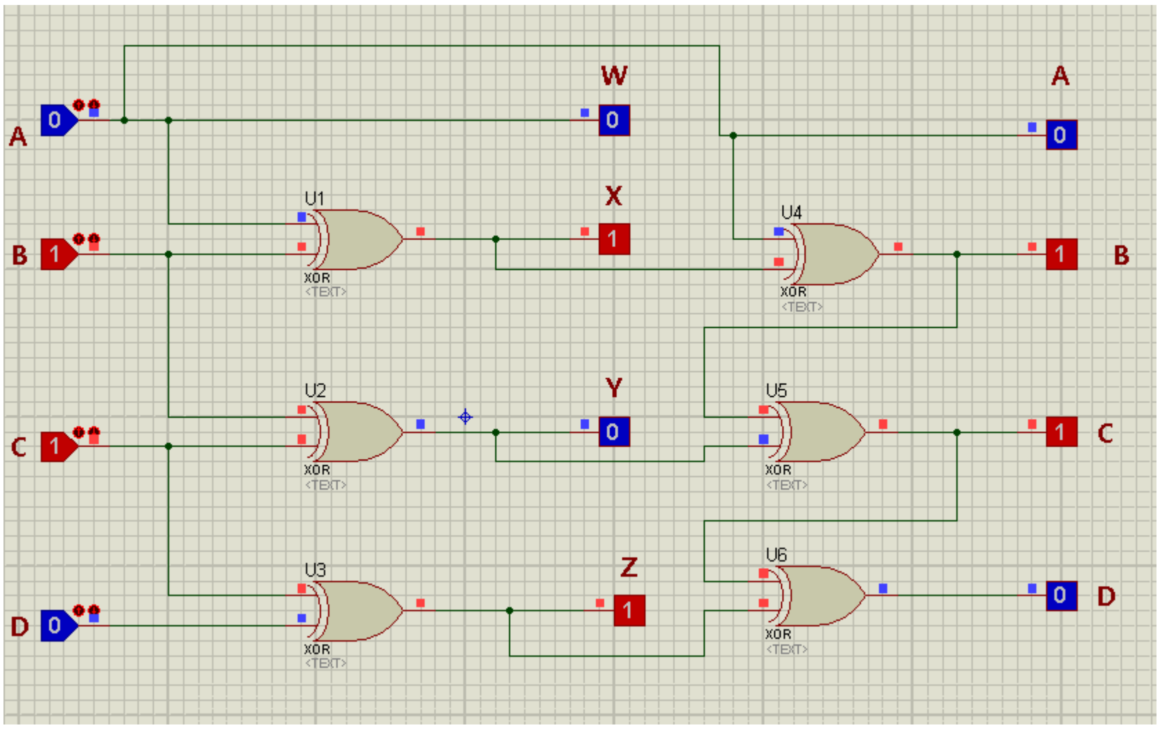


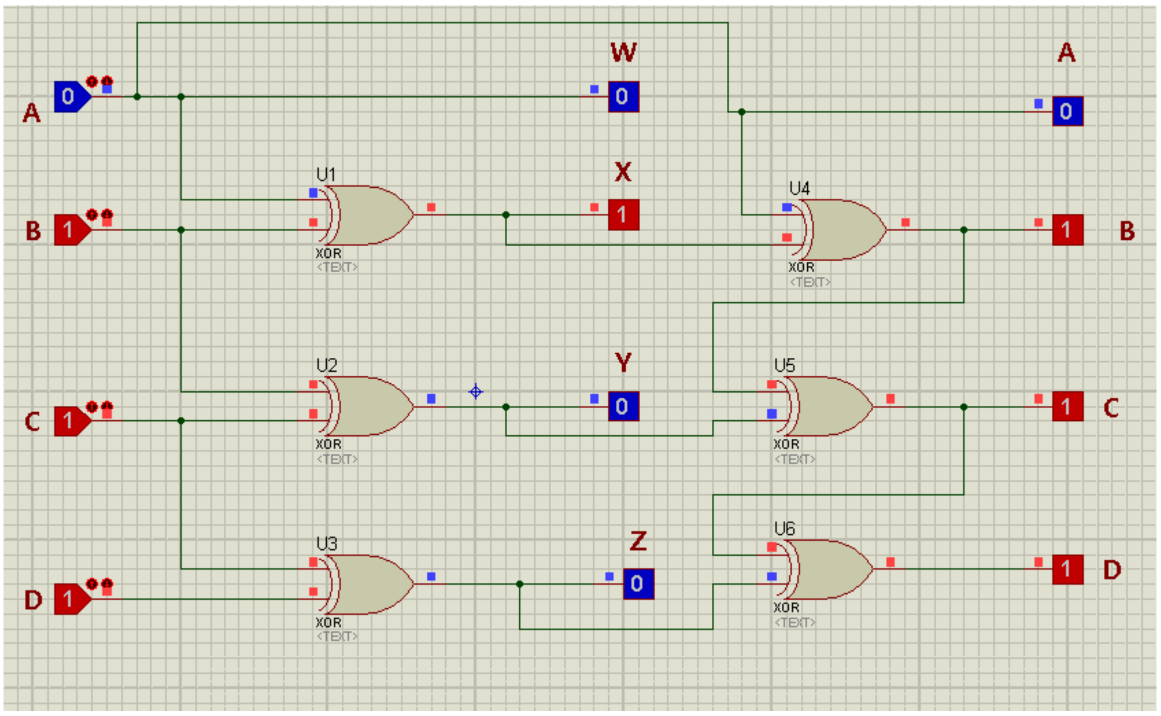
A
B
C
D
0
0
0
1
W
X
Y
Z
0
0
0
1
U1
U2
U3
U4
U5
U6
XOR
<TEXT>
A
B
C
D
0
0
0
1

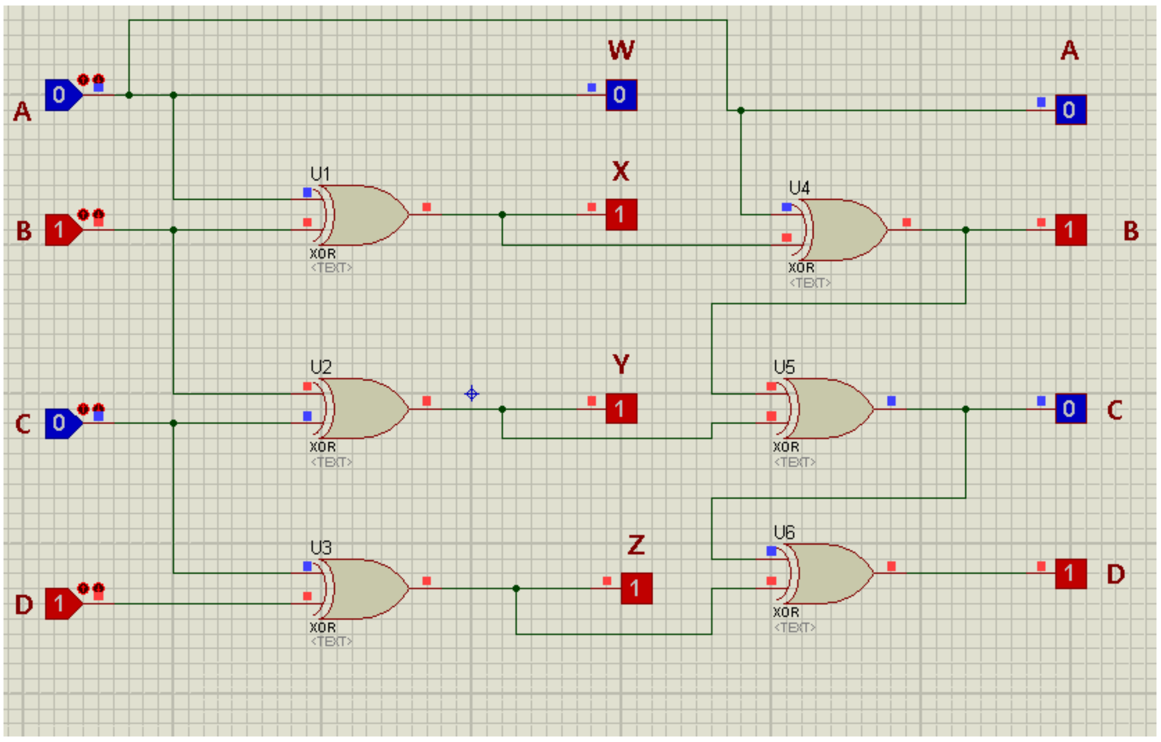


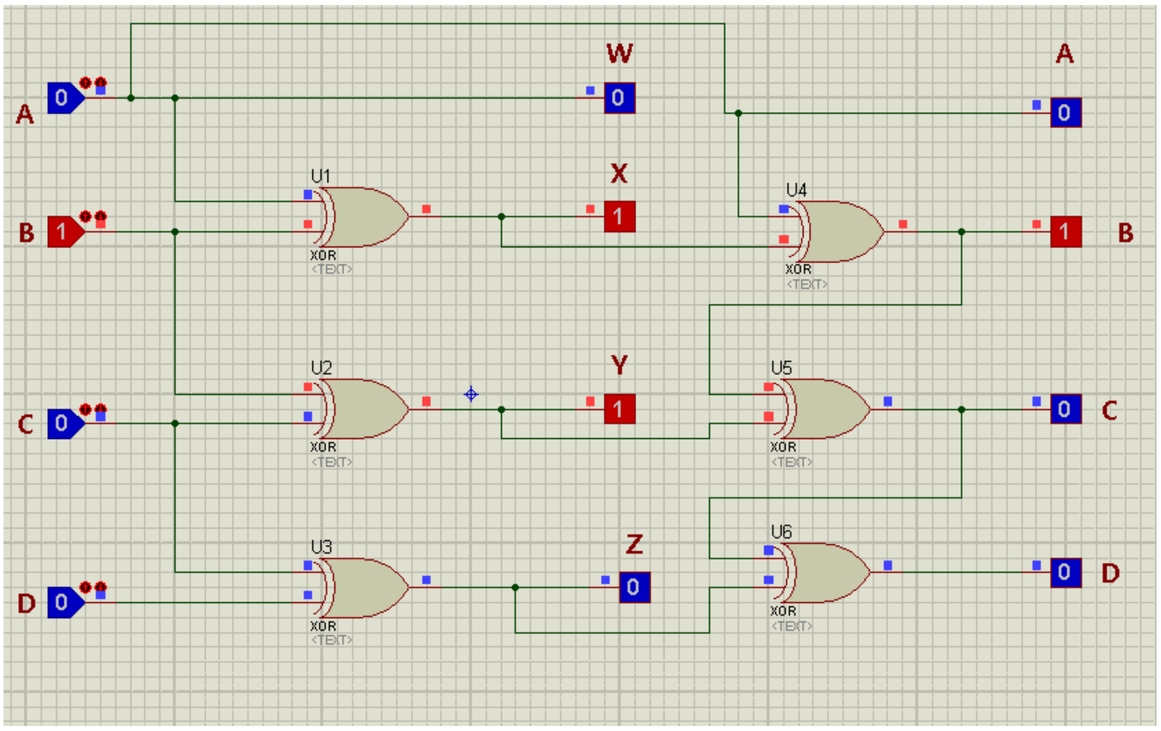
A
B
C
D
0
0
1
1
W
X
Y
Z
0
0
1
0
U1
U2
U3
U4
U5
U6
XOR
<TEXT>
A
B
C
D
0
0
1
1


A
B
C
D
0
0
1
0
W
X
Y
Z
0
0
1
1
U1
U2
U3
U4
U5
U6
XOR
<TEXT>
A
B
C
D
0
0
1
0


A
B
C
D
0
1
1
0
W
X
Y
Z
0
1
0
1
U1
U2
U3
U4
U5
U6
XOR
<TEXT>
A
B
C
D
0
1
1
0


A
0
B
1
C
1
D
1
W
0
X
1
Y
0
Z
0
U1
U2
U3
U4
U5
U6
XOR
A
0
B
1
C
1
D
1


A
0
B
1
C
0
D
1
W
0
X
1
Y
1
Z
1
U1
U2
U3
U4
U5
U6
XOR
A
0
B
1
C
0
D
1



Lab Task 4: (2.5)

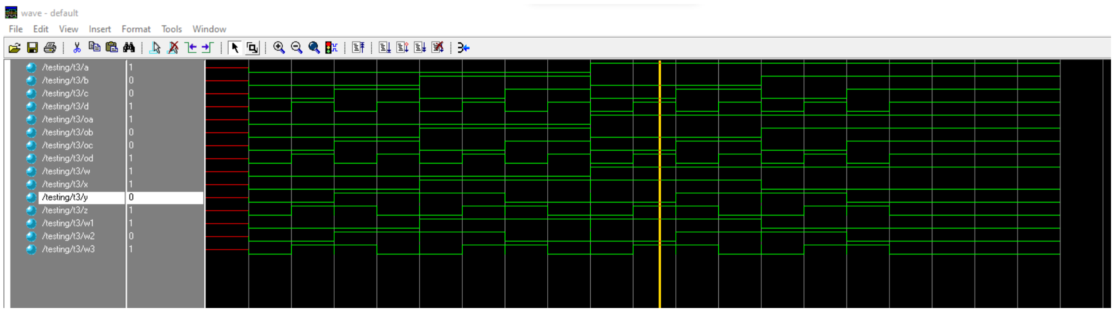
Design and simulate the gate-level model of the circuit you patched in **task 3**. Give the code in the space provided below.

CODE:

```
module task3(oa, ob, oc, od, w, x, y, z, a, b, c, d);
input a, b, c, d;
output oa, ob, oc, od, w, x, y, z;
wire w1, w2, w3;

   buf b1(w, a);
   buf b2(oa, a);

   xor x1(w1, a, b);
   buf b3(x, w1);
```

```
    xor x2(w2, b, c);
    buf b4(y, w2);

    xor x3(w3, c, d);
    buf b5(z, w3);

    xor x4(ob, a, w1);
    xor x5(oc, ob, w2);
    xor x6(od, oc, w3);

endmodule

module testing();
    reg A, B, C, D;
    wire OA, OB, OC, OD, W, X, Y, Z;

    task3 t3(OA, OB, OC, OD, W, X, Y, Z, A, B, C, D);

    initial begin
        #100 A = 0; B = 0; C = 0; D = 0;
        #100 A = 0; B = 0; C = 0; D = 1;
        #100 A = 0; B = 0; C = 1; D = 0;
        #100 A = 0; B = 0; C = 1; D = 1;
        #100 A = 0; B = 1; C = 0; D = 0;
        #100 A = 0; B = 1; C = 0; D = 1;
        #100 A = 0; B = 1; C = 1; D = 0;
        #100 A = 0; B = 1; C = 1; D = 1;
        #100 A = 1; B = 0; C = 0; D = 0;
        #100 A = 1; B = 0; C = 0; D = 1;
        #100 A = 1; B = 0; C = 1; D = 0;
        #100 A = 1; B = 0; C = 1; D = 1;
        #100 A = 1; B = 1; C = 0; D = 0;
        #100 A = 1; B = 1; C = 0; D = 1;
        #100 A = 1; B = 1; C = 1; D = 0;
        #100 A = 1; B = 1; C = 1; D = 1;
    end
endmodule
```

WAVE FORM:



```
/testing/t3/a   1
/testing/t3/b   0
/testing/t3/c   0
/testing/t3/d   1
/testing/t3/oa  1
/testing/t3/ob  0
/testing/t3/oc  0
/testing/t3/od  1
/testing/t3/w   1
/testing/t3/x   1
/testing/t3/y   0
/testing/t3/z   1
/testing/t3/w1  1
/testing/t3/w2  0
/testing/t3/w3  1
```

Observations/Comments:

In this lab we implemented the binary to gray and gray to binary circuit using:

- Basic gates (AND, NOT and OR)
- Using XOR.

We identified the outputs of the converters using LEDs and wrote the Verilog code for the circuit which converted binary to gray to binary.