Project Report

Department of Computer Science & Engineering

CSE345
Digital Logic Design
Section: 05
Spring 2024

Project No.: 02
Project Title: BCD to Excess-3 Code Generator
Group No: 06

Group Members:

1. Tamanna-E-Jahan [2020-1-60-151]
2. Ratry [2020-3-60-002]
3. klinton Biswas [2021-2-60-188]
4. Tanzila Afrin [2022-1-60-045]

Course Instructor:

Musharrat Khan
Senior Lecturer
Department of Computer Science & Engineering
East West University

Date of Submission: 30.05.2024

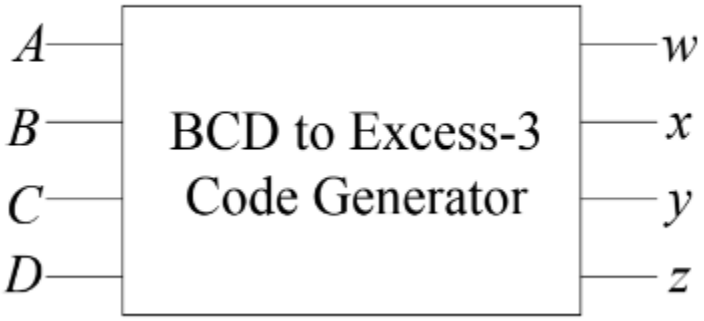
Problem Statement:

In this problem a BCD to Excess-3 Code Generator. BCD is a 4-bit code for 10 digits. For example, 0,1,2 is represented by 0000,0001,0010, etc. The output is 3 greater than the input. For example, if the input is 0000, then the output is 0011.

Design Details:

To construct this circuit, we need to convert a 4-bit BCD input to a 4-bit Excess-3 output. The Excess-3 code is derived by adding 3 (0011 in binary) to the BCD input. First, we need to create a truth table that maps each BCD input to its corresponding Excess-3 output. Then for each output bit (W, X, Y, Z), will create Karnaugh Maps or shortly **K-maps** to simplify the Boolean expressions. Using the simplified Boolean expressions, we will implement the circuit using basic logic gates such as AND, OR, and NOT. Lastly, we will do the simulation and write two types of behavioral Verilog code (procedural model and continuous assign statement) for the logic diagram and check simulation. In the end, we will see that the results of logic diagram simulation, two types of behavioral Verilog code simulation, and the truth table output are the same. This will complete the project.

Block Diagram:



Truth Table:

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

K-map for X:

CD / AB	00	01	11	10
00		1	1	1
01	1			
11	X	X	X	X
10		1	X	X

Boolean Expression:

X= B′C+B′D+BC′D′

K-map for Y:

CD / AB	00	01	11	10
00	1		1	
01	1		1	
11	X	X	X	X
1O	1		X	X

Boolean Expression:

Y= C′D′+CD

= (C⊕D)′

K-map for Z:

AB \ CD	00	01	11	10
00	1			1
01	1			1
11	X	X	X	X
10	1		X	X

Boolean Expression:

Z=D′

K-map for W:

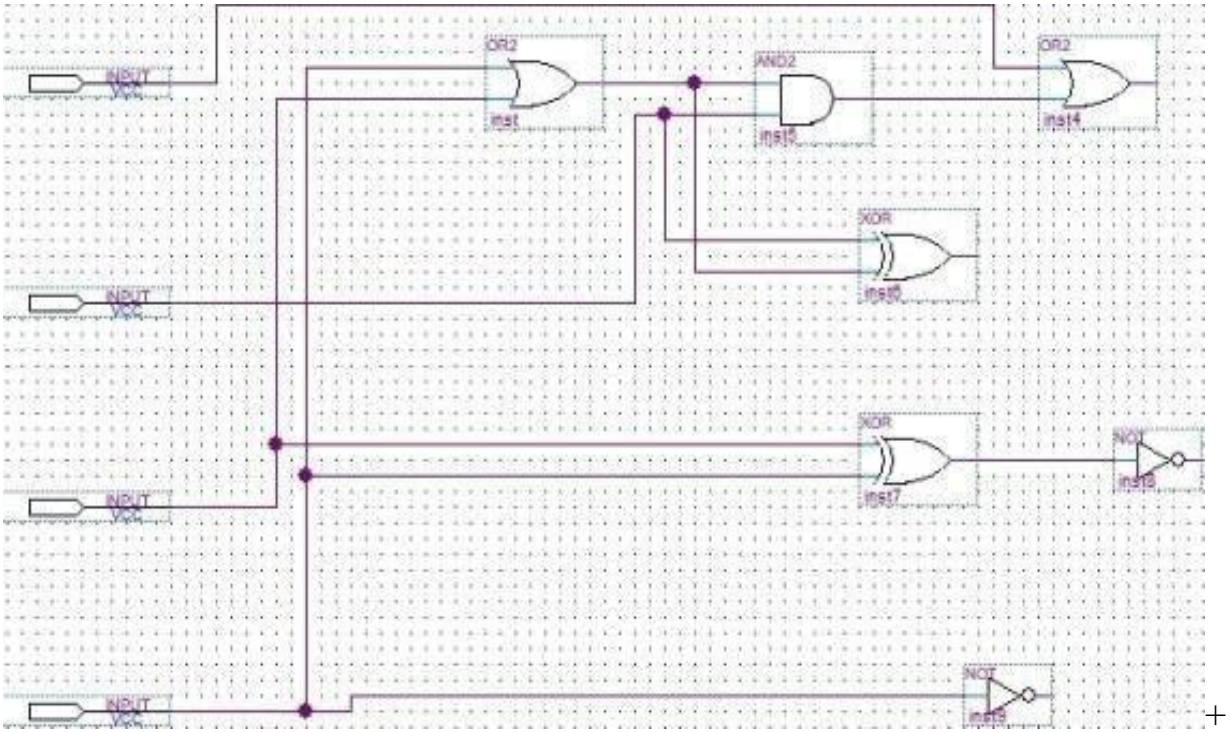
AB \ CD	00	01	11	10
00				
01		1	1	1
11	X	X	X	X
10	1	1	X	X

Boolean Expression:

W= A+BC+BD

=A+B(C+D)

Circuit Diagram:



Behavioral Verilog Code (Procedural Model):

```
module labfinal ( input wire A, B,
C, D, output reg W, X, Y, Z

);
always @(*) begin  case ({A, B, C,
D})
4'b0000: {W, X, Y, Z} = 4'b00l l; // 0 + 3 = 3
```

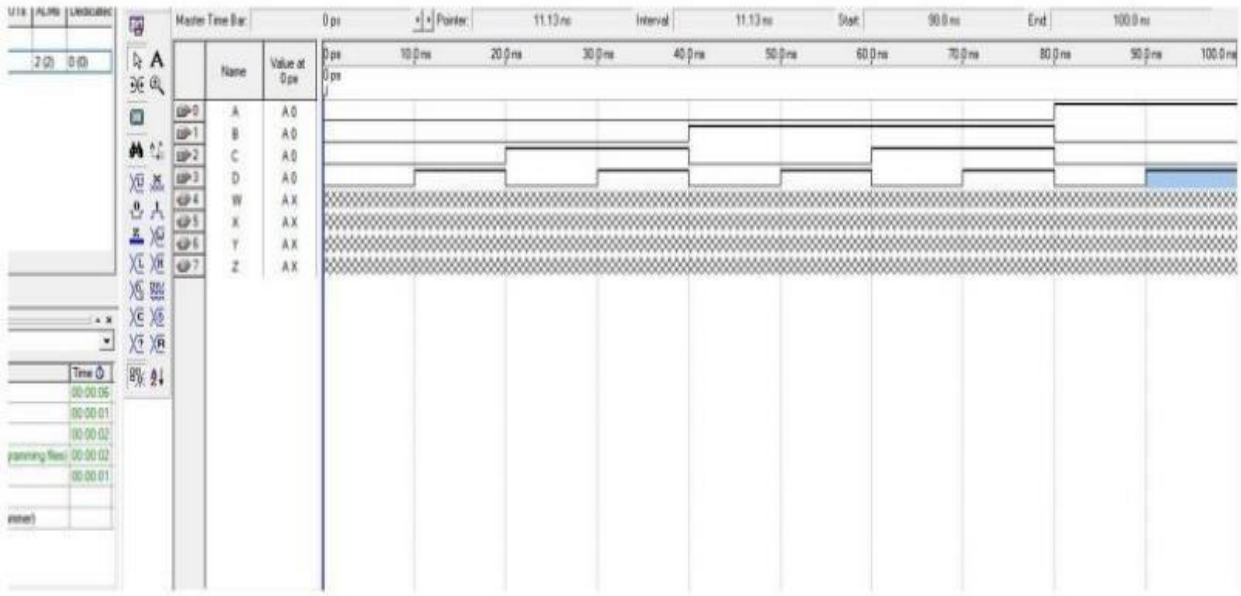
```
4'b000l: {W, X, Y, Z} = 4'b0100; // l + 3 = 4

4'b0010: {W, X, Y, Z} = 4'b010l; // 2 + 3 = 5 4'b0011: {W, X, Y, Z} = 4'b0
l0; // 3 + 3 = 6

4'b0100: {W, X, Y, Z} = 4'b0l l1; // 4 + 3 = 7

4'b0l0l: {W, X, Y, Z} = 4'bl000; // 5 + 3 = 8

4'b0l10: {W, X, Y, Z} = 4'bl00l; // 6 + 3 = 9

4'b0111: {W, X, Y, Z} = 4'b10l0; // 7 + 3 = l0

4'b1000: {W, X, Y, Z} = 4'bl0l l; // S + 3 = l l 4'bl00l: {W, X, Y, Z} =
4'b1100; // 9 + 3 = l2 default: {W, X, Y, Z} = 4'XXXX; // Invalid BCD
input endcase end
endmodule
```

Behavioral Verilog Code (Continuous Assign Statement):

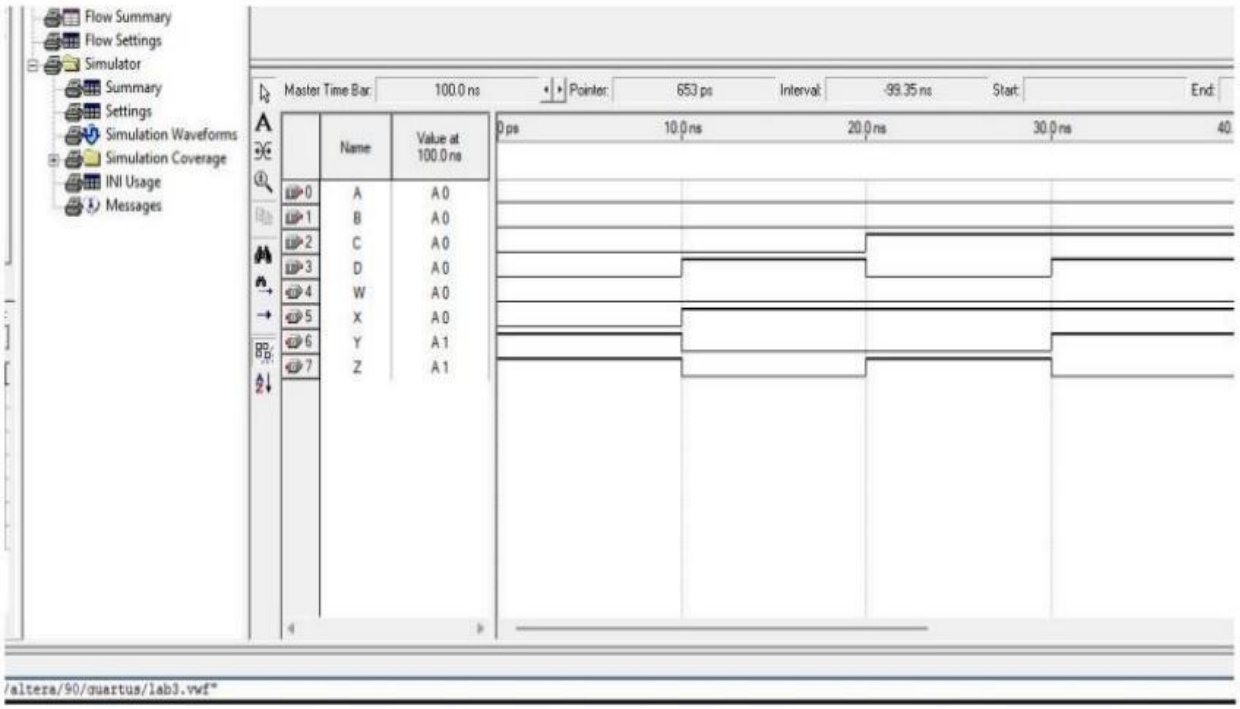
```
module labfinal ( input A, B, C,
D,

output W, X, Y, Z

);

assign W=( A)|(B)&(C|D);

assign X = (~ B & C)|( ~B & D)|(B & ~C & D);
 assign Y = (C & D)|( ~C & ~D);
assign Z = (~D)


endmodule
```

Simulation:

Input:


Master Time Bar:
Pointer:
Interval:
Start:
End:
Name
Value at
A
B
C
D
W
X
Y
Z

OUTPUT:


Flow Summary
Flow Settings
Simulator
Summary
Settings
Simulation Waveforms
Simulation Coverage
INI Usage
Messages
Master Time Bar:
100.0 ns
Pointer:
653 ps
Interval:
-99.35 ns
Start:
End:
Name
Value at
100.0 ns
A 0
B 0
C 0
D 0
W 0
X 0
Y 1
Z 1
/altera/90/quartus/lab3.vwf"