ECE 273: Introduction to Digital Logic

Lab <Project 1> : < Seven Segment Display >

By

<Hussein El-Souri>
Lab Instructor: < Zheming Liang >

March-01-2017
Winter 2017

Honor Code:

I have neither given nor received unauthorized assistance on this graded report.

X___Hussein El-Souri

Objective

Department of Electrical and Computer Engineering
College of Engineering and Computer Science
The University of Michigan-Dearborn

The first objective of this project is to take a BCD input and display the corresponding decimal number on the seven-segment display according to the following picture.

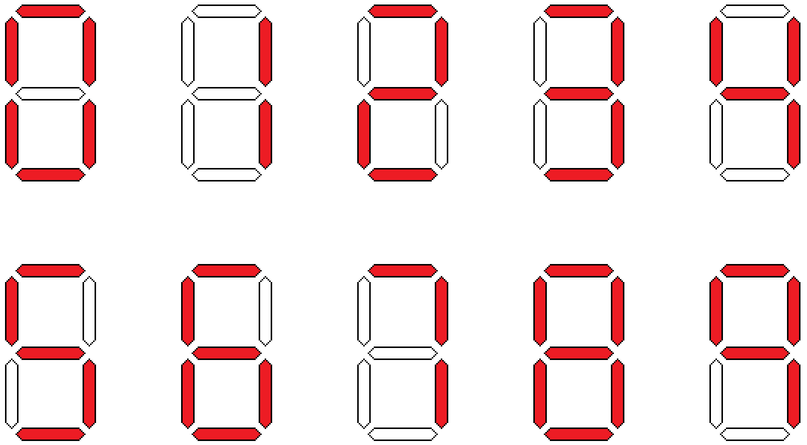

Figure 1: Segmented Display Digits

The second object involves having two control inputs that control the output on the seven-segment display per this table:

C1	C2	Display Output
0	0	Turn all segments OFF
0	1	Turn Half of the segments ON and the other Half OFF
1	0	Alternate the segments when C1C2=01
1	1	Turn all segments ON

Table 1: Second objective function table.

Theory

A truth table was used to specify the output for each segment in the seven-segment display. Boolean Algebra rules such as: Absorption 1, complements, Null Elements, Distributive law, were used to simplify those outputs as much as possible.

Equipment Used

- Software: Xilinx, PlanAhead, Logic Friday, and TinyCAD
- Hardware: BASYS 2 board.

Procedure

First, step as usual is constructing the truth table to be able to get the equation for each output.

W	X	Y	Z	A	B	C	D	E	F	G
0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0
0	0	1	1	0	0	0	0	1	1	0
0	1	0	0	1	0	0	1	1	0	0
0	1	0	1	0	1	0	0	1	0	0
0	1	1	0	0	1	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	1	0	0

Table 2: Seven Segment Truth Table

Notice that the inputs does not include BCD digits between 1010 till 1111 the output for those are treated as don’t cares. From the truth table, I can figure out the equation for each output and their simplification. The simplification is firstly done by hand however the software Logic Friday was used to verify and simply the outputs to the simplest form:

A = W’ X’ Y’ Z + W’ X Y’ Z’ -logic Friday
= X Y’ Z’ + W’ X’ Y’ Z

B = W’ X Y’ Z + W’ X Y Z’ -Logic Friday
= X Y’ Z + X Y Z’

C = W’ X’ Y Z’ -Logic Friday
= X’ Y Z’

D = W’ X’ Y’ Z + W’ X Y’ Z’ + W’ X Y Z -Logic Friday
= X Y Z + X Y’ Z’ + W’ X’ Y’ Z

E = $\underline{W' X' Y' Z}+\underline{W' X' Y Z}+\underline{\underline{W' X Y' Z'}}+\underline{\underline{W' X Y' Z}} + W' X Y + W X' Y'$ -Absorption1
= W’ X’Z(Y’+Y) + W’XY’(Z’+Z) + W’ X Y Z + W X’ Y’ Z -complements
= $\dashuline{W' X'Z} + W'XY' + \dashuline{W' X Y Z} + \dashuline{W X' Y' Z}$ - Absorption 1
= W’ X’Z (1+ Y +Y’) + W’XY’ -complements
= W’ X’Z + W’XY’ -Logic Friday
= Z + X Y’

F = W’ X’ Y’ Z + W’ X’ Y Z’ + W’ X’ Y Z + W’ X Y Z -absorption 1
= W’ X’ Y’ Z + W’ X’ Y Z’ + W’YZ(X’+X) -Null Elements
= $W' X' Y' Z + \underline{W' X' Y Z'} + \underline{W'YZ}$ -Absorption 1
= W’ X’ Y’ Z + W’Y(Z+Z’X) -Absorption 2
= W’ X’ Y’ Z + W’ Y(Z+X) -Distributive law
= $\dashuline{W' X' Y' Z} + \dashuline{W' Y Z} + W' Y X$ -Absorption 1
= W’Z(Y+Y’X’) + W’ Y X -Absorption 2
= W’Z(Y+X’) + W’ X Y -Distributive law
= W’YZ +W’X’Z + W’ X Y -Logic Friday
= X’Y + YZ + W’X’Z

G = $\dashuline{W' X' Y' Z'} + \dashuline{W' X' Y' Z} + W' X YZ$ -Absorption 1
= W’X’Y’(Z’+Z) + W’ X YZ -Complements
= W’X’Y’ + W’XYZ -Logic Friday
= W’X’Y’ + XYZ

Before implementing those equations into a module, I must integrate objective 2 and the control inputs implementing the functions assigned for the controls C1 and C2 in table1. New outputs must then be generated in terms of the initial inputs and C_1 and C_2 and those inputs follow the following truth table:

C1	C2	CA	CB	CC	CD	CE	CF	CG
0	0	1	1	1	1	1	1	1
0	1	0	0	1	1	1	0	0
1	0	1	1	0	0	0	1	1
1	1	0	0	0	0	0	0	0

Table 3: Seven segment truth table with control inputs

$CA = A + C2' = X Y' Z' + W' X' Y' Z + C2'$

$CB = B + C2' = X Y' Z + X Y Z' + C2'$

$CC = C + C1' = X' Y Z' + C1'$

$CD = D + C1' = X Y Z + X Y' Z' + W' X' Y' Z + C1'$

$CE = E + C1' = Z + X Y' + C1'$

$CF = F + C2' = X'Y + YZ + W'X'Z + C2'$

$CG = G + C2' = W'X'Y' + XYZ + C2'$

Now, these equations are implemented, using proper syntax, into a VHDL module to generate a symbol for our circuit as such:

```
A <= (X AND NOT Y AND NOT Z) OR (NOT W AND NOT X AND NOT Y AND Z) OR (NOT C2);

B <= (X AND NOT Y AND Z) OR (X AND Y AND NOT Z) OR (NOT C2);

C <= (NOT X AND Y AND NOT Z) OR (NOT C1);

D <= (X AND Y AND Z) OR (X AND NOT Y AND NOT Z) OR (NOT W AND NOT X AND NOT Y
AND Z) OR (NOT C1);

E <= Z OR (X AND NOT Y) OR (NOT C1);

F <= (NOT X AND Y) OR (Y AND Z) OR (NOT W AND NOT X AND Z) OR (NOT C2);

G <= (NOT W AND NOT X AND NOT Y) OR (X AND Y AND Z) OR (NOT C2);
```

Saving the module and with no syntax errors I can generate the following schematic:

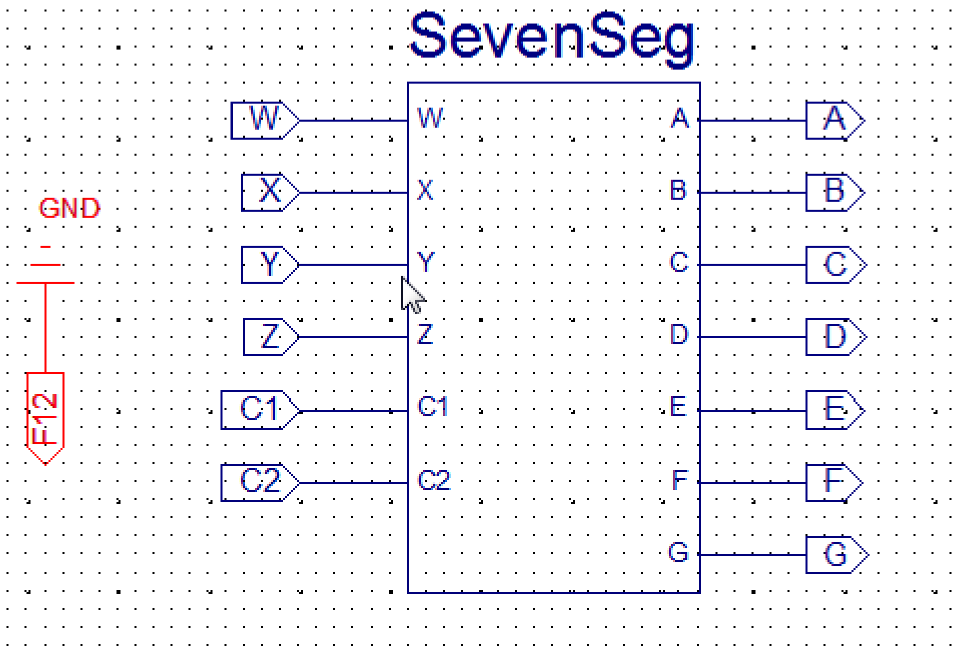

Figure2: Seven Segment Schematic

Notice a ground has been added to the schematic for proper wiring and pin specification on the board using the PlanAhead Software. The schematic has also been labeled with the proper I/O markers allowing for the next step which is writing the VHDL test bench code as such:

```
tb : PROCESS
  BEGIN

        C1 <= '0'; C2 <= '0';

        W <= '0'; X <= '0'; Y <= '0'; Z <= '0';
        WAIT FOR 50ns;
        W <= '0'; X <= '0'; Y <= '0'; Z <= '1';
        WAIT FOR 50ns;
        W <= '0'; X <= '0'; Y <= '1'; Z <= '0';
        WAIT FOR 50ns;
        W <= '0'; X <= '0'; Y <= '1'; Z <= '1';
        WAIT FOR 50ns;
        W <= '0'; X <= '1'; Y <= '0'; Z <= '0';
        WAIT FOR 50ns;
        W <= '0'; X <= '1'; Y <= '0'; Z <= '1';
        WAIT FOR 50ns;
        W <= '0'; X <= '1'; Y <= '1'; Z <= '0';
        WAIT FOR 50ns;
        W <= '0'; X <= '1'; Y <= '1'; Z <= '1';
        WAIT FOR 50ns;
        W <= '1'; X <= '0'; Y <= '0'; Z <= '0';
        WAIT FOR 50ns;
        W <= '1'; X <= '0'; Y <= '0'; Z <= '1';
        WAIT FOR 50ns;

        C1 <= '0'; C2 <= '1';

        W <= '0'; X <= '0'; Y <= '0'; Z <= '0';
        WAIT FOR 50ns;
        W <= '0'; X <= '0'; Y <= '0'; Z <= '1';
        WAIT FOR 50ns;
        W <= '0'; X <= '0'; Y <= '1'; Z <= '0';
        WAIT FOR 50ns;
        W <= '0'; X <= '0'; Y <= '1'; Z <= '1';
        WAIT FOR 50ns
        W <= '0'; X <= '1'; Y <= '0'; Z <= '0';
        WAIT FOR 50ns;
        W <= '0'; X <= '1'; Y <= '0'; Z <= '1';
        WAIT FOR 50ns;
        W <= '0'; X <= '1'; Y <= '1'; Z <= '0';
        WAIT FOR 50ns;
        W <= '0'; X <= '1'; Y <= '1'; Z <= '1';
        WAIT FOR 50ns;
        W <= '1'; X <= '0'; Y <= '0'; Z <= '0';
        WAIT FOR 50ns;
        W <= '1'; X <= '0'; Y <= '0'; Z <= '1';
        WAIT FOR 50ns;
```

```
        C1 <= '1'; C2 <= '0';

        W <= '0'; X <= '0'; Y <= '0'; Z <= '0';
        WAIT FOR 50ns;
        W <= '0'; X <= '0'; Y <= '0'; Z <= '1';
        WAIT FOR 50ns;
        W <= '0'; X <= '0'; Y <= '1'; Z <= '0';
        WAIT FOR 50ns;
        W <= '0'; X <= '0'; Y <= '1'; Z <= '1';
        WAIT FOR 50ns;
        W <= '0'; X <= '1'; Y <= '0'; Z <= '0';
        WAIT FOR 50ns;
        W <= '0'; X <= '1'; Y <= '0'; Z <= '1';
        WAIT FOR 50ns;
        W <= '0'; X <= '1'; Y <= '1'; Z <= '0';
        WAIT FOR 50ns;
        W <= '0'; X <= '1'; Y <= '1'; Z <= '1';
        WAIT FOR 50ns;
        W <= '1'; X <= '0'; Y <= '0'; Z <= '0';
        WAIT FOR 50ns;
        W <= '1'; X <= '0'; Y <= '0'; Z <= '1';
        WAIT FOR 50ns;


        C1 <= '1'; C2 <= '1';

        W <= '0'; X <= '0'; Y <= '0'; Z <= '0';
        WAIT FOR 50ns;
        W <= '0'; X <= '0'; Y <= '0'; Z <= '1';
        WAIT FOR 50ns;
        W <= '0'; X <= '0'; Y <= '1'; Z <= '0';
        WAIT FOR 50ns;
        W <= '0'; X <= '0'; Y <= '1'; Z <= '1';
        WAIT FOR 50ns;
        W <= '0'; X <= '1'; Y <= '0'; Z <= '0';
        WAIT FOR 50ns;
        W <= '0'; X <= '1'; Y <= '0'; Z <= '1';
        WAIT FOR 50ns;
        W <= '0'; X <= '1'; Y <= '1'; Z <= '0';
        WAIT FOR 50ns;
        W <= '0'; X <= '1'; Y <= '1'; Z <= '1';
        WAIT FOR 50ns;
        W <= '1'; X <= '0'; Y <= '0'; Z <= '0';
        WAIT FOR 50ns;
        W <= '1'; X <= '0'; Y <= '0'; Z <= '1';
        WAIT FOR 50ns;

  END PROCESS;
```

Finding no syntax errors and properly saving the everything along with editing the time interval to 2000ns allows us to go to the next step which is generating the waveforms which gives the following output:

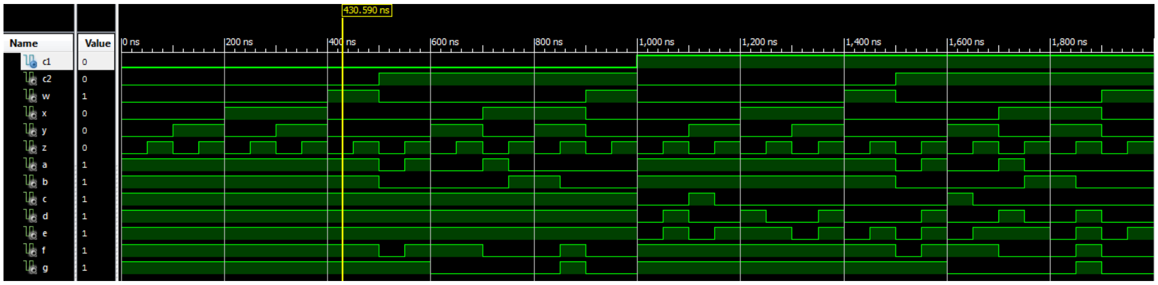

Figure 3: Seven Segment Waveforms

Consider the tine period 400-450ns where C1 and C2 = 0 ALSO W=1 while X, Y, and Z=0(1000 BCD = decimal number 8) the seven segment should display the number 8 or output CA, CB, CC, CD, CE, CF and CG=1. Which is true on the waveforms, however since control inputs are both 0 the seven-segment display will be off.

Next, I must assign the proper pins on the board using the plan ahead software as such:

```
# PlanAhead Generated physical constraints

NET "A" LOC = L14;
NET "B" LOC = H12;
NET "C" LOC = N14;
NET "C1" LOC = N3;
NET "C2" LOC = E2;
NET "D" LOC = N11;
NET "E" LOC = P12;
NET "F" LOC = L13;
NET "F12" LOC = F12;
NET "G" LOC = M12;
NET "W" LOC = F3;
NET "X" LOC = G3;
NET "Y" LOC = B4;
NET "Z" LOC = K3;
```

Notice the F12 ground marker on the schematic has been assigned to the F12 pin on the board that should be grounded. Saving the following constraints allows us to generate a bit file such that I edit the properties correctly allowing us to test the circuit on the board.

Results

The hardware test of the circuit was successful such that even if a proper BCD digit was provided to the circuit through switches the controls inputs were responsible for proper display of the decimal numbers.

Example where C1 and C2 = 0 ALSO W=1 while X, Y, and Z=0(1000 BCD = decimal number 8) the seven segment should display the number 8 or output CA, CB, CC, CD, CE, CF and CG=1 this physically means that switches N3 and E2 are switched down while switch F3 is up and G3, B4 and K3 are switched down this should display the number 8 but the control inputs determine that the seven-digit display is off. If I switch N3 and E2 up, I will see the number 8 on the display.

Conclusion

The circuit was designed correctly and if the BCD digits between 1010 till 1111 were to be taken into consideration I could have been able to tweak this design into also displaying Hex digits A,B,C,D,E, and F.