Justin Youn & Luis Gomez
Professor Callenes
CPE 133-11/12
24 October 2018

Lab 5 : BCD Comparator Display Modules

Experiment 9:

Summary: This digital circuit was created using behavioral modeling in Verilog. The circuit took two 4-bit input values A & B, compared them and displayed the greater of the two inputs on the display or neither if A equaled B. In this circuit we utilized the previously constructed comparator and BCD-to-7-segment-display modules from previous labs.

Figure below: Structural model for 4 bit comparator to 7 segment display

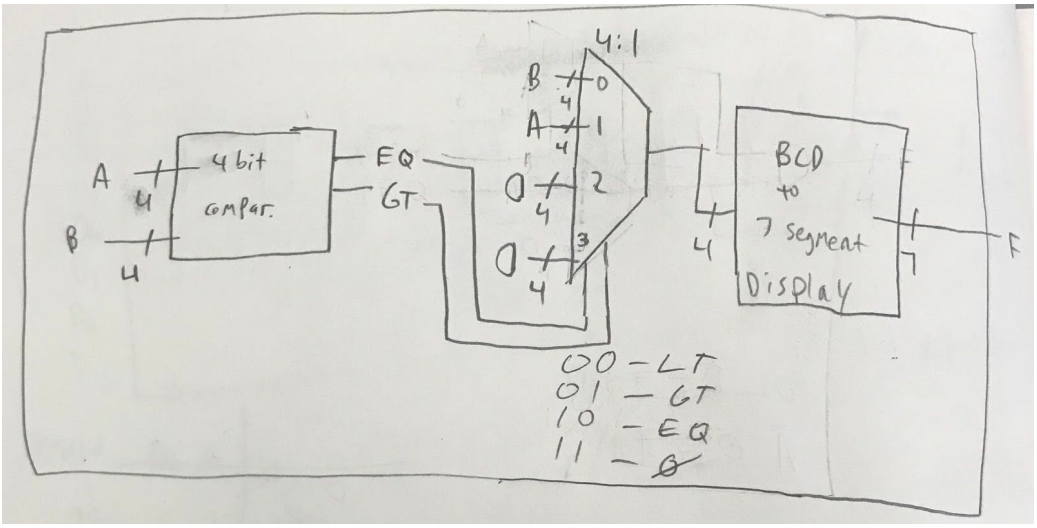
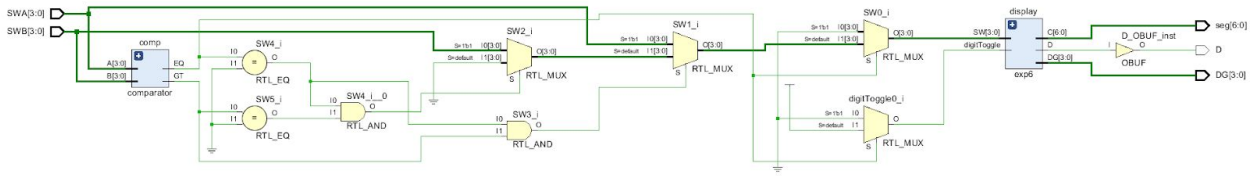


Figure below, incomplete Truth table displaying 8 out of 256 possible input combinations, highly impractical approach.

MSB				MSB								
A3	A2	A1	A0	B3	B2	B1	B0	EQ	GT	LT	DIGIT	RADIX
0	0	0	0	0	0	0	0	1	0	0	1	1
0	0	0	0	0	0	0	1	0	0	1	0	0
0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	0	1	1	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	1	0	0	1	0	0
0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	0	1	1	0	0	1	0	0

Figure below: Circuit diagram of exp9



Verification: To verify our implementation we exhaustively tested our circuit diagram by using various different inputs. We displayed our testing in the video found below.

https://youtu.be/bIoJYzGLPOw

Questions:

1. Would it have been possible to implement this design without using some type of MUX? Briefly explain.

a. No there isn't a way to design this without using some type of MUX. Any logic used to implement the design would route back to the use of a MUX. Using a standard decoder with the EQ as the enabler was my original idea, but this standard decoder will never output the original inputs.

2. For this lab activity, is there any difference between a BCD number and a 4-bit unsigned binary number? Briefly explain.

a. For this lab activity, there aren't any differences a BCD number and a 4-bit unsigned binary number. A 4-bit unsigned binary number will be interpreted the same as a BCD number.

Code:

```
22
23  module exp9(
24      input [3:0] SWA, // A
25      input [3:0] SWB, // B
26      output D,        // radix
27      output [3:0]DG, // digit toggle
28      output [6:0] seg // 7-seg display
29      );
30
31      wire EQ,LT,GT;
32      reg [3:0] RESULT;
33      reg dgToggle ;
34
35      comparator comp( SWA, SWB, EQ, LT, GT);
36
37      always @ (SWA, SWB) begin
38          dgToggle =
39          (EQ == 1) ? 1'b0 : 1'b1;
40          RESULT =
41          ((EQ == 1) ? 4'b0000 :
42          ((EQ == 0 && GT == 1) ? SWA :
43          ((EQ == 0 && GT == 0) ? SWB : 4'b0000)
44          ));
45      end
46
47      exp6 display( RESULT, dgToggle, D, DG, seg);
48
49  /*module exp6(
50      input [3:0] SW,
51      output D,
52      output [2:0] DG,
53      output [6:0] C
54      );*/
55
56  /*module comparator(
57          input [3:0] A,
58          input [3:0] B,
59          output EQ,
60          output LT,
61          output GT
62          );*/
63  endmodule
```

Experiment 10:

Summary: This digital circuit was created using behavioral modeling in Verilog. The circuit took two 4-bit input values A & B and a 1-bit button input. The circuit then compared A & B and displayed: A on the leftmost digit when A > B, B on the rightmost digit when A < B, and A & B on the two center digits when A = B. If at any point Button 0 was pressed, the display would not show anything. In this circuit we utilized the previously constructed comparator and BCD-to-7-segment-display modules from previous labs and an internal clock signal oscillate between displaying A & B.

Figure below: BBD for exp 10

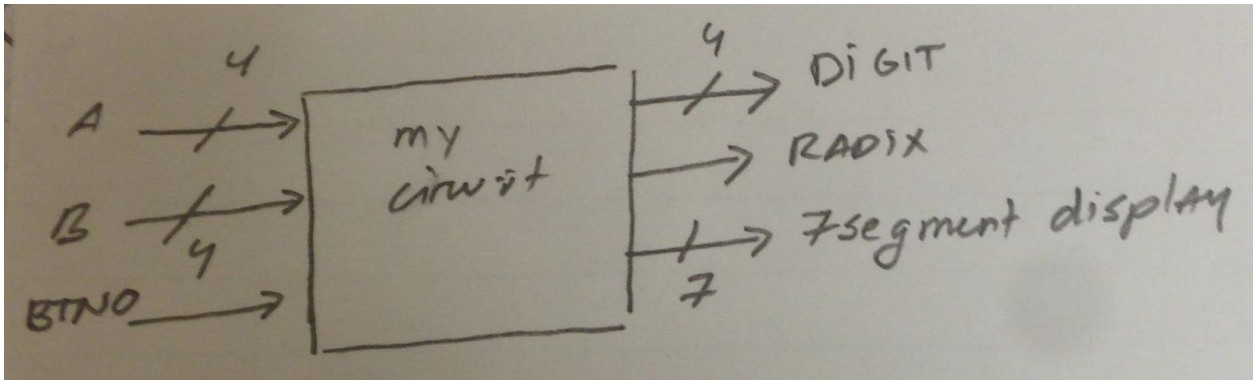
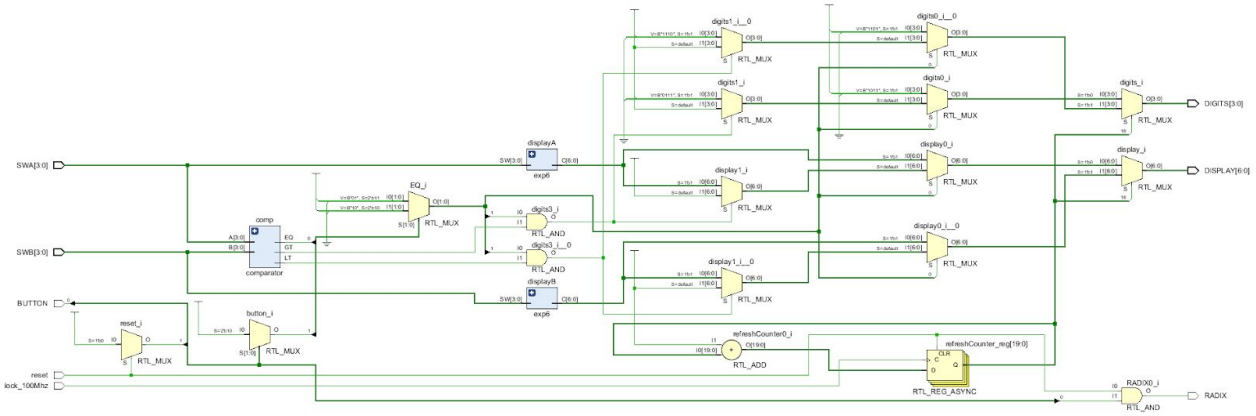


Figure below: Circuit diagram for exp 10



Verification:

To verify our implementation we exhaustively tested our circuit diagram by using various different inputs. We displayed our testing in the video demo found below.

https://youtu.be/-iPHOSUvWn8

Questions:

1. Would it have been possible to modeling this entire circuit with one always block? Support your answer of why you think this is possible or not.

a. Yes, this entire circuit would be possible to implement with one always block. This can be possible by using if else statements to simulate a mux inside of the always block.

Code:

```
23  module exp10(
24      input clock_100Mhz,
25      input reset,
26
27      input [3:0] SWA, // A
28      input [3:0] SWB, // B
29      input BUTTON,    // button 0
30      output RADIX,        // toggles radix
31      output [3:0]DIGITS,  // toggles 4-digit display
32      output [6:0] DISPLAY // toggles 7-seg display
33      );
34
35      wire EQ,LT,GT;
36      wire button = BUTTON;
37      wire LEDcounter;
38
39      reg [19:0] refreshCounter;
40      reg [6:0] display;
41      wire [6:0] displayOutputA;
42      wire [6:0] displayOutputB;
43      reg radix;
44      reg [3:0] digits; // MSB is the radix toggle, others toggle digit display
45
46      comparator comp( SWA, SWB, EQ, LT, GT);
47      exp6 displayB( SWB, displayOutputB);
48      exp6 displayA( SWA, displayOutputA);
49
50  always @(posedge clock_100Mhz or posedge reset)
51      begin
52       if(reset == 1)
53        refreshCounter <= 0;
54       else
55        refreshCounter <= refreshCounter + 1;
56      end
57      assign LEDcounter = refreshCounter[19:18];
58
59  always @ (SWA, SWB, button, LEDcounter) begin
60      case (LEDcounter)
61      1'b0: begin // A cycle
62          radix =
63          (reset & button) ? 1'b1 : 1'b0;
64          digits =
65          ((reset == 0 & button == 0 & EQ == 1) ? 4'b1011 :
66          ((reset == 0 & button == 0 & EQ == 0 & GT == 1) ? 4'b0111 :
67          4'b1111));
68          display =
69          ((reset == 0 & button == 0 & EQ == 1) ? displayOutputA :
70          ((reset == 0 & button == 0 & EQ == 0 & GT == 1) ? displayOutputA :
71          7'b1111111));
72          end
73      1'b1: begin // B cycle
74          radix =
75          (reset & button) ? 1'b1 : 1'b0;
```

```
75          (reset & button) ? 1'b1 : 1'b0;
76          digits =
77          ((reset == 0 & button == 0 & EQ == 1) ? 4'b1101 :
78          ((reset == 0 & button == 0 & EQ == 0 & LT == 1) ? 4'b1110 :
79          4'b1111));
80          display =
81          ((reset == 0 & button == 0 & EQ == 1) ? displayOutputB :
82          ((reset == 0 & button == 0 & EQ == 0 & LT == 1) ? displayOutputB :
83          7'b1111111));
84          end
85      default: begin
86          radix = 1'b1;
87          digits = 4'b1111;
88          display = 7'b1111111;
89          end
90      endcase
91      end
92
93      assign RADIX = radix;
94      assign DIGITS= digits;
95      assign DISPLAY = display;
96  endmodule
```