Lab 4

Seven Segment Display Counter

Group G

Chang, Philbert (013179257)

Hua, Russell (013184015)

Thai, Paul (014760252)

ECE 3300L.01

Professor Aly

07/12/2021

Problem

Use two seven segment displays to count from 00 to 99, and then repeat.

Code Detail

A counter is created that counts from 0-9 for both the tens digit and ones digit. The code is setup in a nested if statement where once the counter reaches 9, it resets to 0. This counter variable is then passed into a “blinker” that sequentially blinks the ones digits and tens digit at a speed that is undiscernible to the human eye. This is done because the seven segment displays on the Artix 7 FPGA are all connected to the same pins, which would not allow users to display different numbers on each seven segment LED.

Schematic

Results

```
23  module sseg(
24          input SW,
25          input mclk,
26          input reset,
27          output reg [6:0] a_to_g,
28          output reg [7:0] AN,
29          output DP
30              );
31          //make decimal zero
32          assign DP = 1'b1;
33
34          reg [19:0] blink_refresh;
35          wire sseg_activate;
36
37          parameter max_count = 10_000_000-1; //0.1 ms counter
38          wire    count;
39          reg     [26:0] counter_100;
40          reg     [3:0] counter_10l;
41          reg     [3:0] counter_10h;
42
43          //counter to divide the 100Mhz
44          always @(posedge mclk, posedge reset)
45              if(reset)
46                  counter_100 <=0;
47              else if (counter_100 == max_count)
48                  counter_100 <=0;
49              else
50                  counter_100 <= counter_100 + 1'b1;
51
52          //signal active everything 100 mil clocks
53          assign count = counter_100 == 0;
54          //counter to count evey number in 1hz
```

Figure 1: Seven Segment Display Counter [P1]

```
54          //counter to count evey number in 1hz
55          always @(posedge mclk or posedge reset)
56              if(reset) begin
57                  counter_10l <= 0;
58                  counter_10h <= 0;
59                  end
60              else if (SW)
61                  if (count)
62                  //if it goes pass 9, reset
63                      if(counter_10l == 9) begin
64                          counter_10l <= 0;
65                          if(counter_10h == 9)
66                              counter_10h <= 0;
67                          else
68                              counter_10h = counter_10h + 1'b1;
69                          end
70                      else
71                          counter_10l <= counter_10l + 1'b1;
72
73      always @ (posedge mclk or  posedge reset)
74          begin
75              if (reset)
76                  blink_refresh <= 0;
77              else
78                  blink_refresh <= blink_refresh + 1'b1;
79          end
80
81          assign sseg_activate = blink_refresh[19:18];
82
```

Figure 2: Seven Segment Display Counter Code [P2]

```
83      always @ (*)
84          begin
85              case (sseg_activate)
86                  0: begin
87                      AN <= 8'b11111110;
88                      case(counter_10l)
89                          0: a_to_g = 7'b0000001;
90                          1: a_to_g = 7'b1001111;
91                          2: a_to_g = 7'b0010010;
92                          3: a_to_g = 7'b0000110;
93                          4: a_to_g = 7'b1001100;
94                          5: a_to_g = 7'b0100100;
95                          6: a_to_g = 7'b0100000;
96                          7: a_to_g = 7'b0001111;
97                          8: a_to_g = 7'b0000000;
98                          9: a_to_g = 7'b0000100;
99                          default: a_to_g = 7'b0000001;
100                     endcase
101                 end
102                 1: begin
103                     AN <= 8'b01111111;
104                     case(counter_10h)
105                         0: a_to_g = 7'b0000001;
106                         1: a_to_g = 7'b1001111;
107                         2: a_to_g = 7'b0010010;
108                         3: a_to_g = 7'b0000110;
109                         4: a_to_g = 7'b1001100;
110                         5: a_to_g = 7'b0100100;
111                         6: a_to_g = 7'b0100000;
112                         7: a_to_g = 7'b0001111;
113                         8: a_to_g = 7'b0000000;
114                         9: a_to_g = 7'b0000100;
115                         default: a_to_g = 7'b0000001;
116                     endcase
117                 end
118             endcase
119         end
120
121 endmodule
```

Figure 3: Seven Segment Display Counter Code [P3]

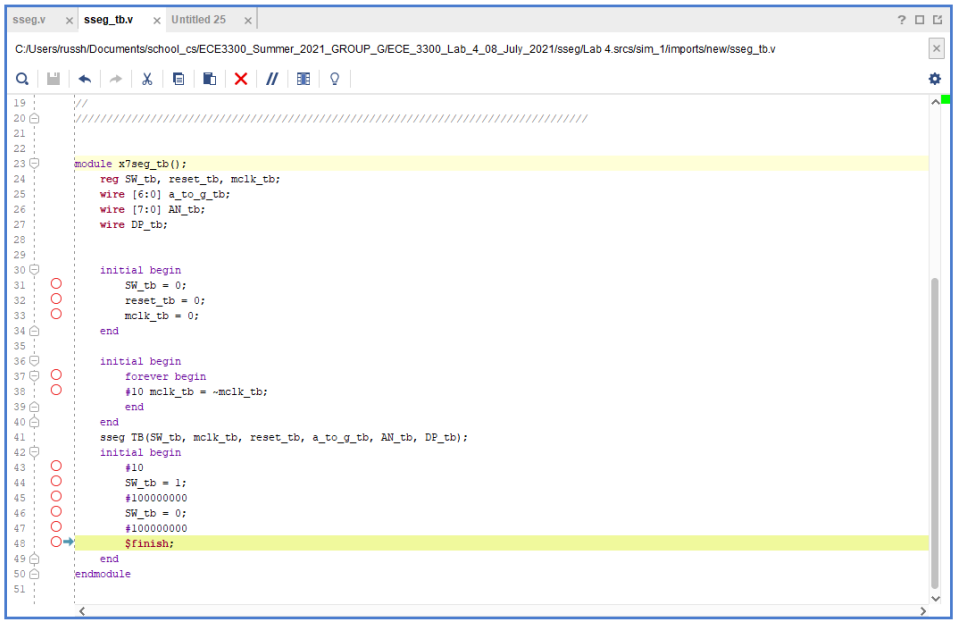

Figure 4: Simulation testbench code

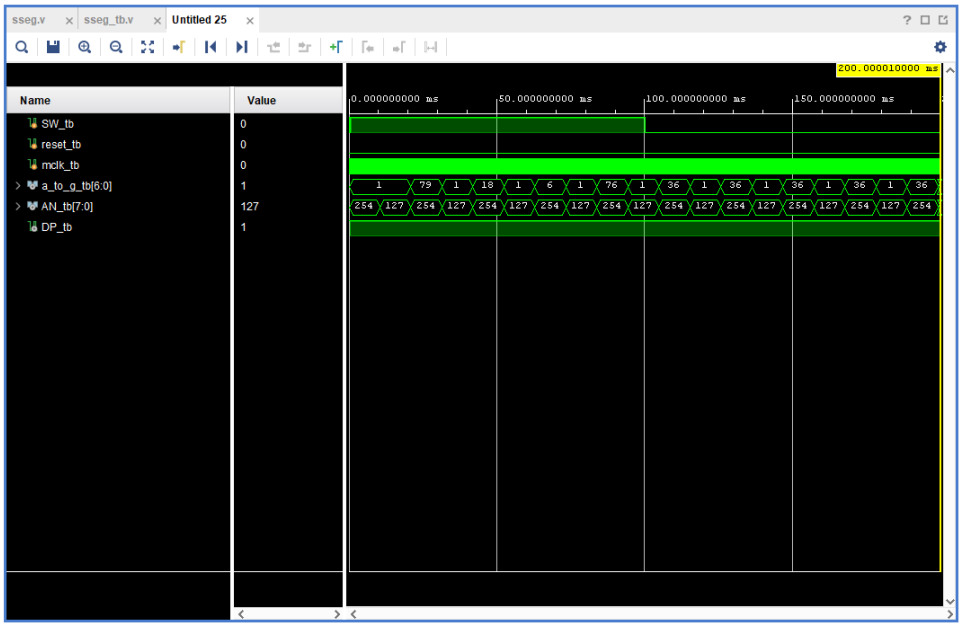

Figure 5: Simulation results

Name	Slice LUTs (32600)	Slice Registers (65200)	Slice (8150)	LUT as Logic (32600)	Bonded IOB (210)	BUFGCTRL (32)
N sseg	61	54	30	61	19	1

Figure 6: Report Utilization

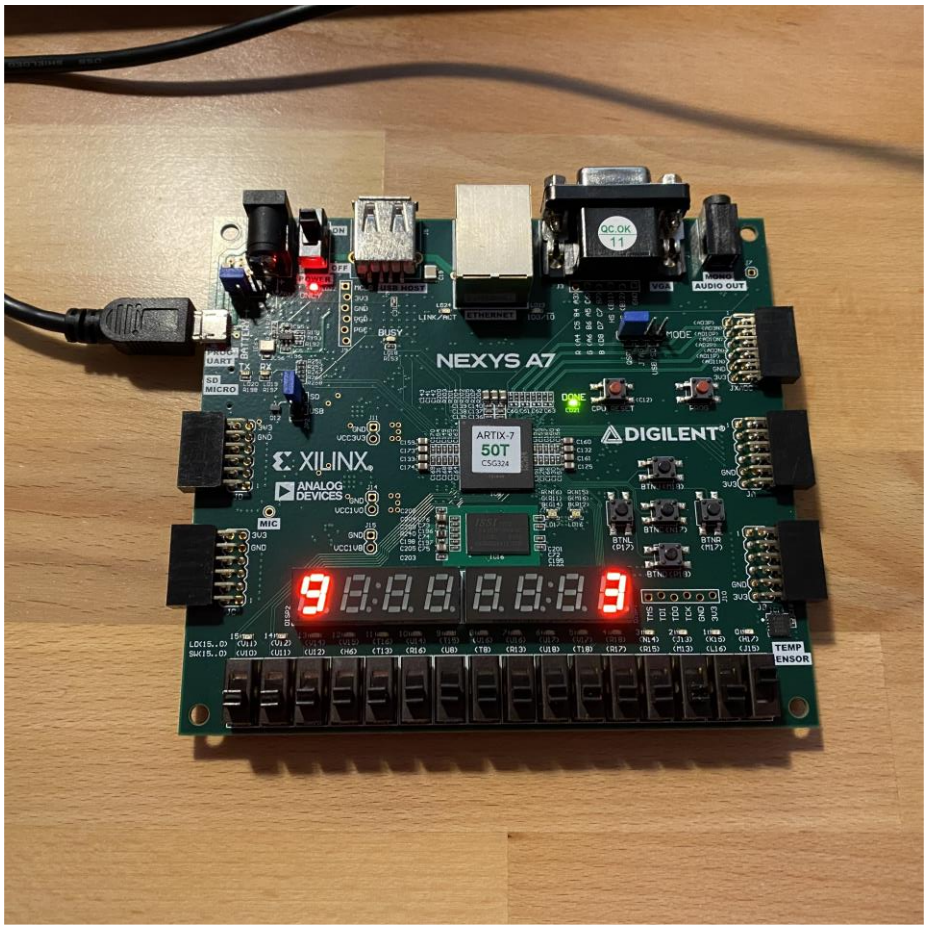

Figure 7: Picture of FPGA with Switch ON

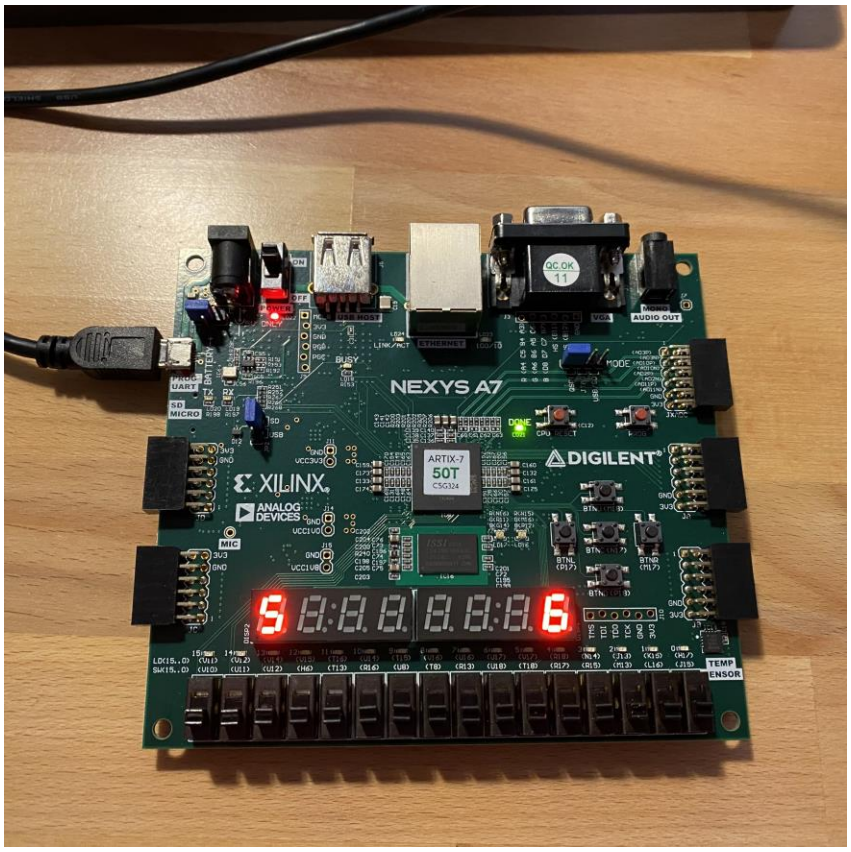

Figure 8: Picture of FPGA with Switch OFF

Discussion

The display counting works, both in simulation and in hardware. In the simulation, the blinking intervals and a-g outputs are as expected. Figuring out blinking intervals were difficult, along with getting the switch to work. The switch wasn't working because using an if statement required it to be in an initial or always block. Eventually putting it in the counter section with an else if worked. The blinking intervals were done by filling a register based on the clock and when it reached a certain value, it would select the appropriate display. The result can be seen in slow motion: https://livecsupomona-my.sharepoint.com/:v:/g/personal/rjhua_cpp_edu/EXCybMD0aANHjlrPMB4qSQABNYEKFr1r0mq6XHg8slo0zQ?e=0jrkTF

Conclusion

It works, the display behaves properly and the inputs – switch and button – all work as intended. The speed of counting should be changed to properly see all the values but making it too fast will make it hard to see. Much optimization can be done to reduce resource utilization, such as in the counting and in the blinking intervals. Perhaps it's possible to avoid using a register for the blinking interval or making the code more efficient for the switch.

Work Distribution

Russell: Code, testbench, and video

Philbert: Code and video

Paul: Code and video