ELC 2137 Lab 11: FSM: Guessing Game

Jake Simmons

April 23, 2020

Summary

The purpose of this lab was to learn how to use the technique called Time-Division Multiplexing. We are using Time-Division Multiplexing to slow down the signals going to the 4 digit display. This is so there will be enough time for a human to process and see what is actually being displayed. First, a counter module was made and succesfully simulated. Secondly, the sseg4 from a previous lab was modified and the counter was added to the module. A schematic of the new, sseg4TDM, is provided. After testing sseg4TDM, the top level module, calclab10, was created. The top level module toplab9 was used in this module which was also created in a previous lab.

Q&A

1. At what time in the simulation did the debounce circuit reach each of the four states?

   (a) state zero: , state wait1: , state one: , state wait0:

2. Why can this game not be implemented with regular sequential logic?

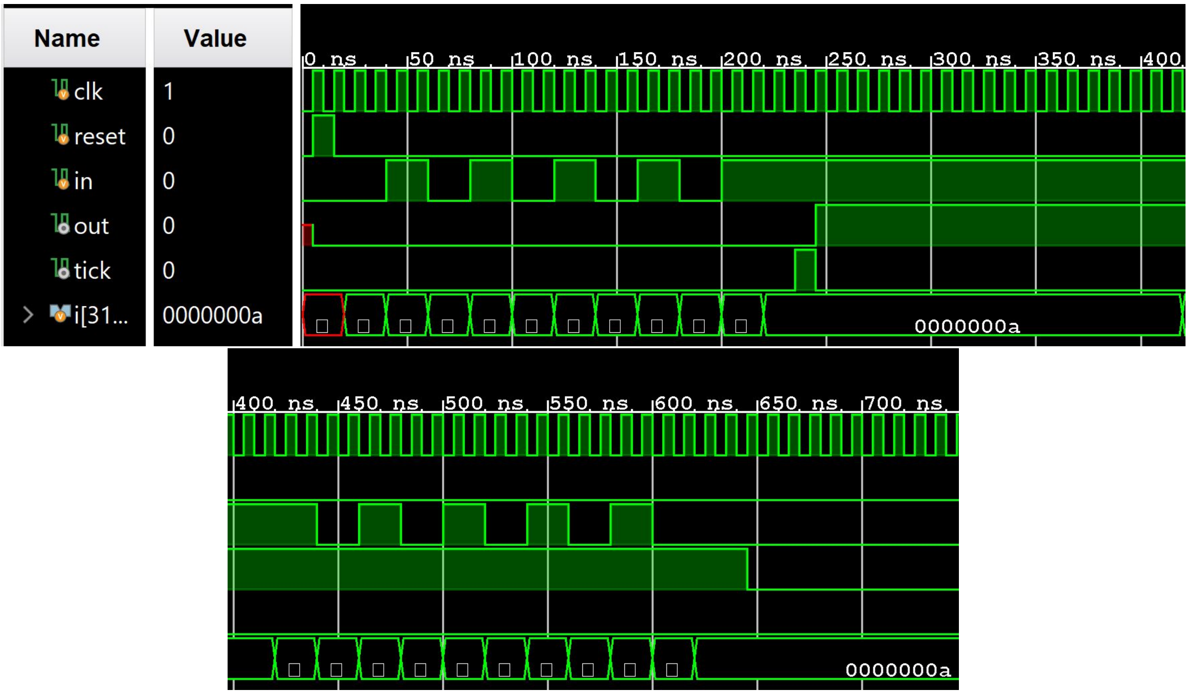
   (a) state zero: , state wait1: , state one: , state wait0:

3. What type of outputs did you use for your design(Mealy or Moore)? Explain.

   (a)

Results

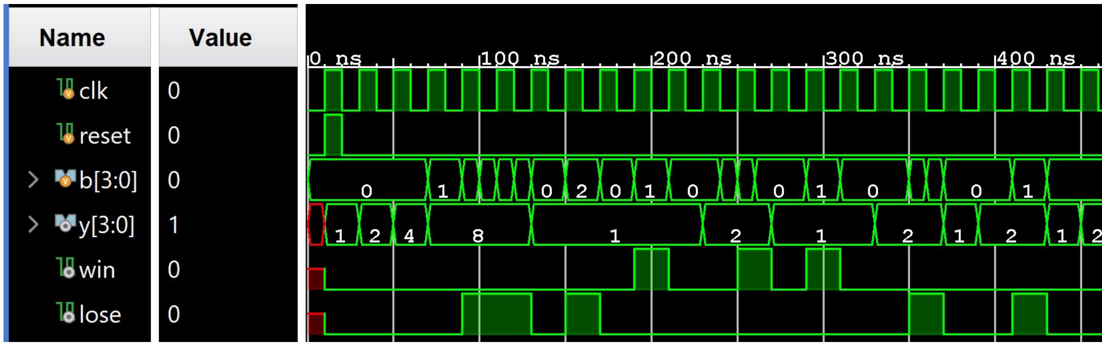
Time (ns):	0	50	100	150	200	250	300	350	400	450	500	550	600	650	700	750
clk	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
rst	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
in	0	1	1	0	1	1	1	1	1	0	1	1	1	0	0	0
out	X	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0
tick	X	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
i	X	0	1	4	6	9	a	a	a	a	1	4	6	9	a	a



Name	Value
clk	1
reset	0
in	0
out	0
tick	0
i[31...	0000000a

Figure 1: debounce simulation waveform and ERT

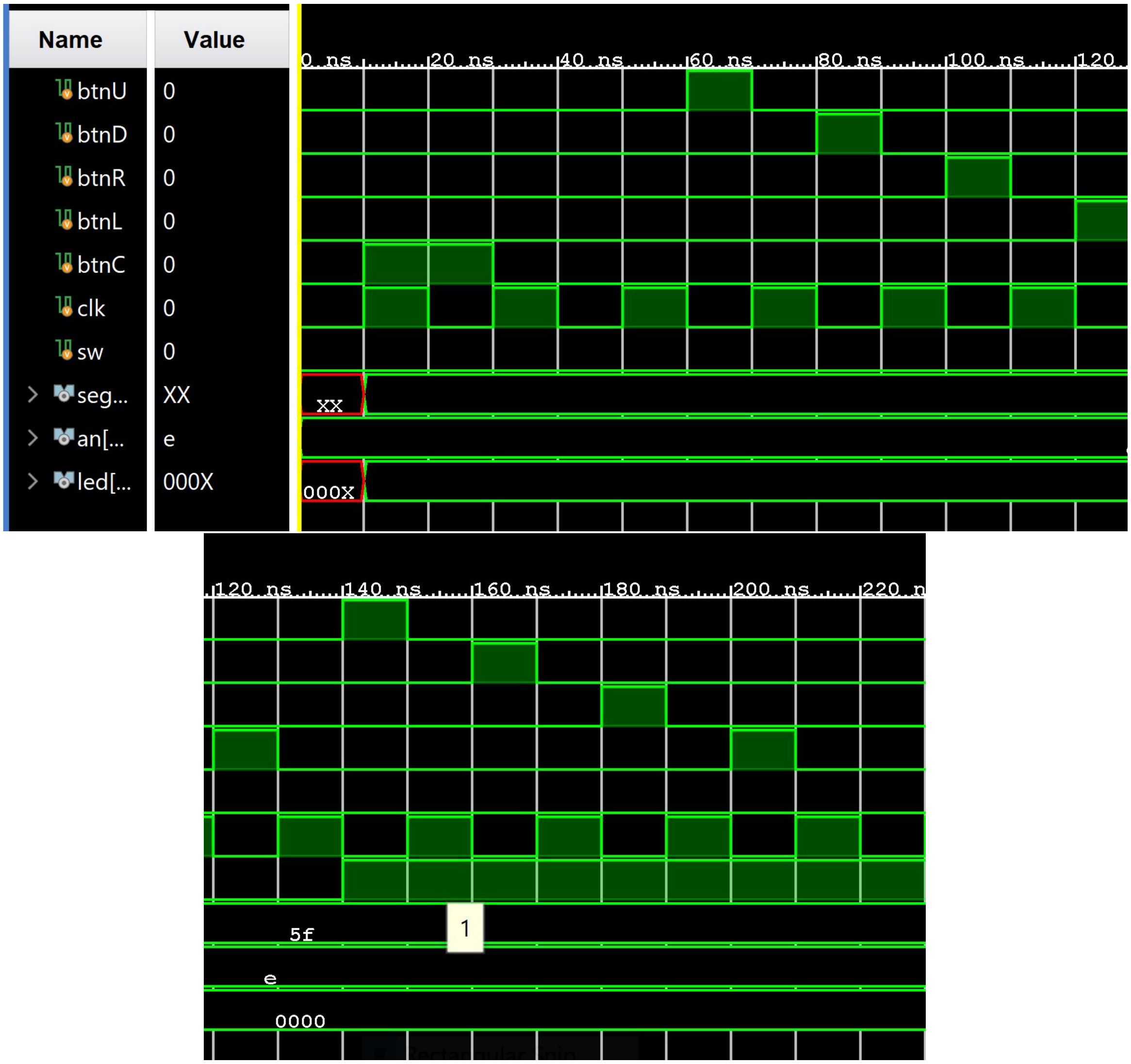
Time (ms):	0	25	50	75	100	125	150	175	200	225	250	275	300	325	350	375	400
clk	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
rst	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b	0	0	0	1	2	1	0	0	1	0	4	0	1	8	b	0	0
y	X	1	2	4	8	8	1	1	1	1	2	2	1	1	2	2	1
win	X	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0
lose	X	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0	1



Name	Value
clk	0
reset	0
b[3:0]	0
y[3:0]	1
win	0
lose	0

Figure 2: guess FSM simulation waveform and ERT

Time (ms):	0	20	40	60	80	100	120	140	160	180	200	220
btnU	0	0	0	1	0	0	0	1	0	0	0	0
btnD	0	0	0	0	1	0	0	0	1	0	0	0
btnR	0	0	0	0	0	1	0	0	0	1	0	0
btnL	0	0	0	0	0	0	1	0	0	0	1	0
btnC	0	1	0	0	0	0	0	0	0	0	0	0
clk	1	1	1	1	1	1	1	1	1	1	1	1
sw	0	0	0	0	0	0	0	1	1	1	1	1
seg	X	5f	5f	5f	5f	5f	5f	5f	5f	5f	5f	5f
an	e	e	e	e	e	e	e	e	e	e	e	e
led	X	0	0	0	0	0	0	0	0	0	0	0



Name	Value
btnU	0
btnD	0
btnR	0
btnL	0
btnC	0
clk	0
sw	0
seg...	XX
an[...	e
led[...	000X

Figure 3: guess FSM simulation waveform and ERT

Figure 4: sseg4TDM Schematic

Figure 5: calclab10 Schematic

Code

Listing 1: guess FSM Module

```
‘timescale 1ns / 1ps
//ELC 2137, Jake Simmons, 2020-04-20

module guess_FSM (
   input [3:0]b,
   input reset,
   input clk,
   output reg [3:0]y,
   output reg win,
   output reg lose
   );

   //states
   localparam [2:0]
      S0 = 3’b000,
      S1 = 3’b001,
      S2 = 3’b010,
      S3 = 3’b011,
      SWIN = 3’b100,
      SLOSE = 3’b101;

   //internal signals
   reg[2:0] nState, State;

   always_ff @(posedge clk or posedge reset)
      if(reset == 1) begin
         State <= S0;
      end
      else begin
         State <= nState;
      end


   always_comb begin
      case(State)
      S0: begin
         y[0] = 1;
         y[3:1] = 0;
         lose = 1’b0;
         win = 1’b0;
         if(b==1)
            nState = SWIN;
         else if(b==0)
            nState = S1;
         else
            nState = SLOSE;
         end

      S1: begin
         y[1] = 1;
         y[0] = 0;
         y[3:2] = 0;
         if(b==2)
            nState = SWIN;
         else if(b==0)
            nState = S2;
         else
            nState = SLOSE;
         end

      S2: begin
         y[2] = 1;
         y[3] = 0;
         y[1:0] = 0;
         if(b==4)
            nState = SWIN;
         else if(b==0)
            nState = S3;
         else
            nState = SLOSE;
         end

         S3: begin
            y[3] = 1;
            y[2:0] = 0;
            if(b==8)
               nState = SWIN;
            else if(b==0)
               nState = S0;
            else
               nState = SLOSE;
            end

            SWIN: begin
               win = 1'b1;
               lose = 1'b0;
               if(b==0)
                  nState = S0;
               else
                  nState = SWIN;
               end

            SLOSE: begin
               lose = 1'b1;
               win = 1'b0;
               if(b==0)
                  nState = S0;
               else
                  nState = SLOSE;
               end
            endcase
         end
endmodule
```

Listing 2: guess FSM Test Bench

```
`timescale 1ns / 1 ps
//ELC 2137, Jake Simmons, 2020-04-21
module guess_FSM_test();

   reg clk, reset;
   reg [3:0] b;
   wire [3:0] y;
   wire win, lose;

   guess_FSM gs( .clk(clk), .b(b), .reset(reset), .y(y), .win(win),
   .lose(lose) );

   always begin
      clk = ~clk; #10;
   end

   initial begin
      clk = 0; reset = 0; b = 0; #10;
      reset = 1; #10;
      reset = 0; #10;

      b = 0; #10; //S1
      b = 0; #10; //S2
      b = 0; #10; //S3

      b = 0; #10; //S0
      b = 1; #20; //SWIN
      b = 2; #10; //SWIN

      b = 0; #10 //S0;
      b = 2; #10; //SLOSE
      b = 1; #10; //SLOSE

      b = 0; #10; //S0
      b = 0; #10; //S1
      b = 2; #10; //SWIN
      b = 2; #10; //SWIN

      b = 0; #10; //S0
      b = 0; #10; //S1
      b = 1; #10; //SLOSE
      b = 1; #10; //SLOSE

      b = 0; #10; //S0
      b = 0; #10; //S1
      b = 0; #10; //S2
      b = 4; #10; //SWIN
      b = 2; #10; //SWIN
      b = 0; #10; //S0
      b = 0; #10; //S1
      b = 0; #10; //S2
      b = 1; #10; //SLOSE
      b = 1; #10; //SLOSE
      b = 0; #10; //S0

      b = 0; #10; //S1
      b = 0; #10; //S2
      b = 0; #10; //S3
      b = 8; #10; //SWIN
      b = 2; #10; //SWIN
      b = 0; #10; //S0
      b = 0; #10; //S1
      b = 0; #10; //S2
      b = 0; #10; //S3
      b = 1; #10; //SLOSE
      b = 1; #10; //SLOSE
      b = 0; #10; //S0


   end
endmodule
```

Listing 3: guessing game Module

```
‘timescale 1ns / 1ps
// ELC 2137, Jake Simmons, 2020-04-22

module guessing_game(
   input btnU,
   input btnD,
   input btnR,
   input btnL,
   input btnC,
   input clk,
   input [15:0]sw,
   output [6:0] seg,
   output [3:0] an,
   output [15:0] led,
   output dp
   );

   wire [3:0] W1;
   wire [1:0] W2;
   wire [15:0] W3;
   wire [3:0] W4;
   wire W5, W6;
   wire [1:0] W7;

   debounce #(.N(21)) d1( .in(btnU), .clk(clk), .reset(btnC), .out(W1[3]))
      ;
   debounce #(.N(21)) d2( .in(btnR), .clk(clk), .reset(btnC), .out(W1[2]))
      ;
   debounce #(.N(21)) d3( .in(btnD), .clk(clk), .reset(btnC), .out(W1[1]))
      ;
   debounce #(.N(21)) d4( .in(btnL), .clk(clk), .reset(btnC), .out(W1[0]))
      ;
   Counter #(.N(25)) count( .clk(clk), .en(1’b1), .tick(W2));
   Counter #(.N(23)) count1( .clk(clk), .en(1’b1), .tick(W7));

   mux2 #(.N(25)) m( .in1(W2), .in0(W7), .sel(sw[0]), .out(W3));

   guess_FSM gFSM( .b(W1), .clk(W3), .y(W4), .win(W5), .lose(W6), .reset(
      btnC));

   //top
   assign seg[0] = ~W4[3];

   //right
   assign seg[1] = ~W4[2];

   assign seg[4:2] = 3’b111;

   //left
   assign seg[5] = ~W4[0];

   //bottom
   assign seg[6] = ~W4[1];

   //win
   assign led[0] = W5;

   //lose
   assign led[1] = W6;


   assign led[15:2] = 14’b00000000000000;
   assign an = 4’b1110;

   assign dp = 1’b1;

   endmodule
```

Listing 4: guessing game Test Bench

```
‘timescale 1ns / 1ps
//ELC 2137, Jake Simmons, 2020-04-22

module guessing_game_test();
   reg btnU;
   reg btnD;
   reg btnR;
   reg btnL;
   reg btnC;
   reg clk;
   reg sw;
   wire [6:0] seg;
   wire [3:0] an;
   wire [15:0] led;

   guessing_game test( .btnU(btnU), .btnD(btnD), .btnR(btnR),
   .btnL(btnL), .btnC(btnC), .clk(clk), .seg(seg), .an(an),
   .led(led));

   always begin
      clk = ~clk; #10;
   end

   initial begin
      clk = 0; btnC = 0; btnU = 0; btnD = 0; btnR = 0; btnL = 0; sw = 0;
         #10;
      btnC = 1; #20;
      btnC = 0; #20;
      btnU = 0; btnD = 0; btnR = 0; btnL = 0; sw = 0; #10;
      btnU = 1; btnD = 0; btnR = 0; btnL = 0; sw = 0; #10;
      btnU = 0; btnD = 0; btnR = 0; btnL = 0; sw = 0; #10;
      btnU = 0; btnD = 1; btnR = 0; btnL = 0; sw = 0; #10;
      btnU = 0; btnD = 0; btnR = 0; btnL = 0; sw = 0; #10;
      btnU = 0; btnD = 0; btnR = 1; btnL = 0; sw = 0; #10;
      btnU = 0; btnD = 0; btnR = 0; btnL = 0; sw = 0; #10;
      btnU = 0; btnD = 0; btnR = 0; btnL = 1; sw = 0; #10;
      btnU = 0; btnD = 0; btnR = 0; btnL = 0; sw = 0; #10;
      btnU = 1; btnD = 0; btnR = 0; btnL = 0; sw = 1; #10;
      btnU = 0; btnD = 0; btnR = 0; btnL = 0; sw = 1; #10;
      btnU = 0; btnD = 1; btnR = 0; btnL = 0; sw = 1; #10;
      btnU = 0; btnD = 0; btnR = 0; btnL = 0; sw = 1; #10;
      btnU = 0; btnD = 0; btnR = 1; btnL = 0; sw = 1; #10;
      btnU = 0; btnD = 0; btnR = 0; btnL = 0; sw = 1; #10;
      btnU = 0; btnD = 0; btnR = 0; btnL = 1; sw = 1; #10;
      btnU = 0; btnD = 0; btnR = 0; btnL = 0; sw = 1; #10;

   end
endmodule
```