ELC 2137 Lab 11: FSM: Guessing Game

Mariah Montgomery

November 22, 2020

Summary

In this lab, we built on our knowledge of sequential logic by creating a game on the Basys 3 Board. The game requires the player to push a button that corresponds to a light that "circles" around the top half of a single digit of the seven-segment display. The game consisted of two difficulties based on the speed of the moving light. If the player won the game, all of the LEDs on the board would illuminate, and if the player lost, only every other LED would turn on.

The first step in creating this game was to implement the given debounce module. This module eliminated the "bounce" produced by a button push on the Basys board. Debouncing the buttons allowed for clean signals. After implementing the debounce module, we tested the module and ensured that the simulation waveform was up to our standard before creating the finite state machine.

The finite state machine we created was names guess FSM. This design source handled irregular conditions that set up the logic of our guessing game. The module determines what "state" the light on the Basys board is in, i.e., what light is illuminated on the seven segment display, and then adjusted the board's state when a button is pushed. Given that there are four segments on the upper half of a single digit on a seven-segment display, there are four states that the game can be in if not already in a win or lose state. If the correct button is pushed, the state moves to win. If the wrong button is pressed, the state moves to lose, and if no button is pushed, the game proceeds to the next state.

Next, I created a decoder similar to the seven segment decoder used in previous labs. This decoder allowed for a single segment on the seven segment display to illuminate based on the game's current state.

Lastly, I created the top-level module to connect the debounce module to four inputs to the guess FSM and then two different counters with separate parameters to a mux that acted as the enable of the guess FSM module. The guess FSM module's output was fed into the decoder's input, and the desired anode was set turned on. The game was now ready to play!

Q&A

1. The simulation of the debounce circuit reached the zero state at 645 ns, the wait1 state at 200 ns, the one state at 245 ns, and the wait0 state at 600 ns.

2. This game could not be implemented with regular sequential logic because we needed a finite state machine to implement the game design. A finite state machine can handle irregular, non repeating conditions, while regular sequential logic cannot. The finite state machine can switch between states, and that was necessary for the creation and implementation of the game.

3. I used Moore outputs for my design because Moore outputs are safer. A Moore machine's output is based only on its current state whereas a Mealy machine's output is based on the machine's current state and input. The output of the FSM in this game was only based on the current state of the machine; therefore, I used a Moore design.

Results

1. Box link to video of the game on the slow setting:
https://baylor.box.com/s/l5b1s5ufkmd51pzethhme3shgchvvvkj

2. Box link to video of the game on the fast setting:
https://baylor.box.com/s/adenkbnsijlpyxvq7n43ewpriapzqsy7

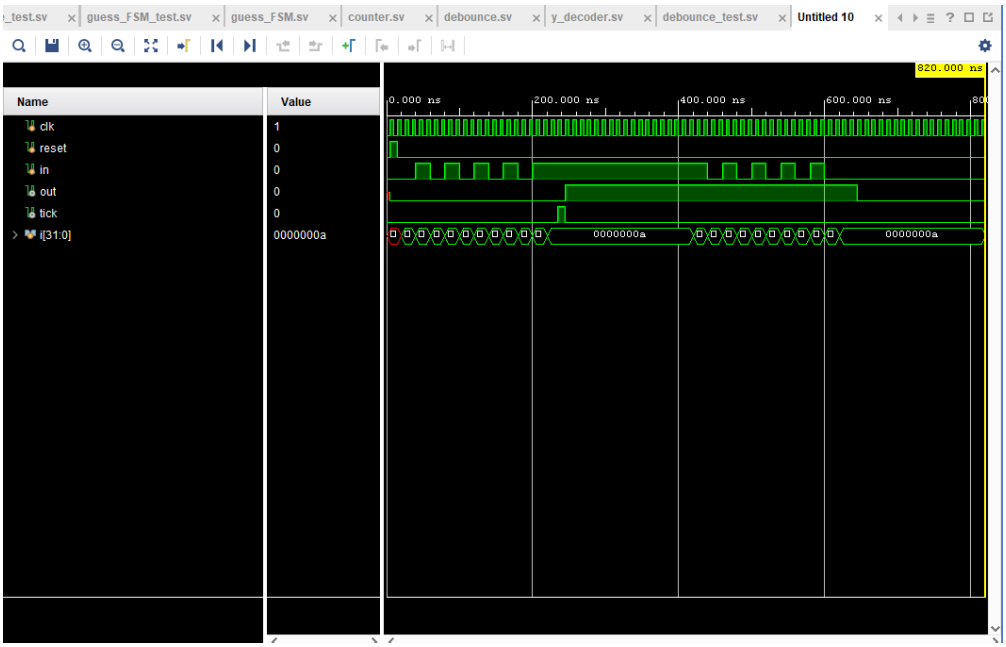
3. Game results on slow setting: 60% win percentage.

Game Number	Win/Lose	Segment Illuminated
1	Lose	Top
2	Win	Left
3	Lose	Right
4	Win	Right
5	Lose	Top
6	Lose	Bottom
7	Win	Right
8	Win	Right
9	Win	Left
10	Win	Top

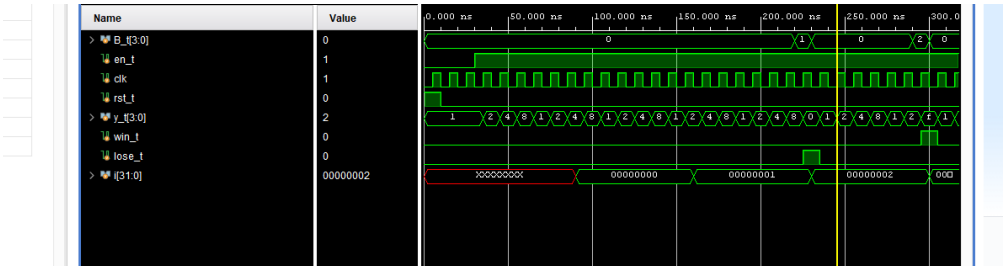
4. Game results on fast setting: 40% win percentage.

Game Number	Win/Lose	Segment Illuminated
1	Lose	Top
2	Lose	Bottom
3	Win	Right
4	Win	Left
5	Lose	Left
6	Lose	Top
7	Win	Bottom
8	Win	Right
9	Lose	Left
10	Lose	Top

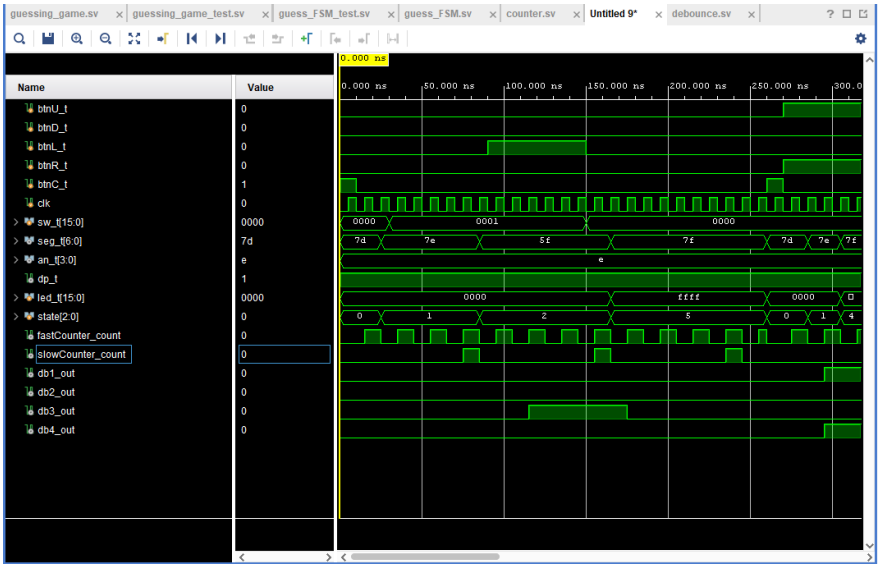
1. Debounce Test Simulation



2. Guess FSM Test Simulation



3. Guessing Game Test Simulation



Code

Listing 1: Counter

```
module counter #(parameter N=1)(
   input clk,
   input rst,
   output tic
   );

   wire [N-1:0] Qnext;
   wire [N-1:0] Qreg;
   assign Qnext = Qreg +1;

   register #(.N(N)) my_reg_in_count1(
      .D(Qnext),
      .clk(clk),
      .en(1),
      .rst(rst),
      .Q(Qreg)
   );

   assign tic = &Qreg;
   endmodule
```

Listing 2: Guess FSM

```
module guess_FSM(
   input [3:0] B,
   input en,
   input clk,
   input rst,
   output reg [3:0] y,
   output reg win,
   output reg lose,
   output reg [15:0] display
   );


   localparam s0    = 3'b000;
   localparam s1    = 3'b001;
   localparam s2    = 3'b010;
   localparam s3    = 3'b011;
   localparam slose = 3'b100;
   localparam swin  = 3'b101;

   // internal signals
   reg [2:0] state, state_next;

   // state memory (register)
   always_ff @(posedge(clk), posedge(rst))
      if (rst)
         begin
            state <= s0;
         end
```

```
    else if(en)
       begin
          state <= state_next;
       end


always @*
   case(state)
      s0:
         if(B == 4'b0000)
            state_next = s1;
         else if (B[3]==1 || B[2]==1 || B[1]==1)
            state_next = slose;
         else
            state_next = swin;

      s1:
         if(B == 4'b0000)
            state_next = s2;
         else if (B[3]==1 || B[2]==1 || B[0]==1)
            state_next = slose;
         else
            state_next = swin;

      s2:
         if(B == 4'b0000)
            state_next = s3;
         else if (B[3]==1 || B[1]==1 || B[0]==1)
            state_next = slose;
         else
            state_next = swin;

      s3:
         if(B == 4'b0000)
            state_next = s0;
         else if (B[2]==1 || B[1]==1 || B[0]==1)
            state_next = slose;
         else
            state_next = swin;

      swin:
         state_next = swin;
      slose:
         state_next = slose;
   endcase


always @*
   begin
      win = 1'b0;
      lose = 1'b0;
      display = 16'b0;
      case(state)
         s0: y = 4'b0001;
```

```
            s1: y = 4'b0010;
            s2: y = 4'b0100;
            s3: y = 4'b1000;
            swin:
               begin
               y = 4'b1111;
               win = 1'b1;
               lose = 1'b0;
               display = 16'b1111111111111111;
            end
               slose:
               begin
               y = 4'b0000;
               win = 1'b0;
               lose = 1'b1;
               display = 16'b1010101010101010;
            end
         endcase
      end

endmodule
```

Listing 3: Guess FSM Test

```
module guess_FSM_test();

   reg [3:0] B_t;
   reg en_t;
   reg clk;
   reg rst_t;
   reg [3:0] y_t;
   reg win_t;
   reg lose_t;
   integer i;

   guess_FSM #(.N(21)) dut(
      .B(B_t),
      .en(en_t),
      .clk(clk),
      .rst(rst_t),
      .y(y_t),
      .win(win_t),
      .lose(lose_t)
      );

   always begin
      clk = ~clk; #5;
   end

   initial begin
      clk = 0; rst_t = 1; en_t = 0; B_t = 4'b0000; #10;
      rst_t = 0; #20;
      en_t = 1; #60;
```

```
      for (i = 0; i <= 4'b1111; i = i + 1) begin
         B_t = 4'b0000;
         #60;
         B_t = i;
         #10;
      end

      B_t = 4'b0000;
      en_t = 0; #50;
      en_t = 1; #60;
      rst_t = 1; #10;
      rst_t = 0;

      for (i = 0; i <= 4'b1111; i = i + 1) begin
         B_t = 4'b0000;
         #50;
         B_t = i;
         #10;
      end

      B_t = 4'b0000;
      en_t = 0; #50;
      en_t = 1; #60;
      rst_t = 1; #10;
      rst_t = 0;

      for (i = 0; i <= 4'b1111; i = i + 1) begin
         B_t = 4'b0000;
         #70;
         B_t = i;
         #10;
      end

      B_t = 4'b0000;
      en_t = 0; #50;
      en_t = 1; #60;
      rst_t = 1; #10;
      rst_t = 0;

      for (i = 0; i <= 4'b1111; i = i + 1) begin
         B_t = 4'b0000;
         #80;
         B_t = i;
         #10;
      end
      $finish;
   end

endmodule
```

Listing 4: y decoder

```
module y_decoder(
   input [3:0] in,
```

```
   output reg [6:0] out
   );

   always @*
      case (in)
         4'b0001: out = 7'b1111101;
         4'b0010: out = 7'b1111110;
         4'b0100: out = 7'b1011111;
         4'b1000: out = 7'b0111111;
         default: out = 7'b1111111;
      endcase
endmodule
```

Listing 5: Guessing Game

```
module guessing_game #(parameter N = 26, D = 21)(
   input btnU,
   input btnD,
   input btnL,
   input btnR,
   input btnC,
   input clk,
   input [15:0] sw,
   output [6:0] seg,
   output [3:0] an,
   output dp,
   output reg [15:0] led
   );

   wire db1_out;
   wire db2_out;
   wire db3_out;
   wire db4_out;
   wire db1_tick;
   wire db2_tick;
   wire db3_tick;
   wire db4_tick;
   wire fastCounter_count;
   wire slowCounter_count;
   wire mux2_out;
   wire [3:0] guess_FSM_y;
   wire guess_FSM_win;
   wire guess_FSM_lose;

   debounce #(.N(D))db1(
      .clk(clk),
      .reset(btnC),
      .in(btnU),
      .out(db1_out),
      .tick(db1_tick)
   );

   debounce #(.N(D))db2(
      .clk(clk),
```

```
    .reset(btnC),
    .in(btnD),
    .out(db2_out),
    .tick(db2_tick)
);

debounce #(.N(D)) db3(
    .clk(clk),
    .reset(btnC),
    .in(btnL),
    .out(db3_out),
    .tick(db3_tick)
);

debounce #(.N(D)) db4(
    .clk(clk),
    .reset(btnC),
    .in(btnR),
    .out(db4_out),
    .tick(db4_tick)
);

counter #(.N(N-2)) fastCounter_game(
    .clk(clk),
    .rst(btnC),
    .tic(fastCounter_count)
);

counter #(.N(N)) slowCounter_game(
    .clk(clk),
    .rst(btnC),
    .tic(slowCounter_count)
);


mux2 #(.BITS(1)) mux2_guess(
    .in0(fastCounter_count),
    .in1(slowCounter_count),
    .sel(sw[0]),
    .out(mux2_out)
);

guess_FSM guess_FSM_game(
    .B({db2_out, db3_out, db1_out, db4_out}),
    .en(mux2_out),
    .clk(clk),
    .rst(btnC),
    .y(guess_FSM_y),
    .win(guess_FSM_win),
    .lose(guess_FSM_lose),
    .display(led)
);

y_decoder guess_y_decoder(
```

```
      .in(guess_FSM_y),
      .out(seg)
   );

   assign dp = 1'b1;
   assign an = 4'b1110;


endmodule
```

Listing 6: Guessing Game test

```
module guessing_game_test();

   reg btnU_t;
   reg btnD_t;
   reg btnL_t;
   reg btnR_t;
   reg btnC_t;
   reg clk;
   reg [15:0] sw_t;
   wire [6:0] seg_t;
   wire [3:0] an_t;
   wire dp_t;
   reg [15:0] led_t;

   guessing_game #(.N(3), .D(1)) dut(
      .btnU(btnU_t),
      .btnD(btnD_t),
      .btnL(btnL_t),
      .btnR(btnR_t),
      .btnC(btnC_t),
      .clk(clk),
      .sw(sw_t),
      .seg(seg_t),
      .an(an_t),
      .dp(dp_t),
      .led(led_t)
   );


   always begin
      clk = ~clk; #5;
   end

   initial begin
      clk = 0; btnC_t = 1; sw_t = 0; btnR_t = 0;
      btnU_t = 0; btnL_t = 0; btnD_t = 0; #10;
      btnC_t = 0; #20;
      sw_t = 1; #60;


      btnR_t = 0; btnU_t = 0;
      btnL_t = 1; btnD_t = 0; #60;
```

```
        btnR_t = 0; btnU_t = 0;
        btnL_t = 0; btnD_t = 0;
        sw_t = 0; #110;
        btnC_t = 1; #10;
        btnC_t = 0;

        btnR_t = 1; btnU_t = 1;
        btnL_t = 0; btnD_t = 0; #60;

        btnR_t = 0; btnU_t = 0;
        btnL_t = 0; btnD_t = 0;
        sw_t = 0; #50;
        sw_t = 1; #60;
        btnC_t = 1; #10;
        btnC_t = 0;

        btnR_t = 0; btnU_t = 1;
        btnL_t = 0; btnD_t = 0; #60;

        btnR_t = 0; btnU_t = 0;
        btnL_t = 0; btnD_t = 0;
        sw_t = 0; #50;
        sw_t = 1; #60;
        btnC_t = 1; #10;
        btnC_t = 0;

        btnR_t = 0; btnU_t = 0;
        btnL_t = 0; btnD_t = 0; #60;

        btnR_t = 0; btnU_t = 0;
        btnL_t = 0; btnD_t = 0;
        sw_t = 0; #50;
        sw_t = 1; #60;
        btnC_t = 1; #10;
        btnC_t = 0;

        btnR_t = 0; btnU_t = 0;
        btnL_t = 0; btnD_t = 0; #60;

        btnR_t = 0; btnU_t = 0;
        btnL_t = 1; btnD_t = 0;
        sw_t = 0; #50;
        sw_t = 1; #60;
        btnC_t = 1; #10;
        btnC_t = 0;

        btnR_t = 0; btnU_t = 0;
        btnL_t = 0; btnD_t = 0; #60;

        btnR_t = 0; btnU_t = 0;
        btnL_t = 0; btnD_t = 0;
        sw_t = 0; #50;
        sw_t = 1; #60;
```

```
      btnC_t = 1; #10;
      btnC_t = 0;

      btnR_t = 0; btnU_t = 0;
      btnL_t = 0; btnD_t = 1; #60;

      btnR_t = 0; btnU_t = 0;
      btnL_t = 0; btnD_t = 0;
      sw_t = 0; #50;
      sw_t = 1; #60;
      btnC_t = 1; #10;
      btnC_t = 0;

      btnR_t = 1; btnU_t = 0;
      btnL_t = 1; btnD_t = 1; #60;

      btnR_t = 0; btnU_t = 0;
      btnL_t = 0; btnD_t = 0;
      sw_t = 0; #50;
      sw_t = 1; #60;
      btnC_t = 1; #10;
      btnC_t = 0;
      $finish;
   end



endmodule
```