

# ELC 2137 Lab 11: FSM: Guessing Game

Ashlie Lackey

April 27, 2020

## Summary

This lab sought the construction of a finite state machine, FSM, to handle an irregular input from a user. We designed a synchronous FSM using ideas from both Mealy and Moore machines to create a guessing game. By first understanding how to "debounce" signals that may have an irregular input, the first step is taken to create an FSM. By the end of the experiment, participants have the skills to explain the role of finite state machines in digital systems design, explain the difference between a Mealy output and a Moore output, and implement a state machine in Verilog.

## Q&A

1. At what time in the simulation did the debounce circuit reach each of the four states(zero, wait1, one, wait0)?

The simulation reached state wait1 at 200ns, one at 245ns, wait0 at 600ns, and zero at 645ns.

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

This game cannot be implemented with regular sequential logic because the input from the user is going to be irregular, and regular sequential logic cannot handle this. Therefore, an FSM is used to handle the irregular, non-repeating input that can be expected from the user.

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

A Moore output was used for this design because at each state, the output was only dependent in the current state, not the present state and present input. This can be seen as at each "s" state, the y-value was set independent of the input. Additionally, with the swin and close states, they set either win or lose to 1 when they were at that state, not dependent on how they got to the swin or close states.

## Results

Board testing, expected results table, and simulation waveforms are included below

### Game Play Results on Two Settings

Fast:(20%) Lose, Win, Lose, Win, Lose, Lose, Lose, Lose, Lose

Slow: (40%) Win, Lose, Lose, Lose, Win, Win, Lose, Lose, Win, Lose



Figure 1: *fast test lose* display



Figure 2: *fast test win* display

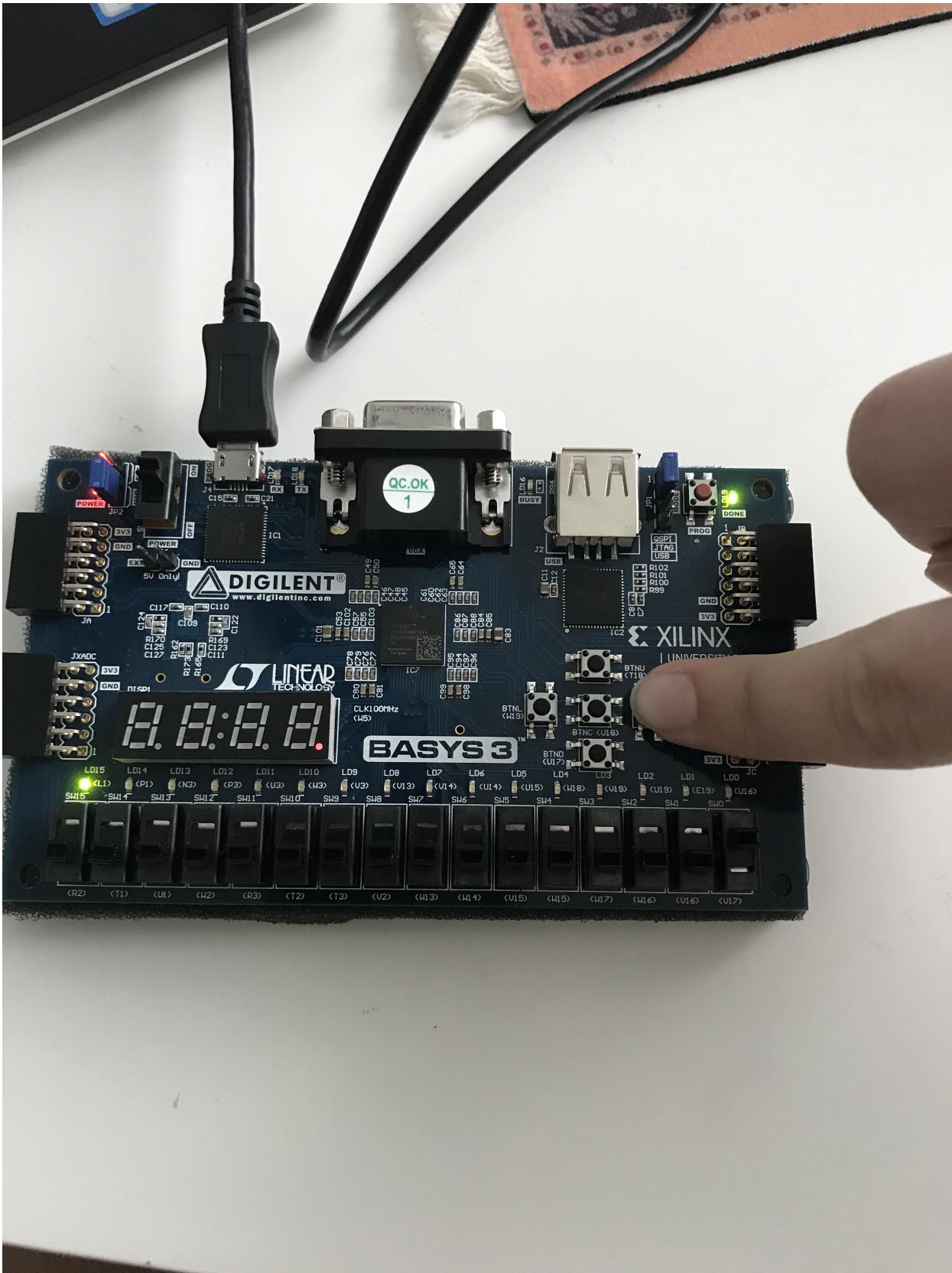


Figure 3: *slow test win* display



Figure 4: *slow test lose display*

## Expected results tables

When clock signal continue at the rate of 5ns on and off, a “-” symbol is used to show continuation of the clock signal in time ranges greater than 5ns. This symbol will also be used to show the continuation of the segment display pattern in the guessing-game ERT.

Table 1: *debounce* expected results table

Time (ns):	0-5	5-10	10-15	15-20	20-40	40-60	60-80	80-100	100-120
clk	0	1	0	1	-	-	-	-	-
reset	0	1	1	0	0	1	0	0	0
in	0	0	0	0	0	0	0	1	0
out	X	0	0	0	0	1	0	0	0
tick	0	0	0	0	0	0	0	0	0
i(hex)	X	X	X	X	0	1	2	3	4
Time (ns):	120-140	140-160	160-180	180-200	200-220	220-235	235-245	245-420	420-440
clk	-	-	-	-	-	-	-	-	-
reset	0	0	0	0	0	0	0	0	0
in	1	0	1	0	1	1	1	1	1
out	0	0	0	0	0	0	0	1	1
tick	0	0	0	0	0	1	0	0	0
i(hex)	5	6	7	8	9	a	a	a	0
Time (ns):	440-460	460-480	480-500	500-520	520-540	540-560	560-580	580-600	600-620
clk	-	-	-	-	-	-	-	-	-
reset	0	0	0	0	0	0	0	0	0
in	0	1	0	1	0	1	0	1	0
out	1	1	1	1	1	1	1	1	1
tick	0	0	0	0	0	0	0	0	0
i(hex)	1	2	3	4	5	6	7	8	9
Time (ns):	620-645	645-							
clk	-	-							
reset	0	0							
in	0	0							
out	0	0							
tick	0	0							
i(hex)	a	a							

Table 2: *guess\_FSM* expected results table

Time (ns):	0-5	5-10	10-15	15-20	20-25	25-30	30-35	35-40	40-45
clk	0	1	0	1	0	1	0	1	0
reset	0	1	0	0	0	0	0	0	0
b[3:0]	0	0	0	0	1	1	0	0	0
y[3:0]	X	1	1	2	2	2	2	1	1
win	X	0	0	0	0	0	0	0	0
lose	X	0	0	0	1	1	1	0	0
Time (ns):	45-50	50-55	55-60	60- 65	65-70	70-75	75-80	80-85	85-90
clk	1	0	1	0	1	0	1	0	1
reset	0	0	0	0	0	0	0	0	0
b[3:0]	2	2	0	0	0	0	0	0	0
y[3:0]	2	2	2	2	1	1	2	2	4
win	0	0	1	1	0	0	0	0	0
lose	0	0	0	0	0	0	0	0	0
Time (ns):	90-95	95-100	100-105	105-110	110-115	115-125	125-135	135-145	145-155
clk	0	1	0	1	0	-	-	-	-
reset	0	0	0	0	0	0	0	0	0
b[3:0]	4	4	1	0	0	0	0	8	0
y[3:0]	4	4	4	1	1	2	4	8	8
win	0	1	1	0	0	0	0	1	1
lose	0	0	0	0	0	0	0	0	0
Time (ns):	155-160	160-165	165-170	170-175	175-185	185-190	190-195	195-200	200-205
clk	1	0	1	0	-	1	0	1	0
reset	0	0	0	0	0	0	0	0	0
b[3:0]	0	2	2	0	0	0	4	4	0
y[3:0]	0	0	1	1	0	2	2	2	2
win	0	0	0	0	0	0	0	0	0
lose	0	0	1	1	0	0	0	1	1
Time (ns):	205-210	210-215	215-220	220-225	225-230	230-235	235-240		
clk	1	0	1	0	1	0	1		
reset	0	0	0	0	0	0	0		
b[3:0]	0	8	8	0	0	1	1		
y[3:0]	1	1	1	1	0	1	1		
win	0	0	0	0	0	0	1		
lose	0	0	1	1	0	0	0		

Table 3: *ggame\_test* (*guessing-game test*) expected results table

Time (ns):	0-5	5-10	10-15	15-25	25-30	30-35	35-45	45-55	55-65
btnU	0	0	0	0	0	0	0	0	0
btnR	0	0	0	0	0	0	0	0	0
btnD	0	0	0	0	1	1	1	1	1
btnL	0	0	0	0	0	0	0	0	0
btnC	0	1	0	0	0	0	0	0	0
clk	0	1	0	-	1	0	-	-	-
sw[15:0]	0000	0000	0000	0000	0000	0000	0000	0000	0000
seg[6:0]	X	7e	7e	7d	5f	5f	3f	7e	7d
an[3:0]	e	e	e	e	e	e	e	e	e
led[15:0]	X	0	0	0	0	0	0	0	0
Time (ns):	65-75	75-125	125-175	175-225	225-230	230-265	265-275	275-325	325-375
btnU	0	0	0	0	1	1	1	1	0
btnR	0	0	0	0	0	0	0	0	0
btnD	1	1	0	0	0	0	0	0	0
btnL	0	0	0	0	0	0	0	0	0
btnC	0	0	0	0	0	0	0	0	0
clk	-	-	-	-	1	-	-	-	-
sw[15:0]	0000	0000	0000	0000	0000	0000	0000	0000	0000
seg[6:0]	5f	5f	5f	-	7d	-	7d	7d	7d
an[3:0]	e	e	e	e	e	e	e	e	e
led[15:0]	0000	8000	8000	0000	0000	0000	0000	0001	0001
Time (ns):	375-425	425-465	465-475	475-525	525-575	575-625	625-665	665-675	675-725
btnU	0	0	0	0	0	0	1	1	1
btnR	0	0	0	0	0	0	0	0	0
btnD	0	0	0	0	0	0	0	0	0
btnL	0	1	1	1	0	0	0	0	0
btnC	0	0	0	0	0	0	0	0	0
clk	-	-	-	-	-	-	-	-	-
sw[15:0]	0000	0000	0000	0000	0000	0000	0000	0000	0000
seg[6:0]	-	-	7d	7d	7d	-	-	7d	7d
an[3:0]	e	e	e	e	e	e	e	e	e
led[15:0]	0000	0000	0000	0001	0001	0000	0000	0000	8000
Time (ns):	725-775	775-825	825-						
btnU	0	0	0						
btnR	0	0	0						
btnD	0	0	0						
btnL	0	0	0						
btnC	0	0	0						
clk	-	-	-						
sw[15:0]	0000	0000	0001						
seg[6:0]	7d	7e	7e						
an[3:0]	e	e	e						
led[15:0]	8000	0000	0000	8					

## Simulation Waveforms

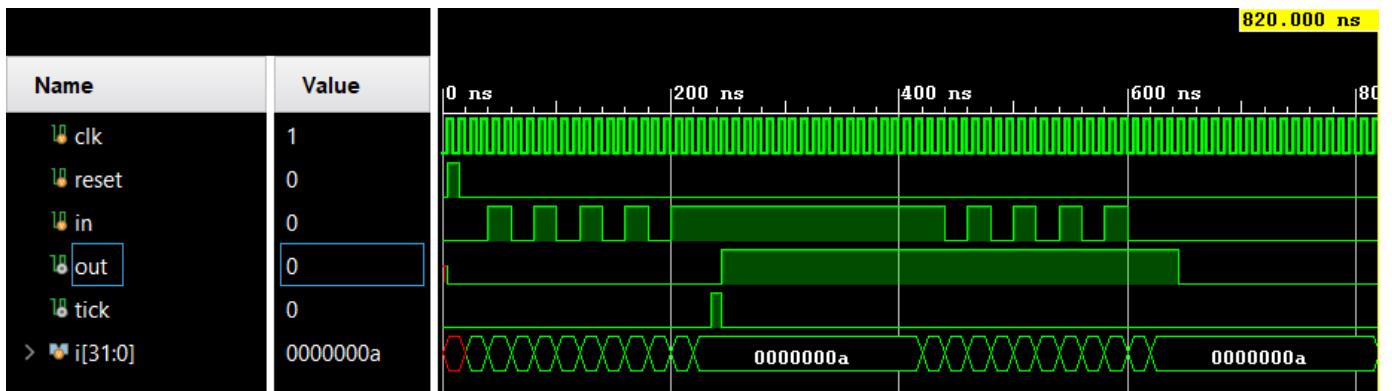


Figure 5: *debounce testbench* Simulation Waveform

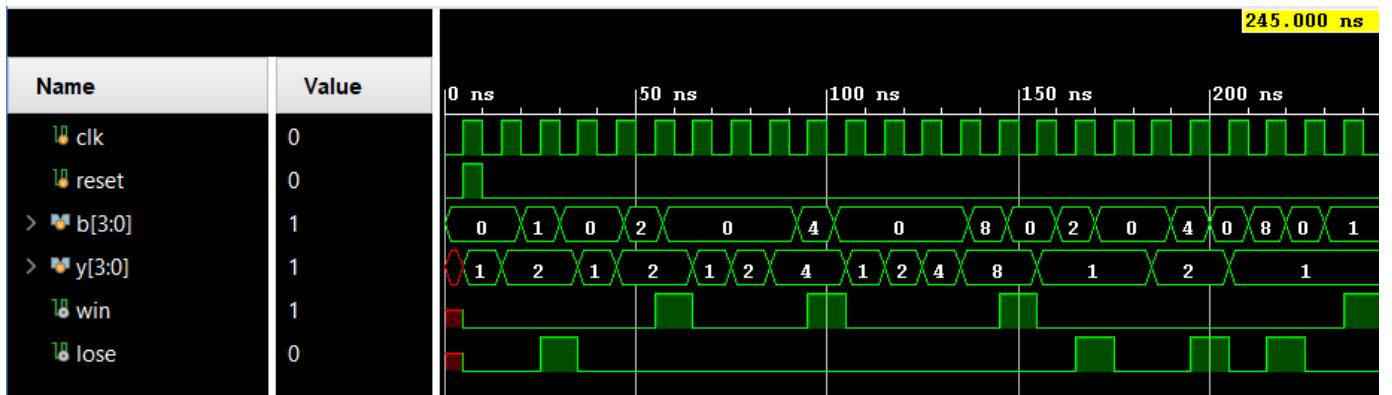


Figure 6: *guess\_FSM testbench* Simulation Waveform

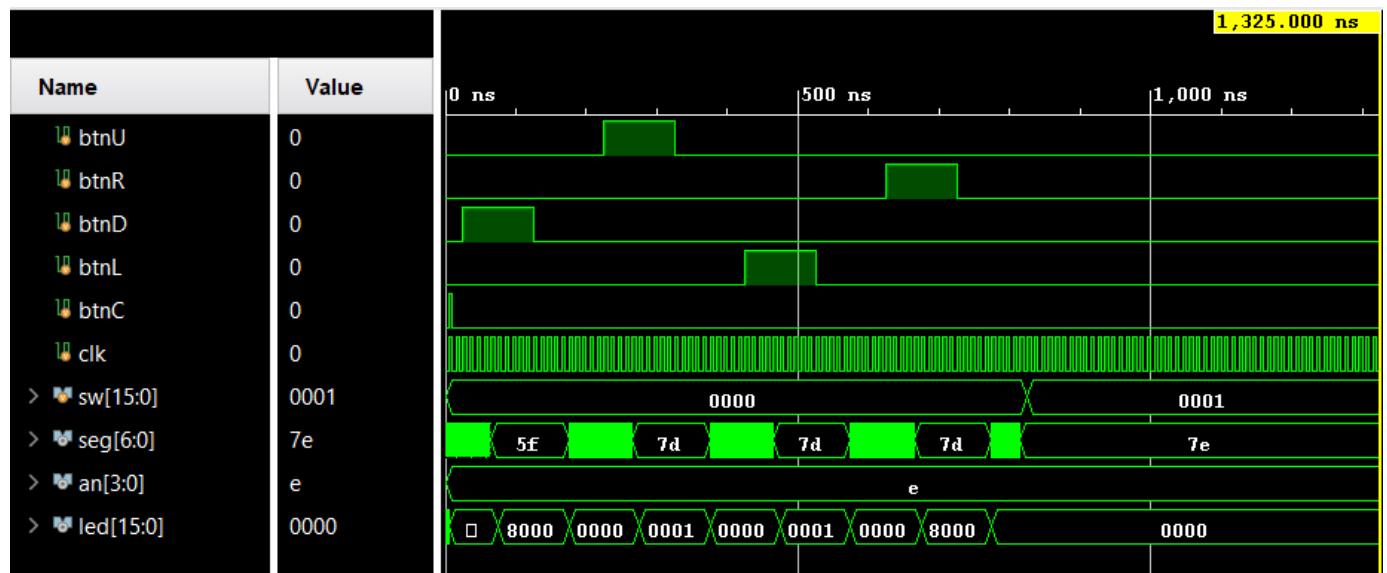


Figure 7: *guessing-game testbench* Simulation Waveform

## Code

Listing 1: guess\_FSM Verilog Code

```
'timescale 1ns / 1ps
// ELC 2137, Ashlie Lackey, 2020 -04 -21
module guess_FSM#(parameter N = 21)(input clk, reset,
    input reg [3:0] b,
    output reg [3:0] y,
    output reg win,
    output reg lose
);

localparam [2:0]
    s0 = 3'b000,
    s1 = 3'b001,
    s2 = 3'b010,
    s3 = 3'b011,
    swin = 3'b100,
    slose = 3'b101;

reg [2:0] state, state_next;

always_ff @(posedge clk or posedge reset)
    if(reset) begin
        state <= s0;
    end
    else begin
        state <= state_next;
    end

always_comb begin
    // default behavior
    state_next = state;
    //y = 4'b0000;

    case (state)
        s0: begin
            win = 0;
            lose = 0;
            y = 4'b0001;
            if (~b[3] & ~b[2] & ~b[1] & b[0])
                state_next = swin;
            else if (b[3] | b[2] | b[1])
                state_next = slose;
            else if (~b[3] & ~b[2] & ~b[1] & ~b[0])
                state_next = s1;
        end

        s1: begin
            y = 4'b0010;
            if (~b[3] & ~b[2] & b[1] & ~b[0])
                state_next = swin;
            else if (b[3] | b[2] | b[0])

```

```

        state_next = slose;
    else if (~b[3] & ~b[2] & ~b[1] & ~b[0])
        state_next = s2;
    end

    s2: begin
        y = 4'b0100;
        if (~b[3] & b[2] & ~b[1] & ~b[0])
            state_next = swin;
        else if (b[3] | b[1] | b[0])
            state_next = slose;
        else if (~b[3] & ~b[2] & ~b[1] & ~b[0])
            state_next = s3;
    end

    s3: begin
        y = 4'b1000;
        if (b[3] & ~b[2] & ~b[1] & ~b[0])
            state_next = swin;
        else if (b[2] | b[1] | b[0])
            state_next = slose;
        else if (~b[3] & ~b[2] & ~b[1] & ~b[0])
            state_next = s0;
    end

    swin: begin
        win = 1;
        if (~b[3] & ~b[2] & ~b[1] & ~b[0])
            state_next = s0;
        else if (b[3] | b[2] | b[1] | b[0])
            state_next = swin;
    end

    slose: begin
        lose = 1;
        if (~b[3] & ~b[2] & ~b[1] & ~b[0])
            state_next = s0;
        else if (b[3] | b[2] | b[1] | b[0])
            state_next = slose;
    end
    endcase
end
endmodule

```

Listing 2: guessing\_game Verilog Code

```

'timescale 1ns / 1ps
// ELC 2137, Ashlie Lackey, 2020 -04 -21
module guessing_game(input btnU, btnR, btnD, btnL, btnC,
    input clk,
    input [15:0] sw,
    output [6:0] seg,
    output [3:0] an,
    output [15:0] led);

```

```

wire [3:0] dbo;
wire t1, t2, t3, t4;
debounce #(N(2)) db1(.clk(clk) , .reset(btnC), .in(btnU), .out(dbo[0])
, .tick(t1));
debounce #(N(2)) db2(.clk(clk) , .reset(btnC), .in(btnR), .out(dbo[1])
, .tick(t2));
debounce #(N(2)) db3(.clk(clk) , .reset(btnC), .in(btnD), .out(dbo[2])
, .tick(t3));
debounce #(N(2)) db4(.clk(clk) , .reset(btnC), .in(btnL), .out(dbo[3])
, .tick(t4));

wire newclock;
wire countdc;
counter #(N(24)) counter1 (.clk(clk), .rst(btnC), .en(1) ,.count(
countdc),.tick(newclock));

wire muxout;
mux2_4b (.in0(clk),.in1(newclock), .sel(sw[0]),.out(muxout));

wire [3:0] yout;
guess_FSM #(N(4)) topmod (.clk(muxout), .reset(btnC), .b(dbo), .y(yout)
, .win(led[15]), .lose(led[0]));

assign seg[0] = ~yout[0];
assign seg[1] = ~yout[1];
assign seg[5] = ~yout[2];
assign seg[6] = ~yout[3];
assign seg[4:2] = 3'b111;
assign an[3:1] = 3'b111;
assign an[0] = 0;
assign led[14:1] = 0;

endmodule

```

Listing 3: guess\_FSM testbench Verilog Code

```

'timescale 1ns / 1ps
// ELC 2137, Ashlie Lackey, 2020 -04 -21
module guess_FSM_test();

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

guess_FSM #(N(4)) gFSM (.clk(clk), .reset(reset),.b(b), .y(y), .win(
win), .lose(lose));

always begin
#5 clk = ~clk;
end

initial begin

```

```

clk =0;    reset =0;  b=4'b0000; #5;
reset =1;   #5;
reset =0;   #10;

b = 4'b0001; #10;
b = 4'b0000; #17;

b = 4'b0010; #10;
b=4'b0000; #35;

b = 4'b0100; #10;
b = 4'b0000; #35;

b = 4'b1000; #10;
b = 4'b0000; #13;

b = 4'b0010; #10;
b = 4'b0000; #20;

b = 4'b0100; #10;
b = 4'b0000; #10;

b = 4'b1000; #10;
b = 4'b0000; #10;
b = 4'b0001; #15;
$finish;
end
endmodule

```

---

Listing 4: guessing\_game testbench Verilog Code

```

`timescale 1ns / 1ps
// ELC 2137,  Ashlie Lackey, 2020 -04 -21
module ggame_test();

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

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

always begin
#5 clk = ~clk;
end

initial begin
clk =0;  btnC =0; btnU = 0; btnR = 0;
btnD = 0; btnL = 0;
sw[15:0] = 16'b0000000000000000; #5;

```

```
btnC =1; #5;
btnC =0; #5;

btnD = 0; #10;
btnD = 1; #100;

btnD = 0; #100;

btnU = 1; #100;

btnU = 0; #100;

btnL = 1; #100;

btnL = 0; #100;

btnR = 1; #100;

btnR = 0; #100

sw[0] = 1'b1; #500;

$finish;
end

endmodule
```

---