ELC 2137 Lab 1: FSM: Guessing Game

Spencer Stinson

November 24, 2020

Summary

In this lab, we learned how to use state diagrams as a guideline to write a source code in Verilog. Doing this helped us to show our knowledge of both how state diagrams work, as well as how verliog works as we integrated the two together. We used moore machine design to implement this and it allowed us to create this game. By applying the knowldge we learned in class, we helped to make ourselves remember this, and to figure out the things that we may not have known in its entirety.

Q&A

- At what time in the simulation did the debounce circuit reach each of the four state? (zero, wait1, one, wait0) In my simulation, the values for the zero is from 0-40 ns, for wait1 is from 220-420, for one is from 600-620, for wait 0 from 620-800.
- Why can this game not be implemented with regular sequential logic? Regular sequential logic cannot be used because many of these need to happen simultaneously, and state changes will happen too many times if we use sequential logic.
- What type of outputs did you use for your design(Mealy or Moore)? This machine is a Moore machine becuase the output depends on the state.

Results

Code

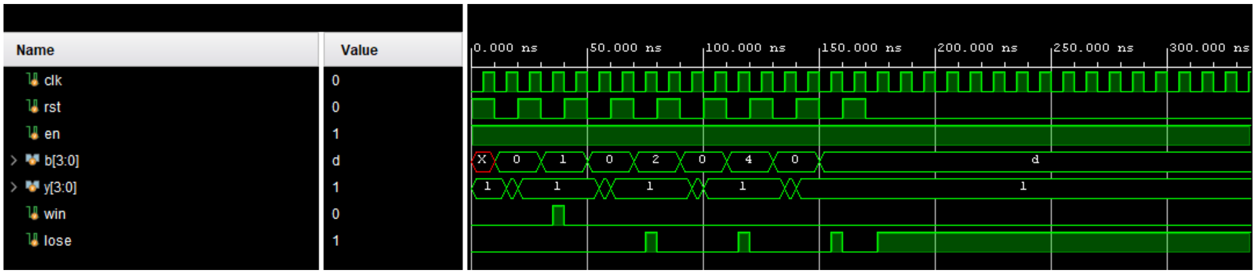
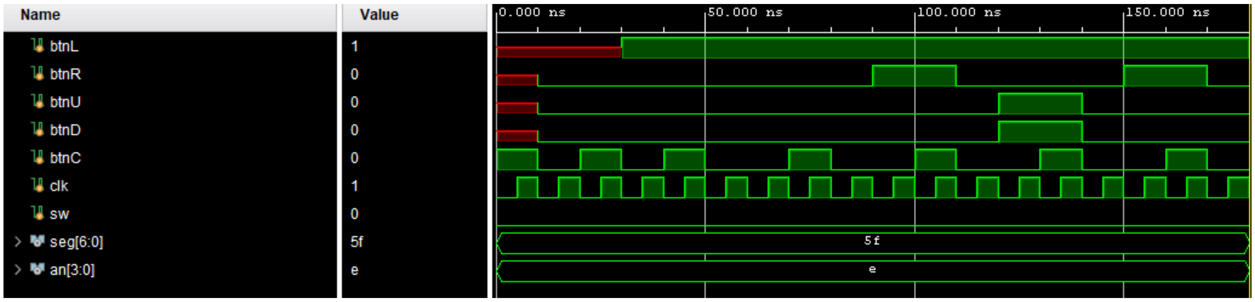
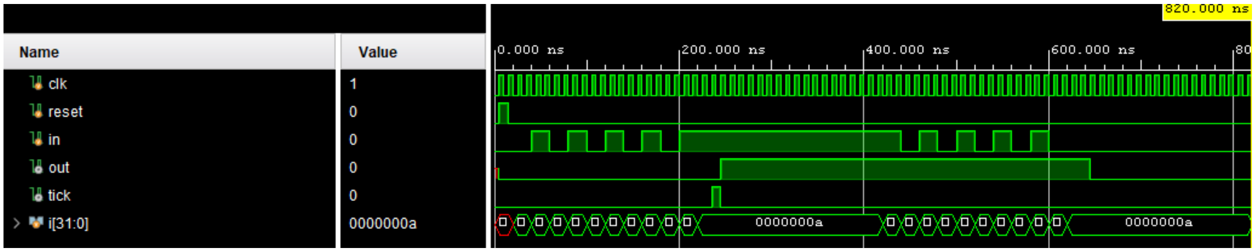


Figure 1: Guess FSM waveform



Name	Value
btnL	1
btnR	0
btnU	0
btnD	0
btnC	0
clk	1
sw	0
seg[6:0]	5f
an[3:0]	e

Figure 2: Guessing game waveform



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

Figure 3: Debounce waveform

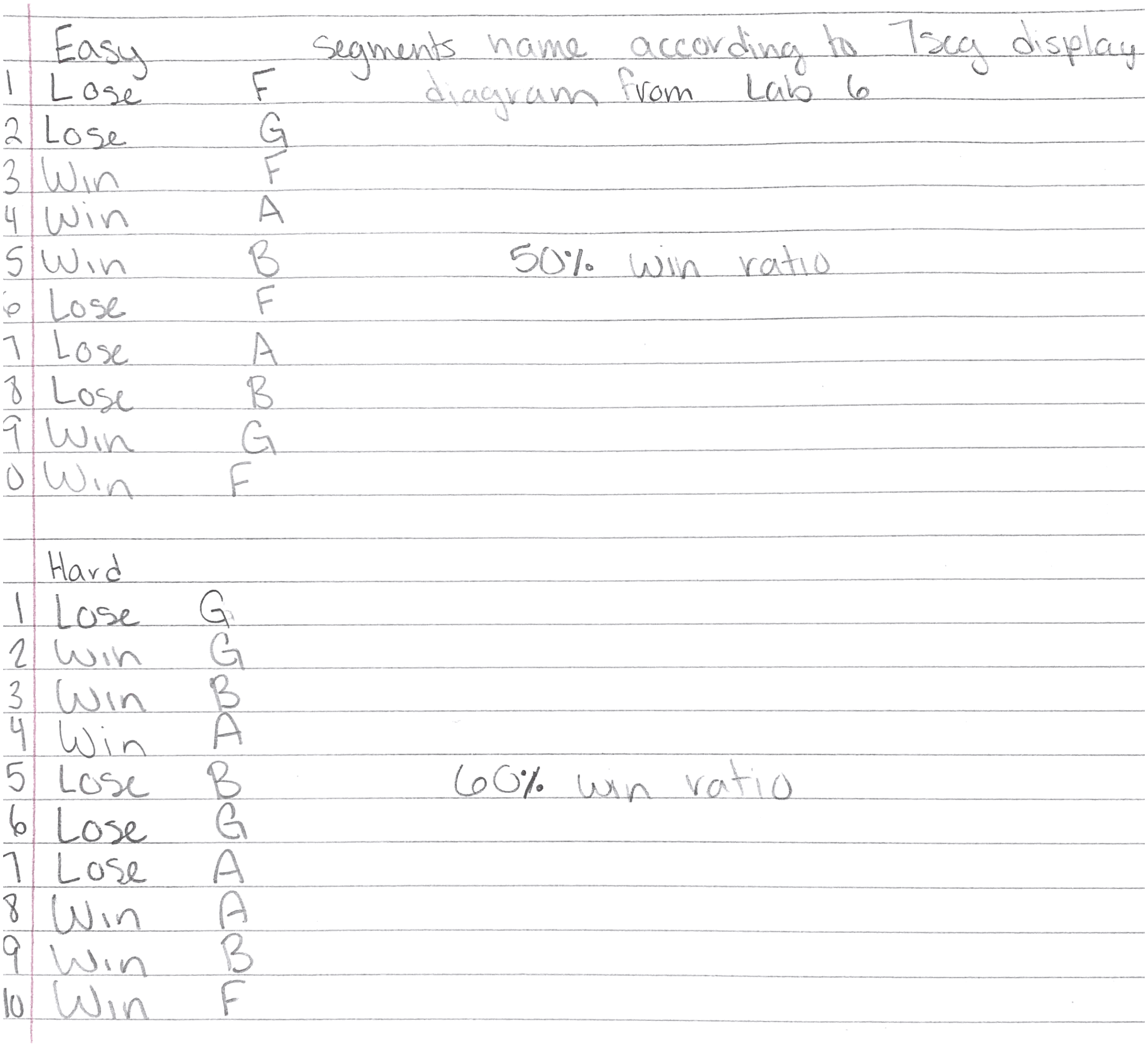
Listing 1: Counter module

```
`timescale 1ns / 1ps
//
   ///////////////////////////////////////////////////////////////////////////////

// Company:
// Engineer:
//
// Create Date: 11/12/2020 11:20:14 AM
// Design Name:
// Module Name: counter
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   ///////////////////////////////////////////////////////////////////////////////



module counter #(parameter N =21)  (
    input clk,
    input rst,
```



Easy

segments name according to 7seg display diagram from Lab 6

1	Lose	F
2	Lose	G
3	Win	F
4	Win	A
5	Win	B
6	Lose	F
7	Lose	A
8	Lose	B
9	Win	G
0	Win	F

50% win ratio

Hard

1	Lose	G
2	Win	G
3	Win	B
4	Win	A
5	Lose	B
6	Lose	G
7	Lose	A
8	Win	A
9	Win	B
10	Win	F

60% win ratio

Figure 4: Game results table

```
    output out,
    output out2
    );
    reg [N-1:0] q;
    reg [N-1:0] qnext;

    always @(posedge(clk), posedge(rst)) begin
    if (rst)
        q <= 0 ;
    else
        q <= qnext;
    end

    always @*
        qnext = q +1;

    assign out  =& q[N-1:0];
    assign out2 =& q[N-2:0];

endmodule
```

Listing 2: Display setup module

```
'timescale 1ns / 1ps
//
   //////////////////////////////////////////////////////////////////////////////

// Company:
// Engineer:
//
// Create Date: 11/23/2020 07:56:25 PM
// Design Name:
// Module Name: sseg_game
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   //////////////////////////////////////////////////////////////////////////////



module sseg_game(
input win,
input lose,
input [3:0] state,
output reg [6:0] seg
```

```
    );

always @* begin
    if (win)
    seg = 7'b0001000;
    else if (lose)
    seg = 7'b1000111;
    else if (state[3])
    seg = 7'b1111110;
    else if (state[2])
    seg = 7'b1111101;
    else if (state[1])
    seg = 7'b0111111;
    else
    seg = 7'b1011111;
    end
















endmodule
```

Listing 3: Guess FSM module

```
`timescale 1ns / 1ps
//
   //////////////////////////////////////////////////////////////////////////////

// Company:
// Engineer:
//
// Create Date: 11/12/2020 11:16:50 AM
// Design Name:
// Module Name: guess_FSM
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
```

```
//
//
   ///////////////////////////////////////////////////////////////////////////////



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

    parameter slose = 3’b100;
    parameter swin = 3’b101;
    parameter s0 = 3’b000;
    parameter s1 = 3’b001;
    parameter s2 = 3’b010;
    parameter s3 = 3’b011;

    reg [2:0] NS, ST;

always @(posedge(clk), posedge(rst))
    if (rst)
        ST <= s0;
    else if (en)
        ST <= NS;
always @*
    case(ST)
       swin: if ( b == 4’b0000)
         NS = s0;
            else
         NS = swin;

       slose: if (b == 4’b0000)
         NS = s0;
            else
         NS = slose;

       s0: if  (b == 4’b0001)
         NS = swin;
           else if (b == 4’b0000)
         NS = s1;
            else
         NS = slose;

       s1: if (b == 4’b0010)
        NS = swin;
            else if  (b == 4’b0000)
       NS = s2;
            else
       NS = slose;

       s2: if (b == 4'b0100)
        NS = swin;
            else if (b == 4'b0000)
        NS = s3;
            else
       NS = slose;

       s3: if (b == 4'b1000)
       NS = swin;
            else if (b == 4'b0000)
      NS = s0;
            else
      NS = slose;
       default: NS = s0;
endcase
    always @* begin
        win = 0;
        lose = 0;
        y = 4'b0001;
     case(ST)
        swin: win = 1'b1;
        slose: lose = 1'b1;
        s0: y = 4'b0001;
        s1: y = 4'b0010;
        s2: y = 4'b0100;
        s3: y = 4'b1000;
    endcase
    end

endmodule
```

Listing 4: Guessing game Module

```
`timescale 1ns / 1ps
//
   ////////////////////////////////////////////////////////////////////////////////

// Company:
// Engineer:
//
// Create Date: 11/23/2020 06:46:39 PM
// Design Name:
// Module Name: guessing_game
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   ///////////////////////////////////////////////////////////////////////////////



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

    reg [3:0] deb_out;
    reg [3:0] tick_out;
    reg f;
    reg s;
    reg w;
    reg l;
    reg [3:0] out_g;

    debounce debU (
    .clk(clk),
    .reset(btnC),
    .in(btnU),
    .out(deb_out[3]),
    .tick(tick_out[3])
    );

    debounce debR (
    .clk(clk),
    .reset(btnC),
    .in(btnR),
    .out(deb_out[2]),
    .tick(tick_out[2])
    );

    debounce debD (
    .clk(clk),
    .reset(btnC),
    .in(btnD),
    .out(deb_out[1]),
    .tick(tick_out[1])
    );

    debounce debL (
    .clk(clk),
    .reset(btnC),
    .in(btnL),
    .out(deb_out[0]),
    .tick(tick_out[0])
    );

    counter #(.N(25)) c_guess(
    .clk(clk),
    .rst(btnC),
    .out(s),
    .out2(f)
    );

    assign game_lvl = sw ? f : s;

    guess_FSM fsm_game(
     .b(deb_out),
    .clk(clk),
    .rst(btnC),
    .en(game_lvl),
    .win(w),
    .lose(l),
    .y(out_g)
    );

    sseg_game disp_game(
    .win(w),
    .lose(l),
    .state(out_g),
    .seg(seg)
    );

    assign an[0] = 0;
    assign an[1] = 1;
    assign an[2] = 1;
    assign an[3] = 1;


endmodule
```

Listing 5: Guess FSM testbench

```
`timescale 1ns / 1ps
//
   //////////////////////////////////////////////////////////////////////////////

// Company:
// Engineer:
//
// Create Date: 11/23/2020 06:46:39 PM
// Design Name:
// Module Name: guessing_game
// Project Name:
// Target Devices:
// Tool Versions:
```

```
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   ////////////////////////////////////////////////////////////////////////////////



module guess_FSM_test ();

reg clk, rst, en;
reg [3:0] b, y;
reg win, lose;


guess_FSM dut(
    .b(b),
    .clk(clk),
    .rst(rst),
    .en(en),
    .win(win),
    .lose(lose),
    .y(y)
);
always begin
clk = ~clk; #5;
end

initial begin
en = 1;
clk = 0; rst = 1; #10; rst = 0;
b= 4’b0000; #10; rst = 1; #10; rst = 0;
b= 4’b0001; #10; rst = 1; #10; rst = 0;
b= 4’b0000; #10; rst = 1; #10; rst = 0;
b= 4’b0010; #10; rst = 1; #10; rst = 0;
b= 4’b0000; #10; rst = 1; #10; rst = 0;
b= 4’b0100; #10; rst = 1; #10; rst = 0;
b= 4’b0000; #10; rst = 1; #10; rst = 0;
b= 4’b1101; #10; rst = 1; #10; rst = 0;
end
endmodule
```

Listing 6: Game Testbench

```
‘timescale 1ns / 1ps
//
   ////////////////////////////////////////////////////////////////////////////////

// Company:
```

```
// Engineer:
//
// Create Date: 11/23/2020 06:46:39 PM
// Design Name:
// Module Name: guessing_game
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
   ///////////////////////////////////////////////////////////////////////////////


module guess_FSM_test ();

reg clk, rst, en;
reg [3:0] b, y;
reg win, lose;


guess_FSM dut(
    .b(b),
    .clk(clk),
    .rst(rst),
    .en(en),
    .win(win),
    .lose(lose),
    .y(y)
);
always begin
clk = ˜clk; #5;
end

initial begin
en = 1;
clk = 0; rst = 1; #10; rst = 0;
b= 4’b0000; #10; rst = 1; #10; rst = 0;
b= 4’b0001; #10; rst = 1; #10; rst = 0;
b= 4’b0000; #10; rst = 1; #10; rst = 0;
b= 4’b0010; #10; rst = 1; #10; rst = 0;
b= 4’b0000; #10; rst = 1; #10; rst = 0;
b= 4’b0100; #10; rst = 1; #10; rst = 0;
b= 4’b0000; #10; rst = 1; #10; rst = 0;
b= 4’b1101; #10; rst = 1; #10; rst = 0;
end
endmodule
```