ELC 2137 Lab 11: FSM: Guessing Game

Ashlie Lackey

April 23, 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)?
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.

Results

Expected results table, simulation waveforms, and schematic drawings are included in this portion of the report.

Game Play Results on Two Settings

Fast:(20%)

Slow: (40%)

Expected results tables

Table 1: *debounce* expected results table

Time (ns):	0-5	5-15	15-40	40-60	60-80	80-100	100-120	120-140	140-160	160-180	180-200	200
clk	0	1	1	0	1	0	1	0	1	0	1	
rst	0	0	0	1	0	1	0	1	0	1	0	
in	0	0	1	0	0	0	0	0	0	0	0	
out	X	X	0	0	1	1	2	2	3	3	0	
tick	X	X	0	0	0	0	0	0	1	1	0	

Table 2: *guess_FSM* expected results table

Time (ns):	0-5	5-7	7-10	10-15	15-20	20-25	25-30	30-35	35-40	40-45	45-50	...
clk	0	1	1	0	1	0	1	0	1	0	1	...
en	0	0	0	1	0	1	0	1	0	1	0	...
rst	0	0	1	0	0	0	0	0	0	0	0	...
count	X	X	0	0	1	1	2	2	3	3	0	...
tick	X	X	0	0	0	0	0	0	1	1	0	...

Table 3: *ggame_test* expected results table

Time (ns):	0-2		2-5	5-10	...	1000005-2000000	2000000-2621435	2621435-3000005
data(hex)	1234	1234	123		...	1234	1234	1234
hex_dec	0		0	0	...	1	0	0
sign	0		0	0	...	0	1	1
reset	0		1	0	...	0	0	0
clock	0		0	5	...			
seg (hex)	X		19	19	...	19	19	30
dp	1		1	1	...	1	1	1
an (hex)	X		e	e	...	e	e	d

Simulation Waveforms

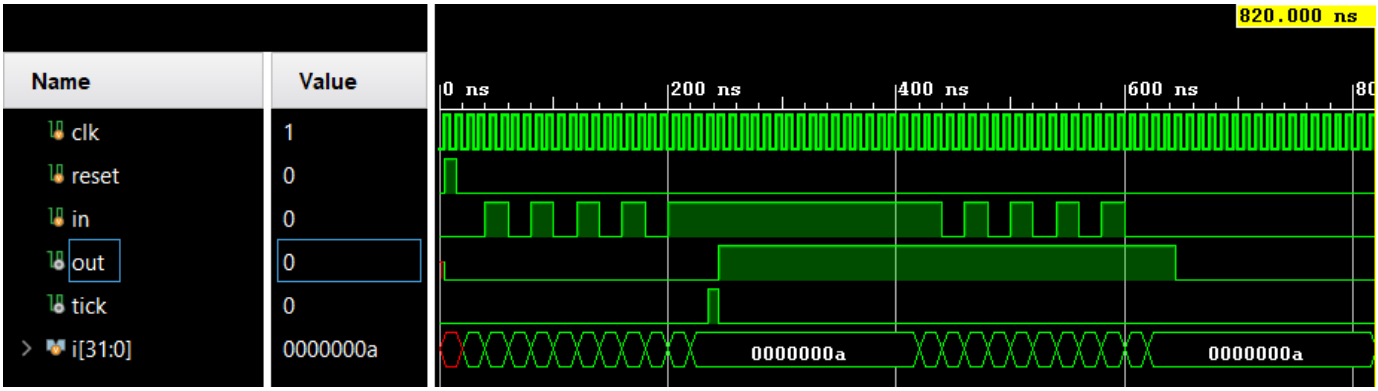

Figure 1: *debounce testbench* Simulation Waveform

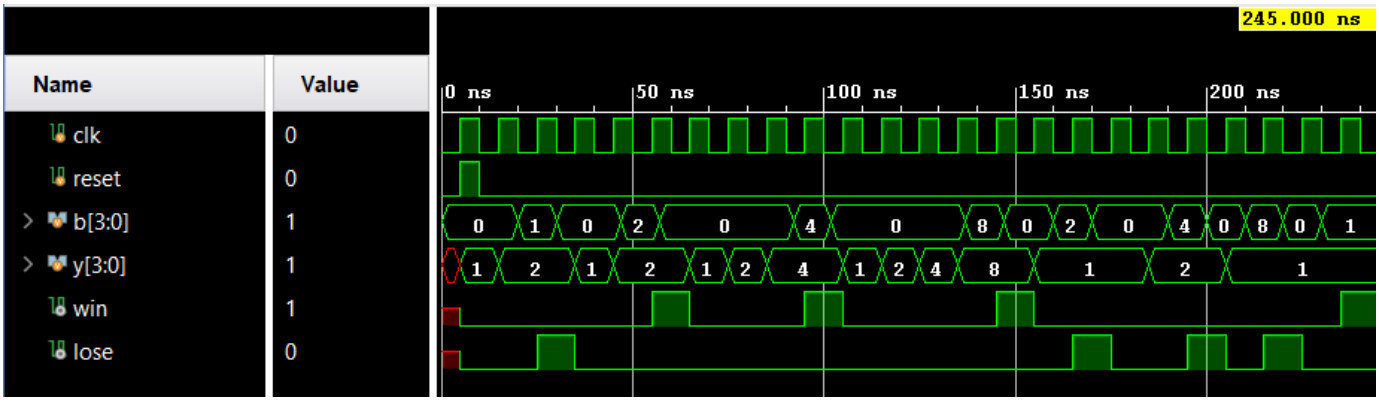

Figure 2: *guess_FSM testbench* Simulation Waveform

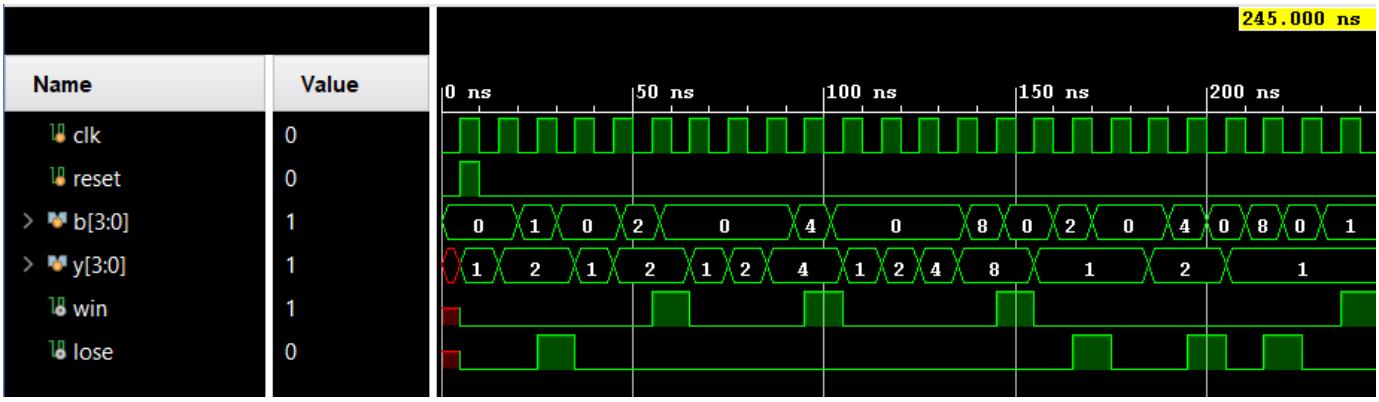

Figure 3: *guessing_game testbench* Simulation Waveform

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