ELC 2137 Lab 11: Lab 11 FSM: Guessing Game

Makenna Meyers

April 23, 2020

Summary

This lab uses a Finite State Machine (FSM) to develop a game in which irregular, non-repeating conditions are used. The code used can be found in Listings 1, 2, 3.

Q/A

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

   Zero was reached at 0 ns and 645 ns, wait 1 was reached at 200 ns, one was reached at 245 ns, and wait 0 was reached at 600 ns.

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

   This design involves irregular, non-repeating conditions that cannot be described with regular sequential logic.

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

   Mealy outputs were used because the output depends on both the present state and the present input. Moore outputs rely only on the present state.

Results

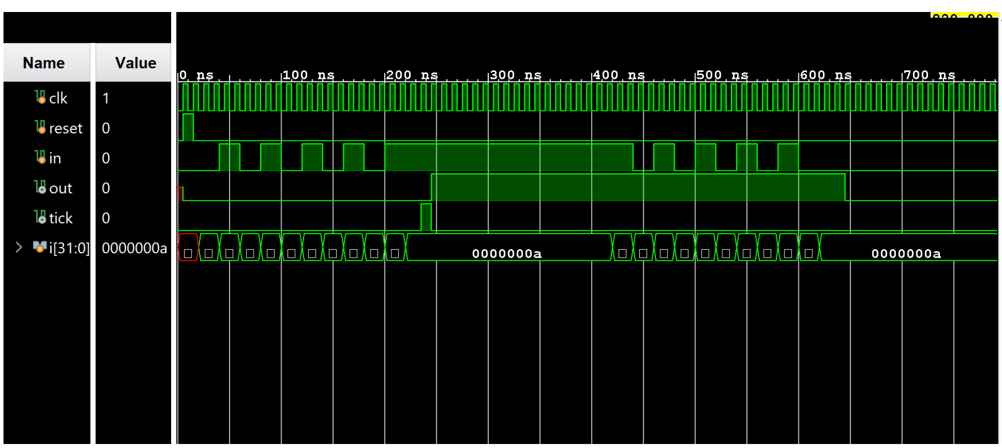

Figure 1: Debounce Simulation Waveform

Code

Listing 1: Guess FSM Module

```
`timescale 1ns / 1ps
// ELC 2137 , Makenna Meyers , 2020 -04 -22

module guess_FSM #( parameter N =21)
    ( input clk , reset ,
    input [3:0]b,
    output reg [3:0]y,
    output reg win, lose );

// define states as local parameters ( constants )
  localparam [3:0]
    s0 = 4'b0000 ,
    s1 = 4'b0001 ,
    s2 = 4'b0011 ,
    s3 = 4'b0010 ,
    swin = 4'b0110 ,
    slose = 4'b0100 ;

// internal signals
    reg [1:0] state , state_next ;

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

// combined next - state and output logic
always_comb begin
    // default behavior
    state_next = state ;

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

  s1 : begin
     y[1] = 1;
    if (~b[0] & ~b[1] & ~b[2] & ~b[3])
      state_next = s2 ;
    else if (~b[3] & ~b[2] & b[1] & ~b[0])
```

```
      state_next = swin ;
    else if (b[3] | b[2] | b[0])
      state_next = slose ;
  end

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

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

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

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

 endcase
end

endmodule
```

Listing 2: Guess FSM Test Bench

```
`timescale 1ns / 1ps
// ELC 2137 , Makenna Meyers , 2020 -04 -22


module guess_FSM_test ();

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

```
    integer i;

    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; #10;
      reset =0; #5;

      for (i=0; i <10; i=i+1) begin
        #20
        b = 4'b0001; #10;
        b = 4'b0000; #10;
      end

      for (i=0; i <10; i=i+1) begin
        #20
        b = 4'b0000; #10;
        b = 4'b0010; #10;
      end

      for (i=0; i <10; i=i+1) begin
        #20
        b = 4'b0010; #10;
        b = 4'b0011; #10;
      end

      for (i=0; i <10; i=i+1) begin
        #20
        b = 4'b0011; #10;
        b = 4'b0111; #10;
      end

      for (i=0; i <10; i=i+1) begin
        #20
        b = 4'b0110; #10;
        b = 4'b0100; #10;
      end

      for (i=0; i <10; i=i+1) begin
        #20
        b = 4'b0100; #10;
        b = 4'b0101; #10;
      end

      $finish ;
    end
endmodule
```

Listing 3: Guessing Game Module

```
`timescale 1ns / 1ps
// ELC 2137 , Makenna Meyers , 2020 -04 -22


module guessing_game #( parameter N =21)

    ( input btnU , btnD , btnL , btnR , btnC , clk ,
    input [15:0] sw ,

    output [6:0] seg ,
    output [3:0] an ,
    output [15:0] led
);

    wire db0_out ;
    wire db1_out ;
    wire db2_out ;
    wire db3_out ;
    wire [7:0] y ;
    wire win ;
    wire lose ;
    wire mux_out ;
    wire counter_out ;
    wire counter_tick ;
    wire tick ;
    wire b_in ;

    debounce #(.N(21)) db0(.clk(clk), .reset(btnC), .in(btnU), .out(
       db0_out));
    debounce #(.N(21)) db1(.clk(clk), .reset(btnC), .in(btnD), .out(
       db1_out));
    debounce #(.N(21)) db2(.clk(clk), .reset(btnC), .in(btnL), .out(
       db2_out));
    debounce #(.N(21)) db3(.clk(clk), .reset(btnC), .in(btnR), .out(
       db3_out));

    guess_FSM #(.N(21)) gfsm(.b(b_in),
                             .y({seg[6], seg[5], seg[1], seg[0]}),
                             .win(led[1]), .lose(led[0]),
                             .clk(mux_out), .reset(btnC));

    counter #(.N(8)) counter(.clk(clk), .en(1), .rst(btnC), .count(
       counter_out), .tick(counter_tick));

    mux_2 #(.N(8)) mux_guess(.in0(counter_out), .in1(clk), .out(mux_out),
       .sel(sw[0]));

    assign b_in = {db3_out, db2_out, db1_out, db0_out};
    assign an = 4'b1110;
    assign led[15:2] = 0;

endmodule
```