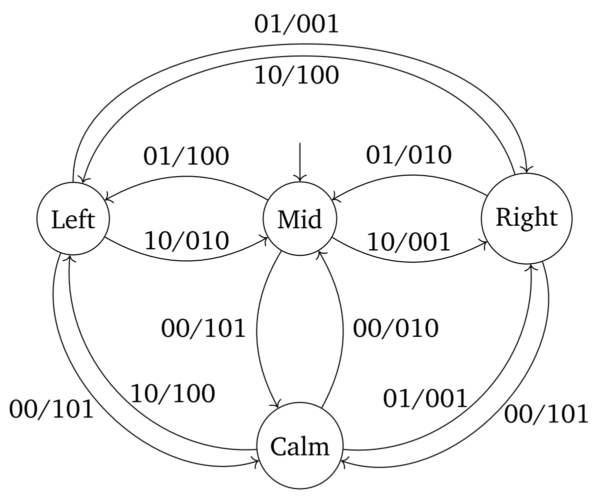
CSE 369 Lab 5

Sequential Logic

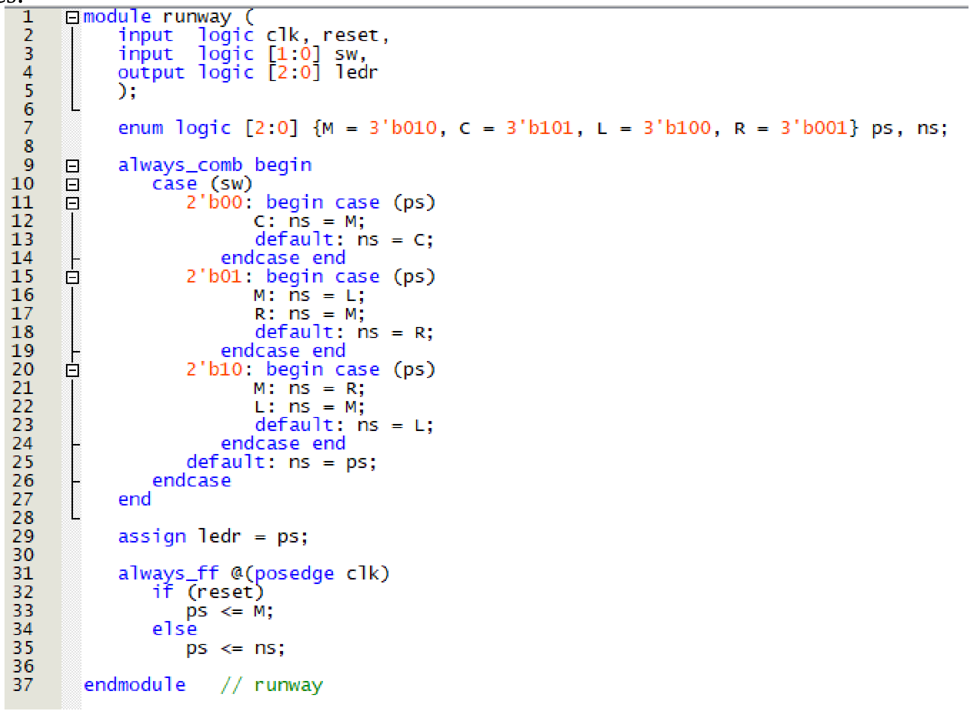
Isaac Wu
2360957
November 6, 2024

1. Finite State Machine



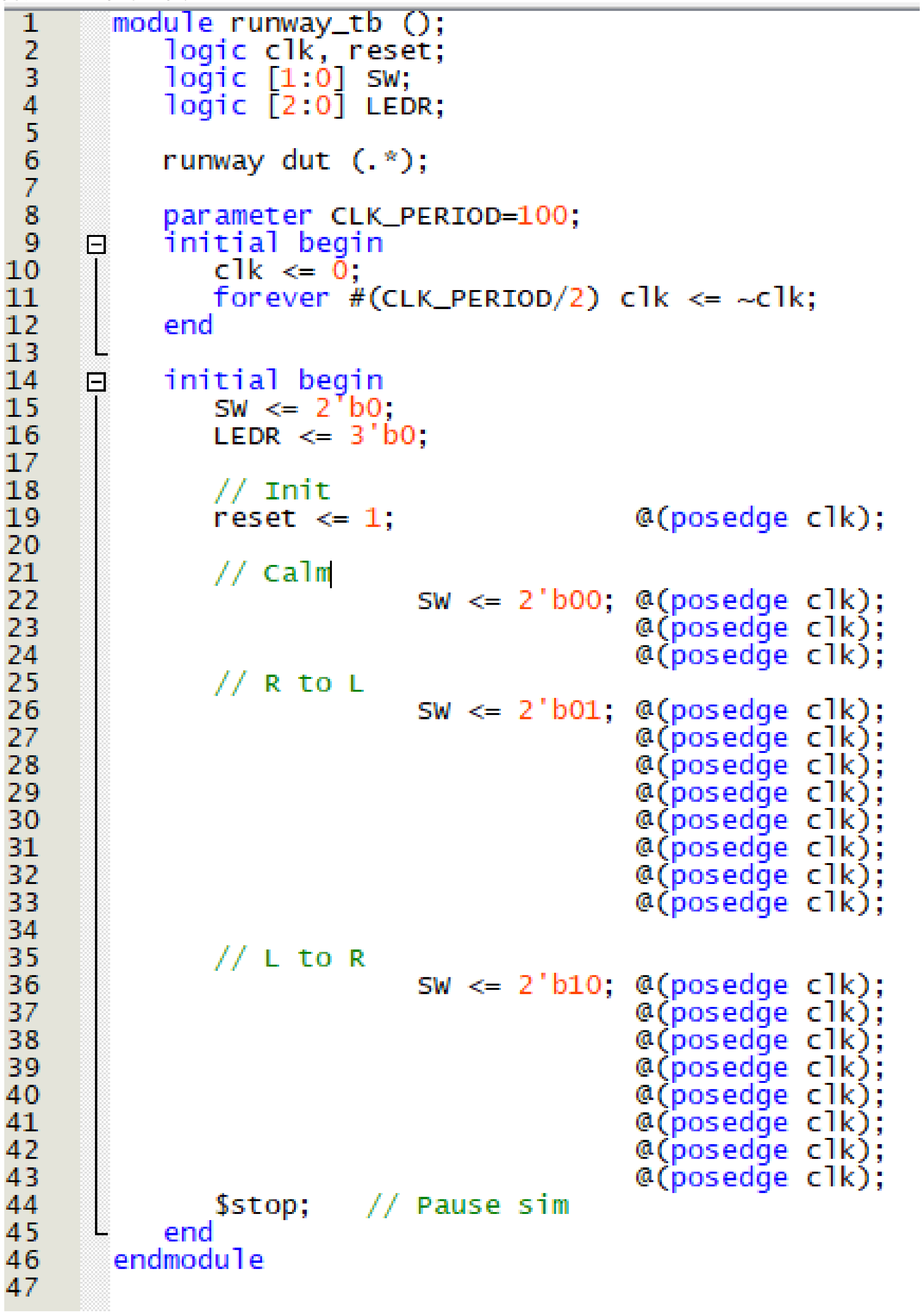
2. ModelSim Simulations

This is the code for my runway light display. I defined my states in the enum and set their values to the LED pattern to display.
I assign ledr to the present state instead of the next state so that a reset input is respected on the next clock cycle. I have the reset state be the middle state, since that LED pattern is common among all wind states.



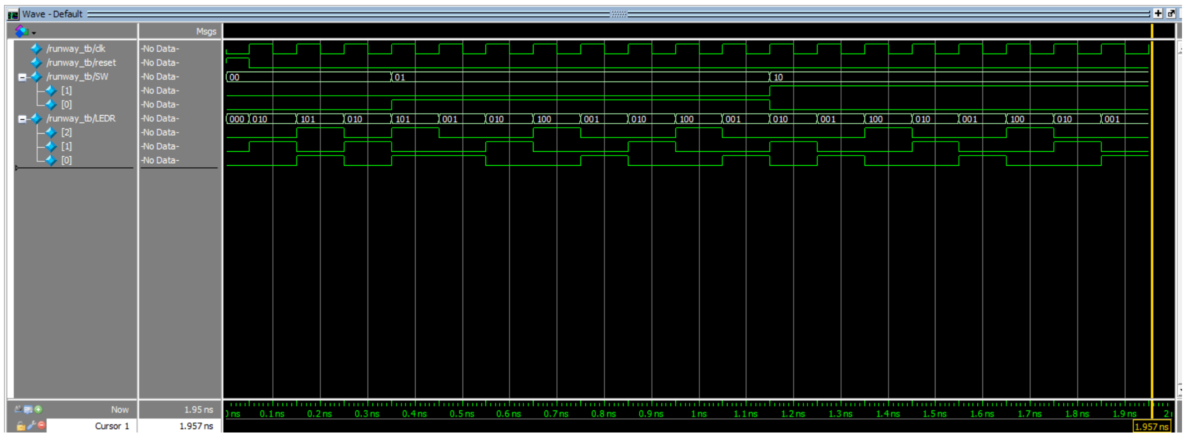
```
module runway (
   input  logic clk, reset,
   input  logic [1:0] sw,
   output logic [2:0] ledr
   );

   enum logic [2:0] {M = 3'b010, C = 3'b101, L = 3'b100, R = 3'b001} ps, ns;

   always_comb begin
      case (sw)
         2'b00: begin case (ps)
               C: ns = M;
               default: ns = C;
            endcase end
         2'b01: begin case (ps)
               M: ns = L;
               R: ns = M;
               default: ns = R;
            endcase end
         2'b10: begin case (ps)
               M: ns = R;
               L: ns = M;
               default: ns = L;
            endcase end
         default: ns = ps;
      endcase
   end

   assign ledr = ps;

   always_ff @(posedge clk)
      if (reset)
         ps <= M;
      else
         ps <= ns;

endmodule   // runway
```

Here is the test bench used. I test each wind position and wait for some clock cycles to ensure correct LED behavior.

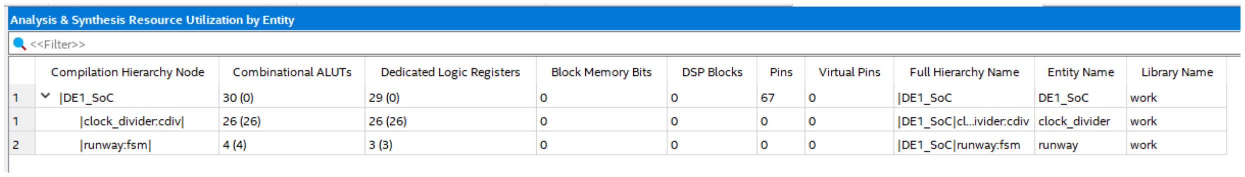


```
module runway_tb ();
   logic clk, reset;
   logic [1:0] SW;
   logic [2:0] LEDR;

   runway dut (.*);

   parameter CLK_PERIOD=100;
   initial begin
      clk <= 0;
      forever #(CLK_PERIOD/2) clk <= ~clk;
   end

   initial begin
      SW <= 2'b0;
      LEDR <= 3'b0;

      // Init
      reset <= 1;             @(posedge clk);

      // Calm
                  SW <= 2'b00; @(posedge clk);
                               @(posedge clk);
                               @(posedge clk);
      // R to L
                  SW <= 2'b01; @(posedge clk);
                               @(posedge clk);
                               @(posedge clk);
                               @(posedge clk);
                               @(posedge clk);
                               @(posedge clk);
                               @(posedge clk);
                               @(posedge clk);

      // L to R
                  SW <= 2'b10; @(posedge clk);
                               @(posedge clk);
                               @(posedge clk);
                               @(posedge clk);
                               @(posedge clk);
                               @(posedge clk);
                               @(posedge clk);
                               @(posedge clk);
      $stop;    // Pause sim
   end
endmodule
```

This is what the wave diagram looks like. The top wave is the clock used, it triggers the positive edge every 0.1ns.
The second wave is the reset input, which is only true for the first clock cycle.
The next 3 lines are the wind pattern inputs with switch inputs.
The last 4 lines are the LED outputs. During each wind pattern, the LEDs cycle through each pre-defined display, shown on the bottom waves.



3. Resource Utilization by Entity


Analysis & Synthesis Resource Utilization by Entity

<<Filter>>

	Compilation Hierarchy Node	Combinational ALUTs	Dedicated Logic Registers	Block Memory Bits	DSP Blocks	Pins	Virtual Pins	Full Hierarchy Name	Entity Name	Library Name
1	|DE1_SoC	30 (0)	29 (0)	0	0	67	0	|DE1_SoC	DE1_SoC	work
1	|clock_divider:cdiv|	26 (26)	26 (26)	0	0	0	0	|DE1_SoC|cl...ivider:cdiv	clock_divider	work
2	|runway:fsm|	4 (4)	3 (3)	0	0	0	0	|DE1_SoC|runway:fsm	runway	work

4. Misc.

How many hours (estimated) it took to complete this lab in total, including reading, planning, designing, coding, debugging, and testing.

It took around 3 hours to complete this lab.