EE5123 VLSI Design Lab: Experiment 2

Saurav Kumar Pandey
System on Chip Design
IIT Palakkad
152502019@smail.iitpkd.ac.in

October 2025

Abstract

This report details the design and implementation of sequential logic circuits using Verilog, with a focus on Finite State Machines (FSMs). The experiment commences with the design of a 4-bit synchronous up/down counter. Building on this, a practical FSM-based traffic light controller is developed for a two-way intersection, initially managing red and green light cycles with countdown timers. The design is then enhanced to include a yellow warning light, creating a complete and realistic three-phase traffic control system.

1 Introduction

Finite State Machines (FSMs) are a fundamental concept in digital logic design, providing a powerful model for creating sequential circuits. This experiment focuses on the practical application of these concepts using the Verilog HDL to design and verify a 4-bit synchronous up/down counter and a complete traffic light controller FSM.

2 4-bit Up/Down Counter

2.1 Circuit Design

The counter's design features an adder/subtractor to generate the next count value, multiplexers controlled by 'mode', 'en', and 'reset' signals to select the correct input, and a D-type flip-flop to store the current count, synchronized to the clock.

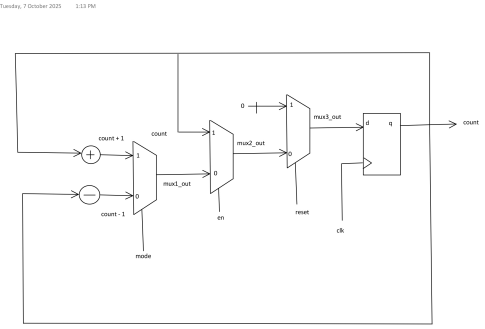

Figure 1: Counter circuit diagram.

2.2 Verilog Implementation

```
module up_down_counter #(
    parameter N = 4
)(
    input wire clk,
    input wire reset,
    input wire en,
    input wire mode,
    output reg [N-1:0] count
);

    always @(posedge clk) begin
        if (reset) begin
            count <= 0;
        end else if (en) begin
            if (mode) begin
                count <= count + 1;
            end else begin
                count <= count - 1;
            end
        end
    end
```

```
endmodule
```

2.3 Verification

Testbench Code

```
‘timescale 1ns/1ns
module up_down_counter_tb;
  parameter N = 4;
  reg clk, reset, en, mode;
  wire [N-1:0] count;

  up_down_counter #(.N(N)) uut (
    .clk(clk),.reset(reset),.en(en),
    .mode(mode),.count(count)
  );

  initial clk = 0;
  always #5 clk = ~clk;

  initial begin
    $dumpfile("dump.vcd");
    $dumpvars(0, up_down_counter_tb);
    reset = 1; en = 0; mode = 0;
    #12 reset = 0; #5 en = 1;
    mode = 1; #40;
    mode = 0; #40;
    en = 0; #20;
    en = 1; mode = 1; #40;
    reset = 1; #10 reset = 0;
    #30; $finish;
  end
endmodule
```

Simulation Output

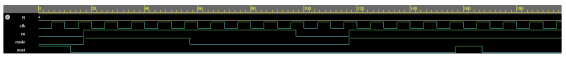

Figure 2: Simulation waveform for the counter.

3 Traffic Light Controller FSM

3.1 Two-Phase Controller

The simple controller cycles between two states: HIGHWAY_GO and CROSSROAD_GO, with a counter determining the duration of each state.

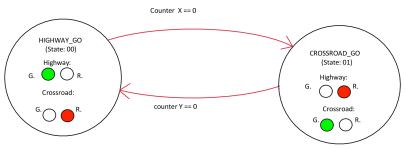

Figure 3: FSM diagram for two-phase controller.

Verilog Implementation

```
module traffic_light_controller #(
    parameter X = 6,
    parameter Y = 3
)(
    input  wire        clk,
    input  wire        reset,
    output reg         highway_green,
    output reg         highway_red,
    output reg         crossroad_green,
    output reg         crossroad_red,
    output reg [2:0]   counter
);

    localparam HIGHWAY_GO   = 2’b00;
    localparam CROSSROAD_GO = 2’b01;

    reg [1:0] current_state, next_state;

    always @(posedge clk or posedge reset) begin
        if (reset) begin
            current_state <= HIGHWAY_GO;
            counter       <= X;
        end else begin
            current_state <= next_state;
            if (counter == 0) begin
                if (current_state == HIGHWAY_GO)
                    counter <= Y;
                else
                    counter <= X;
            end else begin
                counter <= counter - 1;
```

```
            end
        end
    end

    always @(*) begin
        highway_green   = 1'b0;
        highway_red     = 1'b1;
        crossroad_green = 1'b0;
        crossroad_red   = 1'b1;
        next_state      = current_state;

        case (current_state)
            HIGHWAY_GO: begin
                highway_green = 1'b1;
                highway_red   = 1'b0;
                if (counter == 0) begin
                    next_state = CROSSROAD_GO;
                end
            end

            CROSSROAD_GO: begin
                crossroad_green = 1'b1;
                crossroad_red   = 1'b0;
                if (counter == 0) begin
                    next_state = HIGHWAY_GO;
                end
            end
        endcase
    end
endmodule
```

Testbench

```
`timescale 1ns / 1ps

module traffic_light_controller_tb;

    reg clk;
    reg reset;

    wire highway_green;
    wire highway_red;
    wire crossroad_green;
    wire crossroad_red;
    wire [2:0] counter;

    traffic_light_controller #(
        .X(6),
        .Y(3)
    ) dut (
        .clk(clk),
        .reset(reset),
        .highway_green(highway_green),
        .highway_red(highway_red),
        .crossroad_green(crossroad_green),
        .crossroad_red(crossroad_red),
        .counter(counter)
    );

    initial begin
        clk = 0;
        forever #5 clk = ~clk;
    end

    initial begin
        reset = 1;
        #25;
        reset = 0;
        #300;
        $finish;
    end

    initial begin
        $dumpfile("traffic.vcd");
        $dumpvars(0, traffic_light_controller_tb);
    end
endmodule
```

Simulation Output

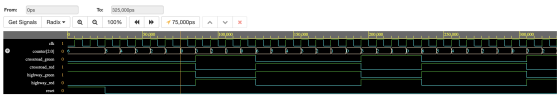

Figure 4: Simulation waveform for the two-phase traffic controller.

3.2 Three-Phase Controller

The design was modified to include a yellow light. This adds yellow warning states, creating a four-state machine that provides a safe transition time for traffic.

Verilog Implementation

```
module tlc_3phase #(
    parameter H_GREEN_T = 6,
    parameter H_YELLOW_T = 2,
    parameter C_GREEN_T = 4,
    parameter C_YELLOW_T = 2
)(
```

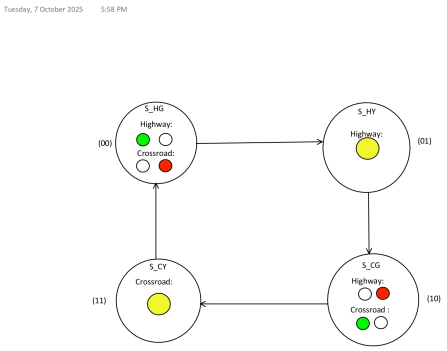

Figure 5: FSM diagram for the three-phase controller.

```
    input wire clk, reset,
    output reg h_green, h_yellow, h_red,
    output reg c_green, c_yellow, c_red
);
    localparam S_HG = 2'b00, S_HY = 2'b01,
               S_CG = 2'b10, S_CY = 2'b11;
    reg [1:0] state, next_state;
    reg [2:0] counter;

    always @(posedge clk or posedge reset) begin
        if(reset) state <= S_HG;
        else state <= next_state;
    end

    always @(posedge clk or posedge reset) begin
        if(reset) counter <= H_GREEN_T;
        else if (counter == 0)
            case(next_state)
                S_HG: counter <= H_GREEN_T;
                S_HY: counter <= H_YELLOW_T;
                S_CG: counter <= C_GREEN_T;
                S_CY: counter <= C_YELLOW_T;
            endcase
        else counter <= counter - 1;
    end

    always @(*) begin
        case(state)
            S_HG: if(counter==0) next_state = S_HY;
                  else next_state = S_HG;
            S_HY: if(counter==0) next_state = S_CG;
                  else next_state = S_HY;
            S_CG: if(counter==0) next_state = S_CY;
                  else next_state = S_CG;
            S_CY: if(counter==0) next_state = S_HG;
                  else next_state = S_CY;
        endcase
    end

    always @(*) begin
        h_green=0; h_yellow=0; h_red=1;
        c_green=0; c_yellow=0; c_red=1;
        case(state)
            S_HG: {h_green,h_red}={1'b1,1'b0};
            S_HY: {h_yellow,h_red}={1'b1,1'b0};
            S_CG: {c_green,c_red}={1'b1,1'b0};
            S_CY: {c_yellow,c_red}={1'b1,1'b0};
        endcase
    end
endmodule
```

Testbench

```
`timescale 1ns / 1ps
module tlc_3phase_tb;
    reg clk, reset;
    wire h_green, h_yellow, h_red;
    wire c_green, c_yellow, c_red;

    tlc_3phase dut (
        .clk(clk), .reset(reset),
        .h_green(h_green), .h_yellow(h_yellow),
        .h_red(h_red), .c_green(c_green),
        .c_yellow(c_yellow), .c_red(c_red)
    );

    initial begin
        clk = 0;
        forever #5 clk = ~clk;
    end

    initial begin
        reset = 1; #15;
        reset = 0; #300;
        $finish;
    end
endmodule
```

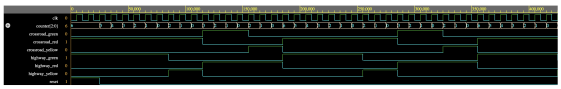

Figure 6: Simulation waveform for the three-phase traffic controller.

Simulation Output

4 Conclusion

This experiment successfully demonstrated the design of sequential circuits in Verilog. A synchronous 4-bit up/down counter was created and verified, and an FSM was used to model and implement a realistic three-phase traffic light controller, reinforcing the principles of state-based design.