Assignment no 05:
Chapter 4: Hardware Description Languages

Note: You can check the exercises after the Chapter. In our assignment, we are using the 2nd Edition of "Digital Design and Computer Architecture" By David and Sarah Harris.

Note: In our course, we use Verilog While the book uses SystemVerilog. Verilog is a subset of SystemVerilog. So, carefully prepare your answers in Verilog.

Exercise 4.3 Write a Verilog module that computes a four-input XOR function. The input is $a_{3:0}$, and the output is y.

Exercise 4.4 Write a self-checking testbench for **Exercise 4.3**. Create a test vector file containing all 16 test cases. Simulate the circuit and show that it works.

Exercise 4.24 Sketch the state transition diagram for the FSM described by the following Verilog code.

```
Module fsm2(clk, reset, a, b, y);

input clk, reset;
input a, b;
output y;
reg [1:0] state, nextstate;

parameter S0 = 2'b00;
parameter S1 = 2'b01;
parameter S2 = 2'b10;
parameter S3 = 2'b11;

always @(posedge clk, posedge reset)
begin
      if (reset)
            state <= S0;
      else
            state <= nextstate;
end
always @(*)
begin
      case (state)
            S0: if (a ^ b) nextstate = S1;
                  else nextstate = SO;
            S1: if (a & b) nextstate = S2;
                  else nextstate = SO;
            S2: if (a | b) nextstate = S3;
                  else nextstate = SO;
            S3: if (a | b) nextstate = S3;
                  else nextstate = SO;
      endcase
end
assign y = (state== S1) | (state== S2);
endmodule
```

Exercise 4.25 Sketch the state transition diagram for the FSM described by the following Verilog code. An FSM of this nature is used in a branch predictor on some microprocessors.

```
module fsm1(clk, reset, taken, back, predicttaken);

input clk, reset;
input taken, back;
output predicttaken;

reg [4:0] state, nextstate;

parameter S0 = 5'b00001;
parameter SI = 5'b00010;
parameter S2 = 5'b00100;
parameter S3 = 5'b01000;
parameter S4 = 5'b10000;

always @(posedge clk, posedge reset)
begin
      if (reset)
            state <= S2;
      else
            state <= nextstate;
end

always @(*)
begin
      case (state)
            S0: if (taken) nextstate = S1;
                  else nextstate = S0;
            S1: if (taken) nextstate = S2;
                  else nextstate = S0;
            S2: if (taken) nextstate = S3;
                  else nextstate = S1;
            S3: if (taken) nextstate = S4;
                  else nextstate = S2;
            S4: if (taken) nextstate = S4;
                  else nextstate = S3;
            default: nextstate = S2;
      endcase
end

assign predicttaken = (state == S4) |(state == S3) |(state == S2 && back);
endmodule
```

Exercise 4.29 Write a Verilog module for the traffic light controller from **Section 3.4.1.**