TÉCNICO LISBOA

1ST PTFSE PROJECT

From: David Muzikař, Simon Prato, João Barros

Student ID: ist1116882, ist1117175, ist103939

Course: Design, Test and Reliability of Electronic Systems

Subject: Verilog State Machine

Date: November 15, 2025

---

1 Behavior of the State Machine

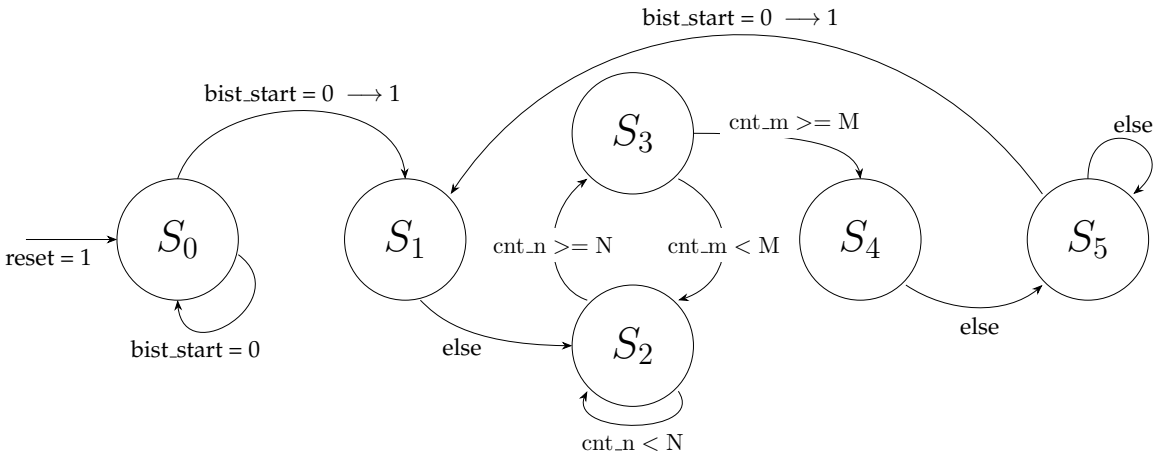

Figure 1: State diagram

State	mode	init	running	finish	bist_end
S_0	0	0	0	0	0
S_1	0	1	0	0	0
S_2	1	0	1	0	0
S_3	0	0	1	0	0
S_4	0	0	0	1	0
S_5	0	0	0	0	1

Table 1: Output variables at every state

The state machine module contains two registers. The first register cnt_n is responsible for keeping track how many clock periods have passed. In the state S_2, it is repeatedly compared with the user-defined number of clock cycles N. The second register cnt_m follows how often the sequence occurred on the output. It enables a repetition of the sequence, until the user-defined variable M is reached.

By setting the input reset=1, the state S_0 will be reached at the next rising edge of the clock. It has priority over all other input variables. Every input signal is read at the rising edge of the clock signal, i.e. they are synchronous. The output variables at every state are shown in Table 1, and their transitions are described in the state diagram in Figure 1.

Do we need to specify in the diagram at every transition, that reset has a priority?

2 Implementation in Verilog

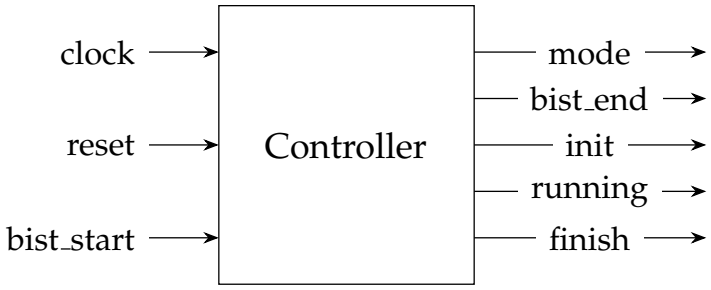

Figure 2: Controller interface

Table 2 shows the selected parameters for our Group (group 3) in Project 1. These values of N and M are used as the basis for the project's implementation.

Group	N	M
3	7	10

Table 2: Project 1 - Parameters (Group 3)

2.1 Design

```
`timescale 100ps / 1ps

////////////////////////////////////////////////
//
// Company: Instituto Superior Tcnico de Lisboa
// Module Name: state_machine
// Description: Generic state machine
//
////////////////////////////////////////////////

module state_machine(clock, reset, bist_start, mode, bist_end, init,
    running, finish);
input clock, reset, bist_start;
output reg mode, bist_end, init, running, finish;

// Registers containing the current and next state
reg [2:0] state, next_state;

// Parameters identifying states
localparam [2:0] S0=0, S1=1, S2=2, S3=3, S4=4, S5=5;

// Parameters N and M defining the output sequence
parameter N = 7;
```

```
parameter M = 10;

// Calculate required register sizes to contain the number of iterations
   conducted.
parameter N_SIZE = $clog2(N + 1);
parameter M_SIZE = $clog2(N + 1);

// Declaration of registers keeping track of iterations producting hte
   output sequence.
reg [N_SIZE:0] cnt_n;
reg [M_SIZE:0] cnt_m;

// Process the next state
always @(*)
begin
    case (state)
    S0: begin
        cnt_n = 0;
        cnt_m = 0;
        if (bist_start) next_state = S1;
        else next_state = S0;
        end
    S1: next_state = S2;
    S2: if (cnt_n >= N) next_state = S3;
        else begin
        next_state = S2;
        cnt_n = cnt_n + 1;
        end
    S3: if (cnt_m >= M) next_state = S4;
        else begin
        next_state = S2;
        cnt_m = cnt_m + 1;
        cnt_n = 0;
        end
    S4: next_state = S5;
    S5: if (bist_start == 1) begin
        next_state = S1;
        cnt_m = 0;
        cnt_n = 0;
        end
        else next_state = S5;

endcase
end

// Set the next state to S0 if reset is HIGH
always @(posedge clock)
    begin
        if (reset == 1'b1)
        state <= S0;
        else
        state <= next_state;
end

// Set output depending on state
  always @(posedge clock) begin
    case (state)
        S0: begin
```

```
        mode <= 0;
        bist_end <= 0;
        init <= 0;
        running <= 0;
        finish <= 0;
        end

        S1: begin
        mode <= 0;
        bist_end <= 0;
        init <= 1;
        running <= 0;
        finish <= 0;
        end

        S2: begin
        mode <= 1;
        bist_end <= 0;
        init <= 0;
        running <= 1;
        finish <= 0;
        end

        S3: begin
        mode <= 0;
        bist_end <= 0;
        init <= 0;
        running <= 1;
        finish <= 0;
        end

        S4: begin
        mode <= 0;
        bist_end <= 0;
        init <= 0;
        running <= 0;
        finish <= 1;
        end

        S5: begin
        mode <= 0;
        bist_end <= 1;
        init <= 0;
        running <= 0;
        finish <= 0;
        end
    endcase
end

endmodule
```

Listing 1: State machine Verilog code

2.2 Testbench

```
`timescale 1ns / 1ps
```

```
module state_machine_tb;

    reg clock, reset, bist_start;
    wire mode, bist_end, init, running, finish;

    state_machine state_machine(
        .clock(clock),
        .reset(reset),
        .bist_start(bist_start),
        .mode(mode),
        .bist_end(bist_end),
        .init(init),
        .running(running),
        .finish(finish)
    );

    initial begin
        clock = 0;
        reset = 1;
        bist_start = 0;
    end

    always
        #50 clock = !clock;

    initial
        begin
        #100 reset = 0;
        #100 bist_start = 1;
        #200 $finish;
        end

endmodule
```

Listing 2: State machine testbench Verilog code