Computation II: embedded system design (5EIB0)

Available from: Friday, 8 March 2024, 2:30 PM
Requested files: seq_top_tb.v (Download)
Type of work: Individual work

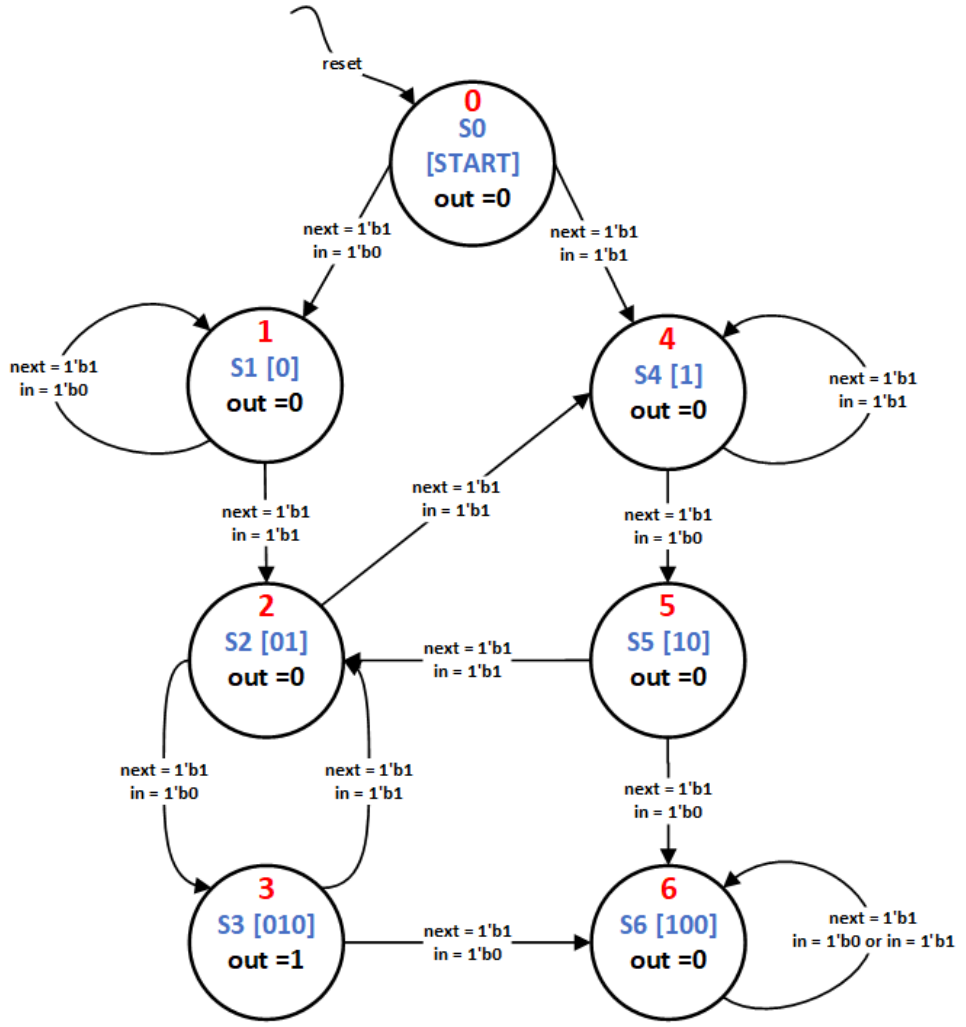
Introduction

In this exercise, you will create a verilog implementation of a Finite State Machine (FSM) which detects a finite string pattern. It has the following properties:

- one input (**in**) and one output (**out**)
- output is asserted whenever the input sequence **... 010 ...** is observed, as long as the sequence **...100...** is not seen.

Example input/output behavior:

```
in:     0 0 1 0 1 0 1 0 0 1 0 ....
out:    0 0 0 1 0 1 0 1 0 0 0 ...
```



The diagram above illustrates the transitions of the FSM that should be implemented to achieve this. It shows a Moore Machine where the output is defined by the state. The machine has six states. The state numbers are marked in red.

Assignment

Write a FSM testbench named **seq_top_tb.** It will drive the module named **seq_top** which takes **input signals clk, reset, in, next** and produces **output signals out, state_display**. Assume all input and output signals to be 1 bit wide except state_display which is 3 bits wide and little endian. You have to write the testbench which tests the given finite string pattern recognizer for the following input sequence:

```
0 => 1  => 0  => 1  => 1 => 0  => 1 => 0  => 0  => 1 =>  0
```

Print the following after each input stimuli:

```
$display("state %x, out %x", state_display, out);
```

Use the following template for writing the testbench:

```
`timescale 1ns / 1ps
module seq_top_tb();

       <<DECLARE SIGNALS TO DRIVE MODULE seq_top>>

       seq_top seq_top_inst(
               .clk(clk),
               .reset(reset),
               .next(next),
               .in(in),
               .out(out),
               .state_display(state_display));
   <<DECLARE CLOCK>>


   initial begin
          $dumpfile("dut.vcd");
          $dumpvars(0, seq_top_inst);

          <<RESET>>
          #500
          $display("state %x, out %x", state_display, out);


          <<STIMULI 1>>
          #500
          $display("state %x, out %x", state_display, out);

          <<STIMULI 2>>
          #500
          $display("state %x, out %x", state_display, out);

          ..........................................

          #100
      $finish;
   end
endmodule
```

STRICT NOTE: Just change the portions marked in red in the tesbench template.

Debug

Click on the symbol marked below to see the waveforms produced by your design. Please note that if your code has an error that prevents it being simulated it will not produce any waveforms.

Requested files

seq_top_tb.v

```
1  `timescale 1ns / 1ps
2  module seq_top_tb();
3
4
5      <<DECLARE SIGNALS TO DRIVE MODULE seq_top>>
6
7      seq_top seq_top_inst(
8          .clk(clk),
9          .reset(reset),
10         .next(next),
11         .in(in),
12         .out(out),
13         .state_display(state_display));
14
15         <<DECLARE CLOCK>>
16
17
18
19     initial begin
20        $dumpfile("dut.vcd");
21        $dumpvars(0, seq_top_inst);
22
23        <<RESET>>
24        #500
25        $display("state %x, out %x", state_display, out);
26
27
28        <<STIMULI 1>>
29        #500
30        $display("state %x, out %x", state_display, out);
31
32        <<STIMULI 2>>
33        #500
34        $display("state %x, out %x", state_display, out);
35
36        ..............................................
37
38        #100
39        $finish;
40
41     end
42 endmodule
```

VPL