Computation II: embedded system design (5EIB0)

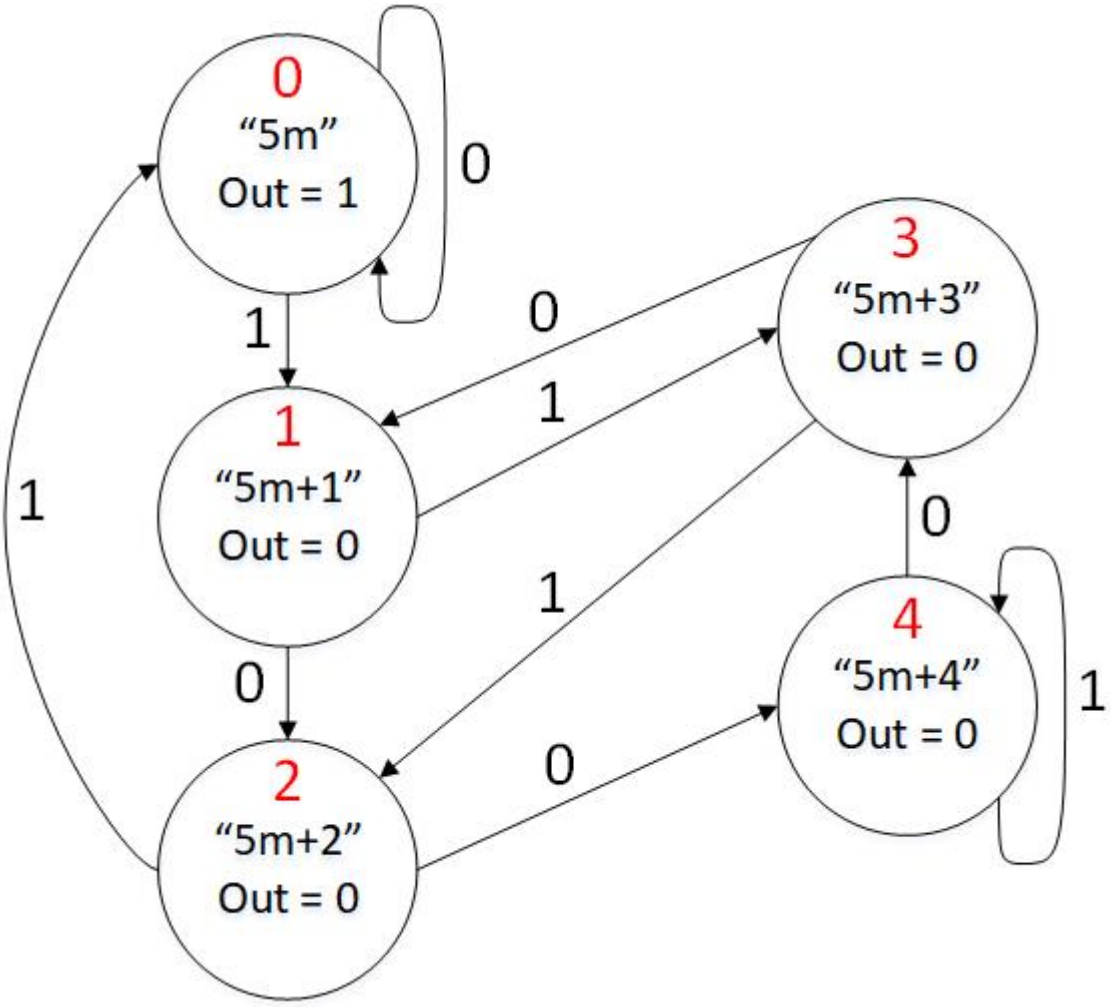
Description | Submission view

Part 2c: 2020-04-16 Finite State Machine (Debugging FSM Divisibility Test - 3pt)

Avaliable from: Thursday, 16 April 2020, 1:30 PM
Due date: Thursday, 16 April 2020, 3:25 PM
Requested files: div5.v (Download)
Type of work: Individual work

Introduction

The figure below shows the state transition diagram of a Finite State Machine(FSM) which checks the obtained sequence for divisibility by 5. The output is set to 1 whenever the input sequence is divisible by 5. For example,a sequence "**011110101**" goes through states **0=>1=>3=>2=>0=>1=>2=>0**. The corresponding ouput trace will be **1=>0=>0=>0=>1=>0=>0=>1**.



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 five states (that are numbered in red). Each state in the above diagram is represented as 5m+x, where m is an interger and x is the remainder obtained when divided by 5.

Assignment

You will be provided with a buggy implementation of the above fsm. You need to find the bug and change it to make it working as per the state transition diagram given above. The FSM module you will modify is named **div5. It** takes three input signals **clk, reset, in** and produces two output signals **out** and **state_display**. The **reset** input always makes a transition to the **5m** state. Assume all input and output signals to be 1 bit wide except state_display which is 3 bits wide and little endian. The **state_display** must output the number of the current state using the same numbers as the states in the FSM diagram above (shown in red).

Debug

Click on the symbol marked below to see waveforms.

Requested files

div5.v

```
1  timescale 1ns / 1ps
2
3
4  module div5(clk,reset,in,out, state_display);
5
6  input clk, reset, in; // Assume asynchronous reset;
7  output out;
8  output [2:0] state_display;
9
10 parameter S_REM0 = 0; parameter S_REM1 = 1; // state assignments
11 parameter S_REM2 = 2; parameter S_REM3 = 3;
12 parameter S_REM4 = 4;
13
14 reg [2:0] curr_state,next_state;
15
16 always @(*) begin // implement state transition diagram
17 if (reset) next_state = S_REM0;
18 else case (curr_state)
19 S_REM0: next_state = in ? S_REM1 : curr_state;
20 S_REM1: next_state = in ? S_REM3 : S_REM2;
21 S_REM2: next_state = in ? S_REM0 : S_REM4;
22 S_REM3: next_state = in ? S_REM1 : S_REM2;
23 S_REM4: next_state = in ? curr_state : S_REM3;
24 default: next_state = S_REM0; // handle unused states
25 endcase
26 end
27
28 always @ (posedge clk) curr_state <= next_state;
29
30 assign out = (curr_state == S_REM3); // assign output: Moore machine
31 assign state_display = curr_state; // debugging
32
33 endmodule
```

VPL

◄ Part 2b: 2020-04-16 Finite State Machine (Write Testbench for Mealy One Accumulator - 3pt)

Part II: 2020-04-16 ENABLE PROCTORING (NOT GRADED) (Remotely Proctored) ►

Return to: Final Exam 2020...