Computation II: embedded system design (5EIB0)

Description | Submission view

Part 2d: 2020-04-16 Finite State Machine (Design an Odd-Even Sequence Detector - 6pt)

Avaliable from: Thursday, 16 April 2020, 3:35 PM
Due date: Thursday, 16 April 2020, 5:30 PM
Requested files: seq_top.v (Download)
Type of work: Individual work

Introduction

In this assignment, you will create a Verilog implementation of a Finite State Machine (FSM) which detects whether the number of input events is even or odd.

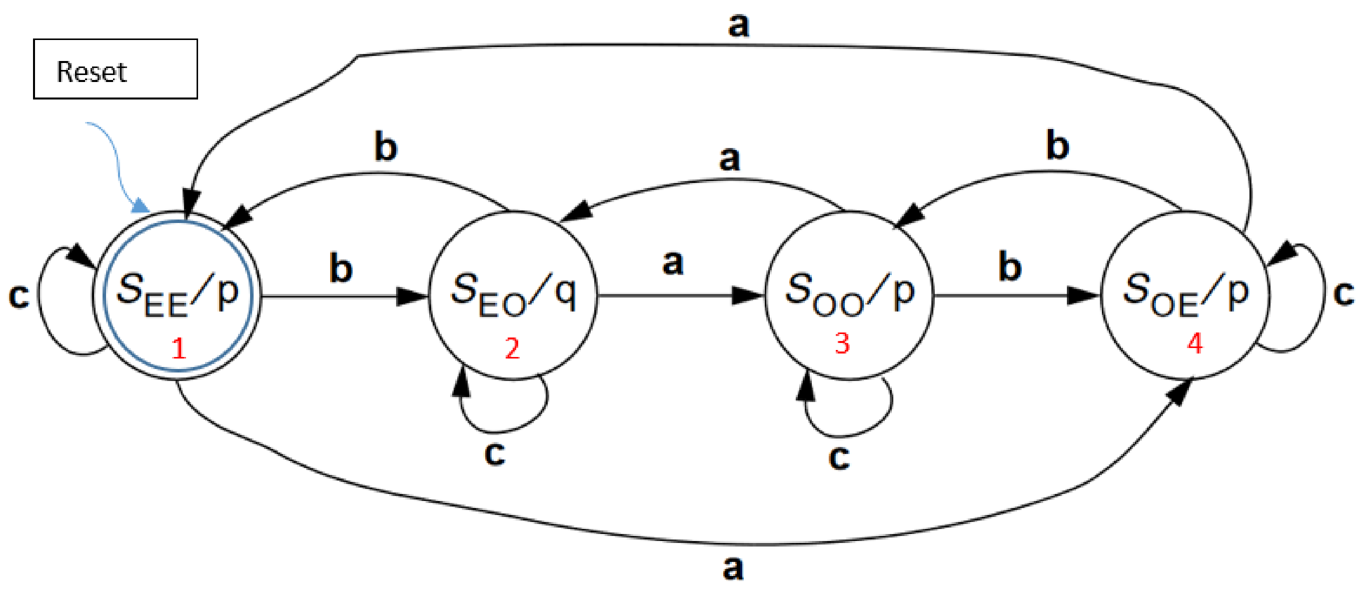
It has the following inputs and outputs:

- **in**, a two-bit input which encodes the following events: {a = 01; b = 10; c = 00}
- **q**, a one-bit output which is high if and only if the input sequence had an even number of a's and an odd number of b's
- **p**, a one-bit output which is high if and only if **q** is low.

Example input / output behaviour:

in	b	b	a	c	c	c	b	b	b	a	a
p	1	1	1	0	0	0	0	1	1	1	1
q	0	0	0	1	1	1	1	0	0	0	0

State diagram



The diagram below illustrates the transitions of the FSM that should be implemented to achieve this. It shows a Moore Machine, which means that the output is defined by the state. The machine has four states. The start state is shown by the double circle. The states are:

```
S_EE: even number of a's and even number of b's / output is p
S_EO: even number of a's and odd number of b's / output is q
S_OO: odd number of a's and odd number of b's / output is p
S_OE: odd number of a's and even number of b's / output is p
```

Assignment

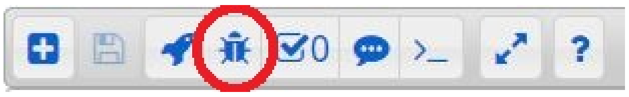
Write a FSM module named **seq_top** which implements above state machine. Besides the input **in** and the outputs **p** and **q** as described above, it must have a clock input **clk**, an update signal **update**, and a reset input **reset**.

The state machine must only transition to the next state (on rising edges of **clk**) when **update** was high in this clock cycle and low in the previous clock cycle. The active-high synchronous **reset** always causes a transition to the start state (regardless of **update**).

*Hint: do not use: always @ (posedge **update**).*

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 from being simulated it will not produce any waveforms.



Requested files

seq_top.v

```
`timescale 1ns / 100ps

module seq_top (
   input  wire [0:0] clk,
   input  wire [0:0] reset,
   input  wire [0:0] update,
   input  wire [1:0] in,
   output reg  [0:0] p,
   output reg  [0:0] q
   );

endmodule

```

VPL

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

Part 2d: 2020-04-16 Finite State Machine (Design an Odd-Even Sequence Detector - 6pt) EDITABLE ►

Return to: Final Exam 2020...