University of Washington – Computer Science & Engineering

Spring 2017 Instructor: Justin Hsia 2017-05-16

CSE 369 QUIZ 2

Name: _Perry_Perfect__________________

UWNetID: _perfect_______________________

Please do not turn the page until 10:30.

Instructions

- This quiz contains 4 pages, including this cover page. You may use the backs of the pages for scratch work.
- Please clearly indicate (box, circle) your final answer.
- The quiz is closed book and closed notes.
- Please silence and put away all cell phones and other mobile or noise-making devices.
- Remove all hats, headphones, and watches.
- You have 25 minutes to complete this quiz.

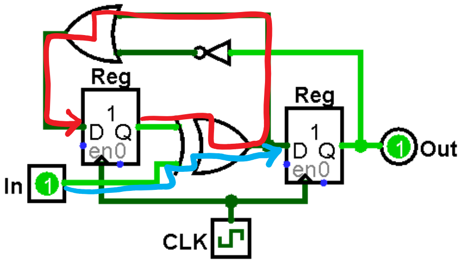
Advice

- Read questions carefully before starting. Read *all* questions first and start where you feel the most confident to maximize the use of your time.
- There may be partial credit for incomplete answers; please show your work.
- Relax. You are here to learn.

Question	Points	Score
(1) SL & Timing	6	6
(2) FSM Implementation	10	10
(3) FSM Design	10	10
Total:	**26**	**26**

Question 1: Sequential Logic & Timing [6 pts]

Consider the following circuit diagram with t_{setup} = **50 ps** (10^{-12} s), t_{hold} = **20 ps**, t_{C2Q} = **70 ps**, t_{NOT} = **15 ps**, t_{OR} = **80 ps**, and t_{XOR} = **100 ps**. Consider each part below *independently* and fill in your answers in the boxes below, making sure to ***include units***.



(A) If the input `In` changes exactly on clock triggers, what is the minimum clock period that we can use and still ensure proper behavior? [4 pts]

300 ps

The critical path is shown above in red and goes through the XOR and OR gates.
So we need $t_{C2Q} + t_{XOR} + t_{OR} \le t_{period} - t_{setup}$.
Then $t_{period} \ge 70 + 100 + 80 + 50 = 300$ ps.

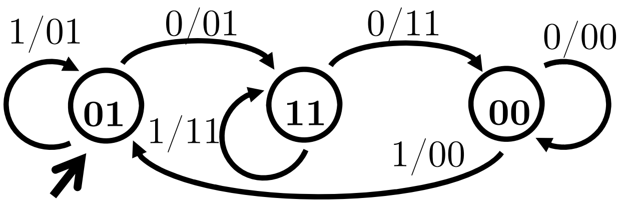
(B) If the input `In` changes **10 ps** after each clock trigger, what is the minimum t_{XOR} delay we need to prevent a *hold time violation*? [2 pts]

10 ps

Shortest path is shown above in blue and goes from In through the XOR gate.
So we need $10 \text{ ps} + t_{XOR} \ge t_{hold}$ and $t_{XOR} \ge 20 - 10 = 10$ ps.

Question 2: Finite State Machine Implementation [10 pts]

(A) Fill in the provided truth table based on the FSM shown. [2 pts]



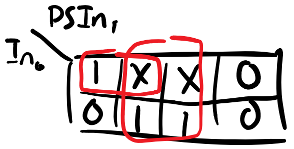
PS_1	PS_0	In	NS_1	NS_0	Out_1	Out_0
0	0	0	0	0	0	0
0	0	1	0	1	0	0
0	1	0	1	1	0	1
0	1	1	0	1	0	1
1	0	0	X	X	X	X
1	0	1	X	X	X	X
1	1	0	0	0	1	1
1	1	1	1	1	1	1

(B) Complete the circuit diagram below using *minimal logic* based on the truth table shown below. You are welcome to use 2- and 3-input logic gates. [8 pts]

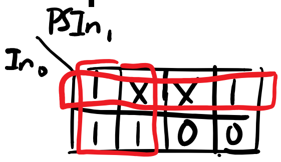
PS	In_1	In_0	NS	Out
0	0	0	1	1
0	0	1	0	1
0	1	0	X	X
0	1	1	1	1
1	0	0	0	1
1	0	1	0	0
1	1	0	X	X
1	1	1	1	0

Wire connection: ┼

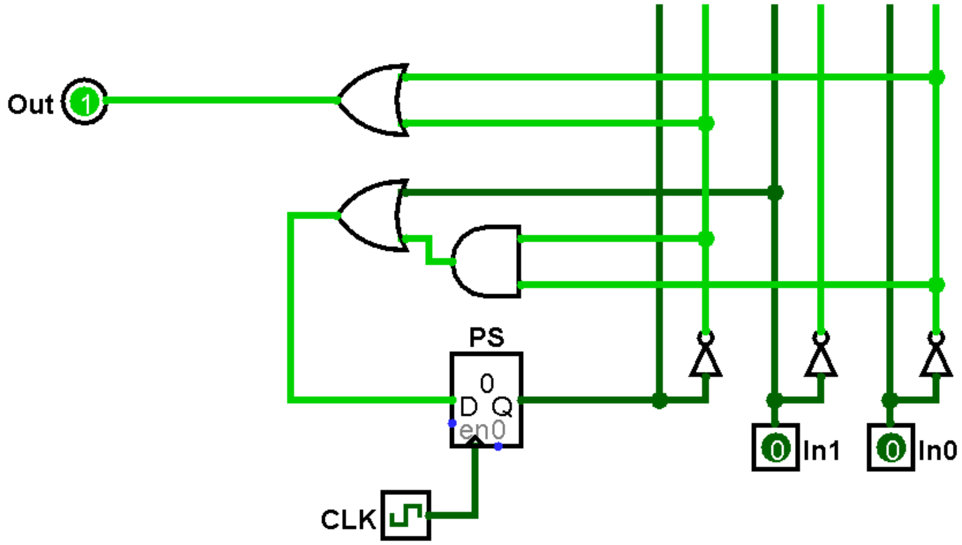
Wire crossing: ⌒



$NS = In_1 + \overline{In_0}\,\overline{PS}$



$Out = \overline{PS} + \overline{In_0}$



Question 3: Finite State Machine Design [10 pts]

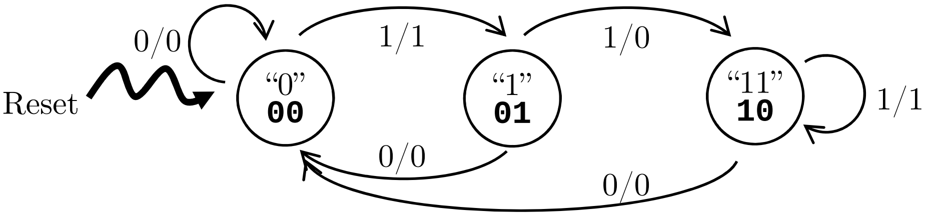
(A) If we have an FSM with five states and a transition arrow from each state to all the other states (20 transitions total), how many bits does our system require? [2 pt]

5 states and 4 transitions per state.

State Bits: **3**	Input Bits: **2**

(B) The following FSM takes a stream of inputs and removes the *second* 1 from every consecutive string of 1's.

```
Input:    1 0 1 1 0 1 1 1 0 1 1 1 1
Output:    1 0 1 0 0 1 0 1 0 1 0 1 1
```



Complete the testbench `initial` block to *thoroughly* test the FSM. You are welcome to fill out the Verilog comments to help you keep track of state, but these will not be graded. [3 pts]

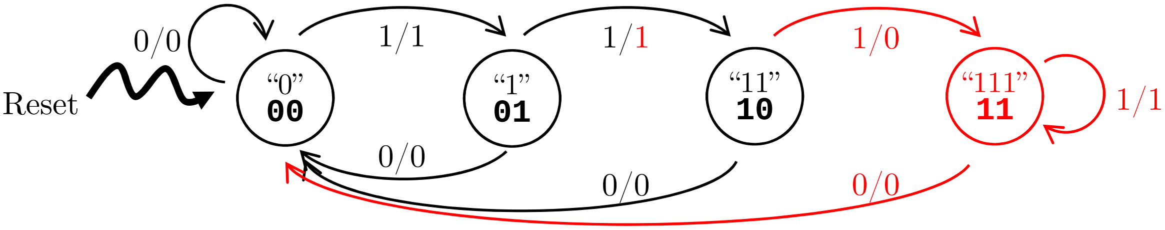
```
initial begin
                        In <= 1;      // state:  00

   @(posedge clk);   In <= __0__;   // state: _01_
   @(posedge clk);   In <= 0;       // state: _00_
   @(posedge clk);   In <= 1;       // state: _00_

   @(posedge clk);   In <= __1__;   // state: _01_
   @(posedge clk);   In <= 1;       // state: _10_

   @(posedge clk);   In <= __0__;   // state: _10_
   @(posedge clk);                  // state: _00_
   $stop();
end
```

(C) Draw a state diagram for an FSM that removes the *third* 1 from every consecutive string of 1's: [5 pt]


0/0
1/1
1/1
1/0
Reset
"0"
00
"1"
01
"11"
10
"111"
11
1/1
0/0
0/0
0/0