University of Washington – Computer Science & Engineering

Winter 2022 Instructor: Justin Hsia 2022-02-22

CSE 369 QUIZ 2

Name: _Perry_Perfect____________________

Student ID Number: _1234567________

Please do not turn the page until 12:20.

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 30 (+5) minutes to complete this quiz.

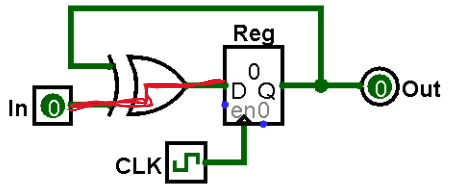
Advice

- 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	12	12
Total:	28	28

Question 1: Sequential Logic & Timing [6 pts]

Consider the following circuit diagram with a clock period of 100 ns (10^{-9} s), t_{setup} = 12 ns, t_{hold} = 6 ns, t_{C2Q} = 10 ns, and t_{XOR} = 20 ns. Assume that `In` is connected to an asynchronous input and can change at any point during a clock period (in the range [0, 100] ns).



(A) Calculate the earliest time in a clock period where changing `In` will cause a setup violation. *Your answer should be between 0 and 100 ns, inclusive.* [3 pts]

$\boxed{\text{68 ns}}$

The only path from `In` to the register input is shown above in red.
The setup time constraint violation range is $[t_{period} - t_{setup}, t_{period}] = [88, 100]$ ns.
Accounting for the t_{XOR} combinational logic delay, then the range where changing `In` will cause a setup violation is [68, 80] ns.

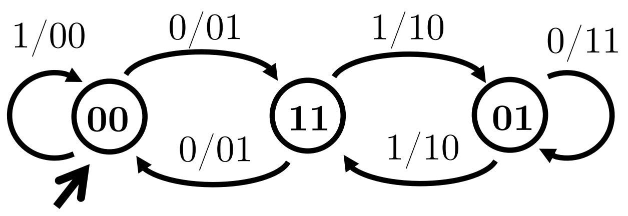
(B) Calculate the latest time in a clock period where changing `In` will cause a hold violation (in the *next* clock period). *Your answer should be between 0 and 100 ns, inclusive.* [3 pts]

$\boxed{\text{86 ns}}$

The hold time constraint violation range is $[0, t_{hold}] = [0, 6]$ ns.
This range for the *next* clock cycle would then be [100, 106] ns.
Accounting for the t_{XOR} combinational logic delay, then the range where changing `In` will cause a hold violation is [80, 86] ns.

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	1	1	0	1
0	0	1	0	0	0	0
0	1	0	0	1	1	1
0	1	1	1	1	1	0
1	0	0	X	X	X	X
1	0	1	X	X	X	X
1	1	0	0	0	0	1
1	1	1	0	1	1	0

(B) Complete the circuit diagram below using *minimal logic* based on the truth table shown below. Use only 2-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	1	0
0	1	1	X	X
1	0	0	X	X
1	0	1	0	0
1	1	0	1	1
1	1	1	1	1

Wire connection:

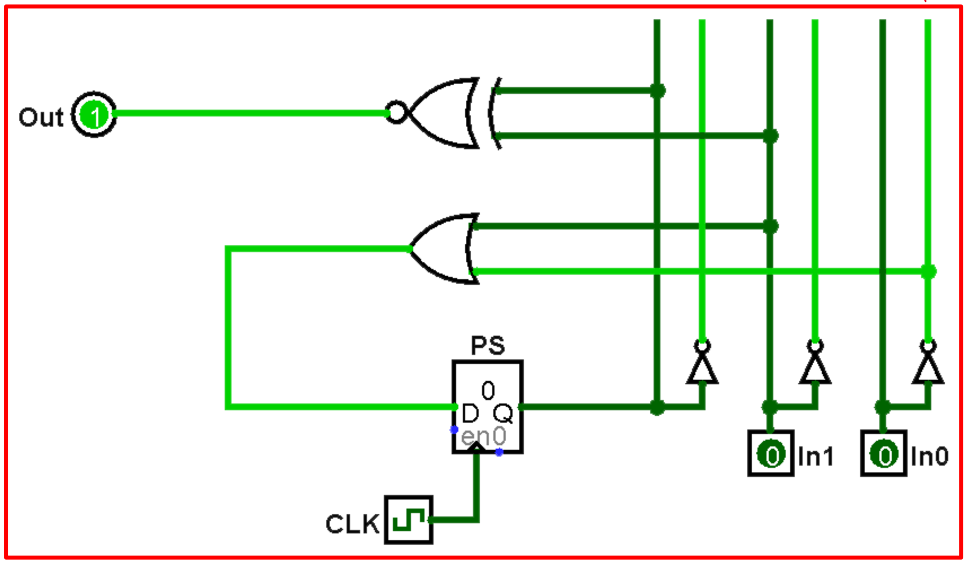
Wire crossing:

PS \ In	00	01	11	10
0	1	0	X	1
1	X	0	1	1

$NS = In_1 + \overline{In_0}$

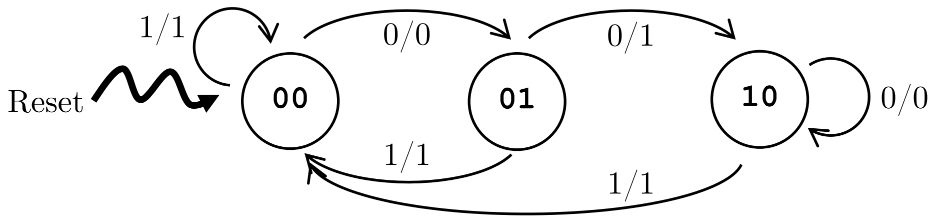
PS \ In	00	01	11	10
0	1	1	X	0
1	X	0	1	1

$Out = \overline{PS}\cdot\overline{In_1} + PS\cdot In_1$
$= PS \text{ xnor } In_1$



Question 3: Finite State Machine Design [12 pts]

The following FSM is a string manipulator; it outputs a modified version of its stream of inputs:



(A) Assume we pass the following stream of inputs immediately after resetting the FSM. What is the corresponding stream of outputs? [4 pt]

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

(B) As *briefly* as you can, describe how this FSM manipulates its input stream. [3 pt]

This FSM replaces the 2nd `0` in each string of consecutive `0`s with a `1`.

(C) Complete the testbench `initial` block to *thoroughly* test the state diagram. You need to fill in all bolded blanks. You are welcome to fill out the Verilog comments to help you keep track of state, but these will not be graded. [5 pts]

```
initial begin       // Soln 1           | // Soln 2
                    In <= 0;  // ps: 00 | In <= 0;  // ps: 00
    @(posedge clk); In <= 1;  // ps: 00 | In <= 0;  // ps: 01
    @(posedge clk); In <= 0;  // ps: 01 | In <= 0;  // ps: 10
    @(posedge clk); In <= 0;  // ps: 10 | In <= 1;  // ps: 10
    @(posedge clk); In <= 0;  // ps: 10 | In <= 0;  // ps: 00
    @(posedge clk); In <= 1;  // ps: 00 | In <= 1;  // ps: 01
    @(posedge clk); In <= 1;  // ps: 00 | In <= 1;  // ps: 00
    @(posedge clk); $stop();
end
```