University of Washington – Computer Science & Engineering

Spring 2018 Instructor: Justin Hsia 2018-05-15

CSE 369 QUIZ 2

Name: _Perry_Perfect________________

UWNetID: _perfect______________________

Please do not turn the page until 10:00.

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.

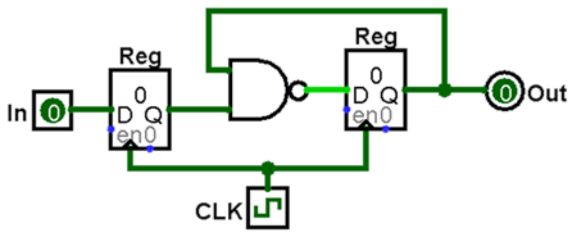
Advice

- 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	9	9
(3) FSM Design	11	11
Total:	**26**	**26**

Question 1: Sequential Logic & Timing [6 pts]

Consider the following circuit diagram with t_{period} = **150 ns** (10^{-9} s), t_{C2Q} = **45 ns**, t_{setup} = **15 ns**, and t_{hold} = **10 ns**.



(A) Assume that `In` does not violate any timing constraints. Calculate the **maximum NAND gate delay** that will allow the circuit to function correctly. Make sure to *include units.* [3 pts]

$t_{NAND,max}$ = **90 ns**

Both paths through the NAND are equally long.
So we need $t_{C2Q} + t_{NAND} \leq t_{period} - t_{setup}$.
Then $t_{NAND} \leq 150 - 15 - 45 = 90$ ns.

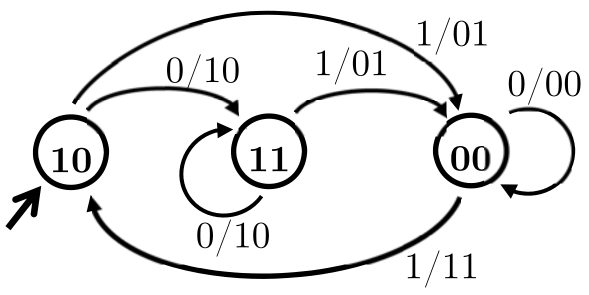
(B) Within what range of times (measured from each clock trigger) will changing the input `In` **not** cause a *timing violation*? Answer using inclusive interval notation: $[t_{start}, t_{end}]$. [3 pts]

[**10** , **135**] ns

By definition, a timing violation will occur if an input to a register changes in either the first t_{hold} or the last t_{setup} of the clock period. Since `In` is directly attached to a register input, we can directly use the provided timing constant values.

Question 2: Finite State Machine Implementation [9 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	1	0	1	1
0	1	0	X	X	X	X
0	1	1	X	X	X	X
1	0	0	1	1	1	0
1	0	1	0	0	0	1
1	1	0	1	1	1	0
1	1	1	0	0	0	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. [7 pts]

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

Wire connection:

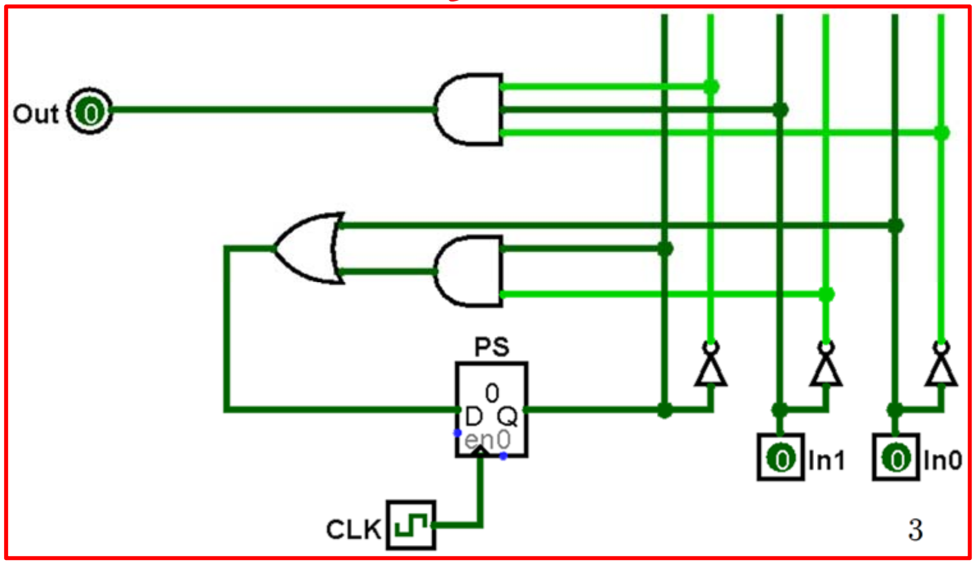
Wire crossing:

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

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

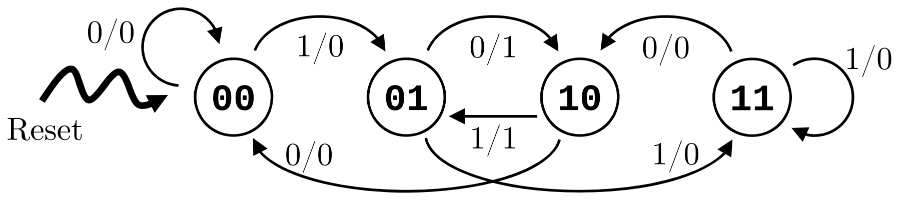
PS \ In	00	01	11	10
0	0	X	0	1
1	0	X	0	0

$Out = \overline{PS} \cdot In_1 \cdot \overline{In_0}$



Question 3: Finite State Machine Design [11 pts]

For this problem, consider the FSM below:



(A) Answer the following about the corresponding *truth table*. [2 pt]

2 state bits, 1 input bit.

Rows: **8**	Rows of Don't Cares: **0**

(B) Complete the testbench `initial` block to *thoroughly* test the state diagram. Even though they may be unnecessary, please fill in all 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
                    In <= 0____;    // state: 00
   @(posedge clk);  In <= 1;        // state: 00___
   @(posedge clk);  In <= 1;        // state: 01___
   @(posedge clk);  In <= 1;        // state: 11___
   @(posedge clk);  In <= 0____;    // state: 11___
   @(posedge clk);  In <= 1____;    // state: 10___
   @(posedge clk);  In <= 0____;    // state: 01___
   @(posedge clk);  In <= 0____;    // state: 10___
   @(posedge clk);                  // state: 00___
   $stop();
end
```

(C) What two input sequences does this FSM "recognize" (*i.e.* when it outputs a `1`)? [4 pt]

010	**101**