University of Washington – Computer Science & Engineering

Spring 2022 Instructor: Mark Wyse 2022-05-17

CSE 369 QUIZ 2

Name: ___SOLUTION_____

Student ID Number: _____

Please do not turn the page until 11:30.

Instructions

- This quiz contains 4 pages, including this cover page.
- Please write your name on and turn in all pages of scratch paper.
- 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 minutes to complete this quiz.

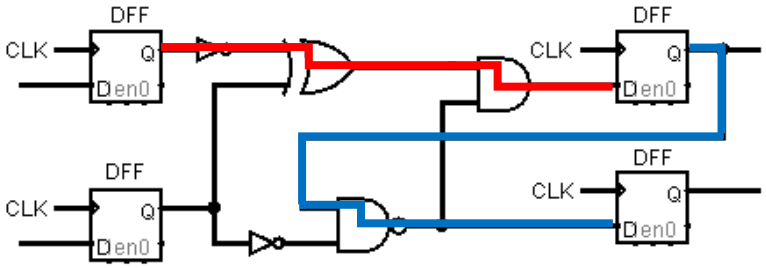
Advice

- 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	
(2) FSM Implementation	10	
(3) FSM Design	12	
Total:	**28**	

Question 1: Sequential Logic & Timing [6 pts]

Consider the following circuit diagram with $t_{\text{setup}} = \mathbf{8\ ns}$ (10^{-9} s), $t_{\text{C2Q}} = \mathbf{12\ ns}$, $t_{\text{NOT}} = \mathbf{4\ ns}$, $t_{\text{XOR}} = \mathbf{16\ ns}$, $t_{\text{NAND}} = \mathbf{8\ ns}$, and $t_{\text{AND}} = \mathbf{12\ ns}$.



(A) Calculate the **minimum clock period** that will allow the circuit to function correctly. Make sure to *include appropriate units in your answer*. [3 pts]

52 ns

Critical path is shown above in red.
We need $t_{C2Q} + t_{NOT} + t_{XOR} + t_{AND} \leq t_{period} - t_{setup}$
Then $t_{period} \geq 12 + 4 + 16 + 12 + 8 = 52\ ns$
[2 pts] longest path indicated
[1.5 pts] second longest path indicated = bottom left, XOR, AND, top right = 48 ns
[1 pts] 3rd longest path indicated = bottom left, NOT, NAND, AND, top right = 44 ns
[1 pts] correct computation for chosen path

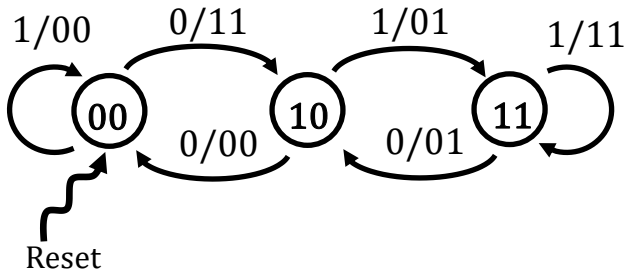
(B) Calculate the **maximum hold time** that will allow the circuit to function correctly. Make sure to *include appropriate units in your answer*. [3 pts]

20 ns

The shortest path to a register input is shown above in blue.
We need $t_{C2Q} + t_{NAND} \geq t_{hold}$
Then $t_{hold} \leq 12 + 8 = 20\ ns$
[2 pts] shortest path indicated
[1.5 pts] second shortest path indicated = bottom left, NOT, NAND, bottom right = 24 ns
[1 pts] 3rd shortest path indicated = bottom left, NOT, NAND, AND, top right = 36 ns
[1 pts] correct computation for chosen path

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	0	1	1
0	0	1	0	0	**0**	**0**
0	1	0	X	X	X	X
0	1	1	**X**	**X**	X	X
1	0	0	0	0	**0**	**0**
1	0	1	1	1	0	1
1	1	0	1	0	0	1
1	1	1	**1**	**1**	1	1

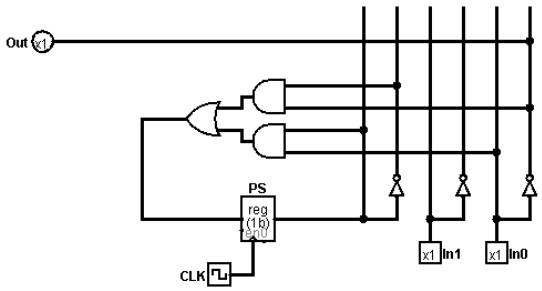
[0.25 pts each] Correct term

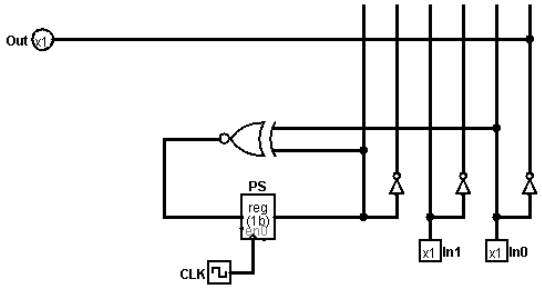
(B) Complete the circuit diagram below using *minimal logic* based on the truth table shown below. **Show a K-map for each of NS and Out and use only 2-input logic gates in the circuit implementation.** [8 pts]

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

Wire connection:

Wire crossing:





For both NS and Out:
[1 pts] Truth table transcribed to K-Map correctly
[2 pts] Correct simplification
[1 pts] Correct circuit diagram

$Out = \overline{In_0}$

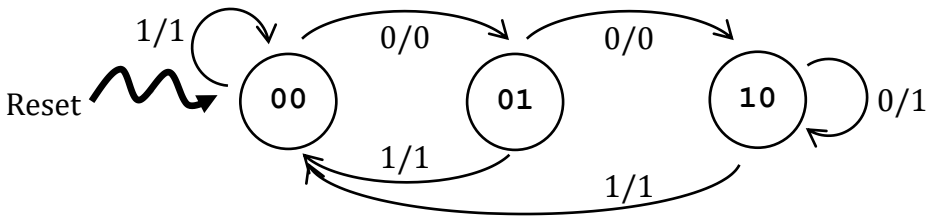
$NS = \overline{PS} \cdot \overline{In_0} + PS \cdot In_0$

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

NS: PS \ In	00	01	11	10
0	1	0	0	1
1	0	1	1	0

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 (left-to-right) immediately after resetting the FSM. What is the corresponding stream of outputs? [4 pts]

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

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

This FSM replaces the 3rd and all following 0's in each string of consecutive 0s 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
                        Reset <= 1; In <= 1;
      @(posedge clk);
      @(posedge clk);   Reset <= 0;   // state: 00
      @(posedge clk);   In <= 0;      // state: 00
      @(posedge clk);   In <= 1;      // state: 01
      @(posedge clk);   In <= 0;      // state: 00
      @(posedge clk);   In <= 0;      // state: 01
      @(posedge clk);   In <= 0;      // state: 10
      @(posedge clk);   In <= 1;      // state: 10
      @(posedge clk);   $stop();
end
```

Q3-C notes see next page

Solution must test:

- 00-01-00 loop, In = 0, 1 from 00
- 10 self-transition, In = 0 from 10
- 00-01-10-00 loop, In = 0, 0, 1 from 00
- 00 self-transition is tested by provided code. Reset goes low, but In is still 1, thus posedge clk before first blank tests the 00 self-transition
- 10 self-transition is only possible in a sequence of In = 0, 0, 0
- 00-01-10-00 loop must include self-transition test at 10, otherwise it is not possible to cover all test cases

Possible solutions:

In = 0, 1, 0, 0, 0, 1

In = 0, 0, 0, 1, 0 , 1