University of Washington – Computer Science & Engineering
Spring 2024 Instructor: Justin Hsia 2024-05-14

CSE 369 QUIZ 2

Name: ______________________________

Student ID Number: ________________

Please do not turn the page until 2:20.

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.

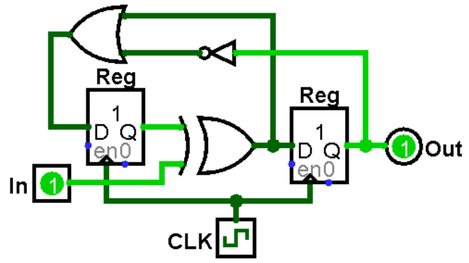
Advice

- 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	
(2) FSM Implementation	10	
(3) FSM Design	11	
Total:	**27**	

Question 1: Sequential Logic & Timing [6 pts]

Consider the following circuit diagram with t_{setup} = **11 ns**, t_{C2Q} = **9 ns**, t_{NOT} = **3 ns**, t_{OR} = **8 ns**, and t_{XOR} = **10 ns**. Assume that `In` changes **7 ns** after every clock trigger.



(A) Calculate the **minimum clock period** that will allow the circuit to function correctly. [3 pts]

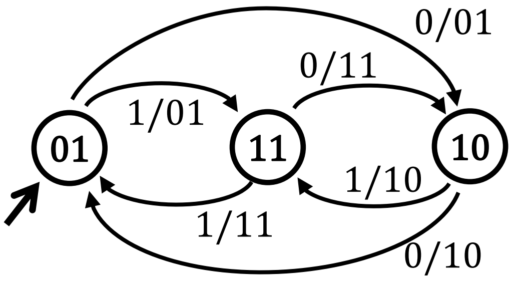
ns

(B) Calculate the **maximum hold time** (t_{hold}) that will allow the circuit to function correctly. [3 pts]

ns

Question 2: Finite State Machine Implementation [10 pts]

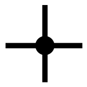
(A) Fill in the provided truth table based on the FSM shown. [2 pts]

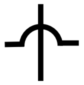


PS_1	PS_0	In	NS_1	NS_0	Out_1	Out_0
0	0	0	X	X	X	X
0	0	1	X	X		
0	1	0			0	1
0	1	1	1	1	0	1
1	0	0	0	1		
1	0	1	1	1	1	0
1	1	0	1	0	1	1
1	1	1			1	1

(B) Complete the circuit diagram below using *minimal logic* based on the truth table shown below. **Use only 2-input logic gates.** [8 pts]

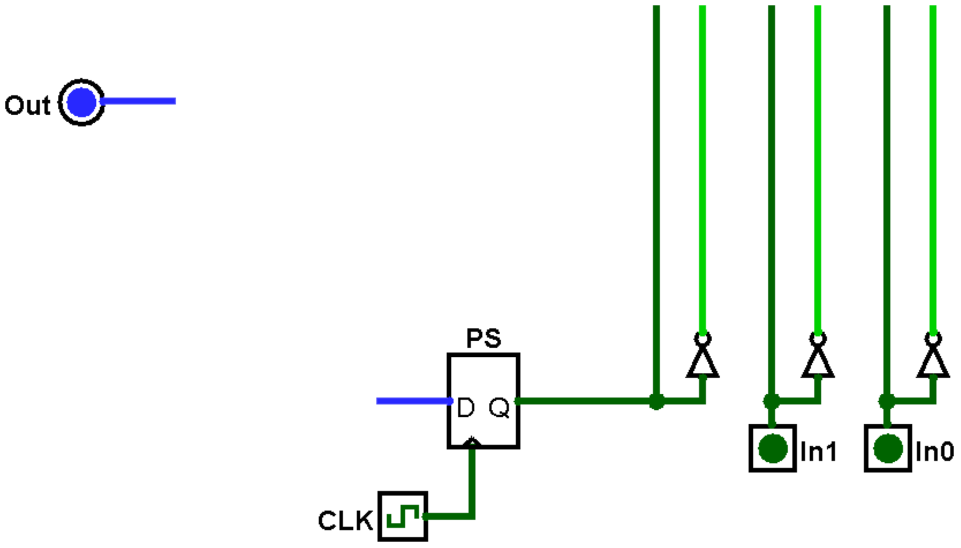
PS	In_1	In_0	NS	Out
0	0	0	1	0
0	0	1	0	1
0	1	0	X	X
0	1	1	0	0
1	0	0	X	X
1	0	1	1	0
1	1	0	1	0
1	1	1	0	0

Wire connection: 

Wire crossing: 

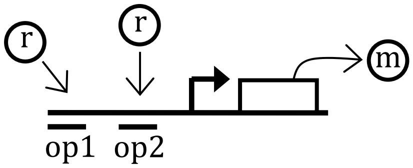
	00	01	11	10
0				
1				

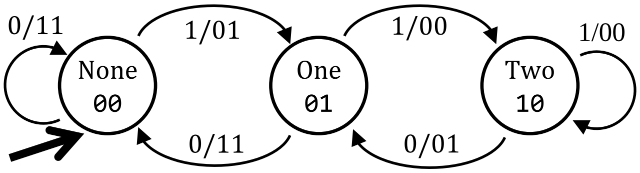
	00	01	11	10
0				
1				



Question 3: Finite State Machine Design [11 pts]

The following FSM represents a DNA construct known as a *promoter*, whose output level of `m` is determined inversely to the number of *repressors* (`r`) bound to its two binding sites (op1 and op2). The value of the one-bit input `In` represents a repressor binding (`1`) or a repressor unbinding (`0`).





(A) How many total rows are in the truth table for this FSM? How many of the rows are filled with Don't Cares? [2 pts]

Rows:	Don't Care Rows:

(B) Complete the test bench `initial` block to *thoroughly* test the FSM by filling in all bolded blanks. You may fill out the comments to track the state, but these won't be graded. Don't worry about situations we don't expect to see during normal operation. [5 pts]

```
initial begin
                       In <= 0;     // state: 00
      @(posedge clk);  In <= ____;  // state: ____
      @(posedge clk);  In <= 0;     // state: ____
      @(posedge clk);  In <= 1;     // state: ____
      @(posedge clk);  In <= ____;  // state: ____
      @(posedge clk);  In <= 1;     // state: ____
      @(posedge clk);  In <= ____;  // state: ____
      @(posedge clk);               // state: ____
      $stop();
end
```

(C) Is there any way that the hardware implementation could output `10`? *Briefly* explain. [2 pts]

(D) Consider the limitations of FSMs in representing certain systems. Name one unrealistic behavior (compared to a real promoter) that is implied by the operation of this FSM. [2 pts]