CSE 369 QUIZ 2

(VERSION B)

Name: *Serafina Solutionario*

Student ID Number: 369369

Please do not turn the page until 4:05.

Instructions
- This quiz contains **6** 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, smart glasses, watches, and other digital wearables.
- 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. If you've been practicing, you got this. If you haven't, you'll learn something now.
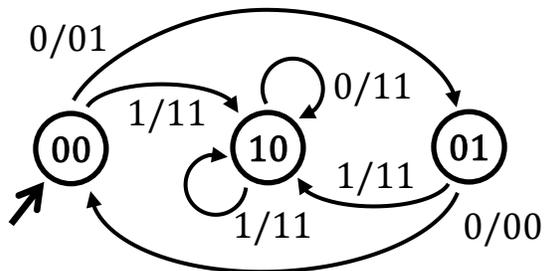
Question	Points	Score
(1) SL & Timing	9	
(2) FSM Implementation	14	
(3) FSM Design	7	
Total:	30	

(This page left intentionally blank)

Question 1: Sequential Logic & Timing [9 pts]

Consider the following circuit diagram with $t_{setup} = $ **6 ns**, $t_{C2Q} = $ **7 ns**, $t_{NOT} = $ **2 ns**, and $t_{XOR} = $ **10 ns**. Assume that `In` changes **9 ns** after every clock trigger and that there are no timing requirements for `Output` ports. Assume all CLK inputs are connected to the same clock source.

(A)   Calculate the **minimum clock period** that will allow the circuit to function correctly. [4 pts]

$$\boxed{25 \quad \text{ns}}$$

$$t_{Input} + t_{XOR} \leq t_{period} - t_{setup}$$
$$9 + 10 \leq t_{period} - 6$$
$$25 \leq t_{period}$$

(B)   Calculate the **maximum hold time** (t_{hold}) that will allow the circuit to function correctly. [4 pts]

$$\boxed{15 \quad \text{ns}}$$

$$t_{C2Q} + 4 * t_{NOT} \geq t_{hold}$$
$$7 + 8 \geq t_{hold}$$
$$15 \geq t_{hold}$$

(C)   What is the purpose of the inverter chain in the top path of the circuit? [1 pt]

To relax hold time constraints: without the inverters, the output would pass from the left-most flip-flop to the top right flip-flop too quickly. With the inverters, we can accept flip-flop designs with larger (more demanding) hold time constraints.

Remember, you got this 💪 😊

Question 2: Finite State Machine Implementation [14 pts]

(A)   Fill in the provided truth table based on the FSM shown. [8 pts]

Typo! This should be a 1

PS$_1$	PS$_0$	In	NS$_1$	NS$_0$	Out$_1$	Out$_0$
0	0	0	0	1	0	1
0	0	1	1	0	1	0
0	1	0	0	0	0	0
0	1	1	1	0	1	1
1	0	0	1	0	1	1
1	0	1	1	0	1	1
1	1	0	x	x	x	x
1	1	1	x	x	x	x

(B)   Complete the circuit diagram below using *minimal logic* based on the truth table shown below. **Use only 2-input logic gates.** [6 pts]

PS	In$_1$	In$_0$	NS	Out
0	0	0	0	1
0	0	1	0	1
0	1	0	1	1
0	1	1	1	1
1	0	0	x	x
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Wire connection:

Wire crossing:

NS

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

Out

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

Remember, you got this 💪 😊
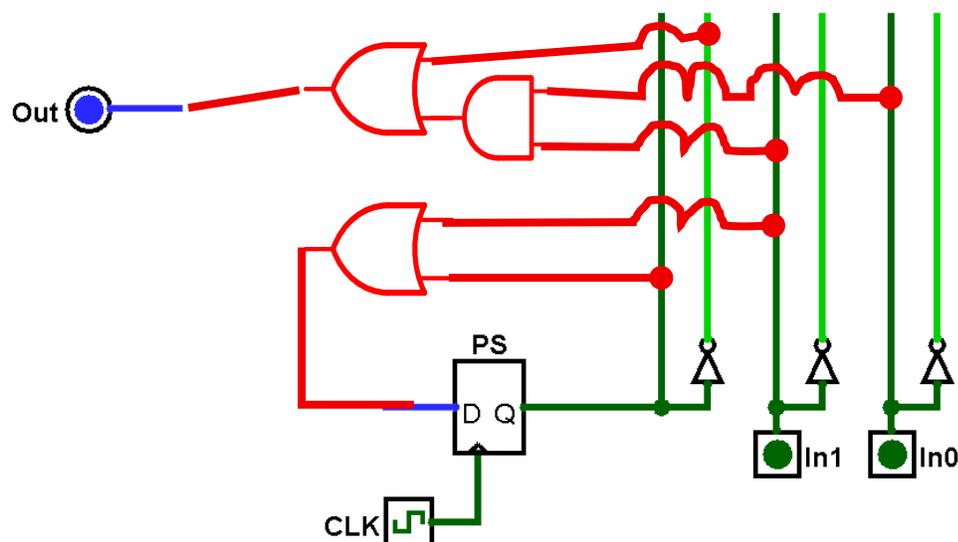
Question 3: Finite State Machine Design [7 pts]

We've built an FSM representing what will happen if we pet Danni the Cat at various times. The input bit represents the presence of your hand petting her. The output bits represent her state of {claws_out, purring}
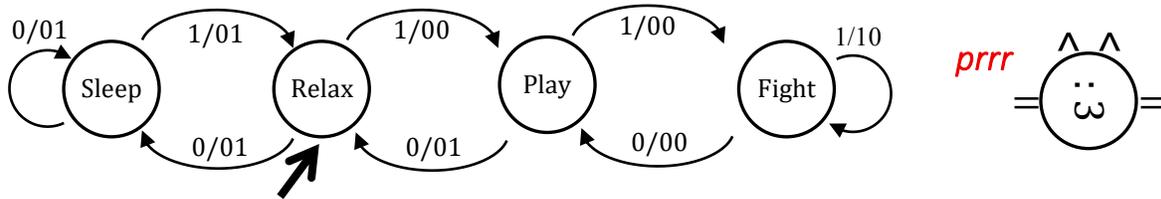
(A) How many total rows are in the truth table for this FSM? How many of the rows are filled with Don't Cares? [2 pts]

2 bits for state code, 1 input bit = 3 bits total. $2^3 = 8$. All states and transitions are accounted for in the diagram, so none are don't-care

Rows: **8**	Don't Care Rows: **0**

(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. [4 pts]

```
initial begin
                            In <= 0;      // state: relax
        @(posedge clk);     In <= 0;      // state: sleep
        @(posedge clk);     In <= 1;      // state: sleep
        @(posedge clk);     In <= 1;      // state: relax
        @(posedge clk);     In <= 1;      // state: play
        @(posedge clk);     In <= 1;      // state: fight
        @(posedge clk);     In <= 0;      // state: fight
        @(posedge clk);     In <= 0;      // state: play
        @(posedge clk);                   // state: relax
        $stop();
    end
```

(C) Based on the two outputs of this FSM alone, what advice would you give to the groomer who wanted to get Danni's claws out so they could be clipped? [1pt]

Just pet her continuously and eventually her claws will come out

(This page left intentionally blank)