

Name \_\_\_\_\_

EE2700 Final Exam (Spring 2012). 6 Pages. Open book. Open Notes. Closed Internet. *Cheating will not be tolerated and will result in grade of 0 for this exam.*

1. (4 pts) (a) Convert  $-107$  to an 8-bit 2's complement number. (b) Represent that number in (unsigned) hexadecimal. Show your work.

(a) \_\_\_\_\_

(b) \_\_\_\_\_

2. (4 pts) (a) Add the following 8-bit 2's complement numbers. Show your work. (b) Convert both addends and the sum to decimal. (c) State whether or not overflow occurred.

$$\begin{array}{r} 11010111 \\ +01100101 \\ \hline \end{array}$$

3. (6 pts) Loop and give the simplest SOP expression for each of the following maps:

(a)

		AB			
		00	01	11	10
CD	00	1	0	0	1
	01	0	X	1	1
	11	0	0	1	X
	10	1	0	0	1

F = \_\_\_\_\_

(b)

		AB			
		00	01	11	10
CD	00	X	$\bar{E}$	$\bar{E}+xE$	0
	01	$x\bar{E}$	0	$x\bar{E}$	$x\bar{E}+E$
	11	0	X	E	1
	10	$xE$	$x\bar{E}$	1	$xE$

F = \_\_\_\_\_

4. (3 pts) Name the property or theorem that justifies each of the following logical equations:

(a)  $XY + X\bar{Y} = X$  \_\_\_\_\_

(b)  $X\bar{Y} = \bar{Y}X$  \_\_\_\_\_

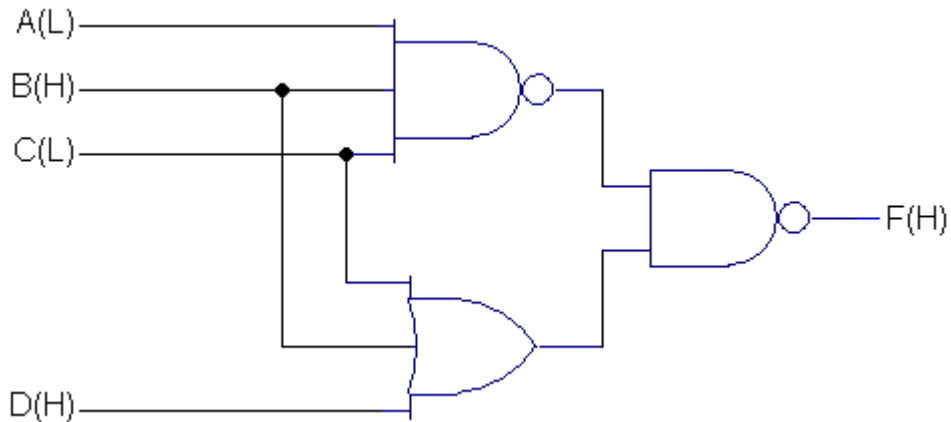
(c)  $\overline{X + Y + Z} = \bar{X}\bar{Y}\bar{Z}$  \_\_\_\_\_

5. (4 pts) Simplify the following expressions using logical identities, properties and/or theorems:

(a)  $AC + A\overline{B}C\overline{D}$  = \_\_\_\_\_

(b)  $(\overline{ABC} + \overline{AD})(\overline{ABC} + \overline{AD})$  = \_\_\_\_\_

6. (4 pts) Analyze the circuit below. Simplify to a sum of products expression.

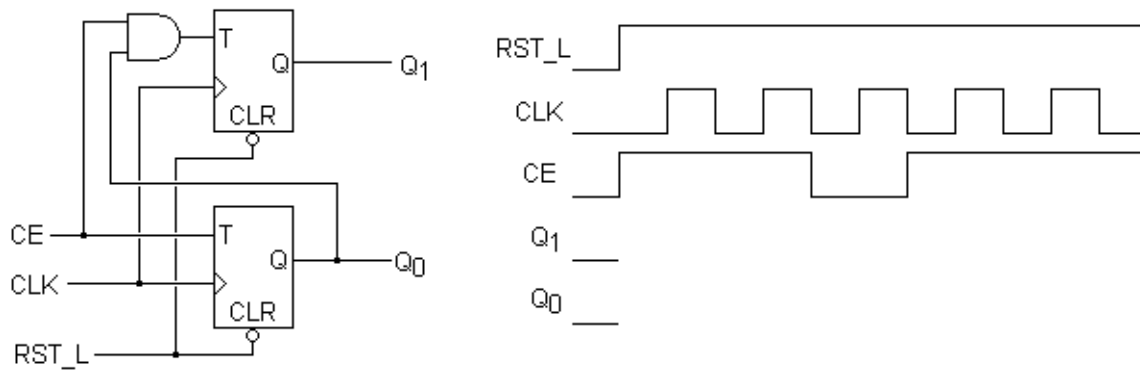


F = \_\_\_\_\_

7. (2 pts) If NAND gates have a propagation delay of 5ns and OR gates have a propagation delay of 7 ns, what is the total propagation delay for the circuit in problem 6?

\_\_\_\_\_ ns

8. (5 pts) Complete the timing diagram below for Q<sub>1</sub> and Q<sub>0</sub>.



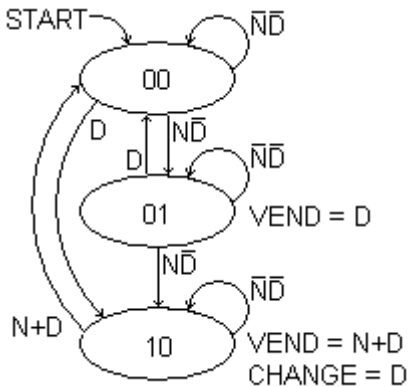
9. (5 pts) List (in order) the 5 design steps we discussed in class.

1. \_\_\_\_\_
2. \_\_\_\_\_
3. \_\_\_\_\_
4. \_\_\_\_\_
5. \_\_\_\_\_

10. (2 pts) For a flip-flop, the time that the data must be stable before the rising edge of the clock is called:

- (a) Propagation delay.
- (b) Hold time.
- (c) Setup time.
- (d) Party time.

11. The state machine below represents the controller for a 15¢ chewing gum vending machine that accepts nickels and dimes. Whenever 15¢ or more is inserted, the VEND output causes the gum to be dispensed. If 20¢ is inserted, the CHANGE output causes a nickel to be returned. Note that inputs N and D pulse one clock cycle each time a nickel or dime is inserted, respectively.



(a) (8 pts) Complete the present state/next state/output table below.

Present State		Inputs		Next State		Outputs	
Q <sub>1</sub>	Q <sub>0</sub>	N	D	NS <sub>1</sub>	NS <sub>0</sub>	VEND	CHANGE
0	0	0	0				
0	0	0	1				
0	0	1	0				
0	0	1	1				
0	1	0	0				
0	1	0	1				
0	1	1	0				
0	1	1	1				
1	0	0	0				
1	0	0	1				
1	0	1	0				
1	0	1	1				

(b) (6 pts) Use the table from (a) to draw Karnaugh maps for “next state” bits NS<sub>0</sub> and NS<sub>1</sub>.

(c) (6 pts) Use the original state diagram to draw VEMs for VEND and CHANGE.

(d) (6 pts) Using your results from (b) and (c), write the logic equations for  $NS_1$ , VEND and CHANGE. (Note: The equation for  $NS_0$  is given later, so do not find it.)

(e) (5 pts) The logic equation for next state bit,  $NS_0$ , is:

$$NS_0 = Q_0 \overline{N} \overline{D} + \overline{Q_1} \overline{Q_0} N \overline{D}.$$

Assume  $Q_1, \overline{Q_1}, N, \overline{N}, D$  and  $\overline{D}$  (asserted high) are all available. Design a circuit that generates  $Q_0$  and  $\overline{Q_0}$  using logic gates and a D flip-flop. Use only NAND or NOR gates (no inverters, and you may not use a gate as an inverter).

(e) (3 pts) Using your results from (d) and (e), find the minimum clock period (the time between rising edges of the clock) for this state machine. Assume NAND and NOR gates have the following propagation delays: 2-input: 5ns, 3-input: 6ns and 4-input: 7ns. Also assume a clock-to-Q propagation delay of 10ns and a setup time of 3ns. For simplicity, neglect propagation delay through inverters.

---

- (f) (15 pts) Complete the VHDL module for this state machine, below, without using the equations you developed in part (d). The inputs nickel and dime are asynchronous and may be asserted for multiple clock cycles, so you will need to use the single-pulse technique we discussed in class to generate synchronous signals N and D that are active for just one clock cycle. Assume the outputs are asserted low. Use the back of this page if you need more room.

```
entity gum_vend is
```

```
  port (clk, reset_L, nickel, dime: in std_logic; vend_L, change_L: out std_logic);  
end gum_vend;
```

```
architecture behavior of gum_vend is
```

```
  type state_type is (home, have_5_cents, have_10_cents); -- State names
```

```
end behavior;
```

12. (5 pts) Draw a state diagram for a Moore state machine that detects the sequence 0110, where the final 0 in one sequence may serve as the first zero in the next. (e.g. your state machine should detect 0110 twice in the sequence 0110110.) Hint: you will need 5 states.

13. (2 pts) If the state machine in problem 12 uses the minimum number of bits to encode the state, how many flip-flops will be needed?

---

14. (2 pts) If the state machine in problem 12 uses a “one-hot” state assignment, how many flip-flops will be needed?

---

15. (3 pts) Why do tools like ISE favor using one-hot state assignments?

---

---

---

Extra Credit (3 pts) The processor we designed in this class cannot handle arrays or pointers. To do that, we would need *indirect addressing*, in which the instruction refers to a memory location that contains the *address* of the data instead of the data itself.

Draw the extra states for the controller you did in Project 3 to implement the instruction *load indirect* (i.e. the second byte of the instruction names a memory location that in turn names the memory location to load).