EECS 270 *Final Exam*

Spring 2014

Name: _____ unique name: _____

Sign the honor code:

I have neither given nor received aid on this exam nor observed anyone else doing so.

_____

Scores:

Problem #	Points
1	/12
2	/7
3	/8
4	/14
5	/15
6	/7
7	/13
8	/12
9	/12
Total	**/100**

NOTES:

1. Open book (our text only) and Open notes
2. Calculators are allowed, but no PDAs, Portables, Cell phones, etc.
3. Don't spend too much time on any one problem.
4. You have about 120 minutes for the exam.
5. *Be sure to fully label any MSI device you use.*
   Be sure to show work and explain what you've done when asked to do so

1) Fill in each blank or circle the best answer. **[12 points, -2 per wrong or blank answer, min 0]**

   a) The 4-bit 2's complement representation for -7 is **_____.**

   b) DRAM cells tend to be ***larger / smaller*** than SRAM cells. DRAM cells are also generally ***faster / slower*** than SRAM cells.

   c) A*B + A*C' in canonical sum-of-products form is **_____.**

   d) Consider the an error correction scheme where P(…) is the even one's parity function (as in class) and there are 4 bits of data {ABCD}. If we are correcting 1 bit of error, and two of the error correction bits are sent as X=P(B,C,D), Y=P(A,C,D) then it must be the case that

      Z=P**(_____)** or Z=P**(_____)**

   e) If we took the carry look ahead adder described in class and implemented it as a 64-bit adder

      (using CLA logic at each level), we'd have a worst case delay of _____ gates.

   f) When building a 512 by 2 memory out of a square memory of minimum size, the row

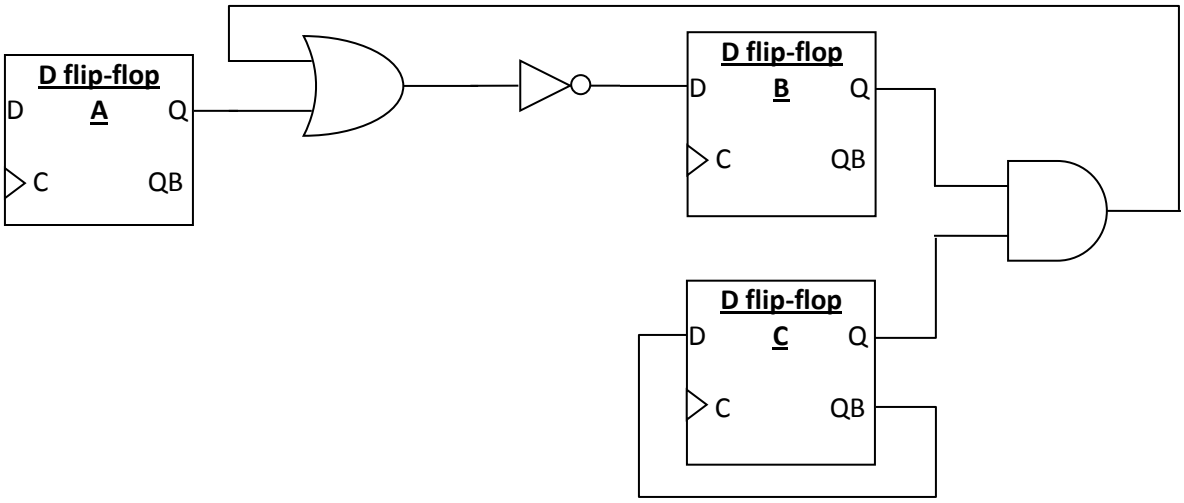
      decoder will have _____ bits of input while the column MUX will have _____

      bits needed for the selector.

2) Using a K-map, find the minimal sum-of-products of $\Sigma_{(W,X,Y,Z)}$ (1,3,4,5,9,10,11,12)+d(2,6,13).
[7 points]

3) Design a state transition diagram for a state machine which has one input “A” and one output “Z”. Z should go high if the most recent inputs were either “00111” or “110”. For full credit, you should use as few states as possible. **[8 points]**

4) Sequential circuit timing **[14 points]**



Consider flip-flops A, B and C each nominally clocked off of the same clock. Assume:

- Each flip-flop has a set-up time of 5ns and a clock-to-Q delay of 3ns to 4ns.
- The AND and OR gates each have a delay of 2 to 6 ns.
- The NOT gate has a delay of 1 to 5 ns.

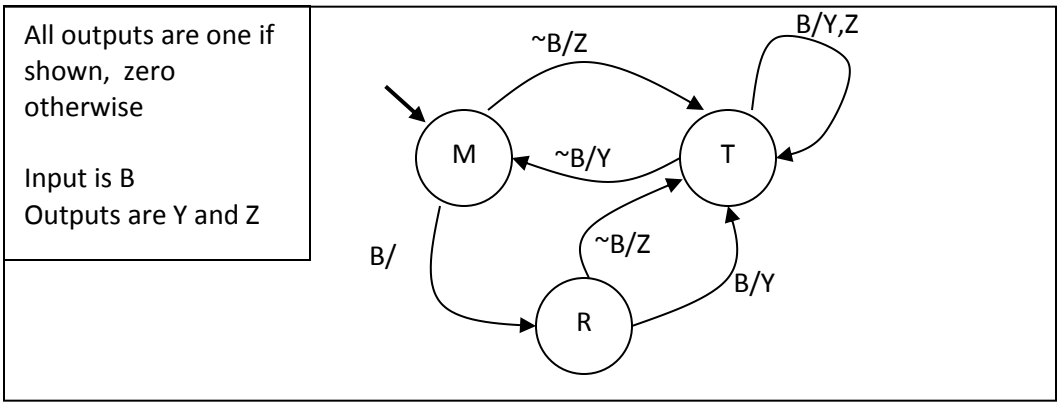
a) What is the shortest clock period you could safely clock this system at? Show your work. **[3]**

b) What is the (non-negative) range of values for the flip-flop's hold time would be sufficient? Show your work. **[3]**

c) Redo part a) assuming flip-flop A's rising edge may be as much as 3.25 ns after B and C's rising edge or as much 1.5 ns before (B and C's edge are simultaneous). Again, show your work. **[4]**

d) Redo part b) assuming flip-flop A's rising edge may be as much as 3.25 ns after B and C's rising edge or as much 1.5 ns before (B and C's edge are simultaneous). Again, show your work. **[4]**

5) Converting a state transition diagram to logic.



You are to assign state bits S[1:0] as 00 for state M, 11 for state R, and 01 for state T. You are to find logic equations for the next state (NS[1:0]) and the outputs (Y and Z) and put them all in ***project-of-sums*** form. Neatly use K-maps to do the minimization. Clearly show your work and provide your final answer where shown. You don't care what happens if the machine somehow gets into state 10. **[15 points]**

NS1=_____ (minimal PoS)

NS0=_____ (minimal PoS)

Y =_____ (minimal PoS)

Z =_____ (minimal PoS)

6) Following the rules for CMOS and using only transistors, design a device that computes F=A'*B'+C. You may freely use the values A, B, C as inputs, but not their inverses. Your design must use *eight* or fewer transistors to receive credit. **[7 points]**

7) Answer the following questions using Verilog. You will be graded for correctness, syntax and efficiency of you design and code. (The second part of this question is found on the next page). **[13 points]**
   a) Write a Verilog module called **Tff.** It is to implement a T flip-flop with enable. **[5]**
      - Inputs are **clock** and **en.**
      - Output is the flip-flop's value **Q**.

b) Write a Verilog module called Dff. It is to implement a D flip-flop with enable by instantiating the T flip-flop you designed in the previous part. ***It may not use any always blocks*** *and the state must be stored in the T flip-flop*. **[8]**
   - Inputs are **clock, D** and **en.**
   - Output is the flip-flop's value **Q**.

8) You wish to design a state machine that has one input "A" and one Moore-type output "Z". Z should be a 1 if and only if the last 6 values of A were "0". Due to budget cuts, you only have the following devices available:
    - 2-input gates of any standard type (AND, OR, NOR, NAND, XOR, and XNOR).
    - Inverters
    - Modulo-16 counters with enable and reset
    - 4 to 16 decoders
    - 2-to-1 MUX (1-bit wide).

   Using as few devices as possible, implement the state machine described above. Solutions which use more than 4 devices will receive no credit. **[12 points]**

9) Using the "CLB" (complex logic block) below, implement both F and G:
   - F=A*B*C+B'*C+D
   - G=A*B*C*D.

   You are to fill in the blank table entries (in the input/output table, the 8x2 memories, and the switch matrix). **[12 points]**

P0	P1	P2	P3	P4	P5	P6	P7	P8	P9