Name: Email: ID#

UNIVERSITY OF CALIFORNIA
Department of Electrical Engineering and Computer Sciences
EECS150 Fall 2001

Prof. Subramanian

**Midterm II**

1) You are implementing an 4:1 Multiplexer that has the following specifications:
   Inputs: $I_0$-$I_3$
   Output: Z
   Control Inputs: $C_2$-$C_0$ ($C_2$ is MSB)
   The input is selected based on a Johnson counter scheme (000→$I_0$, 100→$I_1$, 110→$I_2$, 111→$I_3$). All other states select $I_0$.

   a) Write out the functional form of the truth table for this multiplexer

4 pts

| $C_2$ | $C_1$ | $C_0$ | Z |
|---|---|---|---|
| **0** | 0 | 0 | $I_0$ |
| **0** | 0 | 1 | $I_0$ |
| **0** | 1 | 0 | $I_0$ |
| **0** | 1 | 1 | $I_0$ |
| **1** | 0 | 0 | $I_1$ |
| **1** | 0 | 1 | $I_0$ |
| **1** | 1 | 0 | $I_2$ |
| **1** | 1 | 1 | $I_3$ |

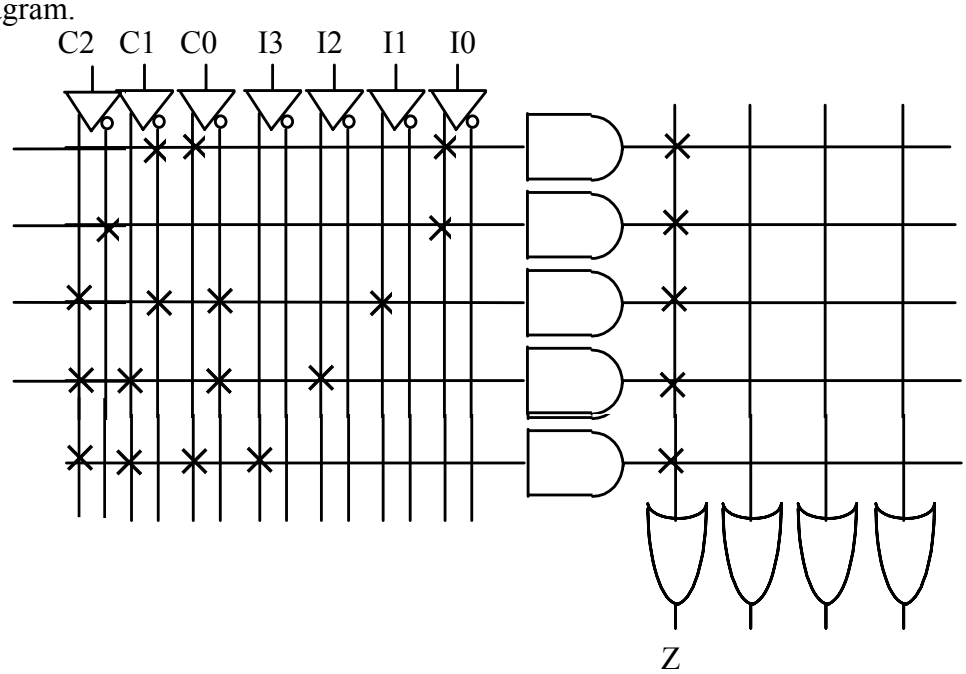
   b) Determine the output Boolean function. Simplify as appropriate.

4 pts

$Z=C_2.C_1.C_0.I_3+C_2.C_1.C_0'.I_2+C_2.C_1'.C_0'.I_1+C_2'.I_0+C_1'.C_0.I_0$

   c) Implement the multiplexer using the PLA below. Indicate connections using the standard cross scheme used in class. Indicate the inputs, product terms, and outputs on the diagram.

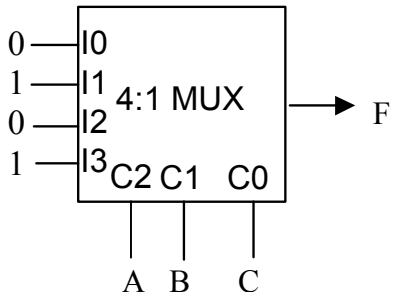
4 pts



Name: Email: ID#

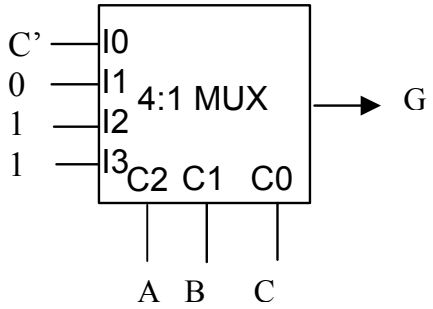
d) Implement F(A,B,C) = Σm(4, 7) using the above multiplexer. Indicate the connections in the diagram below. Simplify as appropriate.

4 pts



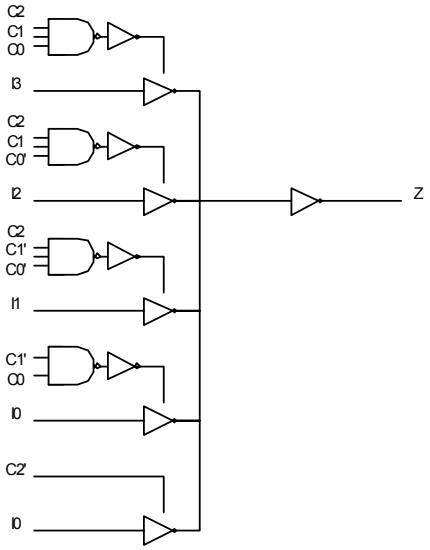

e) Implement G(A,B,C) = Σm(0, 2, 6, 7) using the above multiplexer. Indicate the connections in the diagram below. Simplify as appropriate.

6 pts




f) Implement the multiplexer using NAND gates (and inverters) and Tri-State Inverters. Draw the circuit below (Assume Tri-State Inverters are active-high enabled)

4 pts




g) Assume Tri-State Inverters require 4 transistors. How many transistors are required to implement the multiplexer (Do not count complemented external inputs, of course):

4 pts

i) Using Tri-State Inverters / NAND gates: 52 transistors

ii) Using NAND / NAND gates: 44 transistors

Name: Email: ID#

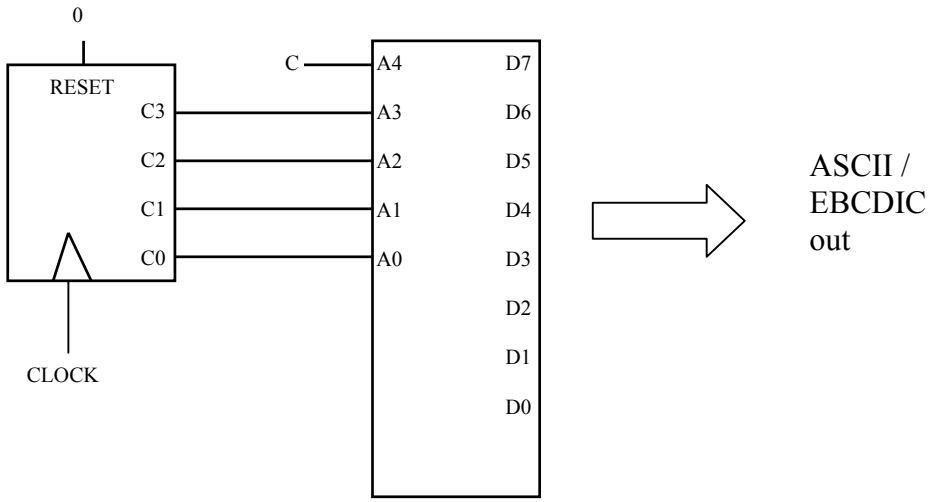
2) Back in the days of mainframe computing, there were two standard for textual information interchange – ASCII and EBCDIC. You are implementing a counter that counts in hexadecimal. The output of the counter is either in ASCII (0-9 = 48-57, A-F = 65-70) or EBCDIC (0-9 = 240-249, A-F = 193-198) depending on the value of a control switch (C = 0 → ASCII, C = 1 → EBCDIC). You are provided with the following parts:

- A 4-bit binary counter (Pins: CLK, RESET, C3-C0 outputs)
- A 4:16 DEMUX (Pins: G enable, S3-S1 select lines, Z output)
- A 32×8 bit ROM (Pins: A4-A0 address lines, D7-D0 data lines)
- As many NAND gates as you need

You may use some or all of the parts above. You may choose to program the ROM with whatever values you need.

a) Design the ASCII/EBCDIC counter. Use block diagrams for the individual parts and label the pinouts.

7 pts




b) Fill out the following table with the values that you chose to store in the ROM. You should write the data in decimal for convenience (and ease of grading).

8 pts

| Address | Data | Address | Data | Address | Data | Address | Data |
|---|---|---|---|---|---|---|---|
| 00000 | 48 | 01000 | 56 | 10000 | 240 | 11000 | 248 |
| 00001 | 49 | 01001 | 57 | 10001 | 241 | 11001 | 249 |
| 00010 | 50 | 01010 | 65 | 10010 | 242 | 11010 | 193 |
| 00011 | 51 | 01011 | 66 | 10011 | 243 | 11011 | 194 |
| 00100 | 52 | 01100 | 67 | 10100 | 244 | 11100 | 195 |
| 00101 | 53 | 01101 | 68 | 10101 | 245 | 11101 | 196 |
| 00110 | 54 | 01110 | 69 | 10110 | 246 | 11110 | 197 |
| 00111 | 55 | 01111 | 70 | 10111 | 247 | 11111 | 198 |

Name: Email: ID#

3) Consider the following truth table:

| Inputs | | | Output |
|---|---|---|---|
| A | B | C | Z |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

a) Minimize the truth table using a K-Map. Identify the prime implicants and essential prime implicants on the map.

6 pts

| A\BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |

Essential Prime Implicant

Non-essential Prime Implicant

b) Write a minimal SoP function (Z1) that describes the above truth table, and an SoP function (Z2) that has product terms covering all the prime (essential and non-essential) implicants.

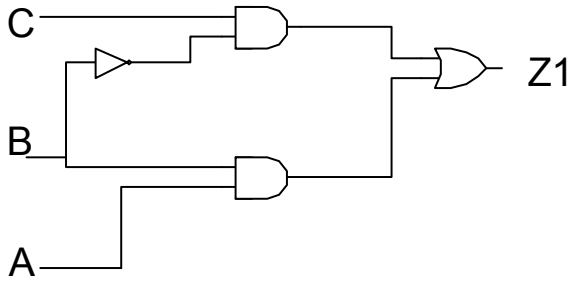
4 pts

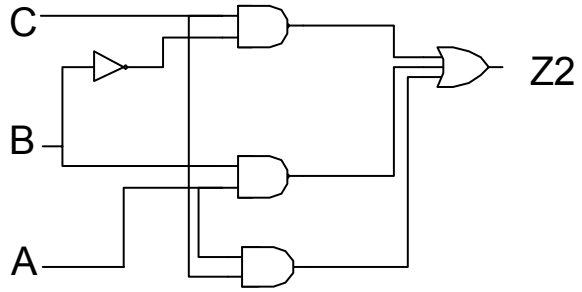
Z1(A,B,C) = B'.C+A.B

Z2(A,B,C) =B'.C+A.B+A.C

c) You are not provided with complemented inputs, so you will need to use inverters as necessary. Draw circuits that implement Z1 and Z2 using AND gates, OR gates, and Inverters.

4 pts

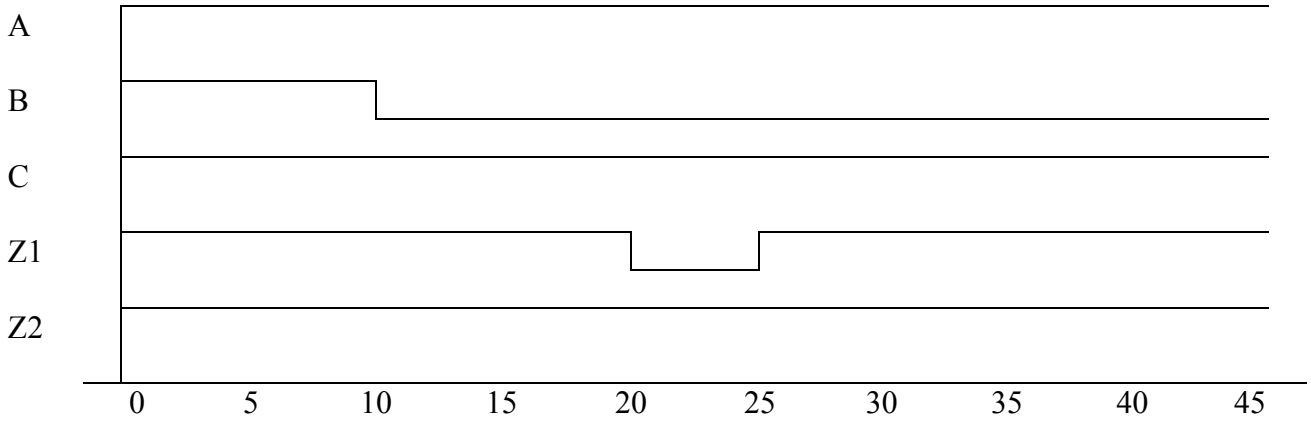


Name: Email: ID#

d) Complete the timing diagram below, that shows the transition from ABC = 111 to ABC = 101 for both Z1 and Z2. Assume the propagation delay is 5ns, independent of gate type.

6 pts

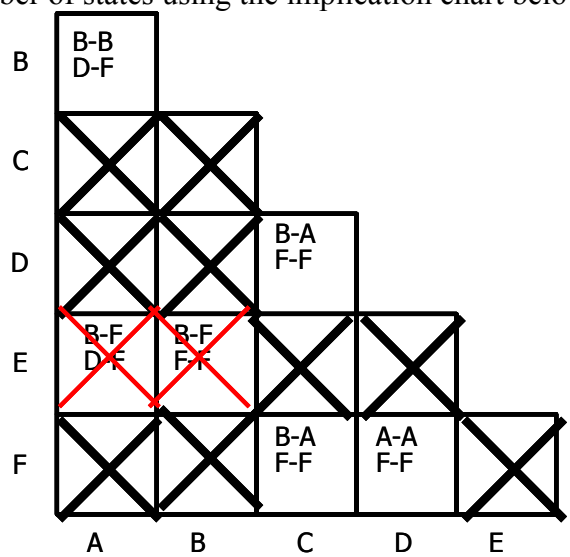



4) Consider a finite state machine defined by the following table (X= don't care):

| Present State | Next State | | Output | |
|---|---|---|---|---|
| | IN=0 | IN=1 | IN=0 | IN=1 |
| **A** | B | D | 0 | 0 |
| **B** | B | F | 0 | 0 |
| **C** | B | F | X | 1 |
| **D** | A | F | 1 | X |
| **E** | F | F | 0 | 0 |
| **F** | A | F | X | 1 |

a) Minimize the number of states using the implication chart below:

14 pts



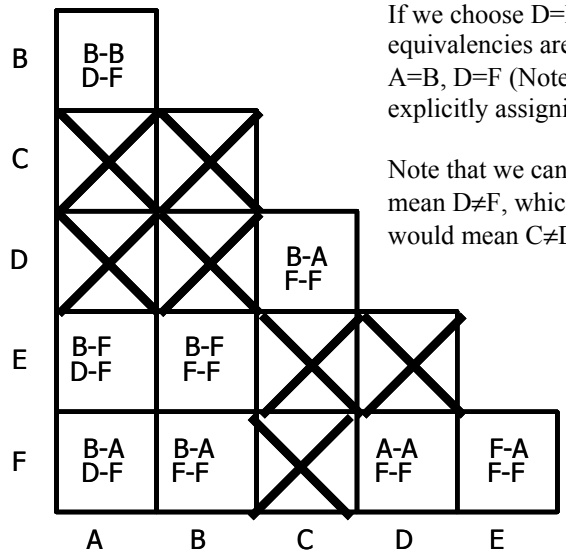

List the equivalencies: A=B, C=D=F

Name: Email: ID#

b) Now, suppose state "F" is changed such that the output is "0" when IN=1. Repeat your simplification under these new conditions.

| Present State | Next State | | Output | |
|---|---|---|---|---|
| | IN=0 | IN=1 | IN=0 | IN=1 |
| **A** | B | D | 0 | 0 |
| **B** | B | F | 0 | 0 |
| **C** | B | F | X | 1 |
| **D** | A | F | 1 | X |
| **E** | F | F | 0 | 0 |
| **F** | A | F | X | 0 |

16 pts




If we choose D=F, then C≠D based on outputs, so equivalencies are:
A=B, D=F (Note that B≠E≠F and A≠F since we are explicitly assigning the don't care in F to be 1)

Note that we cannot choose, C=D, since this would mean D≠F, which would mean A≠B, which in turn would mean C≠D, which is inconsistent.

List the equivalencies: A=B, D=F

c) Is your simplification in part B unique? Are there other possible equivalencies? If so, list them. If not, give reasons for your answer.

5 pts

Based on the reasoning above, the only choice of don't cares is D=F, so therefore, the solution is unique.