ECE 198 JL Second Midterm Exam  
Spring 2013 
Tuesday, March 5th, 2013 

Name: 
NetID: 

Discussion Section: 

<table>
<thead>
<tr>
<th>Time</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>10:00 AM</td>
<td>[ ] JD1</td>
</tr>
<tr>
<td>11:00 AM</td>
<td>[ ] JD2</td>
</tr>
<tr>
<td>2:00 PM</td>
<td>[ ] JD3</td>
</tr>
<tr>
<td>4:00 PM</td>
<td>[ ] JD4</td>
</tr>
</tbody>
</table>

- Be sure your exam booklet has 10 pages.
- Be sure to write your name and lab section on the first page.
- Do not tear the exam booklet apart; you can only detach the last page.
- We have provided Boolean properties at the back.
- Use backs of pages for scratch work if needed.
- This is a closed book exam. You may not use a calculator.
- You are allowed one handwritten 8.5 x 11” sheet of notes.
- Absolutely no interaction between students is allowed.
- Be sure to clearly indicate any assumptions that you make.
- The questions are not weighted equally. Budget your time accordingly.
- Don’t panic, and good luck!

Problem 1 11 points: 
Problem 2 14 points: 
Problem 3 15 points: 
Problem 4 10 points: 
Problem 5 22 points: 
Problem 6 18 points: 
Problem 7 10 points: 

Total 100 points:
Problem 1 (11 pts): Boolean algebra

1. Simplify expression \(y'(x'z + y'z')\). Write each step separately in the space provided. Name the property used for each step. First step is already written for you. (Refer to Boolean algebra properties on the last page of the exam booklet.)

\[
y'(x'z + y'z')' = \frac{y'(x'z)'(y'z)'}{DeMorgan}
\]

\[
= y'(x + z')(y + z')
= y'(xy + z'z)
= y'xy + y'z'
= xyz' + y'z'
= xy + y'z'
= 0 + y'z'
= y'z'
\]

Property

\(\text{DeMorgan}\)

\(\text{DeMorgan}\)

\(\text{Distributivity}\)

\(\text{Distributivity}\)

\(\text{Commutativity}\)

\(\text{Complementarity}\)

\(\text{Null}\)

\(\text{Identity}\)

2. Prove by perfect induction consensus property: \(xy + yz + x'z = xy + x'z\)

\[
\begin{array}{c|c|c|c}
 x & y & z & xy + yz + x'z \\
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{c|c|c}
 x & y & x'z' \\
0 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 1 \\
\end{array}
\]

Since the truth tables for both functions are the same, then they must be equivalent.

3. Write dual for \(x + y' z' x' + 0 \cdot x\). Do not simplify.

Answer: \(x(y' + z + x')(1 + x')\)

4. Let \(f(w, x, y, z) = m_9\). Find its dual and write it in \(M_1\) notation.

Answer: \(M_{10}\)

\[
M_q = (w x' + x' y z) \rightarrow \text{Dual} \quad w + x' + y + z = M_{10}
\]
Problem 2 (14 pts): Canonical forms

A committee has members A, B, and C. Variables a, b, c have value 1 iff A, B, C respectively vote in favor of a proposal. Design a combinational circuit whose output \( g \) is 1 iff there is a majority in favor.

1. Fill in truth table.

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>g</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

2. Using above table, write canonical SOP representation using literals.
   
   Answer: 
   \[ g = a'b'c + a'b'c' + a'b'c + a'b'c + a'b'c + a'b'c + a'b'c + a'b'c \]

3. Using above table, write canonical SOP representation using minterm notation \( m_4 \).
   
   Answer: 
   \[ g = m_8 + m_5 + m_6 + m_7 \]

   
   Answer: 
   \[ g = (a + b + c')(a + b + c')(a + b + c')(a + b + c') \]

5. Using above table, write canonical POS representation using maxterm notation \( M_4 \).
   
   Answer: 
   \[ g = M_0 M_1 M_2 M_4 \]

6. For function \( f (a, b, c, d) = a'b'c + a'c'd' \), write corresponding canonical SOP.
   
   Answer: 
   \[ f = a'b'c + a'c'd + a'b'c + a'b'c' \]

7. For function \( g (w, x, y, z) = (w + x') (w + x' + y + z') \), write corresponding canonical POS.
   
   Answer: 
   \[ g = (w + x + y + z') (w + x + y + z') (w + x + y + z') (w + x + y + z') \]
Problem 3 (15 pts): Function simplification

Consider a 4-variable Boolean function $f(w, x, y, z)$ given by its K-map (drawn twice):

1. List the essential prime implicants.
   Answer: $w'y$ and $x'yz'$

2. Give a minimal SOP expression for $f(w, x, y, z)$ and show the corresponding loops on the left map.
   Answer: $f = w'y + x'yzz' + w'zz' + wxx'$

3. Give a minimal POS expression for $f(w, x, y, z)$ and show the corresponding loops on the right map.
   Answer: $f = (x + y)'(w + y + z')'(w' + y + z')' (w' + x' + y')$  
   (The solution is not unique, other minimal POS exist)

4. Do your answers to Part 2 and Part 3 represent the same Boolean function? Justify your answer.
   No, because their values for the "don't cares" may differ. For example:
   Part 2: $f(0,0,0,0) = 1$  Part 3: $f(0,0,0,0) = 0$

5. Let $g(x, y, z) = x(y \oplus z)$. Fill in its K-map on the right and write minimal POS below. Show the corresponding loops.
   Answer: $g = x(yz)(y'z')$
Problem 4 (10 pts): 2-level circuits

1. Implement Boolean function \( g(a, b, c, d) = a c' d + a b' + c \) as a two-level network using AND and OR gates only. Assume that inverted inputs are available. Draw the circuit.

![Circuit Diagram 1](image1)

2. Re-implement the same function using NAND gates only. Do not use more than 6 NAND gates. Assume that inverted inputs are available. Draw the circuit.

![Circuit Diagram 2](image2)

3. Re-implement the same function using a 4:1 MUX and no more than one extra gate. Assume that inverted inputs are available. Draw the circuit. Show your work. *Hint: Draw K-map.*

![K-Map and MUX Diagram](image3)

- \( a = 0, c = 0 \Rightarrow g = 0 \)
- \( a = 0, c = 1 \Rightarrow g = 1 \)
- \( c = 1, c = 0 \Rightarrow g = b' + d \)
- \( a = 1, c = 1 \Rightarrow g = 1 \)
Problem 5 (22 pts): Combinational logic design

Part A. An n-bit arithmetic unit takes inputs $A = a_{n-1} \ldots a_0$ and $B = b_{n-1} \ldots b_0$, interpreted as the n-bit two’s-complement representations of numbers.

The control signals are $k_1, k_0$, and $c_0$ (the carry-in to stage 0). At each stage $i$, the inputs to the full adder are

$\begin{align*}
p_i &= a_0 k_i \oplus k_0 \oplus b_0 k_i \\
q_i &= a_0 k_i \oplus k_0 \oplus b_0 k_i + b_i k_1 + b_i \bar{k}_0
\end{align*}$

1. Determine the function of $A$ and $B$ produced by each of the following combinations of control signals:

$\begin{array}{ccc}
\text{k}_1 & \text{k}_0 & \text{c}_0 & \text{Function} \\
0 & 1 & 1 & A - B \\
1 & 0 & 0 & 2B
\end{array}$

Hint: write out truth tables for $p_i$ and $q_i$ as functions of $k_1$ and $k_0$.

2. Determine the values for the control signals to produce each of the following functions:

$\begin{array}{ccc}
\text{Function} & \text{k}_1 & \text{k}_0 & \text{c}_0 \\
\text{A plus 1} & 0 & 0 & 1 \\
\text{B minus 1} & 1 & 1 & 0
\end{array}$

3. Write $c_0$ as a function of $k_1$ and $k_0$.

Answer: $c_0 = k_1$
**Part B.** You are designing a Full Subtractor (FS) circuit. The FS has inputs \( x_i, y_i \), and a borrow input \( b_i \). There are two outputs: difference \( d_i \) and borrow-out \( b_{i+1} \).

![Diagram of Full Subtractor](image)

The FS cell should be designed so that the n-bit subtractor network shown above correctly computes \( D = X - Y \), where \( X = x_{n-1} \ldots x_1 x_0 \) and \( Y = y_{n-1} \ldots y_1 y_0 \) are nonnegative n-bit binary numbers. Assume \( X \geq Y \).

**Examples for n=3:**

\[
\begin{align*}
X &= x_2 x_1 x_0 & 101 & 100 \\
-Y &= y_2 y_1 y_0 & 011 & 001 \\
D &= d_2 d_1 d_0 & 010 & 011
\end{align*}
\]

1. Draw K-maps for \( d_i \) and for \( b_{i+1} \). **Hint:** Remember that \( b_{i+1} \) and \( d_i \) are functions of only the 3 inputs: \( x_i, y_i, b_i \). Try some examples, and start with the rightmost FS cell.

\[
\begin{array}{c|c|c|c|c|c}
 & x_i y_i & x_i y_i & x_i y_i & x_i y_i & x_i y_i \\
\hline
b_i & 0 & 0 & 1 & 1 & 0 \\
\hline
& 0 & 0 & 1 & 1 & 0 \\
\hline
\end{array}
\]

\[
\begin{array}{c|c|c|c|c|c}
 & x_i y_i & x_i y_i & x_i y_i & x_i y_i & x_i y_i \\
\hline
b_i & 0 & 0 & 1 & 1 & 0 \\
\hline
& 0 & 0 & 1 & 1 & 0 \\
\hline
\end{array}
\]

\[
d_i = \left( b_i + X_i + Y_i \right) \left( b_i + X_i + !Y_i \right) \left( b_i + X_i + Y_i \right) \left( b_i + X_i + !Y_i \right) \quad \text{(POS)}
\]

2. Give minimal POS expressions for \( d_i \) and for \( b_{i+1} \).

\[
d_i = b_i \cdot X_i \cdot Y_i + b_i \cdot X_i \cdot !Y_i + b_i \cdot X_i \cdot Y_i + b_i \cdot X_i \cdot !Y_i \quad \text{(SOP)}
\]

\[
b_{i+1} = \left( b_i + Y_i \right) \left( b_i + X_i \right) \left( X_i + Y_i \right) \quad \text{(SOP)}
\]

3. Implement circuit for computing \( d_i \) value using XOR gate(s) only.

\[
d_i = b_i \cdot (X_i \oplus Y_i) + b_i \cdot (X_i \oplus !Y_i)
\]

\[
= b_i \cdot (X_i \oplus Y_i) + b_i \cdot (\overline{X_i} \oplus Y_i)
\]

\[
= b_i \cdot (X_i \oplus Y_i)
\]

![Diagram of XOR gate](image)
Problem 6 (18 pts): Sequential logic components

Part A. Shown below is the logic diagram of a gated D latch. It consists of 4 NAND gates and an inverter. It has 2 inputs: D and C.

![D latch logic diagram](image)

1. Complete the next-state table for this latch circuit.

<table>
<thead>
<tr>
<th>C</th>
<th>D</th>
<th>Q+</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Q</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Q</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

2. Express next state Q+ as a function of C, D, and Q.

Answer: \( Q^+ = Q \cdot C' + D \cdot C \)

Part B. Complete the design of a 3-bit register that performs the operations listed in the table to the right. Parallel load inputs are labeled and indexed as \( P_i \). Serial input is labeled as \( S_{in} \). You may use inputs without drawing the wires, just write the appropriate labels at the MAX inputs.

<table>
<thead>
<tr>
<th>( F_1 )</th>
<th>( F_0 )</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Parallel load</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Circular shift right</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Logical shift left</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>No change</td>
</tr>
</tbody>
</table>

![3-bit register diagram](image)
Problem 7 (10 pts): Program analysis

Consider the following C program:

```c
#include <stdio.h>

int main()
{
    unsigned int a,b,c;
    int function;
    int notfirst=0;

    for ( a = 0; a <= 1; a = a + 1 )
    {
        for ( b = 0; b <= 1; b = b + 1 )
        {
            for ( c = 0; c <= 1; c = c + 1 )
            {
                function = a & (b | ~c);
                if (function)
                {
                    if (notfirst) printf("+");
                    if (a) printf("a"); else printf("a'");
                    if (b) printf("b"); else printf("b'");
                    if (c) printf("c"); else printf("c'");
                    notfirst = 1;
                }
            }
        }
    }

    printf("\n");
    return 0;
}
```

1. Write down EXACTLY the formatted text that will be printed on the terminal screen by the program.

   \[ a'bc' + abc' + abc \]

2. Explain in one sentence the function of the program; that is, what does it print?

   It prints the canonical SOP of the boolean function \( f = a(b + c') \)
### Boolean algebra properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Equation 1</th>
<th>Equation 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commutativity</td>
<td>$x \cdot y = y \cdot x$</td>
<td>$x + y = y + x$</td>
</tr>
<tr>
<td>Associativity</td>
<td>$(x \cdot y) \cdot z = x \cdot (y \cdot z)$</td>
<td>$(x + y) + z = x + (y + z)$</td>
</tr>
<tr>
<td>Distributivity</td>
<td>$x \cdot (y + z) = x \cdot y + x \cdot z$</td>
<td>$x + y \cdot z = (x + y) \cdot (x + z)$</td>
</tr>
<tr>
<td>Idempotence</td>
<td>$x \cdot x = x$</td>
<td>$x + x = x$</td>
</tr>
<tr>
<td>Identity</td>
<td>$x \cdot 1 = x$</td>
<td>$x + 0 = x$</td>
</tr>
<tr>
<td>Null</td>
<td>$x \cdot 0 = 0$</td>
<td>$x + 1 = 1$</td>
</tr>
<tr>
<td>Complementarity</td>
<td>$x \cdot x' = 0$</td>
<td>$x + x' = 1$</td>
</tr>
<tr>
<td>Involution</td>
<td>$(x')' = x$</td>
<td></td>
</tr>
<tr>
<td>DeMorgan’s</td>
<td>$(x \cdot y)' = x' + y'$</td>
<td>$(x + y)' = x' \cdot y'$</td>
</tr>
<tr>
<td>Absorption</td>
<td>$x \cdot (x + y) = x$</td>
<td>$x + x \cdot y = x$</td>
</tr>
<tr>
<td>No-Name</td>
<td>$x \cdot (x' + y) = x \cdot y$</td>
<td>$x + x' \cdot y = x + y$</td>
</tr>
<tr>
<td>Consensus</td>
<td>$(x+y) \cdot (y+z) \cdot (x'+z) = (x+y) \cdot (x'+z) = x \cdot y + y \cdot z + x' \cdot z = x \cdot y + x' \cdot z$</td>
<td></td>
</tr>
</tbody>
</table>