Only Problem Set Part B will be graded. Turn in only Problem Set Part B which will be due on June 8, 2012 (Friday) at 3:00pm.

1 Problem Set Part A

- textbook 6.7
- textbook 6.8
- textbook 6.10
- textbook 6.11
- textbook 6.12
- textbook 6.14
- textbook 6.15
- textbook 6.17
- textbook 6.18
- textbook 6.19
- textbook 6.22
- textbook 5.2
- textbook 5.3
- textbook 5.7(a)
- textbook 5.10(a)(b)
- textbook 5.13
- textbook 5.16
- textbook 5.17(c)
- textbook 5.21
- textbook 5.22(a)(b)
2 Problem Set Part B

1 (FSM Analysis)

We have studied in class the state diagram of a modulo-4 counter, as shown in the following figure. The circuit takes a single bit input $C$, and produces a single bit output $Y$.

In the following three parts of this question, we show three distinct implementations of this modulo-4 counter, two implementations with T flip-flops, and one with D flip-flops. Please identify the encoding mechanism that has been used in each implementation. (Hint: while you can of course solve this problem by working through methodically the various stages of FSM analysis, perhaps thinking about the logic relationships in the next state and output logic may enable you to solve this problem a lot more quickly.)

(Part A) An implementation using two T flip flops.
(Part B) An implementation using two D flip flops and two selectors.

(Part C) An implementation using four T flip flops. The 4-bit state $Q_3Q_2Q_1Q_0$ is initialized to 0110 using asynchronous set/reset signals.
2 (State Encoding Strategies)

A CAD company is considering the development of a new state encoding tool for D flip-flops that can maximize the probability of generating optimal state encodings for FSMs. A group of engineers is exploiting the various conditions of assigning adjacent encodings (code words with Hamming distance 1) to all states that have a common destination or source, more specifically, the two cases shown in the following figure.

![Diagram showing two potential candidates for adjacent encodings](https://example.com/diagram.png)

**Figure 1: Two potential candidates for adjacent encodings**

The engineers do remember a rather eccentric professor from two dozen years ago discussing some similar looking diagrams, but time has taken its toll, and they are no longer quite sure under what input conditions these diagrams should be applied. As ignorance could be a great breeding ground for creativity (you do have a lot more options after all), the engineers are passionately arguing on the types of input conditions (i.e., the values of \(i\) and \(j\)) under which adjacent encodings should be assigned to \(S_0\) and \(S_1\). There are four prevailing ideas for each of the two cases:

- **Idea W**: adjacent encodings should **only** be assigned to \(S_0\) and \(S_1\) if \(i\) and \(j\) are the **same** input combination.
- **Idea X**: adjacent encodings should **only** be assigned to \(S_0\) and \(S_1\) if \(i\) and \(j\) are **adjacent** input conditions of Hamming distance 1.
- **Idea Y**: adjacent encodings should **only** be assigned to \(S_0\) and \(S_1\) if \(i\) is a **superset** of \(j\) (or vice versa), for example, \(i = 1X\) and \(j = 10\).
- **Idea Z**: adjacent encodings should **always** be assigned to \(S_0\) and \(S_1\) **independent** of the values of \(i\) and \(j\).

You are in a rather unfortunate position as the manager has tasked you with helping her make a decision as to which of these idea(s) are right. Unfortunately, as you remember from class, optimality is a probabilistic issue, implying that specific decisions may turn out best for a particular FSM but may fail on the average. The probabilistic aspect deprives you of the ability to identify the correct answer by looking at a particular example. Therefore, you need to evaluate, based on your conceptual understanding of the material, for each of the four ideas whether:

**A** The engineers involved in promoting this idea should be **fired** because the input condition suggested by them will **never happen** in any FSM.

**B** The engineers involved in promoting this idea should be **demoted** because their suggested input condition **cannot** generate the best encoding result probabilistically speaking for most FSMs.

**C** The engineer involved in promoting this idea should be **promoted** because their suggested input condition on the average does end up generating the best encoding result for most FSMs.
(Part A) For Case A (two states with a common destination) shown in the left of Figure 1, please clearly present your evaluation for each engineer and also give a brief reasoning (otherwise the manager will not believe in your judgment!).

Your evaluation for Idea W:
Your reasoning:

Your evaluation for Idea X:
Your reasoning:

Your evaluation for Idea Y:
Your reasoning:

Your evaluation for Idea Z:
Your reasoning:

(Part B) For Case B (two states with a common source) shown in the right of Figure 1, please clearly present your evaluation for each engineer and also give a brief reasoning to convince the manager.

Your evaluation for Idea W:
Your reasoning:

Your evaluation for Idea X:
Your reasoning:

Your evaluation for Idea Y:
Your reasoning:

Your evaluation for Idea Z:
Your reasoning:
(Part C) After figuring out the input conditions for each of the two adjacent encoding cases, the engineers in the company now are wondering about the application priority of the two cases. More specifically, if the two cases imply conflicting encoding guidelines, one would need to adjudicate among the competing demands of the two heuristics.

Please help these engineers order the two cases by analyzing the maximum amount of benefit that each case can deliver if adjacent encodings are first assigned to the two source states in Case A or alternatively first to the two destination states in Case B.
3 (PLA Implementations of Magnitude Comparators)

An \( n \)-bit magnitude comparator can be implemented by connecting \( n-1 \) 2-bit magnitude comparators in a serial manner, as shown in the figure below. Recall that a 2-bit magnitude comparator has two 2-bit inputs \((a_1a_0, b_1b_0)\) and two outputs \((G, L)\). \( G \) equals 1 iff \( a_1a_0 > b_1b_0 \), while \( L \) equals 1 iff \( a_1a_0 < b_1b_0 \).

In this question, you are asked to implement 2-bit magnitude comparators using Programmable Logic Arrays (PLAs). Recall that the efficiency of a PLA implementation is usually determined by the number of rows (used to represent subcubes of inputs) that need to be used. To improve the efficiency of an implementation, two optimization techniques can be applied, listed as follows:

- Because each PLA has an XOR array that enables us to conditionally complement the output values, one can program the AND and OR arrays to implement the complement of a function, if the function has only a few 0’s, particularly if they are contiguously positioned.

- If multiple functions are to be implemented simultaneously, one can exploit the possibility of sharing common row(s) among several functions to reduce the total number of words that need to be used.
(Part A) In this part, you are asked to implement the $G$ and $L$ functions using an AND-OR PLA in the **most economical** way, i.e., using the minimal number of rows.

Please first fill in the Karnaugh maps below for the functions $G$ and $L$, as defined in the previous page.

Karnaugh map for $G$:  

<table>
<thead>
<tr>
<th>$a_1b_1 \backslash a_0b_0$</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$G =$

Karnaugh map for $L$:

<table>
<thead>
<tr>
<th>$a_1b_1 \backslash a_0b_0$</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$L =$

Subsequently, please implement the $G$ and $L$ functions by denoting the variables and output and marking the connections in the figure provided below. Note that you might not need to use all the wires in the figure.
(Part B) For the 8-bit **serial** magnitude comparator, it turns out that except for the rightmost 2-bit comparator, the other six comparators can be further optimized.

Please first briefly tell us the insight that you think enables this optimization, and then fill in the two maps below for the functions $G$ and $L$.

The insight that enables this optimization is:

Karnaugh map for $G$:

<table>
<thead>
<tr>
<th>$a_1b_1\bar{a}_1\bar{b}_0$</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$G =$

Karnaugh map for $L$:

<table>
<thead>
<tr>
<th>$a_1b_1\bar{a}_1\bar{b}_0$</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$L =$

Subsequently, please implement the $G$ and $L$ functions using an AND-OR PLA in the **most economical** way, by denoting the variables and output and marking the connections in the figure provided below. Note that you might not need to use all the wires in the figure.