CSE 140 Homework Two

July 12, 2012

*Only Problem Set Part B will be graded. Turn in only Problem Set Part B which will be due on July 23, 2012 (Monday) at 4:00pm.*

1 Problem Set Part A

- textbook 3.7
- textbook 3.10(b)(c)
- textbook 3.13
- textbook 3.14
- textbook 3.15(a)(b)
- textbook 3.16(a)
- textbook 3.19(a)
- textbook 4.3
- textbook 4.5
- textbook 4.6
- textbook 4.7(b)(d)
- textbook 4.9(b)(c)
- textbook 4.11(a)(c)
- textbook 4.17(b)
- textbook 4.19
2 Problem Set Part B

1 (Logic Functions)

All possible 16 functions of 2 Boolean variables are provided in the table shown below. These Boolean functions may exhibit certain properties. We would like you to identify whether the properties listed as (a) to (d) at the bottom of the page are applicable for each of these 16 functions. If you think they do apply, please put a check mark in the boxes at the left hand side of the table.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Name</th>
<th>Function values for x, y =</th>
<th>Algebraic Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>ZERO</td>
<td>0 0 0 0</td>
<td>$F_0 = 0$</td>
</tr>
<tr>
<td>(b)</td>
<td>AND</td>
<td>0 0 0 1</td>
<td>$F_1 = xy$</td>
</tr>
<tr>
<td>(c)</td>
<td>Inhibition</td>
<td>0 0 1 0</td>
<td>$F_2 = xy'$</td>
</tr>
<tr>
<td>(d)</td>
<td>Transfer</td>
<td>0 0 1 1</td>
<td>$F_3 = x$</td>
</tr>
<tr>
<td></td>
<td>Inhibition</td>
<td>0 1 0 0</td>
<td>$F_4 = x'y$</td>
</tr>
<tr>
<td></td>
<td>Transfer</td>
<td>0 1 0 1</td>
<td>$F_5 = y$</td>
</tr>
<tr>
<td></td>
<td>XOR</td>
<td>0 1 1 0</td>
<td>$F_6 = x'y + x'y$</td>
</tr>
<tr>
<td></td>
<td>OR</td>
<td>0 1 1 1</td>
<td>$F_7 = x + y$</td>
</tr>
<tr>
<td></td>
<td>NOR</td>
<td>1 0 0 0</td>
<td>$F_8 = (x + y)'$</td>
</tr>
<tr>
<td></td>
<td>Equivalence</td>
<td>1 0 0 1</td>
<td>$F_9 = xy + x'y'$</td>
</tr>
<tr>
<td></td>
<td>Complement</td>
<td>1 0 1 0</td>
<td>$F_{10} = y'$</td>
</tr>
<tr>
<td></td>
<td>Implication</td>
<td>1 0 1 1</td>
<td>$F_{11} = x + y'$</td>
</tr>
<tr>
<td></td>
<td>Complement</td>
<td>1 1 0 0</td>
<td>$F_{12} = x'$</td>
</tr>
<tr>
<td></td>
<td>Implication</td>
<td>1 1 0 1</td>
<td>$F_{13} = x' + y$</td>
</tr>
<tr>
<td></td>
<td>NAND</td>
<td>1 1 1 0</td>
<td>$F_{14} = (xy)'$</td>
</tr>
<tr>
<td></td>
<td>ONE</td>
<td>1 1 1 1</td>
<td>$F_{15} = 1$</td>
</tr>
</tbody>
</table>

(a): $F(x, y) = F(y, x)$

(b): $F(x', y') = F(x, y)$

(c): $F(x', y') = F'(x, y)$

(d): $F(x, F(y, z)) = F(F(x, y), z)$
2 (Tabulation Method)

When applying the Tabulation method to an \textit{n-variable} Boolean function, we noticed that some groups (i.e., \( G_i \)) are \textbf{fully specified} (i.e., all possible elements that can exist in these groups are included). This question concerns the size of these fully specified groups and the possible groups that can be generated on the subsequent list of subcubes by merging the elements of these groups. Please keep in mind that an \textit{n-subcube} is composed of \( 2^n \) minterms (or maxterms) or, alternatively, you can recognize them as having \( n \) dashes (’-’).

(Part A) Let’s start with a 4-variable Boolean function. In the Tabulation method application, we noticed that the \( G_1 \) and \( G_2 \) are \textit{fully} specified in the list of 1-subcubes. What is the number of elements in these groups, \( G_1 \) and \( G_2 \), in 1-subcubes? Please provide a brief reasoning.

\( G_1 \) in 1-subcubes:

\( G_2 \) in 1-subcubes:

(Part B) Given that we know \( G_1 \) and \( G_2 \) in the list of 1-subcubes are \textit{fully} specified for the 4-variable Boolean function in \textit{Part A}, is it possible to determine the number of elements in \( G_1 \) and \( G_2 \) groups of 2-subcubes? Notice that we only know \( G_1 \) and \( G_2 \) in 1-subcubes are fully specified and have no knowledge whatsoever of other groups in 1-subcubes. If it is possible, please provide the number of elements in these groups. If it is not, please provide a brief reasoning.

\( G_1 \) in 2-subcubes:

\( G_2 \) in 2-subcubes:
(Part C) We would like to generalize our observations in Part A and Part B to any $n$-variable Boolean function. Given that $G_i$, $G_{i+1}$ and $G_{i+2}$ are fully specified in the list of $m$-subcubes of an $n$-variable function ($n >> m$), what are the number of elements in $G_i$, $G_{i+1}$ and $G_{i+2}$ of m-subcubes? Please provide the answer as a function of $m$, $n$ and $i$.

$G_i$ in m-subcubes:

$G_{i+1}$ in m-subcubes:

$G_{i+2}$ in m-subcubes:

(Part D) Given that $G_i$, $G_{i+1}$ and $G_{i+2}$ are fully specified in the list of $m$-subcubes of an $n$-variable function ($n >> m$) in Part C and that we have no knowledge of any other group, is it possible to determine the number of elements in $G_i$, $G_{i+1}$ and $G_{i+2}$ of (m+1)-subcubes? If it is possible, please provide the number of elements in the group as a function of $m$, $n$ and $i$. If it is not, please provide a brief reasoning.

$G_i$ in (m+1)-subcubes:

$G_{i+1}$ in (m+1)-subcubes:

$G_{i+2}$ in (m+1)-subcubes:
A startup company, founded by your fellow UCSD CSE students who had taken CSE 140 a few years ago, has announced a major breakthrough and, in a clear break with the perceived orthodoxy of the CMOS world, has started advertising a design library in which the heretofore much maligned XOR and XNOR gates have had a transistor implementation which has left in the dust not only standard gates such as AND and OR, but also the much hyped up NAND and NOR gates. Based on this much-touted implementation, they were able to extend their multi-input design libraries to multi-input XOR and XNOR gates as well, a feat heretofore unmatched.

Once the announcement had been made in the technical press, sneers and envy have greeted the announcement in equal measure. Of course, the competition’s envy is all too easily understood, but the sneers seem to have focused on the point that this startup company must be full of nincompoops since a 3-input XOR gate is well known to be equivalent to a 3-input XNOR gate. If they are advertising both gates, either they are trying to extract extra mileage from a simple gate (not of course unheard of in the world of advertising) or they must be spending time implementing the same gate without realizing it (possibly not unheard of in the technical world).

While the greater world is trying to decide whether to indulge in a smear campaign about the incompetence of the startup or to start a research program of their own to catch up with them (or to do both), you, while you are still taking CSE 140, have been indulging in secret negotiations with the company founded by your predecessors. As such, you know that while two 2-input XOR gates or two 2-input XNOR gates are equivalent and represent the functionality of a 3-input XOR gate, one 2-input XOR connected to one 2-input XNOR (or vice-versa) is the functionality delivered by a 3-input XNOR gate.

As a matter of fact, the company has plans to deliver even more input XOR and XNOR gates, which they define as being equivalent to decomposed gates embedding an even number or an odd number of 2-input XNORs, respectively. Of course, the reason why the startup has been involved in discussions with you is not for your clarifications regarding the multi-input XOR or XNOR gates, but rather because of your digital logic and software background, something that they have overlooked as they have delved increasingly into the dark arts of transistor-level implementations. Yet, now that they have made this breakthrough, they realize that the only path to monetize their discovery is through making progress on delivering software tools, such as logic synthesis tools that can exploit their much coveted XOR and XNOR gates.

Refreshing their memory by looking at old textbooks has reminded them that they sorely need to expand this logic synthesis theory to XORs and XNORs since this whole theory as it stands seems to be essentially oriented to discovering minimal sum-of-products or product-of-sums representations. Having heard of your superior intelligence and knowing that your understanding of these concepts is still fresh in your mind, they have turned to you with a number of questions that have been perplexing them for a while, hoping that you will be able to provide insightful directions to their quest.

The questions follow.
(Part A) Are these 3-input XOR gates that you previously discussed (and detailed in the previous paragraphs) associative? Please provide a counterexample if you think they are not; please provide a brief reasoning if you think they are.

(Part B) Are these 3-input XNOR gates that you previously discussed (and detailed in the previous paragraphs) associative? Please provide a counterexample if you think they are not; please provide a brief reasoning if you think they are.
(Part C) In their early dabblings with the synthesis ideas, this company was able to observe that the quest for XNOR or XOR expressions in the Karnaugh maps seemed not to be correlated with adjacency but rather with diagonal relationships. For example, in a Karnaugh map as given, the founders seem to have noticed that these two minterms are providing an XOR relationship:

\[
\begin{array}{c|cccc}
wx/yz & 00 & 01 & 11 & 10 \\
00 & 1 & & & \\
01 & 1 & & & \\
11 & & & & \\
10 & & & & \\
\end{array}
\]

Which can be written as \( w'y'z' + w'x'y'z = w'y'(x \oplus z) \).

What is perplexing them though is how to extend this observation to a number of bigger XOR gates, such as a 3-input XOR gate. Could you enlighten them as to which additional minterm patterns would need to be added to this K-map to generate a 3-input XOR? Please mark them on the K-map above.

(Part D) While you are at it, they are also of course looking for a breakthrough in logic synthesis to enable them to utilize their XNOR gates. What does a 2-input XNOR gate (of course multiplied by a 2-input product term as in the XOR example) look like? How does one extend it to a 3-input XNOR gate? Fill in both in the K-map showns below, clearly differentiating between the 2-input and 3-input XNOR functions.
(Part E) Now that you have guided them to these logic synthesis breakthroughs (and accumulated unimaginable quantities of stock options, which can be cashed in for a few measly points on your midterm), it should be a cinch for you to start enjoying the fruits of the theory you developed. Please take a look at this Karnaugh map and provide an implementation that uses two 2-input AND gates, one 2-input OR gate, and 2 XOR or XNOR gates of either 2 or 3 inputs.

<table>
<thead>
<tr>
<th>wx/yz</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

(Part F) While you are working on this exciting startup, the founders at the startup are pondering not only the issue of the software packages that they will be delivering, but the packaging of their hardware product as well. As you remember from the discussion in class, for medium volumes, gate-arrays offer advantages of manufacturing cost, and the company is evaluating whether to deliver gatearrays based on their technology. As it so often happens, the 4 founders of the company are involved in a long and protracted battle, with one arguing that they should sell gatearrays of XOR gates only, the other arguing for gatearrays of XNOR gates only, with the third in a mediating role trying to argue for gatearrays of XORs and XNORs in equal measure, and the fourth and most pessimistic founder arguing that it is all hopeless and that they should give up on gatearrays and try to move to higher volume market points, dominated by standard cells. Which of the 4 is right and why?
(Part G) While this high strategic battle is brewing among the founders, the gatearray group needs to tackle one more question. As they start appreciating the impact of your logic synthesis results, they realize that eventually the logic synthesis software would end up producing multi-input XOR and XNOR gates over and above the measly looking by now 2-input or 3-input X(N)OR gates. One question is how these multi-input X(N)OR representations delivered by their logic synthesis tool can be decomposed into the largest gates that they have, namely the 3-input X(N)OR gates. Please provide an algorithm and show the implementation in terms of a decomposition of a 9-input XNOR gate and a 10-input XOR gate into the minimal number of 2-input or 3-input X(N)OR gates in their current library.