Solution for CSE140 HW2

May 3, 2017

Due 11:59PM, Mon 4/24/2017.

Homework 2 covers the Boolean algebra, combinational logic specification, and implementation. The first problem reviews the definition of Boolean algebra using a multiple-element system to distinguish from a switching function system. For the next two problems, we practice more on the specification, in particular, when the number of input bits is not small. For the last two problems, we practice design minimization using Karnaugh maps in sum of products and product of sums formats.

1 Boolean Algebra

Given a mathematical system M=(a,b,c,d,#, &) where the two operators # and & are defined in the following two subtables.

<table>
<thead>
<tr>
<th>(a)</th>
<th>(b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>c</td>
<td>a</td>
</tr>
<tr>
<td>d</td>
<td>a</td>
</tr>
</tbody>
</table>

Table 1: 2-input operators # and &.

1. Verify whether the system is a Boolean algebra.

2 points for an attempt of 1.1.

1.1 Solution for 1.1

To verify whether the system is a boolean algebra, is to see if the system satisfies the following four of laws.
1) Commutative laws
2) Distribution laws
3) Identify laws
4) Complement laws

Test with the four laws
Assume that x,y,z are elements are ∈ M
1) Commutative laws
if is "x # y = y # x" satisfied? Satisfied by observing from the table;
if is "x & y = y & x" satisfied? Satisfied by observing from the table;
2) Distribution laws
From the table we know that,
\[ a \# x = a; d \# x = x; \]
\[ a \& x = x; d \& x = d; \]

i). Operator \# must be distributive over operator \&
if \( x \# (y \& z) \) equals \( (x \# y) \& (x \# z) \) satisfied?

Truth table
<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
<th>x # (y &amp; z)</th>
<th>(x#y) &amp; (x#z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>a</td>
<td>y</td>
<td>z</td>
<td>a</td>
<td>a</td>
</tr>
</tbody>
</table>

x y z x \# (y \& z) x \# z
b b b b b
b d c c c
b b d b b
b c d b b
b d c c c
b c d c c
b b c c c
b c b c a
b c b c a
b b b c a
b c b c a
b c c d d
b c c d d
b b b d d
b c b c c
b c c d d
b b c c c
b c b c a
b c b c a
b c c d d
b b b d d

ii). Operator \& must be distributive over operator \#
if \( x \& (y \# z) \) equals \( (x \& y) \# (x \& z) \) exits?

Truth table
<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
<th>x &amp; (y # z)</th>
<th>(x&amp;y) # (x&amp;z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>d</td>
<td>y</td>
<td>z</td>
<td>d</td>
<td>d</td>
</tr>
</tbody>
</table>

x y z x \& (y \# z) x \& z
b d z x \& z x \& z
b d c x \& z x \& z
b c a x \& z x \& z
b c b x \& z x \& z
b a c x \& z x \& z
b c a x \& z x \& z
b c b x \& z x \& z
b a c x \& z x \& z
b c b x \& z x \& z
b a c x \& z x \& z
b c b x \& z x \& z
b c b x \& z x \& z
b a c x \& z x \& z
b c b x \& z x \& z
b a c x \& z x \& z
b c b x \& z x \& z
b c b x \& z x \& z
b a c x \& z x \& z
b c b x \& z x \& z
b a c x \& z x \& z
b c b x \& z x \& z
b a c x \& z x \& z
b c b x \& z x \& z
b a c x \& z x \& z
b c b x \& z x \& z
b a c x \& z x \& z
b c b x \& z x \& z
b a c x \& z x \& z
b c b x \& z x \& z
b a c x \& z x \& z
b c b x \& z x \& z
b a c x \& z x \& z
b c b x \& z x \& z
b a c x \& z x \& z
b c b x \& z x \& z

if \( b \&(c \# d) \) = \( b \& c \# (b \& d) \)

if \( b \&(c \# d) \) = \( b \& c \# (b \& d) \)

3) Identify laws
\[ x + 0 = x \]
x * 1 = x For any element x ∈ M, there is 

x # d = x
x & a = x

Satisfied by observing from the table.

4) Complement laws
Assume that d is an identify element with respect to the operators.
For each element x:
d#x = x#d = x
a is an identify element with respect to &.
For each element x:
a&x = x&a = x
Assume i and z are identify elements with respect to operator # and & each. For any element x ∈ M satisfied that,
x # x′ = a
x & x′ = d
from the table, there are two such pairs, (a,d), (b,c) satisfying the above condition: a # d = a, a & d = d;
b # c = a, b & c = d;

2. List the complements of elements a, b, c, and d if the system is a Boolean algebra.

1.2 Solution for 1.2

<table>
<thead>
<tr>
<th>element x</th>
<th>complement x′</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>d</td>
</tr>
<tr>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>c</td>
<td>b</td>
</tr>
<tr>
<td>d</td>
<td>a</td>
</tr>
</tbody>
</table>

2 Bit Counting Machine

A bit counting machine reads a binary input vector (a,b,c,d,e) and produces a binary number (s2,s1,s0) that counts the number of 1s in the input bits. For example when (a,b,c,d,e) = (0,1,1,1,1), we have output (s2,s1,s0) = (1,0,0), and when (a,b,c,d,e) = (1,1,0,1,0), we have output (s2,s1,s0) = (0,1,1).

1. Derive the canonical sum of min-terms expression of the three output bits.

2 points for an attempt of 2.1
2.1 Solution for 2.1

Truth table

<table>
<thead>
<tr>
<th>id</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>s₂</th>
<th>s₁</th>
<th>s₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>26</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>27</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>29</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>31</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[s₀(a,b,c,d,e) = \sum m \left(1,2,4,7,8,11,13,14,16,19,21,22,25,26,28,31\right)\]

\[s₁(a,b,c,d,e) = \sum m \left(3,5,6,7,9,10,11,12,13,14,17,18,19,20,21,22,24,25,26,28\right)\]

\[s₂(a,b,c,d,e) = \sum m \left(15,23,27,29,30,31\right)\]

2. Fill in the following Bluespec code that implements the bit counting machine. (Notice that \(i++\) and \(++i\) are not supported in Bluespec, please use \(i = i + 1\) for incrementing \(i\))

```plaintext
function Bit#(3) bitCountMachine(Bit#(5) in);

Integer count = 0;
for (Integer i = 0; i < 5; i = i + 1) begin
    if (in[i] == ________) begin
    ______________________;
    end
end
return fromInteger(count);
endfunction
```

Deduct 1 points for incorrect answer for each blank.
2.2 Solution for 2.2

```plaintext
function Bit#(3) bitCountMachine(Bit#(5) in);
    Integer count = 0;
    for ( Integer i = 0; i < 5; i = i + 1) begin
        if (in[i] == 1) begin
            count = count + 1;
        end
    end
    return fromInteger(count);
endfunction
```

3 Priority Encoder

A priority encoder reads a vector of six binary bits (a5, a4, a3, a2, a1, a0) and produces a binary number (d2, d1, d0) that presents the lowest index of the input bit which is true. For example, when (a5, a4, a3, a2, a1, a0) = (0, 1, 0, 1, 0, 0), the output (d2, d1, d0) = (0, 1, 0); (a5, a4, a3, a2, a1, a0) = (1, 0, 0, 0, 0, 0), the output (d2, d1, d0) = (1, 0, 1); and when (a5, a4, a3, a2, a1, a0) = (0, 0, 0, 0, 0, 1), the output (d2, d1, d0) = (0, 0, 0). However, when none of the input bits asserts a true value, i.e. (a5, a4, a3, a2, a1, a0) = (0, 0, 0, 0, 0, 0), the output remains to be (d2, d1, d0) = (0, 0, 0).

1. Derive the sum of products expression of the three output bits.

3.1 Solution for 3.1

Truth table

<table>
<thead>
<tr>
<th>id.</th>
<th>a5</th>
<th>a4</th>
<th>a3</th>
<th>a2</th>
<th>a1</th>
<th>a0</th>
<th>d2</th>
<th>d1</th>
<th>d0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ d_0 = a_1a_0' + a_3a_2a_1a_0' + a_3a_4a_3a_2a_1a_0' \]
\[ d_1 = a_2a_1a_0' + a_3a_2a_1a_0' \]
\[ d_2 = a_4a_3a_2a_1a_0' + a_4a_3a_3a_2a_1a_0' \]

2. Fill in the following Bluespec code that implements the priority encoder.

```plaintext
function Bit#(3) priorityEncoder(Bit#(6) in);
    Integer lowestIndex = 0;
    for ( Integer i = 5; i >= 0; i = i - 1 ) begin
        if ( in[i] == __________ ) begin
            ______________________;
        end
    end
    Bit#(3) ret = fromInteger(lowestIndex);
    return ret;
endfunction
```

3.2 Solution for 3.2

```plaintext
function Bit#(3) priorityEncoder(Bit#(6) in);
    Integer lowestIndex = 0;
```
for ( Integer i = 5; i >= 0; i = i - 1 ) begin
    if ( in[i] == 1 ) begin
        lowestIndex = i;
    end
end
Bit#(3) ret = fromInteger(lowestIndex);
return ret;
endfunction

4 Minimal Sum of Products Expression

Implementation from truth table to sum of products expressions.
1. Use Karnaugh map to simplify function
   \[ f(a, b, c) = \sum m(1, 3, 4, 6) + \sum d(0). \]
   List all possible minimal two-level sum of products expressions. Show the switching functions. No need for the schematic diagram.
   2 points for an attempt of 4.1.

4.1 Solution for 4.1
\[ f(a,b,c) = a'c + ac' \]

2. Use Karnaugh map to simplify function
   \[ f(a, b, c, d) = \sum m(4, 5, 6, 9, 11, 15) + \sum d(0, 1, 2, 8, 10, 14). \]
   List all possible minimal two-level sum of products expressions. Show the switching functions. No need for the schematic diagram.
   deduct 0.5 point for each missing or incorrect equation.

4.2 Solution for 4.2
K-map
\[
\begin{array}{cccc}
  cd/ab & ab = 00 & 01 & 11 & 10 \\
  cd = 00 & X & 1 & 0 & X \\
  cd = 01 & X & 1 & 0 & 1 \\
  cd = 11 & 0 & 0 & 1 & 1 \\
  cd = 10 & X & 1 & X & X \\
\end{array}
\]

\[ f(a,b,c,d) = a'c' + ac + cd' + ab \]
5 Minimal Product of Sums Expression

Implementation from truth table to product of sums expressions.
1. Use Karnaugh map to simplify function
   \[ f(a, b, c) = \sum m(0, 3, 6) + \sum d(2, 5). \]
   List all possible minimal two-level product of sums expressions. Show the switching functions. No need for the schematic diagram.

   2 points for an attempt of 5.1.

5.1 Solution for 5.1

K-map
\[
\begin{array}{c|cc|c|c|c}
\hline
a/bc & 00 & 01 & 11 & 10 \\
\hline
0 & 1 & 0 & 1 & x \\
1 & 0 & x & 0 & 1 \\
\hline
\end{array}
\]

\[ f(a, b, c) = (b+c')(a+c+d') \]

2. Use Karnaugh map to simplify function
\[ f(a, b, c, d) = \sum m(1, 4, 9, 10) + \sum d(0, 5, 7, 15). \]

List all possible minimal two-level product of sums expressions. Show the switching functions. No need for the schematic diagram.

**5.2 Solution for 5.2**

K-map

<table>
<thead>
<tr>
<th>cd/ab</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>00</td>
<td>1</td>
<td>X</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ f(a,b,c,d) = (a'+b')(c'+d')(a+c')(a'+c+d) \]