CSE140 Homework 2, Due 11:59PM, Mon 4/24/17

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.

(a)

#	a	b	c	d
a	a	a	a	a
b	a	b	a	b
c	a	a	c	c
d	a	b	c	d

(b)

&	a	b	c	d
a	a	b	c	d
b	b	b	d	d
c	c	d	c	d
d	d	d	d	d

Table 1: 2-input operators # and &.

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

2 Bit Counting Machine

A bit counting machine reads a binary input vector (a, b, c, d, e) and produces a binary number (s_2, s_1, s_0) that counts the number of "1"s in the input bits. For example when $(a, b, c, d, e) = (0, 1, 1, 1, 1)$, we have output $(s_2, s_1, s_0) = (1, 0, 0)$, and when $(a, b, c, d, e) = (1, 1, 0, 1, 0)$, we have output $(s_2, s_1, s_0) = (0, 1, 1)$.
1. Derive the canonical sum of **minterms** expression of the three output bits.
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`)

```
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
```

3 Priority Encoder

A priority encoder reads a vector of six binary bits $(a_5, a_4, a_3, a_2, a_1, a_0)$ and produces a binary number (d_2, d_1, d_0) that presents the lowest index of the input bit which is true. For example, when $(a_5, a_4, a_3, a_2, a_1, a_0) = (0, 1, 0, 1, 0, 0)$, the output $(d_2, d_1, d_0) = (0, 1, 0)$; $(a_5, a_4, a_3, a_2, a_1, a_0) = (1, 0, 0, 0, 0, 0)$, the output $(d_2, d_1, d_0) = (1, 0, 1)$; and when $(a_5, a_4, a_3, a_2, a_1, a_0) = (0, 0, 0, 0, 0, 1)$, the output $(d_2, d_1, d_0) = (0, 0, 0)$. However, when none of the input bits asserts a true value, i.e. $(a_5, a_4, a_3, a_2, a_1, a_0) = (0, 0, 0, 0, 0, 0)$, the output remains to be $(d_2, d_1, d_0) = (0, 0, 0)$.
1. Derive the **sum of products** expression of the three output bits.
2. fill in the following Bluespec code that implements the priority encoder.

```
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
```

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. 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.

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. 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.