Fundamentals of Digital Logic Design

ECE/CS 3700
Spring 2013, Homework # 4
Due Date: Mon, April 1, 5pm sharp, due in the HW locker.

1) (15 points) **Propagate-generate-delete signals in adders**. In class, we studied the carry look-ahead adder as a means to speed-up carry propagation delay of the ripple carry adder. Look-ahead adders make use of the generate (G_i) and propagate (P_i) signals to *precompute* whether or not stage i would output a carry. Similar to using the propagate (P_i) and generate (G_i) signals, a look-ahead adder can be designed by using a propagate (P_i), generate (G_i) and delete (D_i) signal. Consider the Truth table of the full adder shown in Table I:

TABLE I
TRUTH TABLE OF A FULL ADDER

a	b	c_i	S	c_{i+1}	Carry Status
0	0	0	0	0	Delete
0	0	1	1	0	Delete
0	1	0	1	0	Propagate
0	1	1	0	1	Propagate
1	0	0	1	0	Propagate
1	0	1	0	1	Propagate
1	1	0	0	1	Generate
1	1	1	1	1	Generate

The first two minterms m_0, m_1 correspond to the condition where the carry-out signal gets suppressed (deleted) at c_{i+1}, independent of the value at c_i. Prove that:

- The Delete signal $D_i = a' \cdot b'$.
- $Sum = P_i \cdot \overline{C_i} + D_i \cdot C_i + G_i \cdot C_i$
- $C_{i+1} = G_i + \overline{D_i} \cdot C_i$.

2) (10 points) Given that A, B, C_i are the inputs to a full adder, S and C_o are sum and carry-out, respectively, prove that:

- $S = ABC_i + \overline{C_o}(A + B + C_i)$.

3) (25 points) **Multiplier design**. You are asked to design an array multiplier that multiplies a 4-bit number $A = (a_3, a_2, a_1, a_0)$ by a 3-bit number $B = (b_2, b_1, b_0)$. Consider that pre-designed 4-bit adders are available to you. (Recall that a 4-bit adder adds two four-bit numbers). Design the multiplier using only **two 4-bit adders** and a minimum number of two-input AND/OR/NOT/XOR/XNOR gates. Show a block diagram or a schematic of your design depicting input and output bits clearly. (Solve this problem properly, and you've understood the concept of array multipliers!).

4) (15 points) **Two's complement numbers**. Suppose that you are given two **3-bit unsigned** numbers $A[2:0], B[2:0]$ that have to be added together ($C = A + B$). Suppose, further, that you are given a pre-designed *4-bit adder* that you have to use for this purpose. A 4-bit adder takes inputs $X[3:0], Y[3:0]$ and adds them. In order to do the addition correctly, we can take vectors A, B and concatenate a leading 0 to make them 4-bit vectors and then map the inputs; i.e. in Verilog terms: $X[3:0] = \{1'b0, A[2:0]\}$; and similarly $Y[3:0] = \{1'b0, B[2:0]\}$. This way, n-bit unsigned integers can be scaled to larger bits.
However, this technique may not work for 2's complement scheme. So, now you have to answer the following: You are given two 3-bit vectors $A[2:0], B[2:0]$ that are already given in **3-bit two's complement form**. You are asked to subtract $A - B$. Suppose that you are already given a *4-bit subtractor* (say, the design of Fig. 5.13, pp. 248 in the textbook) that takes $X[3:0], Y[3:0]$ and computes $C = X - Y$, where C is a 4-bit two's complement number. You have to use this 4-bit subtractor to subtract the given 3-bit numbers A, B. How will you scale (or modify) the inputs correspondingly? Show your design/schematic (or a Verilog code, if you wish), and demonstrate the correct functioning of your circuit using an example.

5) (10 points) **Decoders**. Solve problem 6.1, pp. 375, from the textbook.

6) (10 points) **Shannon's Expansion and MUXes**. Solve problem 6.5, pp. 375, from the textbook.

7) (15 points) **A practical example of Code-Converters**. Braille is a system of raised dots that can be read by a blind person. You are asked to design an *encoder* circuit that converts Binary Coded Decimal (BCD) numbers to Braille. The Braille patterns for the BCD numbers are shown below in Fig. 1.

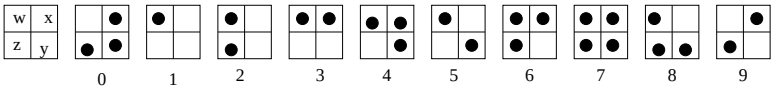

Fig. 1. Braille representation of digits 0 to 9.

Derive a minimum sum-of-product form representation for each of the four Braille dot outputs X, Y, W, Z in terms of a 4-bit BCD number. Denote the 4-bit BCD number as $ABCD$ where A is the most significant bit, and D the least significant bit.