Homework 2

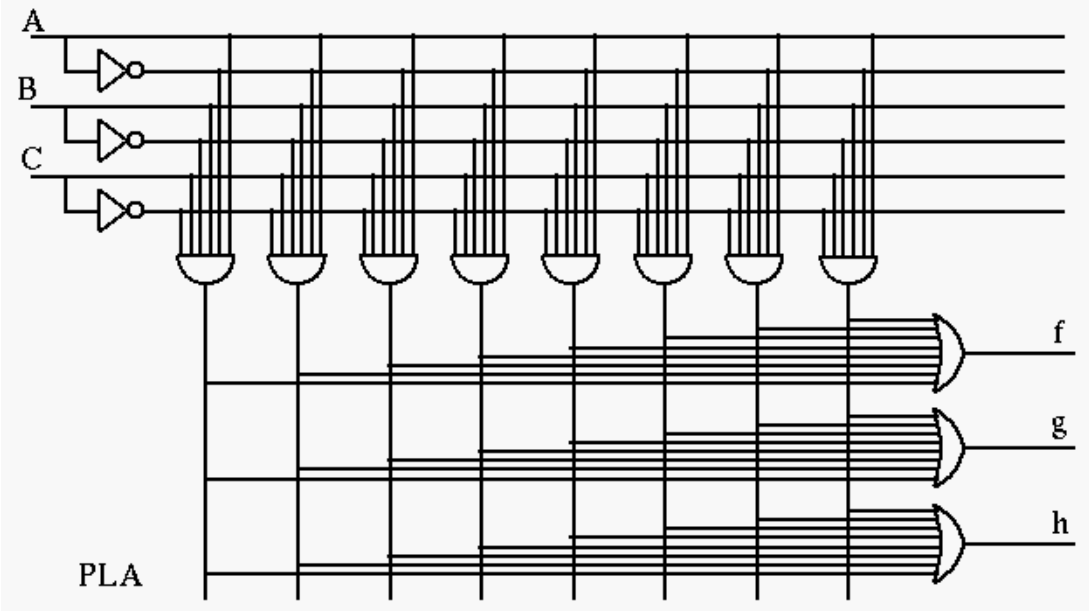
1. Design two separate combinational circuits at the gate level for the middle and bottom segments of a seven-segment display. Make sure your design uses the minimum number of logic gates with minimum number of inputs. The 4-bit inputs are (x_3, x_2, x_1, x_0), and the single output is the signal for lighting the desired segment, which is `1` when needed for displaying the decimal digit (0–9) represented by the binary input, or `0` otherwise. Treat the combinations of inputs that represent 10–15 as "don't cares."

2. Assume you need a 64-to-1 multiplexer controlled by six select signals A through F, i.e., a binary number ABCDEF specifies the index of the input to be sent to the output. However, as we assume a logic gate can not have more than eight inputs; such a MUX is not directly available.

   ■ How would you build this 64-to-1 MUX using 4-to-1 MUXs? Label clearly in your design how the signals A through F are used in your design.
   ■ Assume you have only eight 8-to-1 MUXs available. How would you build the 64-to-1 MUX using these MUXs and minimum additional logic gates (NOT, AND and OR only)? Again, show how the six select signals are used in your design.

   Hint: in your design, you can assume each circuit has an $\overline{\text{enable}}$ input that, when set to `0`, enables the circuit to do whatever it is supposed to do, but when set to `1`, disables the circuit so that the output is always `0`.

3. Implement the three functions $f(A, B, C) = \Sigma(1, 2, 4, 6, 7)$, $g(A, B, C) = \Sigma(0, 2, 3, 5)$, and $h(A, B, C) = \Sigma(2, 3, 4, 6)$ by disconnecting (marking by "x") all necessary links in the AND and OR arrays in the PLA given in the following figure.



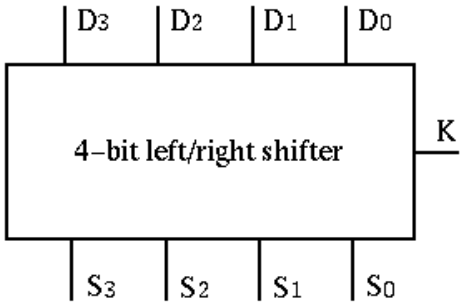
4. A full adder was discussed and simulated in Lab 1. Now consider a 1-bit full adder/subtractor. The two input bits and are to be operated on, under the control of a signal k (k = `0` for addition; k = 1 for subtraction). Another input is for either the carry or borrow from the lower bit. The outputs include the result r (sum or difference) and *cout* for carry or borrow to the next higher bit.

   ■ Design this 4-input 2-output circuit and implement it at teh logic gate level. Make sure to minimize the hardware as much as possible.
   ■ How would you implement this circuit with two 8-input MUXs and minimal number of inverters?
   Hint: Try the example `1101000110 - 011001010 = ?` to help yourself figure out how to do binary

subtraction. If you are still not sure how to subtract, try a decimal subtraction such as `123 - 89 = ?`.

5. Design a comparator circuit that takes two 4-bit words, $a_3a_2a_1a_0$ and $b_0b_1b_2b_3$, as inputs, and generates one output f that is 1 if the two words are identical, i.e. $a_i = b_i$, i = 0, 1, 2, 3. Note that there are eight input variables and $2^8 = 256$ minterms in the truth table. Obviously K-map simplification (containing 256 squares) cannot be used. (Hint: consider XNOR, which is a 1-bit comparator.)

6. Design a left/right shifter that takes a word of four bits $D_3D_2D_1D_0$ as input, and generates as output a word also of four bits $S_3S_2S_1S_0$, which is a shifted version of the input. When a control signal k is equal to `1`, the input word is shifted to the right by `1` bit to become the output, i.e, $S_i = D_{i+1}$, for i = 0, 1, 2, and $S_3 = D_0$ (right rotation). When $k = 0$, the input word is shifted to the left by `1` bit, so $S_i = D_{i-1}$, for i = 1, 2, 3, and $S_0 = D_3$ (left rotation). Note that the truth table of this circuit with five (4 + 1) inputs has $2^5 = 32$ minterms, and the K-map simplification method would be very difficult to use.



7. Design the interface circuit between computer 1 and computer 2 for transmitting any of the four letters A, B, C and D, which are coded by three bits x_3, x_2, x_1 in computer 1 but 2 bits y_2, y_1 in computer 2, as shown below. Note that all xs in the table mean "don't care" (either `1` or `0`); i.e., the letter A is coded in computer 1 by `010`, `011` or `100`.

	A	B	C	D
x_1	0 1	1	0	1
x_2	1 0	1	0	0
x_3	x 0	x	x	1
y_1	0	0	1	1
y_2	0	1	0	1

Give the gate level implementation of the interface circuit. Can you use only five logic gates (excluding NOT gates) for the circuit to generate both y_2 and y_1?