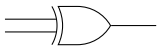
Homework Exercises week 2: Boolean Algebra, Gates & Circuits

IPC006 Processoren

Hand in before Monday November 19th 2018 at 20:00.

Discuss the following two exercises in the exercise class:

1. The NEXOR function is the negation of the XOR function (NEXOR = 'Not XOR'), sometimes called XNOR.
   (a) Give the truth table of NEXOR of two inputs, A and B.
   (b) Write down a Boolean function for the NEXOR.
   (c) The following symbol is commonly used for the XOR gate:

   Compare the NAND and AND gates, and the NOR and OR gates in the book, and propose a symbol for the NEXOR gate.
   (d) Give a circuit for computing the NEXOR using only NAND, NOR, AND and OR gates.

2. Consider the Boolean function F, with inputs A, B, C and D, with the following truth table:

A	B	C	D	F
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

   (a) Give a Boolean expression for F in (full) Disjunctive Normal Form.
   (b) Create a Karnaugh map for this function, and use this to derive a simplified expression for F.

Hand in the solutions to the following exercises via Brightspace.

3. For each of the Boolean expressions below, perform the following two steps:

   (i) Find the simplest expression that is equivalent to the given expression, using a Karnaugh map. (You may either do this directly, or write down a truth table first.)

   (ii) Show that the given expression can be rewritten to its simplified form using the laws of Boolean algebra (refer to the lecture slides or to Figure 3-6, page 156, of Tanenbaum for these). For each rewrite step, name the rules that you apply. Associativity and commutativity may be applied without mentioning them explicitly.

   The expressions are:

   (a) $a\,\overline{b} + \overline{b}\,c + a\,b\,\overline{c}$

   (b) $\overline{a}\,b\,c + a\,c + \overline{b}\,c$

4. Using the three-variable multiplexer chip (shown in the left part of the figure below), implement a function whose output is the *parity* of the inputs A, B, C. That is, the output is 1 if an even number of these inputs is 1. For inspiration, the right part of the figure shows how to do this for the majority function.
   (Figure 3-12 and Exercise 3.6 from Tanenbaum, 5th ed.)

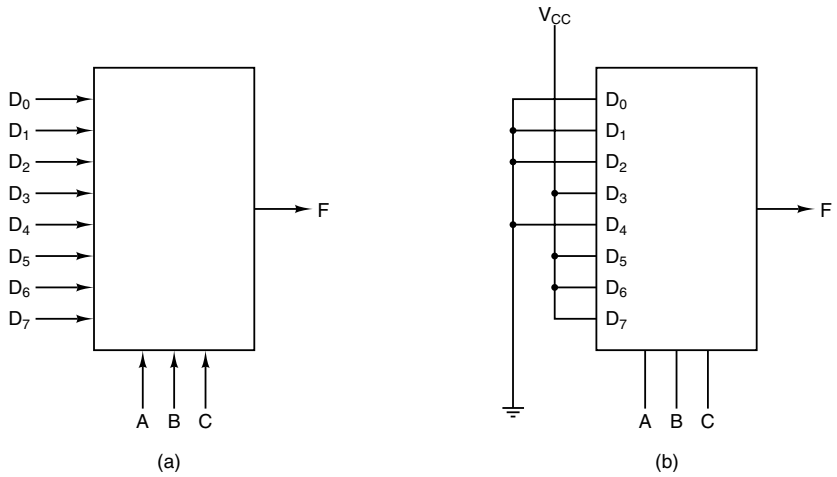


Figure 3-12. (a) An MSI multiplexer.. (b) The same multiplexer wired to compute the majority function.

5. We are designing the circuit of a 2-bit encoder with four input lines (I_0, I_1, I_2, I_3). If exactly one of the input lines is high, the two output lines B_1, B_0 are such that their 2-bit binary value B_1B_0 tells which input (I_0, I_1, I_2, I_3) is high. For example, when line I_2 is high, the output lines represent 10. When no input line is high, or more than one input line is high, you can choose the output value (these inputs are irrelevant).

   (a) Give the truth table of the circuit. You can use X (don't care) to denote that the output value is irrelevant.

   (b) Draw a circuit for the 2-bit encoder. Your circuit should be as simple as possible; this can be achieved by looking carefully at the truth table, or by creating Karnaugh maps for the outputs. (If you decide to use Karnaugh maps, each X may be either included or excluded from the blocks, whichever produces the simplest result. Also, you might find a slightly simpler circuit if you consider the maps for both outputs simultaneously.)