

ENEE 244 (01**). Spring 2006

Homework 4

Due back in class on Friday, April 7.

1. Implement the following Boolean expression with exclusive-OR and AND gates only:
 $F = AB'CD' + A'BCD' + AB'C'D + A'BC'D.$

$$\begin{aligned} F &= AB'CD' + A'BCD' + AB'C'D + A'BC'D \\ &= AB'(CD' + C'D) + A'B(CD' + C'D) \\ &= (AB' + A'B)(CD' + C'D) \\ &= (A \oplus B)(C \oplus D) \end{aligned}$$

2. Determine whether the gate set {AND, EXCLUSIVE-OR} is functionally complete. Prove your answer.

We know that {AND, OR, NOT} is functionally complete.

To prove that another set is functionally complete, it is enough to show that AND, OR and NOT can each be implemented in terms of the new set.

Here that implies that OR and NOT need to be implemented using AND, XOR.

To implement NOT:

$$a' = a'.1 + a.0 = a \oplus 1$$

To implement OR:

$$\begin{aligned} A + b &= (a + b)'' \\ &= (a'.b')' \quad // \text{Using Demorgan's law.} \end{aligned}$$

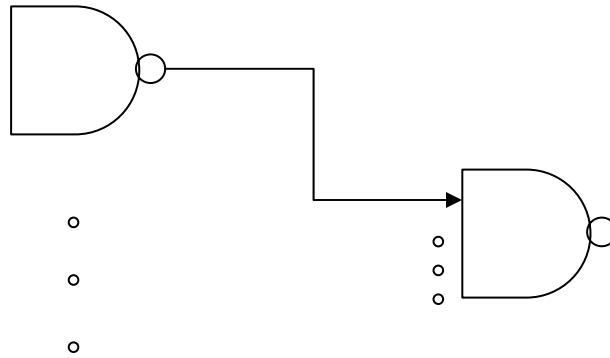
This last term contains only AND and NOT gates, each of which have already been shown to be expressible in the set {AND, XOR}.

Hence both NOT and OR can be implemented in terms of {AND,XOR}.

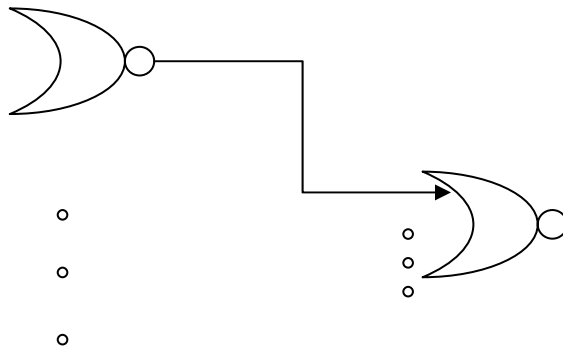
\Rightarrow {AND,XOR} is functionally complete.

3. A two-level all-NAND implementation of a complex logic function $z(x_1, x_2, \dots, x_n)$ has been fabricated in a IC at considerable expense. As the IC is about to be installed in a system it is discovered that all the gates it contains are actually NORs instead of NANDs. It is now too late to fix the IC itself, but it is possible to add some extra logic to the IC's input/output lines. So what should the IC designer do to fix this flaw so that the IC in question can be used?

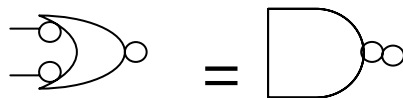
We need:



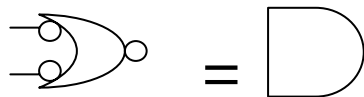
But we have:



We can add NOT gates at all inputs and the output to convert the second into the first:
This is because, at inputs:



This is a NAND gate with an extra NOT gate at output. We can move this extra NOT gate between the two levels of the circuit to the inputs of the second level. This gives:



Since a NOT gate is added at the output as well, this second-level gate becomes a NAND gate as well. Thus, adding NOT gates at all inputs and the output is equivalent to converting all the internal NOR gates to NAND gates, as desired.

4. Design a 4 bit combinational circuit 2's completer for an input $x = x_3x_2x_1x_0$. (The output generates the 2's complement of the input binary number). Show that the circuit can be constructed using {AND,NOT,XOR} gates only. Do not use any full adders.

Computing the 2's complement involves complementing each bit and adding 1 to the overall result.

Adding 1 is the same as adding 0000 to the number, but with a 1 as input carry.

Since the number added is always 0000, we can omit it and use a 2-bit adder at each stage of the ripple-adder, instead of the usual 3-bit adder. The two bit adder will add x_i' and c_i to yield $c_{i+1} s_i$. (Incidentally a building block that adds 2 bits is called a half adder, unlike a 3-bit block which is called a full adder).

This circuit can be implemented by specifying a 2-bit adder directly in terms of {AND, NOT, XOR}. The formulas for a 2-bit adder can be derived by first principles, or, in the equations for s_i and c_{i+1} on page 232 of the Givone textbook, substituting $y_i=0$ and x_i' for x_i . This yields:

$$\begin{aligned} s_i &= x_i' \oplus c_i && // \text{This is the same as } (x_i \oplus c_i)' \\ c_{i+1} &= x_i' c_i \end{aligned}$$

These equations use only {AND,NOT, XOR}, so we are done. Cascading four of these circuits in the way of an adder yields the desired completer.

5. A combinational circuit has 3 outputs F1, F2 and F3:

$$\begin{aligned} F1 &= x'y'z' + xz \\ F2 &= x y'z' + x'y \\ F3 &= x'y'z + xy \end{aligned}$$

Design the circuit with a decoder and external gates.

First we express the functions in minterm canonical form by expanding the second term with the missing literal. For example:

$$F1 = x'y'z' + xz(y+y') = x'y'z' + xyz + xy'z = \Sigma m(0,5,7).$$

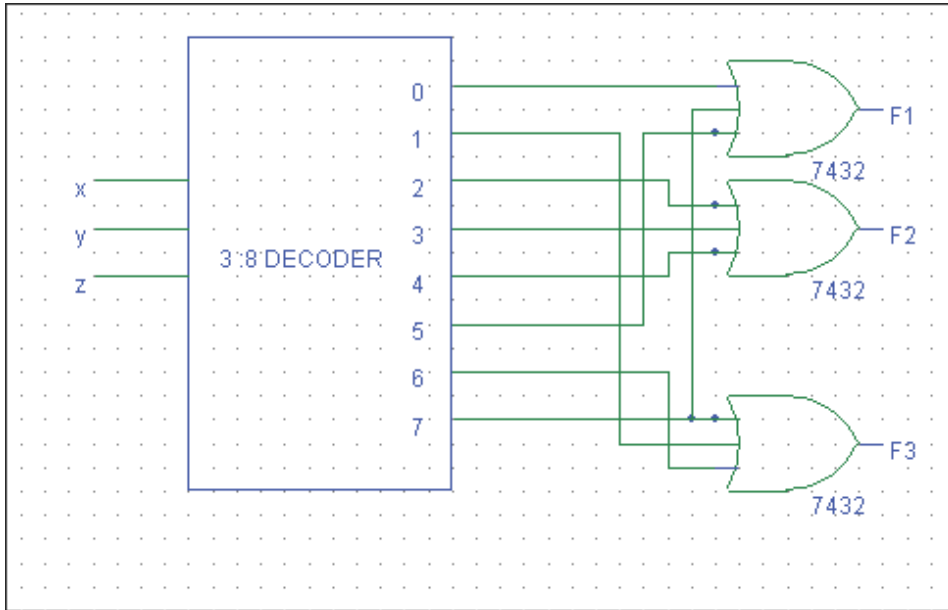
Similarly:

$$F2 = \Sigma m(2,3,4).$$

$$F3 = \Sigma m(1,6,7).$$

The circuit is designed by using one 3-to-8 decoder, but using different minterm outputs for each function to be ANDed together. Circuit is as below:

(In the circuit, 7432 is the component number of an OR gate.)

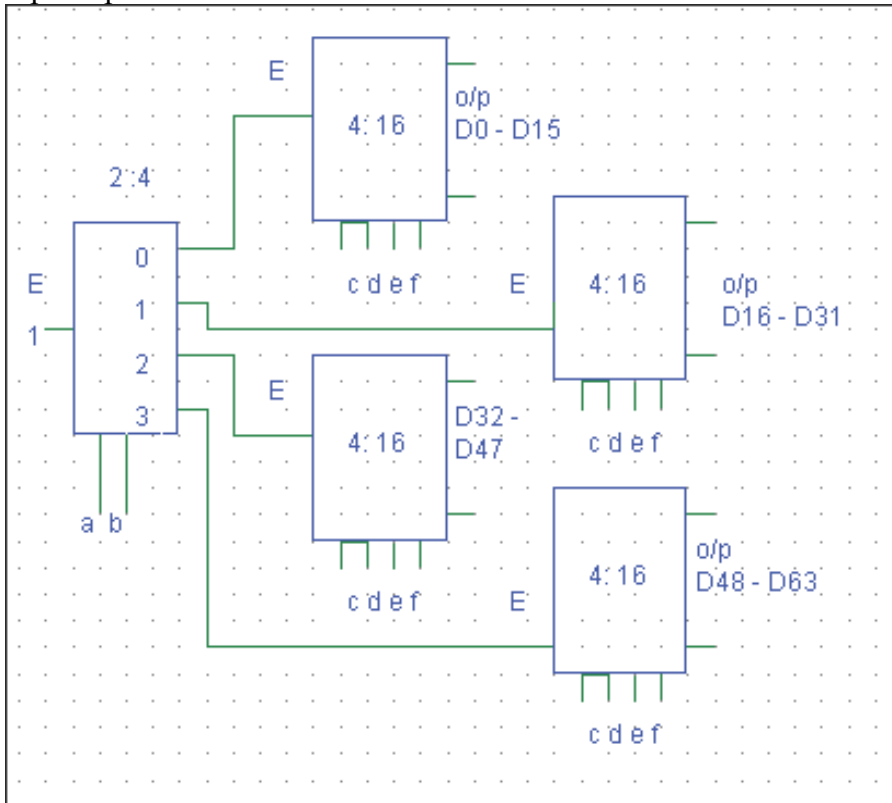


6. Demonstrate how to implement a 6:64 decoder using generic 2:4 and 4:16 decoders. The input decoders have enable inputs.

To design a 6:64 decoder let the inputs be a,b,c,d,e,f and let the outputs be D0–D63. The design is as shown below.

E: enable input

o/p: output

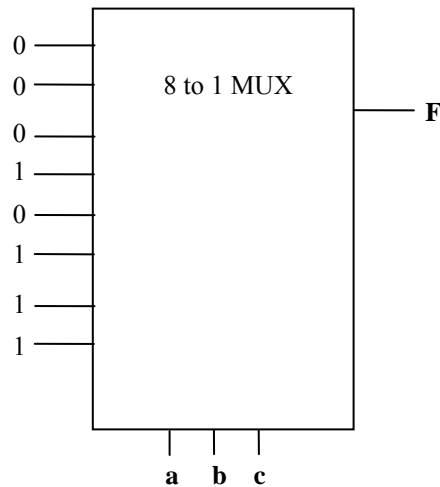


7. A majority function F takes three 1-bit inputs a, b and c , and produces a single bit as output that is one if and only if two or more bits of the input are one. Design this circuit using an 8:1 multiplexor.

We can write F directly by observing the minterms whose numerical values have two or more bits which are 1.

$$F = \sum m(011, 101, 110, 111) \\ = \sum m(3, 5, 6, 7)$$

To implement a function using a MUX, we provide the function inputs as the select inputs to the MUX, and connect the minterms in the canonical form to 1. Thus the circuit is:



8. The outputs of a 3-to-8 decoder are connected in order to the inputs of an 8-to-1 MUX. The inputs of the decoder are a, b and c ; the select inputs of the MUX are d, e and f . What function does the 1-bit output of the MUX implement in terms of a, b, c, d, e and f ?

A decoder turns its i^{th} output high if the input $abc=i$. A MUX transmits the value of its j^{th} data input to the output if the select input $def=j$.

Thus a 1 will be transmitted to the output by the MUX only if $i=j$; and 0 otherwise. Hence the output function F is 1 if the number $abc = def$. More formally:

$$F = (a \odot d).(b \odot e).(c \odot f) \quad // \odot \text{ is } XNOR.$$