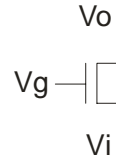


**Problem 1**

For problems 2-4 below, find the voltages required to keep the transistor 'on' by applying the rules discussed in class. Assume  $V_{DD} = 2.2V$



	FET type	$ V_t $ (V)	$V_g$ (V)	$V_i$ (V)
a	n-type	0.5	2.2	1
b	n-type	0.6	1.5	1
c	p-type	0.4	0	2
d	p-type	0.5	1.7	2
e	p-type	0.5	1	1.5

solution

- a)  $V_g - V_i = 2.2 - 1 = 1.2 > V_{tn}$ , so  $V_o = V_i = \underline{1V}$   
 b)  $V_g - V_i = 2 - 1.5 = 0.5 < V_{tn} = 0.6$ , so  $V_o = V_g - V_{tn} = 1.5 - 0.6 = \underline{0.9V}$   
 c)  $V_i - V_g = 2 - 0 = 2 > |V_{tp}|$ , so  $V_o = V_i = \underline{2V}$   
 d)  $V_i - V_g = 2 - 1.7 = 0.3 < |V_{tp}|$ , so  $V_o = V_g + |V_{tp}| = 1.7 + 0.5 = \underline{2.2V}$   
 e)  $V_i - V_g = 1.5 - 1 = 0.5 = |V_{tp}|$ , so  $V_o = V_i = \underline{1.5V}$   
 or  $V_o = V_g + |V_{tp}| = 1 + 0.5 = \underline{1.5V}$

which proves that in the equal case, either solutions works and gives the same answer

**Problem 2**

Find the midpoint voltage,  $V_x$ , and output voltage,  $V_o$ , for the chain of two nFET pass transistors shown below for the following cases. Assume  $V_{DD} = 2.5V$  and  $V_{tn} = 0.5V$ .

solution

- (a)
- $V_i = 0V$

Here  $V_{DD} - V_i = 2.5 > V_{tn}$ , so  $V_x = V_i = \underline{0V}$

and  $V_{DD} - V_x = 2.5 > V_{tn}$ , so  $V_o = V_x = \underline{0V}$

notice this is a shows that two series nMOS can pull

the output to ground when the  $V_i$  node is at ground (as in a NAND gate).

- (b)
- $V_i = 1.1V$

Here  $V_{DD} - V_i = 1.4 > V_{tn}$ , so  $V_x = V_i = \underline{1.1V}$

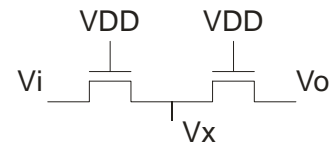
and  $V_{DD} - V_x = 1.4 > V_{tn}$ , so  $V_o = V_x = \underline{1.1V}$

- (c)
- $V_i = 2.2V$

Here  $V_{DD} - V_i = 0.3 < V_{tn}$ , so  $V_x = V_{DD} - V_{tn} = \underline{2.0V}$

and  $V_{DD} - V_x = 0.5 = V_{tn}$ , so  $V_o = V_x = \underline{2V}$

*Notice when  $V_g - V_i = V_{tn}$  you get the same answer if you use case 1 or case 2, for example, in this problem case 2 gives  $V_o = V_{DD} - V_{tn} = 2V$ , which is the same as case 1.*



### Problem 3

Using fundamental logic properties, prove the following logic relationships

#### solution

(a)  $(a+b)(a+c) = a+bc$

First expand the term (use the FOIL method of mathematics)

$$\begin{aligned}(a+b)(a+c) &= aa + ab + ac + bc = a + ab + ac + bc \text{ (since } a \cdot a = a\text{)} \\ &= a(1+b) + ac + bc = a + ac + bc \text{ (since } 1+b = 1 \text{ and } a \cdot 1 = a\text{)} \\ &= a(1+c) + bc \\ &= a + bc \text{ (since } 1+c = 1 \text{ and } a \cdot 1 = a\text{)}\end{aligned}$$

(b)  $a + a'b = a + b$

First the easy way. Since we know from part (a) above that  $x+yz = (x+y)(x+z)$

let  $x = a$ ,  $y = a'$ , and  $z=b$

thus  $a+a'b = (a+a')(a+b)$

$= a+b$  (since  $a+a' = 1$ )

Or, if we stick with the 'fundamental properties' we can follow the steps in part (a) in reverse

$$\begin{aligned}a+a'b &= a(1+a') + a'b = a + aa' + a'b \\ &= a(1+b) + aa' + a'b = a + ab + aa' + a'b \\ &= (a+a')(a+b) = a+b \text{ (because } a+a' = 1 \text{ and } 1(a+b) = a+b\text{)}\end{aligned}$$

### Problem 4

Design a CMOS logic gate for the function  $f = \overline{x + y \cdot (z + x)}$  using the least number of transistors by completing the following steps:

- Identify the nMOS portion of the circuit by setting up and reducing the equation for  $f_n$  and then implementing this equation as a **gate-level** schematic. Note, because this is nMOS logic, the gate-level schematic should have an inversion at the output but no inversions at the inputs (assert-high logic).
- Apply bubble pushing on the **gate-level** schematic in (a) to construct the gate-level schematic for the pMOS portion of the circuit. Here, the bubble at the output must be pushed to the inputs and all inputs must have a bubble on them (assert-low logic). Be sure to properly apply the DeMorgan relations and modify logic gates appropriately.
- Use the gate-level schematic in (a) to construct the **transistor-level** schematic for the nMOS portion of the circuit, properly applying the series/parallel transistor rules for each AND and OR function.
- Repeat part (c) for the pMOS portion of the circuit, adding this to the top of the nMOS portion to form a complete CMOS logic circuit.

solution

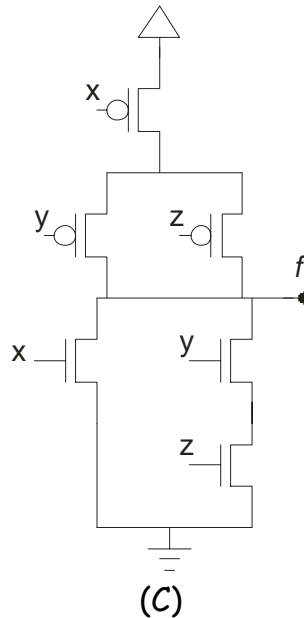
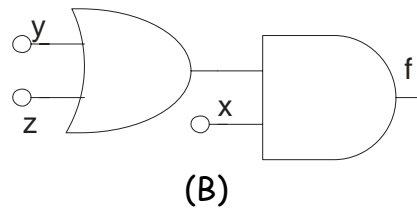
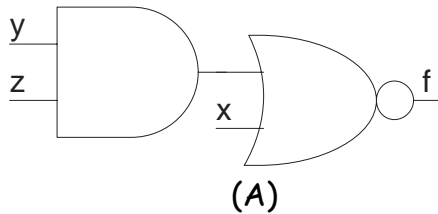
a) First, setup the nMOS equation and reduce it.

$$F_n = \overline{f} = x+y(z+x) = x+yz+yx = x(y+1) + yz = \underline{x + yz}$$

which can be implemented in logic gates as shown in Fig. A where the bubble at the output represents the inverting nMOS logic.

b) Next, use this logic gate schematic and push the bubble at the output to the inputs in order to implement the pMOS network. See Fig. B.

c, d) Finally, use AND = series and OR = parallel to implement both networks with transistors, as shown in Fig. C.



**Problem 5**

Follow the procedure below to construct the CMOS logic gate for the function  $f = \overline{\overline{a + b \cdot c}}$ .

- a) Write the equation for the nMOS network.
- b) Write the equation for the pMOS network.
- c) Use the equations in (a) and (b) to construct a schematic for  $f$ .
- d) Verify the nMOS and pMOS networks are proper complements (series groups in nMOS are parallel in pMOS, etc.).

solution

a) Since the entire function  $f$  is inverted, we just remove the top-level inversion to form  $f_n$ .

$$f_n = (b \cdot c)' + a$$

But we still need to remove the inverted operations

$$f_n = (b \cdot c)' + a = b' + c' + a$$

b) First, remove all inverted operations. Below the ' symbol represents a NOT

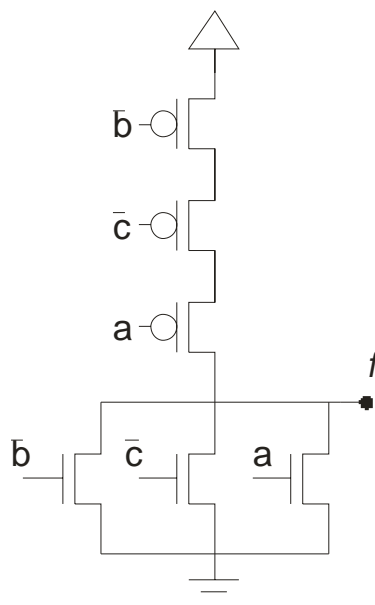
$$f = ((b \cdot c)' + a)' = (b \cdot c) \cdot a'$$

Now invert all of the inputs to form  $f_p$

$$f_p = b' \cdot c' \cdot a$$

Notice that now  $f_p$  and  $f_n$  both have the same inputs ( $b'$ ,  $c'$ ,  $a$ ) and the operations in  $f_p$  (all ANDs) are complements of the operations in  $f_n$  (all ORs).

c)

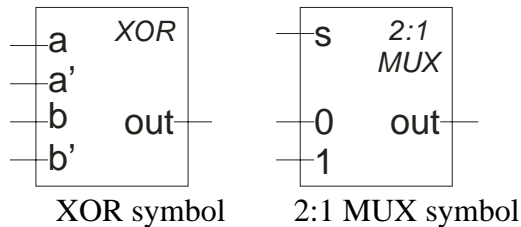


d) Observation of the schematic shows that the pMOS and nMOS networks have the same input signals ( $b'$ ,  $c'$ ,  $a$ ) and each of the operations is complemented within the other network, i.e., all series combinations in pMOS are parallel combinations in nMOS.

**Problem 6 (Design Challenge)**

a) As discussed in lecture, the XOR and XNOR CMOS schematics are very similar; they have exactly the same structure but two of the inputs are different. Based on this, construct a **gate-level** schematic (no transistors, just gate symbols) for a circuit that uses only one XOR gate to implement both XOR and XNOR functions. The symbol for the CMOS XOR gate is shown below, having inputs **a**, **a'** (NOT a), **b**, and **b'** (NOT b). A signal **S** should be used to select between the XOR and XNOR function. You can use any additional INV, MUX, NAND and NOR gates needed to implement this function. The completed gate should only have inputs **M**, **N**, and **S** and output **Z** such that:

$$\begin{aligned} \text{if } S = 0, Z &= M \text{ XOR } N \\ \text{if } S = 1, Z &= M \text{ XNOR } N \end{aligned}$$



Notice that you will have to use inverters to eliminate the need for inverted inputs (N' or M').

b) Assuming the 2:1 MUX has a built-in control signal inverter and requires 6 transistors, how many transistors does your design require?

solution

To switch the XOR gate to an XNOR function, we simply need to invert the **a** input (or, alternatively the **b** input). If we let **M** be the **a** input to the XOR gate, we need our circuit to pass **M** if **S** = 0 or invert **M** if **S** = 1. There are many ways to do this, but an obvious choice might be to use a 2:1 MUX at both the **a** and **a'** inputs of the XOR gate, so signal **S** could pick which input to use and thus alter the function from XOR to XNOR. This is shown in Fig. A and requires 24 transistors.

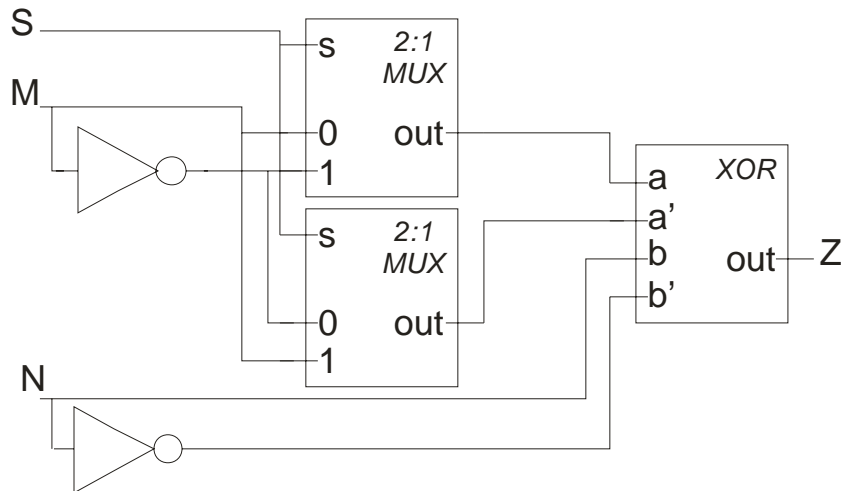


Fig. A. Two MUX solution to Problem 6

A better choice would be to eliminate one MUX and simply invert the output of the MUX after the MUX as shown in Fig. B. This requires only 20 transistors. Check it out and make sure you see how this circuit meets the design goals.

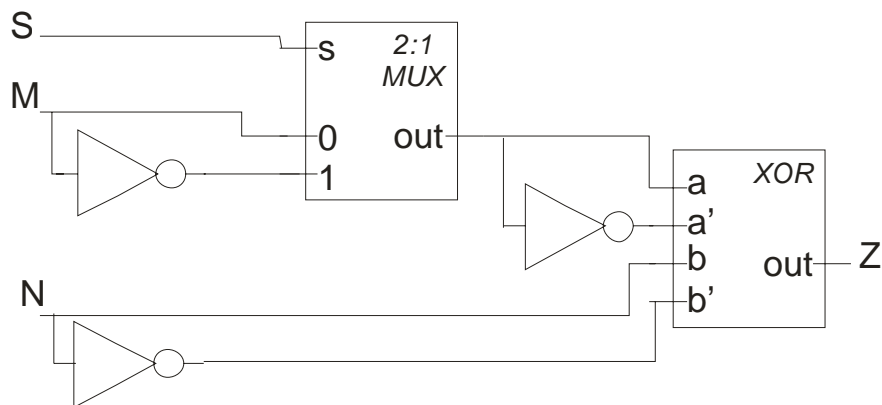


Fig. B. One MUX solution to Problem 6

Notice that this problem says we can only use one XOR gate, but if we were not limited to that, a more elegant (fewer transistors) solution might be available. Take a look at the AND and OR truth tables below. Do either of them implement the function M if  $S = 0$ ,  $M'$  if  $S = 1$ ? No, they don't. But notice this is exactly what the XOR function does. Can you implement a selectable XOR/XNOR using two XOR gates? Is it fewer transistors than your design?

S	M	M AND S	M OR S	M XOR S
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Think About It: In this course, you often see more than one possible solution and must learn to analyze the options to find the *best* solution. Let's think about this problem. Why would we want a selectable XOR/XNOR gate? Perhaps so we could use less circuitry to implement both functions, reusing a single XOR gate for multiple tasks. This is a fine idea and one you will want to consider often, particularly in your design project. The straightforward design, implementing both XOR and XNOR functions in different gates, would require 1 XOR, 1 XNOR and 1 MUX (to select between outputs). This takes  $8 + 8 + 6 = 22$  transistors, plus 2 inverters (4 transistors) to invert the  $a$  and  $b$  inputs. This gives us a grand total of 26 transistors. Did your solution use fewer transistors? Would the alternative option with 2 XOR gates discussed above save transistors? Once you learn to analyze the options and make the best choice to meet design specifications you'll be well on your way to becoming a better digital circuit designer.