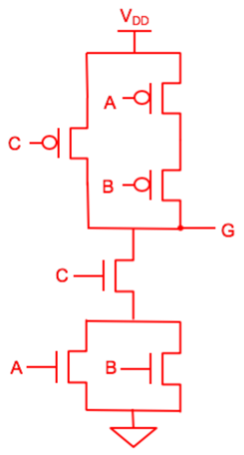
# 6.004 Tutorial Problems
# L07 – CMOS Logic

**Note:** A subset of essential problems are marked with a red star (★). We especially encourage you to try these out before recitation.

## Problem 1. ★

For each of the functions F and G, if the function can be implemented using a **single CMOS gate**, please draw the corresponding single CMOS gate. If it cannot be implemented using a single CMOS gate, then write NONE. **For full credit, use a minimum number of FETs.**

| A | B | C | F | G |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

| **Draw CMOS implementation of F(A,B,C) below or write NONE if F cannot be implemented as single CMOS gate.** | **Draw CMOS implementation of G(A,B,C) below or write NONE if G cannot be implemented as single CMOS gate.** |
|---|---|
| **F(1, 1, 1) = 1 is non-inverting so it cannot be implemented as a single CMOS gate.**<br><br>(There are many other reasons that F is non-inverting. For example, F(0, 1, 0) = 0 but F(0, 1, 1) = 1; changing the last input from a 0 to a 1 also changed F from a 0 to a 1, which means F is non-inverting.) | $\overline{G} = \overline{A}BC + A\overline{B}C + ABC$<br>$= BC + A\overline{B}C$<br>$= C(B + A\overline{B})$<br>$= C(B + A)$<br>This corresponds to the pulldown circuit (on the bottom), by translating the AND to series and the OR to parallel: C in series with (B in parallel with A).<br><br>The dual is the pullup circuit (on top): $\overline{C}$ in parallel with ($\overline{B}$ in series with $\overline{A}$).<br><br> |

**Problem 2.** ★

(A) A single CMOS gate, consisting of an output node connected to a single pFET-based pullup circuit and a single nFET-based pulldown circuit (as described in lecture), computes F(A, B, C, D). It is observed that F(1, 0, 1, 0) = 1. What can you say about the following values?

**A: 1⇨0 turns pfet on** **(circle one) F(0, 0, 1, 0) = : 0 .. [✓ 1] .. (can't say)**

That is, from the given inputs in the equation F(1, 0, 1, 0) = 1, we get the inputs to the question by changing the A input from 1 to 0. Doing this to an inverting function, implemented by a CMOS gate, cannot change the output from a 1 to a 0, so it must stay a 1.

**(circle one) F(1, 1, 1, 0) = : 0 … 1 … [✓ (can't say)]**

Nothing can be deduced about this output. Changing a 0 input to a 1 might change the output from 1 to 0, or let it stay as 1.

All CMOS gates output 0 if all inputs are 1 **(circle one) F(1, 1, 1, 1) = [✓ 0] .. 1 … (can't say)**

(B) The Boolean function F(A,B,C) can be implemented using a *single* CMOS gate operating as a combinational device that obeys the static discipline. It's known that F(1,1,0) = 1 and F(0,1,1) = 0. What can be determined about the value of F in the following cases? Please circle one of "0", "1" or "Can't tell".

**B: 1⇨0 turns pfet on** **(circle one) F(1,0,0) = 0 … [✓ 1] … Can't tell**

That is, from the given inputs in the equation F(1, 1, 0) = 1, we get the inputs to the question by changing the B input from 1 to 0. Doing this to an inverting function, implemented by a CMOS gate, cannot change the output from a 1 to a 0, so it must stay a 1.

**(circle one) F(1,0,1) = 0 … 1 … [✓ Can't tell]**

This triple of inputs is not comparable to anything we've given: from either triple of given inputs, you have to change a 0 to a 1 and a 1 to a 0 to get this triple, and when both changes happen, nothing can be deduced about the outputs.

All CMOS gates output 0 if all inputs are 1 **(circle one) F(1,1,1) = [✓ 0] … 1 … Can't tell**

(C) A single CMOS gate, consisting of an output node connected to a single pullup circuit containing one or more PFETs and a single pulldown circuit containing one or more NFETs (as described in lecture), computes $F(A,B)$. $F$ has the property that for all A, $F(A,0) = \overline{F(A,1)}$. What can you say about the value of $F(1,0)$?

**$F(1, 0) = \overline{F(1, 1)} = \overline{0} = 1$** **(circle one) F(1,0) = [✓ 1] .. 0 … can't tell**

Above we used the given equation with A = 1 and also the fact that any CMOS gate outputs 0 if all inputs are 1.

**Problem 3.**

A minority gate has three inputs (call them A, B, C) and one output (call it Y). The output will be 0 if two or more of the inputs are 1, and 1 if two or more of the inputs are 0.

In the space below, draw the *pulldown* circuit for a single CMOS gate that implements the minority function, using the minimum number of NFETs. You needn't draw the *pullup* circuit.

If you're convinced that the function cannot be implemented as a single CMOS gate, give a brief, convincing explanation.

**Can it be implemented as single CMOS gate? Circle one:** [✓ **YES**] **can't tell** **NO**

| A | B | C | Y |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

$$\overline{Y} = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

$$= BC + AB + AC$$

$$= BC + A(B + C)$$

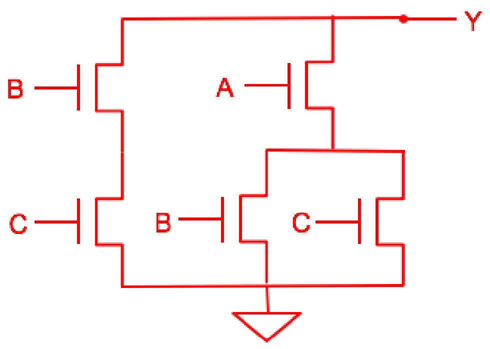
This equation for $\overline{Y}$ has none of the inputs inverted, so we can draw a pulldown circuit. Since the expression is

(B AND C) OR (A AND (B OR C)),

we convert AND → "in series with", and OR → "in parallel with" to get the description,

(B in series with C) in parallel with (A in series with (B in parallel with C)),

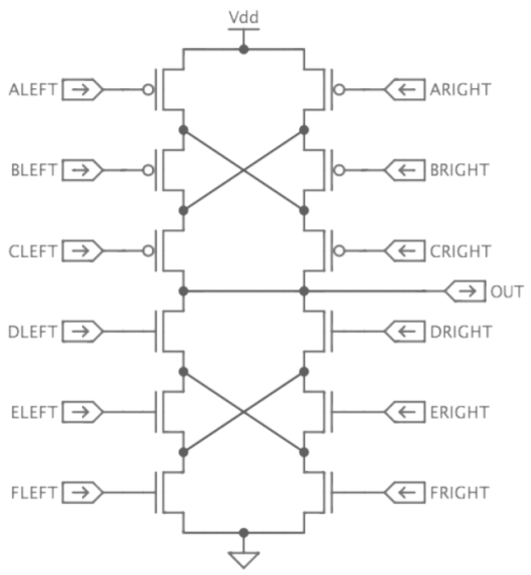
which looks like:



Note that if you didn't simplify the expression with the distributive law and directly drew $BC + AB + AC$ as a pullup circuit, you would use 6 NFETs instead of the 5 above, so it would not have the minimum number of nFETs.

**Problem 4.**

In his bid for the Lemelson Prize, Ben Bitdiddle has invented the "flexible gate," a single CMOS gate that implements different functions depending how its inputs are wired up. The FlexGate® (see figure at right) uses 6 pFETs in its pullup circuit and 6 nFETs in its pulldown circuit.



Each of the FlexGate's twelve inputs can connected to an input signal (X, Y, …), GND (logical "0"), or VDD (logical "1"). To show off its versatility, Ben has asked you to show how to connect the inputs so the FlexGate computes several different functions whose Boolean equations are given below. Associated with each equation is a table with 12 entries; in each cell of the table please write an input name, GND, or VDD as appropriate. Note that there may be several possible implementations for each of the three functions – any correct answer will be acceptable. (Note: there should be an entry in each cell, i.e., a connection should be specified for each input!)

If the desired function cannot be implemented, please draw a big "X" through the table.

**Lots of possible solutions** **Fill in the tables below or mark with "X"**

Hardwiring 0s to the pFETs in the pullup or 1s to nFETs in the pulldown is equivalent to replacing that edge with a regular conducting wire. Hardwiring 1s to the pFETs in the pullup or 0s to nFETs in the pulldown is equivalent to erasing that edge. So the general strategy is to erase or replace transistors until you get pullup and pulldown circuits matching each OUT.

$$OUT = \overline{X \times Y}$$

| input | LEFT | RIGHT |
|---|---|---|
| A | **0** | **0** |
| B | **1** | **1** |
| C | **X** | **Y** |
| D | **X** | **0** |
| E | **0** | **0** |
| F | **0** | **Y** |

$$OUT = \overline{X + Y + Z}$$

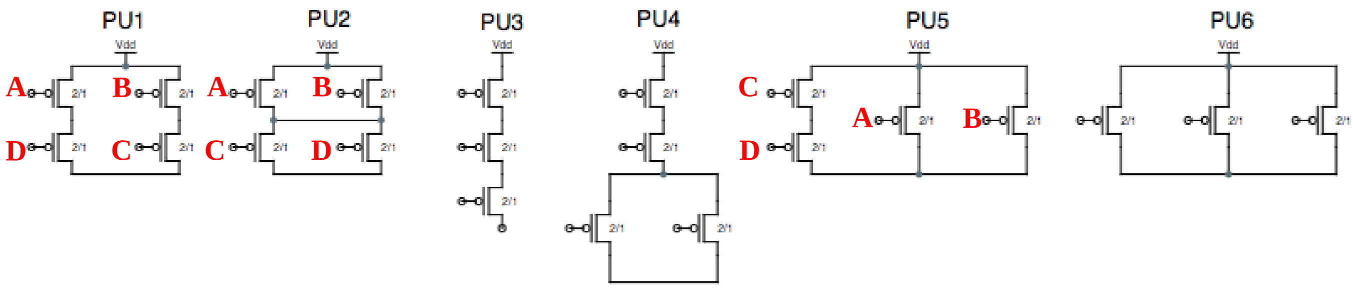
| input | LEFT | RIGHT |
|---|---|---|
| A | **Z** | **1** |
| B | **Y** | **1** |
| C | **X** | **1** |
| D | **X** | **1** |
| E | **0** | **Y** |
| F | **Z** | **1** |

$$OUT = \overline{X + Y \times Z}$$

| input | LEFT | RIGHT |
|---|---|---|
| A | **Y** | **Z** |
| B | **0** | **1** |
| C | **X** | **1** |
| D | **X** | **Y** |
| E | **0** | **Z** |
| F | **0** | **1** |

**Problem 5.** ★

You are trying to select pullups for several 3- and 4-input CMOS gate designs. You can choose between seven different pullups, given names PU1 through PU6, diagrammed below:



You can connect one of the inputs (A, B, C, or D) or a constant (GND, or VDD) to each pFET, allowing these pullups to be used in various ways to build gates with various numbers of inputs. Your goal, however, is to select the pullup that uses the minimum number of active transistors for each of the gates you are designing (i.e., a pFET that is never on is *not active*).

For each of the following 3- and 4-input Boolean functions, choose the appropriate pullup design, i.e., the one which, properly connected, implements that gate's pullup using the **minimum number of active** transistors. This may require applying De Morgan's Law or minimizing the logic equation first. If none of the above pullups meets this goal, write "NONE".

**Draw pulldown and take dual of it to find pull up circuit.**

(A) $F(A,B,C,D) = \overline{AB + CD}$ **Choice or NONE: ____PU2______**
Pulldown: (A in series with B) in parallel with (C in series with D).
Dual/pullup: (A in parallel with B) in series with (C in parallel with D), which matches PU2.

(B) $F(A,B,C,D) = \overline{(A + D)(B + C)}$ **Choice or NONE: ____PU1______**
Pulldown: (A in parallel with D) in series with (B in parallel with C)
Dual/pullup: (A in series with D) in parallel with (B in series with C), which matches PU1.

(C) $F(A,B,C) = A \cdot \overline{BC}$ **Choice or NONE: __NONE______**
F is not inverting, so no pullup could implement it. Intuitively, note that A appears without any inversion. To formally prove that F is not inverting, one can observe that, for example, F(0, 0, 0) = 0, when every inverting function must output 1 when given all 0s.

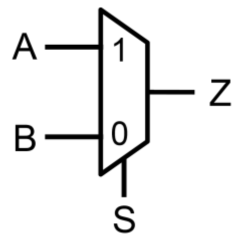
(D) $F(A,B,C,D) = \overline{A} + \overline{B} + \overline{C} \cdot \overline{D}$ **Choice or NONE: ____PU5______**

This is in the exact format to directly convert to the pullup circuit: A in parallel with B in parallel with (C in series with D). It matches PU5.

(E) $F(A,B,C,D) = \overline{(AB + C)D + (AB + C)\overline{D}}$ **Choice or NONE: ____PU2______**

Can be simplified to $F(A,B,C,D) = \overline{(AB + C)(D + \overline{D})} = \overline{AB + C}$. We can then use PU2 like in part (A), but set D = 1. It's also possible to use PU4 and hardwire one of the top transistors to 0.

**Problem 6.** ★

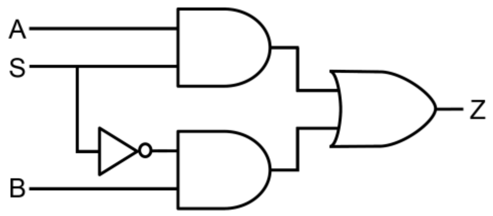


Muxes are used often so it is important to optimize them. In this problem you will design several variants of a 1-bit, 2-to-1 mux (shown to the right) using CMOS gates, and will compare their costs in number of transistors.

$\mathbf{Z = A \cdot S + B \cdot \overline{S}}$

**Note:** Remember that a CMOS gate consists of an output node connected to a *single* pFET-based pullup circuit and a *single* nFET-based pulldown circuit. Gates obtained by combining multiple CMOS gates are not a CMOS gate.

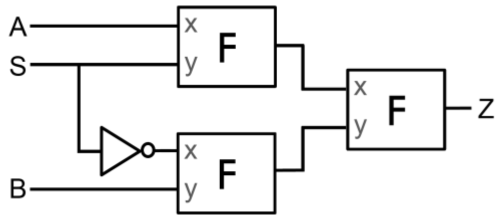
(A) Consider the implementation shown below, which uses two AND gates and an OR gate. Because a single CMOS gate cannot implement AND or OR, each AND gate is implemented with a CMOS NAND gate followed by a CMOS inverter, and the OR gate is implemented with a CMOS NOR gate followed by a CMOS inverter. How many transistors does this implementation have?



**Number of transistors in mux: ______20__________**

Each inverter has 2 transistors, one in the pullup and one in the pulldown. Each NAND or NOR gate has 4 transistors, 2 in the pullup and 2 in the pulldown. So, each AND or OR gate has 6 transistors, 4 for the NOR or NAND and 2 for the inverter. All together, this circuit has 2 + 6 + 6 + 6 = 20 transistors.

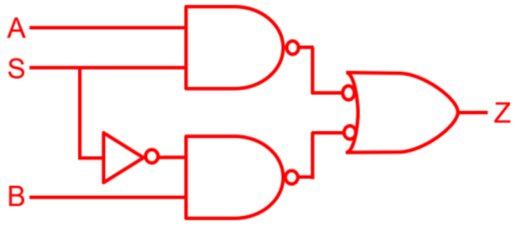
(B) Consider the implementation shown below, which uses three instances of gate F. Find the Boolean expression for F. If F can be built using a single CMOS gate, draw its CMOS implementation. Otherwise, give a convincing explanation for why F cannot be implemented as a CMOS gate. How many transistors does this implementation have?



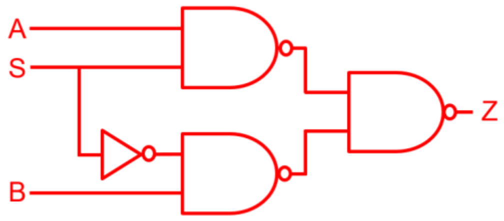
**F(x,y) = ________$\overline{x \cdot y}$__________**

**Draw CMOS gate that implements F or explain why it cannot be built.**

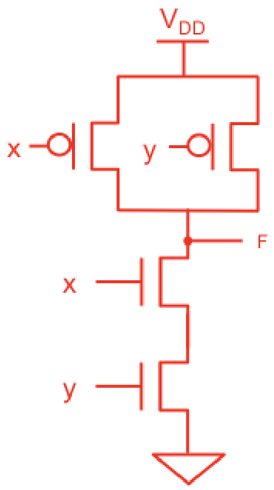
An intuitive way to look for possibilities F is to consider that Z is an OR of two things, so F is likely to be an OR of two things. It turns out that if F is the OR of the inverse of both inputs, which is equivalent to a NAND by De Morgan's Law, the implementation matches the mux. You can also find it by trying to modify the circuit diagram from part A without changing its logical behavior by inserting two inverters between the usages of F, like so:



Then, you can realize that the rightmost gate is equivalent to a NAND by De Morgan's law, and get three identical gates in the positions of the F gates.



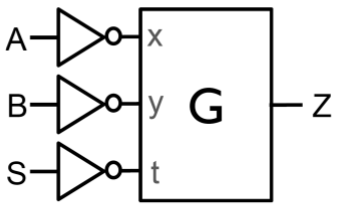
In any case, F is just a NAND gate, so the pulldown corresponds to the AND and has two nFETs in series; the pullup corresponds to the OR and has two pFETs in parallel.



Each F gate takes 4 transistors, and the transistor still needs 2, for a total of 4 + 4 + 4 + 2 = 14 transistors.

**Number of transistors in mux (if F can be built as a CMOS gate): _______14_________**

(C) Consider the implementation shown below, which uses gate G. Find the Boolean expression for G. If G can be built using a single CMOS gate, draw its CMOS implementation. Otherwise, give a convincing explanation for why G cannot be implemented as a CMOS gate. How many transistors does this implementation have?



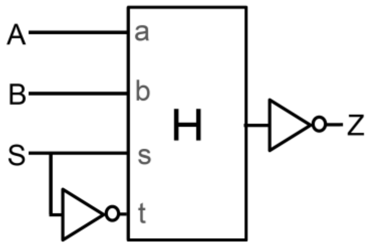
**G(x,y,t) =** ____$\bar{x} \cdot \bar{t} + \bar{y} \cdot t$______

**Draw CMOS gate that implements G or explain why it cannot be built.**

This can be derived by taking the equation $\mathbf{Z = A \cdot S + B \cdot \bar{S}}$ and flipping all inputs, or by drawing a truth table. G cannot be built as a single CMOS gate because it is not inverting: G(1, 0, t) = t, so a rising input (G(1, 0, 0) ⇨ G(1, 0, 1)) causes a rising output.

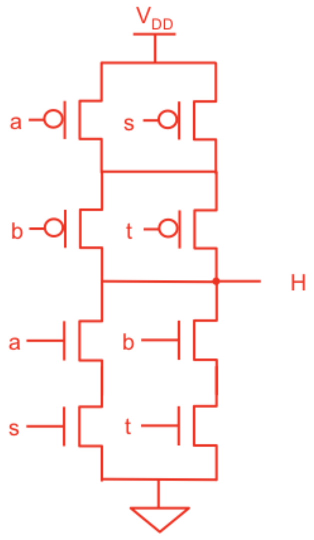
**Number of transistors in mux (if G can be built as a CMOS gate):** ________________

(D) Consider the implementation shown below, which uses gate H. Find the Boolean expression for H. If H can be built using a single CMOS gate, draw its CMOS implementation. Otherwise, give a convincing explanation for why H cannot be implemented as a CMOS gate. How many transistors does this implementation have?



**H(a,b,s,t) =** ____$(\bar{a} + \bar{s})(\bar{b} + \bar{t})$______

**Draw CMOS gate that implements H or explain why it cannot be built.**



**Number of transistors in mux (if H can be built as a CMOS gate):** ______12________

Using the original equation for Z we can write $\mathbf{H = \bar{Z} = \overline{A \cdot S + B \cdot \bar{S}}}$. To make this properly inverting in terms of H's inputs, we can replace $\overline{\mathbf{S}}$ with T to get $\mathbf{H} = \overline{\mathbf{A} \cdot \mathbf{S} + \mathbf{B} \cdot \mathbf{T}}$**, or** $\overline{\mathbf{H}} = \mathbf{A} \cdot \mathbf{S} + \mathbf{B} \cdot \mathbf{T}$**,** which we can translate into the pulldown circuit: (A in series with S) in parallel with (B in series with T). We can take the dual or apply De Morgan's Law three times to get $\mathbf{H} = (\overline{\mathbf{A}} + \overline{\mathbf{S}})(\overline{\mathbf{B}} + \overline{\mathbf{T}})$, which we can translate into the pullup circuit: (A in parallel with S) in series with (B in parallel with T).

H thus takes 8 transistors, 4 for the pullup and 4 for the pulldown. There are also two inverters in this implementation of the mux, which take 2 transistors each. The total number of transistors is 8 + 2 + 2 = 12.