MOSFET-based Logical AND

Learning Objectives

- Construct a type of logical operation to cause an output to operate only when two inputs are simultaneously high.

What is a Logical AND?

A binary number system is much like our decimal number system, except it utilizes only two digits {0,1} instead of ten digits {0, 1, …, 9}. In a binary computer, the logical values of {0,1} are just two different voltage values in a circuit and "math" is achieved through binary operations which, in turn, is just a dedicated circuit manipulating those voltages. A binary operation is often described through a ***truth table***, a table of all possible input combinations and the desired output from each combination. One basic operator is the logical **AND**.

The logical AND for two "one-bit" binary numbers is defined in the truth table of Table 1.

A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

***Table 1**: Desired binary value of output Z given possible binary inputs A and B for a logical AND.*

This table tells us that the output Z will remain low (a logical 0) unless both inputs A and B are both logical 1 in which case Z also becomes a logical 1. Examples of the logical AND "gate" can be seen at https://en.wikipedia.org/wiki/AND_gate.

You may have already learned about the motor-drive circuit that utilizes a single nMOS transistor so that one input signal fed to the gate of the nMOS can control the motor speed. Often, we want to use data signals from multiple sources to achieve more complex motor response. For example, we might want a robotic car to use photo sensors to seek light (for battery recharging) while also using ultrasonic sensors to avoid obstacles. To attain a way in which to combine two signals into a single wheel control, we can leverage the logical AND. Imagine that we have two signals, one from each sensor, and that we are trying to achieve the logical AND operation shown in Table 2 where we are referring to 0's as "low" voltage and 1's as "high" voltage.

While "AND gate" is the typical language used to refer to the implementation of Table 1, the term "gate" is already used in ECE 110 in reference to one of the three terminals of the MOSFET which we will be using to build this device! To avoid confusion, we will refer to this as a "logical AND."

$V_G^{(1)}$	$V_G^{2()}$	left wheel
low	low	off
low	high	off
high	low	off
high	high	on

***Table 2**: Desired relationship between the left-wheel control signal and inputs $V_G^{(1)}$ and $V_G^{(2)}$.*

We can achieve this through a clever mix of the nMOS-based logical AND from Wikipedia and the operation of our motor-drive circuit. This is demonstrated in Figure 1. Note that both nMOS devices must have high gate voltages applied before current can flow freely through the motor.

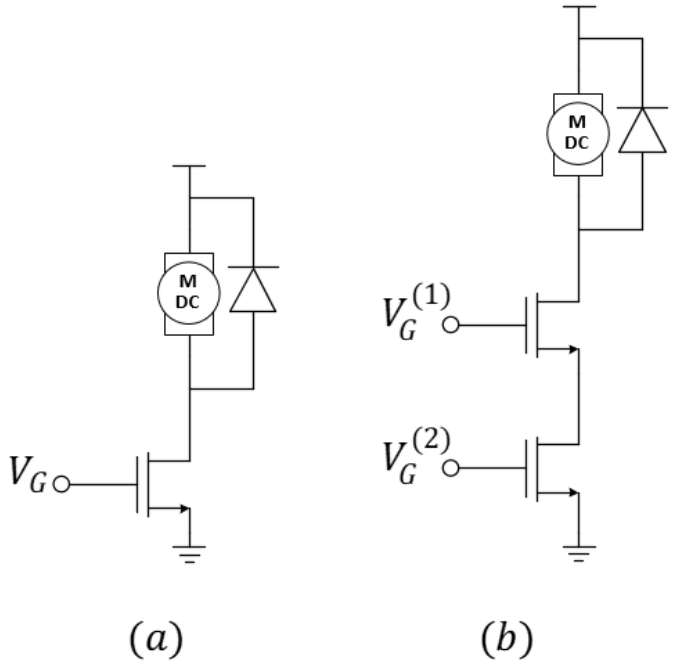


***Figure 1**: (a) nMOS motor-drive circuit and (b) a two-nMOS logical-AND circuit for "joining" two PWM signals.*

Notes:

What's with the diode? The diode across the motor in Figure 1a and 1b is often called a flyback diode. It protects the transistor from large voltage fluctuations that occur when a motor (which has high inductance) is quickly cycled on and off as do with PWM motor-drive circuits. The diode achieves this protection by providing an alternate path for current to flow when the cycles occur. In ECE 110, our motor is small and our nMOS is robust presenting an exception where the diode might be safely neglected.

Notes:

Once we have the basic structure of a nMOS-based logical AND motor drive circuit, we can think of multiple uses. For instance, suppose we have a car that turns slightly to the right because the left wheel runs slightly faster than the left. We could use a PWM signal to slow the left wheel and the "inverse" of that same PWM signal to speed the right wheel to cause the car run straight. This could be done using the traditional single-nMOS motor drive for each wheel of Figure 1a. If we also find that we want to slow down both motors, we could generate another PWM signal fed into a second nMOS transistor on each wheel and reduce the duty cycle to slow both wheels simultaneously (Figure 1b).

To ensure that the two PWM signals work together, we should use two frequencies that are about an "order-of-magnitude" different. This is demonstrated in Figure 2 for one of the wheels showing the PWM signal for the speed control to be a factor of 7 or 8 times larger than the frequency of the PWM used for wheel balance. Using two different frequencies ensures that the lower frequency "modulates" the higher frequency by effectively pulsing it on and off. If you were to attempt to use the same frequency for each, you could run into a situation where the "phase difference" between the two PWM signals *randomly* affects the car operation.

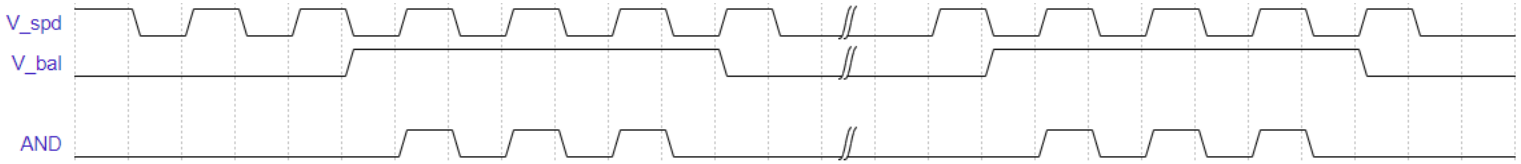


***Figure 2**: Example: Increasing the duty cycle of either V_{bal} or V_{spd} will change the output of the AND to drive the corresponding wheel at a higher speed allowing either input to achieve its intended purpose.*

Procedure

To test our design quickly, we may want to quickly generate binary input signals {0,1}. We'll want the binary 0 to be a voltage near 0 volts (relative to the negative end of our battery) and the binary 1 to be close to the full battery voltage. Figure 3 demonstrates an easy way to achieve a binary signal with a button and a pull-down resistor. We call this a pull-down resistor because it has the effect of pulling down the voltage to 0 volts when the button is not pressed. This is thanks to Ohm's law as zero current through a resistor means zero voltage drop across it!

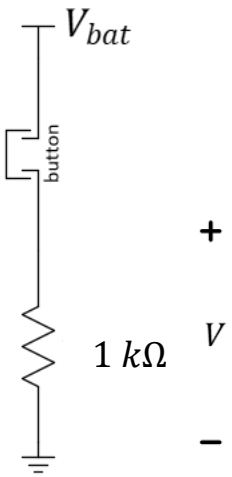


***Figure 3**: An easy circuit to generate a logical input.*

Let's build two of these inputs and replace the motor with a current-limiting resistor and a blue LED as shown in Figure 4. The LED should only illuminate when both buttons are pressed.

Notes:

Notes:

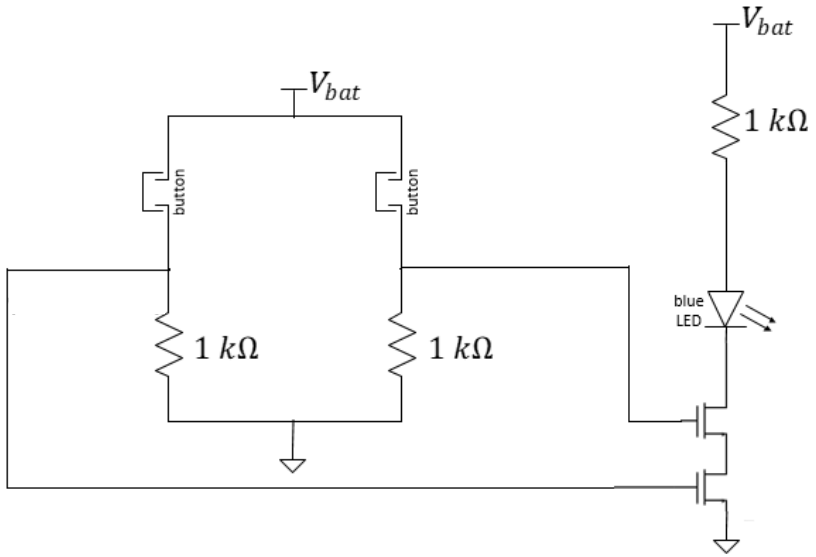


***Figure 4**: Circuit schematic of a two-input logical AND using buttons for inputs.*

Remove the LED and the current-limiting resistor and replace it with one of your motors. Again, check for proper circuit operation. The motor should run only when both buttons are pressed.

Notes:

Explore More!

As discussed earlier, this design is for an advanced motor-control circuit and not just for simple push-button inputs. Consider Figure 5 that includes an adjustable wheel-speed balance potentiometer combined with speed control. You should recognize the familiar motor-drive circuits as well as two oscillators and two copies of the logical AND. You do not need to build this circuit now, but this idea should open a wealth of potential robotics projects!

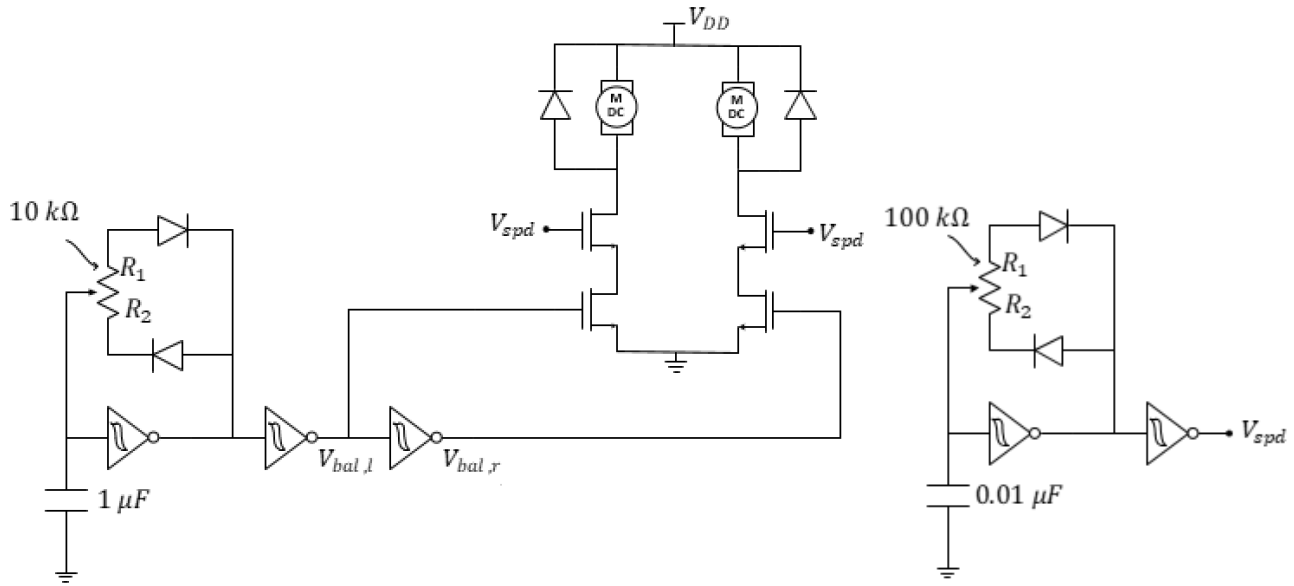


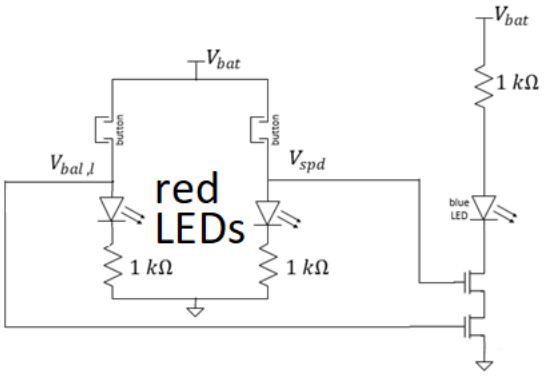
Figure 5: *PWM-based wheel balancer plus speed control. The speed control circuit is drawn separately for clarity but notice that the nodes labeled* V_{spd} *must all be connected.*

Name:_____ UIN: | | | | | | | | | | Section AB/BB: |

Notes:

Question 1: Please submit a video demonstrating the operation of Figure 4.

If you have only a DMM: Add two red LEDs to the button circuits as shown below. They must be ***red***!



Or…If you have an oscilloscope handy: Use your oscilloscope on the voltages of the gates of the two MOSFETs of the figure 4 diagram**.**

In the video, explain how the buttons change the voltage at each gate of the two MOSFETs and the situation in which both MOSFETs allow current flow. If we were to consider (as before) the drain-to-source connection of each MOSFET as a variable resistor, what is happening to that resistance when the gate voltage goes "high?"

Why do the LEDs need to be red? The gate of the MOSFET behaves like a capacitor. For the MOSFET to turn off, that gate capacitor needs to discharge below a certain "threshold" voltage. Of the light-emitting diodes, red has the lowest turn-on voltage (near 2 volts) which should be just below the threshold voltage of the MOSFET's gate. For fun, you can try a blue LED in one of your button circuits. You will see that the blue LED nearly stops conducting current at a voltage higher than 3 volts and the gate capacitor is not capable of fully discharging in a short amount of time.

Notes: