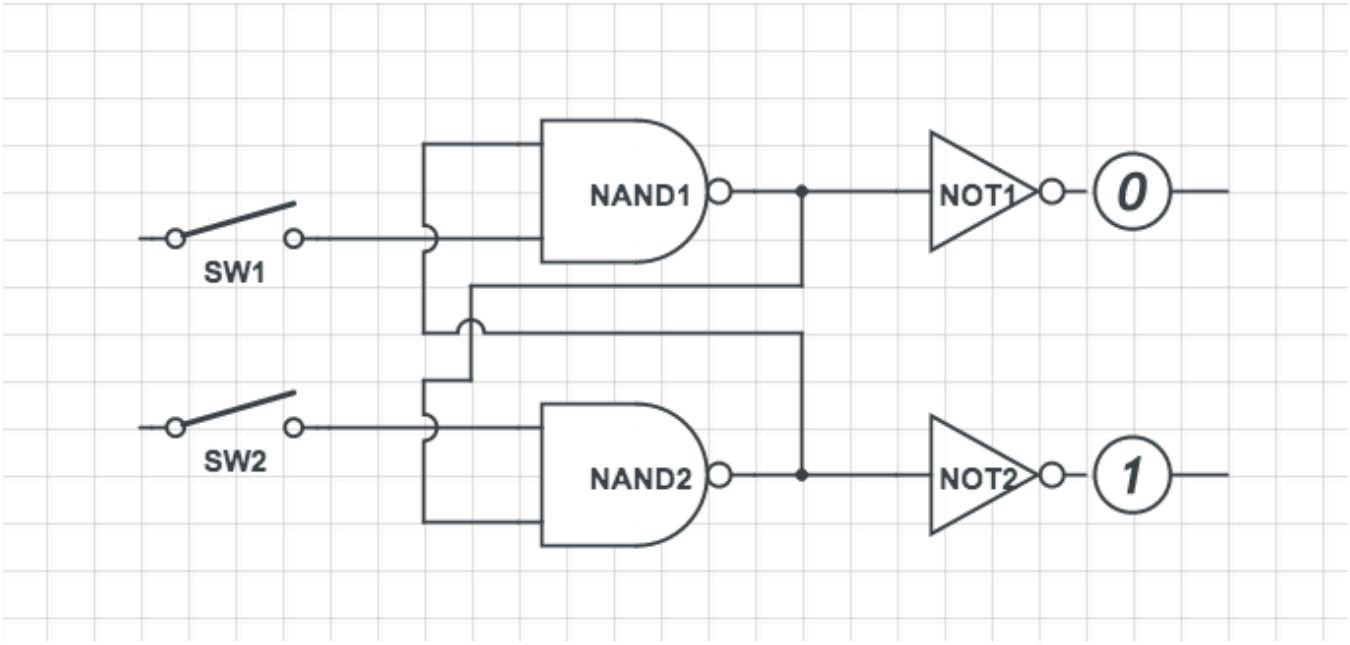
LOCKOUT CIRCUIT EXPLANATION!

This circuit is commonly used in a gameshow like Jeopardy where each contestant has a buzzer to press to answer a question. The person who presses his/her buzzer first gets the first chance to give an answer. The awesome thing about this circuit is that the logic gates make it impossible to cheat. When one person buzzes in, the other contestant's buzzers are disabled. This is especially useful in situations where two people might buzz in at what seems like the same time, but one person is actually nanoseconds first!

The other neat thing about this circuit is that it can be expanded to add as many contestants as you need. Additionally, you can build using either AND gates or NAND gates. We will show you how to do both. The first example shows a NAND gate lockout with 2 contestants, and the second example shows an AND lockout with 3 contestants. Once you understand these, you can follow the pattern for adding in another contestant and build your own version! Here is the schematic for the circuit, you can look at how we built it with bits in the file labeled "2playerNAND"



Each contestant has a switch input and a buzzer, which will go off if they toggle their switch first. When the switch is off it will output a false signal and when it toggled on it will output a true signal.

Path 1: Start with contestant 1's switch, labeled "switch1". It is connected to an input for a NAND gate labeled "NAND1". The second input for NAND1 is the output of the gate labeled NAND2. The output of NAND1 is connected to a branch because it is used twice in the circuit. One output from NAND1 it is connected to an inverter module and then connects to the buzzer. If you look at the truth table for NAND

below you will probably understand why we use the inverter, if we didn't the buzzer would be going off in 3 different scenarios, but we only want it to go off in 1 scenario! The second output of NAND1 is connected to the input for the NAND2 module.

NAND

Input 1	**Input 2**	**Output**
F	F	T
F	T	T
T	F	T
T	T	F

Path 2: Contestant 2's switch is labeled "switch2". Like switch1, this switch is feeding into and a NAND gate (NAND2) and the second input for NAND2 is the non-inverted output from NAND1. We then have another inverter connected to the output of NAND2 that leads to contestant 2's buzzer.

Here is the truth table for an inverter:

Input	**Output**
T	F
F	T

Let's take a closer look at what is happening with the NAND gates when buzzer 1 goes off. Begin with both switches toggled off. For buzzer1 to buzz we want the output of inverter1 to be true, this means that the output from the NAND1 gate must be false before passing through the inverter.
As we can see in the truth table NAND gates only output a false signal when both inputs to the gate are true. In order for the gate to be getting 2 true inputs, the first input to the gate, switch1, must be switched on. or the second input to be true, NAND2 must be sending a true signal to NAND1. As you can see from the truth table, this occurs when NAND2 is receiving at least one false input. If switch 2 is off we can clearly see that NAND2 is getting a false input signal and will output a true signal to NAND1.

What happens when we toggle switch2 on as well? Buzzer2 does not buzz but buzzer1 continues to buzz! We just said that switch2 is now toggles on, so NAND2 must be receiving a false signal from it's other input NAND1 if it doesn't turn buzzer2 on. Because switch1 was already turned on, switch2 was activated after switch1, which means that there was time (whether a nanosecond or five hours) between

switch1 being toggles on and switch2 being toggled on. As we explained before, during this time NAND2 was receiving a false input from switch2 and sending out a true signal. Since NAND1 was getting 2 true signals it was sending a false signal to NAND1. Now NAND2 is constantly receiving one false signal, and as the truth table shows it will output a true signal, which when inverted keeps buzzer 2 off. The false input from NAND1 means that whether switch2 is toggled on or off NAND2 will not be able to set off buzzer2 and contestant 2 is “locked out”. If switch2 were toggled on, the opposite would be true and contestant2’s buzzer would go off, locking contestant 1 out.

Now let’s rebuild this circuit with AND gates instead of NAND gates. Look at the file labeled 3 contestant AND lockout.

Here is the truth table for AND gates:

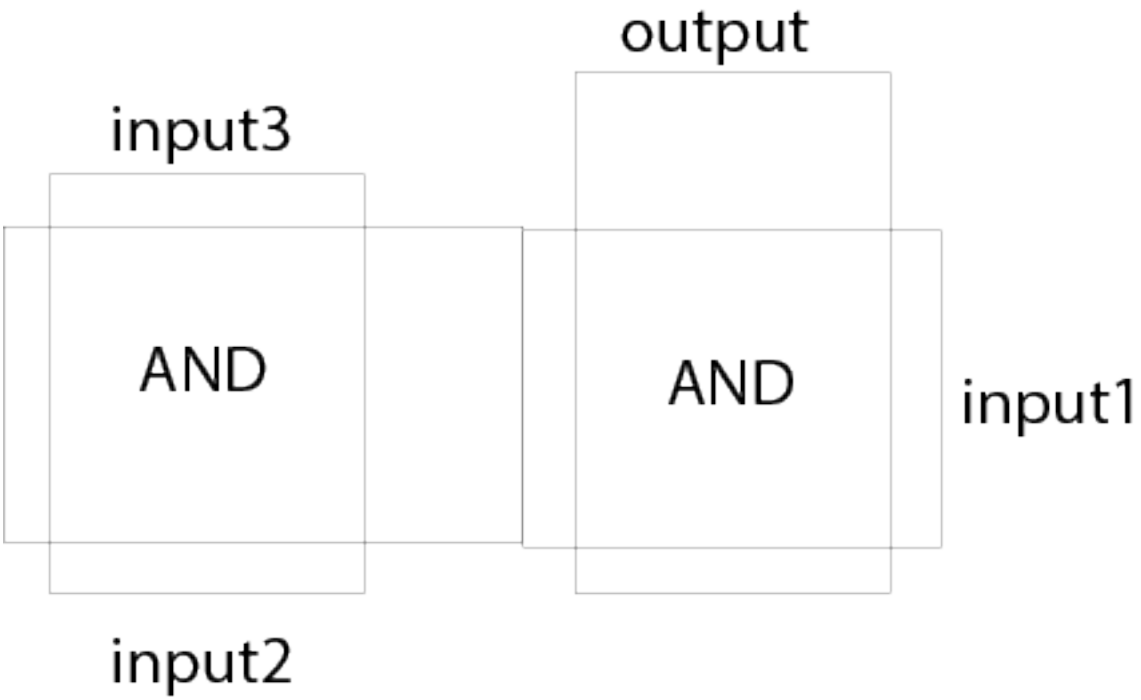
Input 1	**Input 2**	**Output**
F	F	F
F	T	F
T	F	F
T	T	T

As you probably noticed AND is the opposite of NAND. When switch1 is toggled on it will send a true signal to the AND gate labeled AND1. The other input to AND1 is the inverted output from AND2. This means that the output of AND2 is false before going through the inverter and sending a signal to the input of AND1. If switch2 is toggled off then AND2 will be receiving a false input and will always send a false output, that is true after it passes through the inverter. This means that AND1 is receiving 2 true signals (one from switch1 and one from the inverted false output of AND2) and will always output a true signal.

The inputs for AND2 are switch2 and the inverted output of AND1, which based on the logic above would always be a false signal when switch1 is toggle on first. Since AND2 is never able to receive 2 true signals if switch1 is toggled on before switch2, contestant 2 will be “locked out”.

As with the NAND gates, the opposite is true when switch2 is toggled on, and contestant 1 is locked out. And how does the third contestant fit in? Luckily it’s pretty easy to add on to this circuit. We literally double the logic to add one more player, triple it to add 2 players and so on.

Let's start by adding another switch (switch3 in diagram) and AND gate (AND3 in diagram) to the fork that is splitting the power module. Now we have to connect the output from this switch to the other two AND gates, this means that each AND gate will need to have an input from all 3 switches. Since the AND gates only have 2 inputs, we can add a third input by attaching another AND gate to one of the inputs of the AND, thus creating a triple AND, or an AND gate with 3 inputs.
Here is a diagram:



Now we have placed a second AND gate on all of the AND gates from the circuit. Each of these AND gates with be getting 3 inputs: one from the switch, and two inverted signals from the AND gates of each of the other contestants. In order to supply more outputs from each AND gate we have connected branch modules to the outputs of all of the second AND gates. So, for each path there is a switch connected to an AND gate, which receives an inverted signal from another path and outputs the combination of these inputs to a second AND gate.

The logic is the same as it was for the 2 contestant scenario, when one switch is on, it sends a false signal to the other 2 contestant's AND gates disabling them from being able to turn on their buzzers.

Here is a truth table for a triple AND gate where A, B, C are inputs and Y is an output:

Input 1	Input 2	Input 3	Output
F	F	F	F
T	F	F	F
T	T	F	F
F	T	F	F
F	T	T	F
F	F	T	F
T	F	T	F
T	T	T	T

Got it? Now try sketching out a circuit for 4 players.