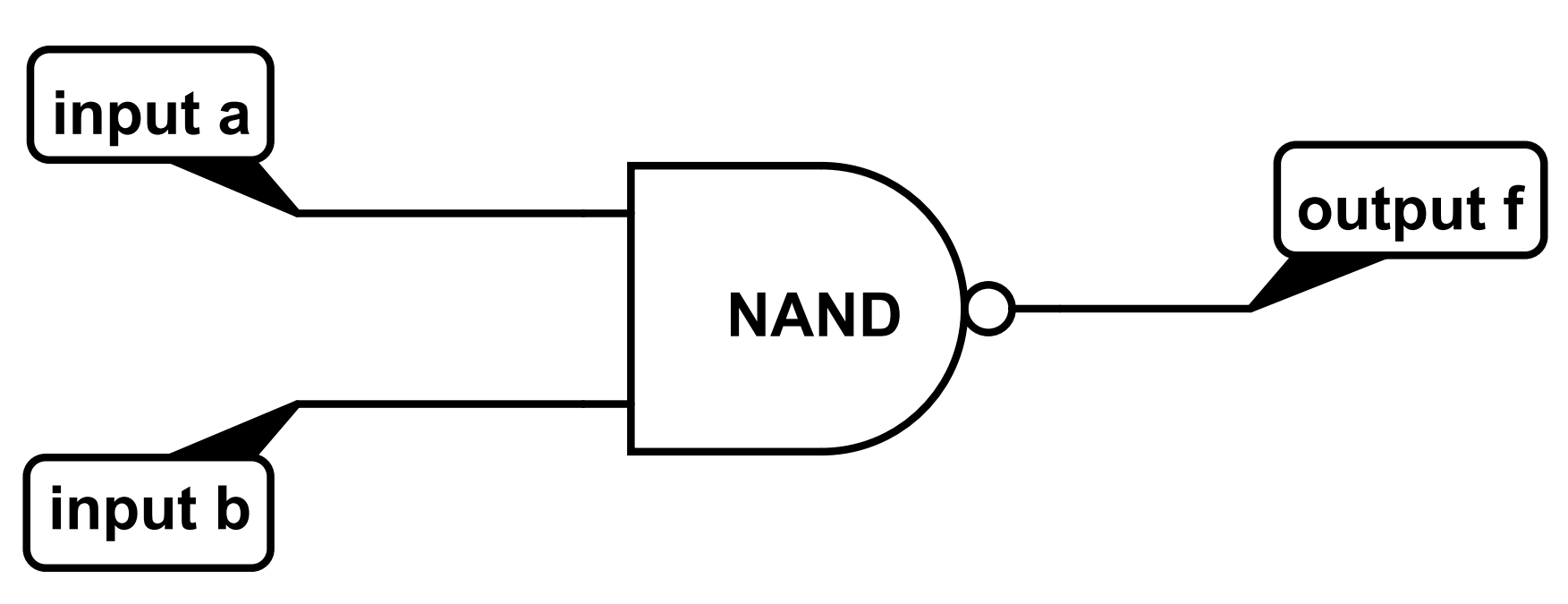
**Lab S2: Special Topic of Electronics – CMOS Logic Gates**

**Part 1: NAND gate – transistor-level circuit, and gate-level IC**

Now let’s explore the CMOS NAND gate. NAND means NOT AND. Here is the circuit symbol for the NAND gate (the big D means AND; the small circular bubble means inversion):



Here is the truth table of the NAND function:

|  |  |  |
| --- | --- | --- |
| Input a | Input b | Output f = a NAND b |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

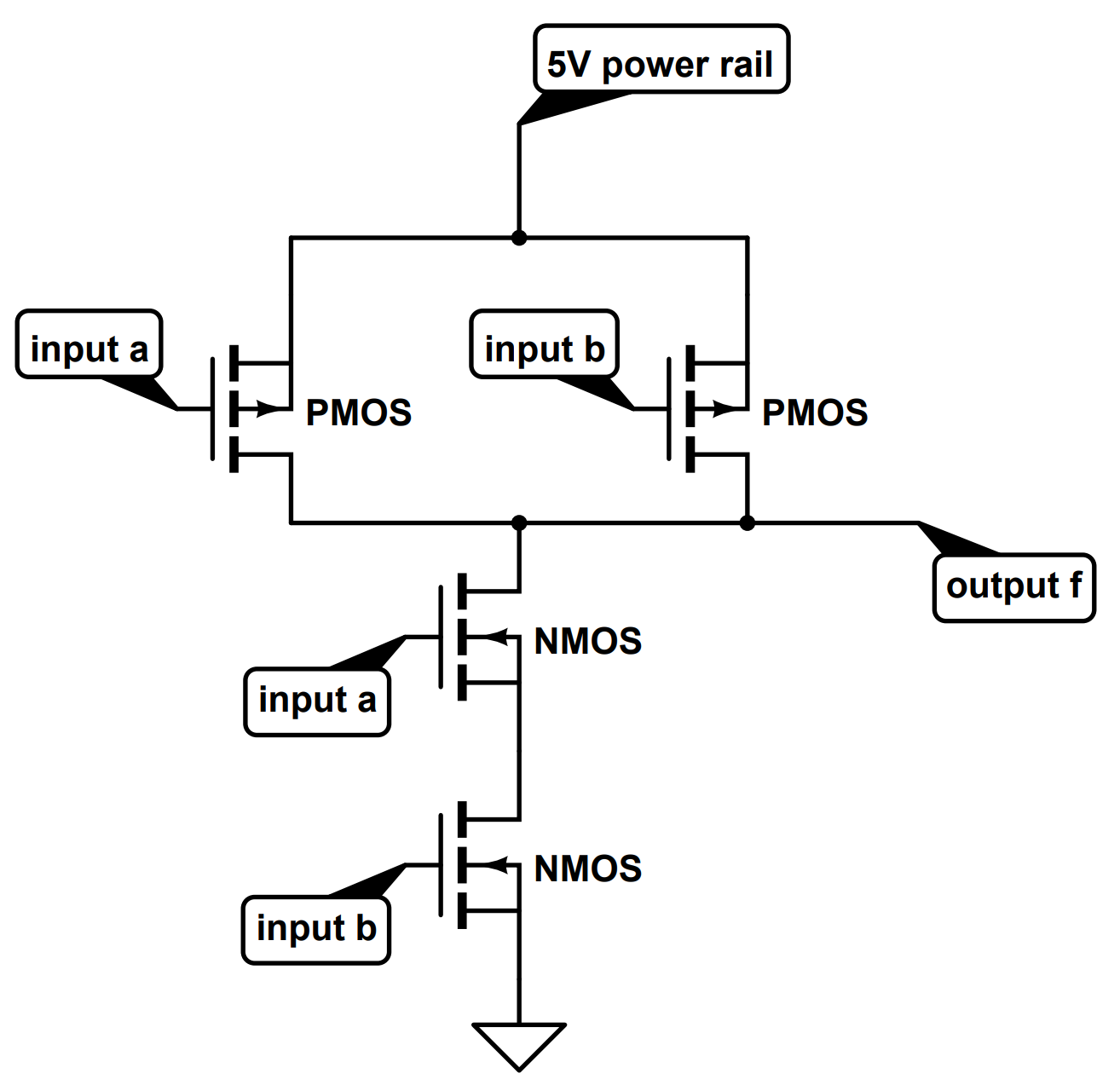
The NAND function is based on the AND function. The AND function is one of the 3 basic logic functions (the other two being OR and NOT). The NAND gate is one of the 2 basic logic gates (the other one being NOR).

AND function truth table:

|  |  |  |
| --- | --- | --- |
| Input a | Input b | Output f = a AND b |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

AND function is output f = a AND b. The output f is 1 only when both inputs (that is, a AND b) are 1. Otherwise, the output f is 0. Once you understand the AND function, you can compare the truth tables of the NAND function and the AND functions to see that NAND is just the inversion of AND.

The basic topology of the CMOS NAND gate circuit is like this:



Build the above transistor-level NAND circuit. You need to modify the above circuit to accomplish the following:

1. Connect the two input “a” nodes together to have only one input “a” node. Implement a sliding switch to toggle the input “a” node between the ground and the 5V power rail.
2. Connect the two input “b” nodes together to have only one input “b” node. Implement another sliding switch to toggle the input “b” node between the ground and the 5V power rail.
3. Use an LED to display the logic value of the output node f. Optionally, use two LEDs to also display the logic values of the two input nodes a, b. The input LEDs and the output LED should have different colors.

This is basically the same circuit organization as the CMOS inverter, but the NAND circuit is more complex than the inverter circuit in that you have now two input nodes (instead of only one input for the inverter), two PMOS in parallel (instead of one PMOS for the inverter), and two NMOS in series (instead of one NMOS for the inverter).

In general, a **CMOS logic gate has the following properties:**

1. The NMOS half-block of the circuit is tied to the lower rail. The PMOS half-block of the circuit is tied to the higher rail. Output is taken at the midpoint where the two NMOS and PMOS half-blocks meet.
2. In a NAND gate, for the NMOS block, the two NMOS transistors are in series, signifying the AND operation. **In general, serial connection produces AND**, because both transistors in series need to be both turned on in order to establish a conduction path. On the other hand, **parallel connection produces OR**, because when either transistor (as well as both transistors) in a parallel configuration is turned on, a conduction path will be established.
3. In a NAND gate, for the PMOS portion, the two PMOS transistors are in parallel, which is the **dual** of the NMOS topology (two NMOS transistors in series). **In general, the PMOS topology must be a dual of the NMOS topology.** Dual means swapping the serial AND topology with the parallel OR topology, or vice versa; and swapping the low-voltage logic 0 with the high-voltage logic 1, or vice versa.

This leads us to formulate the **general rule of using CMOS to construct a logic gate:**

1. First look at the inverse of the function you want to achieve. (For the NAND gate, you want to achieve the NAND operation; so the inverse of NAND is AND).
2. Connect the NMOS half-block to ground. Within the NMOS block, use the NMOS transistors to achieve the inverse of the function you want to achieve. (For the NAND gate, the inverse function of NAND is AND, so you connect the two NMOS transistors in series to achieve the AND). Why the inverse of the function you want to achieve? It is because of the nature of the NMOS transistors – a high-voltage input at the NMOS gate makes the NMOS transistor conductive, thereby connecting the output node at the NMOS drain to the source at the ground node to produce a low-voltage output.
3. Connect the PMOS half-block to the power rail. Within the PMOS block, use the PMOS transistors to construct the dual of the NMOS block. (For the NAND gate, you connect the two PMOS transistors in parallel because the dual of a serial connection is a parallel connection; the dual of AND is OR).
4. The output is taken from the middle where the NMOS and PMOS half-blocks meet. Notice the output is the inverse of the logic function AND within the NMOS half-block. (For the NAND gate, the NMOS transistor connection signifies AND, so the final output is NAND, which is the inverse of AND).

Operate the CMOS NAND gate circuit, and compare the results to the truth table of the NAND function to verify that you have indeed achieved the NAND function with this circuit.

A commercially available quad NAND IC chip (CD4011) has four individual NAND gates in it. The standard pin layout of a quad NAND IC is as follows. Note that pin 7 should be connected to the low-voltage rail Vss (usually the ground), and pin 14 should be connected to the high-voltage rail (VDD), in order to supply power to the four NAND gates on this chip.

Diagram

Description automatically generated

Now use one of the NAND gates of a quad NAND IC chip to make a single NAND gate. This single NAND gate should perform the same logic function as your previously built, transistor-level CMOS NAND gate. Just like your previous CMOS NAND gate, use LEDs to display the output logic value (LED required) and the two input logic values (LEDs optional). Note that when you use the quad NAND chip, the power rail VDD can be any value within 3-15V . Here, we will conveniently use a single 9V battery (and a battery snap) as the power source. Look up the CD4011 quad NAND IC product sheet (posted on Canvas) for the reason **why a 9V battery is suitable as a power source** for this IC chip. A helpful note is that each protective resistor placed in series with an LED should be less than 1k (e.g., 0.1k).

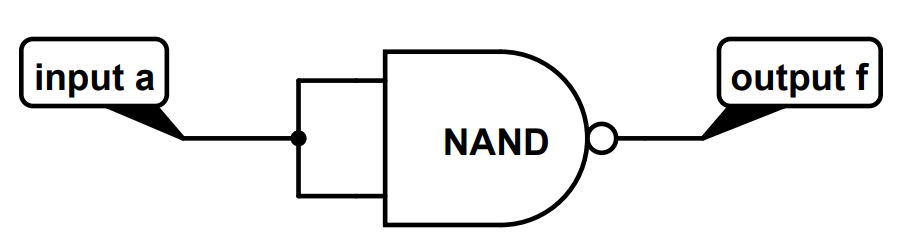
From now on, we are going to forgo using discrete transistors to build transistor-level CMOS logic gates, and use only the quad NAND IC chips and other logic gate IC chips to build all the logic circuits. Compared with a transistor-level CMOS circuit, a readily made quad NAND IC chip is much easy to use, since you are no longer connecting individual PMOS and NMOS transistors.

Can you make a NOT gate from a NAND gate? Yes, you can – by making the two input values equal to each other, which is achieved by connecting the two input nodes a and b together. Refer to the following NAND truth table, which makes the two input values equal, that is: a = b

NAND truth table where the two input values are equal: a = b (thereby producing the NOT function)

|  |  |  |
| --- | --- | --- |
| Input a | Input b | Output f = a NAND b |
| 0 | 0 | 1 |
| 0 | 0 | 1 |
| 1 | 1 | 0 |
| 1 | 1 | 0 |

Can you see that when you simply connect the two inputs together, your NAND gate will become a NOT gate?



Verify this by using a quad NAND gate to form a NOT gate. Use LEDs and protective resistors to display the output value (required) and input value (optional).

**Part 2: AND, OR, and NOR gates**

The NAND gate is called the **universal gate**, because with this type of gate alone, one can build all types of logic gates – NOT, AND, OR, NOR (NOT OR), XOR (Exclusive OR), etc. The entire logic chip to perform a complex logic function may consist exclusively of NAND gates. This NAND-only design is useful because only the NAND process flow is used to make the whole chip. Having a NAND-only design helps to streamline the chip manufacturing process and reduces cost.

**Now use the quad NAND IC chip to form an AND gate.** What is the easiest way to do this? If you simply add an inverter after your NAND gate, you are correct! Use LEDs to display the output logic value (required) and the input logic values (optional), just like you have done before. Operate your AND gate circuit, and use the AND truth table (see above) to verify that you have achieved the AND function.

Diagram

Description automatically generated

The AND gate has the following circuit symbol:

Diagram

Description automatically generated

The OR function has the following circuit symbol:

Diagram

Description automatically generated

Here is the truth table of the OR function:

|  |  |  |
| --- | --- | --- |
| Input a | Input b | Output f = a OR b |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

OR function is output f = a OR b. The output f is 1 when at least one input (that is, a OR b) is 1. Of course, under the same rule, when both inputs are 1, the output must be 1. Only when both inputs are 0 will the output be 0.

Note the **Boolean algebra** notation for AND (dot ∙ ), OR (plus + ), and NOT (prime ’)

a AND b is written as: a ∙ b

a OR b is written as: a + b

NOT a is written as: a’

a NAND b is NOT (a AND b), which is written as: (a ∙ b)’

a NOR b is NOT (a OR b), which is written as: (a + b)’

NOT (NOT a) is: (a’)’, which is a’’, which is simply a (the two NOT operations cancel each other)

**It is important to keep in mind that the regular algebra operation does not readily apply to Boolean algebra.**

A very useful rule of the Boolean algebra is De Morgan’s law:

(a ∙ b)’ = a’ + b’

(a + b)’ = a’ ∙ b’

You can easily verify De Morgan’s law by using truth tables.

Using De Morgan’s law, you can construct OR from NAND:

a + b = (a + b)’’ = ((a + b)’)’ = (a’ ∙ b’)’

Again, you can easily verify this rule by using truth tables. This rule means you can construct a OR b by:

1. Inverting the two inputs a, b to produce a’, b’
2. Feeding the two inverted inputs a’, b’ into a NAND gate

The output will be a OR b. In Boolean logic notation, this is written as: a + b = (a’ ∙ b’)’

Diagram

Description automatically generated

**Now use a quad NAND gate to construct the OR function.** Use LEDs to display the output logic value (required) and the input logic values (optional), just like you have done before. Operate your OR gate circuit, and use the OR truth table (see above) to verify that you have achieved the OR function.

The NOR function has the following circuit symbol:

Diagram

Description automatically generated

Here is the truth table of the NOR function:

|  |  |  |
| --- | --- | --- |
| Input a | Input b | Output f = a NOR b |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

**Now use the quad NAND IC chip to build the NOR function.** I leave it up to your to figure out how to use the NAND gates to achieve the NOR function. Use LEDs to display the output logic value (required) and the input logic values (optional), just like you have done before. Operate your NOR gate circuit, and use the NOR truth table (see above) to verify that you have achieved the NOR function.

You should know that both NAND and NOR gates are universal gates, because with either NAND or NOR alone, you can achieve all the logic functions.

You may wonder why AND and OR are not universal gates. This is because to achieve all the logic functions, you must have the inverting function, which is achievable with NAND and/or NOR, but not with AND and/or OR. Before the invention of transistors, only diodes were used to build logic circuits. These diode logic (DL) gates can achieve the functions made up of AND and OR, but there was no inverting function. After the invention of transistors, the inverting function became possible, and the full set of logic functions was achieved. Nowadays, logic circuits are routinely built with NAND logic or NOR logic.

**Appendix**

**List of circuits – take a picture of each circuit to include in your lab report**

CMOS\_NAND transistor-level circuit

NAND\_quadnand circuit

NOT\_quadnand circuit

AND\_quadnand circuit

OR\_quadnand circuit

NOR\_quadnand circuit

**Items to be included in the lab report appendix section**

Images of breadboard circuits of

1. CMOS NAND gate transistor level circuit
2. Inverter using QuadNAND IC
3. AND gate using QuadNAND IC
4. OR gate using QuadNAND IC
5. NOR gate using QuadNAND IC