

Module #2: Basic Logic Circuits and Functions

Revision: August 24, 2003

Overview

The three primary logic relationships, AND, OR, and NOT (or inversion) can be used to express any logical relationship between any number of variables. These simple logic functions form the basis for all digital electronic devices – from a simple microwave oven controller to a desktop PC. We can write logic equations of the form "F = A AND B" that use these three relationships to specify the behavior of any given digital system. Pause a moment and think about this: *any digital system*, up to and including a highly complex computer system, can be built entirely of devices that do no more than implement these three simple functions.

As engineers, we must address two primary concerns: how to express a given requirement or problem statement in terms of these simple logic relationships; and how to build electronic devices (or circuits) that can be used to implement these relationships in real devices. This lab will begin to explore the second of these questions – how to arrange switching devices so that these relationships are realized.

Before beginning this module, you should...

- Read the section of your text that covers transistors and logic circuits;
- Know the truth-table definitions of AND, OR, NOT, NAND, NOR, XOR, and XNOR (or EQV) logic relationships;
- Be able to apply Ohm's law to circuits that contain just resistors and switches, and understand how node voltages change when switches are opened or closed;
- Know how to construct breadboard circuits.

After completing this module, you should...

- Understand how switching circuits can be used to implement basic logic functions;
- Understand how transistor circuits implement basic logic functions;
- Be able to create a truth table from a worded logic problem;
- Be able to sketch a logic circuit from a logic equation, and be able to read a logic equation from a circuit;
- Be able to create a logic circuit from a truth table definition.

This module requires:

- A Digilab circuit board
- The Digital chip kit

Background

Logic equations are used to show how an output logic signal should be driven in response to changes on one or more input signals. The equal sign (“=”) is typically used as an assignment operator to indicate how information should flow through a logic circuit. For example, the simple logic equation “ $F = A$ ” specifies that the output signal F should be assigned whatever voltage is currently on signal A . Note this does not imply that F and A are the same circuit node – in fact, the use of a logic equation to specify circuit behavior implies that the inputs and outputs (in the case, F and A) are separated by a circuit component. In digital circuits, circuit components act like one-way gates. Thus, the logic equation “ $F = A$ ” dictates that a change on the signal A will result in a change on the signal F , but a change on F will not result in a change on A . Because of this directionality, assignment operators that indicate direction, such as “ $F \leftarrow A$ ”, are often used. Here, we will just assume that signals always flow from input to output, and we will carry on using the equal sign.

Most useful logic equations specify an output signal that is some function of input signals. For example, the logic equation “ $F = A \text{ and } B$ ” specifies a logic circuit whose output will be driven to LHV only when both inputs are driven to LHV. Below are six common logical functions written as conventional logic equations. The AND relationship, $F = AB$, can be written without an operator between the A and B (but more properly, a dot (\cdot) should be placed between the variables to make the relationship clear). The OR relationship uses the plus sign, and the NOT or inversion relationship is shown by placing a bar over the inverted variable or by placing a single quote character after the variable or quantity to be inverted (two possible notations are shown for several relationships).

$$\begin{array}{cccccc}
 F = AB & F = A + B & F=A \text{ XOR } B & F = \overline{A} & F = \overline{A \cdot B} & F = \overline{A + B} \\
 F = A \cdot B & & F=A \oplus B & F = A' & F = (A \cdot B)' & F = (A + B)'
 \end{array}$$

Compound logic expressions can be built from these basic functions. For example, an output might be need to driven to LHV if input signals A and B are both at LHV, or if input C is LLV, or if C is LHV at the same time that A is LLV. This relationship can be concisely written as “ $F = AB + C' + A'C$ ”.

A truth table is the primary tool for capturing logical relationships in a concise and universally understood format. All possible combinations of inputs are shown in rows on the left of a truth table. A truth table with N inputs requires 2^N rows to list all possible input combinations. A ‘0’ or ‘1’ in the rightmost column indicates whether the logical relationship evaluates to a “true” for the combination of inputs shown in the adjacent row. For example, a truth table with two inputs, A and B , will require 2^2 , or 4 rows to list all possible combinations: “0 0”, “0 1”, “1 0”, and “1 1”. For the ANDing operation, the output is “true” only when both inputs are true, so the rightmost column would have a ‘1’ only in the last row. For “ $F = A' \text{ and } B$ ”, the truth table would have a ‘1’ only in the second row.

Problem 1: Complete the truth tables in the submission form for the basic logic functions and logic equations shown.

In engineering, we are interested more in performing actions than in the "truth" of a given relationship. For example, let’s say we have produced a circuit that can turn on an automobile's dashboard warning light whenever the coolant level is too low AND the engine is too hot. If the coolant level ever becomes too low and the engine temperature too high, the circuit’s output is said to be **asserted** to indicate the output signal is ready to do some work (like illuminating a warning light). We likewise

apply the term to inputs – the AND relationship in this example may be stated as "the output F is asserted when the inputs A and B are both asserted".

Some input signals to logic circuits might normally be at LLV, changing to LHV only when some input device or circuit is activated (like the pushbuttons on the Digilab board). Other input signals might normally be at LHV, changing to LLV only when an input device is activated. In either case, we can use the term “asserted” to indicate the input is producing a signal at either LLV or LHV in response to some action. Using this definition, an asserted signal at LHV is said to be *asserted high*, and an asserted signal at LLV is said to be *asserted low*. An *asserted high* signal at LLV is said to be “not asserted”, and an asserted low signal at LHV is said to be “not asserted”. The same signal definitions are also applied to output signals from logic circuits. If a logic circuit produces a LHV when its inputs are asserted, its output is said to be asserted high, and if a circuit produces a LLV at its output, the output is said to be asserted low.

An example of a physical circuit composed of **series** switches and a resistor that can be used to implement the ANDing operation $F = AB$ is shown in the circuit diagram below. Observe that when the switches are both closed, the output F is connected directly to GND, and so F is at LLV. But when one switch or the other is left open, no direct path to GND exists and the output is "pulled high" (to LHV or VDD) by the resistor.

The operation of the circuit is concisely described as follows:

- if both switches are open, then $F = VDD$ (or LHV);
- if one switch is open and the other closed, then $F = VDD$ (or LHV);
- if both switches are closed, then $F = GND$ (or LLV);

If we assume that placing VDD on a switch input (labeled SW1 and SW2) causes it to close, and placing GND or 0V on a switch causes it to open, then we can complete a “voltage truth table” that clearly shows the behavior of the circuit network under all conditions.

Problem 2: Complete the truth tables and answer the questions regarding figure 1.

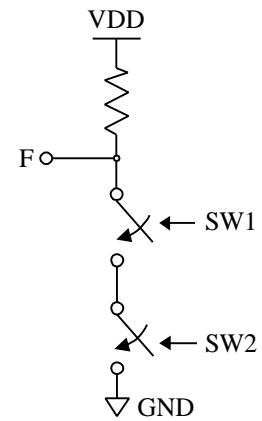


Figure 1.

The circuit of figure 1 can show both the AND relationship and the OR relationship: F is **LLV** if SW1 **and** SW2 are **closed**; and F is **LHV** (or VDD) if SW1 **or** SW2 are **open**. All logic circuits illustrate this property of **duality**, which simply means that any given logic circuit can be interpreted as performing and and'ing relationship or an or'ing relationship, depending on how the inputs and outputs are interpreted.

A second circuit (figure 2) can also be used to implement the function $F = AB$ or $F = A + B$. This circuit uses a **parallel** configuration instead of the series configuration shown in figure 1. Here again, we assume that Vdd closes a switch and 0V opens a switch. This circuit, like that of figure 1, can demonstrate either the AND relationship or the OR relationship depending on how the input and output signals are interpreted - F is LLV (or GND) if SW1 **and** SW2 are open, and F is LHV if SW1 **or** SW2 are closed.

Problem 3: Complete the truth tables and answer the questions regarding figure 2.

Note that in both cases above, the physical circuit *always behaves the same way*, but the behavior can be interpreted as OR-like, or as AND-like. As will be seen in later work, which interpretation is used is a matter of convenience.

Problem 4: State a simple theorem statement for converting between AND-like and OR-like circuit interpretations of a given circuit.

Using just switches and resistors, it is also possible to create logical circuits that perform compound logical relationships, like “ $F = (A \text{ and } B) \text{ or } C$ ”. Such circuits will be discussed in more detail a later section.

Problem 5: Complete the design of a more complex switch/resistor circuit.

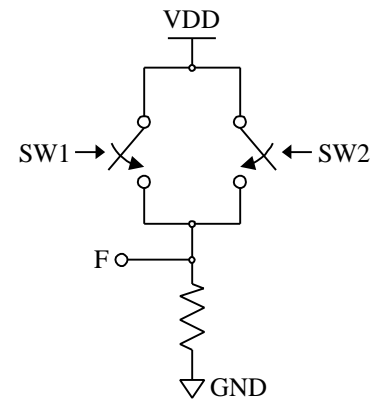
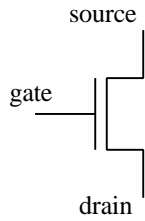


Figure 2.

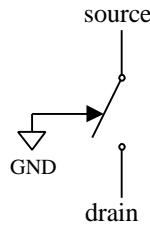
Transistor as switches

Digital electronic circuits are built from electronic switches called transistors instead of the mechanical switches and resistors discussed above. The basic concept is the same – the switches (transistors) are arranged so that they can be turned on or off by signals carrying either LLV or LHV. The transistor switches used in modern digital circuits are called “Metal Oxide Semiconductor Field Effect Transistors”, or MOSFETs (or just FETs). FETs are three terminal devices that can conduct current between two terminals (the **source** and the **drain**) when a third terminal (the **gate**) is driven by an appropriate logic signal. In the simplest FET model (which is appropriate for our use here), the electrical resistance between the source and the drain is a function of the gate-to-source voltage – the higher the gate voltage, the lower the resistance (and therefore, the more current that can flow). In analog circuits (like audio amplifiers), the gate-to-source voltage is allowed to assume any voltage between GND and Vdd; but in digital circuits, the gate-to-source voltage is constrained to be either Vdd or GND (of course, when the gate voltage changes from Vdd to GND or vice-versa, it must necessarily assume voltages between Vdd and GND – we assume that this happens infinitely fast, so that we can ignore FET characteristics during the time the gate voltage is switching).

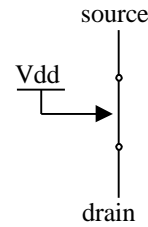
In a simple digital model, FETs can be thought of as electrically controllable "on/off" switches. An electrical connection is created between the source and the drain (i.e., the FET is turned “on”) when the gate input is **asserted**. One kind of FET, called an **nFET**, is turned on when Vdd is present at the control input, and a second type, called a **pFET**, is turned on when GND is present at the control input. Thus, an "asserted" input for an nFET means that the control signal is at Vdd, and for a pFET means the control input is at a GND. The figures below show the circuit symbols and equivalent switch diagrams for both nFETs and pFETs.



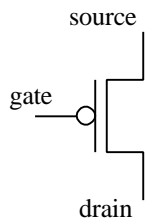
Circuit symbol used for an **nFET** in a schematic drawing



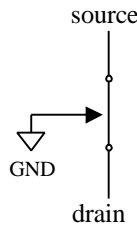
When the gate of an nFET is at GND, the nFET behaves like an **open switch**



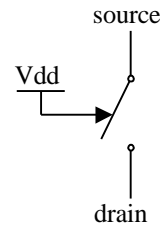
When the gate of an nFET is at Vdd, the nFET behaves like an **closed switch**



Circuit symbol used for a **pFET** in a schematic drawing



When the gate of a pFET is at GND, the pFET behaves like an **closed switch**

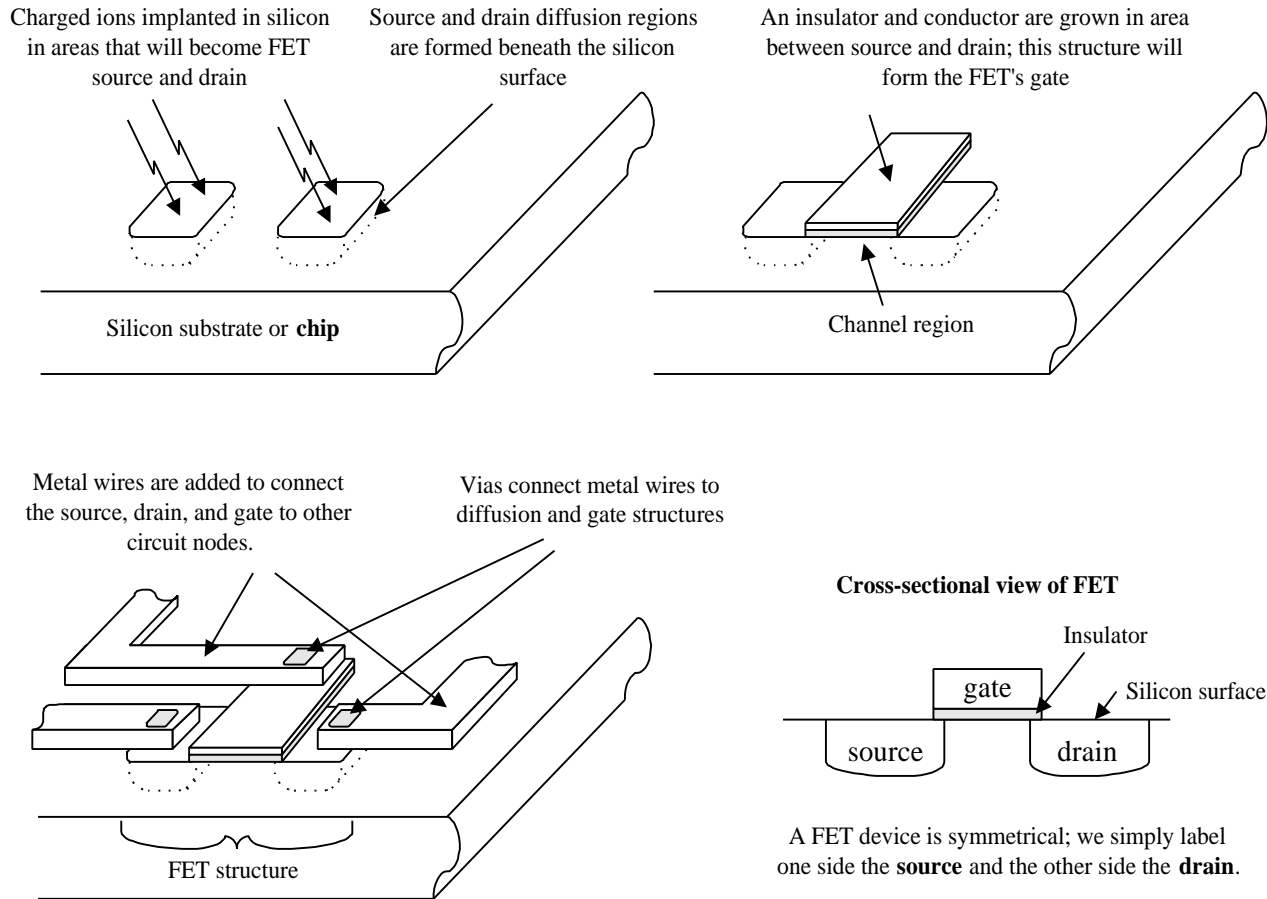


When the gate of a pFET is at Vdd, the pFET behaves like an **open switch**

Individual FETs are often used as stand-alone electrically controllable on-off switches. As an example, if a pFET were used to turn on and off an appliance, then a power source might be connected to the source connection, and a load (such as a motor, lamp, or other electrical component in an appliance) might be connected to the drain connection. A signal applied to the gate could then turn the load device on (gate = GND) or off (gate = Vdd). Typically, a relatively small voltage (on the order of a few volts) is required to turn on a FET, even if the FET is switching large voltages and currents. Individual FETs used for this purpose are typically rather large (macroscopic) devices.

FETs can also be arranged into circuits that perform useful logic functions such as AND, OR, NOT, etc. In this application, several very small FETs are constructed on a single small piece of silicon (or **chip** of silicon) and then interconnected with equally small metal wires. These microscopic FETs typically occupy an area of less than $1 \times 10^{-7} \text{ m}^2$. Since a silicon chip might measure several millimeters on a side, several millions of FETs can be constructed on a single chip. Circuits assembled in this fashion are said to form "integrated circuits" (or **IC**'s), because all circuit components are constructed and integrated on the same piece of silicon.

Most FETs are manufactured using the semi-conductor **silicon**. During manufacturing, a silicon chip is implanted with ions to make it more conductive in the areas that will become the FET source and the drain regions – these regions are commonly called **diffusion** regions. Next, a thin insulating layer is created between these diffusion regions, and another conductor is "grown" on top of this insulator.

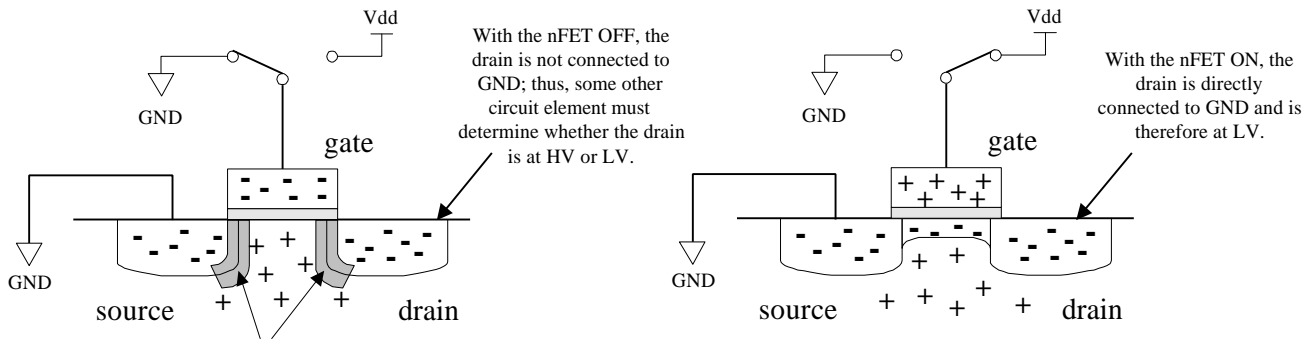


This grown conductor (typically silicon) forms the gate, and the area immediately under the gate and between the diffusion regions is called the **channel**. Finally, wires are connected to the source, drain, and gate structures so that the FET can be connected in a larger circuit. Several processing steps involving high temperatures, precise machine alignments, and various materials are required to produce transistors. Although a description of these processes is beyond the scope of this document, the processes are well documented and many very readable references exist (e.g. see the IBM website <http://www.chips.ibm.com/bluelogic/manufacturing/makechip/makechip1.html>).

The basic principles of FET operation are actually quite straightforward. The following very basic discussion applies only to nFETs; pFET operation is entirely similar, but the voltages must be reversed. Refer to one of the many available texts for a more proper and detailed presentation of FET operation.

As the figure below shows, both the source and drain diffusion areas of an nFET are implanted with negatively charged particles. When an nFET is used in a logic circuit, its source lead is connected to GND, so that the nFET source, like the GND node, has an abundance of negatively charged particles. If the gate voltage of an nFET is at the same voltage as the source lead (i.e., GND), then the presence of the negatively charged particles on the gate repels negatively charged particles from the channel region immediately under the gate (note that in semiconductors such as silicon, positive and negative charges are mobile and can move about the semiconductor lattice under the influence of charged-particle induced electric fields). A net positive charge accumulates under the gate, and two back-to-back positive-negative junctions of charge (called **pn junctions**) are formed. These pn-junctions

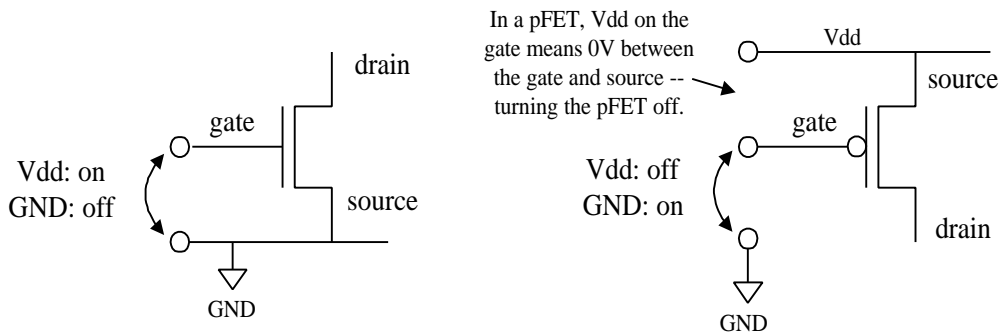
prevent current flow in either direction. If the voltage on the gate is raised above the source voltage by an amount exceeding the **threshold** voltage (or V_{th} , which equals about 0.5V), positive charges begin to accumulate on the gate and positive charges in the channel region immediately under the gate are repelled. A net negative charge accumulates under the gate, forming a **channel** of continuous conductive region in the area under the gate and between the source and drain diffusion areas. When the gate voltage reaches Vdd, a large conductive channel forms and the nFET is “strongly” on.



With the gate held at GND, back-to-back pn junctions are formed, current flow is prevented, and the nFET is **off**

With the gate at Vdd, a conductive channel of negatively charged particles forms under the gate and the nFET is **on**.

As the following figure shows, nFETs used in logic circuits have their source leads attached to GND and Vdd on their gate turns them on, while pFETs have their source leads attached to Vdd and GND on their gate turns them on.



For reasons that will become clear later, an nFET with its source attached to Vdd will not turn on very strongly, so nFET sources are rarely connected to Vdd. Similarly, a pFET with its source attached to GND will not turn on very well either, so pFETs are rarely connected to GND.

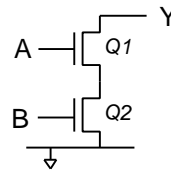
Logic circuits built from FETs

Armed only with this basic description of FET operation, it is possible to construct the basic logic circuits that form the backbone of all digital and computer circuits. These logic circuits will combine one or more input signals to produce an output signal according to the logic function requirements. The following discussion is restricted to circuits for basic logic functions (like AND, OR, and INV), but FET circuits can readily be built for more complex logic circuits as well.

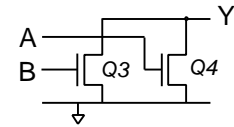
When building FET circuits to implement logic relationships, four basic rules must be followed:

- pFET sources must be connected to Vdd and nFET sources must be connected to GND;
- the circuit output must always be connected to Vdd via an “on” pFET or to GND via an “on” nFET (i.e., the circuit output must never be left floating);
- the logic circuit output must never be connected to both Vdd and GND at the same time (i.e., the circuit output must not be “shorted”);
- the circuit must use the fewest possible number of FETs.

Following these rules, a circuit that can form the AND relationship between two input signals is developed. But first, note that in the circuit on the right, the output (labeled Y) is connected to GND only if the two inputs A and B are at Vdd. The two nFETs labeled Q1 and Q2 are said to be in **series**; in general, a series connection of FETs is required for an AND function. In the circuit on the right below, the output Y is connected to GND if A or B are at Vdd. The two nFETs labeled Q3 and Q4 are said to be in **parallel**; in general, a parallel connection of FETs is required for an OR function.

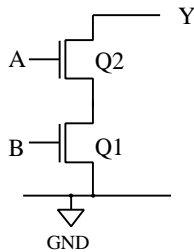


Series configuration:
Y = LLV if A **and** B are LHV

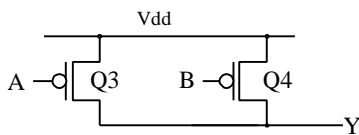


Parallel configuration:
Y = LLV if A **or** B are LHV

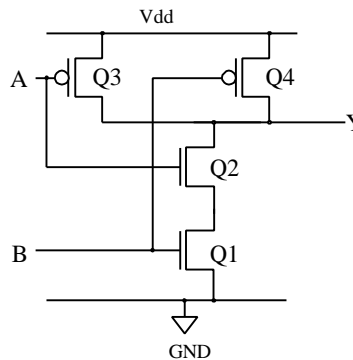
Keeping in mind the rules for FET logic circuits, an AND structure is created from Q1 and Q2 below. Using just these two FETs, Y is driven to GND whenever A and B are not both at Vdd. But we must also ensure the output Y is at Vdd when A and B are not both at Vdd; restated, we must ensure the output Y is at Vdd whenever A or B are at GND. This can be accomplished with an OR'ing structure of pFETs (Q3 and Q4 below). The AND'ing structure and OR'ing structure are assembled in the circuit on the right below. The adjacent operation table shows the input and output voltages for all four possible combinations of inputs. Note that this circuit obeys all the rules above – pFETs are connected only to Vdd, nFETs are connected only to ground, the output is always driven to Vdd or to GND but never to both simultaneously, and the fewest possible number of FETs are used.



Y <= GND when
A and B = Vdd



Y <= Vdd when
A or B = GND



Y <= GND **iff**
A and B = Vdd;
else Y <= GND

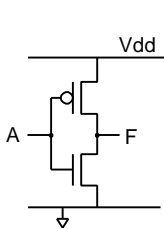
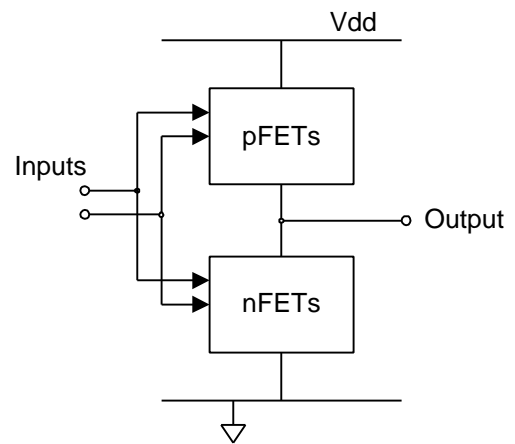
A	B	Y
GND	GND	Vdd
GND	Vdd	Vdd
Vdd	GND	Vdd
Vdd	Vdd	GND

Operation table

This AND'ing circuit has the interesting property of producing an output signal at **GND** when both inputs A and B are at **Vdd**. In order to have this circuit's performance match the AND logical truth table above, we must associate an input signal at Vdd with a logic 1 (and therefore, an input signal at

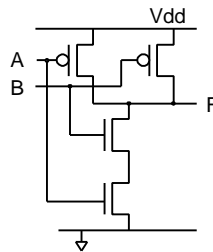
GND must be associated with a logic 0); and we must associate an output signal at GND with a logic 1. This creates a potentially confusing situation – considering the “1” symbol to represent a signal at Vdd on the input of a gate, and then considering that same “1” symbol to represent a signal at GND on the output of a gate. Note that if the outputs in the Y column of the truth table were **inverted** (that is, if Vdd were changed to GND and GND were changed to Vdd), then a “1” symbol could represent Vdd for both the inputs and outputs, resulting in the AND truth table presented earlier. Because of this, the circuit shown above is called a NOT AND gate (where NOT means inversion), which is shortened to “NAND” gate. To create an AND circuit in which both the input signals and output signals can associate a Vdd signal with a logic “1”, an inverter circuit must be added to the output of the NAND gate (as the name implies, an inverter produces a Vdd output for a GND input, and vice-versa). Shown below are the five basic logic circuits: NAND, NOR (for “NOT OR”), AND, OR and INV (for inverter). The reader should verify that all truth tables show the correct circuit operation. These basic logic circuits are frequently referred to as **logic gates**.

In each of these logic gates, a minimum number of FETs has been used to produce the required logic function. Each circuit has nFETs "on the bottom" and pFETs "on the top" performing complementary operations; that is, when an OR relationship is present in the nFETs, an AND relationship is present in the pFETs. FET circuits that exhibit this complementary nature are called Complementary Metal Oxide Semiconductor, or CMOS, circuits. CMOS circuits are by far the dominant circuits used today in digital and computer circuits. (Incidentally, the **Metal-Oxide-Semiconductor** name refers to older technologies where the gate material was made of **metal** and the insulator beneath the gate made of silicon **oxide**). These basic logic circuits form the basis for all digital and computer circuits.



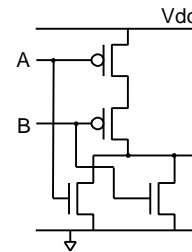
A	F
L	H
H	L

CMOS INVERTER



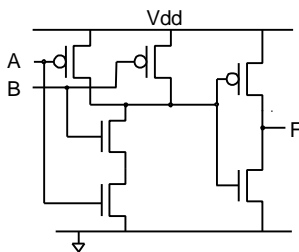
A	B	F
L	L	H
L	H	H
H	L	H
H	H	L

CMOS NAND



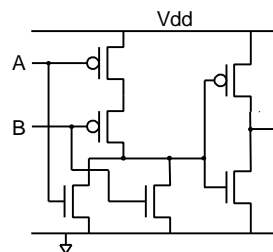
A	B	F
L	L	H
L	H	L
H	L	L
H	H	L

CMOS NOR



A	B	F
L	L	L
L	H	L
H	L	L
H	H	H

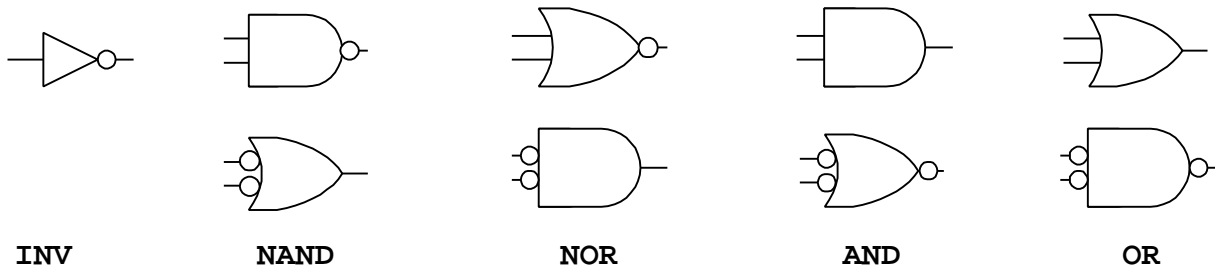
CMOS AND



A	B	F
L	L	L
L	H	H
H	L	H
H	H	H

CMOS OR

When these circuits are used in schematic drawings, the well-known symbols shown below are used rather than the FET circuit diagrams (it would simply be too tedious to draw the FET circuits). A straight edge on the input side of a symbol and smoothly curved output side means AND, while a curved edge on the input side and pointed output side means OR. A bubble on an input means that input must be at LLV to produce the indicated logic function output, and a bubble on the output means that a LLV output signal is produced as a result of the logic function. The lack of a bubble on inputs means that signals must be at LHV to produce the indicated function, and the lack of a bubble on the output means that a LHV signal is produced as a result of the logic function.



Note that each of the symbols above has two appearances. The symbols on the top may be considered the primary symbols, and those on the bottom may be considered the **conjugate** symbols (properly, each symbol is the conjugate of the other). Conjugate symbols swap AND and OR shapes, and input and output assertion levels. The reader should verify that both symbols are appropriate for the underlying CMOS circuit. For example, the AND shaped symbol for the NAND circuit shows that if two inputs A and B are at LVH, then the output is at LLV. The OR shaped symbol for the NAND circuit shows that if either of two input A and B are at LLV, then the output is at LHV. Both statements are true, illustrating that any logic gate can be thought of in conjugate forms. (Why conjugate forms? In certain settings, it can be easier for humans to follow circuit schematics if the appropriate symbol is used – more on this later).

Problem 6: Complete the tables in the submission form to illustrate CMOS circuit behavior.

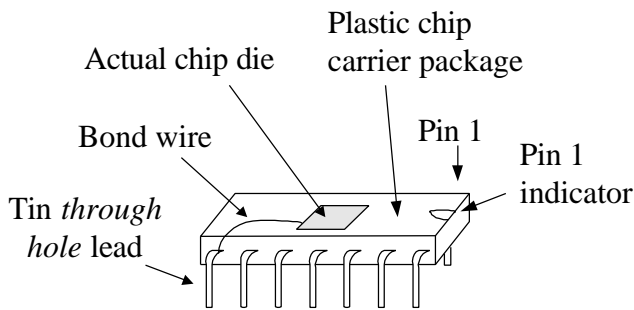
Integrated Circuits (or “chips”)

The terms **chip** and **integrated circuit** refer to FET circuits using microscopic transistors that are all co-located on the same small piece of silicon. Chips have been designed to do all sorts of functions, from very simple and basic logical switching functions to highly complex processing functions. Some chips contain just a handful of transistors, while others contain several million transistors. Some of the longest-surviving chips perform the most basic functions. These chips, denoted with the standard part numbers "74XXX", are simple small-scale integration devices that house small collections of logic circuits. For example, a chip known as a 7400 contains four individual NAND gates, with each input and output available at an external pin.

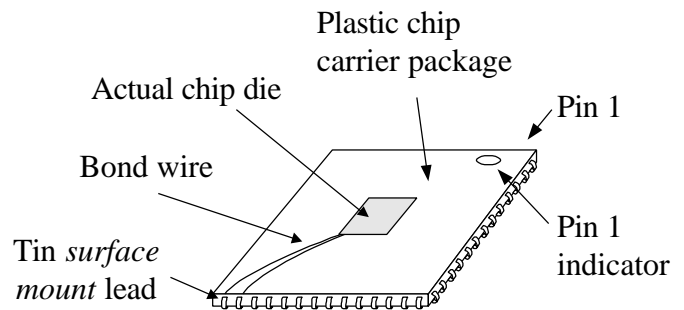
As shown in the figures below, the chips themselves are much smaller than their packages. During manufacturing, the small, fragile chips are glued (using epoxy) onto the bottom half of the package, bond-wires are attached to the chip and to the externally available pins, and then the top half of the chip package is permanently affixed. Smaller chips may only have a few pins, but larger chips can

have more than 500 pins. Since the chips themselves are on the order of a centimeter on each side, very precise and delicate machines are required to mount them in their packages.

Smaller chips are usually packaged in a "DIP" package (DIP is an acronym for Dual In-line Package) as shown below. Typically on the order of 1" x 1/4", DIP packages are most often made from black plastic, and they can have anywhere from 8 to 48 pins protruding in equal numbers from either side. DIPs are used exclusively in through-hole processes. Larger chips use many different packages -- one common package, the "PLCC" (for Plastic Leaded Chip Carrier) is shown below. Since these larger



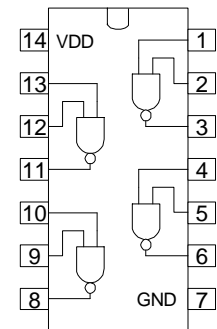
DIP package



PLCC package

packages can have up to several hundred pins, it is often not practical to use the relatively large leads required by through-hole packages. Thus, large chips usually use surface mount packages, where the external pins can be smaller and more densely packed.

Shown on the right is a representation of a 7400 logic IC that contains 16 transistors organized as four 2-input NAND gates. This small chip is housed in a 14-pin DIP package that provides pins for each of the NAND gates inputs and outputs, as well as a power and ground pin (labeled V_{dd} and GND). Note the picture shows the four logic gates placed inside a DIP outline, thereby showing both the function and **pinout** (or pin definition) of the IC. On schematics and on circuit boards, chips are most often shown as square boxes denoted with a "U_" reference designator.



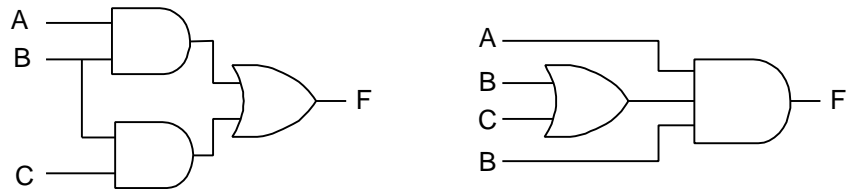
When placing chips in a circuit, **pin 1** must be correctly oriented so that all connections can be properly made. The circuit board silkscreen, IC sockets, and IC's all indicate the location of pin1. For smaller chips and their sockets, a small **notch** is located on one end indicating pin 1 is to the immediate left. By convention, that same notch pattern appears in the circuit board silk screen. For larger IC's, either the corner of the IC nearest (and to the left) of pin 1 is shaved off, or a small indentation (or dot) is located at the corner nearest pin 1.

Logic Circuit Schematics

Digital logic circuits can be built from individual logic chips, or from resources available on larger chips (like the user-programmable Xilinx chip on the Digilab board). Regardless of how logic circuits are implemented, they can be fully specified by truth tables, logic equations, or schematics. This section will present the preparation and reading of logic circuit schematics. A later lab will explore the relationships between circuits and truth tables.

A circuit schematic for any logic equation can be easily created by substituting logic gate symbols for logical operators, and by showing inputs as signal wires arriving at the logic gates. Perhaps the only step requiring some thought is in deciding which logic operation (and therefore, which logic gate) drives the output signal, and which logic operations drive internal circuit nodes. Any confusion can be avoided if parenthesis are used liberally in logic equations to show operator precedence, or if rules of precedence are established. For example, a schematic for the logic equation “ $F = A \cdot B + C \cdot B$ ” might use an OR gate to drive the output signal F, and two AND gates to drive the OR gate inputs, or it might use a three-input AND gate to drive F, with AND inputs coming from the A and B signals directly and a “ $B + C$ ” OR gate.

If no parentheses are used, then NAND/AND has the highest precedence, followed XOR, then NOR/OR, and finally INV. In general, it is easiest to sketch circuits from logic equations if the output gate is drawn first.

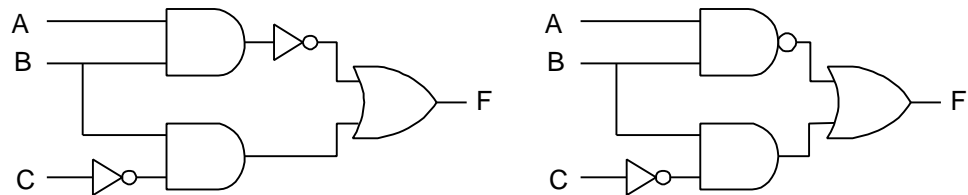


“ $F = A \cdot B + C \cdot B$ ” can be interpreted in two different ways as shown

Inverters can be used in logic equations to show that an input signal must be inverted prior to driving a logic gate. For example, a schematic for “ $F = A' \cdot B + C$ ” would use an inverter on the A input prior to a 2-input AND gate. Equations may also show that the output of a logic function must be inverted – in this case, an inverter

can be used, or the preceding circuit symbol can show an inverted output (i.e., the preceding symbol can show an output bubble).

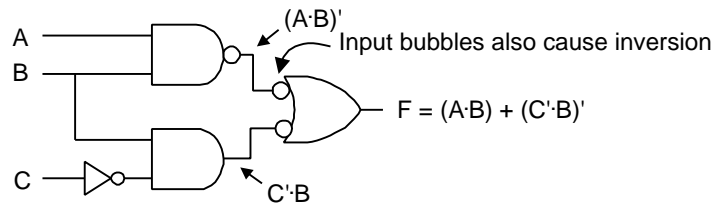
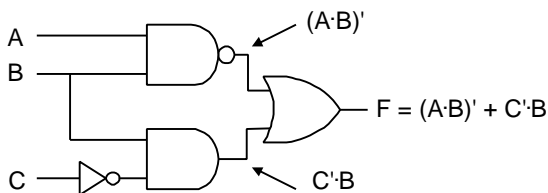
The figure on the right shows an example.



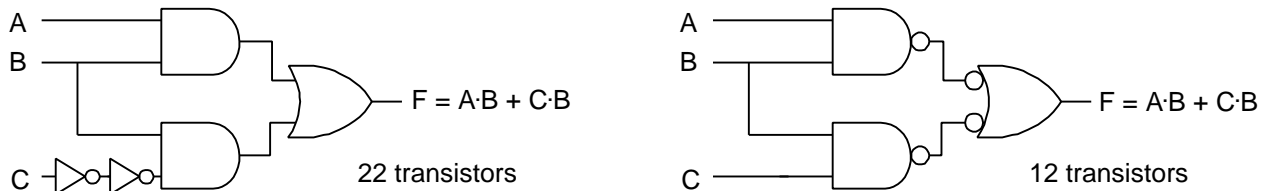
“ $F = (A \cdot B)' + C' \cdot B$ ” can be implemented in two different ways as shown

Problem 7. Sketch circuits for the logic equations in the submission form.

Reading logic equations from schematics is straightforward. The logic gate that drives the output signal defines the “major” logic operation, and it can be used to determine how other terms must be grouped in the equation. An inverter, or an output bubble on a logic gate, requires that the inverted signal or function output be shown in the output of the “downstream” gate (see example below). A bubble on the input of a logic gate can be thought of as an inverter on the signal leading to the gate.



Two “back-to-back” signal inversions cancel each other. That is, if a signal is inverted, and immediately inverted again before it is used anywhere else, then the circuit would perform identically if both inversions were simply removed. This observation can be used to simplify circuits, or to make them more efficient. As an example, consider the circuits below, both of which perform identical logic functions. The circuit on the right has been simplified by removing the two inverters on signal C, and made more efficient by adding inversions on internal nodes so that NAND gates (at four transistors each) could be used instead of AND/OR gates (at six transistors each).



Problem 8. Write logic equations for the circuits shown in the submission form.

Lab Procedure

This lab procedure requires that two simple circuits be constructed on the solderless breadboard, using the logic chips and wires that were introduced in the previous lab. Both designs will be presented as worded problems. You must transfer the requirements in the worded problem first to a formal engineering representation (i.e., a truth table or a Boolean equation), and then build a circuit based on that truth table or equation.

A simple temperature controller (25 points)

Design a circuit to control the temperature of a building. Circuit inputs include a signal A that is driven to a LHV whenever the outside temperature is less than 16°C , a signal B that is driven to a LHV whenever the inside temperature is less than 23°C , and a safety override signal S that is driven to LLV whenever the safety monitor device indicates a fault condition (so the heater should not run when S is at LLV). The circuit must drive a signal F to LHV (to turn on the heater) whenever the outside temperature is less than 16°C or the inside temperature is less than 23°C , provided the safety signal does not indicate a fault.

Problem 9. Write the Boolean equation that defines the required logical relationship (you may want to sketch a truth table to focus your thinking). Then sketch the design for a circuit to implement your temperature control system on the submission form using any logic gates that you desire. All the inputs and the output are active high. (Hint: Can you use three logic gates available in a *single chip* to implement this circuit?).

Problem 10. Using the chips and wires that came with your Digilab kit, implement the circuit. Use switches on the Digilab board to take the place of the three inputs, and use an LED to show when the heater would be turned ON. To construct the circuit, choose the appropriate chips and add them to the solderless breadboards so that they straddle the center groove. Check the IC pin-out diagrams in the previous lab to assist you in choosing which chips to use (i.e., if the circuit requires a 2-input AND'ing operation,

choose an IC that contains 2-input AND's). Once the appropriate chips have been added to the breadboards, connect the Vdd and GND power supplies, and then connect the switch inputs and LED output according to your design. After your circuit is complete, test it by completing the truth table in the submission form, and demonstrate your circuit to the lab assistant.

A three-switch light controller (25 points)

Design a light switch system for a room with three switches located near three doors (the switches are labeled S1, S2, and S3). The circuit must turn the lights ON only if one of the following switch patterns are present: S1 and S2 are ON when S3 is off, or S2 is ON while S1 and S3 are OFF. (Hint: Can you implement this circuit using just one logic gate and one inverter?).

Problem 11. Implement the circuit on the Digilab breadboard using the logic chips and three switches for the light switches, and an LED for the light. Demonstrate your circuit for the lab assistant.