10.1.2 Logic states

By "digital electronics" we mean circuits in which there are only two (usually) states possible at any point, e.g., a transistor that can either be in saturation or be nonconducting. We usually choose to talk about voltages rather than currents, calling a level HIGH or LOW. The two states can represent any of a variety of "bits" (binary digits) of information, such as the following:

one bit of a number;
whether a switch is opened or closed;
whether a signal is present or absent;
whether some analog level is above or below some preset limit;
whether or not some event has happened yet;
whether or not some action should be taken;
and so on.

A. HIGH and LOW

The HIGH and LOW voltage states represent the TRUE and FALSE states of Boolean logic, in some predefined way. If at some point a HIGH voltage represents TRUE, that signal line is called "active HIGH" (or "HIGH true") and vice versa. This can be confusing at first. Figure 10.1 shows an example. SWITCH CLOSED is true when the output is LOW; that's an "active-LOW" (or "LOW-true") signal, and you might label the lead as shown (a bar over a symbol means NOT; that line is HIGH when the switch is *not* closed). Just remember that the presence or absence of the negation bar over the label tells whether the wire is at a LOW or HIGH voltage state when the stated condition (SWITCH CLOSED) is true.[3] At first the idea of active-LOW may seem, well, *backward*; why not just keep it simple and outlaw this upside-down logic? As we'll see, though, there are good reasons to do things "backwards" sometimes. Be patient.

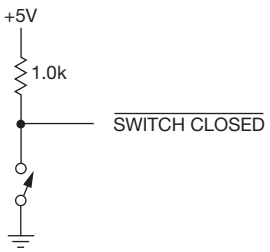

Figure 10.1. A LOW-true ("active-LOW") logic level.

[3] You'll sometimes see the terms "positive-true" and "negative-true" used for HIGH-true and LOW-true, respectively. Those terms are OK, but they can be confusing to the inexperienced, especially since no negative voltages are involved.

A digital circuit "knows" what a signal represents by where it comes from, just as an analog circuit might know what the output of some op-amp represents. However, added flexibility is possible in digital circuits; sometimes the same signal lines are used to carry different kinds of information, or even to send it in different directions, at different times. To do this "multiplexing," additional information must also be sent (address bits, or status bits). You will see many examples of this very useful ability later. For now, imagine that any given circuit is wired up to perform a predetermined function and that it knows what that function is, where its inputs are coming from, and where the outputs are going.

To lend a bit of confusion to a basically simple situation, we introduce 1 and 0. These symbols are used in Boolean logic to mean TRUE and FALSE, respectively, and are sometimes used in electronics in exactly that way. Unfortunately, they are also used in another way, in which 1 = HIGH and 0 = LOW! In this book we try to avoid any ambiguity by using the word HIGH (or the symbol H) and the word LOW (or the symbol L) to represent logic states, a method that is in wide use in the electronics industry. We use 1 and 0 only in situations where there can be no ambiguity.

B. Voltage range of HIGH and LOW

In digital circuitry, the voltage levels corresponding to HIGH and LOW are allowed to fall in some range, according to the particular logic family.[4] For example, with high-speed CMOS ("HC" family) logic running from a +5 V supply, *input* voltages within about 1.5 volts of ground are interpreted as LOW, whereas voltages within 1.5 volts of the +5 V supply are interpreted as HIGH.[5] Those inputs are driven from the outputs of some other devices, for which typical LOW- and HIGH-state *output* voltages are usually within a tenth of a volt of 0 and +5 V, respectively (the output is a saturated transistor switch to one of the rails; see Figure 10.25). This allows for manufacturing spread,

[4] A "family" is a particular hardware implementation of digital logic, characterized by operating voltage, logic voltage levels, and speed. For historical reasons, most logic families name their standard logic parts with a prefix of 74, followed by a few letters that name the family, and ending with some numbers that specify the logic function. The logical functions themselves are the same across families. For example, a 74LVC08 is a 2-input AND gate (actually, four of them in one package) in the low-voltage CMOS (LVC) electrical family; the "08" designates a quad 2-input AND, and the "74" designates a logic chip for operation at standard temperatures.

[5] See the box "Logic Levels" for additional examples.

variations of the circuits with temperature, loading, supply voltage, etc., and the presence of "noise," the miscellaneous garbage that gets added to the signal in its journey through the circuit (from capacitive or inductive coupling, external interference, etc.). The circuit receiving the signal decides if it is HIGH or LOW and acts accordingly.[6] As long as noise does not change 1s to 0s, or vice versa, all is well, and any noise is eliminated at each stage, where "clean" 0s and 1s are regenerated. In that sense, digital electronics is noiseless and perfect.

The term *noise immunity* is used to describe the maximum noise level that can be added to logic levels (in the worst case) while still maintaining error-free operation. As an example, the formerly popular family of logic known as "TTL" (transistor–transistor logic) struggled with just this issue, because it has just 0.4 V of noise immunity: a TTL *input* is guaranteed to interpret anything less than +0.8 V as LOW and anything greater than +2.0 V as HIGH, whereas the worst-case *output* levels are +0.4 V and +2.4 V, respectively (see the accompanying box on logic levels). In practice, noise immunity will be better than the worst-case 0.4 V margin, with typical LOW and HIGH voltages of +0.2 V and +3.4 V and an input decision threshold near +1.3 V. But always remember that if you are doing good circuit design, you use worst-case values. It is worth keeping in mind that different logic families have different amounts of noise immunity. CMOS has greater voltage-noise immunity than TTL, whereas the speedy ECL (emitter-coupled logic) family has less. Of course, susceptibility to noise in a digital system depends also on the amplitude of noise that is present, which in turn depends on factors such as output-stage stiffness, inductance in the ground leads, the existence of long "bus" lines, and output slew rates during logic transitions (which produce transient currents, and therefore voltage spikes on the ground line, because of capacitive loading). We will worry about some of these problems in Chapter 12 (Logic Interfacing).

10.1.3 Number codes

Most of the conditions we listed earlier that can be represented by a digital level are self-explanatory. How a digital level can represent part of a number is a more involved, and very interesting, question. Put another way, we've seen *bits as indicators*; now we will see *groups of bits as a number*.

A decimal (base-10) number is simply a string of integers that are understood to multiply successive powers of 10, the individual products then being added together. For instance,

$$137.06 = 1\times10^2 + 3\times10^1 + 7\times10^0 + 0\times10^{-1} + 6\times10^{-2}.$$

Ten symbols (0 through 9) are needed, and the power of 10 each multiplies is determined by its position relative to the decimal point. If we want to represent a number using two symbols only (0 and 1), we use the *binary*, or base-2, number system. Each 1 or 0 then multiplies a successive power of 2. For instance,

$$\begin{aligned} 1101_2 &= 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 \\ &= 13_{10}. \end{aligned}$$

The individual 1s and 0s are called "bits" (binary digits). The subscript (always given in base 10) tells what number system we are using, and often it is essential in order to avoid confusion, since the symbols all look the same.

We convert a number from binary to decimal by the method just described. To convert the other way, we keep dividing the number by 2 and write down the remainders. To convert 13_{10} to binary, therefore:

$$\begin{aligned} 13/2 &= 6 \quad \text{remainder} \quad 1, \\ 6/2 &= 3 \quad \text{remainder} \quad 0, \\ 3/2 &= 1 \quad \text{remainder} \quad 1, \\ 1/2 &= 0 \quad \text{remainder} \quad 1, \end{aligned}$$

from which $13_{10} = 1101_2$. Note that the answer comes out in the order LSB (least significant bit) to MSB (most significant bit).

A. Hexadecimal ("hex") representation

The binary-number representation is the natural choice for two-state systems (although it is not the only way; we'll see some others soon). Because the numbers tend to get rather long, it is common to write them in hexadecimal (base-16) representation: each position represents successive powers of 16, with each hex symbol having a value from 0 to 15 (the symbols A–F are assigned to the values 10–15). To write a binary number in hexadecimal, just group it in 4-bit groups, beginning with the LSB, and write the hexadecimal equivalent of each group:

$$707_{10} = 1011000011_2\ (= 10\ 1100\ 0011_2) = 2C3_{16}.$$

[6] Sometimes digital signals are sent as a differential voltage pair, rather than "single-ended." This is particularly popular with longer high-speed signals, or signals that go off-board over some distance, for example fast serial buses such as USB, Firewire, and SATA; it is also commonly used to distribute high-frequency clock signals. A popular format is "LVDS" (low-voltage differential signaling), in which the differential signal amplitude is ∼0.3 V, centered at +1.25 V.

LOGIC LEVELS

The diagram in Figure 10.2 shows the ranges of voltages that correspond to the two logic states (HIGH and LOW) for popular families of digital logic. For each logic family it is necessary to specify legal values of both output and input voltages corresponding to the two states HIGH and LOW. The shaded areas above the line show the specified range of output voltages within which a logic LOW or HIGH is guaranteed to fall, with the pair of arrows indicating typical output values (LOW, HIGH) encountered in practice. The shaded areas below the line show the range of input voltages guaranteed to be interpreted as LOW or HIGH, with the arrow indicating the typical *logic threshold* voltage, i.e., the dividing line between LOW and HIGH. In all cases a logic HIGH is more positive than a logic LOW. Table 10.1 and Figure 10.26 provide additional information about these families, a subject we'll see in more detail in Chapter 12.

The meanings of "minimum," "typical,"and "maximum," in electronic specifications are worth a few words of explanation. Most simply, the manufacturer guarantees that the components will fall in the range minimum-to-maximum, with many close to "typical." What this means is that typical specifications are an approximate guide to use when designing circuits; however, those circuits must work properly over the whole range of specifications from minimum to maximum (the extremes of manufacturing variability). In particular, a well-designed circuit must function under the worst possible combination of minimum and maximum values. This is known as *worst-case design*, and it is essential for any instrument produced from off-the-shelf (i.e., not specially selected) components.

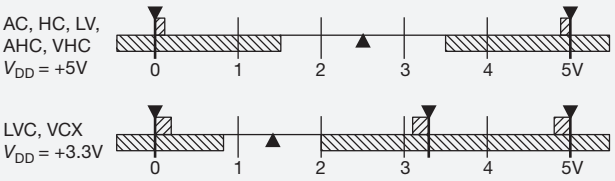

Figure 10.2. Logic levels of some popular logic families.

Table 10.1 Selected Logic Families

Family	Manufacturers	Supply voltage (V_{cc}) min (V)	Supply voltage (V_{cc}) max (V)	V_{in} max	t_{pd} @ Vcc (ns, typ)	t_{pd} @ Vcc (V)	Available packages DIP	SMT	1G,2G
74HC00	9	2	6	V_{cc}	9	5	●	●	●
74AC00	5	3	6	V_{cc}	6	5	●	●	-
74AHC00	2	2	5.5	V_{cc}	3.7	5	●	●	●
74LV00	2	1.2[a]	5.5	5.5	3.6	5	●	●	●
74LVC00	4	1.7	3.6	5.5	3.5	3.3[b]	-	●	●
74ALVC00	3	1.7	3.6	3.6	2	3.3[c]	-	●	-
74AUC00	1	0.8	2.7	3.6	0.9	1.8	-	●	●

Notes: (a) NXP only (TI specs to 2V only). (b) 6ns @ 1.8V. (c) 2.7ns @ 1.8V.

Hexadecimal representation[7] is well suited to the popular "byte" (8-bit) organization of computers, which are most often organized as 16, 32, or 64-bit computer "words"; a word is then 2, 4, or 8 bytes. So in hexadecimal, each byte is 2 hex digits, a 16-bit word is 4 hex digits, etc. For example, the memory locations in a microcontroller with 65,536 ("64K") bytes of memory can be identified by a 2-byte address, because 2^{16}=65,536; the lowest address in hex is 0000h (the trailing "h" means hex), the highest address is FFFFh, the second half of memory begins at 8000h, and the fourth quarter of memory begins at C000h.

A byte sitting somewhere in computer memory can represent an integer number, or part of a number. But it can also represent other things: for example, an alphanumeric character (letter, number, or symbol) is commonly represented as one byte. In the widely used ASCII representation (more in §14.7.8), lowercase "a" is represented as ASCII value 01100001 (61h), "b" is 62h, etc. Thus the word "nerd" could be stored in a pair of 16-bit words whose hex values are 6E65h and 7264h.

B. Binary-coded decimal

Another way to represent a number is to encode each decimal digit into binary. This is called BCD (binary-coded decimal), and it requires a 4-bit group for each digit. For instance,

$$137_{10} = 000100110111 \quad (\text{BCD}).$$

[7] Alternative notations for a hexadecimal number (like $2C3_{16}$) are 2C3H, 2C3h, 2c3h, and 0x2C3.