Technology Brief 8
Digital and Analog

Most of electrical engineering depends on the manipulation of voltages and currents. The real world interfaces with our circuits through sensors (such as the resistive sensors in Technology Brief 4), and we interface back to the real world through user interfaces (such as turning on an LED in Technology Brief 5). In between these ***transducers*** are *circuits*! In the physical world, most signals of interest are ***analog signals***; that is, they vary continuously with time and can take on any value between their possible minimum and maximum values. When electrical sensors transduce these signals into changes in voltage or current, the electrical signals produced are thus also analog. Analog electrical signals can be transduced from sound (using a microphone), mechanical vibration (using a piezoelectric vibration sensor), light or images (using sensor arrays in a camera), temperature (using a thermistor), and many other sources.

All of the circuits we have examined so far are analog circuits. The voltages (and currents) present in these circuits can take on any value between a maximum and a minimum (typically set by the power source). By contrast, a ***digital*** signal can only assume a few discrete values. Most digital systems are binary, which is to say they can only assume two such values, usually called "0" and "1" (alternatively, "on" and "off"). The exact voltages which represent the two logic states depend on the type of digital logic used; for example, many modern digital processors represent "0" with 0 V and "1" with 1.2 V.

Because any single digital line can only assume two values, many such lines can be used to represent a range of numbers. Consider, for example, **Table TT8-1**: three digital lines (or ***bits***) are used to encode 8 different numbers within a given range. In the same way that base-10 numbers can encode 10^N different values with N discrete numbers in the range 0–9 (e.g., two base 10 numbers can encode 0–99), 2^N different values can be encoded by N binary bits. Eight such bits make up a ***byte*** (e.g., the value 01101111 is a byte). Two bytes (16 bits) are a ***word***. Standard encoding schemes exist for representing commonly used data. For example, letters, carriage returns, and other typographics can be represented using the 7-bit American Standard Code for Information Interchange (***ASCII***, pronounced "ask-ee"). **Table TT8-2** gives these codes for capital letters. Many such standards exist (ranging from the data encoding format for, say, Blu-ray data to data transmission across ATM networks).

When representing floating point numbers (such as -2.3), the computer must encode the sign (-1), the number and the exponent. The precision to which a number can be represented depends on how many bits are used. Four words (32 bits) are considered single precision, and 64 bits are double precision. The first bit is the sign (1 = negative), and the next 8 bits are the exponent. The remaining 23 bits (single precision) or 55 bits (double precision) are used to represent the number. This means that the floating point representation of the number has a certain predictable ***round-off error***, and when the computer adds, subtracts, multiplies, etc., this error is also present in the calculations. Usually it is too small to be noticed, but in some cases ($2 - 1.9999\ldots \neq 0$) it can cause unexpected problems in computer programs.

We commonly convert back and forth between analog and digital voltages. Almost all analog signals are converted to digital signals for storage (e.g., images), wireless transmission (your voice in a cell phone call), and performing mathematical functions (in your calculator). This is done with an ***analog-to-digital converter*** (ADC). Sometimes the digital signal must be converted back to

Table TT8-1: Three-bit counting scheme.

Bits		Number
000	=	0
001	=	1
010	=	2
011	=	3
100	=	4
101	=	5
110	=	6
111	=	7

Table TT8-2: ASCII characters for capital letters.

0100 0001 = A	0100 1110 = N
0100 0010 = B	0101 1111 = O
0100 0011 = C	0101 0000 = P
0100 0100 = D	0101 0001 = Q
0100 0101 = E	0101 0010 = R
0100 0110 = F	0101 0011 = S
0100 0111 = G	0101 0100 = T
0100 1000 = H	0101 0101 = U
0100 1001 = I	0101 0110 = V
0100 1010 = J	0101 0111 = W
0100 1011 = K	0101 1000 = X
0100 1100 = L	0101 1001 = Y
0100 1101 = M	0101 1010 = Z

analog (so your friend can hear your voice on his cell phone). This is done with a ***digital-to-analog converter*** (DAC).

The analog voltage in **Fig. TF8-1** can be converted to digital using an ADC to sample it, find the closest step that matches the signal, and convert the value of that step to a digital value. The number of steps (controlled by the number of bits in the ADC) controls the precision of the ADC. **Figure TF8-1** shows a very coarse 3-bit ADC that can represent 8 levels. The difference between the actual analog signal and the level that can be represented with the ADC is called the ***quantization error***.

One of the strengths of digital representations of data is that manipulations of this data (mathematical operations, storage, etc.) can be carried out efficiently with switching networks. These are circuits of components wherein each component can only produce one of two voltage values. Transistors, in particular MOSFETS (see Chapter 4), are particularly well-suited to act as switches in these circuits; modern integrated circuits contain on the order of a billion MOSFETS arranged into circuits to manipulate digital data. Importantly, most modern integrated circuits contain both analog and digital circuits and are known as ***mixed-signal circuits*** (see Section 13-9). Using built-in ADC and

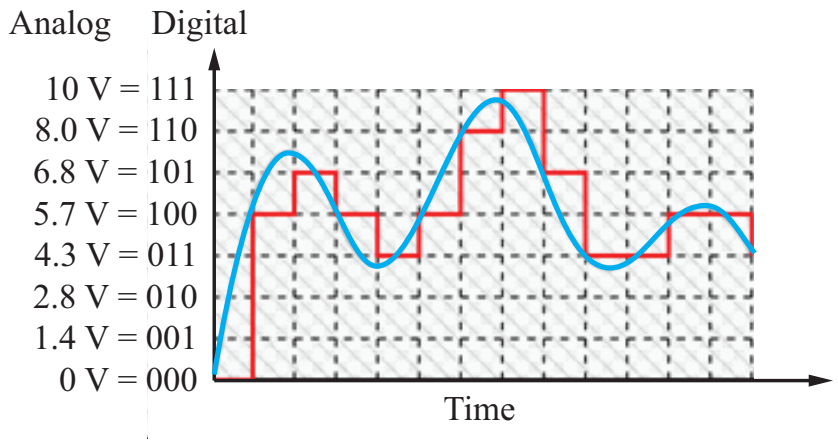
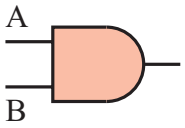


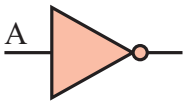
Figure TF8-1: Three-bit digital representation of a continuous signal.

AND



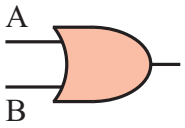
Input A	Input B	Output A and B
0	0	0
0	1	0
1	0	0
1	1	1

NOT



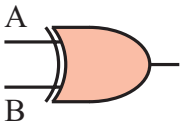
Input A	Output Not A
0	1
1	0

OR



Input A	Input B	Output A or B
0	0	0
0	1	1
1	0	1
1	1	1

XOR



Input A	Input B	Output A xor B
0	0	0
0	1	1
1	0	1
1	1	0

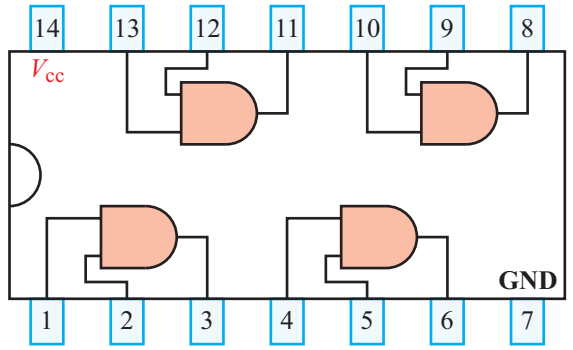
Figure TF8-2: Logic gates.

DAC circuits, data is moved from the analog to the digital domain within a single chip.

But sometimes we use only a few gates for simple control operations or prototyping. Each gate takes two digital signals (which can be either a 0 or 1) as input, and outputs a different digital signal (based on these inputs). **Figure TF8-2** shows a few of these common logic gates. An ***AND*** gate outputs a 1 if both input A AND input B are 1. An ***OR*** gate outputs a 1 if either input A OR input B are 1. A ***NOT*** gate outputs a 1 if input A is NOT a 1; i.e., it inverts the input. An exclusive OR gate, called an ***XOR*** gate, outputs a 1 if one and only one of input A OR input B is 1.

One way to prototype with logic gates is to use a chip that plugs into your protoboard (see Appendix F). **Figure TF8-3** shows an example of a quad AND package. Each pin on the chip is numbered 1–14 and plugs into a separate node (row) on the protoboard. Logic gates are active devices, which means they require an external power supply, so V_{cc} is plugged into pin 14, and GND into pin 7.

Interfacing from the real world to a computer most often involves an analog sensor (such as a thermistor for measuring temperature), a level-shifter (amplifier or de-amplifier or comparator that converts the analog output voltage to digital levels), and then a logic circuit to act upon the output (turn a switch to a heater on or off, for instance). When interfacing back to the real world, the digital signal may be applied in digital form, or may need to be converted back to an analog signal (to drive speakers for voice and music, or precision control of an engine air intake, for example).



7408 Quad 2 Input AND

Figure TF8-3: Quad AND package.