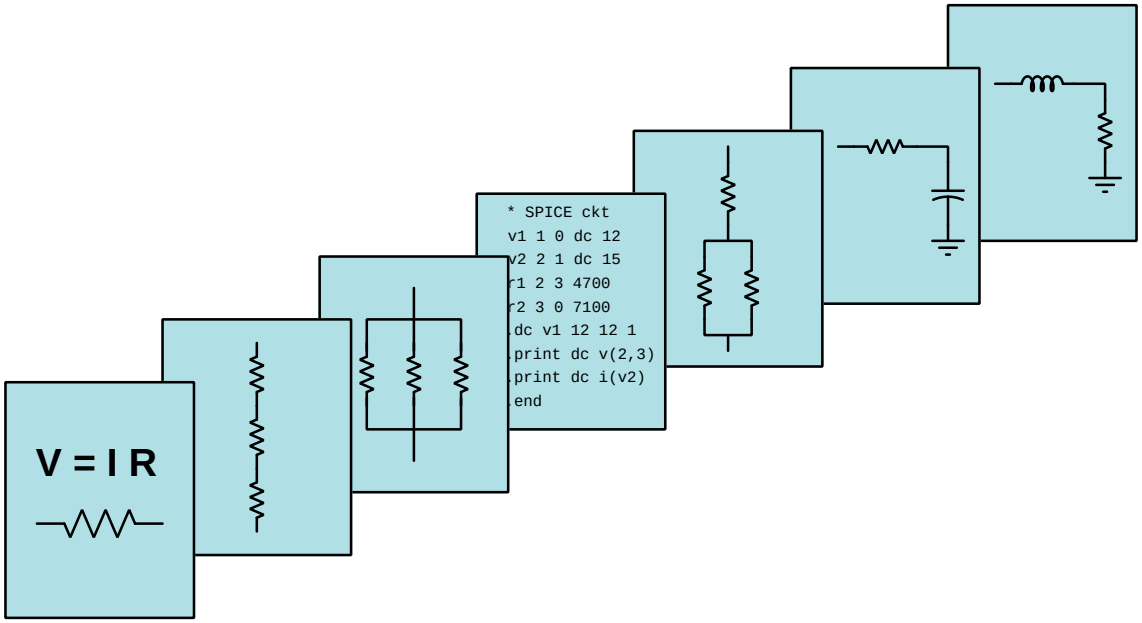
Modular Electronics Learning (ModEL) project



Analog-Digital Conversion

Last update = 2 April 2025

Contents

Chapter 1

Introduction

1.1 Recommendations for students

Digital computing is a powerful technology, capable of serving as a practical tool in many real-world applications. However, the real world is mostly analog in nature. In order to interface digital computing circuitry to analog sensors and control devices, there must be some form of "converter" to translate between the two different data types. Analog-to-Digital Converters (ADCs) perform the task of "digitizing" analog voltage or current signals so as to be acceptable to a digital computer; Digital-to-Analog Converters (DACs) take in digital data and output analog voltage or current signals to devices in the outside world.

Important concepts related to analog-digital conversion include **analog** versus **digital** quantities, **high** and **low** logic states, **binary** numeration, **serial** versus **parallel** data, **resolution**, **conversion speed**, **hexadecimal** notation, **count** value, **quantization error**, **sample rate**, **aliasing**, **filter** networks, **harmonic** frequencies, **clock** signal, **operational amplifier**, **voltage divider**, **R/2R ladder** network, **flash** conversion, **tracking** conversion, **bit bobble**, **successive approximation** conversion, **delta-sigma** conversion, **integration**, and **oversampling**.

Here are some good questions to ask of yourself while studying this subject:

- How might an experiment be designed and conducted to explore the concept of aliasing? What hypothesis (i.e. prediction) might you pose for that experiment, and what result(s) would either support or disprove that hypothesis?
- How might an experiment be designed and conducted to test whether or not an ADC was monotonic? What hypothesis (i.e. prediction) might you pose for that experiment, and what result(s) would either support or disprove that hypothesis?
- How do analog and digital quantities differ from one another?
- What are some common examples of analog and digital quantities we experience in life?
- How does the number of *bits* in a digital *word* affect the number of possible values for that word?

- What does *resolution* mean for a converter circuit and why is it important?
- What does *speed* mean for a converter circuit and why is it important?
- What is *quantization error*, and how may it be minimized in a system?
- What is the central claim of the Nyquist Sampling Theorem?
- How does *aliasing* occur, and why is it undesirable?
- What are some ways to avoid aliasing?
- How come aliasing is not a problem for DACs, but only for ADCs?
- What is a *harmonic* frequency, and how does it relate to the *fundamental* frequency?
- What does it mean to say that a waveform is "equivalent" to a harmonic series?
- What are some different ways to convert digital signals into analog?
- What are some different ways to convert analog signals into digital?
- What purpose does V_{ref} serve in a converter circuit?
- What does it mean to "oversample" a signal when converting it?

1.2 Challenging concepts related to analog-digital conversion

The following list cites concepts related to this module's topic that are easily misunderstood, along with suggestions for properly understanding them:

- **Aliasing** – the act of capturing any analog signal at discrete moments in time (i.e. sampling) and then piecing together a facsimile of that analog signal from those sampled values only works when we sample a waveform many times per cycle. If our sample internal is too long we wil miss important details of the analog waveform, and this in a nutshell is the problem of *aliasing*. A good way to comprehend this is to sketch the original waveform and then mark on it (using dots) those points at which our analog-digital converter system samples its value, and then using only those dots try to re-create the waveshape.

1.3 Recommendations for instructors

This section lists realistic student learning outcomes supported by the content of the module as well as suggested means of assessing (measuring) student learning. The outcomes state what learners should be able to do, and the assessments are specific challenges to prove students have learned.

- **Outcome** – Demonstrate effective technical reading and writing

  Assessment – Students present their outlines of this module's instructional chapters (e.g. Case Tutorial, Tutorial, Historical References, etc.) ideally as an entry to a larger Journal document chronicling their learning. These outlines should exhibit good-faith effort at summarizing major concepts explained in the text.

  Assessment – Students show how quantitative results were obtained by the author in the Tutorial chapter's examples.

- **Outcome** – Relate analog signal values to corresponding digital values for an ADC or DAC

  Assessment – Calculate corresponding analog voltage, binary digital, hexadecimal digital, and decimal digital values for a given ADC; e.g. pose problems in the form of the "ADC0804 signal values" Quantitative Reasoning question.

  Assessment – Calculate corresponding analog voltage values for digital inputs to an R-2R ladder network; e.g. pose problems in the form of the "R-2R ladder voltage calculations" Quantitative Reasoning question.

- **Outcome** – Design an overvoltage protection network for a data converter circuit

  Assessment – Sketch a schematic diagram of a circuit clamping the input voltage of an ADC to acceptable levels; e.g. pose problems in the form of the "Scaling and overvoltage protection for ADC" Conceptual Reasoning question.

- **Outcome** – Diagnose a faulted system employing ADC/DAC circuitry

  Assessment – e.g. pose problems in the form of the "Faulty flash ADC" and "Anti-aliasing design flaw" Diagnostic Reasoning questions.

- **Outcome** – Independent research

  Assessment – Locate ADC datasheets and properly interpret some of the information contained in those documents including digital output format, analog input signal range limits, reference voltage source, maximum sampling rate, digital resolution, etc.

  Assessment – Locate DAC datasheets and properly interpret some of the information contained in those documents including digital input format, analog output signal range limits, reference voltage source, maximum conversion rate, digital resolution, etc.

Chapter 2

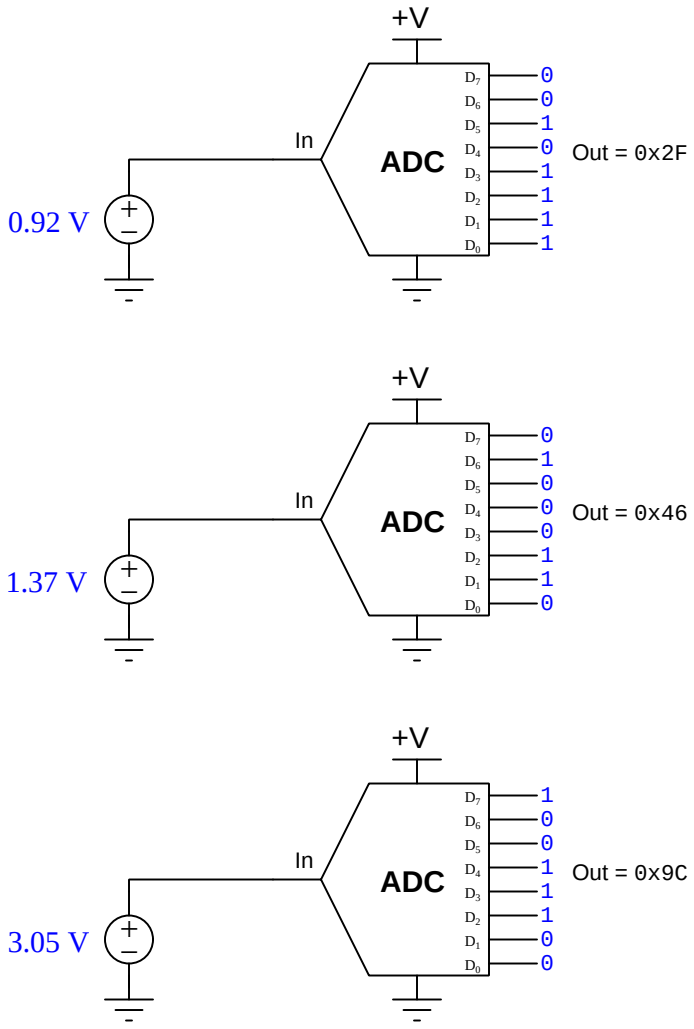
Case Tutorial

The idea behind a *Case Tutorial* is to explore new concepts by way of example. In this chapter you will read less presentation of theory compared to other Tutorial chapters, but by close observation and comparison of the given examples be able to discern patterns and principles much the same way as a scientific experimenter. Hopefully you will find these cases illuminating, and a good supplement to text-based tutorials.

These examples also serve well as challenges following your reading of the other Tutorial(s) in this module – can you explain *why* the circuits behave as they do?

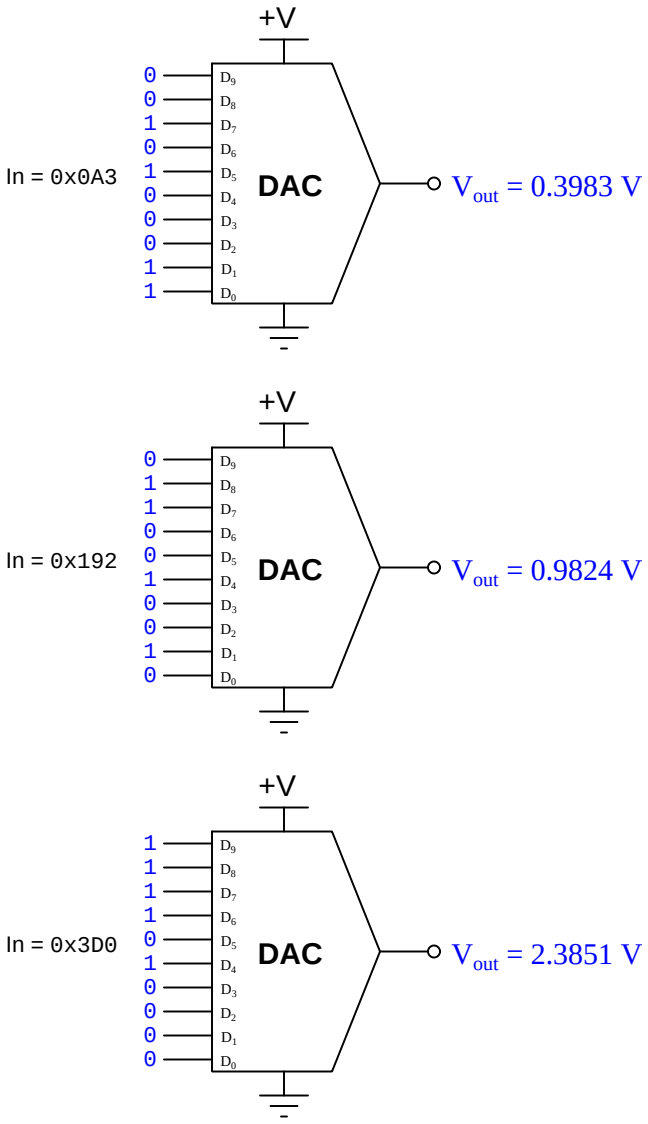
2.1 Example: Single-ended, 8-bit, unipolar ADC

The following diagrams show an 8-bit ADC with a single-ended input and a unipolar range of 0 to 5 Volts DC, experiencing multiple input voltages:



2.2 Example: 10-bit unipolar DAC

The following diagrams show a 10-bit DAC with a unipolar range of 0 to 2.5 Volts DC, experiencing multiple data words:



Chapter 3

Tutorial

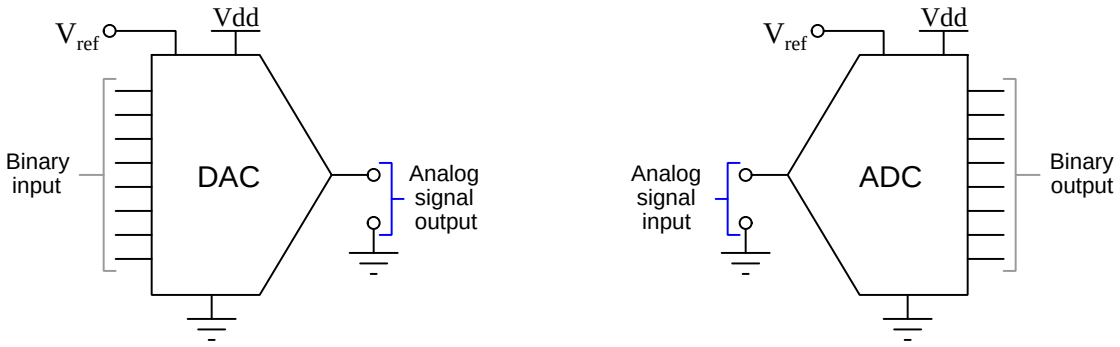
Digital signals are based on discrete states, such as the binary on/off states of digital electronic logic circuits. *Analog* signals, by contrast, are able to vary continuously within specific ranges. This contrast is clearly evident when comparing analog versus digital representations of identical quantities, such as the analog clock shown on the left versus the digital clock shown on the right:

Electronic computing circuits may be built on either digital or analog technology, but digital computers are generally far more flexible in their application. However, most quantities in the physical world are analog by nature, not limited on on/off states like the gate circuits within a digital computer.

In order to provide an interface between the internal (digital) world of an electronic computer and the external (analog) world of sensors and control devices, there must be some form of *conversion* taking place between these two types of data. Devices that generate digital representations of analog measurements are called *analog-to-digital converters*, or *ADC*s. You will find ADC circuits embedded in many microcontrollers as well as data acquisition circuitry. Devices that generate analog representations of digital quantities are called *digital-to-analog converters*, or *DAC*s. These are also found within microcontrollers and data acquisition hardware, where the computer must exert control over some external device such as a meter or actuating motor.

3.1 ADCs and DACs

A *digital-to-analog converter* or *DAC* converts a binary digital word into an analog signal, usually a voltage. An *analog-to-digital converter* or *ADC* does the reverse, converting an analog electrical signal (typically a voltage) into a binary digital word. In each case there is a correspondence between the binary number value, the analog signal magnitude, and the magnitude of an analog *reference*. In block diagram form, these converters are represented with the analog I/O ("input/output") terminal at the pointed end:



The converters shown both have *parallel* digital I/O. Some converters, though, use *serial* digital I/O where the binary bits are communicated one at a time as sequential logic states over a single terminal.

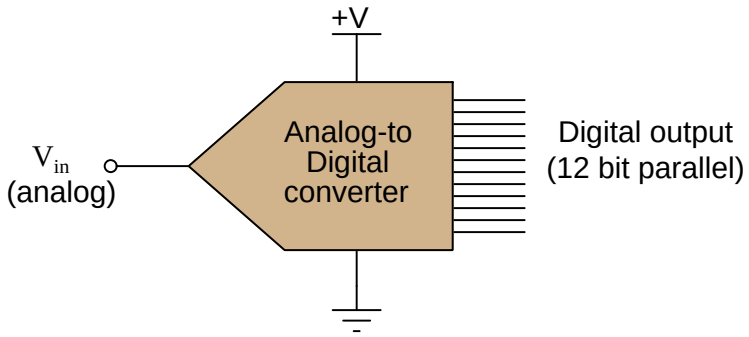
Converting between digital and analog representations of signals is always an imperfect process, and it is important to understand the various imperfections and how they affect real-world performance in analog/digital hybrid systems. At root, the fundamental mismatch between digital and analog is that analog is by its very nature *continuous* while digital is by its very nature *discrete*. While an analog signal is free to vary by the smallest degree over any span of time, a digital signal can only change value in discrete "steps" and at discrete moments in time.

Two major performance specifications for any converter, DAC or ADC, are *resolution* and *speed*. In the next two sections we will explore these important topics.

3.2 Resolution

One of the fundamental challenges of converting between analog and digital quantities is the mismatch between the two formats' amplitudes: an analog signal is infinitely variable in magnitude while a digital word is limited to a fixed number of discrete steps. Perhaps the most obvious measure of ADC or DAC performance, then, is the degree to which its analog range is divided by the digital word.

A simplified diagram of a 12-bit ADC is shown below for illustration. The major difference between this ADC and a DAC would be the fact that a DAC would *input* a 12-bit binary word and *output* a corresponding analog voltage signal:



Since the digital data "width" of this ADC is 12 bits, its digital output ranges from `000000000000` to `111111111111`, constituting a 12-bit binary integer with a range extending from `000` hexadecimal to `FFF` hexadecimal, or 0 decimal to 4095 decimal. This integer number is called the *count* value of the converter. Although the ADC shown outputs its digital data in *parallel* form (with separate terminals for the 12 individual bits), many modern ADC chips are designed for *serial* data output, where a single terminal generates a sequential series of bits timed to the pulse of a clock signal.

Supposing this 12-bit ADC has an analog input voltage range of 0 to 10 Volts (i.e. its reference voltage V_{ref} is set either internally or externally at 10.0 Volts DC), how do we relate any given digital number value to a voltage value, or vice-versa? The key here is to understand that the 12-bit *resolution* of this ADC means it has 2^{12}, or 4096 possible count values. The 10 Volt DC input range is therefore divided up into $2^{12} - 1$, or 4095, discrete increments:

$$\text{Analog resolution} = \frac{\text{Analog span}}{2^n - 1}$$

Where,

n = Number of bits in the binary "word"

For our hypothetical 0-10 VDC, 12-bit converter, the analog resolution is $\frac{1}{4095}$ of 10 Volts, or 2.442 milliVolts. Thus, for any analog signal between 0 mV and 2.442 mV, the ADC's output should be zero (binary `000000000000`); for any analog signal between 2.442 mV and 4.884 mV, the ADC's output should be one (binary `000000000001`); and so on.

If this were a DAC rather than an ADC, the analog output would increment by 2.442 milliVolts for every increment of the 12-bit binary value sent to it, from 0.0 Volts at `000000000000` to 10.0 Volts at `111111111111`.

As previously mentioned, the digital value output by an ADC or input by a DAC is commonly referred to as a *count*[1]. The word "count" is used in this context as a unit of measurement. For instance, if we subjected our 12-bit ADC to a full-scale input signal of 10 VDC, we would expect to see a full-scale digital output (binary `111111111111`) of 4095 "counts." Since most ADC circuits are designed to be linear, the mathematical relationship between input voltage and digital output "counts" is a simple proportionality:

$$\frac{V_{in}}{V_{fullscale}} = \frac{\text{Counts}}{2^n - 1}$$

We may use this formula to generate a partial table of input and output values for our 0-10 VDC, 12-bit ADC:

V_{in}	**Counts** (decimal)	**Counts** (hex)
0 V	0	`000`
2.46 mV	1	`001`
3.85 V	1576	`628`
4.59 V	1879	`757`
6.11 V	2502	`9C6`
9.998 V	4094	`FFE`
10 V	4095	`FFF`

In order to calculate a digital count value from a given input voltage, simply divide that voltage value by the full-scale voltage, then multiply by the full-scale count value and round down[2] to the nearest whole number. For any given voltage value input to the ADC, there is (ideally) one corresponding output "count" value. The converse cannot be said, however: for any given output "count" value, there is actually a *range* of possible input voltages (the span of that range being the analog resolution of the ADC, in this case 2.442 mV).

To illustrate, let us take one of the table entries as an example: an analog input of 6.11 Volts should yield a digital output of (precisely) 2502 counts. However, a digital output of 2502 counts could represent any analog input voltage ranging between 6.10989 Volts and 6.11233 Volts. This uncertainty is inherent to the process of "digitizing" an analog signal: by using a discrete quantity to represent something infinitely variable, some detail is inevitably lost. This uncertainty is referred

[1] The origin of this word has to do with the way many ADC circuits are designed, using binary *counters*. In the *tracking* design of ADC, for instance, an up-down binary counter "tracks" the varying analog input voltage signal. The binary output of this counter is fed to a DAC (digital-to-analog converter) sending an analog voltage to a comparator circuit, comparing the digital counter's equivalent value to the value of the measured analog input. If one is greater than the other, the up-down counter is instructed to either count up or count down as necessary to equalize the two values. Thus, the up-down counter repeatedly steps up or down as needed to keep pace with the value of that analog voltage, its digital output literally "counting" steps along a fixed scale representing the full analog measurement range of the ADC circuit.

[2] Whether or not the actual ADC will round *down* depends on how it is designed. Some ADCs round down, others "toggle" equally between the two nearest digital values, and others yet "toggle" proportionately between the two nearest values. No matter how you round in your calculation of count value, you will never be more than 1 count off from the real ADC's value.

to as *quantization error*: the (potential) error resulting from "quantizing" (digitizing) an inherently analog quantity into a discrete representation.

Quantization error may be reduced (but never eliminated) by using an ADC with greater resolution. A 14-bit ADC operating over the same 0-10 VDC analog input range would have approximately one-quarter the uncertainty of the 12-bit ADC (0.610 mV instead of 2.442 mV). A 16-bit ADC's uncertainty would only be (approximately) one-sixteenth that of the 12-bit ADC. The number of bits chosen for any particular ADC application is therefore a function of how precise the digitization must be.

Converters with more binary bits are internally more complex and more expensive than converters with fewer binary bits, all other performance measures being equal. Not surprisingly, you must pay for high-resolution performance.

3.3 ADC sampling and aliasing

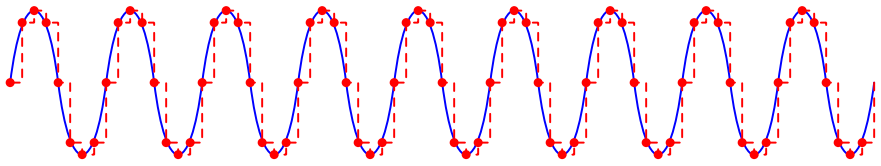
The next major performance metric for analog-digital conversion is how often the digital signal updates in value. Each time an ADC circuit "samples" its analog input signal, the resulting digital number is fixed until the next sample. Each time a DAC circuit receives a new digital number, its analog output holds that corresponding voltage value until the next digital value is received. This is analogous to monitoring a continuously moving object by taking a series of still-photographs (ADC), or conversely generating the illusion of a continuously-moving object by presenting a series of still-photographic images in rapid sequence (DAC).

For an ADC, it stands to reason that the sampling rate must be at least as often as significant changes are expected to take place in the analog signal. According to the *Nyquist Sampling Theorem*, the absolute minimum sample rate necessary to capture an analog waveform is twice the waveform's fundamental frequency. More realistic is to have the ADC sample the waveform *ten times* or more per cycle.

In general electronics work, for example with the design of electronic test equipment such as digital multimeters (DMMs) and digital storage oscilloscopes (DSOs), sampling rates must be rather fast. Modern digital oscilloscopes boast sampling rates in the *billions* of samples per second, as is necessary for the successful digitization of radio-frequency analog signals.

The following illustration shows a sinusoidal analog signal being sampled eight times per cycle (as marked by the red dots). The "digitized" waveform is shown as a dashed red line, which follows the original (blue) sinusoid fairly well due to the frequent sampling interval:

Eight samples per wave cycle

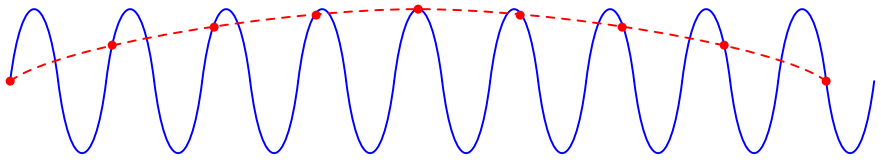


A detrimental effect of low sampling rate is something called *aliasing*[3]: a condition where the digital system "thinks" the frequency of an analog signal is far lower than it actually is.

[3] A less-commonly-used synonym for aliasing is *folding*.

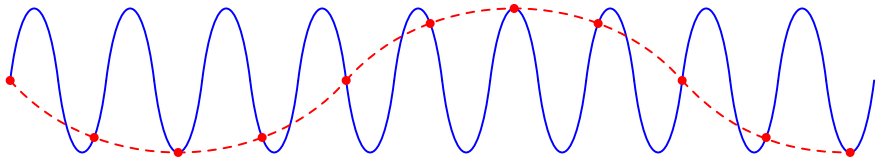
An example of signal aliasing is shown in the following illustration, where a sinusoidal signal (colored blue) is sampled at periods slightly slower than once per cycle (samples marked by red dots). The result (colored red[4]) appears to be a much lower-frequency signal as seen by the digital system, which only "sees" the values represented by the red dots[5]:

Sample interval slightly longer than one wave cycle



Aliasing can even occur when the sampling rate is slightly *faster* than the sinusoidal signal's period, as shown in this illustration:

Sample interval slightly shorter than one wave cycle



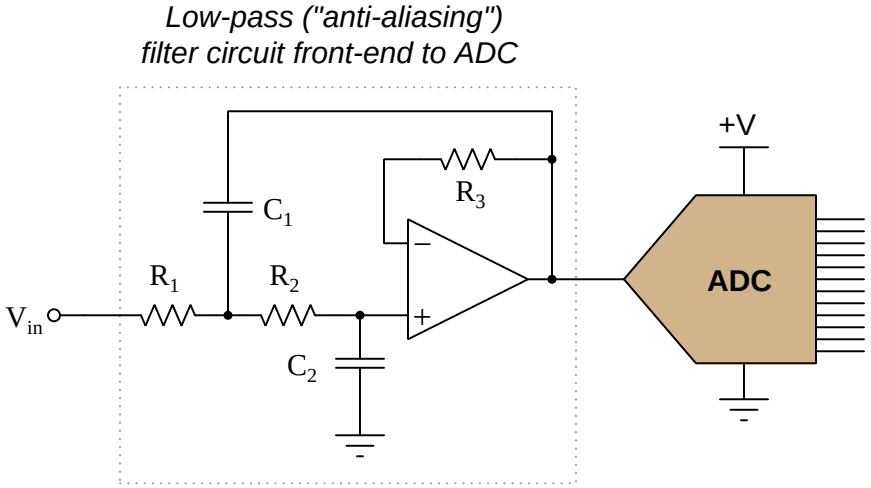
As you can see from these illustrative examples, the troubling nature of aliasing is that it causes the ADC to report a *completely incorrect*, yet *completely plausible* signal. The only way to avoid aliasing is to ensure that the sampling rate is *much* faster than the signal period: a good rule here is to sample at least 10 times per period[6], for the highest signal frequency of interest.

[4] This waveform is also "smoothed" rather than "stepped" for the explanatory purpose of making the aliased frequency easy to see.

[5] A mechanical demonstration of aliasing may be seen by using a *stroboscope* to "freeze" the motion of a rotating object. If the frequency of a flashing strobe light is set to exactly match the rotational speed of the object (e.g. 30 Hz flashing = 1800 RPM rotation), the object will appear to stand still because your eyes only see the object when it is at the exact same position every flash. This is equivalent to sampling a sinusoidal signal exactly once per cycle: the signal appears to be constant (DC) because the sine wave gets sampled at identical points along its amplitude each time. If the strobe light's frequency is set slightly slower than the object's rotational speed, the object will appear to slowly rotate in the forward direction because each successive flash reveals the object to be in a slightly further angle of rotation than it was before. This is equivalent to sampling a sinusoidal signal at a rate slightly slower than the signal's frequency: the result appears to be a sinusoidal wave, but at a much slower frequency.

[6] A very interesting case where we intentionally sample much slower than the frequency of interest is if we wish to measure a high-frequency periodic signal using a slow instrument. If we intentionally set the sample rate of the slow instrument to a value *slightly slower than an integer-multiple of the signal's period*, what we will obtain an (intentionally) aliased signal bearing the same wave-shape as the signal of interest but at a subharmonic (i.e. integer-fractional) frequency. For more information on this, refer to section 5.3 beginning on page 57.

If we cannot set the sampling rate of the ADC to be significantly faster than the highest signal frequency we might encounter, we may avoid aliasing by preventing those high signal frequencies from ever reaching the ADC. This may be done by placing an analog low-pass filter circuit before the ADC's input. Such a "front-end" circuit is called an *anti-aliasing filter*[7]:



Aliasing may still occur within digital systems, though, if one portion of a system "samples" the digital output of another portion at too slow of a rate. An example of this is a digital signal processing technique called *decimation* which entails saving every nth sample in a serial data stream and discarding the rest, which if inappropriately applied to a digitized signal will result in aliasing. The best guard against such potential troubles is to synchronize the sampling rates throughout the system, or (alternatively) ensure data sources always output values at a significantly slower rate than any functions reading them. Remember that a practical (minimum) sample rate to signal period ratio is 10:1 to ensure good analog-to-digital conversion integrity.

Digital-to-analog converters (DACs) do not suffer from aliasing, since they *output* analog waveforms rather than *sample* them as ADCs do. However, the update rate of a DAC certainly limits how well it may synthesize a high-frequency analog signal. For example, if the update rate for a DAC is 100 kHz (i.e. 100,000 times per second) and you attempt to output a 50 kHz sine wave, all you will get is a 50 kHz *square* wave because 100,000 updates per second is only 2 updates per *cycle* for a 50 kHz waveform.

The best any DAC can do is create a "stair-step" waveform emulating the desired analog wave-shape, and the more updates per wave cycle the "smoother" that waveform's shape will be. From a frequency-domain perspective, large "steps" in the wave-shape equate to the presence of high-order *harmonic* frequencies in addition to the intended fundamental frequency (and indeed, to any intended harmonics of that fundamental frequency). Low-pass filtering on the output of a DAC is therefore helpful in "smoothing" the wave-shape by attenuating those high-frequency harmonics. However, too much filtering will also attenuate any *intended* harmonics which means certain wave-

[7] This particular active filter circuit is the common *Sallen-Key* design. Other low-pass filters may be employed for this purpose at the designer's discretion.

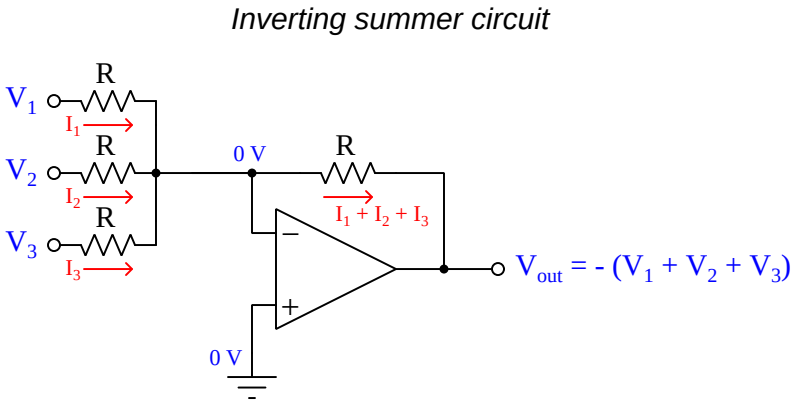
shapes such as square waves (which consist of an *infinite* series of harmonics) will be distorted by this filtering to some degree.

It is important to realize these design compromises are not an artifact of the technology, but rather a logical consequence of the fundamental mis-match between analog and digital signals. There is no fundamental limit to how quickly an analog signal value can change over time, but digital signals are limited by their update rates, and this limitation usually stems from the *clock frequency* of the digital circuitry driving the DAC. These imperfections may be minimized (but never eliminated) by using faster sampling/conversion rates.

Converters with faster update times are internally more complex and more expensive than converters with slower update times, all other performance measures being equal. Not surprisingly, you must pay for high-speed performance.

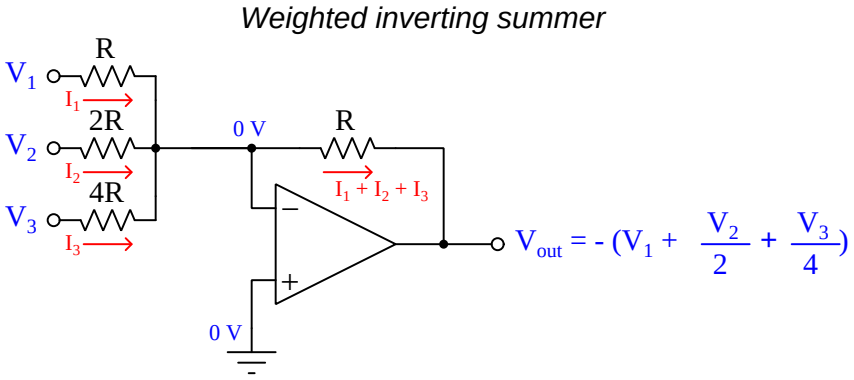
3.4 DAC circuitry: binary-weighted

This DAC circuit, otherwise known as the *binary-weighted-input* DAC, is a variation on the inverting summer opamp circuit. If you recall, the classic inverting summer circuit is an operational amplifier using negative feedback for controlled gain, with several voltage inputs and one voltage output. The output voltage is the inverted (opposite polarity) sum of all input voltages:

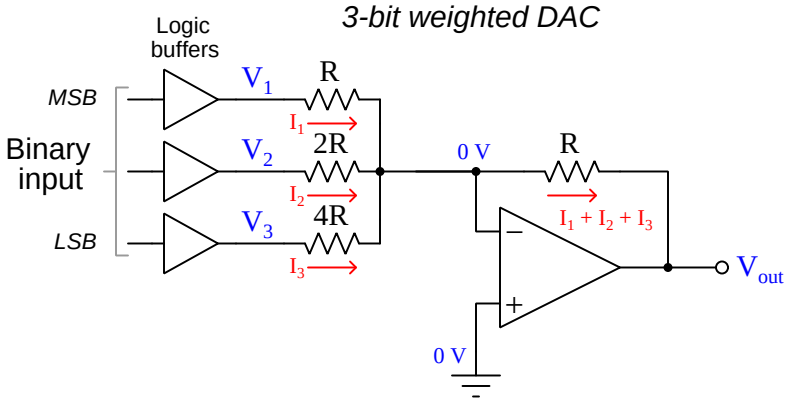


In a simple inverting summer circuit all resistors are of equal value. If any of the input resistors were different, the input voltages would have different degrees of effect on the output, and the output voltage would not be a true sum.

Let's now consider *intentionally* sizing the input resistors within the inverting summer circuit to different values. Suppose we were to set the input resistor values at multiple powers of two: R, $2R$, and $4R$, instead of all the same value R:



Beginning with V_1 and proceeding through V_3, each input voltage has exactly half the effect on the output as the preceding input voltage. In other words, input voltage V_1 has a 1:1 effect on the output voltage (i.e. a voltage gain of 1), while input voltage V_2 has half that much effect on the output (i.e. a gain of $\frac{1}{2}$), and V_3 half of that (i.e. a gain of $\frac{1}{4}$). These ratios are not arbitrary: they are the same ratios corresponding to place weights in the binary numeration system. If we drive the inputs of this circuit with digital gates so that each input is either 0 Volts or full supply voltage, the output voltage will be an analog representation of the binary value of these three bits:



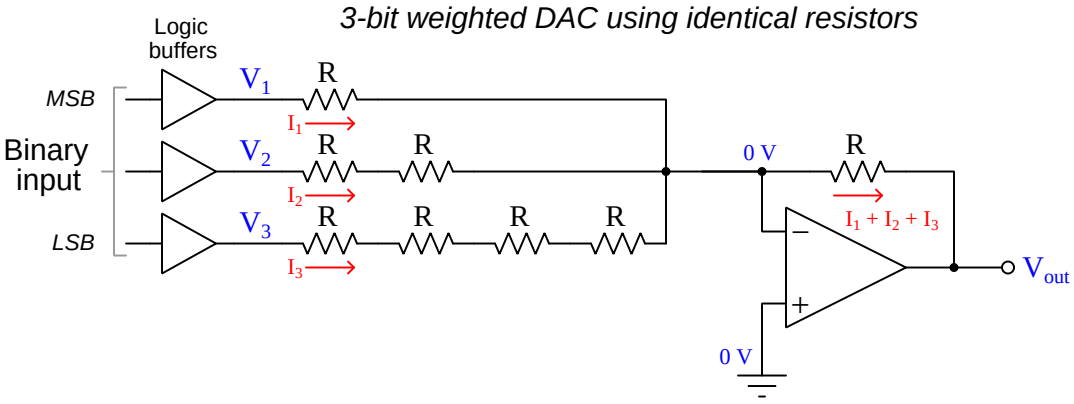
Tabulating the output voltage of this DAC circuit for all eight combinations of binary bits (000 through 111), and assuming each digital buffer has an output of either 0 Volts or +5 Volts, we obtain the following progression of voltages:

Binary count	V_{out}
000	0.00 V
001	−1.25 V
010	−2.50 V
011	−3.75 V
100	−5.00 V
101	−6.25 V
110	−7.50 V
111	−8.75 V

This DAC's analog output range is adjustable by altering the value of the feedback resistor. For example, by reducing the feedback resistor's value to $\frac{8}{10}$ of R, the DAC will output one Volt per count (e.g. −1 Volt for binary 001, −4 Volts for binary 100, −7 Volts for binary 111, etc.). Alternatively, if we wished this DAC to have a "normalized" analog range matching the digital logic supply voltage rails (i.e. 0 to 5 Volts), we could resize the feedback resistor in the summer circuit to be $\frac{5}{8.75}$ of R.

A three-bit DAC has terribly coarse resolution, and therefore to be practical this circuit would have to be equipped with many more resistors, buffers, and discrete logic inputs. The basic configuration of the inverting summer would remain the same, though, using resistors with power-of-two resistance coefficients as large as needed for the number of bits. Instead of just R, $2R$, and $4R$ resistors, a binary-weighted DAC with additional bits would also have an $8R$ resistor on the fourth line, a $16R$ resistor on the fifth line, a $32R$ resistor on the sixth line, etc. Each successive bit, with its progressively larger resistor, would have less effect on V_{out} and thereby provide finer resolution.

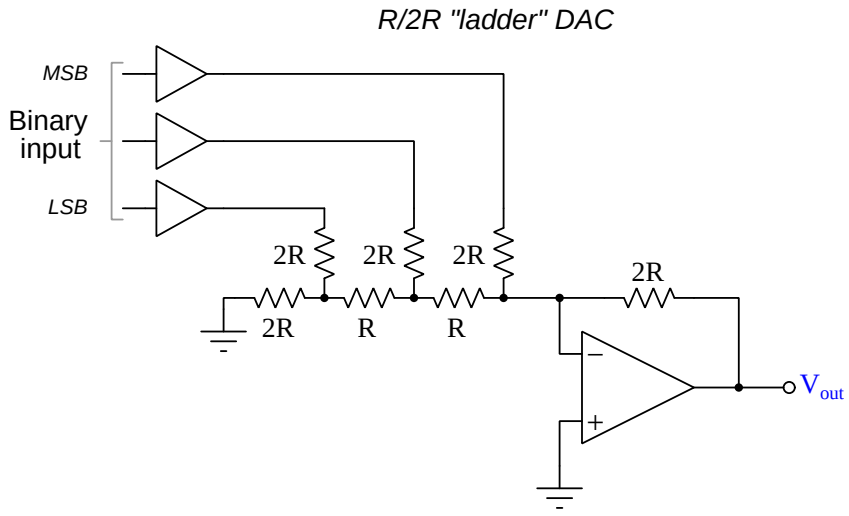
One weakness of the binary-weighted DAC design is the need for precision resistors with n unique values (where n is the number of bits). Generally it is less expensive to build circuits using common component values wherever possible, and so it is worth considering ways to reduce the number of unique resistor values here. A modification we could make to the previous design is replacing the three input resistors with series networks consisting solely of resistors having one value (R):



Unfortunately, this approach merely substitutes one type of complexity for another: volume of components over diversity of component values. There is, however, a more efficient design methodology explored in the next section.

3.5 DAC circuitry: $R - 2R$

By constructing a different kind of resistor network on the input of our summing circuit, we may achieve the same kind of binary weighting with only two kinds of resistor values, and with only a modest increase in resistor count. This "ladder" network looks like this:



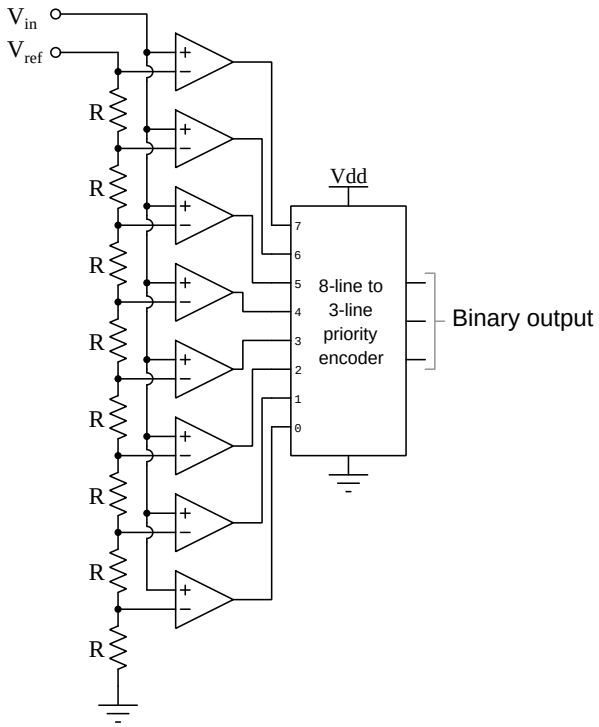
With the ladder design, each additional input bit requires just two resistors, an R and a $2R$, while adding no new resistor *values* to the bill of materials necessary to construct the DAC circuit.

A quantitative analysis of the $R - 2R$ ladder network is beyond the scope of this Tutorial, but is explained in the Derivation section 5.1 beginning on page 46. Suffice it to say that the digital input furthest from the virtual ground node of the opamp circuit will have the least effect, and each successive input line closer to the virtual ground has twice the effect of the one previous. Thus, the $R - 2R$ ladder network provides natural binary-weighting with an economy of resistor values.

The operational amplifier is not strictly necessary, either, although its presence in the circuit permits larger output voltages as well as larger output currents which is useful for maintaining good digital-analog conversion accuracy when driving loads. Without the opamp in place, the output voltage will "sag" when driving a load.

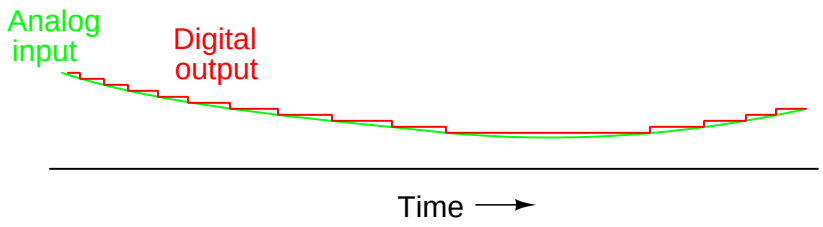
3.6 ADC circuitry: flash conversion

ADC circuitry tends to be much more complex than DAC circuitry, but the simplest ADC to comprehend is the so-called *flash* converter, named for its very high conversion rate (i.e. speed). The following schematic diagram shows a 3-bit flash ADC circuit:



V_{ref} is a stable reference voltage provided by a precision voltage regulator as part of the converter circuit, omitted from this schematic. As the analog input voltage exceeds the divided reference voltage value presented at the inverting input of each comparator, the comparator outputs sequentially switch to high states. The priority encoder generates a binary number based on the highest-order active input, ignoring all other active inputs.

When given a time-varying analog input signal as shown in the graph below (green curve), a flash ADC will produce a digital output resembling the "stepped" trace (red) with each "step" representing a new count value:



Not only is the flash converter the simplest in terms of operational theory, but it is extremely fast, being limited only by comparator and gate propagation delays[8]. Unfortunately, it is the most component-intensive for any given resolution. A three-bit flash ADC would requires eight comparators; a four-bit version would require sixteen comparators. The number of necessary comparators is equal to 2^n, with n being the number of bits. Considering that eight bits is generally considered the minimum necessary for any practical ADC ($2^8 = 256$ comparators needed!), the flash ADC design quickly reveals its weakness. Even when constructing the entire ADC as an integrated circuit (IC) on a single wafer of silicon, the parts count is daunting for all but the lowest-resolution designs.

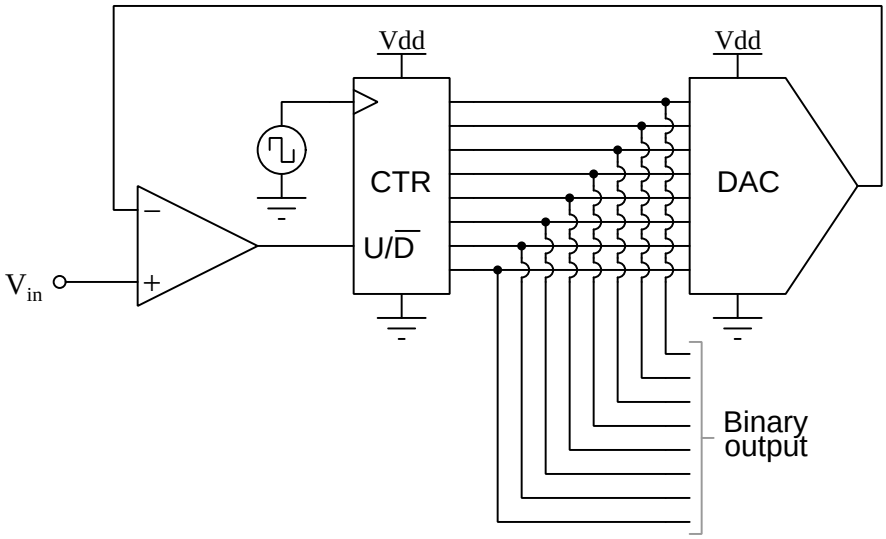
A noteworthy advantage of the flash converter, easily overlooked, is its ability to render a customized *non-linear* output. With equal-value resistors in the reference voltage divider network, each successive binary count represents the same amount of analog signal increase, providing a proportional response. For special applications, however, the resistor values in the divider network may be intentionally made unequal to grant the ADC a custom response to the analog input signal. Such a feature could be useful if digitizing the analog signal from a non-linear sensor (e.g. thermistor temperature sensor) with a known characteristic function: the sensor's non-linear response to temperature could be automatically "linearized" by a custom flash ADC to produce a digital count value directly proportional to temperature.

[8] Such delay times, especially when mismatched between different comparators in the flash ADC, may cause incorrect digital values to appear for very brief periods of time. These erroneous count values are sometimes called *sparkle codes* because in the early days of digitally-encoded television systems such errors would produce transient points of light on the television screen.

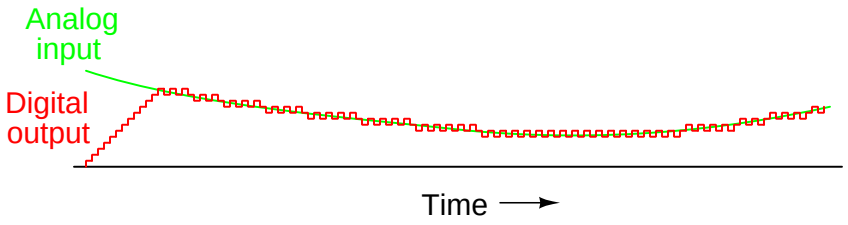
3.7 ADC circuitry: tracking conversion

An ADC's digital output value is conventionally known as a *count*, and the reason for this interesting choice of labels comes from the fact that many ADC circuit designs incorporate a clock-driven digital *counter* to match the analog signal value.

In the following schematic we see an ADC design utilizing an up/down counter connected to a DAC, which drives one input of a comparator, which in turn controls the counter's direction. When the analog input signal voltage exceeds the DAC's output voltage, the counter goes into the "count up" mode to drive the DAC's output closer to equality with the analog signal. When the DAC output exceeds the analog input, the comparator output changes state and commands the counter "count down" rather than count up. Either way, the DAC output always counts in the proper direction to *track* the input signal. For this reason, it is called a *tracking ADC*:



When given a time-varying analog input signal as shown in the graph below (green curve), a tracking ADC will produce a digital output resembling the "stepped" trace (red) with each "step" representing a new count value:



Some immediate comparisons may be made between the tracking ADC design and the flash ADC design. Regarding hardware, the tracking ADC requires far fewer components for large-bit resolutions than a comparable flash ADC. A 10-bit flash converter, for example, would require

$2^{10} = 1024$ comparators in addition to a 1024-line to 10-line encoder. By contrast, a 10-bit tracking ADC would only require a 10-bit up/down counter (containing ten flip-flops), 10-bit DAC (containing 21 resistors and an opamp), a single comparator, and a clock source. The price we pay for such an economy of components is reduced performance. As we can see from the analog/digital graph, the tracking converter has a maximum rate-of-change it is capable of tracking which means it will fail to properly digitize the highest-frequency harmonics (i.e. steepest rise and fall times) of an analog signal. This limit is a function of the ADC's resolution and its clock frequency[9].

The inability of a tracking-style ADC to quickly follow changes in the analog signal is related to an ADC performance metric called *step recovery*. This is a measure of how quickly an ADC changes its output to match a large, sudden change in the analog input. Tracking ADC circuits exhibit relatively poor step recovery in comparison to their update rates.

A unique limitation of the tracking ADC design is the fact that the binary output never stabilizes: it always switches between counts with every clock pulse, even with a perfectly constant analog input signal. This phenomenon is informally known as *bit bobble*, and it can be problematic in some digital systems. A direct-reading digital instrument equipped with a tracking ADC exhibits the annoying characteristic of constantly alternating between adjacent digits in the least-significant place of the display. Any computer software processing this "bobbling" value will likewise have to contend with this noise artifact of the ADC, oscillating at half the frequency of the ADC counter's clock pulse.

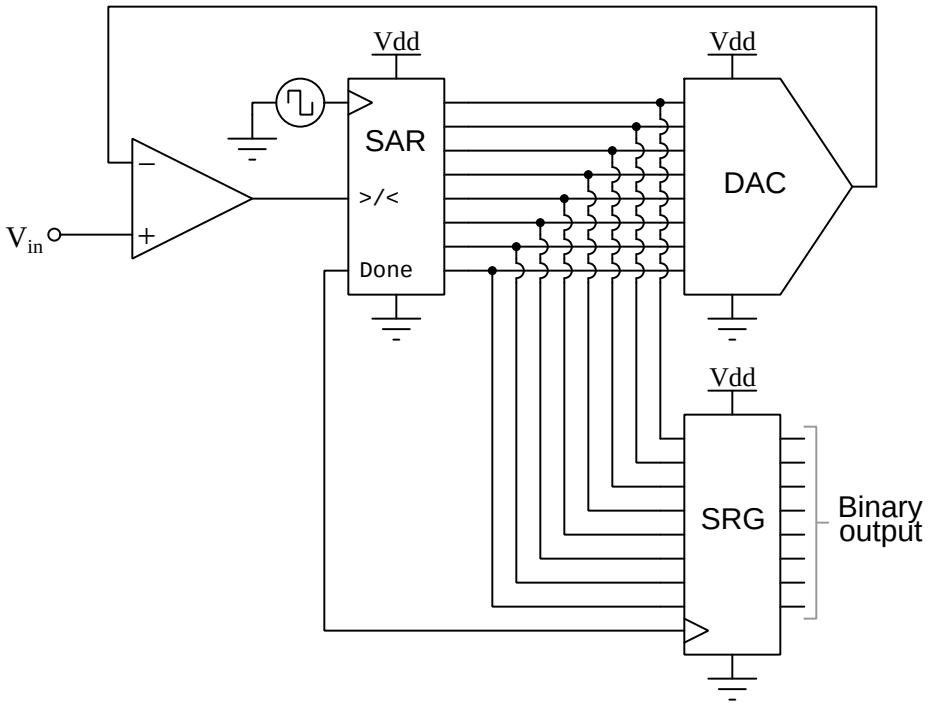
This tendency can be overcome, though, through the creative use of a shift register. For example, the counter's output may be latched through a parallel-in/parallel-out shift register only when the output changes by two or more steps. Building a circuit to detect two or more successive counts in the same direction requires some ingenuity, but is worth the effort.

[9] Imagine increasing the number of bits on the counter and DAC while maintaining the same clock frequency. With finer resolution, each increment or decrement of the counter covers a smaller span of analog voltage, which means for the same clock frequency the ADC will "track" at a slower rate. We may compensate for this by increasing clock frequency, but then we may run into speed limitations of the counter and perhaps the internal DAC.

3.8 ADC circuitry: successive approximation conversion

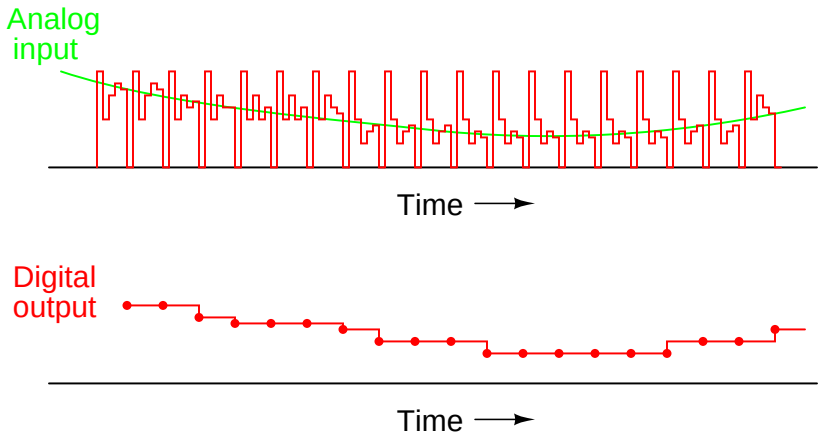
A different counter-based approach to analog-to-digital conversion is to use a counter that does not count in binary sequence. An example of this is a device called a *successive-approximation register*. Instead of incrementing in binary sequence, this register counts by trying all values of bits starting with the most-significant bit and finishing at the least-significant bit. Throughout the count process, the register monitors the comparator's output to see if the binary count is less than or greater than the analog signal input, adjusting the bit values accordingly. The way the register counts is identical to the "cut-and-try" method of decimal-to-binary conversion, whereby different values of bits are tried from MSB to LSB to derive a binary number equaling the original decimal number. The advantage to this counting strategy is much faster convergence on a changing analog signal because the DAC's takes larger (initial) steps than with the 0-to-full count sequence of a regular counter.

Without showing the inner workings of the successive-approximation register (SAR), the circuit looks like this:



It should be noted that the SAR is generally capable of outputting the count value *serially* (one bit at a time), thus eliminating the need for a parallel shift register.

Plotted over time, the operation of a successive-approximation ADC looks like this:

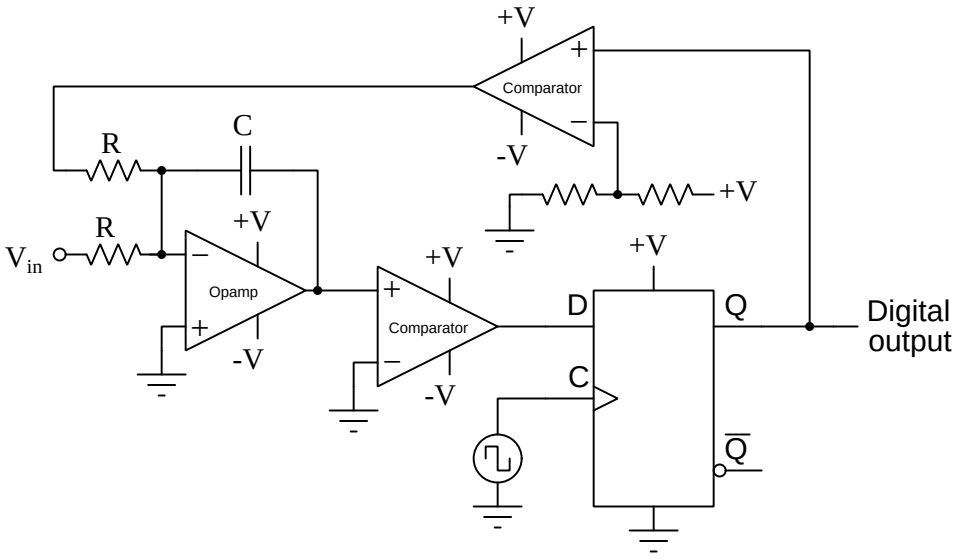


Note how the digital output as seen at the output of the shift register doesn't "cycle" high and low as it does at the output of the SAR, and so these two sets of digital counts must be shown separately on the graph.

3.9 ADC circuitry: delta-sigma conversion

One of the more advanced ADC technologies is the so-called delta-sigma, or $\Delta\Sigma$ (using the proper Greek letter notation). In mathematics and physics, the capital Greek letter delta (Δ) represents *difference* or *change*, while the capital letter sigma (Σ) represents *summation*: the adding of multiple terms together. Sometimes this converter is referred to by the same Greek letters in reverse order: sigma-delta, or $\Sigma\Delta$.

In a $\Delta\Sigma$ converter, the analog input voltage signal is connected to the input of an *integrator*, producing a voltage rate-of-change, or slope, at the output corresponding to input magnitude. This ramping voltage is then compared against ground potential (0 Volts) by a comparator. The comparator acts as a sort of 1-bit ADC, producing 1 bit of output ("high" or "low") depending on whether the integrator output is positive or negative. The comparator's output is then latched through a D-type flip-flop clocked at a high frequency, and *fed back* to another input channel on the integrator, to drive the integrator in the direction of a 0 volt output. The basic circuit looks like this:

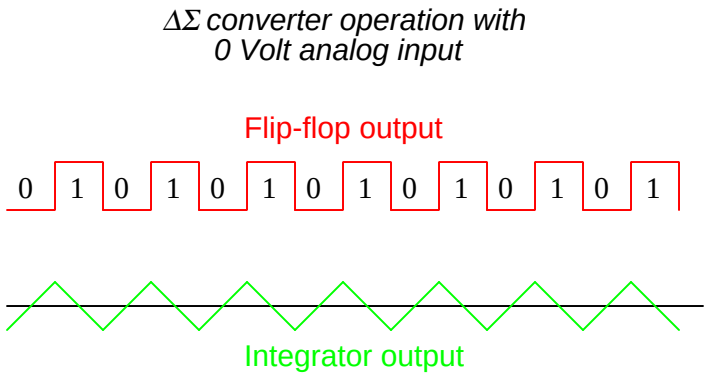


The operational amplifier forms the heart of the (summing) integrator. Two resistors feed current to the capacitor proportional to the magnitude and polarity of the two voltage-signal inputs while the opamp acts as a current source: maintaining steady (DC) current through the capacitor for steady (DC) input voltage signals. The integrator circuit's output will stabilize only when the algebraic sum of the two input voltages is zero. Any time these inputs do not sum to zero, the integrator's output voltage *ramps* at a rate proportional to the disparity.

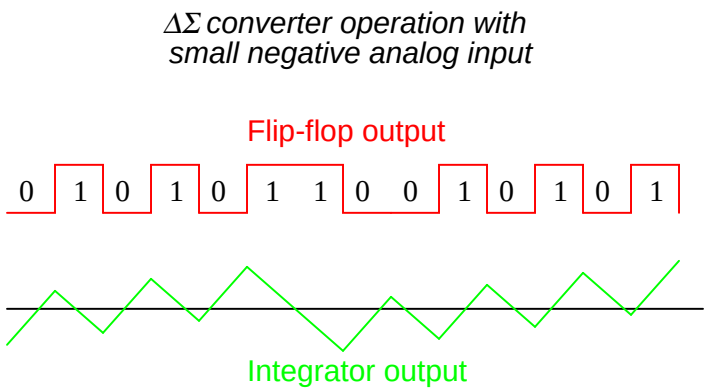
The integrator's output signal feeds the noninverting input of a comparator (i.e. the 1-bit ADC). Next comes the D-type flip-flop, latching the comparator's output at every clock pulse, sending either a "high" or "low" signal to the next comparator at the top of the circuit. This final comparator is necessary to convert the single-polarity 0V / 5V digital logic level output voltage of the flip-flop into a +V / −V voltage signal to be fed back to the integrator.

If the integrator output is positive, the first comparator will output a "high" signal to the D input of the flip-flop. At the next clock pulse, this "high" signal will be output from the Q line into the noninverting input of the last comparator. This last comparator, seeing an input voltage greater than the threshold voltage of $\frac{1}{2}$ +V, saturates in a positive direction, sending a full +V signal to the other input of the integrator. This +V feedback signal drives the integrator's output in a negative direction. If that output voltage ever becomes negative, the feedback loop will send a corrective signal (−V) back around to the top input of the integrator to drive it in a positive direction. This is the delta-sigma concept in action: the first comparator senses a *difference* (Δ) between the integrator output and zero Volts. The integrator *sums* (Σ) the comparator's output with the analog input signal.

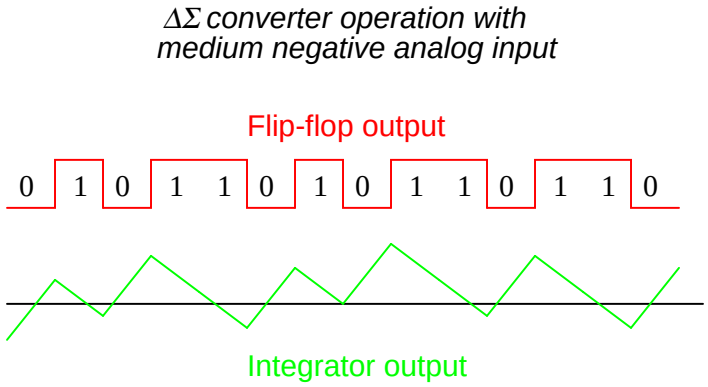
Functionally, this results in a serial stream of bits output by the flip-flop. If the analog input is zero Volts, the integrator will have no tendency to ramp either positive or negative, except in response to the feedback voltage. In this scenario, the flip-flop output will continually oscillate between "high" and "low" as the feedback system "hunts" back and forth trying to maintain the integrator output at zero Volts:



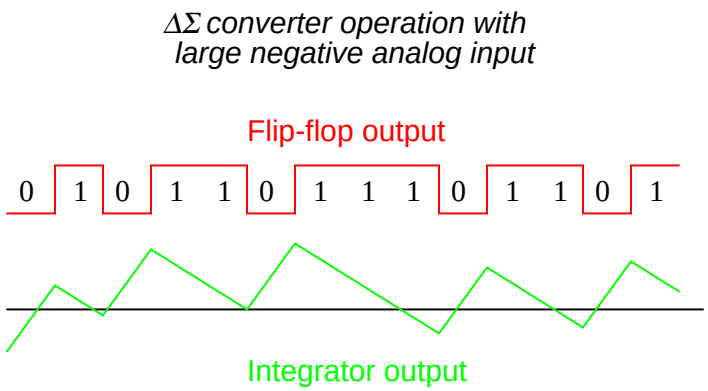
If, however, we apply a negative analog input voltage, the integrator will have a tendency to ramp its output in a positive direction. Feedback can only add to the integrator's ramping by a fixed voltage over a fixed time, and so the bit stream output by the flip-flop will not be quite the same:



By applying a larger (negative) analog input signal to the integrator, we force its output to ramp more steeply in the positive direction. Thus, the feedback system has to output more 1's than before to bring the integrator output back to zero Volts:



As the analog input signal increases in magnitude, so does the occurrence of 1's in the digital output of the flip-flop:



A parallel binary number output is obtained from this circuit by *averaging* the serial stream of bits together. For example, a counter circuit could be designed to collect the total number of 1's output by the flip-flop in a given number of clock pulses. This count would then be indicative of the analog input voltage.

Variations on this theme exist, employing multiple integrator stages and/or comparator circuits outputting more than 1 bit, but one concept common to all $\Delta\Sigma$ converters is that of *oversampling*. Oversampling is when multiple samples of an analog signal are taken by an ADC (in this case, a 1-bit ADC), and those digitized samples are averaged. The end result is an effective increase in the number of bits resolved from the signal. In other words, an oversampled 1-bit ADC can do the same job as an 8-bit ADC with one-time sampling, albeit at a slower rate.

Chapter 4

Historical References

This chapter is where you will find references to historical texts and technologies related to the module's topic.

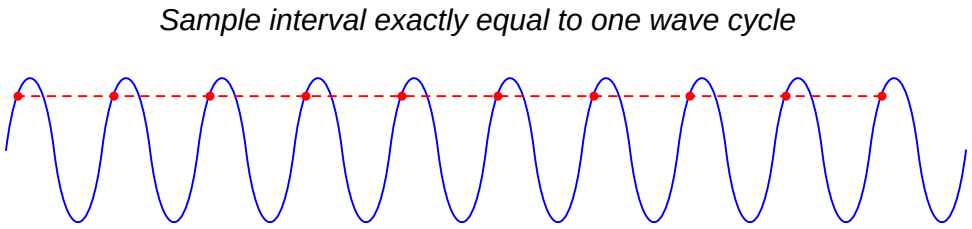
Readers may wonder why historical references might be included in any modern lesson on a subject. Why dwell on old ideas and obsolete technologies? One answer to this question is that the initial discoveries and early applications of scientific principles typically present those principles in forms that are unusually easy to grasp. Anyone who first discovers a new principle must necessarily do so from a perspective of ignorance (i.e. if you truly *discover* something yourself, it means you must have come to that discovery with no prior knowledge of it and no hints from others knowledgeable in it), and in so doing the discoverer lacks any hindsight or advantage that might have otherwise come from a more advanced perspective. Thus, discoverers are forced to think and express themselves in less-advanced terms, and this often makes their explanations more readily accessible to others who, like the discoverer, comes to this idea with no prior knowledge. Furthermore, early discoverers often faced the daunting challenge of explaining their new and complex ideas to a naturally skeptical scientific community, and this pressure incentivized clear and compelling communication. As James Clerk Maxwell eloquently stated in the Preface to his book *A Treatise on Electricity and Magnetism* written in 1873,

> It is of great advantage to the student of any subject to read the original memoirs on that subject, for science is always most completely assimilated when it is in its nascent state . . . `[page xi]`

Furthermore, grasping the historical context of technological discoveries is important for understanding how science intersects with culture and civilization, which is ever important because new discoveries and new applications of existing discoveries will always continue to impact our lives. One will often find themselves impressed by the ingenuity of previous generations, and by the high degree of refinement to which now-obsolete technologies were once raised. There is much to learn and much inspiration to be drawn from the technological past, and to the inquisitive mind these historical references are treasures waiting to be (re)-discovered.

4.1 Stroboscopic voltage sampling

If we sample a periodic AC signal exactly once per cycle (i.e. sample rate = waveform period) what we get is a constant value representing the same point on each of the repeated cycles:



Normally this is an undesirable result for digitizing or otherwise measuring an AC signal, because it yields the false impression that the signal is actually DC rather than AC. However, this very technique has its practical uses. Consider the case of early researchers of electromagnetic generators (called "dynamos" at the time) who needed to plot the waveforms of the alternating voltages and currents produced by these machines, using measurement technology suitable only for DC and *very* low-frequency AC signals.

A method devised by some of these researchers consisted of a switch contact closed for a brief moment in time at one angular position of the generator's rotating shaft. This switch contact periodically connected a capacitor and DC voltmeter to the AC power lines of the generator, allowing the voltmeter to "sample" the generator's instantaneous voltage at that point of the shaft's rotation. The fact that this sampling occurred at the closing of the switch contact – which was mechanically synchronized with the generator's rotation – absolutely ensured a once-per-revolution sampling interval regardless of generator speed. The capacitor served as an analog "memory" unit to hold the sampled voltage value for the voltmeter until the next switch closure event.

In the book *Electrical Measurements* written by Frank Laws in the year 1917, we read a description of how one such apparatus functioned:

> The fundamental idea is to connect periodically the measuring apparatus to the circuit for a time so short that during it the current or voltage remains practically unchanged. This is accomplished by an apparatus which is the equivalent of a key operated by a rigid connection from the dynamo shaft. The [page 613]
>
> key is closed for an instant once during each revolution of the dynamo, and at a definite point on the wave.
>
> If the voltage is high, a large non-reactive resistance, R, Fig. 378, is placed across the circuit, and by means of a tap a definite fraction of the total voltage is impressed on the apparatus.
>
> Referring to Fig. 378, the contact wheel of hard rubber or fiber is at W. In the original arrangement this wheel was attached directly to the dynamo shaft; B_1 is a brush which rests on a collector ring and gives permanent connection to the contact point P, which projects very slightly from the periphery of the wheel, W. B_2 is a thin and very light brush which rests on the

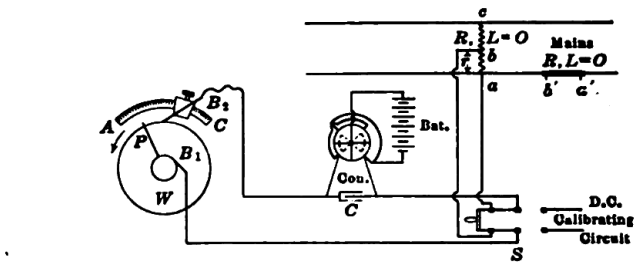


Fig. 378.—Connections for contact method for wave form, using quadrant electrometer.

> contact wheel. It is supported from a movable brush holder which may be set at any desired position along a uniformly graduated arc, AC.
>
> The measurements may be made by the aid of an electrostatic voltmeter, a quadrant electrometer or a ballistic galvanometer. In the arrangements shown in Fig. 378 the needle of the electrometer is kept charged to a high potential by the battery and consequently the deflection is sensibly proportional to the applied voltage; that is, to the potential difference between a and b at the instant of contact. The well-insulated condenser, C, adds to the capacity of the electrometer so that the voltage on the instrument will not be appreciably altered by leakage during the time between the successive contacts of B_2 and P.
>
> The process is to set the brush at a definite position on the arc and to read the electrometer; then to move the brush forward to another position and take another reading, and so on. [page 614]

The magnitude of the deflections may be controlled by means of the battery and the resistance *ba*.

The electrometer is calibrated by use of direct-current voltages as indicated.

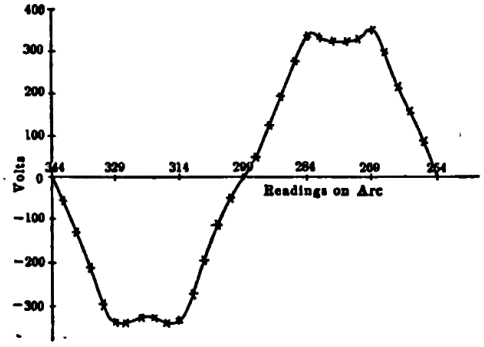


Fig. 379.—Wave form determined by contact method.

The electrometer readings, reduced to volts, are plotted against the readings on the uniformly graduated arc, as shown in Fig. 379, and a smooth curve drawn through the points. [page 615]

Of course, this waveform plotting would have to be done by hand, based on multiple instantaneous voltage measurements taken at different shaft positions. As we may surmise from this particular hand-plot, the AC generator being tested must have been an eight-pole machine because we see one complete AC cycle for only 90 degrees of shaft revolution (from 344 degrees to 254 degrees).

As crude as this technique may seem to the modern reader, it stands as an ingenious method for performing electrical measurements that would have been impossible for any technology of that era if directly connected to the generator. It is easy to take for granted our modern access to *gigahertz*-sampling oscilloscopes which could have easily performed these measurements without the use of shaft-actuated switch contacts, but the point here is to respect and learn from pioneers in the field of electrical measurement, and perhaps take away lessons that may still find use today. Anywhere we might encounter a periodic signal whose frequency is too high to directly measure, we may (possibly) exploit the principle of aliasing for its precise plotting and measurement (and digitization).

4.2 Scanning electron tube ADC

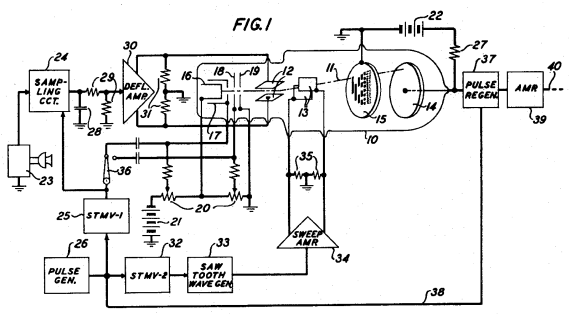
In 1947 a Bell Telephone Laboratories researcher named Frank Gray devised a way to improve a particular form of digital signal communication by replacing standard binary coding with what he called *reflected binary code*. This new code scheme came to be known by his last name, *Gray code*.

To understand Gray's invention, one must grasp the application for which it was invented. Gray's employer, Bell Telephone, designed, built, and maintained long-distance telephone communication networks across the United States. They were interested in converting audio telephone signals into digital form suitable for transmission over long distances with little corruption from external noise and cable attenuation. The digitization of audio telephone signals took the form of voltage pulses following each other in rapid succession, commonly known as a *serial data* stream. Sets of contiguous bits in this data stream represented individually-sampled values of the analog signal.

Gray begins his narrative on page 5 of the patent by explaining how binary coding works:

> This invention relates to pulse code transmission and particularly to the coding of a message signal in a novel code and to the decoding thereof.
>
> In communication by pulse code transmissions the instantaneous amplitudes of a message to be transmitted are successively sampled and each of the successive samples is translated into a code group of on-or-off pulses. By reason of the on-or-off character of the pulses, such a code is denoted a binary code. The number of pulse positions in a code group is the same from group to group. With five such positions the code is a 5-digit binary code. With seven it is a 7-digit binary code; and in general, with in such positions it is an n-digit binary code. A code pulse group of n pulse positions may contain any number, from 0 to n, of "on" pulses. In the conventional n-digit binary code, the number and arrangement of pulses is in accordance with the conventional binary number notation. Thus, for example, with three digits, the number five is written in the conventional binary number notation as 101. Correspondingly, in the conventional 3-digit binary pulse code, the pulses occur in the time sequence P, −, P, where "P" stands for an "on" pulse and "−" stands for an "off" pulse; i. e., a blank pulse position. [page 5]

The technology of that era for converting an analog audio signal into a series of digital pulses used a vacuum tube called a *cathode ray tube*, or CRT. Many different types of CRTs were manufactured, and they all shared the common features of an "electron gun" assembly at one far end generating a thin beam of high-speed electrons, as well as means to bend or deflect this beam before it struck an object at the other end of the evacuated tube. CRTs were once used for television and computer terminal displays, but in this application functioned as a form of *analog to digital converter*. The first page of Gray's patent contains a figure showing the general construction and supporting circuitry of the CRT:



Gray's narrative resumes with a description of the CRT and the analog-to-digital conversion process:

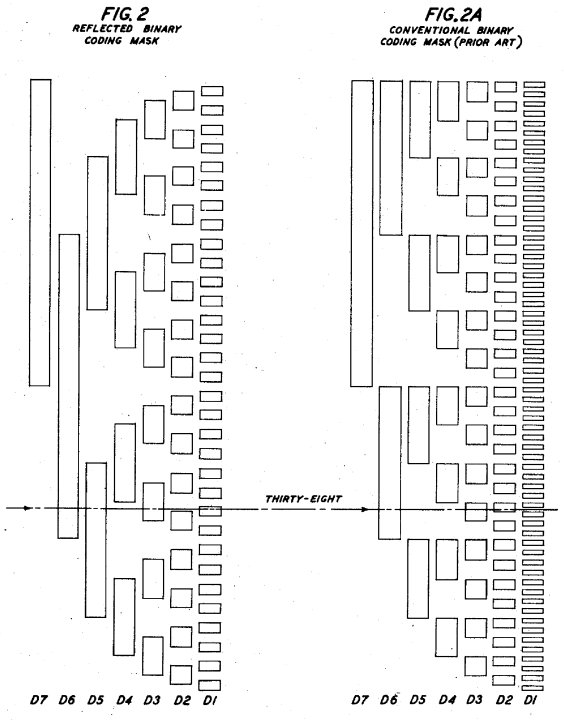
> In Italian Patent 437,300, published June 30, 1948, there is described an instrument for translating message signal samples into code pulse groups in the conventional binary code. In brief, it comprises a cathode beam tube having a coding mask, an electron gun for projecting an electron beam toward the mask, a collector anode for receiving electrons which pass through the mask and deriving pulses from them, means for deflecting the cathode beam in one direction along the mask to a location proportional to the signal sample amplitude, and means for sweeping the beam in a perpendicular direction across the mask between successive sample controlled deflections. As currently employed, the mask comprises a rectangular array of apertures arranged in n columns and 2^n rows, where n is the number of digits of the code. Each aperture row corresponds to a unique value of the signal-controlled beam deflection. The apertures of the various rows are located in conformity with the location of the 1's in a tabulation of successive binary numbers, while the blank portions of the mask are located in conformity with the 0's in the same tabulation. Thus, when the cathode beam is deflected under control of the signal to a particular aperture row and thereupon swept laterally along this row, a train of current pulses may be drawn from the collector whose location on the time scale is in accordance with the arrangement of the 1's and 0's in the binary number whose value is equal to the value of the signal sample being coded. `[page 5]`

The use of a cathode ray tube to convert an analog voltage signal into a serial data stream is rather ingenious[1] and deserves some elaboration. The beam of electrons repeatedly sweeps across the face of the mask in one axis by the direction of a "sawtooth" wave voltage signal applied to a

[1] Not only is this tube design ingenious, but it also stands as an example of just how flexible vacuum tube technology was. Even though vacuum tubes have been made obsolete by semiconductor electronic devices for all but the most

pair of metal plates located to either side of the beam. Another pair of plates perpendicular to the first pair deflects the beam in the other axis, this deflection occurring at the whim of the sampled analog signal. A sample-and-hold circuit (24 in the diagram) captures the analog signal value at the start of the beam sweep and holds that signal voltage constant for the duration of the sweep, so that the beam traces only straight-line paths across the mask. At the conclusion of each sweep, the sample-and-hold circuit re-reads the analog input signal and holds it steady for the next sweep. Holes placed in the mask allow the beam to pass through at certain points where it strikes an anode surface (14) and registers outside the tube as a current pulse. Therefore, as the beam sweeps across a row of the mask, an electrical pulse sequence develops at the anode corresponding to the placement of the holes in that row of the mask.

Figure 2 (from page 2 of the patent) shows two possible mask patterns, one standard binary and the other Gray's "reflected binary" code:



The electron beam's periodic sweep crosses the face of the mask in the horizontal axis (as the masks are shown above), so that the beam "probes" one row at a time. The vertical position of the beam on the mask is determined by the sampled analog signal. Each column of holes on the mask represents one bit (one "digit" as described by Gray) in the binary word. For the sample masks shown, a seven-bit binary sequence is generated for every sweep of the beam from left to right. For an

specialized applications, and for very good reasons (e.g. physical ruggedness, energy efficiency, size, service life), this does not change the fact that this one technology was capable of so many different types of electronic functions. The manipulation of electrons traveling through a vacuum lends itself to a wide range of applications, including amplification, oscillation, rectification, signal modulation, visual display, optical sensing, memory, etc.

analog signal corresponding to the value 38 (shown on the figure), the beam generates the sequence `0100110` as it passes from left to right over that row of holes. Tallying each bit's place-weight, we get 32 + 4 + 2 = 38. A maximum-value analog signal would deflect the beam all the way to the top row where the sequence would be `1111111` (decimal value 127). A minimum-value signal would pull the beam all the way to the bottom to generate `0000001` (decimal value 1). Thus, the sweeping electron beam and the mask together serve as the heart of a seven-bit analog-to-digital converter with a *serial* output.

Gray's narrative continues on page 1, describing the conventional binary coding of the mask shown in Figure 2A. Bear in mind that Gray consistently uses the word "digit" where we would now say "bit":

> It is a characteristic of the conventional binary number notation that a value change of unity may be reflected in the binary number notation by a simultaneous change in several of the digits. Thus, for example, with four digits the number seven is represented by 0111 while the next number, eight, is represented by 1000. In the course of changing the value by one unit, each of the four digits has been changed.
>
> This characteristic of the conventional binary number notation is duplicated in the coding mask of the Italian patent above referred to, and it is a consequence of the resulting arrangement of the apertures that a wander of the beam in the course of its lateral sweep from the correct aperture row to the row immediately above or below it, may result in a coding error which is far greater than the beam deflection error. Various arrangements have been suggested for reducing this coding error by constraining the cathode beam to start its sweep at the correct row and to remain there throughout the sweep . . . All such arrangements involve complexity of apparatus in various degrees.

Gray describes how relatively minor errors in the beam's lateral trajectory ("wander") could cause the beam to graze the holes (or non-holes) of an adjacent row, possibly resulting in a large coding error due to the fact that bit-values often change greatly from one binary integer to the next. To use his example of seven (`0111`) versus eight (`1000`), one can imagine the beam properly deflected to the "eight" row and beginning its left-to-right sweep, successfully passing through the first hole of the "eight" row to create a "1" pulse at the capturing anode, but then a slight downward drift of the beam later in that same sweep might cause it to pass through the three holes of the "seven" row to create three more "1" pulses, the result being a serial data word of `1111` (fifteen) instead of either `1000` (eight) or `0111` (seven). In this example, a vertical alignment "wander" of just 1 "count" results in a digitized error of *seven* counts (i.e. fifteen instead of eight).

Next, Gray begins to describe how his new coding scheme is superior to conventional binary.

> It is a principal object of the present invention to reduce the coding errors in a pulse code transmission system. A more specific object is to provide a pulse code, and a corresponding coding mask, in which the coding error is never greater than the beam deflection error.
>
> Another object is to simplify the manufacture of a coding mask. `[page 5]`
>
> The above objects are attained in accordance with the invention by the selection of a novel form of the binary pulse code which differs from the conventional form by virtue of

a rearrangement of the pulses of the various pulse groups in such a way that the sequence of "on"-pulses [page 5]

and off-pulses which form a pulse group representing a particular signal amplitude differs in only one pulse position from the sequences representing the next lower amplitude and next higher amplitude. The new code is no longer similar to the accepted binary number notation. When it is embodied in a coding mask, the arrangement of the apertures of any one row differs from that of the rows above and below it in not more than one aperture. The resulting mask has certain valuable auxiliary properties and aspects. First, the smallest apertures, that is to say the apertures of the various rows in the column of least digital significance, are twice as large as the apertures of the conventional binary code mask. This makes for ease of manufacture. Second, all of the apertures of the mask, with the sole exception of the single aperture of greatest digital significance, are symmetrically arranged-about a transverse center line. This permits treating the largest digit aperture as an index of polarity only, rectifying the wave to be coded, sampling the rectified wave, and coding the samples using only one half of the coding mask. At the price of some increased complexity of associated apparatus, this greatly reduces the physical dimensions of the coder tube itself. Third, for signals of normal average amplitude range, the beam deflections seldom extend beyond the aperture of the column of second greatest digital significance, so that in the resulting coded signal, the $(n-1)^{th}$ pulse position is nearly always filled. This uniformly filled pulse position affords a convenient source of marker pulses for use in holding a receiver in correct synchronism with the transmitter. [page 6]

Later in the patent (beginning on page 7) the inventor describes his process for developing a "reflected binary" code counting sequence, and in doing so the full meaning of the word "reflected" becomes clear:

> The manner in which the primary reflected binary number system is built up will now be explained.
>
> First: write down the first two numbers in the 1-digit orthodox binary number system, thus:
>
> ```
> Zero     0
> One      1
> ```
>
> Note that the symbols differ in Only one digit. Second: below this array write its "reflection" in a transverse axis:
>
> ```
> Zero     0
> One      1
> ----------
>          1
>          0
> ```
>
> The symbols still differ in not more than one digit. However, the first is identical with the fourth and the second with the third.
>
> Third: to remove this ambiguity, add a second digit to the left of each symbol, 0 for the first two symbols and 1 for the last two, thus:
>
> ```
> Zero    00
> One     01
> Two     11
> Three   10
> ```
>
> and identify the last two symbols with the numbers "two" and "three." Each symbol is now unique and differs from those above and below it in not more than one digit. The array is a representation of the first four numbers in the primary 2-digit reflected binary number system.

The process is next repeated, giving –

First:

```
Zero    00
One     01
Two     11
Three   10
```

Second:

```
Zero    00
One     01
Two     11
Three   10
----------
        10
        11
        01
        00
```

Third:

```
Zero   000
One    001
Two    011
Three  010
Four   110
Five   111
Six    101
Seven  100
```

[page 7]

Gray continues his exposition on this "reflected binary" coding system on page 8 of the patent by describing alternative forms. In every case, though, the coding sequence exhibits the same important property of exhibiting just one bit-change per step, and also the property of all bits being symmetrical about the mid-point except the most-significant bit (MSB).

> Other secondary forms of the reflected binary code may be obtained in various ways. Vertical columns may be interchanged. For any such transposition the pattern may be split along any horizontal division line between rows, and the lower part placed above the upper part, to give a new pattern with the same properties as the primary one. Again, the initial process of building up the code by reflection may be modified, giving two alternatives for the 1-digit code, four for the 2-digit code, and so on. The four alternatives for the 2-digit code are tabulated below. Of these the first is the primary one discussed above, the others being variants.

Number	**Primary**	**First variant**	**Second variant**	**Third variant**
Zero	00	10	01	11
One	01	11	00	10
Two	11	01	10	00
Three	10	00	11	01

[page 8]

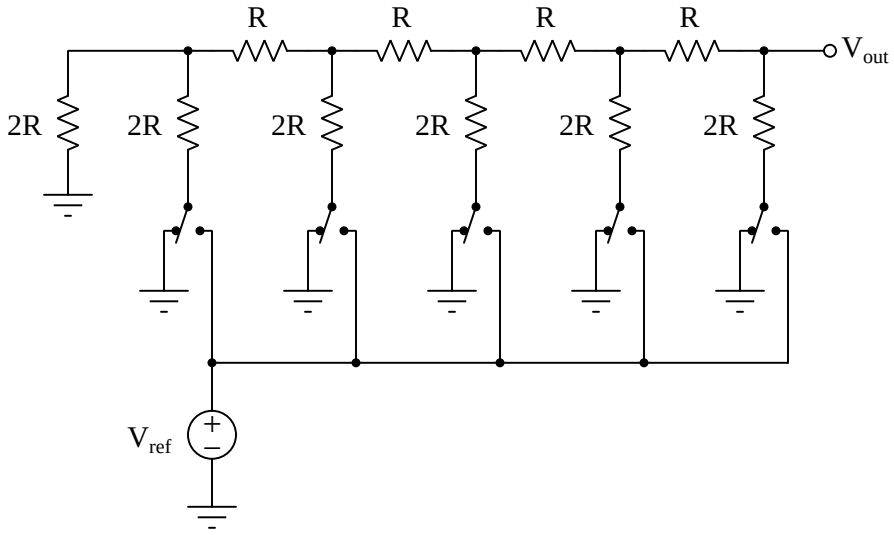
Chapter 5

Derivations and Technical References

This chapter is where you will find mathematical derivations too detailed to include in the tutorial, and/or tables and other technical reference material.

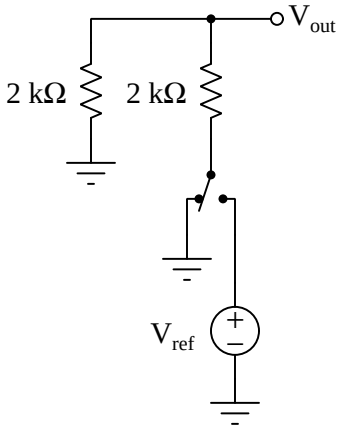
5.1 $R - 2R$ network analysis

A type of resistor network known as an *R-2R ladder* is often used in digital-to-analog conversion circuits. In the example schematic below, the ladder network has five discrete inputs, a reference voltage source (V_{ref}), and an output terminal (V_{out}). Setting the SPDT toggle switch positions to the "V_{ref}" or "Ground" positions affects the magnitude of V_{out} with a binary-weighted pattern, as each switch closer to the V_{out} terminal has twice the effect as the one before it:

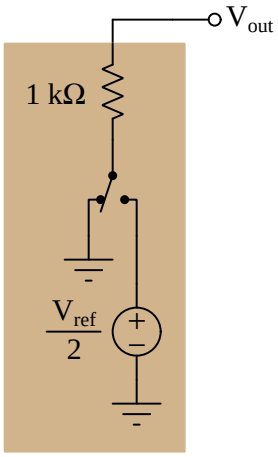


Understanding how the $R - 2R$ network achieves this binary weighting is not intuitive, though. Although it may be apprehended from first inspection that switches toward the left will have less effect on V_{out} than switches on the right, the 2^n weighting is certainly not obvious. Fortunately for us, though, this very feature of the $R - 2R$ ladder network lends itself well to helping explain how it functions, if only we apply Thévenin's Theorem to it.

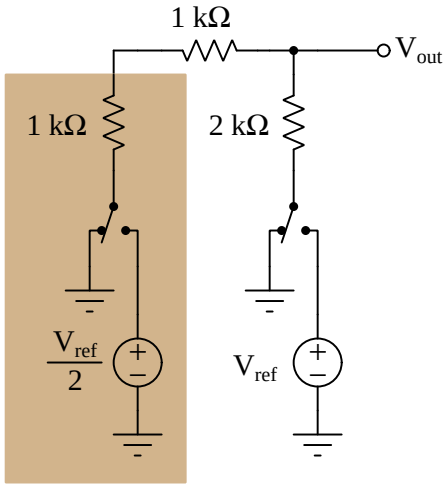
To begin, let's apply the problem-solving strategy of *simplifying the problem*. We will modify our ladder network to have just one bit (i.e. one toggle switch), and give the $2R$ resistors a definite resistance value that will be easy to apply in later computations:



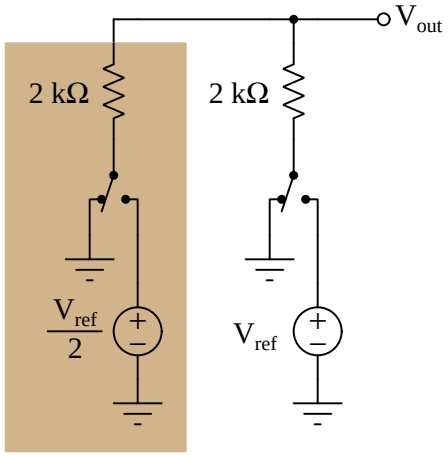
It should be immediately apparent that V_{out} will be 0 Volts with the toggle in the Ground position, and $\frac{V_{out}}{2}$ with the toggle in the V_{out} position. If we re-draw this network as a Thévenin equivalent circuit from the perspective of the V_{out} terminal, we arrive at the following result:



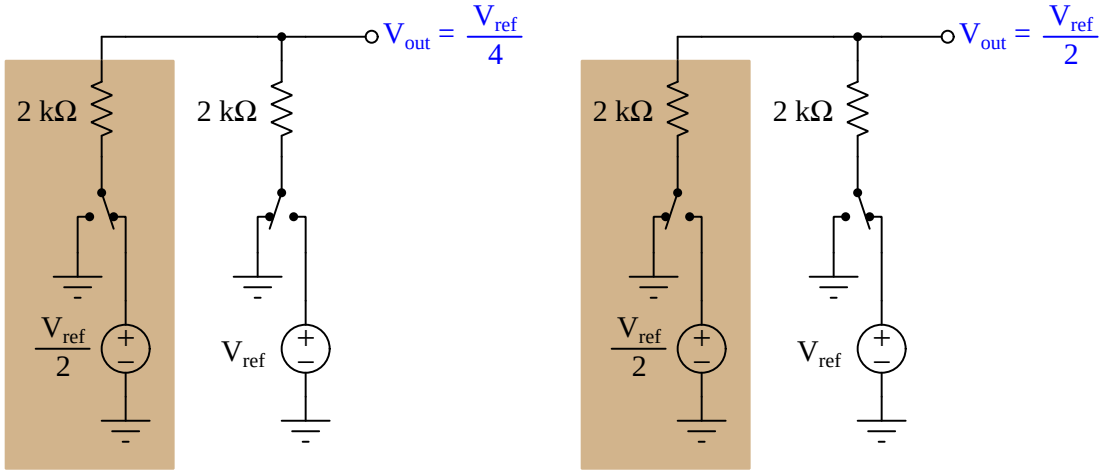
Adding one more bit to our ladder (i.e. V_{ref} at full strength, one R resistor (1 kΩ), one $2R$ resistor (2 kΩ), and another SPDT toggle switch), but keeping the previous Thévenin equivalent circuit in place:



We can see how the added 1 kΩ resistor is directly in series with the Thévenin equivalent network's R_{Th} of 1 kΩ, allowing us to consolidate those two resistances into a single resistance and thereby simplify the network again:



With both toggle switches in their Ground positions, V_{out} will of course be 0 Volts. If we analyze the circuit with each of the switches in the V_{ref} position (one at a time), we begin to see a binary progression of V_{out} values:

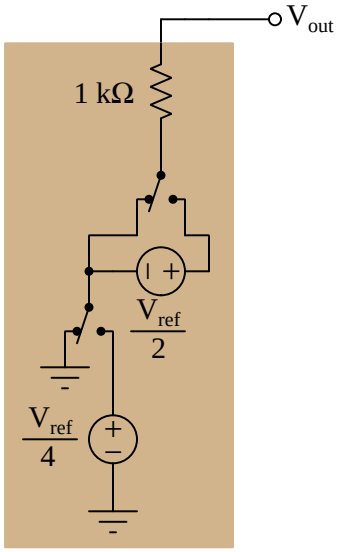


Flipping both toggle switches to their V_{ref} positions will yield an output voltage of $\frac{3}{4}V_{ref}$. Two fairly easy strategies lead us to this same conclusion: (1) if we consider the $2R-2R$ network to be a *passive averager*, with both sources active V_{out} must become the average of $\frac{V_{ref}}{2}$ and V_{ref}, which is of course $\frac{3}{4}V_{ref}$; (2) we may recognize this as being a linear network and therefore amenable to the Superposition Theorem, thus the output voltage with both sources active must be the superposition of their individual contributions to V_{out}, therefore $V_{out} = \frac{V_{ref}}{4} + \frac{V_{ref}}{2} = \frac{3}{4}V_{ref}$.

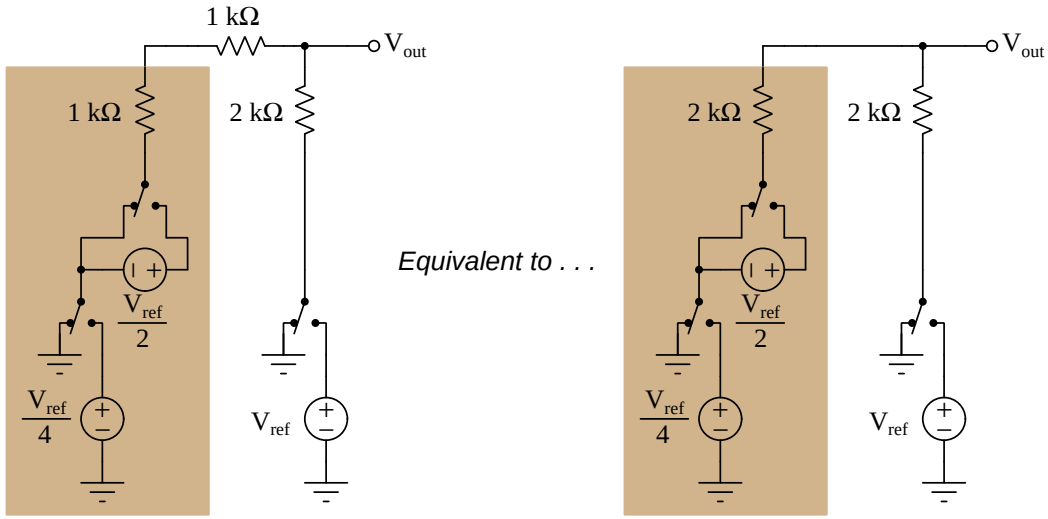
The binary progression of V_{out} values for this 2-bit $R-2R$ ladder network are clearly evident when we tabulate them in reference to the switch positions and show each V_{out} value as a quarter-fraction of V_{ref}:

MSB switch	**LSB switch**	V_{out}
Ground	Ground	$\frac{0}{4}V_{ref}$
Ground	V_{ref}	$\frac{1}{4}V_{ref}$
V_{ref}	Ground	$\frac{2}{4}V_{ref}$
V_{ref}	V_{ref}	$\frac{3}{4}V_{ref}$

If we Thévenize this whole network, we arrive at an equivalent consisting of two voltage sources, two switches, and a single 1 kΩ series resistance:



Adding another bit (stage) to our ladder network should look familiar to us: one more R resistor, one more $2R$ resistor, one more V_{ref}, and another SPDT toggle switch. Just like last time, the series combination of the new 1 kΩ resistor and the Thévenin equivalent resistance of 1 kΩ creates a new equivalent resistance of 2 kΩ for all preceding bits of the ladder:



If we flip the new MSB toggle switch to the V_{ref} position while keeping the other two switches in

their Ground positions, we will find V_{out} is equal to $\frac{V_{ref}}{2}$ because the Thévenin equivalent resistance of the network representing all lower-order bits is 2 kΩ and this functions as a 1:2 voltage divider with the new bit's own 2 kΩ resistance. Likewise, if we move the new MSB's switch to the Ground position and try toggling the other switches, we will find that each of their effects at the V_{out} terminal has been cut in half by the same voltage divider.

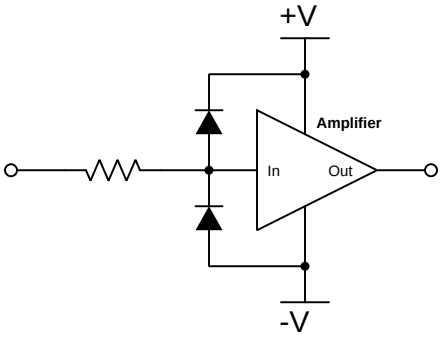
This is the general characteristic of an $R - 2R$ ladder network: with each addition of a new bit (stage) the MSB's effect remains unchanged but all lower-order bits' effects are attenuated by half. The number of stages is irrelevant, since the $R - 2R$ pattern maintains a consistent Thévenin equivalent resistance of $2R$ for any number of lower-order stages. We may continue this compounding of $R - 2R$ and V_{ref} stages as far as we might like and the pattern still holds true.

5.2 Protecting amplifier inputs from over-voltage

Electronic amplifier circuits are extremely useful devices, comprising portions of many practical integrated-circuit (IC) electronic components such as digital logic gates, comparators, operational amplifiers, signal mixers, etc. Solid-state amplifier circuits use *transistors* to allow one electrical signal to control another, and these transistors tend to be susceptible to damage from excessive applied signal voltage. In the case of MOSFET transistors, an excess of signal voltage may puncture the extremely thin layer of metal-oxide insulation separating the transistor's gate terminal from its current-carrying channel. In the case of bipolar (NPN, PNP) transistors, an excess of signal voltage may break down reverse-biased PN junctions inside the transistor, often causing them to fail in "shorted" states. Excessive applied signal voltage may also cause damage to bipolar transistors if forward-biasing PN junctions to the extent that they conduct high amounts of current which may lead to thermal damage.

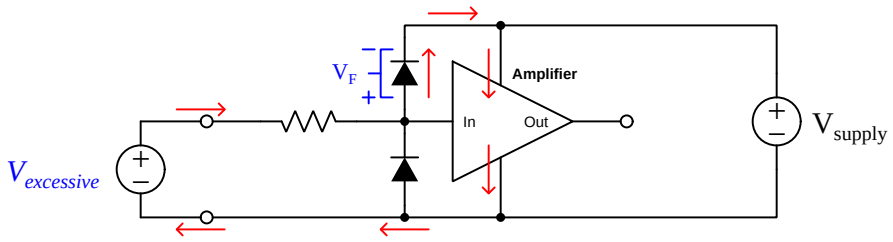
Sources of excessive signal voltage may be broadly categorized as *electro-static discharge* (ESD) or *electrical over-stress* (EOS). ESD happens when some object external to the circuit (including human bodies) accumulates an electro-static charge in its own capacitance, and this capacitively-stored charge is suddenly send to the circuit where it drops a voltage across circuit components high enough to cause damage. EOS is a more broad description of any over-voltage or over-current condition caused by one circuit sourcing energy to another, such as when an electrical test instrument is connected to a signal source that is too great for that instrument to handle.

A simple and effective way to limit both applied voltage and applied current to the input of any electronic amplifier is to connect a current-limiting resistor in series between the input terminal and the amplifier, as well as connect diodes between the input terminal and either DC power supply "rail" terminal:



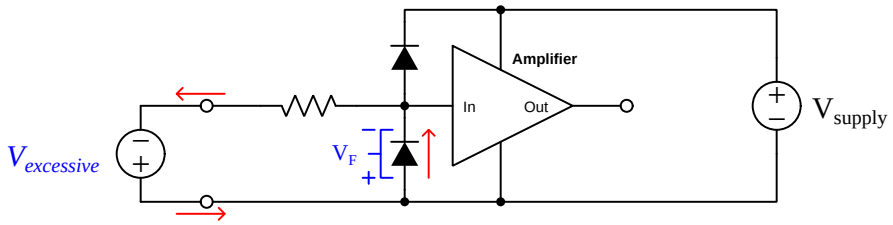
It matters not how the amplifier itself is internally constructed, nor what its larger purpose is – whether it be part of a digital logic gate, a comparator, an operational amplifier, or something even more complex. The purpose of these two diodes and the series resistor is to prevent any component within the amplifier from experiencing either too much voltage and/or too much current.

To understand how this protection network functions, consider the following circumstance where an excessive positive potential is applied to the input terminal by some external source. Here, "excessive" is defined as any voltage greater in magnitude than the DC power source energizing the amplifier:



The upper diode of the protection network forward-biases to permit current from the offending source to pass into the positive power bus of the amplifier. So long as the limiting resistor restricts this resulting current to a value less than what the amplifier requires for its normal operation, the voltage between the amplifier's input terminal and the negative rail cannot exceed the DC power supply's voltage plus the diode's forward-voltage drop ($V_{supply} + V_F$). With a normal silicon diode this means a voltage no greater than 0.7 Volts plus the positive DC rail voltage. If Schottky diodes are used in the protection network, it means a voltage no greater than 0.4 Volts beyond the positive rail. Such mild over-voltage conditions are unlikely to cause damage to any of the amplifier's internal transistors.

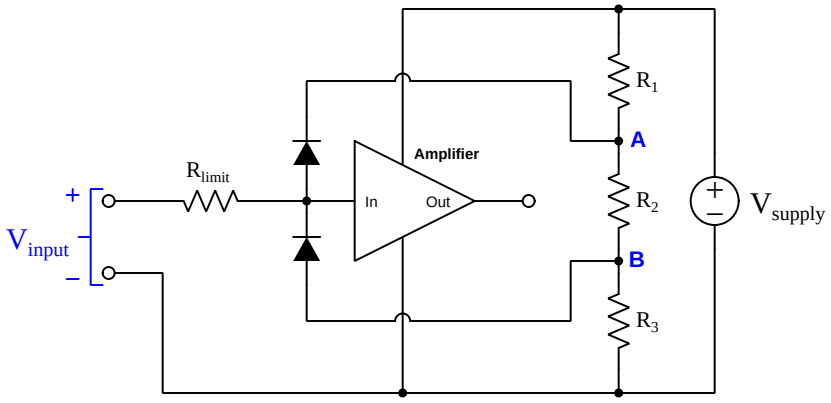
Similarly, if an excessive negative voltage is applied by an external source ("excessive" being any significant potential below that of the negative DC power supply rail):



Here the lower protection diode forward-biases and permits current from the external source to pass through the limiting resistor. While this occurs, the amplifier will never see more than that protection diode's forward voltage drop (V_F) between its input terminal and the negative DC rail bus, which again is unlikely to damage any transistor inside the amplifier.

Such protection networks are often found inside of integrated circuits, sometimes without a series protection resistor in their most basic forms. Protection diodes are standard for CMOS digital logic gates to help protect their constituent MOSFET transistor gates from damage from ESD.

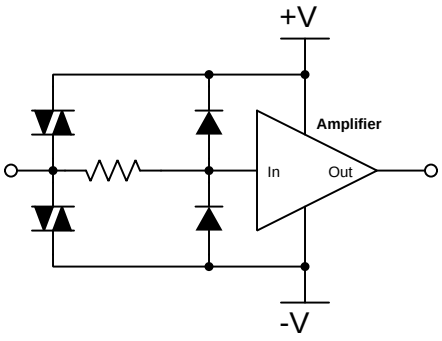
For some devices it is not sufficient for a diode network to clamp the input voltage to some value(s) slightly in excess of the device's power supply rail potential(s). For example, a device operating on a 5 Volt DC supply may tolerate input voltages no lower than zero and no greater than 5 Volts. A relatively simple solution to this problem is to provide the diode network with a set of "protection rails" just shy of the actual power supply rail potentials. Consider the following example as an illustration of this technique:



Here, resistors R_1, R_2, and R_3 form a voltage divider to produce two electrical potentials (labeled **A** and **B**), each one diode forward-voltage drop shy of the nearest rail potentials. For example, if the DC power supply output 5 Volts and the two protection diodes each dropped 0.7 Volts when forward-biased, the voltage divider would need to be designed such that point **A** was +4.3 Volts and point **B** was +0.7 Volts with respect to the DC supply's negative terminal. This would prompt the upper diode to turn on and clamp the amplifier's input potential to +5 Volts if ever the input potential exceeded +5 Volts, the limit resistor dropping the rest. Likewise, the lower diode would turn on if ever the input terminal's potential fell below the power supply's negative rail, clamping the amplifier's input terminal potential to exactly the same as that negative rail with R_{limit} dropping the rest.

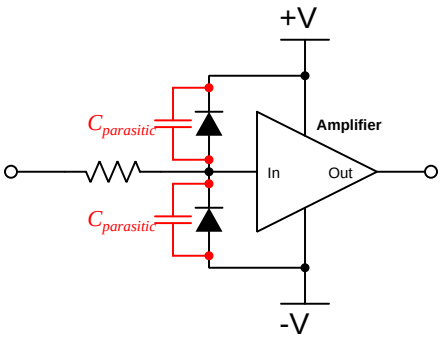
An important caveat to this strategy is that the voltage divider resistors R_1 and R_3 must be relatively small compared to the resistance of R_{limit} in order to ensure that those "protection rail" potentials **A** and **B** do not vary significantly when the protection diodes begin to conduct. A general engineering design principle here is to size R_{limit} at least ten times greater than either R_1 or R_3. If we size the resistors properly, this voltage-divider-based protection strategy may even be made *adjustable* by incorporating potentiometers into the voltage-divider network.

When protection must be provided against extraordinarily strong sources, additional protection may be added in their form of *DIACs* connected between each input terminal and power supply rail:



DIACs are classified as *thyristors*, because once triggered into an electrically conductive state by sufficiently high voltage (and/or sufficiently high *rate of change* of voltage, $\frac{dV}{dt}$) they will remain "on" so long as sufficient "holding" current passes through them even if voltage falls far below the initial triggering value. In summary, a thyristor acts as a very effective snubbing device to tame over-voltage conditions, essentially acting as a near-short to that offending source.

Protection networks, however, are not without their disadvantages. Chief among these is their tendency to adversely affect the signal being sensed by the amplifier being protected. Ideally a protection network should not corrupt the sensed signal at all, and only come into play if and when that signal becomes strong enough to pose threat of damage to the amplifier, but this would only be true if the protection diodes (and DIACs) had no parasitic properties when non-conducting. This is unfortunately untrue, as both diodes and DIACs exhibit *parasitic capacitance* which not only has the effect of storing electrical charge that the amplifier may interpret as a voltage that should not be present, but along with the protection resistor will form a *low-pass filter network* preventing the amplifier from being able to fully sense rapidly-changing input signals:



In other words, even with applied signals weak enough to pose absolutely no threat of harm to the amplifier, the capacitance inherent to the protection diodes will conspire with the series resistor

to "slow down" rates of rise and fall for any voltage arriving at the amplifier's input, thus making the amplifier "think" the signal isn't changing as rapidly as it really is.

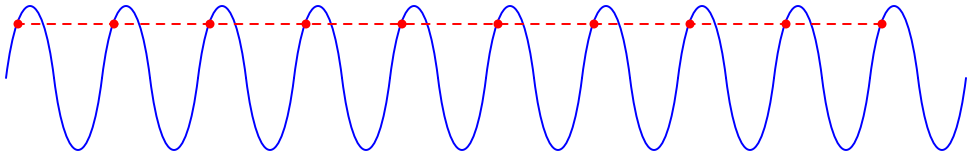
In order to minimize the effects of parasitic capacitance within the protection diodes, we must choose diodes with as little of that capacitance as possible and also select a protection resistor with as low a value as possible that still limits maximum current to a value safe for the amplifier.

5.3 A practical use for aliasing

Normally we consider aliasing to be undesirable, as it results in an apparent signal frequency that is completely incorrect. However, the phenomenon of aliasing does have practical use if we apply it well. If the signal of interest has a frequency that is too high for our instruments to reliably display, it is possible to exploit aliasing to "down-shift" that signal's frequency to a much lower range suitable for measurement. The key assumption for this technique to work is that the signal of interest is *periodic* (i.e. it repeats the same wave-shape with every cycle).

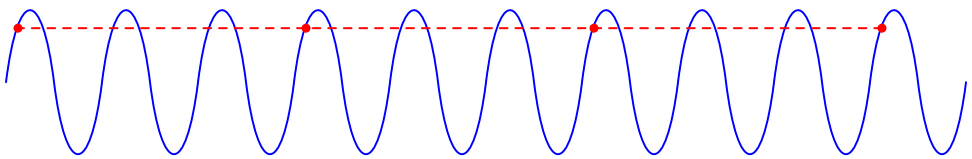
To understand how this might work, first imagine a condition where we sample a signal exactly once per cycle:

Sample interval exactly equal to one wave cycle

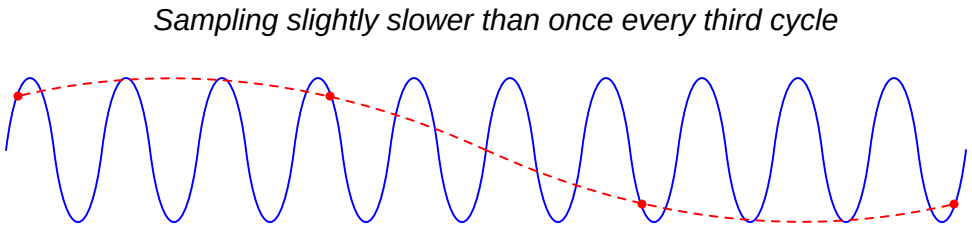


The result of sample at this exact rate would be that the receiving instrument (e.g. ADC) sees an unchanging value, equivalent to DC. This same phenomenon occurs if we sample at a rate equal to the signal frequency or at some integer-factor (i.e. *subharmonic*) of that frequency. For example, we get the same result sampling at $\frac{1}{3}$ this rate:

Sampling every third cycle



However, notice what happens if we sample slightly slower than a once-every-third-cycle rate:



Since the signal of interest is periodic, it doesn't matter if we sample several points within one cycle *or the same (equivalent) points spread out over many successive cycles*. This delayed sampling strategy will eventually capture the entire waveshape, albeit at an apparent frequency that is *much* slower than the actual signal frequency. If our sensing instruments cannot handle the original signal's high frequency, this aliased result may be perfectly suitable!

Illustrating this concept using a numerical example, consider a case where our signal of interest has a fundamental frequency of exactly 200 MHz and we set our ADC to sample at exactly 1 MHz. In such a scenario we would digitize the same analog voltage value of the signal every 200 cycles, resulting in a "DC" measurement. This is the same result as shown previously, either sampling at a rate exactly equal to the signal frequency or sampling every third cycle: in any of these cases, since the sample rate exactly matches an integer-factor (i.e. subharmonic) of the signal we end up sampling the exact same voltage value every time. This is not the result we want.

In order to sample more than just the same spot on the waveform every time, we must sample at a rate slightly slower than 1 MHz. A 200 MHz signal has a period of 5 nanoseconds, and if we wished to sample each cycle (period) a hundred times, our sample rate would have to be $\frac{5}{100}$ nanoseconds slower than the period of 1 MHz (i.e. 1.0000499999 microseconds rather than the 1.0 microsecond sampling period equivalent to 1 MHz). The aliased signal would have its own period of 100 times this sampling rate, or 100.00499999 microseconds. This yields an aliased signal frequency of only 9999.995 Hz, but with a wave-shape faithfully reproducing the original 200 MHz signal at a resolution of $\frac{1}{100}$ of a cycle (i.e. effectively 100 samples per cycle of the 200 MHz waveform, equivalent to an ultra-fast measurement system sampling 20 *billion* times per second, while only sampling slightly slower than 1 million times per second!).

The caveat, of course, is that in order for this aliased sampling scheme to faithfully reproduce the wave-shape, the original signal of interest must be *periodic* (i.e. repeating its same shape cycle after cycle after cycle).

Chapter 6

Questions

This learning module, along with all others in the ModEL collection, is designed to be used in an inverted instructional environment where students independently read[1] the tutorials and attempt to answer questions on their own *prior* to the instructor's interaction with them. In place of lecture[2], the instructor engages with students in Socratic-style dialogue, probing and challenging their understanding of the subject matter through inquiry.

Answers are not provided for questions within this chapter, and this is by design. Solved problems may be found in the Tutorial and Derivation chapters, instead. The goal here is *independence*, and this requires students to be challenged in ways where others cannot think for them. Remember that you always have the tools of *experimentation* and *computer simulation* (e.g. SPICE) to explore concepts!

The following lists contain ideas for Socratic-style questions and challenges. Upon inspection, one will notice a strong theme of *metacognition* within these statements: they are designed to foster a regular habit of examining one's own thoughts as a means toward clearer thinking. As such these sample questions are useful both for instructor-led discussions as well as for self-study.

[1] Technical reading is an essential academic skill for any technical practitioner to possess for the simple reason that the most comprehensive, accurate, and useful information to be found for developing technical competence is in textual form. Technical careers in general are characterized by the need for continuous learning to remain current with standards and technology, and therefore any technical practitioner who cannot read well is handicapped in their professional development. An excellent resource for educators on improving students' reading prowess through intentional effort and strategy is the book textitReading For Understanding – How Reading Apprenticeship Improves Disciplinary Learning in Secondary and College Classrooms by Ruth Schoenbach, Cynthia Greenleaf, and Lynn Murphy.

[2] Lecture is popular as a teaching method because it is easy to implement: any reasonably articulate subject matter expert can talk to students, even with little preparation. However, it is also quite problematic. A good lecture always makes complicated concepts seem easier than they are, which is bad for students because it instills a false sense of confidence in their own understanding; reading and re-articulation requires more cognitive effort and serves to verify comprehension. A culture of teaching-by-lecture fosters a debilitating dependence upon direct personal instruction, whereas the challenges of modern life demand independent and critical thought made possible only by gathering information and perspectives from afar. Information presented in a lecture is ephemeral, easily lost to failures of memory and dictation; text is forever, and may be referenced at any time.

General challenges following tutorial reading

- Summarize as much of the text as you can in one paragraph of your own words. A helpful strategy is to explain ideas as you would for an intelligent child: as simple as you can without compromising too much accuracy.
- Simplify a particular section of the text, for example a paragraph or even a single sentence, so as to capture the same fundamental idea in fewer words.
- Where did the text make the most sense to you? What was it about the text's presentation that made it clear?
- Identify where it might be easy for someone to misunderstand the text, and explain why you think it could be confusing.
- Identify any new concept(s) presented in the text, and explain in your own words.
- Identify any familiar concept(s) such as physical laws or principles applied or referenced in the text.
- Devise a proof of concept experiment demonstrating an important principle, physical law, or technical innovation represented in the text.
- Devise an experiment to disprove a plausible misconception.
- Did the text reveal any misconceptions you might have harbored? If so, describe the misconception(s) and the reason(s) why you now know them to be incorrect.
- Describe any useful problem-solving strategies applied in the text.
- Devise a question of your own to challenge a reader's comprehension of the text.

General follow-up challenges for assigned problems

- Identify where any fundamental laws or principles apply to the solution of this problem, especially before applying any mathematical techniques.
- Devise a thought experiment to explore the characteristics of the problem scenario, applying known laws and principles to mentally model its behavior.
- Describe in detail your own strategy for solving this problem. How did you identify and organized the given information? Did you sketch any diagrams to help frame the problem?
- Is there more than one way to solve this problem? Which method seems best to you?
- Show the work you did in solving this problem, even if the solution is incomplete or incorrect.
- What would you say was the most challenging part of this problem, and why was it so?
- Was any important information missing from the problem which you had to research or recall?
- Was there any extraneous information presented within this problem? If so, what was it and why did it not matter?
- Examine someone else's solution to identify where they applied fundamental laws or principles.
- Simplify the problem from its given form and show how to solve this simpler version of it. Examples include eliminating certain variables or conditions, altering values to simpler (usually whole) numbers, applying a limiting case (i.e. altering a variable to some extreme or ultimate value).
- For quantitative problems, identify the real-world meaning of all intermediate calculations: their units of measurement, where they fit into the scenario at hand. Annotate any diagrams or illustrations with these calculated values.
- For quantitative problems, try approaching it qualitatively instead, thinking in terms of "increase" and "decrease" rather than definite values.
- For qualitative problems, try approaching it quantitatively instead, proposing simple numerical values for the variables.
- Were there any assumptions you made while solving this problem? Would your solution change if one of those assumptions were altered?
- Identify where it would be easy for someone to go astray in attempting to solve this problem.
- Formulate your own problem based on what you learned solving this one.

General follow-up challenges for experiments or projects

- In what way(s) was this experiment or project easy to complete?
- Identify some of the challenges you faced in completing this experiment or project.

- Show how thorough documentation assisted in the completion of this experiment or project.
- Which fundamental laws or principles are key to this system's function?
- Identify any way(s) in which one might obtain false or otherwise misleading measurements from test equipment in this system.
- What will happen if (component X) fails (open/shorted/etc.)?
- What would have to occur to make this system unsafe?

6.1 Conceptual reasoning

These questions are designed to stimulate your analytic and synthetic thinking[3]. In a Socratic discussion with your instructor, the goal is for these questions to prompt an extended dialogue where assumptions are revealed, conclusions are tested, and understanding is sharpened. Your instructor may also pose additional questions based on those assigned, in order to further probe and refine your conceptual understanding.

Questions that follow are presented to challenge and probe your understanding of various concepts presented in the tutorial. These questions are intended to serve as a guide for the Socratic dialogue between yourself and the instructor. Your instructor's task is to ensure you have a sound grasp of these concepts, and the questions contained in this document are merely a means to this end. Your instructor may, at his or her discretion, alter or substitute questions for the benefit of tailoring the discussion to each student's needs. The only absolute requirement is that each student is challenged and assessed at a level equal to or greater than that represented by the documented questions.

It is far more important that you convey your *reasoning* than it is to simply convey a correct answer. For this reason, you should refrain from researching other information sources to answer questions. What matters here is that *you* are doing the thinking. If the answer is incorrect, your instructor will work with you to correct it through proper reasoning. A correct answer without an adequate explanation of how you derived that answer is unacceptable, as it does not aid the learning or assessment process.

You will note a conspicuous lack of answers given for these conceptual questions. Unlike standard textbooks where answers to every other question are given somewhere toward the back of the book, here in these learning modules students must rely on other means to check their work. The best way by far is to debate the answers with fellow students and also with the instructor during the Socratic dialogue sessions intended to be used with these learning modules. Reasoning through challenging questions with other people is an excellent tool for developing strong reasoning skills.

Another means of checking your conceptual answers, where applicable, is to use circuit simulation software to explore the effects of changes made to circuits. For example, if one of these conceptual questions challenges you to predict the effects of altering some component parameter in a circuit, you may check the validity of your work by simulating that same parameter change within software and seeing if the results agree.

[3] *Analytical* thinking involves the "disassembly" of an idea into its constituent parts, analogous to dissection. *Synthetic* thinking involves the "assembly" of a new idea comprised of multiple concepts, analogous to construction. Both activities are high-level cognitive skills, extremely important for effective problem-solving, necessitating frequent challenge and regular practice to fully develop.

6.1.1 Reading outline and reflections

"Reading maketh a full man; conference a ready man; and writing an exact man" – Francis Bacon

Francis Bacon's advice is a blueprint for effective education: reading provides the learner with knowledge, writing focuses the learner's thoughts, and critical dialogue equips the learner to confidently communicate and apply their learning. Independent acquisition and application of knowledge is a powerful skill, well worth the effort to cultivate. To this end, students should read these educational resources closely, journal their own reflections on the reading, and discuss in detail their findings with classmates and instructor(s). You should be able to do all of the following after reading any instructional text:

☑ Briefly SUMMARIZE THE TEXT in the form of a journal entry documenting your learning as you progress through the course of study. Share this summary in dialogue with your classmates and instructor. Journaling is an excellent self-test of thorough reading because you cannot clearly express what you have not read or did not comprehend.

☑ Demonstrate ACTIVE READING STRATEGIES, including verbalizing your impressions as you read, simplifying long passages to convey the same ideas using fewer words, annotating text and illustrations with your own interpretations, working through mathematical examples shown in the text, cross-referencing passages with relevant illustrations and/or other passages, identifying problem-solving strategies applied by the author, etc. Technical reading is a special case of problem-solving, and so these strategies work precisely because they help solve any problem: paying attention to your own thoughts (metacognition), eliminating unnecessary complexities, identifying what makes sense, paying close attention to details, drawing connections between separated facts, and noting the successful strategies of others.

☑ Identify IMPORTANT THEMES, especially GENERAL LAWS and PRINCIPLES, expounded in the text and express them in the simplest of terms as though you were teaching an intelligent child. This emphasizes connections between related topics and develops your ability to communicate complex ideas to anyone.

☑ Form YOUR OWN QUESTIONS based on the reading, and then pose them to your instructor and classmates for their consideration. Anticipate both correct and incorrect answers, the incorrect answer(s) assuming one or more plausible misconceptions. This helps you view the subject from different perspectives to grasp it more fully.

☑ Devise EXPERIMENTS to test claims presented in the reading, or to disprove misconceptions. Predict possible outcomes of these experiments, and evaluate their meanings: what result(s) would confirm, and what would constitute disproof? Running mental simulations and evaluating results is essential to scientific and diagnostic reasoning.

☑ Specifically identify any points you found CONFUSING. The reason for doing this is to help diagnose misconceptions and overcome barriers to learning.

6.1.2 Foundational concepts

Correct analysis and diagnosis of electric circuits begins with a proper understanding of some basic concepts. The following is a list of some important concepts referenced in this module's full tutorial. Define each of them in your own words, and be prepared to illustrate each of these concepts with a description of a practical example and/or a live demonstration.

Digital signal

Analog signal

Serial versus Parallel data

Simplification as a problem-solving strategy

Resolution

Binary number

Hexadecimal

Count

Quantization error

Aliasing

Filter

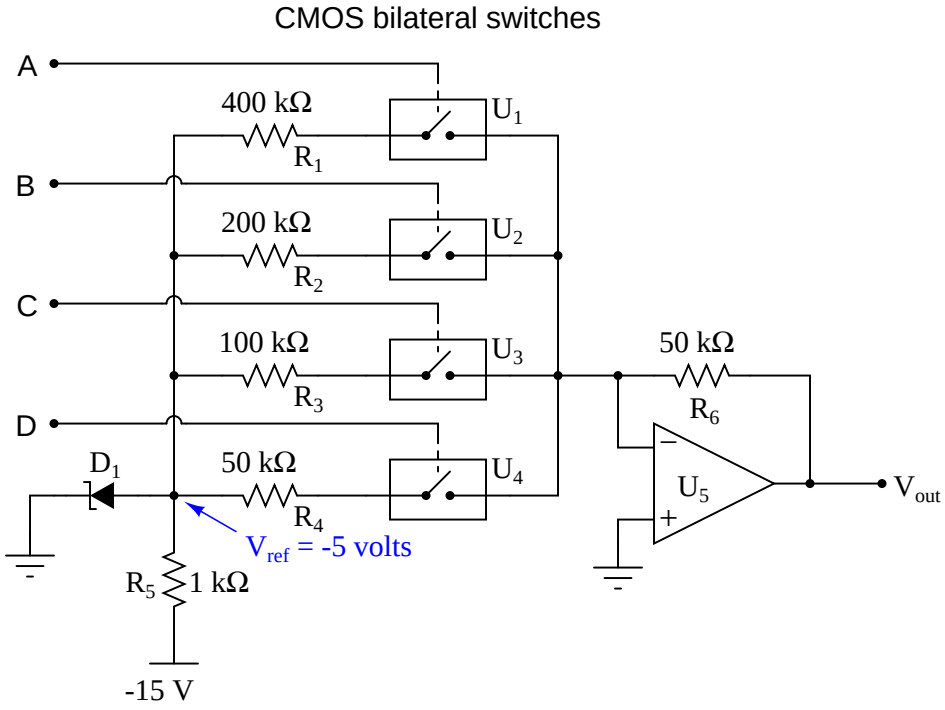
Sinusoidal decomposition (i.e. Fourier's Theorem)

Integration

Oversampling

6.1.3 DAC using bilateral switches

Explain how this *digital-to-analog* converter (DAC) circuit is supposed to function:

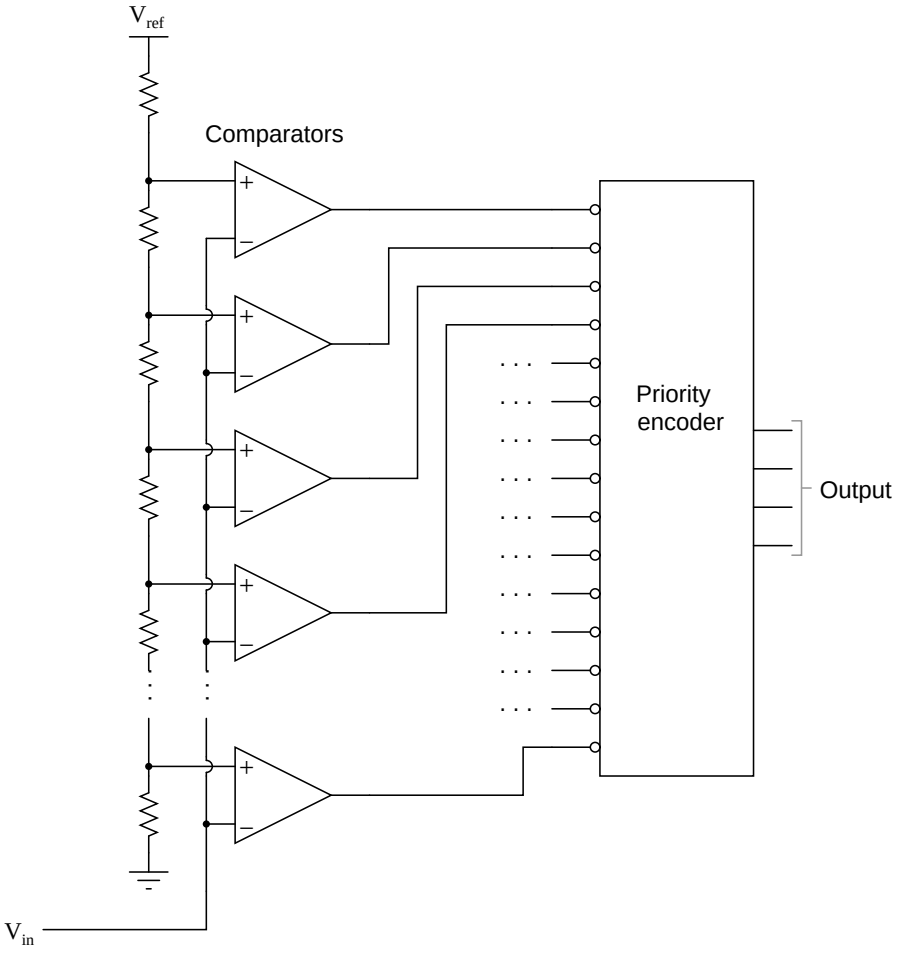


Challenges

- Identify the LSB and MSB bits on this DAC.

6.1.4 Flash ADC circuit using a priority encoder

The circuit shown here is a four-bit analog-to-digital converter (ADC). Specifically, it is a *flash* converter, so named because of its high speed:



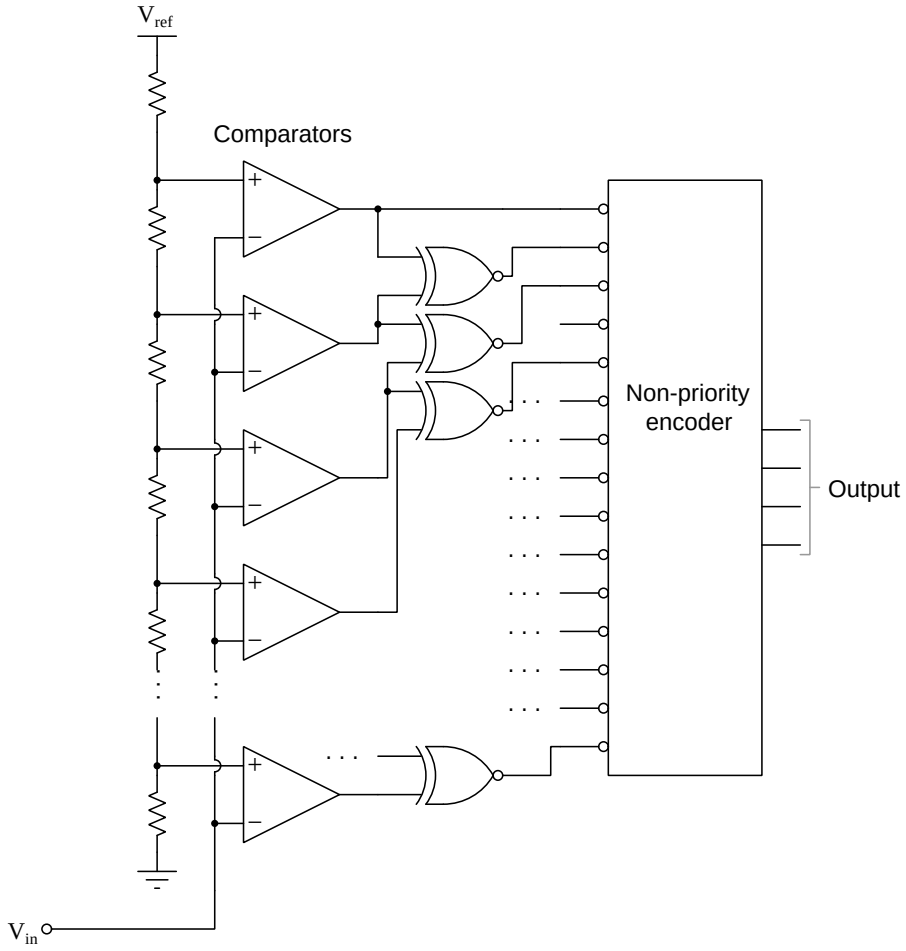
Explain the general principle by which this ADC functions, and also explain why we must use a *priority* encoder to encode the comparator outputs into a four-bit binary code, and not a regular encoder. What problem(s) would we have if we were to use a non-priority encoder in this ADC circuit?

Challenges

- Supposing V_{ref} is equal to +5 Volts, calculate the resolution of this ADC.
- What factor(s) limit the sampling rate of this ADC?

6.1.5 Flash ADC circuit using a non-priority encoder

The circuit shown here is a four-bit analog-to-digital converter (ADC). Specifically, it is a *flash* converter, so named because of its high speed:



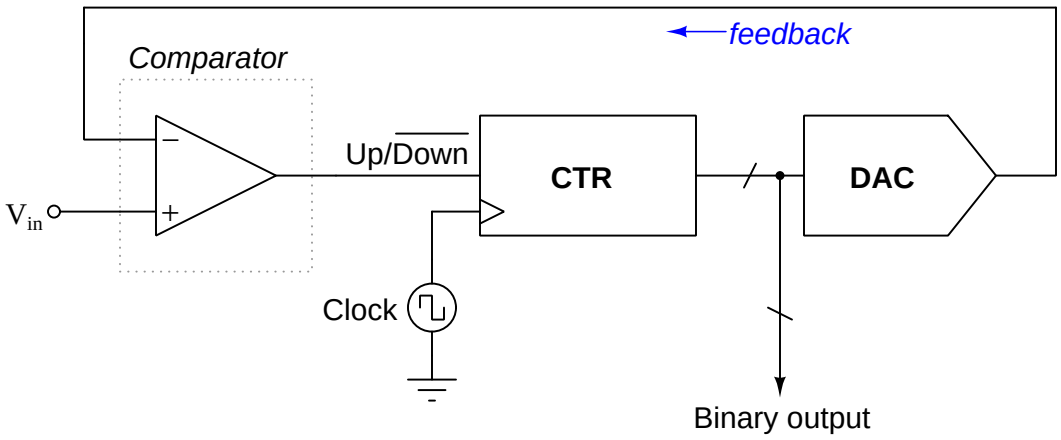
Explain the general principle by which this ADC functions, and also explain why are able to use a *non-priority* encoder while the version of this circuit without Exclusive-NOR gates requires a priority encoder for proper functioning. Also explain whether or not we could use a priority encoder with the XNOR gates, or if this design *requires* a non-priority encoder.

Challenges

- Supposing V_{ref} is equal to +15 Volts, calculate the resolution of this ADC.
- What factor(s) limit the sampling rate of this ADC?

6.1.6 Tracking ADC

Explain the operating principle of this analog-to-digital converter circuit, usually referred to as a *tracking* converter:

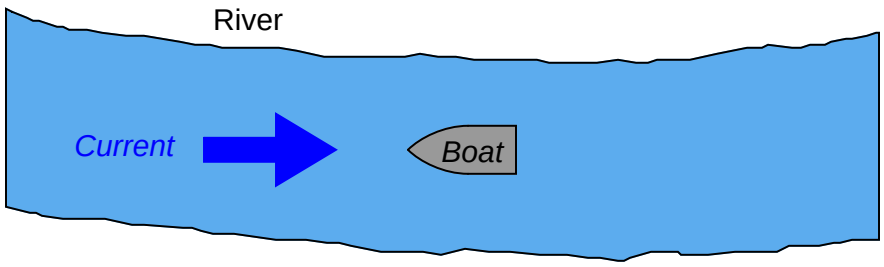


Challenges

- Explain why this form of ADC is not very effective at following fast-changing input signals.

6.1.7 Boat analogy for a Delta-Sigma ADC

Examine this vertical ("bird's eye") view of a boat resisting a river's current:



Suppose the driver of this boat does not own an anchor, and furthermore that the only form of propulsion is an electric "trolling" motor with a simple on/off switch offering no variable-speed control of the thrust motor. By repeatedly pulsing the motor on and off at just the right intervals, it should be possible for the boat to maintain its position relative to the riverbanks against the river's flow.

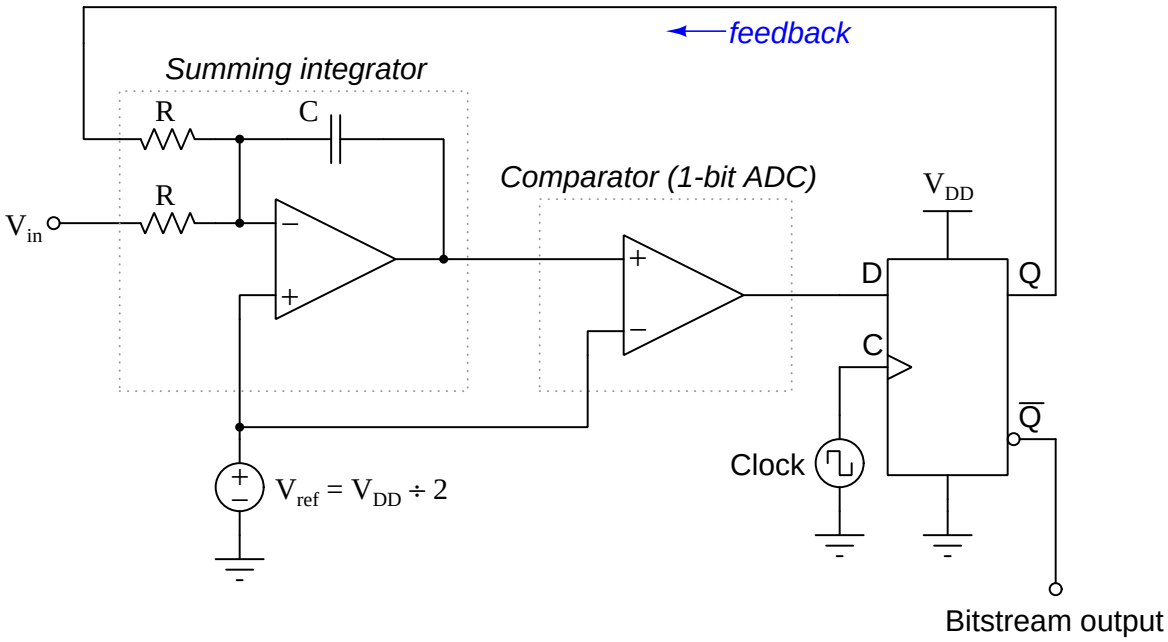
If we happen to know that the boat is actually holding position in the middle of the river by trolling motor power alone, the pattern of on/off switch actuations should tell us something about the speed of the river. Perform a couple of "thought experiments" where you imagine what the driver of the boat would have to do with the motor's on/off switch to maintain position against a fast current, versus against a slow current. What relationship do you see between the switch actuations and the speed of the current? How does this relate to the operation of a Delta-Sigma ADC?

Challenges

- ???.
- ???.
- ???.

6.1.8 Delta-Sigma ADC bitstream values

The *Delta-Sigma* or *Sigma-Delta* analog-to-digital converter works on the principle of *oversampling*, whereby a low-resolution ADC repeatedly samples the input signal in a feedback loop. In many cases, the ADC used is nothing more than a comparator (i.e. a 1-bit ADC), the output of this ADC subtracted from the input signal and integrated over time in an attempt to achieve an average of 0 Volts at the output of the integrator. The result is a "bitstream" of serial data which may be filtered and converted to a binary word of multiple bits:



Identify the bit states of this ADC's output data stream during each of the following input voltage conditions:

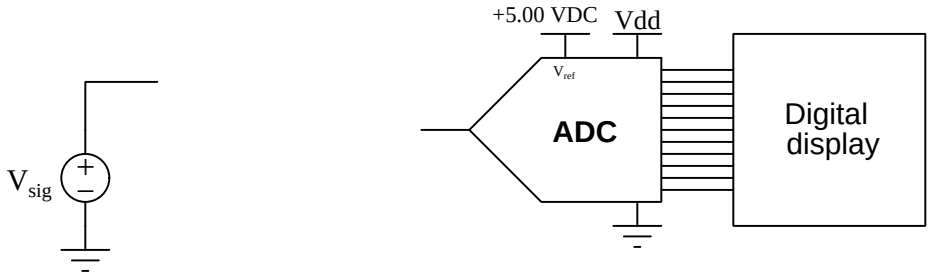
- V_{in} = 0 Volts ; bitstream =
- $V_{in} = V_{DD}$; bitstream =
- $V_{in} = V_{ref}$; bitstream =

Challenges

- Suppose this ADC outputs a bitstream with this pattern: `011110111101111`. Does this represent an input signal voltage greater than V_{ref}, less than V_{ref}, equal to V_{ref}, or is there not enough information given here to determine?

6.1.9 Scaling and overvoltage protection for ADC

Suppose we wished to use an ADC to display the value of some DC signal source, but that signal source's voltage was known to range from 0 to 24 Volts DC while the ADC's full range of measurement was fixed at 0 to 5.00 Volts DC:



Design and sketch an interfacing circuit that will take the input signal voltage and "scale" it proportionately to a range the ADC can handle, so that a normal range of signal voltage (i.e. 0 to 24 VDC) will not over-drive the ADC's input.

A practical feature you should add to your scaling circuit is overvoltage *protection* for the ADC, so that under no circumstances – even a V_{sig} value exceeding its normal maximum of 24 VDC – will the ADC's input be subjected to more than 5.5 Volts DC.

Challenges

- Suppose someone designed a scaling network that scaled 0-24 VDC down to 0-2 VDC. Would this work well?
- Calculate the effective resolution of the ADC if the scaling circuit scales 24 VDC to exactly 5 VDC, assuming a 10-bit ADC.
- Re-design the voltage divider so that a V_{sig} voltage of 30 Volts results in a count value of `0x1C7` from this 10-bit ADC.

6.2 Quantitative reasoning

These questions are designed to stimulate your computational thinking. In a Socratic discussion with your instructor, the goal is for these questions to reveal your mathematical approach(es) to problem-solving so that good technique and sound reasoning may be reinforced. Your instructor may also pose additional questions based on those assigned, in order to observe your problem-solving firsthand.

Mental arithmetic and estimations are strongly encouraged for all calculations, because without these abilities you will be unable to readily detect errors caused by calculator misuse (e.g. keystroke errors).

You will note a conspicuous lack of answers given for these quantitative questions. Unlike standard textbooks where answers to every other question are given somewhere toward the back of the book, here in these learning modules students must rely on other means to check their work. My advice is to use circuit simulation software such as SPICE to check the correctness of quantitative answers. Refer to those learning modules within this collection focusing on SPICE to see worked examples which you may use directly as practice problems for your own study, and/or as templates you may modify to run your own analyses and generate your own practice problems.

Completely worked example problems found in the Tutorial may also serve as "test cases[4]" for gaining proficiency in the use of circuit simulation software, and then once that proficiency is gained you will never need to rely[5] on an answer key!

[4] In other words, set up the circuit simulation software to analyze the same circuit examples found in the Tutorial. If the simulated results match the answers shown in the Tutorial, it confirms the simulation has properly run. If the simulated results disagree with the Tutorial's answers, something has been set up incorrectly in the simulation software. Using every Tutorial as practice in this way will quickly develop proficiency in the use of circuit simulation software.

[5] This approach is perfectly in keeping with the instructional philosophy of these learning modules: *teaching students to be self-sufficient thinkers*. Answer keys can be useful, but it is even more useful to your long-term success to have a set of tools on hand for checking your own work, because once you have left school and are on your own, there will no longer be "answer keys" available for the problems you will have to solve.

6.2.1 Miscellaneous physical constants

Note: constants shown in **bold** type are *exact*, not approximations. Values inside of parentheses show one standard deviation (σ) of uncertainty in the final digits: for example, the magnetic permeability of free space value given as $1.25663706212(19) \times 10^{-6}$ H/m represents a center value (i.e. the location parameter) of $1.25663706212 \times 10^{-6}$ Henrys per meter with one standard deviation of uncertainty equal to $0.0000000000019 \times 10^{-6}$ Henrys per meter.

Avogadro's number (N_A) = **6.02214076** $\times 10^{23}$ **per mole** (mol^{-1})

Boltzmann's constant (k) = **1.380649** $\times 10^{-23}$ **Joules per Kelvin** (J/K)

Electronic charge (e) = **1.602176634** $\times 10^{-19}$ **Coulomb** (C)

Faraday constant (F) = **96,485.33212...** $\times 10^{4}$ **Coulombs per mole** (C/mol)

Magnetic permeability of free space (μ_0) = $1.25663706212(19) \times 10^{-6}$ Henrys per meter (H/m)

Electric permittivity of free space (ϵ_0) = $8.8541878128(13) \times 10^{-12}$ Farads per meter (F/m)

Characteristic impedance of free space (Z_0) = 376.730313668(57) Ohms (Ω)

Gravitational constant (G) = $6.67430(15) \times 10^{-11}$ cubic meters per kilogram-seconds squared ($m^3/kg\text{-}s^2$)

Molar gas constant (R) = **8.314462618... Joules per mole-Kelvin** (J/mol-K) = 0.08205746(14) liters-atmospheres per mole-Kelvin

Planck constant (h) = **6.62607015** $\times 10^{-34}$ **joule-seconds** (J-s)

Stefan-Boltzmann constant (σ) = **5.670374419...** $\times 10^{-8}$ **Watts per square meter-Kelvin**^{4} ($W/m^2 \cdot K^4$)

Speed of light in a vacuum (c) = **299,792,458 meters per second** (m/s) = 186282.4 miles per second (mi/s)

Note: All constants taken from NIST data "Fundamental Physical Constants – Complete Listing", from `http://physics.nist.gov/constants`, National Institute of Standards and Technology (NIST), 2018 CODATA Adjustment.

6.2.2 Introduction to spreadsheets

A powerful computational tool you are encouraged to use in your work is a *spreadsheet*. Available on most personal computers (e.g. Microsoft Excel), *spreadsheet* software performs numerical calculations based on number values and formulae entered into cells of a grid. This grid is typically arranged as lettered columns and numbered rows, with each cell of the grid identified by its column/row coordinates (e.g. cell `B3`, cell `A8`). Each cell may contain a string of text, a number value, or a mathematical formula. The spreadsheet automatically updates the results of all mathematical formulae whenever the entered number values are changed. This means it is possible to set up a spreadsheet to perform a series of calculations on entered data, and those calculations will be re-done by the computer any time the data points are edited in any way.

For example, the following spreadsheet calculates average speed based on entered values of distance traveled and time elapsed:

	A	B	C	D
1	Distance traveled	46.9	Kilometers	
2	Time elapsed	1.18	Hours	
3	Average speed	= B1 / B2	km/h	
4				
5				

Text labels contained in cells `A1` through `A3` and cells `C1` through `C3` exist solely for readability and are not involved in any calculations. Cell `B1` contains a sample distance value while cell `B2` contains a sample time value. The formula for computing speed is contained in cell `B3`. Note how this formula begins with an "equals" symbol (`=`), references the values for distance and speed by lettered column and numbered row coordinates (`B1` and `B2`), and uses a forward slash symbol for division (`/`). The coordinates `B1` and `B2` function as *variables*[6] would in an algebraic formula.

When this spreadsheet is executed, the numerical value `39.74576` will appear in cell `B3` rather than the formula `= B1 / B2`, because 39.74576 is the computed speed value given 46.9 kilometers traveled over a period of 1.18 hours. If a different numerical value for distance is entered into cell `B1` or a different value for time is entered into cell `B2`, cell `B3`'s value will automatically update. All you need to do is set up the given values and any formulae into the spreadsheet, and the computer will do all the calculations for you.

Cell `B3` may be referenced by other formulae in the spreadsheet if desired, since it is a variable just like the given values contained in `B1` and `B2`. This means it is possible to set up an entire chain of calculations, one dependent on the result of another, in order to arrive at a final value. The arrangement of the given data and formulae need not follow any pattern on the grid, which means you may place them anywhere.

[6] Spreadsheets may also provide means to attach text labels to cells for use as variable names (Microsoft Excel simply calls these labels "names"), but for simple spreadsheets such as those shown here it's usually easier just to use the standard coordinate naming for each cell.

Common[7] arithmetic operations available for your use in a spreadsheet include the following:

- Addition (`+`)
- Subtraction (`-`)
- Multiplication (`*`)
- Division (`/`)
- Powers (`^`)
- Square roots (`sqrt()`)
- Logarithms (`ln()` , `log10()`)

Parentheses may be used to ensure[8] proper order of operations within a complex formula. Consider this example of a spreadsheet implementing the *quadratic formula*, used to solve for roots of a polynomial expression in the form of $ax^2 + bx + c$:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

	A	B
1	x_1	= (-B4 + sqrt((B4^2) - (4*B3*B5))) / (2*B3)
2	x_2	= (-B4 - sqrt((B4^2) - (4*B3*B5))) / (2*B3)
3	a =	9
4	b =	5
5	c =	-2

This example is configured to compute roots[9] of the polynomial $9x^2 + 5x - 2$ because the values of 9, 5, and -2 have been inserted into cells `B3`, `B4`, and `B5`, respectively. Once this spreadsheet has been built, though, it may be used to calculate the roots of *any* second-degree polynomial expression simply by entering the new a, b, and c coefficients into cells `B3` through `B5`. The numerical values appearing in cells `B1` and `B2` will be automatically updated by the computer immediately following any changes made to the coefficients.

[7] Modern spreadsheet software offers a bewildering array of mathematical functions you may use in your computations. I recommend you consult the documentation for your particular spreadsheet for information on operations other than those listed here.

[8] Spreadsheet programs, like text-based programming languages, are designed to follow standard order of operations by default. However, my personal preference is to use parentheses even where strictly unnecessary just to make it clear to any other person viewing the formula what the intended order of operations is.

[9] Reviewing some algebra here, a *root* is a value for x that yields an overall value of zero for the polynomial. For this polynomial ($9x^2 + 5x - 2$) the two roots happen to be $x = 0.269381$ and $x = -0.82494$, with these values displayed in cells `B1` and `B2`, respectively upon execution of the spreadsheet.

Alternatively, one could break up the long quadratic formula into smaller pieces like this:

$$y = \sqrt{b^2 - 4ac} \qquad z = 2a$$

$$x = \frac{-b \pm y}{z}$$

	A	B	C
1	x_1	= (-B4 + C1) / C2	= sqrt((B4^2) - (4*B3*B5))
2	x_2	= (-B4 - C1) / C2	= 2*B3
3	a =	9	
4	b =	5	
5	c =	-2	

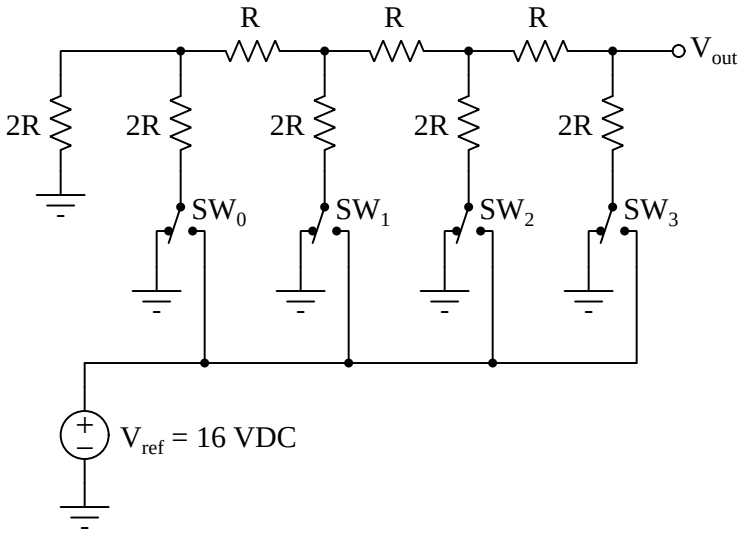
Note how the square-root term (y) is calculated in cell C1, and the denominator term (z) in cell C2. This makes the two final formulae (in cells B1 and B2) simpler to interpret. The positioning of all these cells on the grid is completely arbitrary[10] – all that matters is that they properly reference each other in the formulae.

Spreadsheets are particularly useful for situations where the same set of calculations representing a circuit or other system must be repeated for different initial conditions. The power of a spreadsheet is that it automates what would otherwise be a tedious set of calculations. One specific application of this is to simulate the effects of various components within a circuit failing with abnormal values (e.g. a shorted resistor simulated by making its value nearly zero; an open resistor simulated by making its value extremely large). Another application is analyzing the behavior of a circuit design given new components that are out of specification, and/or aging components experiencing drift over time.

[10] My personal preference is to locate all the "given" data in the upper-left cells of the spreadsheet grid (each data point flanked by a sensible name in the cell to the left and units of measurement in the cell to the right as illustrated in the first distance/time spreadsheet example), sometimes coloring them in order to clearly distinguish which cells contain entered data versus which cells contain computed results from formulae. I like to place all formulae in cells below the given data, and try to arrange them in logical order so that anyone examining my spreadsheet will be able to figure out *how* I constructed a solution. This is a general principle I believe all computer programmers should follow: *document and arrange your code to make it easy for other people to learn from it.*

6.2.3 $R - 2R$ ladder voltage calculations

Determine the voltage output by the following R-2R ladder network given the switch states shown in the table:



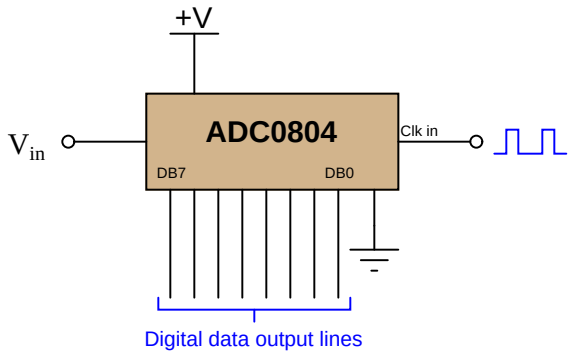
- SW_0 = Gnd ; SW_1 = Gnd ; SW_2 = Gnd ; $\mathrm{SW}_3 = V_{ref}$; V_{out} =
- SW_0 = Gnd ; SW_1 = Gnd ; $\mathrm{SW}_2 = V_{ref}$; SW_3 = Gnd ; V_{out} =
- SW_0 = Gnd ; $\mathrm{SW}_1 = V_{ref}$; SW_2 = Gnd ; SW_3 = Gnd ; V_{out} =
- $\mathrm{SW}_0 = V_{ref}$; SW_1 = Gnd ; SW_2 = Gnd ; SW_3 = Gnd ; V_{out} =

Challenges

- Given the fact that an $R - 2R$ resistor network is inherently linear, we may readily apply the *Superposition Theorem* to figure out what happens when more than one switch is moved to the V_{ref} position. Explain how you would apply Superposition to determine all output voltages for all possible combinations of switch positions.

6.2.4 ADC0804 signal values

The ADC0804 is an example of an integrated circuit analog-to-digital converter (ADC), converting an analog input voltage signal into an 8-bit binary output:



When operated from a 5.0 Volt DC power supply in its simplest mode, the ADC0804 converts any DC input voltage between 0.0 Volts and 5.0 Volts into an 8-bit number at the command of a clock pulse. A 0.0 Volt input yields a binary output (or "count") of `00000000`, of course, while a 5.0 Volt input yields a count of `11111111`.

Complete this table of numbers, relating various DC input voltages with count values (expressed in binary, hex, and decimal) for an ADC0804 having a unipolar input range of 0.0 to 5.0 Volts DC:

DC input voltage	Binary count	Hex count	Decimal count
0.0 Volts	`00000000`		
	`00110011`		51
2.2 Volts		`70`	
		`B3`	179
	`11001100`	`CC`	
5.0 Volts	`11111111`		

Challenges

- Explain why the "count" value generated by an analog-to-digital converter must be an integer number. For example, explain why a count value of 143 might be valid, but a count value of 143.7 is not.

6.2.5 Spreadsheet simulation of 8-bit ADC

A computer spreadsheet program may be used as a simulator for an analog-digital converter, taking an analog voltage signal value and converting it into a digital "count" value.

Begin creating your own spreadsheet by following the format shown below, allowing anyone to enter an analog input value in Volts, while the spreadsheet calculates the "count" value and displays it in decimal, binary, and hexadecimal formats. Note that the yellow and blue shading in this example spreadsheet is strictly for aesthetic value (distinguishing input values from calculated values) and is not necessary for the spreadsheet to function:

	1	2	3	4	5
1	Input (V)		Counts (decimal) =	179	
2	3.5		Counts (binary) =	10110011	
3			Counts (hex) =	B3	
4					
5					

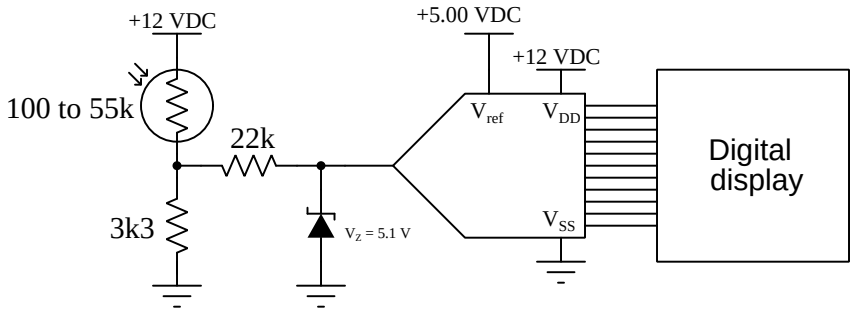
Assume a 0 to 5 Volt (unipolar) analog input range, and a 8-bit (i.e. `00` to `FF` hex) digital output range.

Challenges

- How could you modify the spreadsheet to have its analog input range be user-adjustable? In other words, allowing anyone to simulate the performance of a 0 to 10 Volt range or 0 to 20 Volt range without having to modify any of the formulae in the spreadsheet?
- How could you *test* your spreadsheet cable-length calculator for accuracy (to verify you haven't made any mistakes) once you've entered all your equations?

6.2.6 Re-design input network for an ADC circuit

Re-design this circuit so that with the lowest photocell resistance value the ADC input will see exactly 5.00 Volts, assuming we replace the 12 Volt supply with a 6 Volt supply.



After doing this, calculate the count value output by the 11-bit ADC when the photocell is fully dark.

Challenges

- At what photocell resistance value will the original circuit's zener diode begin functioning as a clamping circuit to protect the ADC from over-voltage?

6.3 Diagnostic reasoning

These questions are designed to stimulate your deductive and inductive thinking, where you must apply general principles to specific scenarios (deductive) and also derive conclusions about the failed circuit from specific details (inductive). In a Socratic discussion with your instructor, the goal is for these questions to reinforce your recall and use of general circuit principles and also challenge your ability to integrate multiple symptoms into a sensible explanation of what's wrong in a circuit. Your instructor may also pose additional questions based on those assigned, in order to further challenge and sharpen your diagnostic abilities.

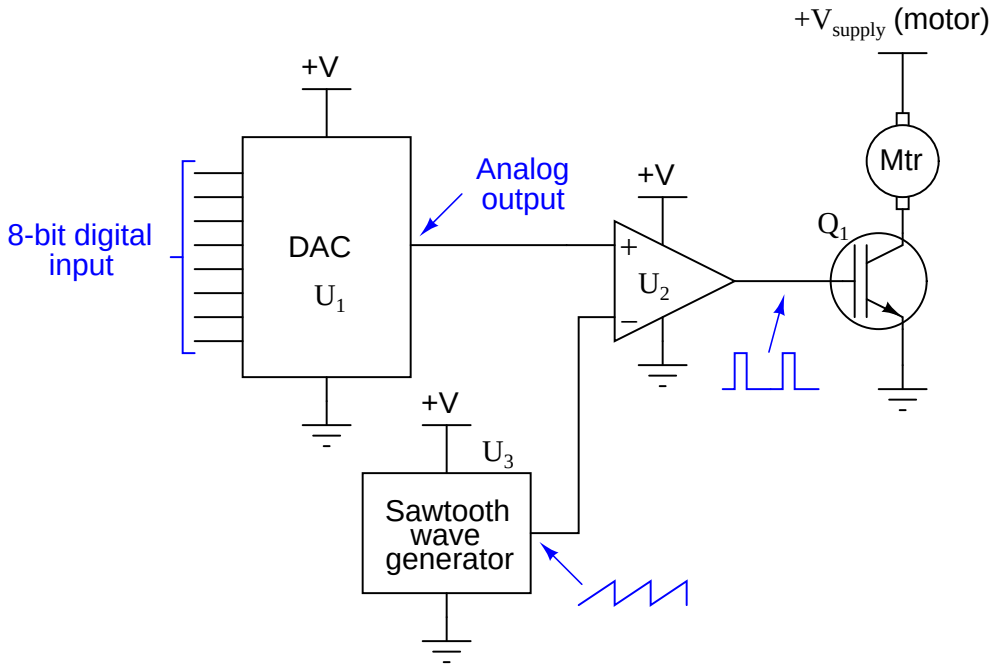
As always, your goal is to fully *explain* your analysis of each problem. Simply obtaining a correct answer is not good enough – you must also demonstrate sound reasoning in order to successfully complete the assignment. Your instructor's responsibility is to probe and challenge your understanding of the relevant principles and analytical processes in order to ensure you have a strong foundation upon which to build further understanding.

You will note a conspicuous lack of answers given for these diagnostic questions. Unlike standard textbooks where answers to every other question are given somewhere toward the back of the book, here in these learning modules students must rely on other means to check their work. The best way by far is to debate the answers with fellow students and also with the instructor during the Socratic dialogue sessions intended to be used with these learning modules. Reasoning through challenging questions with other people is an excellent tool for developing strong reasoning skills.

Another means of checking your diagnostic answers, where applicable, is to use circuit simulation software to explore the effects of faults placed in circuits. For example, if one of these diagnostic questions requires that you predict the effect of an open or a short in a circuit, you may check the validity of your work by simulating that same fault (substituting a very high resistance in place of that component for an open, and substituting a very low resistance for a short) within software and seeing if the results agree.

6.3.1 Faults in a motor speed control circuit

This is a digitally-set motor speed controller circuit, using PWM to modulate power to the motor. Predict how the operation of this circuit will be affected as a result of the following faults. Consider each fault independently (i.e. one at a time, no multiple faults):



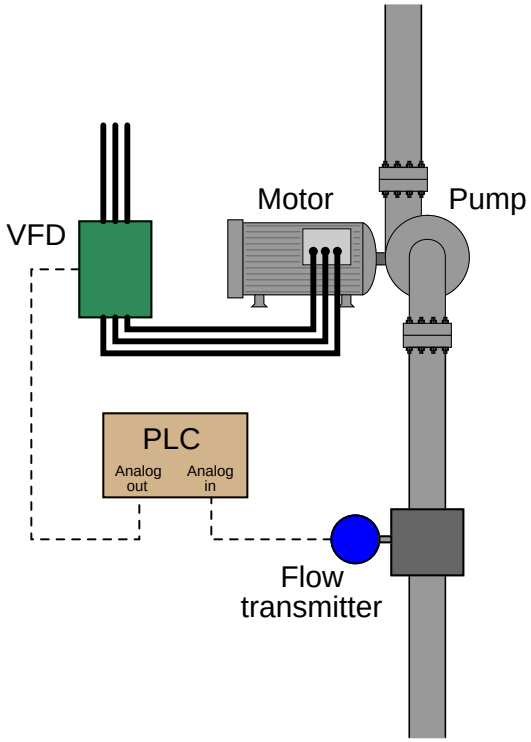
- DAC output fails low (output = 0 Volts DC):
- DAC output fails high (output = +V):
- IGBT Q_1 fails open (collector to emitter):
- Solder bridge (short) between MSB input on U_1 and ground:

Challenges

- How many motor speeds does this circuit offer?

6.3.2 Faulty pump control system

Once upon a time, your instructor was asked to troubleshoot a flow control system where a PLC (Programmable Logic Controller) received a flow measurement signal from a 4-20 mA analog flow sensor and sent a 4-20 mA command signal to a variable-frequency motor drive (VFD) turning a pump. The PLC's job was to turn the pump at the speed necessary to deliver a flow rate at a specified value:



The operator pressed the "Start" button, but the pump refused to turn. Upon examining the PLC's live program, it was found that the analog input value (a 16 bit binary number) was `$FFFF`. The measured current signal from the flow transmitter was 3.99 mA: just a little bit below 0% of range, which was reasonable for a no-flow condition.

As soon as the flow sensor was adjusting to output a current signal of exactly 4.00 mA at zero flow, the analog input in the PLC's program registered a value of `$0000` and the pump started up as it was supposed to when the operator pressed the "Start" button.

Explain why this simple sensor re-calibration was able to allow the system to run again, and why the real problem was a design flaw in the PLC.

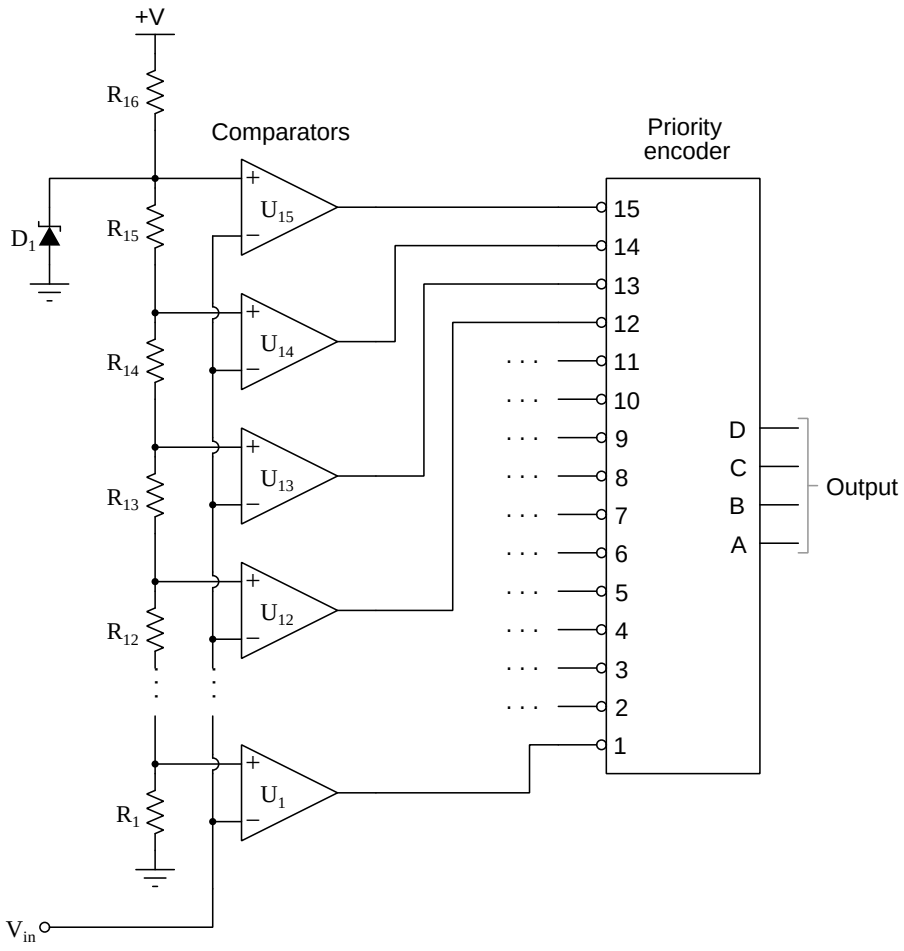
Challenges

- Explain why it would be prudent to adjust the flow sensor's calibration to output a signal

slightly greater than 4.00 mA at zero flow to help avoid this problem in the future.

6.3.3 Faulty flash ADC

This "flash" ADC circuit has a problem. The output code jumps from `0000` to `1111` with just the slightest amount of input voltage (V_{in}). In fact, the only time it outputs `0000` is when the input terminal is slightly negative with reference to ground:



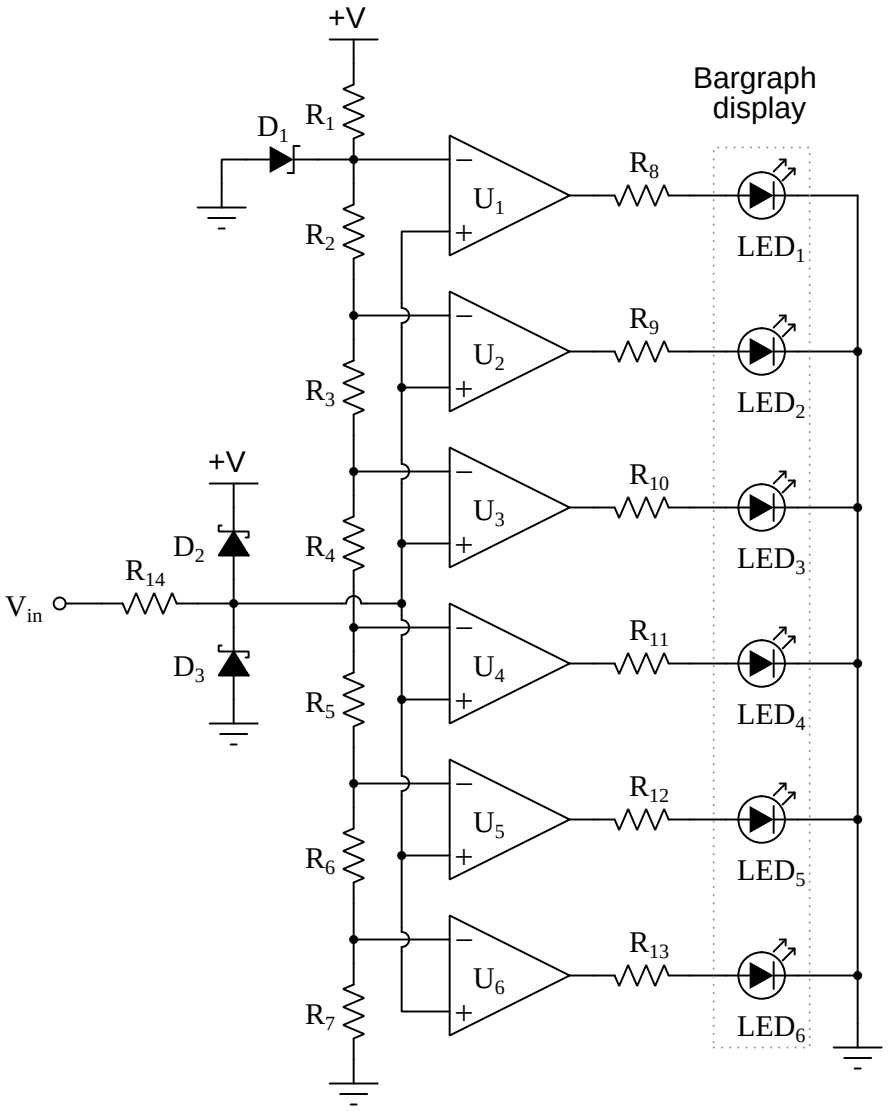
Identify at least two possible component faults that could cause this problem, and explain your reasoning in how you made the identifications.

Challenges

- Identify the next diagnostic step you would take in pinpointing the nature and location of the fault.

6.3.4 Faulty bargraph driver

This *bargraph driver* circuit takes an audio input signal and displays the amplitude as a moving "bar" of lights. The stronger the amplitude of the signal, the more LEDs energize in the bargraph display. Predict how the operation of this circuit will be affected as a result of the following faults. Consider each fault independently (i.e. one at a time, no multiple faults):



- Resistor R_4 failed open:

- Solder bridge (short) past resistor R_3:

- Resistor R_{11} failed open:
- Zener diode D_1 failed shorted:
- Schottky diode D_2 failed shorted:

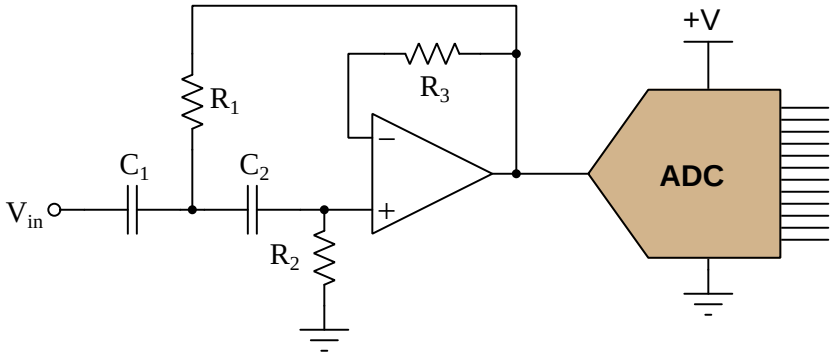
Challenges

- Does this bargraph begin at the top (with LED_1) or at the bottom (with LED_6)?
- Identify the design function of Schottky diodes D_2 and D_3.

6.3.5 Anti-aliasing design flaw

Analog-to-digital converter circuits (ADC) are usually equipped with analog filter circuits on their front ends to pre-condition the signal prior to digitization. These analog pre-filters are known as *anti-aliasing filters*.

Determine what is wrong with the filter circuit in this ADC design:

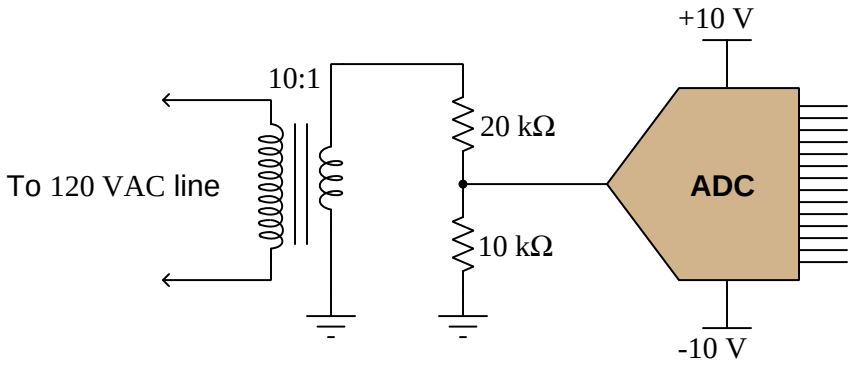


Challenges

- How *would* this circuit function if constructed and tested?

6.3.6 Over/under-flowing ADC

Suppose the following ADC has an input voltage range of +5 Volts to −5 Volts ("bipolar"), and therefore is suitable for digitizing AC input signals. A technician wants to use this ADC to digitize AC line voltage (120 Volts RMS), and builds the following conditioning circuit to safely connect the ADC to the AC line:



Unfortunately, this ADC is not able to fully sample the AC waveform when tested. It "overflows" and "underflows" at the waveform's peaks, as though the input waveform is too large (outside of the +5/−5 Volt ADC chip range). The technician re-checks his calculations, but still thinks the voltage division ratio provided by the potential transformer and resistor network should be sufficient for this task.

What is wrong with this circuit? Why is its range exceeded by the measured AC voltage waveform instead of sampling the 120 Volt waveform with range to spare? Then, once having identified the problem, recommend a solution to fix the problem.

Challenges

- Identify an appropriate ADC sampling rate for this application.

Appendix A

Problem-Solving Strategies

The ability to solve complex problems is arguably one of the most valuable skills one can possess, and this skill is particularly important in any science-based discipline.

- Study principles, not procedures. Don't be satisfied with merely knowing how to compute solutions – learn *why* those solutions work.
- Identify what it is you need to solve, identify all relevant data, identify all units of measurement, identify any general principles or formulae linking the given information to the solution, and then identify any "missing pieces" to a solution. Annotate all diagrams with this data.
- Sketch a diagram to help visualize the problem. When building a real system, always devise a plan for that system and analyze its function *before* constructing it.
- Follow the units of measurement and meaning of every calculation. If you are ever performing mathematical calculations as part of a problem-solving procedure, and you find yourself unable to apply each and every intermediate result to some aspect of the problem, it means you don't understand what you are doing. Properly done, every mathematical result should have practical meaning for the problem, and not just be an abstract number. You should be able to identify the proper units of measurement for each and every calculated result, and show where that result fits into the problem.
- Perform "thought experiments" to explore the effects of different conditions for theoretical problems. When troubleshooting real systems, perform *diagnostic tests* rather than visually inspecting for faults, the best diagnostic test being the one giving you the most information about the nature and/or location of the fault with the fewest steps.
- Simplify the problem until the solution becomes obvious, and then use that obvious case as a model to follow in solving the more complex version of the problem.
- Check for exceptions to see if your solution is incorrect or incomplete. A good solution will work for *all* known conditions and criteria. A good example of this is the process of testing scientific hypotheses: the task of a scientist is not to find support for a new idea, but rather to *challenge* that new idea to see if it holds up under a battery of tests. The philosophical

principle of *reductio ad absurdum* (i.e. disproving a general idea by finding a specific case where it fails) is useful here.

- Work "backward" from a hypothetical solution to a new set of given conditions.
- Add quantities to problems that are qualitative in nature, because sometimes a little math helps illuminate the scenario.
- Sketch graphs illustrating how variables relate to each other. These may be quantitative (i.e. with realistic number values) or qualitative (i.e. simply showing increases and decreases).
- Treat quantitative problems as qualitative in order to discern the relative magnitudes and/or directions of change of the relevant variables. For example, try determining what happens if a certain variable were to increase or decrease before attempting to precisely calculate quantities: how will each of the dependent variables respond, by increasing, decreasing, or remaining the same as before?
- Consider limiting cases. This works especially well for qualitative problems where you need to determine which direction a variable will change. Take the given condition and magnify that condition to an extreme degree as a way of simplifying the direction of the system's response.
- Check your work. This means regularly testing your conclusions to see if they make sense. This does *not* mean repeating the same steps originally used to obtain the conclusion(s), but rather to use some other means to check validity. Simply repeating procedures often leads to *repeating the same errors* if any were made, which is why alternative paths are better.

Appendix B

Instructional philosophy

"The unexamined circuit is not worth energizing" – Socrates (if he had taught electricity)

These learning modules, although useful for self-study, were designed to be used in a formal learning environment where a subject-matter expert challenges students to digest the content and exercise their critical thinking abilities in the answering of questions and in the construction and testing of working circuits.

The following principles inform the instructional and assessment philosophies embodied in these learning modules:

- The first goal of education is to enhance clear and independent thought, in order that every student reach their fullest potential in a highly complex and inter-dependent world. Robust reasoning is *always* more important than particulars of any subject matter, because its application is universal.
- Literacy is fundamental to independent learning and thought because text continues to be the most efficient way to communicate complex ideas over space and time. Those who cannot read with ease are limited in their ability to acquire knowledge and perspective.
- Articulate communication is fundamental to work that is complex and interdisciplinary.
- Faulty assumptions and poor reasoning are best corrected through challenge, not presentation. The rhetorical technique of *reductio ad absurdum* (disproving an assertion by exposing an absurdity) works well to discipline student's minds, not only to correct the problem at hand but also to learn how to detect and correct future errors.
- Important principles should be repeatedly explored and widely applied throughout a course of study, not only to reinforce their importance and help ensure their mastery, but also to showcase the interconnectedness and utility of knowledge.

These learning modules were expressly designed to be used in an "inverted" teaching environment[1] where students first read the introductory and tutorial chapters on their own, then individually attempt to answer the questions and construct working circuits according to the experiment and project guidelines. The instructor never lectures, but instead meets regularly with each individual student to review their progress, answer questions, identify misconceptions, and challenge the student to new depths of understanding through further questioning. Regular meetings between instructor and student should resemble a Socratic[2] dialogue, where questions serve as scalpels to dissect topics and expose assumptions. The student passes each module only after consistently demonstrating their ability to logically analyze and correctly apply all major concepts in each question or project/experiment. The instructor must be vigilant in probing each student's understanding to ensure they are truly *reasoning* and not just *memorizing*. This is why "Challenge" points appear throughout, as prompts for students to think deeper about topics and as starting points for instructor queries. Sometimes these challenge points require additional knowledge that hasn't been covered in the series to answer in full. This is okay, as the major purpose of the Challenges is to stimulate analysis and synthesis on the part of each student.

The instructor must possess enough mastery of the subject matter and awareness of students' reasoning to generate their own follow-up questions to practically any student response. Even completely correct answers given by the student should be challenged by the instructor for the purpose of having students practice articulating their thoughts and defending their reasoning. Conceptual errors committed by the student should be exposed and corrected not by direct instruction, but rather by reducing the errors to an absurdity[3] through well-chosen questions and thought experiments posed by the instructor. Becoming proficient at this style of instruction requires time and dedication, but the positive effects on critical thinking for both student and instructor are spectacular.

An inspection of these learning modules reveals certain unique characteristics. One of these is a bias toward thorough explanations in the tutorial chapters. Without a live instructor to explain concepts and applications to students, the text itself must fulfill this role. This philosophy results in lengthier explanations than what you might typically find in a textbook, each step of the reasoning process fully explained, including footnotes addressing common questions and concerns students raise while learning these concepts. Each tutorial seeks to not only explain each major concept in sufficient detail, but also to explain the logic of each concept and how each may be developed

[1] In a traditional teaching environment, students first encounter new information via *lecture* from an expert, and then independently apply that information via *homework*. In an "inverted" course of study, students first encounter new information via *homework*, and then independently apply that information under the scrutiny of an expert. The expert's role in lecture is to simply *explain*, but the expert's role in an inverted session is to *challenge*, *critique*, and if necessary *explain* where gaps in understanding still exist.

[2] Socrates is a figure in ancient Greek philosophy famous for his unflinching style of questioning. Although he authored no texts, he appears as a character in Plato's many writings. The essence of Socratic philosophy is to leave no question unexamined and no point of view unchallenged. While purists may argue a topic such as electric circuits is too narrow for a true Socratic-style dialogue, I would argue that the essential thought processes involved with scientific reasoning on *any* topic are not far removed from the Socratic ideal, and that students of electricity and electronics would do very well to challenge assumptions, pose thought experiments, identify fallacies, and otherwise employ the arsenal of critical thinking skills modeled by Socrates.

[3] This rhetorical technique is known by the Latin phrase *reductio ad absurdum*. The concept is to expose errors by counter-example, since only one solid counter-example is necessary to disprove a universal claim. As an example of this, consider the common misconception among beginning students of electricity that voltage cannot exist without current. One way to apply *reductio ad absurdum* to this statement is to ask how much current passes through a fully-charged battery connected to nothing (i.e. a clear example of voltage existing without current).

from "first principles". Again, this reflects the goal of developing clear and independent thought in students' minds, by showing how clear and logical thought was used to forge each concept. Students benefit from witnessing a model of clear thinking in action, and these tutorials strive to be just that.

Another characteristic of these learning modules is a lack of step-by-step instructions in the Project and Experiment chapters. Unlike many modern workbooks and laboratory guides where step-by-step instructions are prescribed for each experiment, these modules take the approach that students must learn to closely read the tutorials and apply their own reasoning to identify the appropriate experimental steps. Sometimes these steps are plainly declared in the text, just not as a set of enumerated points. At other times certain steps are implied, an example being assumed competence in test equipment use where the student should not need to be told *again* how to use their multimeter because that was thoroughly explained in previous lessons. In some circumstances no steps are given at all, leaving the entire procedure up to the student.

This lack of prescription is not a flaw, but rather a feature. Close reading and clear thinking are foundational principles of this learning series, and in keeping with this philosophy all activities are designed to *require* those behaviors. Some students may find the lack of prescription frustrating, because it demands more from them than what their previous educational experiences required. This frustration should be interpreted as an unfamiliarity with autonomous thinking, a problem which must be corrected if the student is ever to become a self-directed learner and effective problem-solver. Ultimately, the need for students to read closely and think clearly is more important both in the near-term and far-term than any specific facet of the subject matter at hand. If a student takes longer than expected to complete a module because they are forced to outline, digest, and reason on their own, so be it. The future gains enjoyed by developing this mental discipline will be well worth the additional effort and delay.

Another feature of these learning modules is that they do not treat topics in isolation. Rather, important concepts are introduced early in the series, and appear repeatedly as stepping-stones toward other concepts in subsequent modules. This helps to avoid the "compartmentalization" of knowledge, demonstrating the inter-connectedness of concepts and simultaneously reinforcing them. Each module is fairly complete in itself, reserving the beginning of its tutorial to a review of foundational concepts.

This methodology of assigning text-based modules to students for digestion and then using Socratic dialogue to assess progress and hone students' thinking was developed over a period of several years by the author with his Electronics and Instrumentation students at the two-year college level. While decidedly unconventional and sometimes even unsettling for students accustomed to a more passive lecture environment, this instructional philosophy has proven its ability to convey conceptual mastery, foster careful analysis, and enhance employability so much better than lecture that the author refuses to ever teach by lecture again.

Problems which often go undiagnosed in a lecture environment are laid bare in this "inverted" format where students must articulate and logically defend their reasoning. This, too, may be unsettling for students accustomed to lecture sessions where the instructor cannot tell for sure who comprehends and who does not, and this vulnerability necessitates sensitivity on the part of the "inverted" session instructor in order that students never feel discouraged by having their errors exposed. *Everyone* makes mistakes from time to time, and learning is a lifelong process! Part of the instructor's job is to build a culture of learning among the students where errors are not seen as shameful, but rather as opportunities for progress.

To this end, instructors managing courses based on these modules should adhere to the following principles:

- Student questions are always welcome and demand thorough, honest answers. The only type of question an instructor should refuse to answer is one the student should be able to easily answer on their own. Remember, *the fundamental goal of education is for each student to learn to think clearly and independently.* This requires hard work on the part of the student, which no instructor should ever circumvent. Anything done to bypass the student's responsibility to do that hard work ultimately limits that student's potential and thereby does real harm.

- It is not only permissible, but encouraged, to answer a student's question by asking questions in return, these follow-up questions designed to guide the student to reach a correct answer through their own reasoning.

- All student answers demand to be challenged by the instructor and/or by other students. This includes both correct and incorrect answers – the goal is to practice the articulation and defense of one's own reasoning.

- No reading assignment is deemed complete unless and until the student demonstrates their ability to accurately summarize the major points in their own terms. Recitation of the original text is unacceptable. This is why every module contains an "Outline and reflections" question as well as a "Foundational concepts" question in the Conceptual reasoning section, to prompt reflective reading.

- No assigned question is deemed answered unless and until the student demonstrates their ability to consistently and correctly apply the concepts to *variations* of that question. This is why module questions typically contain multiple "Challenges" suggesting different applications of the concept(s) as well as variations on the same theme(s). Instructors are encouraged to devise as many of their own "Challenges" as they are able, in order to have a multitude of ways ready to probe students' understanding.

- No assigned experiment or project is deemed complete unless and until the student demonstrates the task in action. If this cannot be done "live" before the instructor, video-recordings showing the demonstration are acceptable. All relevant safety precautions must be followed, all test equipment must be used correctly, and the student must be able to properly explain all results. The student must also successfully answer all Challenges presented by the instructor for that experiment or project.

Students learning from these modules would do well to abide by the following principles:

- No text should be considered fully and adequately read unless and until you can express every idea *in your own words, using your own examples.*
- You should always articulate your thoughts as you read the text, noting points of agreement, confusion, and epiphanies. Feel free to print the text on paper and then write your notes in the margins. Alternatively, keep a journal for your own reflections as you read. This is truly a helpful tool when digesting complicated concepts.
- Never take the easy path of highlighting or underlining important text. Instead, *summarize* and/or *comment* on the text using your own words. This actively engages your mind, allowing you to more clearly perceive points of confusion or misunderstanding on your own.
- A very helpful strategy when learning new concepts is to place yourself in the role of a teacher, if only as a mental exercise. Either explain what you have recently learned to someone else, or at least *imagine* yourself explaining what you have learned to someone else. The simple act of having to articulate new knowledge and skill forces you to take on a different perspective, and will help reveal weaknesses in your understanding.
- Perform each and every mathematical calculation and thought experiment shown in the text on your own, referring back to the text to see that your results agree. This may seem trivial and unnecessary, but it is critically important to ensuring you actually understand what is presented, especially when the concepts at hand are complicated and easy to misunderstand. Apply this same strategy to become proficient in the use of *circuit simulation software*, checking to see if your simulated results agree with the results shown in the text.
- Above all, recognize that learning is hard work, and that a certain level of frustration is unavoidable. There are times when you will struggle to grasp some of these concepts, and that struggle is a natural thing. Take heart that it will yield with persistent and varied[4] effort, and never give up!

Students interested in using these modules for self-study will also find them beneficial, although the onus of responsibility for thoroughly reading and answering questions will of course lie with that individual alone. If a qualified instructor is not available to challenge students, a workable alternative is for students to form study groups where they challenge[5] one another.

To high standards of education,

Tony R. Kuphaldt

[4] As the old saying goes, "Insanity is trying the same thing over and over again, expecting different results." If you find yourself stumped by something in the text, you should attempt a different approach. Alter the thought experiment, change the mathematical parameters, do whatever you can to see the problem in a slightly different light, and then the solution will often present itself more readily.

[5] Avoid the temptation to simply share answers with study partners, as this is really counter-productive to learning. Always bear in mind that the answer to any question is far less important in the long run than the method(s) used to obtain that answer. The goal of education is to empower one's life through the improvement of clear and independent thought, literacy, expression, and various practical skills.

Appendix C

Tools used

I am indebted to the developers of many open-source software applications in the creation of these learning modules. The following is a list of these applications with some commentary on each.

You will notice a theme common to many of these applications: a bias toward *code*. Although I am by no means an expert programmer in any computer language, I understand and appreciate the flexibility offered by code-based applications where the user (you) enters commands into a plain ASCII text file, which the software then reads and processes to create the final output. Code-based computer applications are by their very nature *extensible*, while WYSIWYG (What You See Is What You Get) applications are generally limited to whatever user interface the developer makes for you.

The `GNU/Linux` computer operating system

> There is so much to be said about Linus Torvalds' `Linux` and Richard Stallman's `GNU` project. First, to credit just these two individuals is to fail to do justice to the *mob* of passionate volunteers who contributed to make this amazing software a reality. I first learned of `Linux` back in 1996, and have been using this operating system on my personal computers almost exclusively since then. It is *free*, it is completely *configurable*, and it permits the continued use of highly efficient `Unix` applications and scripting languages (e.g. shell scripts, Makefiles, `sed`, `awk`) developed over many decades. `Linux` not only provided me with a powerful computing platform, but its open design served to inspire my life's work of creating open-source educational resources.

Bram Moolenaar's `Vim` text editor

> Writing code for any code-based computer application requires a *text editor*, which may be thought of as a word processor strictly limited to outputting plain-ASCII text files. Many good text editors exist, and one's choice of text editor seems to be a deeply personal matter within the programming world. I prefer `Vim` because it operates very similarly to `vi` which is ubiquitous on `Unix`/`Linux` operating systems, and because it may be entirely operated via keyboard (i.e. no mouse required) which makes it fast to use.

Donald Knuth's TEX typesetting system

Developed in the late 1970's and early 1980's by computer scientist extraordinaire Donald Knuth to typeset his multi-volume magnum opus *The Art of Computer Programming*, this software allows the production of formatted text for screen-viewing or paper printing, all by writing plain-text code to describe how the formatted text is supposed to appear. TEX is not just a markup language for documents, but it is also a Turing-complete programming language in and of itself, allowing useful algorithms to be created to control the production of documents. Simply put, TEX *is a programmer's approach to word processing*. Since TEX is controlled by code written in a plain-text file, this means anyone may read that plain-text file to see exactly how the document was created. This openness afforded by the code-based nature of TEX makes it relatively easy to learn how other people have created their own TEX documents. By contrast, examining a beautiful document created in a conventional WYSIWYG word processor such as Microsoft `Word` suggests nothing to the reader about *how* that document was created, or what the user might do to create something similar. As Mr. Knuth himself once quipped, conventional word processing applications should be called WYSIAYG (What You See Is *All* You Get).

Leslie Lamport's LATEX extensions to TEX

Like all true programming languages, TEX is inherently extensible. So, years after the release of TEX to the public, Leslie Lamport decided to create a massive extension allowing easier compilation of book-length documents. The result was LATEX, which is the markup language used to create all ModEL module documents. You could say that TEX is to LATEX as `C` is to `C++`. This means it is permissible to use any and all TEX commands within LATEX source code, and it all still works. Some of the features offered by LATEX that would be challenging to implement in TEX include automatic index and table-of-content creation.

Tim Edwards' `Xcircuit` drafting program

This wonderful program is what I use to create all the schematic diagrams and illustrations (but not photographic images or mathematical plots) throughout the ModEL project. It natively outputs PostScript format which is a true vector graphic format (this is why the images do not pixellate when you zoom in for a closer view), and it is so simple to use that I have never had to read the manual! Object libraries are easy to create for `Xcircuit`, being plain-text files using PostScript programming conventions. Over the years I have collected a large set of object libraries useful for drawing electrical and electronic schematics, pictorial diagrams, and other technical illustrations.

`Gimp` graphic image manipulation program

Essentially an open-source clone of Adobe's `PhotoShop`, I use `Gimp` to resize, crop, and convert file formats for all of the photographic images appearing in the ModEL modules. Although `Gimp` does offer its own scripting language (called `Script-Fu`), I have never had occasion to use it. Thus, my utilization of `Gimp` to merely crop, resize, and convert graphic images is akin to using a sword to slice bread.

`SPICE` circuit simulation program

`SPICE` is to circuit analysis as TEX is to document creation: it is a form of markup language designed to describe a certain object to be processed in plain-ASCII text. When the plain-text "source file" is compiled by the software, it outputs the final result. More modern circuit analysis tools certainly exist, but I prefer `SPICE` for the following reasons: it is *free*, it is *fast*, it is *reliable*, and it is a fantastic tool for *teaching* students of electricity and electronics how to write simple code. I happen to use rather old versions of `SPICE`, version 2g6 being my "go to" application when I only require text-based output. `NGSPICE` (version 26), which is based on Berkeley `SPICE` version 3f5, is used when I require graphical output for such things as time-domain waveforms and Bode plots. In all `SPICE` example netlists I strive to use coding conventions compatible with all `SPICE` versions.

Andrew D. Hwang's `ePiX` mathematical visualization programming library

This amazing project is a `C++` library you may link to any `C/C++` code for the purpose of generating PostScript graphic images of mathematical functions. As a completely free and open-source project, it does all the plotting I would otherwise use a Computer Algebra System (CAS) such as `Mathematica` or `Maple` to do. It should be said that `ePiX` is *not* a Computer Algebra System like `Mathematica` or `Maple`, but merely a mathematical *visualization* tool. In other words, it won't determine integrals for you (you'll have to implement that in your own `C/C++` code!), but it can graph the results, and it does so beautifully. What I really admire about `ePiX` is that it is a `C++` programming library, which means it builds on the existing power and toolset available with that programming language. Mr. Hwang could have probably developed his own stand-alone application for mathematical plotting, but by creating a `C++` library to do the same thing he accomplished something much greater.

`gnuplot` mathematical visualization software

Another open-source tool for mathematical visualization is `gnuplot`. Interestingly, this tool is *not* part of Richard Stallman's GNU project, its name being a coincidence. For this reason the authors prefer "gnu" *not* be capitalized at all to avoid confusion. This is a much "lighter-weight" alternative to a spreadsheet for plotting tabular data, and the fact that it easily outputs directly to an X11 console or a file in a number of different graphical formats (including PostScript) is very helpful. I typically set my `gnuplot` output format to default (X11 on my Linux PC) for quick viewing while I'm developing a visualization, then switch to PostScript file export once the visual is ready to include in the document(s) I'm writing. As with my use of `Gimp` to do rudimentary image editing, my use of `gnuplot` only scratches the surface of its capabilities, but the important points are that it's *free* and that it *works well*.

`Python` programming language

Both Python and C++ find extensive use in these modules as instructional aids and exercises, but I'm listing Python here as a *tool* for myself because I use it almost daily as a *calculator*. If you open a Python interpreter console and type `from math import *` you can type mathematical expressions and have it return results just as you would on a hand calculator. Complex-number (i.e. *phasor*) arithmetic is similarly supported if you include the complex-math library (`from cmath import *`). Examples of this are shown in the Programming References chapter (if included) in each module. Of course, being a fully-featured programming language, Python also supports conditionals, loops, and other structures useful for calculation of quantities. Also, running in a console environment where all entries and returned values show as text in a chronologically-ordered list makes it easy to copy-and-paste those calculations to document exactly how they were performed.

Appendix D

Creative Commons License

Creative Commons Attribution 4.0 International Public License

By exercising the Licensed Rights (defined below), You accept and agree to be bound by the terms and conditions of this Creative Commons Attribution 4.0 International Public License ("Public License"). To the extent this Public License may be interpreted as a contract, You are granted the Licensed Rights in consideration of Your acceptance of these terms and conditions, and the Licensor grants You such rights in consideration of benefits the Licensor receives from making the Licensed Material available under these terms and conditions.

Section 1 – Definitions.

a. **Adapted Material** means material subject to Copyright and Similar Rights that is derived from or based upon the Licensed Material and in which the Licensed Material is translated, altered, arranged, transformed, or otherwise modified in a manner requiring permission under the Copyright and Similar Rights held by the Licensor. For purposes of this Public License, where the Licensed Material is a musical work, performance, or sound recording, Adapted Material is always produced where the Licensed Material is synched in timed relation with a moving image.

b. **Adapter's License** means the license You apply to Your Copyright and Similar Rights in Your contributions to Adapted Material in accordance with the terms and conditions of this Public License.

c. **Copyright and Similar Rights** means copyright and/or similar rights closely related to copyright including, without limitation, performance, broadcast, sound recording, and Sui Generis Database Rights, without regard to how the rights are labeled or categorized. For purposes of this Public License, the rights specified in Section 2(b)(1)-(2) are not Copyright and Similar Rights.

d. **Effective Technological Measures** means those measures that, in the absence of proper authority, may not be circumvented under laws fulfilling obligations under Article 11 of the WIPO Copyright Treaty adopted on December 20, 1996, and/or similar international agreements.

e. **Exceptions and Limitations** means fair use, fair dealing, and/or any other exception or

limitation to Copyright and Similar Rights that applies to Your use of the Licensed Material.

f. **Licensed Material** means the artistic or literary work, database, or other material to which the Licensor applied this Public License.

g. **Licensed Rights** means the rights granted to You subject to the terms and conditions of this Public License, which are limited to all Copyright and Similar Rights that apply to Your use of the Licensed Material and that the Licensor has authority to license.

h. **Licensor** means the individual(s) or entity(ies) granting rights under this Public License.

i. **Share** means to provide material to the public by any means or process that requires permission under the Licensed Rights, such as reproduction, public display, public performance, distribution, dissemination, communication, or importation, and to make material available to the public including in ways that members of the public may access the material from a place and at a time individually chosen by them.

j. **Sui Generis Database Rights** means rights other than copyright resulting from Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, as amended and/or succeeded, as well as other essentially equivalent rights anywhere in the world.

k. **You** means the individual or entity exercising the Licensed Rights under this Public License. **Your** has a corresponding meaning.

Section 2 – Scope.

a. License grant.

1. Subject to the terms and conditions of this Public License, the Licensor hereby grants You a worldwide, royalty-free, non-sublicensable, non-exclusive, irrevocable license to exercise the Licensed Rights in the Licensed Material to:

A. reproduce and Share the Licensed Material, in whole or in part; and

B. produce, reproduce, and Share Adapted Material.

2. Exceptions and Limitations. For the avoidance of doubt, where Exceptions and Limitations apply to Your use, this Public License does not apply, and You do not need to comply with its terms and conditions.

3. Term. The term of this Public License is specified in Section 6(a).

4. Media and formats; technical modifications allowed. The Licensor authorizes You to exercise the Licensed Rights in all media and formats whether now known or hereafter created, and to make technical modifications necessary to do so. The Licensor waives and/or agrees not to assert any right or authority to forbid You from making technical modifications necessary to exercise the Licensed Rights, including technical modifications necessary to circumvent Effective Technological Measures.

For purposes of this Public License, simply making modifications authorized by this Section 2(a)(4) never produces Adapted Material.

5. Downstream recipients.

A. Offer from the Licensor – Licensed Material. Every recipient of the Licensed Material automatically receives an offer from the Licensor to exercise the Licensed Rights under the terms and conditions of this Public License.

B. No downstream restrictions. You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, the Licensed Material if doing so restricts exercise of the Licensed Rights by any recipient of the Licensed Material.

6. No endorsement. Nothing in this Public License constitutes or may be construed as permission to assert or imply that You are, or that Your use of the Licensed Material is, connected with, or sponsored, endorsed, or granted official status by, the Licensor or others designated to receive attribution as provided in Section 3(a)(1)(A)(i).

b. Other rights.

1. Moral rights, such as the right of integrity, are not licensed under this Public License, nor are publicity, privacy, and/or other similar personality rights; however, to the extent possible, the Licensor waives and/or agrees not to assert any such rights held by the Licensor to the limited extent necessary to allow You to exercise the Licensed Rights, but not otherwise.

2. Patent and trademark rights are not licensed under this Public License.

3. To the extent possible, the Licensor waives any right to collect royalties from You for the exercise of the Licensed Rights, whether directly or through a collecting society under any voluntary or waivable statutory or compulsory licensing scheme. In all other cases the Licensor expressly reserves any right to collect such royalties.

Section 3 – License Conditions.

Your exercise of the Licensed Rights is expressly made subject to the following conditions.

a. Attribution.

1. If You Share the Licensed Material (including in modified form), You must:

A. retain the following if it is supplied by the Licensor with the Licensed Material:

i. identification of the creator(s) of the Licensed Material and any others designated to receive attribution, in any reasonable manner requested by the Licensor (including by pseudonym if designated);

ii. a copyright notice;

iii. a notice that refers to this Public License;

iv. a notice that refers to the disclaimer of warranties;

v. a URI or hyperlink to the Licensed Material to the extent reasonably practicable;

B. indicate if You modified the Licensed Material and retain an indication of any previous modifications; and

C. indicate the Licensed Material is licensed under this Public License, and include the text of, or the URI or hyperlink to, this Public License.

2. You may satisfy the conditions in Section 3(a)(1) in any reasonable manner based on the medium, means, and context in which You Share the Licensed Material. For example, it may be reasonable to satisfy the conditions by providing a URI or hyperlink to a resource that includes the required information.

3. If requested by the Licensor, You must remove any of the information required by Section 3(a)(1)(A) to the extent reasonably practicable.

4. If You Share Adapted Material You produce, the Adapter's License You apply must not prevent recipients of the Adapted Material from complying with this Public License.

Section 4 – Sui Generis Database Rights.

Where the Licensed Rights include Sui Generis Database Rights that apply to Your use of the Licensed Material:

a. for the avoidance of doubt, Section 2(a)(1) grants You the right to extract, reuse, reproduce, and Share all or a substantial portion of the contents of the database;

b. if You include all or a substantial portion of the database contents in a database in which You have Sui Generis Database Rights, then the database in which You have Sui Generis Database Rights (but not its individual contents) is Adapted Material; and

c. You must comply with the conditions in Section 3(a) if You Share all or a substantial portion of the contents of the database.

For the avoidance of doubt, this Section 4 supplements and does not replace Your obligations under this Public License where the Licensed Rights include other Copyright and Similar Rights.

Section 5 – Disclaimer of Warranties and Limitation of Liability.

a. Unless otherwise separately undertaken by the Licensor, to the extent possible, the Licensor offers the Licensed Material as-is and as-available, and makes no representations or warranties of any kind concerning the Licensed Material, whether express, implied, statutory, or other. This includes, without limitation, warranties of title, merchantability, fitness for a particular purpose, non-infringement, absence of latent or other defects, accuracy, or the presence or absence of errors,

whether or not known or discoverable. Where disclaimers of warranties are not allowed in full or in part, this disclaimer may not apply to You.

b. To the extent possible, in no event will the Licensor be liable to You on any legal theory (including, without limitation, negligence) or otherwise for any direct, special, indirect, incidental, consequential, punitive, exemplary, or other losses, costs, expenses, or damages arising out of this Public License or use of the Licensed Material, even if the Licensor has been advised of the possibility of such losses, costs, expenses, or damages. Where a limitation of liability is not allowed in full or in part, this limitation may not apply to You.

c. The disclaimer of warranties and limitation of liability provided above shall be interpreted in a manner that, to the extent possible, most closely approximates an absolute disclaimer and waiver of all liability.

Section 6 – Term and Termination.

a. This Public License applies for the term of the Copyright and Similar Rights licensed here. However, if You fail to comply with this Public License, then Your rights under this Public License terminate automatically.

b. Where Your right to use the Licensed Material has terminated under Section 6(a), it reinstates:

1. automatically as of the date the violation is cured, provided it is cured within 30 days of Your discovery of the violation; or

2. upon express reinstatement by the Licensor.

For the avoidance of doubt, this Section 6(b) does not affect any right the Licensor may have to seek remedies for Your violations of this Public License.

c. For the avoidance of doubt, the Licensor may also offer the Licensed Material under separate terms or conditions or stop distributing the Licensed Material at any time; however, doing so will not terminate this Public License.

d. Sections 1, 5, 6, 7, and 8 survive termination of this Public License.

Section 7 – Other Terms and Conditions.

a. The Licensor shall not be bound by any additional or different terms or conditions communicated by You unless expressly agreed.

b. Any arrangements, understandings, or agreements regarding the Licensed Material not stated herein are separate from and independent of the terms and conditions of this Public License.

Section 8 – Interpretation.

a. For the avoidance of doubt, this Public License does not, and shall not be interpreted to, reduce, limit, restrict, or impose conditions on any use of the Licensed Material that could lawfully

be made without permission under this Public License.

b. To the extent possible, if any provision of this Public License is deemed unenforceable, it shall be automatically reformed to the minimum extent necessary to make it enforceable. If the provision cannot be reformed, it shall be severed from this Public License without affecting the enforceability of the remaining terms and conditions.

c. No term or condition of this Public License will be waived and no failure to comply consented to unless expressly agreed to by the Licensor.

d. Nothing in this Public License constitutes or may be interpreted as a limitation upon, or waiver of, any privileges and immunities that apply to the Licensor or You, including from the legal processes of any jurisdiction or authority.

Creative Commons is not a party to its public licenses. Notwithstanding, Creative Commons may elect to apply one of its public licenses to material it publishes and in those instances will be considered the "Licensor." Except for the limited purpose of indicating that material is shared under a Creative Commons public license or as otherwise permitted by the Creative Commons policies published at `creativecommons.org/policies`, Creative Commons does not authorize the use of the trademark "Creative Commons" or any other trademark or logo of Creative Commons without its prior written consent including, without limitation, in connection with any unauthorized modifications to any of its public licenses or any other arrangements, understandings, or agreements concerning use of licensed material. For the avoidance of doubt, this paragraph does not form part of the public licenses.

Creative Commons may be contacted at `creativecommons.org`.

Appendix E

References

Carlson, R; Krakauer, S.; Magleby, K.; Monnier, R.; Van Duzer, V.; Woodbury, R.; "Sampling Oscillography", Application Note 36, Hewlett-Packard Co., Palo Alto, CA.

Kester, Walt, "Find Those Elusive ADC Sparkle Codes and Metastable States", Tutorial MT-011, Analog Devices, Inc., 2009.

Laws, Frank A., *Electrical Measurements*, First Edition, McGraw-Hill Book Company, Inc., New York, NY, 1917.

Appendix F

Version history

This is a list showing all significant additions, corrections, and other edits made to this learning module. Each entry is referenced by calendar date in reverse chronological order (newest version first), which appears on the front cover of every learning module for easy reference. Any contributors to this open-source document are listed here as well.

2 April 2025 – grammatical error correction courtesy of Jacob Stormes ("are were").

1 April 2025 – added a reference to the Tutorial concerning *decimation* as a possible cause of aliasing within a digital system.

31 October 2024 – minor edits made to the Tutorial.

25 October 2024 – divided the Introduction chapter into sections, one with recommendations for students, one with a listing of challenging concepts, and one with recommendations for instructors.

14 April 2024 – added a new Historical References section on Frank Gray's coding scheme as applied to a unique design of analog-digital converter.

20 January 2024 – added a new Technical Reference section on protecting amplifier inputs against overvoltage from electro-static discharge (ESD) as well as electrical over-stress (EOS) from improperly connected signal sources.

29 November 2022 – placed questions at the top of the itemized list in the Introduction chapter prompting students to devise experiments related to the tutorial content.

10 July 2021 – added more content on R-2R ladder networks.

10 May 2021 – commented out or deleted empty chapters.

18 March 2021 – corrected multiple instances of "volts" that should have been capitalized "Volts".

5 October 2020 – significantly edited the Introduction chapter to make it more suitable as a pre-study guide and to provide cues useful to instructors leading "inverted" teaching sessions.

23 March 2020 – added another Quantitative Reasoning problem.

20 March 2020 – minor typographical error correction.

18 March 2020 – added Case Tutorial chapter. Also fixed several capitalization errors on the word "Volt" and also added "unipolar" and "bipolar" to some of the problems.

20 February 2020 – added some more problems.

13 February 2020 – finished writing Tutorial, as well as added a Derivations section on $R - 2R$ ladder networks.

12 February 2020 – continued writing Tutorial.

11 February 2020 – continued writing Tutorial.

5 February 2020 – added to the Historical References section and added several problems.

4 February 2020 – document first created, borrowing heavily from the "Analog-digital Conversion" section of the "Digital Data Acquisition and Networks" chapter of my book *Lessons In Industrial Instrumentation*.

Index