

Introduction

Signals in the real world are analog: light, sound, you name it. So, real-world signals must be converted into digital, using a circuit called ADC (Analog-to-Digital Converter), before they can be manipulated by digital equipment. In this tutorial we will give an in-depth explanation about analog-to-digital conversion yet keeping a very easy to follow language.

When you scan a picture with a scanner what the scanner is doing is an analog-to-digital conversion: it is taking the analog information provided by the picture (light) and converting into digital.

When you record your voice or use a VoIP solution on your computer, you are using an analog-to-digital converter to convert your voice, which is analog, into digital information.

Digital information isn't only restricted to computers. When you talk on the phone, for example, your voice is converted into digital (at the central office switch, if you use an analog line, or at you home, if you use a digital line like ISDN or DSL), since your voice is analog and the communication between the phone switches is done digitally.

When an audio CD is recorded at a studio, once again analog-to-digital is taking place, converting sounds into digital numbers that will be stored on the disc.

Whenever we need the analog signal back, the opposite conversion – digital-to-analog, which is done by a circuit called DAC, Digital-to-Analog Converter – is needed. When you play an audio CD, what the CD player is doing is reading digital information stored on the disc and converting it back to analog so you can hear the music. When you are talking on the phone, a digital-to-analog conversion is also taking place (at the central office switch, if you use an analog line, or at you home, if you use a digital line like ISDN or DSL), so you can hear what the other party is saying.

But, why digital? There are some basic reasons to use digital signals instead of analog, noise being the number one.

Since analog signals can assume any value, noise is interpreted as being part of the original signal. For example, when you listen to a LP record, you hear noise because the needle is analog and thus don't know the difference from the music originally recorded from the noise inserted by dust or cracks.

Digital systems, on the other hand, can only understand two numbers, zero and one. Anything different from this is discarded. That's why you won't hear any unwanted noise when listening to an audio CD, even if you played it thousands of times before (actually depending on your sound system you can hear some noise when playing audio CDs, but this noise, called white noise, isn't produced by the CD media, but by the CD player, amplifier or cables used, and is introduced in the audio path after the digital data found on the CD was already converted back to analog – as you see, the problem lies in the analog part).

Another advantage of digital system against analog is the data compression capability. Since the digital counterpart of an analog signal is just a bunch of numbers, these numbers can be compressed, just like you would compress a Word file using WinZip to shrink down the file size, for example. The compression can be done to save storage space or bandwidth. On all the examples given so far no compression is used. We will talk again about it when discussing surround sound.

How It Works: Sampling

For our explanations, consider the analog signal found on Figure 1. Let's assume that it is an audio signal, since this the most popular applications for analog-to-digital and digital-to-analog conversions. The "y" axis represents voltage while the "x" axis represents time.

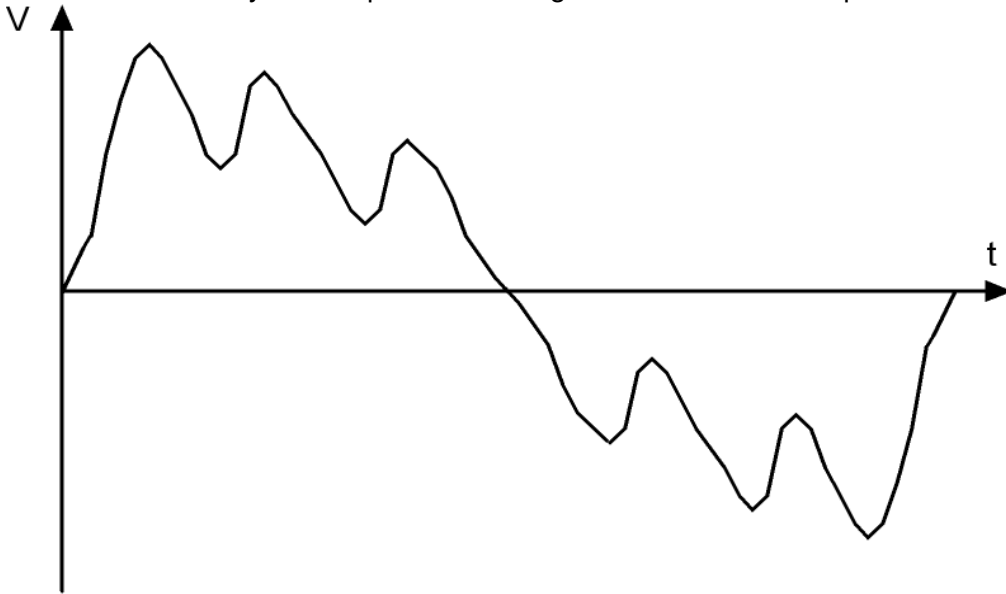


Figure 1: An analog signal.

What the ADC circuit does is to take samples from the analog signal from time to time. Each sample will be converted into a number, based on its voltage level. On Figure 2 you see an example of some sampling points on our analog signal.

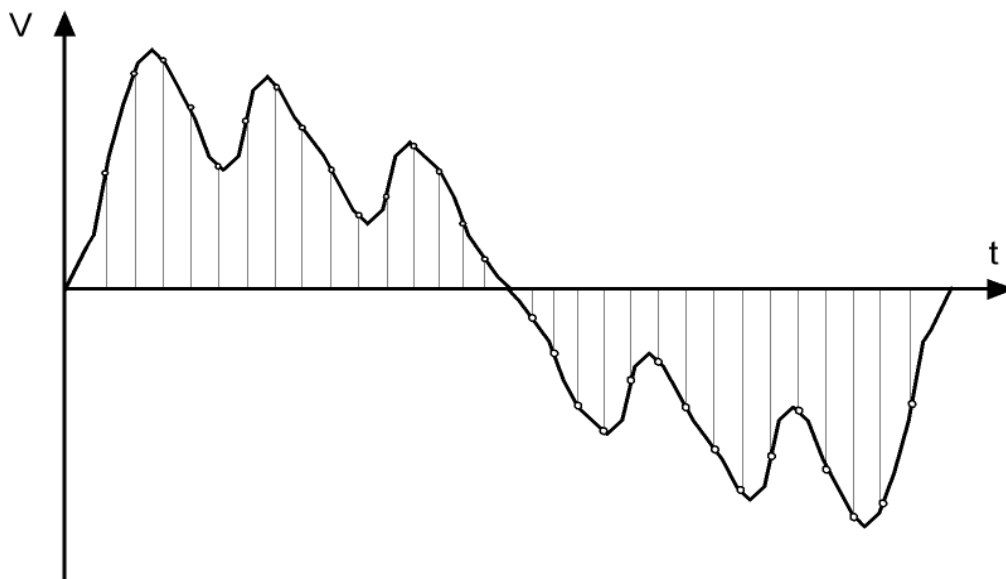


Figure 2: Sampling points.

The frequency on which the sampling will occur is called sampling rate. If a sampling rate of 22,050 Hz is used, for example, this means that in one second 22,050 points will be sampled. Thus, the distance of each sampling point will be of $1 / 22,050$ second (45.35 μ s, in this case). If a

sampling rate of 44,100 Hz is used, it means that 44,100 points will be captured per second. In this case the distance of each point will be of $1 / 44,100$ second or 22.675 μ s. And so on.

During the digital-to-analog conversion, the numbers will be converted again into voltages. If you think about it for a while, you will see that the waveform resulted from the digital-to-analog conversion won't be perfect, as it won't have all the points from the original analog signal, just some of them. In other words, the digital-to-analog converter will connect all the points captured by the analog-to-digital converter, any values that existed originally between these points will be suppressed.

You can see an example on Figure 3, where we show how the signal would be after being converted to digital and back to analog. As you can see, the original waveform is more "rounded".

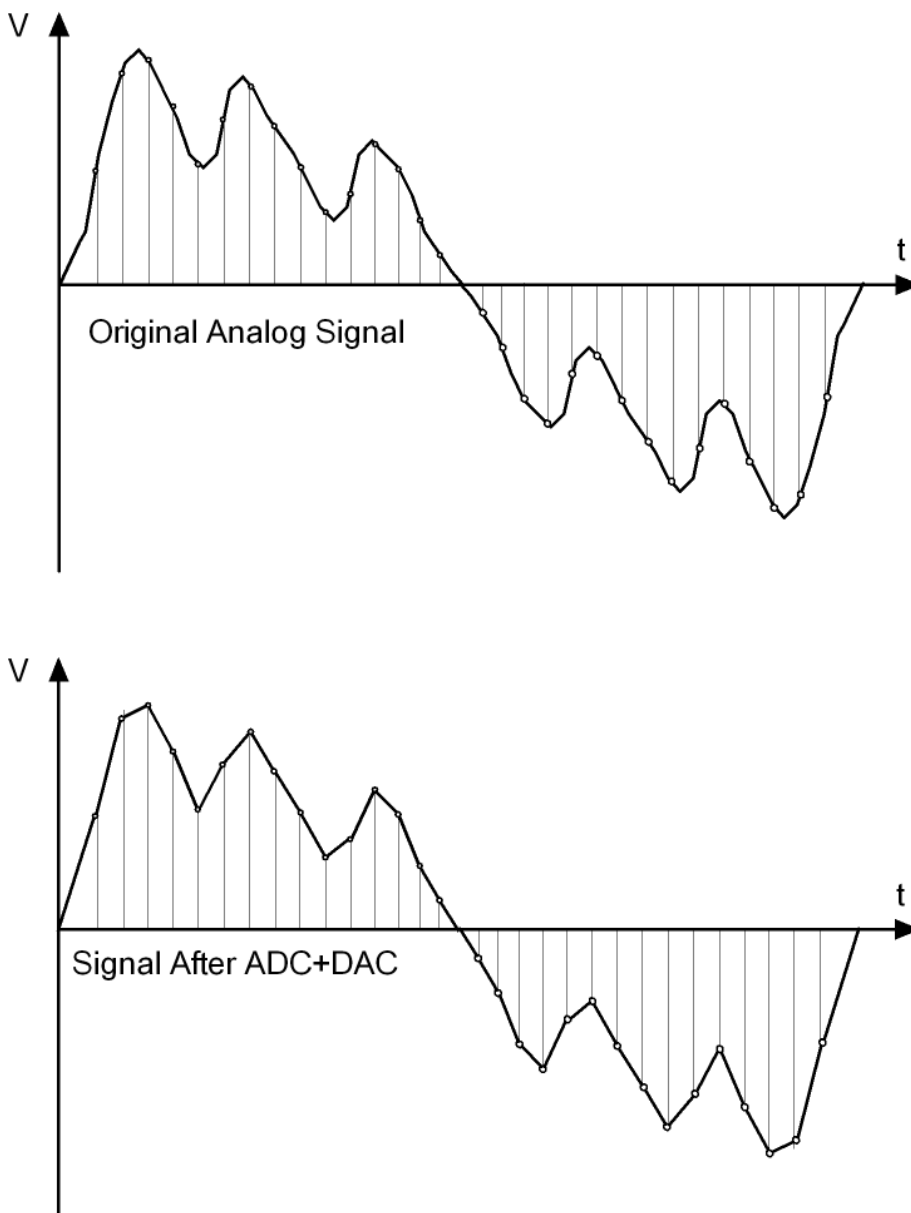


Figure 3: Signal after being converted to digital and back to analog.

So, the more sampling points we use – i.e. the higher the sampling rate –, the more perfect will be the analog signal produced by the digital-to-analog converter (DAC). However, the more samples we capture more storage space is necessary to store the resulting digital data. For example, an analog-to-digital conversion using a 44,100 Hz sampling rate will generate twice the number of data as a conversion using a 22,050 Hz sampling rate, as it will capture twice the samples from the original waveform.

If you use a low sampling rate, the waveform generated at the DAC will be very different from the original analog signal. If it is music, for example, the music you will play will have a very bad quality.

So, we have this dilemma: if the sampling rate is too high, the output quality will be close to perfection, but you will need a lot of storage space to hold the generated data (i.e. the generated file will be very big); if the sampling rate is too low, the output quality will be bad.

How can you know the best sampling rate to be used during analog-to-digital conversions to have the best storage/quality balance? The answer is the Nyquist Theorem.

This theorem states that the sampling rate on analog-to-digital conversions must be at least two times the value of the highest frequency you want to capture.

Since the human ear listens to sounds up to the frequency of 20 KHz, for music we need to use a sampling rate of at least 40,000 Hz. In fact, the CD uses a 44,100 Hz sampling rate, thus capturing more than our ears can hear (this value was arbitrated by Philips and Sony when they created the CD). Some professional audio applications use an even higher sampling rate.

The phone system, on the other hand, was created to transmit only human voice, which has a lower frequency range, up to 4 KHz. So on the digital part of the phone system, an 8,000 Hz sampling rate is used. That's why if you try to transmit music thru the phone the quality is bad: the phone circuitry cancels all frequencies above 4 KHz (ask a friend to put his/her phone near a stereo playing and you will hear what we are talking about).

How It Works: Resolution

The value of each sampled point will be stored on a fixed-length variable. If this variable uses eight bits, this means it can hold values from 0 to 255 ($2^8 = 256$). If this variable uses 16 bits, this means it can hold values from 0 to 65,535 ($2^{16} = 65,536$). And so on.

So, if you are using an 8-bit analog-to-digital converter, the lowest value will be zero and the highest value will be 255. If a 16-bit analog-to-digital converter is used, the lowest value will be zero and the highest value will be 65,535. See Figure 4.

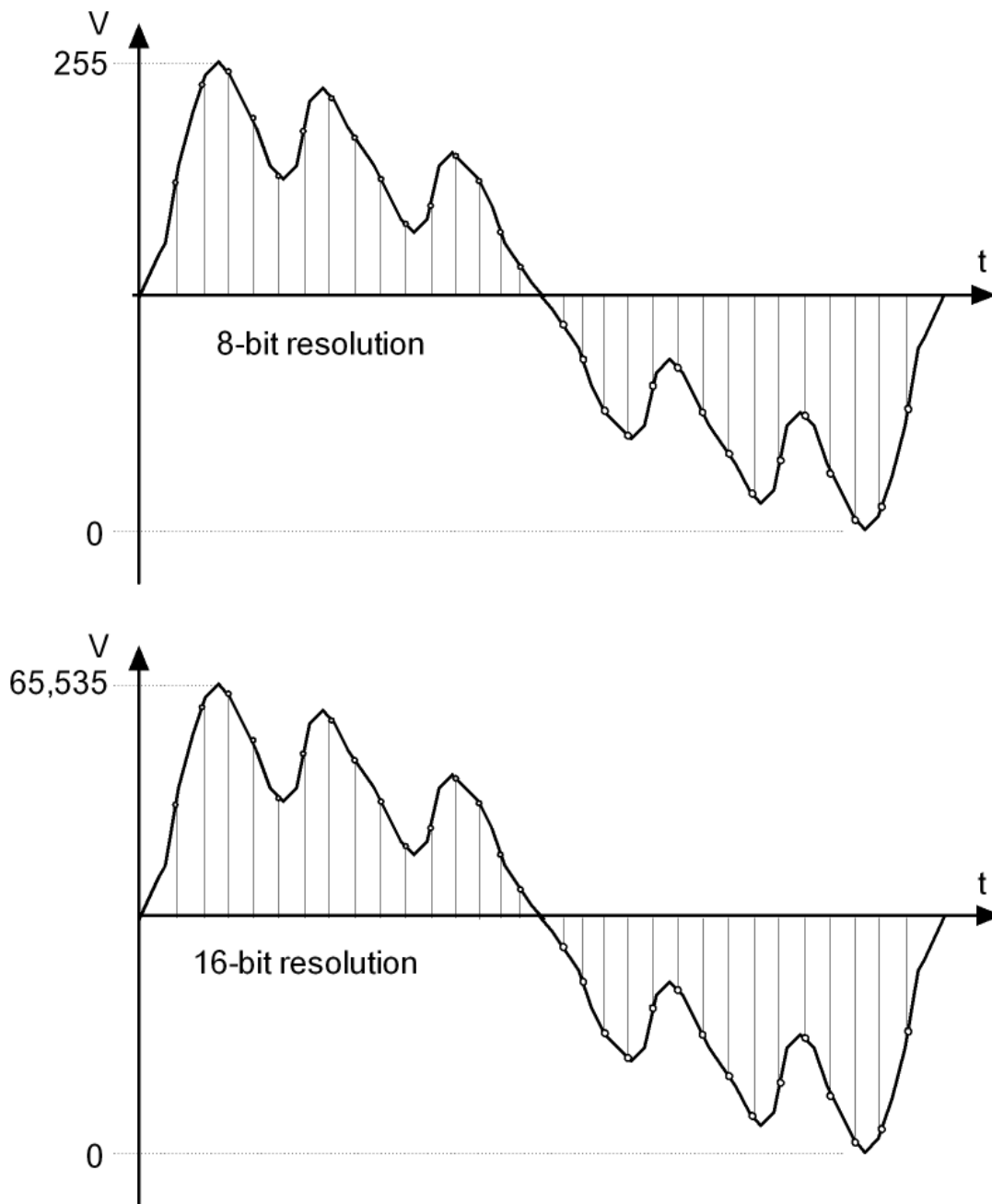


Figure 3: Signal after being converted to digital and back to analog. Inconvenience

So, the more sampling points we use – i.e. the higher the sampling rate –, the more perfect will be the analog signal produced by the digital-to-analog converter (DAC). However, the more samples we capture more storage space is necessary to store the resulting digital data. For example, an analog-to-digital conversion using a 44,100 Hz sampling rate will generate twice the number of data as a conversion using a 22,050 Hz sampling rate, as it will capture twice the samples from the original waveform.

If you use a low sampling rate, the waveform generated at the DAC will be very different from the original analog signal. If it is music, for example, the music you will play will have a very bad quality.

So, we have this dilemma: if the sampling rate is too high, the output quality will be close to perfection, but you will need a lot of storage space to hold the generated data (i.e. the generated file will be very big); if the sampling rate is too low, the output quality will be bad.

How can you know the best sampling rate to be used during analog-to-digital conversions to have the best storage/quality balance? The answer is the Nyquist Theorem.

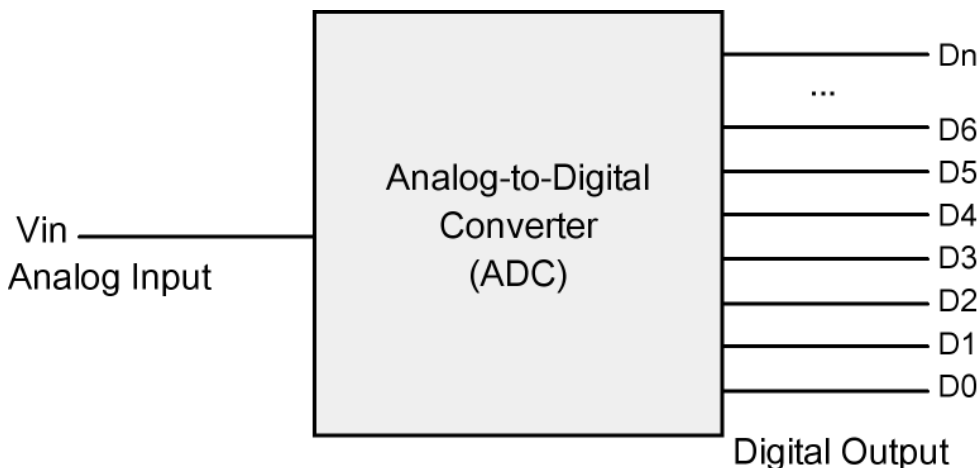
This theorem states that the sampling rate on analog-to-digital conversions must be at least two times the value of the highest frequency you want to capture.

Since the human ear listens to sounds up to the frequency of 20 KHz, for music we need to use a sampling rate of at least 40,000 Hz. In fact, the CD uses a 44,100 Hz sampling rate, thus capturing more than our ears can hear (this value was arbitrated by Philips and Sony when they created the CD). Some professional audio applications use an even higher sampling rate.

The phone system, on the other hand, was created to transmit only human voice, which has a lower frequency range, up to 4 KHz. So on the digital part of the phone system, an 8,000 Hz sampling rate is used. That's why if you try to transmit music thru the phone the quality is bad: the phone circuitry cancels all frequencies above 4 KHz (ask a friend to put his/her phone near a stereo playing and you will hear what we are talking about).

Inside an ADC

You can consider the analog-to-digital converter as a closed box, as shown on Figure 5. But what is inside the box? That is exactly what we are going to explain now



There are several ways to build an ADC. We can divide ADC design into four main groups:

- Parallel design (also known as Flash ADC);
- Digital-to-Analog Converter-based design (e.g. ramp counter, successive approximation, tracking);
- Integrator-based design (e.g. single-slope, dual-slope);
- Sigma-delta design (also known as delta-sigma, 1-bit ADC or oversampling ADC).

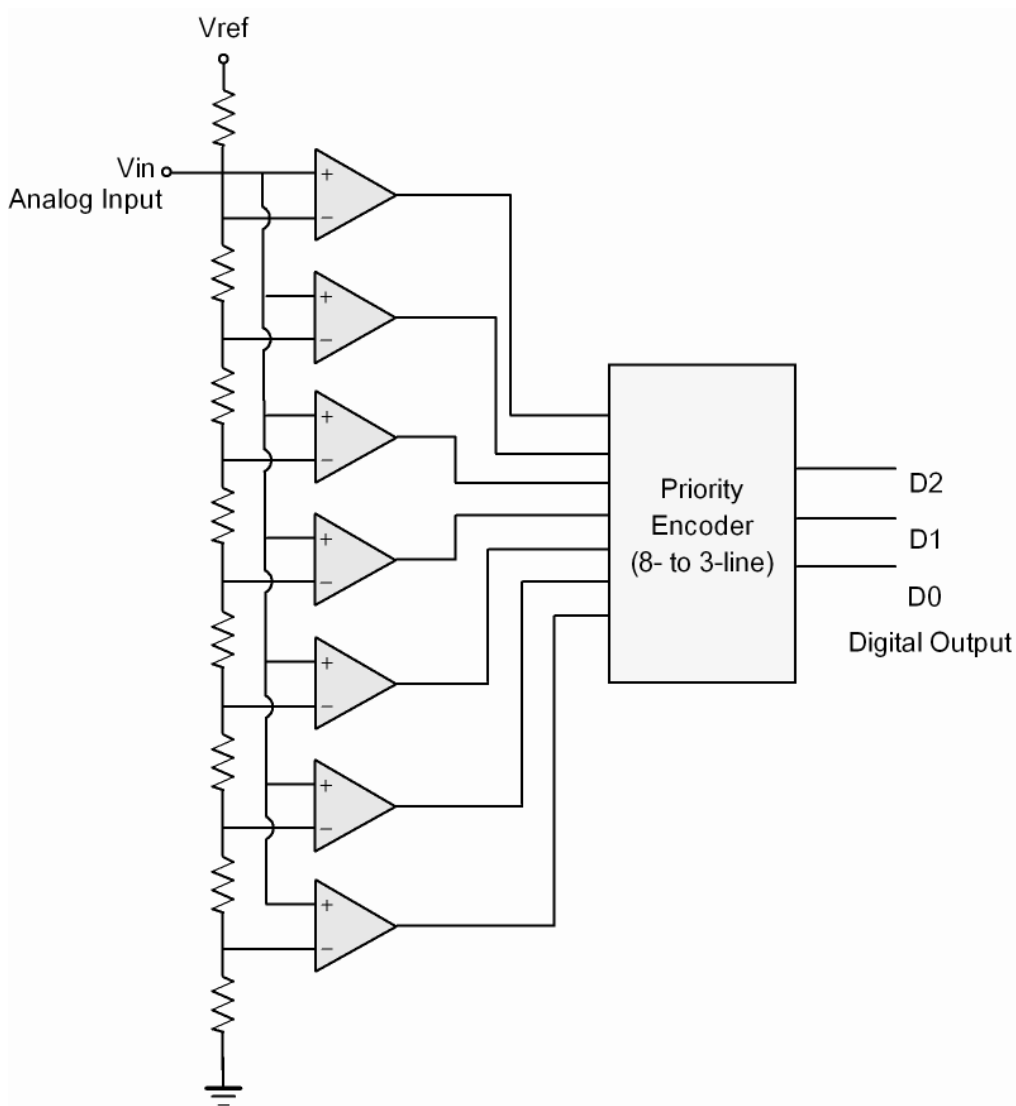
Each one of these main groups can have several different implementations. We are going to talk about each one of these groups individually.

Parallel Design

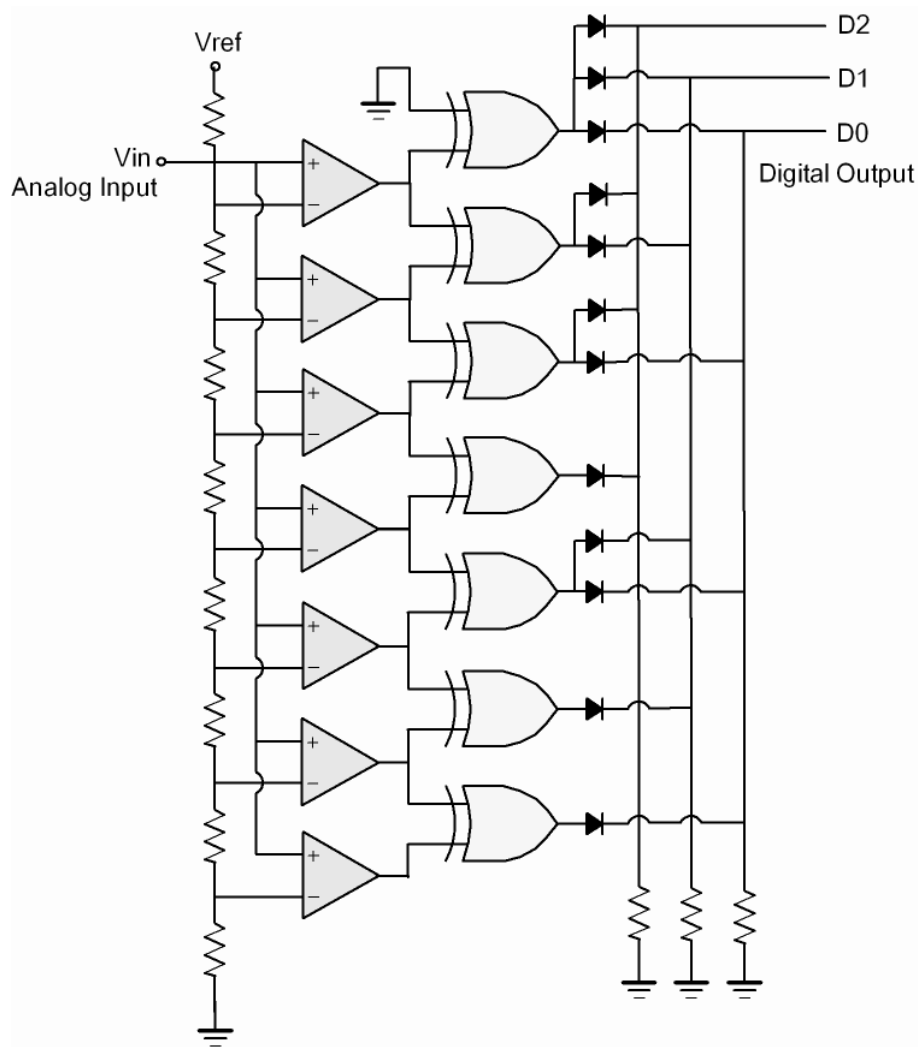
The Flash ADC, also called parallel ADC, is very easy to understand. It works by comparing the input voltage – i.e. the analog signal – to a reference voltage, which would be the maximum value achieved by the analog signal. For example, if the reference voltage is of 5 volts, this means that the peak of the analog signal would be 5 volts. On an 8-bit ADC when the input signal reached 5 volts we would find a 255 (11111111) value on the ADC output, i.e. the maximum value possible.

Then the voltage reference is lowered thru a resistor network and other comparators added, so the input voltage (analog signal) can be compared to other values.

On Figure 6 you can see a 3-bit Flash ADC. The comparison is done thru an op amp. All resistors have the same value.



The priority encoder can be done using XOR gates and a series of diodes and resistors, like shown on Figure 7, or a single chip like 74148 (3-line to 8-line priority encoder).



click to enlarge
Figure 7: Flash ADC.

Even though Flash ADC uses a very simple design, it requires a lot of components. The number of required comparers is $2^n - 1$, where n is the number of output bits. Thus for an eight-bit Flash ADC 255 comparers would be necessary, and for a 16-bit Flash ADC, 65,535!

On the other hand, Flash ADC is the fastest ADC type available. The digital equivalent of the analog signal will be available right away at its output (it will only have the propagation delay inserted by the logic gates) – hence the name “flash”.

Another advantage of Flash ADC is that you can create an ADC with non-linear output. Usually ADCs have a linear output, i.e. each digital number corresponds to a fixed voltage increase on the

It works by counting from 0 to the maximum possible value (2^n-1) until it “finds” the correct digital value for the analog voltage present at V_{in} . When this is true, the END signal is given and the digital value for V_{in} is for at D_n thru D_0 .

So the main problem with this circuit is that it is very slow, as it would require up to 2^n-1 clock cycles to convert each sample. For an eight-bit ADC, it would take up to 255 clock cycles to convert a single sample. For a 16-bit ADC it would take up to 65,535 clock cycles to convert one sample.

Successive Approximation ADC

The second classical ADC circuit using DAC design is called successive approximation, which is the most used one, shown on Figure 9. V_{in} is the analog input and D_n thru D_0 are the digital outputs. As you can see, it uses a buffer, so the digital data is still available while the converter is processing the next sample. SAR stands for Successive Approximation Register. It has the same control signals as the ramp counter ADC: START, which commands the ADC to start the conversion, CLOCK and END, which tells us that the conversion of that particular sample has finished.

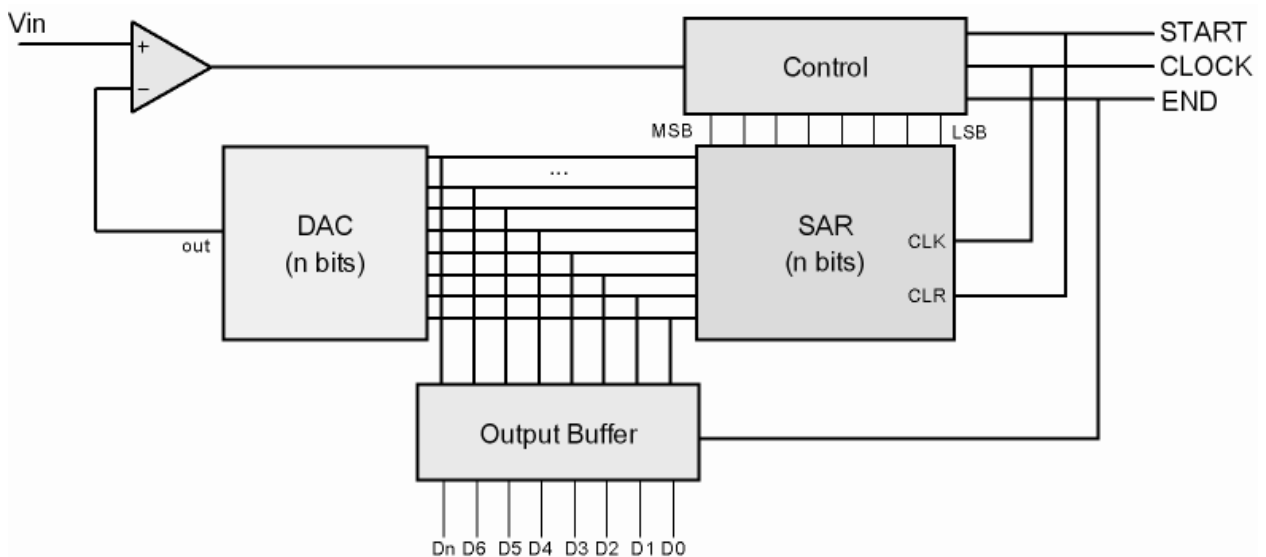


Figure 9: Successive approximation ADC.

While the ramp counter ADC does the analog-to-digital conversion counting from 0 to the maximum possible value (2^n-1) until it “finds” the correct digital value for V_{in} , the successive approximation ADC starts first setting the MSB (most significant bit, on an eight-bit ADC it would be D_7). In order to facilitate the explanations below, consider an eight-bit ADC.

The comparison between V_{in} and the DAC output will tell the control unit if this bit should remain set at 1 or should be set at 0, as the op amp will tell right away the control unit if the sample value is greater or lower than 128 (2^7). Then D_6 is set to one, and from the comparison done by the op amp, the control unit will know if this bit should remain set or not. And so on.

The good thing about the successive approximation ADC is its speed. At the worst case it will find the correct digital value for the sample at n clock cycles, where n is the number of bits used. For

an eight-bit ADC, the digital value for each sample can be found in up to eight clock cycles (compare to 255 on the ramp counter), and for a 16-bit ADC the digital value for each sample can be found in up to 16 clock cycles (compare to 65,535 on the previous circuit).

And, as we mentioned, another great advantage of this circuit is the use of an output buffer, which allows the circuit that is fed by the ADC to read the digital data while the ADC is already working on the next sample.

Integrator-Based Designs

There are a few ways of designing analog-to-digital converters using an integrator. Let's take a look on two of them, single-slope ADC and delta-sigma ADC

Single-Slope ADC

On Figure 10 you can see a single-slope ADC. If you pay close attention, you will see that it is very similar to a ramp counter ADC, as it uses a counter, but instead of using a DAC for generating the comparison voltage, it uses a circuit called integrator, which is basically formed by a capacitor, a resistor and an operational amplifier (op amp). The MOSFET transistor makes the necessary control circuit.

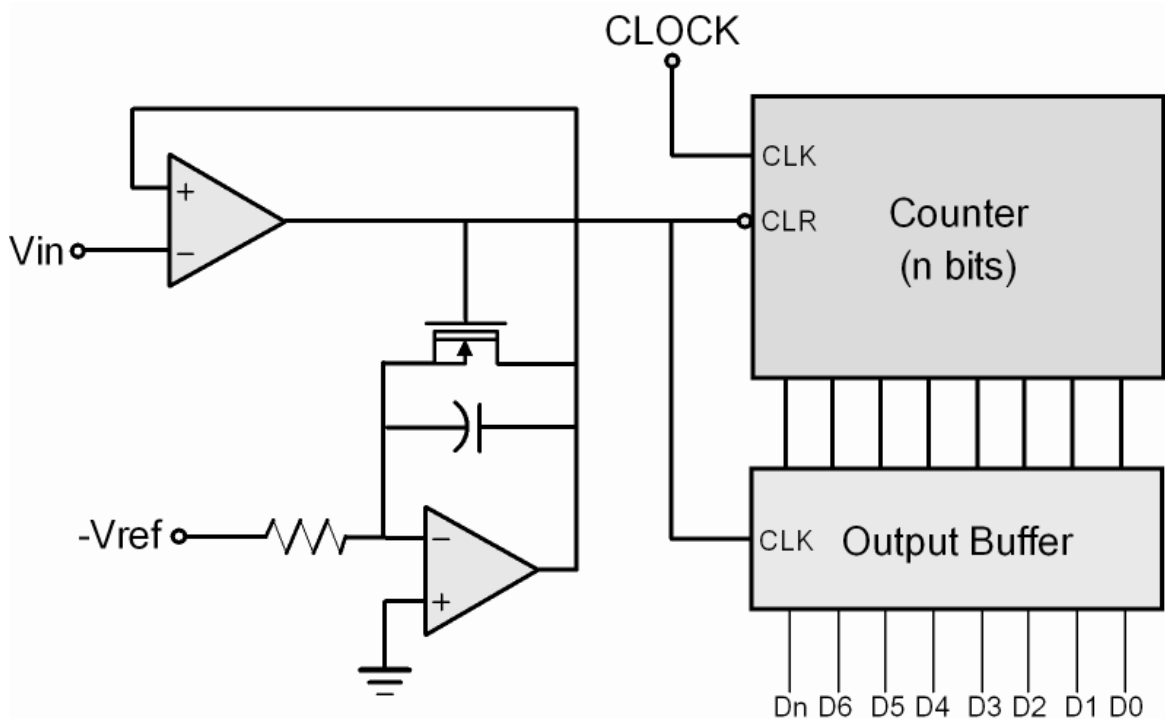


Figure 10: Single-slope ADC.

The integrator produces a sawtooth waveform on its output, from zero to the maximum possible analog voltage to be sampled, set by $-V_{ref}$. The minute the waveform is started, the counter starts counting from 0 to $2^n - 1$, where n is the number of bits implemented by the ADC. When the voltage found at V_{in} (the analog signal) is equal to the voltage achieved by the triangle waveform

generated by the integrator, the control circuit captures the last value produced by the counter (by triggering the output buffer clock pin), which will be the digital correspondent of the analog sample being converted. At the same time, it resets the counter and the integrator, starting the conversion of the next sample.

Like the successive approximation ADC, this circuit uses an output buffer, meaning that the last converted value can be read while the ADC is converting the current value.

Even though its design is simpler than ramp counter design, it is still based on a counter, and thus suffering from the same basic problem found on ramp counter design: speed. It requires up to $2^n - 1$ clock cycles to convert each sample. For an eight-bit ADC, it would take up to 255 clock cycles to convert a single sample. For a 16-bit ADC it would take up to 65,535 clock cycles to convert one sample.

Dual-Slope ADC

Another popular design based on this one is called dual-slope ADC, which solves an inherent single-slope problem called calibration drift, which leads to inaccuracy over time because the integrator isn't linked to the clock signal (i.e., the sawtooth waveform isn't synchronized with the counter clock).

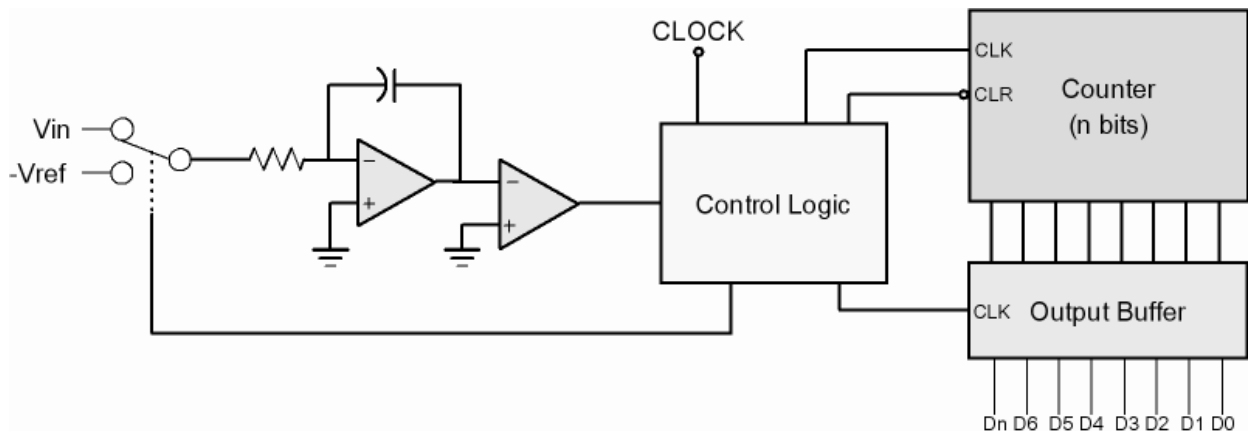


Figure 11: Dual-slope ADC.

The analog switch first connects V_{in} to the integrator. With that, the integrator starts generating the sawtooth waveform, and the switch position will remain set at V_{in} during a fixed number of clock cycles. When this number of clock cycles is reached, the analog switch moves its position to allow $-V_{ref}$ to enter the integrator. Since $-V_{ref}$ is a negative voltage, the sawtooth waveform goes towards zero, using a number of clock cycles proportional of the V_{in} value.

For a better understanding, see Figure 12, where we show the waveform at the integrator output. So, T_1 is fixed, while T_2 duration is proportional to the value of V_{in} . V_{in} sets the slope angle: the higher V_{in} is, the higher the angle will be.

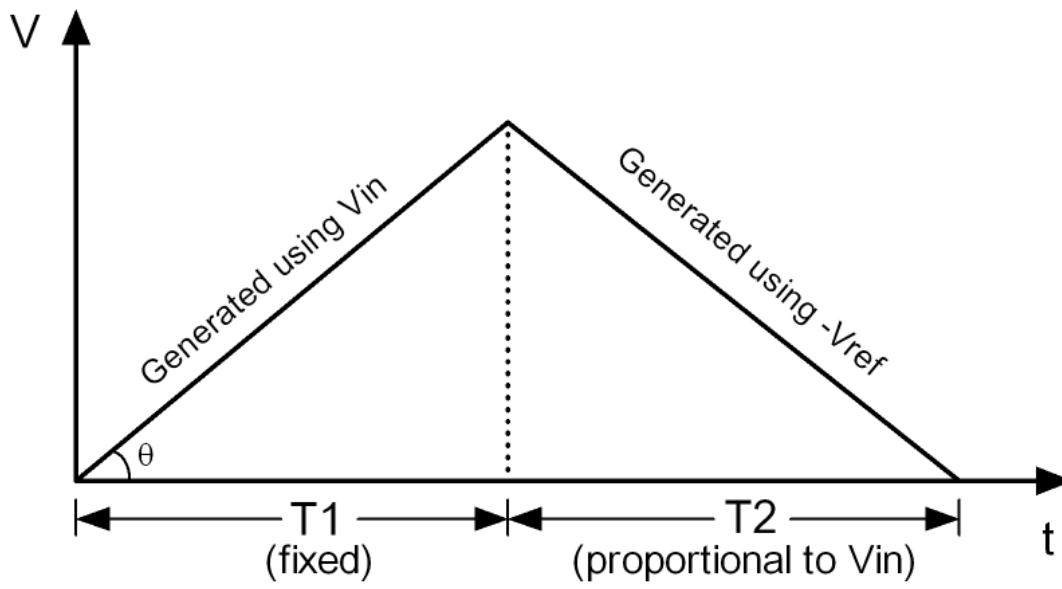


Figure 12: Waveform found at the integrator output.

$$T_2 = T_1 \times V_{in} / V_{ref}.$$