1 Introduction

In old physics experiments the data acquisition was matter of looking and noting on paper, and data reduction was hand made by people particularly suited to make calculations and thus called *Computers*. A typical data acquisition chain in these old times was characterised by the following elements:

- Sensors: systems that excited by a certain form of energy, representing the physical phenomenon to observe, give in exit a different form of energy (e.g. light into electricity).
- Transducers: systems that convert (translate) a form of energy into another (e.g. electrical into mechanical);
- Recorders: systems to translate the acquired signal in a series of numbers or graphs that could be studied in a second moment. They could record continuously or at discrete times and could be also humans with paper and pencils.
- Computers: Humans that take the recorded data and elaborate them to obtain an aggregate result.

When computers started to be electronic devices, the data acquisition chain changed, but the elements remained essentially the same. Recorders and computers became electronic devices, linked to one another and the data that entered into the computing system had to be formatted in a way understandable to computers, i.e. they had to be in *digital* form.

This new requirement to the data acquisition chain introduced a new element: the *Converters*, i.e. electronic devices that convert a continuous-time, continuous-value (Analogic) signal to discrete time, numerical (digital) data; this is the Analog to Digital Converter (ADC). The inverse converter – the Digital to Analog Converter (DAC) – is needed to get an electrical signal from a numerical data stream, in order to realise controlled systems that, from the acquired data, derive a signal that is used to take actions (actuate) on the system itself. An axample of a mechanical controlled system is the damping of a pendulum: a capacitive sensor detects the mass motion, the electrical signal is converted in a numerical stream, a computer applies a numerical filter (e.g. simply invert) to the input data stream and sends the computed *error* data stream to the DAC that produces an electrical signal sent to an actuator that exerts forces on the mass.

2 Sampling

As said in previous section, analogic signals have the characteristics to be continuous in time and value. The computer takes some time to elaborate input signals and thus cannot follow the signal continuous evolution; it will, instead, sample the signal value at discrete times. This operation is the signal *sampling*.

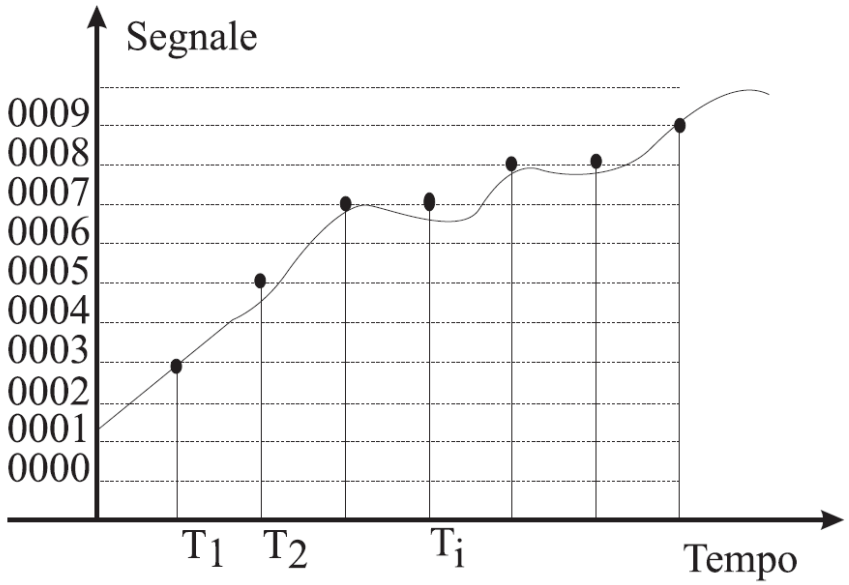


Figure 1: The operation of sampling and quantization: the continuous signal is taken at discrete times and is coded by integer values. This introduces errors.

Moreover, the computer has a limited resolution given by the number of bits of its word, thus it will not give the actual value of the signal at the time of sampling, but only the nearest value representable with this word. This operation is called *Quantization*. The information linked with the sampled and quantized analogic signal is called *digital signal*. The digital signal is affected by errors by the simple digitalization and quantization process. Let's start from the error introduced by the quantization process.

The transducer range Y^s is the full range the values of the output signal can assume from the minimum Y_m to the maximum Y_M, $Y^s = Y_M - Y_m$. Given a computer with a word of N bits, it can represent 2^N numbers by this word, thus, each bit will correspond to a value (called Least Significant Bit – LSB):

$$LSB = \frac{Y^s}{2^N}.$$

This value, also called quantum, is the minimum difference that can be represented by the digital coding. All the values within a LSB will be coded by a unique digital value Y_d, thus the error introduced by the process of quantization to the original signal is one half of the LSB:

$$\epsilon_q = \frac{LSB}{2} = \frac{Y^s}{2^{N+1}}.$$

2.1 Spectral behaviour of sampled signals

Let's denote the continuous signals as function of the continuous time $f(t)$ while the sampled signal as a function of the discrete time reperesented by the sample number n $F(n)$. Let us begin to examine the differences between the continuous and the discrete version of the complex exponential function. For the continuous $f(t) = e^{i\omega t}$ the larger is ω the larger is rate of oscillations and is periodic for any value of ω. The discrete counterpart has a different behaviour, infact considering the complex exponential with frequency $(\Omega_0 + 2\pi)$:

$$F(n) = e^{i(\Omega_0+2\pi)n} = e^{i\Omega_0 n}e^{2i\pi n} = e^{i\Omega_0 n}. \tag{1}$$

As we see it has the *same* frequency as the function with frequency Ω_0, as is for signals with frequency $(\Omega \pm 2\pi)$, $(\Omega \pm 4\pi)$...Therefore in considering discrete time complex exponentials we need to consider only an interval of length 2π.

As far as the periodicity is concerned, in order for the signal to be periodic of period N, it must satisfy the relation:

$$F(n) = F(n+N) \Rightarrow e^{i\Omega n} = e^{i\Omega(n+N)} \tag{2}$$

or equivalently: $e^{i\Omega N} = 1$, that is $\Omega N = 2k\pi$, i.e. $e^{i\Omega n}$ is not periodic for any Ω, but only for $\Omega = 2\pi k/N$.

The discrete-time sequence obtained by sampling a continuous complex exponential signal at equally spaced points in time is:

$$X[n] = e^{i\omega nT}$$

that is a discrete time exponential of period ωT, being T the interval between the samples, i.e. the reciprocal of the sampling pulsation: thus, if $\omega_s = 2\pi/T$ is the sampling frequency, the function will be periodical only if ω/ω_s is a rational number.

Any continuous signal x(t), sampled with a period T can be represented in terms of a sum:

$$x_p(t) = \sum_{n=-\infty}^{\infty} x(nT)\delta(t - nT) \tag{3}$$

By Fourier transforming this expression, we see that it is the convolution:

$$X_p(\omega) = \frac{1}{2\pi}[X(\omega) * P(\omega)] \tag{4}$$

where

$$P(\omega) = \frac{2\pi}{T} \sum_{k=-\infty}^{\infty} \delta(\omega - k\omega_s) \tag{5}$$

so

$$X_p(\omega) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X(\omega - k\omega_s) \tag{6}$$

That is $X_p(\omega)$ is a sum of replicas of $X(\omega)$ scaled of 1/T (see Fig2).

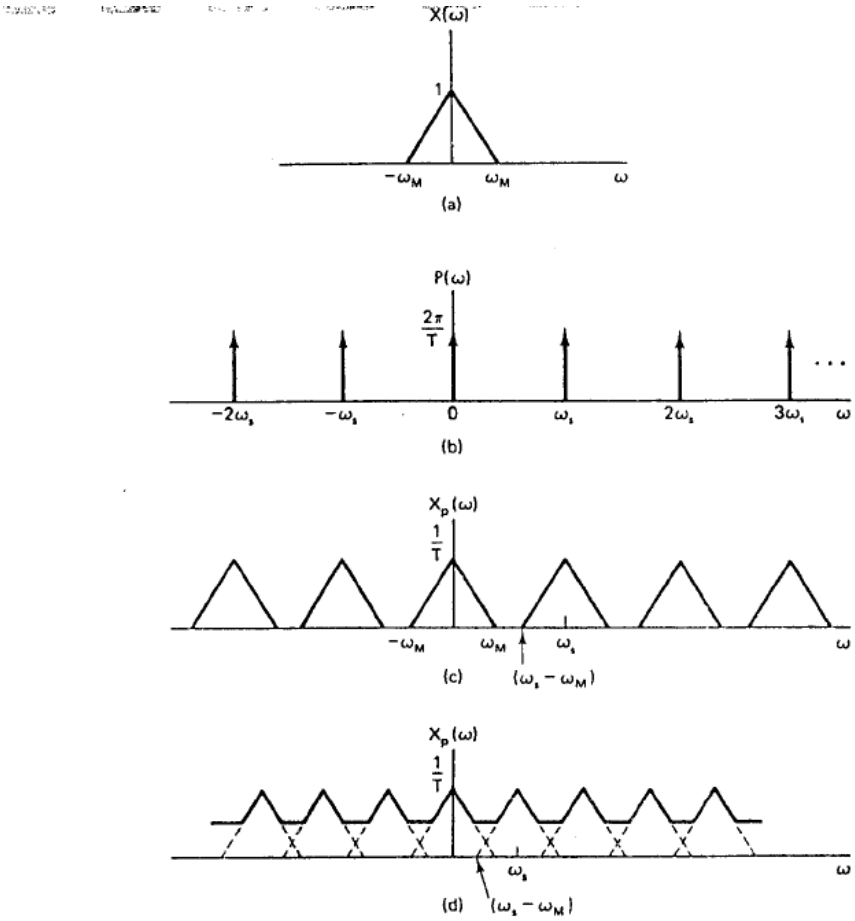


Figure 2: The effect in frequency domain of sampling in time domain. The Fourier transform of the signal to sample has a maximum frequency ω_M, if the sampling frequency $\omega_s > 2\omega_M$ the replicas do not overlap

If the spectrum of the original signal has a maximun frequency ω_M, then if $\omega_M < (\omega_s - \omega_M)$ i.e. $\omega_s > 2\omega_M$ there is no overlap of the replicas and the signal can be recovered exactly from the sampled signal by applying a low-pass filter with cut frequency $\omega_M < \omega_c < (\omega_s - \omega_M)$. This is what is called *the sampling theorem*, the frequency $\omega_s = 2\omega_M$ is called the *Nyquist frequency.*

The samping theorem can be enunciated as follows:

Theorem 1 *Given a band-limited signal x(t) with $X(\omega) = 0$ for each $|\omega| > \omega_M$, then x(t) is uniquelly determined by its samples $x_p(nT)$ if*

$$\omega_s > 2\omega_M$$

where

$$\omega_s = 2\pi/T$$

is the sampling frequency.

2.2 The effect of undersampling: Aliasing

As was illustrated in Fig. 2, with $\omega > 2\omega_M$, the spectrum of the sampled signal consists of exact replications of the spectrum of x(t), and this forms the basis for the sampling theorem. When $\omega < 2\omega_M$, $X(\omega)$, the spectrum of x(t), is no longer replicated in $X_p(\omega)$ and thus is no longer recoverable by lowpass filtering. This effect, in which the individual terms in the sum (6) overlap, is referred to as *aliasing*, and in this section we explore its effect and consequences.

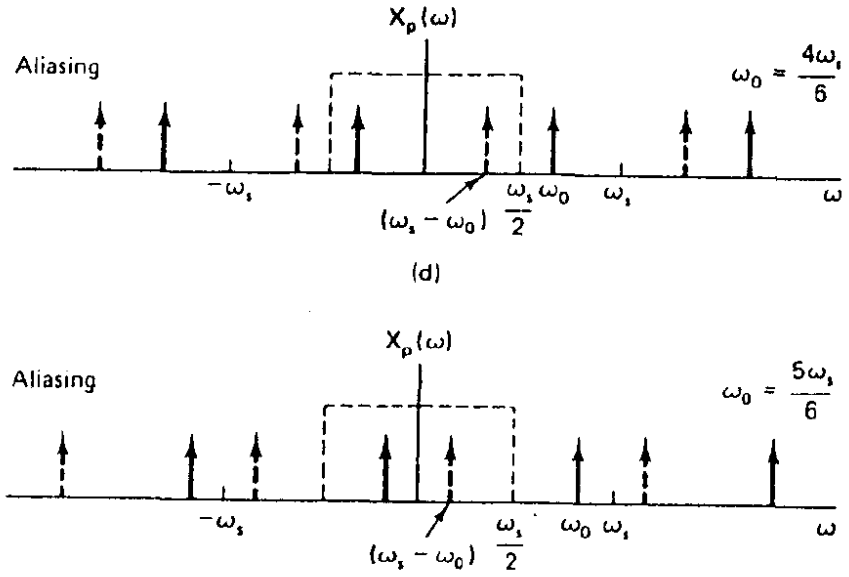


Figure 3: The effect of sampling a sinusoidal function at a frequency lower than the Nyquist frequency.

Let us consider the simple case of a sinusoidal signal:

$$x(t) = cos(\omega_0 t)$$

Its spectrum will consist of only two lines at $\pm\omega_0$. The spectrum of the signal sampled at frequency $\omega_s < 2\omega_0$ is shown in Fig. 3, in the cases of $\omega_0 = 4/6\omega_s$ and $\omega_0 = 5/6\omega_s$, the low-pass filter band is also indicated. In the two cases, the output of the filter is:

$$\omega_s \Rightarrow x_f(t) = cos(\omega_s - \omega_0) \neq x(t)$$

The effect of aliasing in the time frequency is shown in Fig. 4

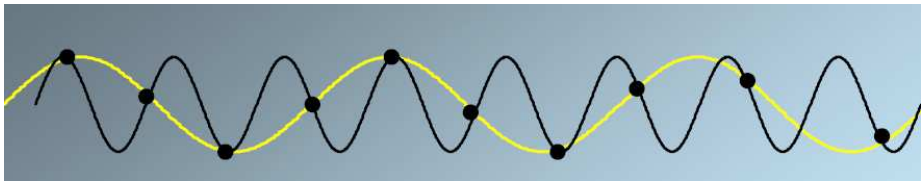


Figure 4: The effect of sampling a sinusoidal function at a frequency lower than the Nyquist frequency in the time domain.

2.3 Other elements of the Acquisition chain

Another element of the modern data acquisition chain are signal amplifiers. They are needed to efficientely use the full scale of the converters (and thus improve the resolution) and to enhance the Signal to Noise Ratio (SNR). To visualize the first use, imagine to have a 12 bit converter with a $\pm 5\ V$ input and a signal of 10 mV peak to peak in input. It will be digitised in a $10\ V/2^{12} = 2.4\ mV/count$ so the signal will be ~ 4 ADC counts (2 bits); if one bit flips due to the internal ADC noise we have an error of 50%. If we amplify the signal of a factor 1000 before sending it to the ADC we use the whole range and a one bit flipping accounts for $1/4096 = 0.02\%$. The other topic can be understood if we imagine the previous 10 mV signal to be transmitted along a transmission line before entering the ADC. It picks up an electromagnetic noise along the line, e.g. 1 mV p-p, the SNR will be: $10mV/1mV = 10$. By inserting a signal amplifier $\times 1000$ before the transmission line, the signal is amplified, while the pick-up remains the same, thus the final SNR will be 10000.

Often it is necessary to digitize several channels using a single converter. This task is accomplished by the *Multiplexer* (MUX see Fig. 5). It is a device with N analogic inputs, $log_2 N$ digital inputs and one output. The digital inputs

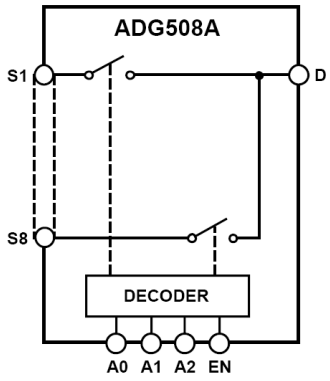
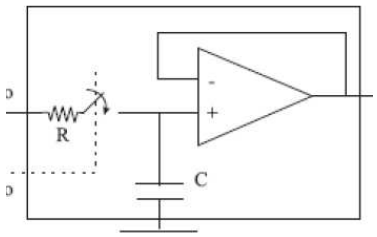


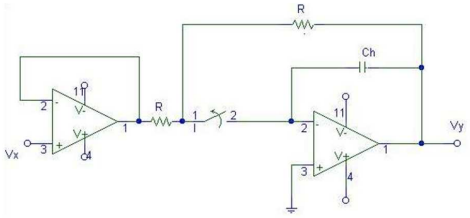
Figure 5: An example of 8 channel multiplexer. The output "D" is set to follow one of the inputs S1-S8 depending on the configuration of the A0-A3 bits if the EN signal is present.

can code a number from 0 to N-1, when this code is present at the same time of a Enable (clock) signal, the output is the signal present on the corresponding input channel. Thus at each clock pulse the output will be a different channel, in a cyclic way. The reverse operation is accomplished by the *Demultiplexer* (DEMUX).

The last element of the acquisition chain, nowadays always included in modern monolithic ADCs, is the *Sample and Hold [Amplifier]* (SHA). The purpose of this circuit is to hold the analogue value steady for the time while the converter or other following system performs some operation. In particular, one wants that the signal do not change while the ADC is converting its value. The basic circuit of the SHA, is shown in Fig. 6. It is essentially a unity gain amplifier in the sample phase and a RC circuit with a long time constant in the hold phase. When the switch is opened the capacitor discharges on the load resitance that must be very high. The requirements on the capacitor in the sample and hold



(a) Basic Circuit



(b) Integrator SHA

Figure 6: The basic circuit for a Sample and Hold amplifier. When the switch is closed it is esssentially a unity gain amplifier (emitter follower). When the switch opens, the capacitor discharges with the $R_L C$ time constant, being R_L the load resistor. In 6b the integrator SHA is shown to overcome the problems in the chioice of the capacitor.

phases are opposite since RC must be low in the sample phase and high in the hold phase , for this reason other schemes are adopted, as e.g. the integrator

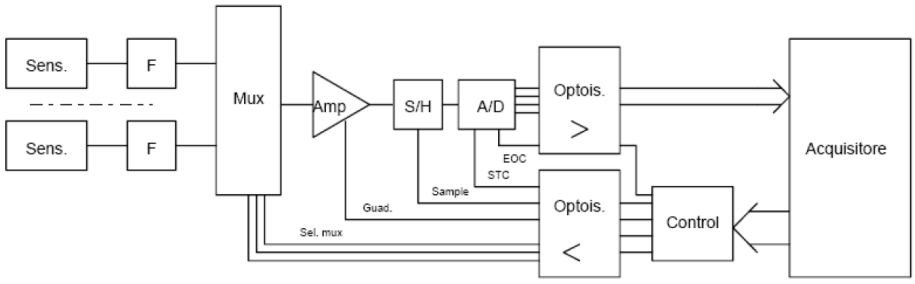


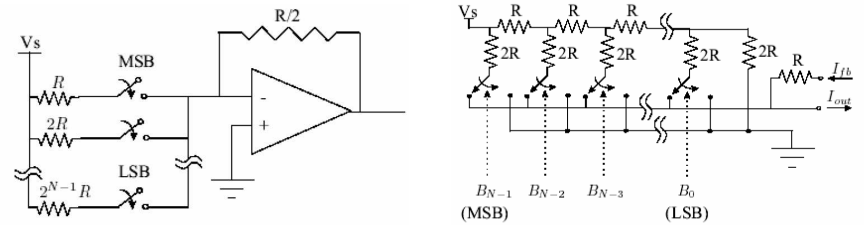
Figure 7: The data acquisition chain and its elements

SHA, shown in Fig.6b.

After these considerations, the acquisition chain can be summarised as in Fig. 7, where the opto-isolators are also shown, to isolate the digital information from EMC noise.

3 Digital to Analog and Analog to Digital Converters

We start by describing the Digital to Analog Converter, because it is needed as a part of some ADCs. Its basic circuit is shown in Fig. 8a and is essentially an analogic adder with switches corresponding to the input bits. Each addendum is the double of the previous, thus the simplest scheme is to have N input resistors of value $2^k R$ for $k = 0..N-1$ and a feedback resistor of value $R/2$, so that the k-th bit will give a contribution $(2^{-1}R/2^k R)V_s = 2^{-(k+1)}V_s$. In practical



(a) Basic DAC

(b) R-2R Scheme

Figure 8: The basic circuit for a Digital ro Analog Converter and the R-2R scheme that uses only two kind of resistors.

circuits, it is difficult to find as many precision resistors as the number of bits with the desired values, thus a different scheme using only two kind of resistors, of values R and 2R, is used as shown in Fig. 8b.

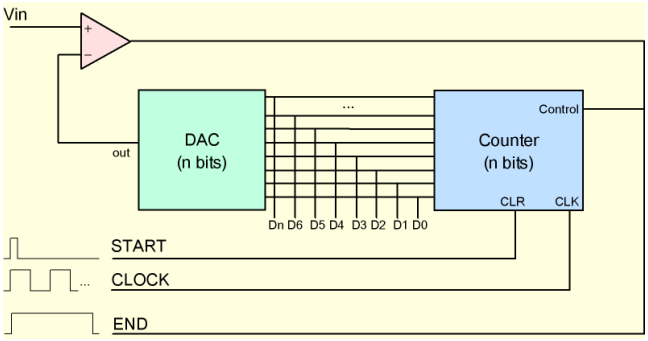


Figure 9: The counter ADC

The simplest type of ADC is the counter ADC. It is composed by a digital counter, a comparator and a DAC as shown in Fig. 9. The counter gives in output a digital number corresponding to the number of clock pulses, the DAC converts it to a voltage that is compared to the input voltage. When the converted output is equal to the input voltage, the comparator switches to a high TTL level that stops the counter. The number of clock pulses is then proportional to the input voltage. To convert N bits it requires 2^N steps and requires a start signal to reset the counter A variation on this scheme that do not use a DAC, uses a RC circuit connected to a V_{cc} in input to the comparator. The capacitor charges with a rate RC and, when the voltage reaches the input value the comparator switches. This is known as the *Single Slope integration ADC* and its resolution is dependent on the slope of the RC charge. The double slope scheme is independent on the time constant RC used to generate the linear ramp, because the output value is given by the ratio of the charge and discharge times of the capacitor. There is another widely used variation on the scheme of the counter ADC that needs only N steps to convert N bits, the *Successive Approximation Register (SAR)* ADC (Fig. 10). The successive approximation ADC starts first setting the MSB. 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. Then (n-1)th bit 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 sigma-delta ADC uses a different approach. We can divide it into two major blocks: analog modulator, which takes the analog signal and converts it into a stream of bits, and digital filter, which converts the serial stream from the modulator into an usable digital number. The basic sigma-delta modulator design can be found on Fig. 11. The analog signal will make the first op amp, which is a summing integrator, to create a sawtooth waveform proportional to the analog signal voltage. This sawtooth waveform found on the integrator output is then compared with zero volts by the second op amp, which is a comparator. It can be considered a 1-bit ADC, since its output will have only

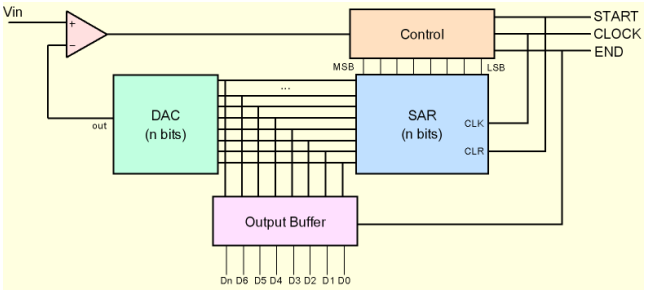


Figure 10: The Successive Approximation ADC

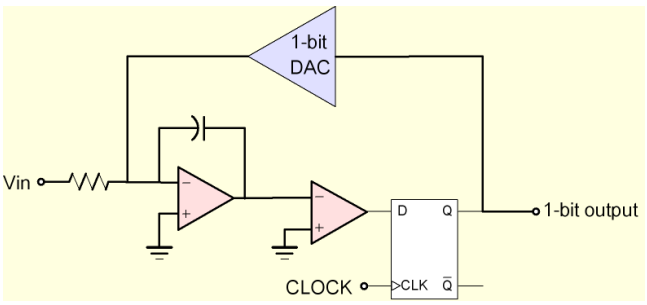


Figure 11: The Sigma-Delta modulator schematics

two states, high or low, depending whether the integrator output is positive or negative. The comparator output is stored on the D-type flip-flop, which is a one-bit static memory. This flip-flop is clocked at a very high frequency. Then the flip-flop output is used to feedback the circuit thru a one-bit DAC. This one-bit DAC will basically convert the 0 or 1 stored at the flip-flop into a positive or negative reference voltage to be added to the input of the summing integrator. So the summing integrator will sum the next sample with the result of the previous sample (a positive or a negative voltage), aiming to maintain zero at the integrator output. The result is that at the flip-flop output we will have a series of zeros and ones that correspond the sampled data: the bitstream average level represents the analog input signal average voltage.

Since the clock rate used at the flip-flop is very high, data is sampled many times over, a technique known as oversampling. As we will see later, the higher the clock, the higher the precision of the sigma-delta ADC. However, due to oversampling, the quantization noise is thrown to the high frequencies of the spectrum, and not spread all over the spectrum as it occurs with other designs. This effect is called shaped noise. With all the noise concentrated in a specific portion of the spectrum on a frequency range above the sampled data, it is quite easy to construct a low pass filter to remove it, thus improving SNR.

The fastest (and most expensive) of all the ADCs is the *flash ADC* (Fig. 12.

It is composed a series of comparators each one comparing the input signal to

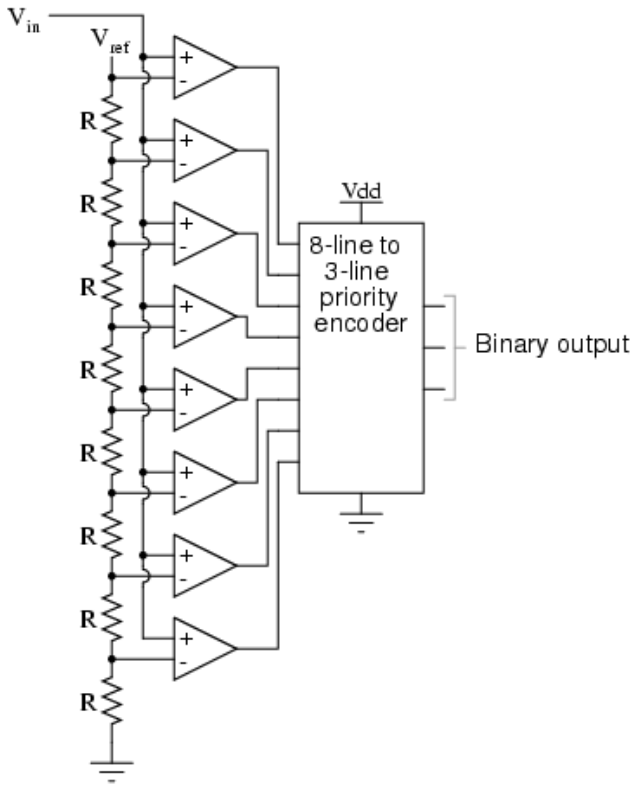


Figure 12: The Flash ADC schematics

a unique reference voltage derived from a reference voltage through a resistive network. The comparator outputs connect to the inputs of a priority encoder circuit, which then produces a binary output. To obtain a N bit flash ADC 2^{N-1} comparators are needed, but it converts the input voltage in only one clock pulse.

4 Effective number of bits

One of the ways to know the necessary number of bits for an ADC is by calculating the desired noise level. As we saw in paragraph 2, the quantization process introduces an error, i.e. a noise in the converted stream. Since the converted value is constant during the interval between two clock pulses, the mean squared quantization error will be (see Fig. 13):

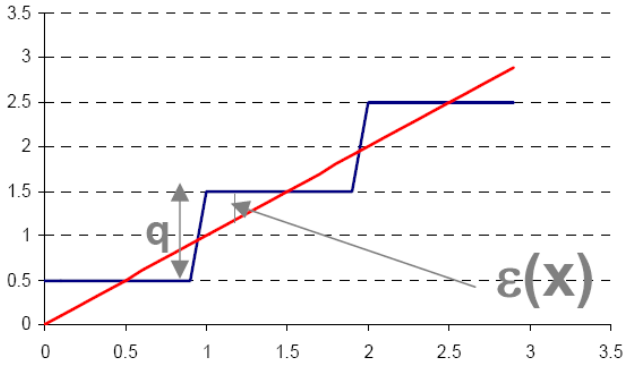


Figure 13: The quantization error parameters

$$\epsilon^2 = \frac{1}{q}\int_{-q/2}^{q/2} x^2\, dx = \frac{q^2}{12} \tag{7}$$

Encoding a sinusoidal signal $(A/2)sin(\omega t)$ with an n-bit ADC with a full scale A, will give an error:

$$\epsilon^2 = \frac{q^2}{12} = \frac{\frac{A^2}{2^{2n}}}{12} \tag{8}$$

The Signal to noise Ratio (SNR) due to quantization error will be:

$$SNR = 10\ log\left(\frac{x^2}{\epsilon^2}\right) = 10\ log\left(\frac{\frac{A^2}{8}}{\frac{A^2}{12\cdot 2^2 n}}\right) = 6n + 1.8\ dB \tag{9}$$

Calling n' the *effective number of bits* of the n bits ADC that give the desired SNR, we have:

$$n' = \frac{SNR - 1.8}{6} \tag{10}$$

For example, a 12-bit ADC giving a SNR=70 dB, will behave as an ADC with:

$$\frac{70 - 1.8}{6} = 11.37$$

effective bits.

4.1 Effect of oversampling

If we sample the signal with a sampling frequency f_s, assuming the noise to be white, its power spectral density (PSD) will be flat in the range $[-f_s/2, f_s/2]$, thus, the error PSD will be:

$$\frac{q}{\sqrt{12}}\frac{1}{\sqrt{f_s}} \tag{11}$$

If the frequency of the sampled signal f_0 is lower than f_s, after filtering the error will be:

$$\frac{q}{\sqrt{12}}\frac{1}{\sqrt{f_s}} = \frac{A}{2^n}\frac{1}{\sqrt{12}}\sqrt{\frac{f_0}{f_s}} \tag{12}$$

thus the previous calculation for the SNR will give:

$$SNR = 6n + 1.8 + 10\ log\left(\frac{f_s}{f_0}\right) = 6n' + 1.8\ dB \tag{13}$$

thus

$$n' = n + 1.67\ log\left(\frac{f_s}{f_0}\right) \tag{14}$$

hence it is possible to increase the resolution by increasing the sampling frequency and filtering.

For example an 8-bit ADC becomes a 9-bit ADC with an over-sampling factor of 4, but the 8-bit ADC must meet the linearity requirement of a 9-bit.