Imperial College London

Lecture 12

Analogue-to-Digital Conversion

Peter Cheung
Department of Electrical & Electronic Engineering
Imperial College London

URL: www.ee.imperial.ac.uk/pcheung/teaching/E2_CAS/
E-mail: p.cheung@imperial.ac.uk

Lecture Objectives

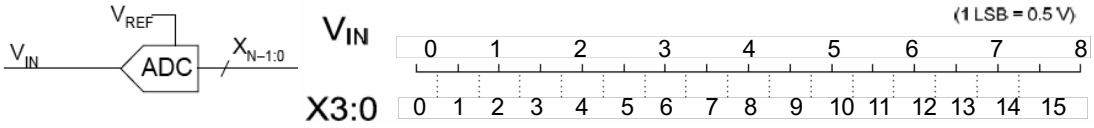
- Understand the relationship between the continuous input signal to an Analog-to-Digital converter and its discrete output
- Understand the source and magnitude of quantisation noise
- Understand how a flash converter works
- Understand the principles behind a successive approximation converter
- Understand how a successive approximation converter can be implemented using a state machine
- Understand the need for using a sample/hold circuit with a successive approximation converter

References:

- "Data Converter Architectures" in Data Conversion Handbook by Analog Devices

The counterpart to a DAC is the ADC, which is generally a more complicated circuit. In this lecture, various ADC architectures will be considered.

Analogue to Digital Conversion



- Converters with only +ve input voltages are called ***unibipolar*** converters and usually round (V_{IN} ÷ 1LSB) to the ***nearest*** integer.

$$X = \text{round}\left(\frac{V_{IN}}{1\,\text{LSB}}\right)$$

- Example:
  - If 1 LSB = 0.5 V, then V_{IN} = 2.8 V will be converted to:

$$X = \text{round}\left(\frac{2.8}{0.5}\right) = \text{round}(5.6) = 6$$

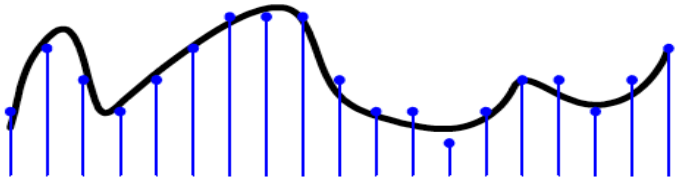
Analog to digital conversion **destroys information**: we convert a range of input voltages to a single digital value.

Let us consider the case where the converted voltage is unipolar. The analogue voltage is converted to a digital number to the nearest least significant bit (or nearest integer).

Therefore ADC operation looses information. This is known as **quantization noise**.

Sampling

- To process a continuous signal in a computer or other digital system, you must first sample it:



Time Quantisation

- Samples taken (almost always) at regular intervals: sample frequency of f_{samp}.
- This causes *aliasing*: A frequency of f is indistinguishable from frequencies $k\,f_{samp} \pm f$ for all integers k.
- No information lost if signal contains only frequencies below $\frac{1}{2}f_{samp}$. This is the *Nyquist limit*.

Amplitude Quantisation

- Amplitude of each sample can only take one of a finite number of different values.
- This adds quantisation noise: an irreversible corruption of the signal.
- For low amplitude signals it also adds distortion. This can be eliminated by adding dither before sampling.

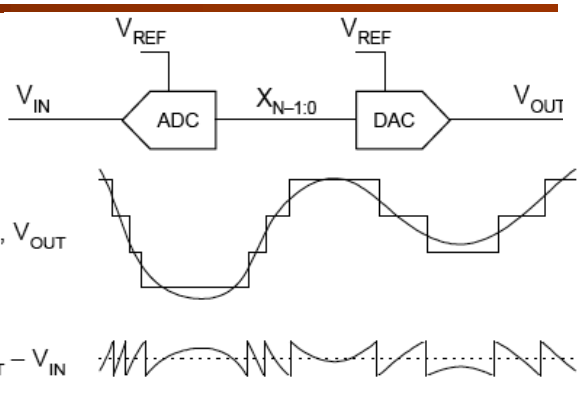
The idea of sampling is fully covered in the Signal and Linear Systems course. Essentially we quantize an analogue signal in time and in amplitude. Quantizing in time does not loose information as long as the sampling frequency is at least twice the maximum frequency component of the signal you are sampling.

Quantising in amplitude is achieved through a ADC and information is lost. The difference between the original analogue signal and the digitized signal is the quantisation noise.

Quantisation Noise

- *VOUT* is restricted to discrete levels so cannot follow VIN exactly. The error, *VOUT – VIN* is the quantisation noise and has an amplitude of ± ½ LSB.
- If all error values are equally likely, the RMS value of the quantisation noise is:

$$\sqrt{\int_{-½}^{+½} x^2 dx} = \frac{1}{\sqrt{12}} = 0.3 \text{ LSB}$$



Signal-to-Noise Ratio (SNR) for an n-bit converter

- Ratio of the maximum sine wave level to the noise level:
  - Maximum sine wave has an amplitude of $\pm 2^{n-1}$ which equals an RMS value of $0.71 \times 2^{n-1} = 0.35 \times 2^n$.
  - SNR is:

$$20\log_{10}\left(\frac{0.35\times 2^n}{0.3}\right) = 20\log_{10}(1.2\times 2^n) = 1.8 + 6n \text{ dB}$$

PYKC 26 Nov 2024 | EE2 Circuits and Systems | Lecture 12 Slide 5

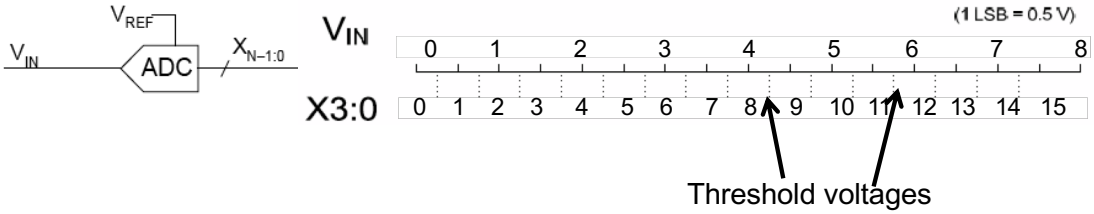
Take a signal V_IN, digitize this using a ADC to produce X[N-1:0] and then convert this back to analogue using a DAC to produce V_OUT. If you now subtract V_IN from V_OUT, you have the quantization oise. This noise signal has an amplitude of +/- ½ a LSB.

If you assume that the input signal is random and therefore the amplitude of the quantisation noise is equally likely to take on a value between - ½ LSB and + ½ LSB, the RMS value of the noise is easily shown to be around 0.3 LSB.

What is the Signal-to-Noise ratio of an n-bit converter? This can also be calculated easily. Consider a sine wave with an amplitude of +/- 2^(n-1). We choose this amplitude because this is centred around 0 (no dc component) and 1LSB = 1V, making this easier to express everything in LSB. The RMS value of this sine wave is easily shown to be 0.71 x 2^(n-1) or 0.35 x 2^n.

Therefore for such a sine wave, the SNR is 1.8 + 6n dB. In other words, for every extra bit of ADC/DAC resolution, we add an extra 6dB to the SNR.

Threshold Voltages

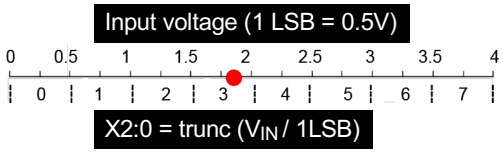


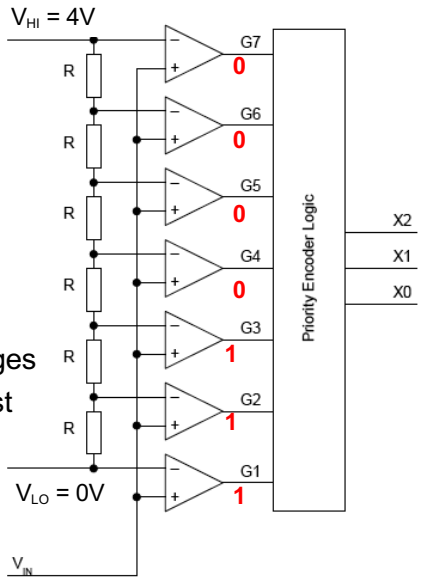
- Each value of X corresponds to a range of values of V_{IN}.
- The voltage at which V_{IN} switches from one value of X to the next is called a ***threshold voltage***.
- The task of an A/D converter is to discover which of the voltage ranges V_{IN} belongs to. To do this, the converter must compare V_{IN} with the threshold voltages.
- The threshold voltages corresponding to X are at (X±½) LSB

Every ADC contains a DAC converter, which provides many threshold voltages. The ADC compares the input voltage to be converter V_IN to these threshold voltages and determine what the converter digital value should be.

Each converter value X therefore corresponds to a range of values of V_IN. This range defines the threshold voltages which is X +/- ½ LSB.

Flash A/D Converter



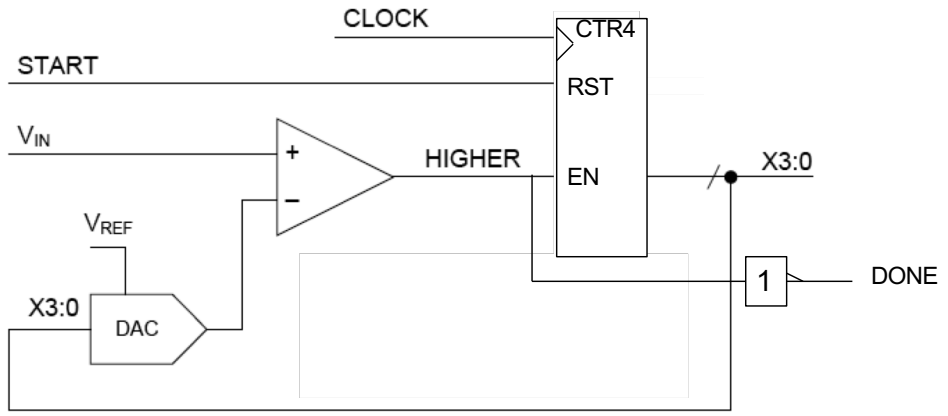


- For an n-bit converter we have 2^n-1 ***threshold voltages***
- Use 2^n-1 comparators:
- Resistor chain used to generate threshold voltages
- Priority encoder logic must determine the highest G_n input that equals 1.
- 12-bit converter needs 4095 comparators on a single chip!
- This example shows a unipolar converter

The simplest ADC is the flash ADC. We are converting from the range of 0V to 4V to a digital range of 0 to 7 in binary.

A voltage divider with a string of resistors R (which is the DAC circuit) is used to provide all the threshold voltages needed – i.e. 0, 0.5, … 3.5. 7 analogue comparators are used to determine which voltage interval V_IN lies. For example, if V_IN = 1.75V, then G1 to G3 are logic '1' and G4 to G7 are '0'. This produces the thermometer code which is decoded into a binary number X[2:0].

A Naïve ADC using a counter



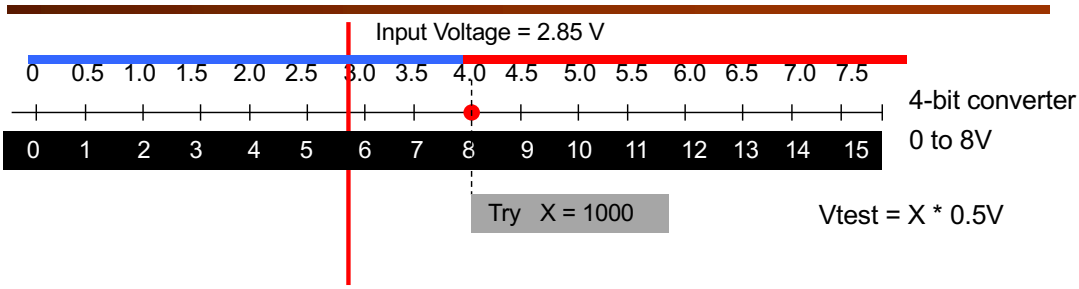
- Here is a simple ADC using a DAC, a comparator and a binary up counter.

This is a simple ADC using a DAC. The START signal is a short pulse that asynchronously reset the counter to zero. This starts the ADC conversion. If V_{IN} is above the lowest value from DAC, the counter is enable (HIGHER=1). The counter then counts up until V_{IN} is now lower than the DAC output, and counter is disabled, and the DONE signal goes high. X3:0 shows the value of the counter that makes the DAC just over the V_IN value.

The disadvantage of this converter is that the time it takes to perform a conversion is dependent on the value of V_{IN}. Furthermore, if this is a 16-bit converter, it could take over 65,000 clock cycles – therefore the conversion time can be very long.

We will next consider a different scheme using the successive approximation algorithm.

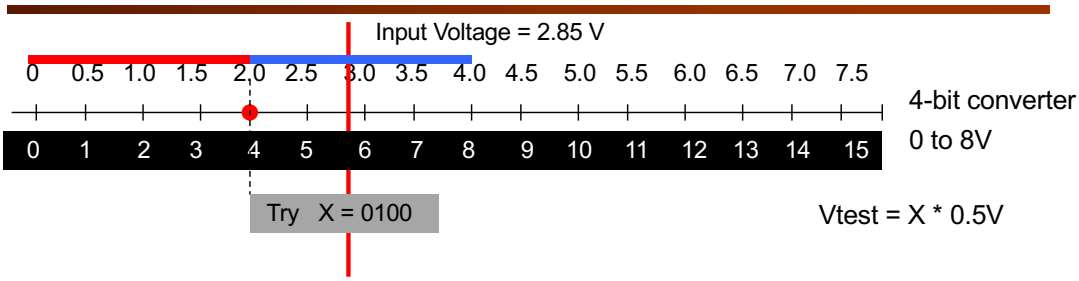
Successive Approximation ADC (1)



- First guess: set MSB to 1
- Set test voltage to 4V
- Is input ≥ 4V?
- If input ≥ 4V, X = 1???, keep MSB
  else X = 0???, clear MSB

Let us assume that the input voltage is 2.85V as shown as the RED line. We first set the DAC input X3:0 to 4'b1000 (i.e. assume MSB to be '1'), and compare V_IN to this threshold. You are effectively dividing the while voltage range into two halves: the lower BLUE range, and the upper RED range. V_IN belongs to the lower BLUE range, so we know setting MSB to '1' is too high. We therefore clear the MSB.

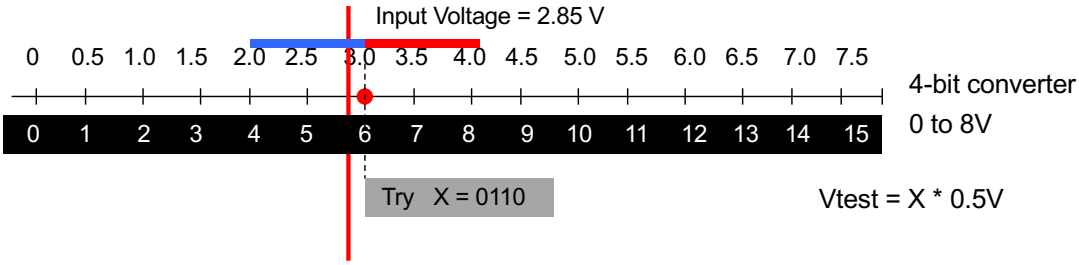
Successive Approximation ADC (2)



- Second guess: set next bit to 1 (X = 0100)
- Set test voltage to 2V
- Is input ≥ 2V?
- **If input ≥ 2V, X = 01??, keep bit as 1**
  else X = 00??, clear bit to 0

Next we divide the lower range from 0 to 4V into two equal halves again by setting X3:0 = 4'b0100. The threshold is now 2.0V. We are now testing the second most significant bit to see if this should be '1' or '0'. Now V_{IN} belongs to the upper half, therefore we keep the '1' bit which we tested for.

Successive Approximation ADC (3)

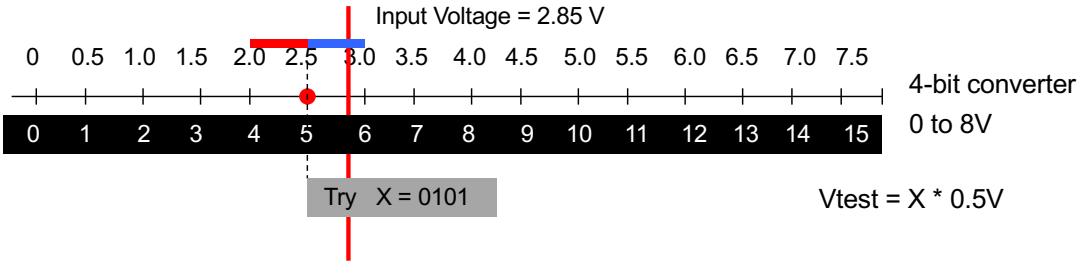


- Third guess: set next bit to 1 (X = 0110)
- Set test voltage to 3V
- Is input ≥ 3V?
- If input ≥ 3V, X = 011?, keep bit as 1

  else **X = 010?, clear bit to 0**

We now test the next bit by setting X3:0 = 4'b0110, testing X1. We are now dividing the range from 2.0V to 4.0V into two halves. V_{IN} belongs to the lower half, therefore we CLEAR X1.

Successive Approximation ADC (4)

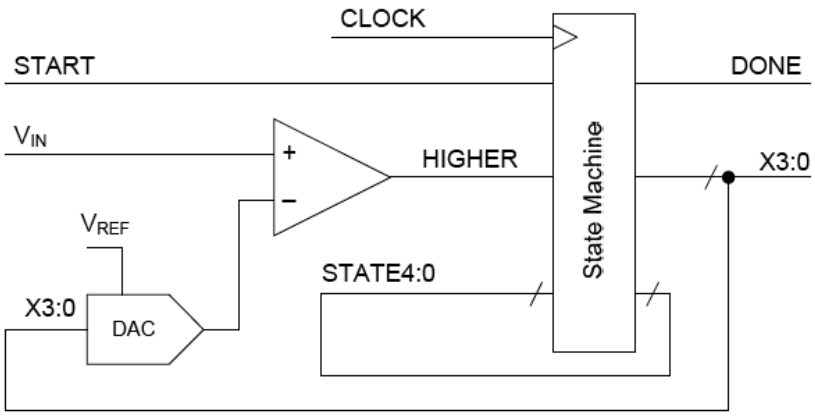


- Fourth guess: set next bit to 1 (X = 0101)
- Set test voltage to 2.5V
- Is input ≥ 2.5V?
- **If input ≥ 2.5V, X = 0101, keep bit as 1**
  else X = 0100, clear bit to 0

- Make successive guesses and use a comparator to tell whether your guess is too high or too low.
- Each guess determines one bit of the answer and cuts the number of remaining possibilities in half.

Finally we test for the last bit. V_{IN} is in the upper range, hence LSB is '1'. We have the final converted digital value: X3:0 = 4'b0101.

Successive Approximation ADC (5)

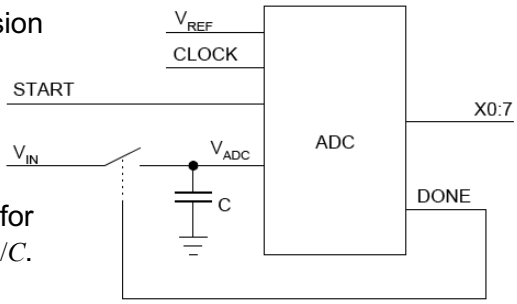


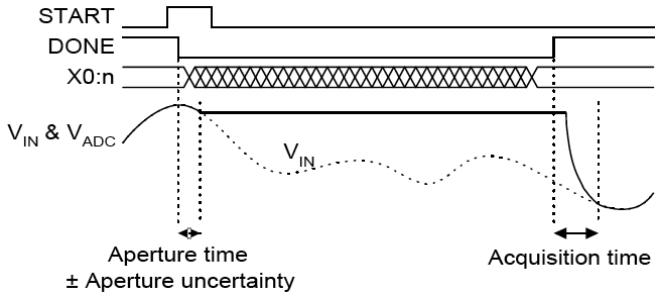
- Use a DAC to generate the threshold voltages and a state machine to create the sequence of guesses
- A DAC input of n generates the threshold between n–1 and n which equals (n–½) × 1 LSB

The hardware architecture for a SA-DAC is shown here. We need to design a FSM to implement the algorithm. It is similar to the counter based ADC we look at earlier, but the counter is now replaced by a state machine that makes decision on whether to reset the '1'-bit which was tested, and what the next DAC value to try.

A/D conversion with sample/hold

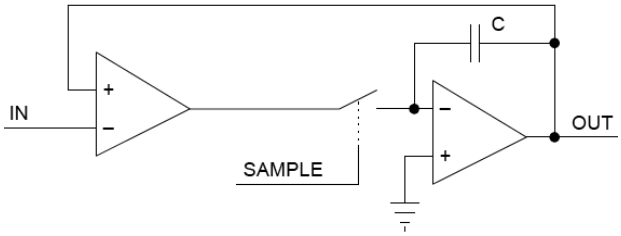
- Input switch is opened during the conversion so V_{ADC} remains constant (HOLD).
- Choice of C is a compromise:
  - Big C keeps constant voltage despite leakage currents since $dV/dt = I_{leakage}/C$
  - Small C allows faster acquisition time for any given input current since $dV/dt = I_{in}/C$.





So far we assume that while the ADC is performing conversion, the input signal is held at a fixed voltage level. If the input signal is in fact changing, the converted digital value will not be an accurate measure of VIN at the time of sampling. To ensure that the ADC input is held at a fixed voltage, we usually include follow-and-hold circuit. An analogue switch is normally turn ON, so that VIN is continuously charging the capacitor C. When the START pulse activates the ADC to take a sample, the DONE signal immediately goes low. This should open the switch and hold the VIN value at the time the conversion started.

Sample/Hold Circuit



- When switch is open:
  - Leakage currents through open switch and op-amp input will cause output voltage to drift up or down.
  - Choose capacitor large enough that this drift amounts to less than 0.5 LSB during the time for a conversion
  - Converters with high resolution or long conversion times need larger capacitors
- When switch closes:
  - Charge rate of capacitor is limited by the maximum op-amp output current. This determines the ***acquisition time***: to acquire the signal to within ½LSB. It is typically of the same order as the conversion time.
- Value of C is a compromise: big C gives slow acquisition, small C gives too much drift.

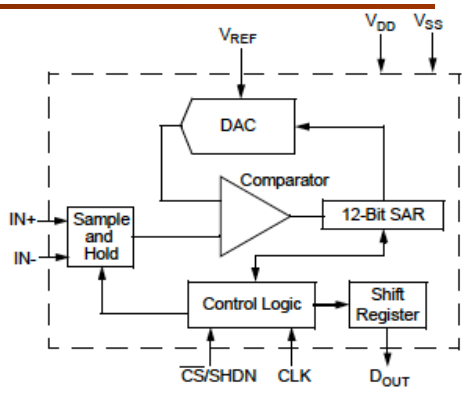
A practical sample/hold or follow/hold circuit is shown here using two operation amplifier. This has the advantage that the leakage current from the capacitor can be made very low. During the sampling or following mode, the right most op amp provides strong charging current to charge the capcitor.

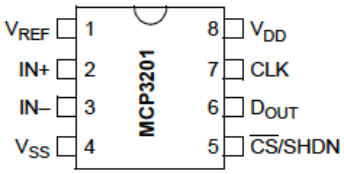
ADC used in Lab

- **Microchip MCP3201** 12-bit ADC
- Uses **successive approximation** architecture
- Serial Peripheral Interface (SPI)

Features

- 12-Bit Resolution
- ±1 LSB max DNL
- ±1 LSB max INL (MCP3201-B)
- ±2 LSB max INL (MCP3201-C)
- On-chip Sample and Hold
- SPI Serial Interface (modes 0,0 and 1,1)
- Single Supply Operation: 2.7V - 5.5V
- 100 ksps Maximum Sampling Rate at V_{DD} = 5V
- 50 ksps Maximum Sampling Rate at V_{DD} = 2.7V





This shows the ADC block diagram we use in the Lab. Again the digital interface obeys the SPI protocol, with Chip Select (CS), Serial Clock (CLK), and Serial Data Out (Dout) signals.

This ADC uses a 12-bit DAC internally, and the successive approximate algorithm (SAR) as described in our earlier lecture on ADCs. It produces a 12-bit output. However, the least significant 2 bits are not accurate. For our lab experiments, we only use the top 10-bits converted with this ADC chip.

Other types of Converter

Sampling ADC

- Many A/D converters include a sample/hold within them: these are ***sampling*** A/D converters.

Oversampling DAC and ADC

- ***Oversampling*** converters (also known as sigma-delta $\Sigma\Delta$ or delta-sigma $\Delta\Sigma$ converters) sample the input signal many times for each output sample. By combining digital averaging with an error feedback circuit they can obtain up to 20 bits of precision without requiring a high accuracy resistor network (hence cheaper). A typical oversampling ratio is 128X, i.e. the input is sampled at 6.4MHz to give output samples at 50 kHz. Most CD players use an oversampling DAC.

There is another class of converters known as oversampling converters. These use a sigma-delta modulator circuit which sample the input signal at a much high frequency than the Nyquist frequency demands. Normally it produces a 1-bit digital signal which is than down sampled and filtered to produce an accurate analogue output for a DAC, and a multi-bit digital value for a ADC. For example, CD players use an oversample DAC with sampling rate of 6.4MHz. This is then down sampled to produce an output sample rate of 50KHz – a oversampling ratio of 128 times.