

The ups and downs of arbitrary sample rate conversion

Ivar Løkken, 12/4-05.

In several applications, it is sometimes necessary to convert a signal from one sample rate to another. An input can be downsampled before processing to ease the computational load, or sometimes two units must be connected whose respective sample rates do not match.

If the desired sample-rate is an integer multiple of the existing one, it is sufficient to oversample the input using an interpolation filter. Likewise, if the existing sample-rate is an integer multiple of the one you want out, you can employ a decimation or undersampling.¹ However, if the ratio between input and output sample-rate is an arbitrary number, matters become more complicated. An arbitrary sample rate converter, or ASRC, must be designed. In such systems, the input and output signals are often derived from two separate clocks as well, in which case the conversion must be done using an asynchronous (arbitrary) sample rate converter (AASRC)².

In consumer digital audio players, sample rate conversion is also becoming more and more common, although the arguments for doing this are somewhat diffuse. One argument often used is that upsampling (sample rate conversion with a factor greater than one) improves the sound by easing the antialias filter requirements. Considering the fact that most DACs have at least 64 times oversampling already, the relevance of this is highly questionable. The advertisements often describe it as if upsampling magically improves the signal quality, while it in reality is a lossy process. Arbitrary sample rate conversion inevitably adds distortion, although in extremely small amounts if the SRC is well-designed.

Another and more realistic argument is that good jitter suppression can be achieved. If a well-designed AASRC is used together with a high-precision output clock, it can provide very good jitter performance on the output, even with a jittery source. The jitter performance of AASRCs will be dealt with later in this document.

Finally, most modern digital audio players can play back both CDs and DVDs, some also DVD-A. Thus the majority of the OEM-market wants circuit boards that can convert the 44,1kHz CD-signal into the 96kHz of DVD or 192kHz of DVD-A. PCBs with an AASRC and a 96 or 192kHz DAC have penetrated the audio market, and even most “pure” CD-players now use this solution.

This document will give an introduction to the principles of arbitrary sample rate conversion and how to model it. Performance issues as well as the AASRCs suitability as a “jitter-killer” will also be examined.

¹ Oversampling and undersampling can also in some literature be referred to as upsampling and downsampling. Here, we will use the former for a straight interpolator or decimator, and the latter when the sample-rate is changed by an arbitrary number through an SRC (upsampling if conversion ratio > 1 , downsampling otherwise).

² In most literature both arbitrary sample rate conversion and asynchronous sample rate conversion is abbreviated as ASRC. However, to avoid confusion, we will in this document use ASRC for the general Arbitrary SRC and AASRC in the special case of an Asynchronous (Arbitrary) SRC.

Sample rate conversion – the concept

Figure 1 shows the principle of sample rate conversion. In the ideal case, if the input is the time continuous signal $x(t)$ sampled at f_{si} , the output will be identical to $x(t)$ sampled at a different rate f_{so} .

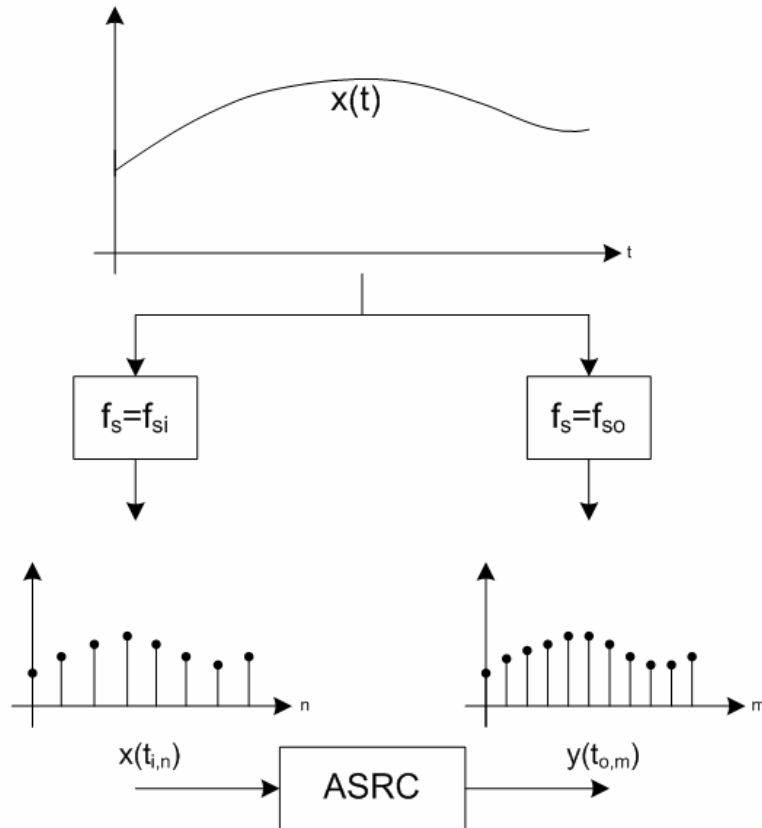


Figure 1: Principle of arbitrary sample rate conversion

How this is realised depends on the circumstances under which the conversion is to be done. If f_{so} is a fixed integer multiple of f_{si} , figure 1 simply can be realised using an interpolator or oversampler. If f_{si} is a fixed integer multiple of f_{so} , we can use a decimator or undersampler. If there is a fixed ratio between f_{si} and f_{so} , a fractional SRC consisting of an oversampling filter and a consecutive undersampler can be used. If the sampling rates are derived from different clocks, the conversion ratio will be arbitrary or even slowly time-varying and we have to use an AASRC.

A short review of oversampling and undersampling

Figure 2 shows the principle of over- and undersampling. The oversampling factor L in the illustration at the left is 2, while for the undersampling-example rightmost $M=2$. The basic principle of interpolation and decimation can be used for integer L or M .

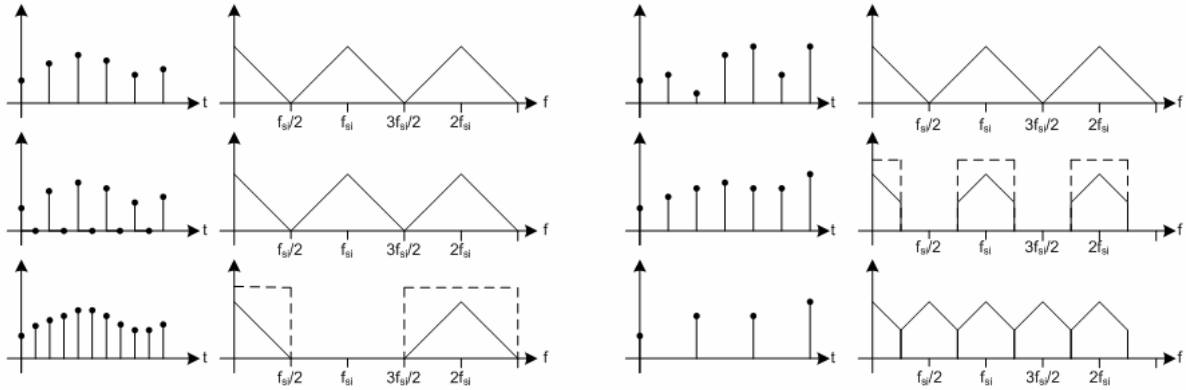


Figure 2: The basic principle of oversampling (left) and undersampling (right).

The figure shows the processes both in the time-domain and frequency-domain. When doing oversampling, the signal is first converted to the new sample-rate by inserting $L-1$ zero samples between the existing ones. This is called zero-padding. Then, the signal is low-pass filtered to remove the aliases within the new baseband. This is done through interpolation, as can be seen in the time-representation. An ideal interpolation-filter has a brickwall-characteristic, given by:

$$H_L(f) = \begin{cases} 1 & , 0 < f < \frac{f_{si}}{2} \\ 0 & , \frac{f_{si}}{2} < f < \frac{L f_{si}}{2} \end{cases} \quad \text{Eq. 1}$$

The entire oversampling process is usually realized using an interpolation filter.

When undersampling, the signal must be low-pass filtered first. This is to prevent signal-components above the new, lower nyquist-frequency to create aliases in the new baseband. Then the sample-train is decimated; every M 'th sample is removed for a undersampling factor of M . The process is usually realized through a decimation filter. An ideal decimation-filter has a frequency response given by:

$$H_M(f) = \begin{cases} 1 & , 0 < f < \frac{f_{si}}{M \cdot 2} \\ 0 & , \frac{f_{si}}{M \cdot 2} < f < \frac{f_{si}}{2} \end{cases} \quad \text{Eq. 2}$$

If we want to do a sample-rate conversion with a fixed ratio L/M , we first interpolate with L and then decimate with M . The two lowpass-filters are then cascaded and can be combined into one, $H_{LM}(f) = H_L(f) \cdot H_M(f)$. It is apparent from the equations above that this will be a lowpass-filter with cutoff-frequency given by the lowest of either $f_{so}/2$ or $f_{si}/2$.

If we look at the interpolation in the time-domain we know the multiplication of the signal and transfer-function in the frequency-domain will correspond to a convolution between the input signal and the impulse-response in the time-domain. In other words:

$$y[m] = x[m] * h_L[m] = \sum_{k=-\infty}^{\infty} h_L[m-k] \cdot x[k] \quad \text{Eq. 3}$$

Since the input to the filter x is zero-padded, with values different from zero only every L 'th sample, we can rewrite the equation as:

$$y[m] = \sum_{k=-\infty}^{\infty} h_L[m-k] \cdot x\left[\frac{k}{L}\right] \quad , k = \pm L, \pm 2L, \pm 3L, \dots \quad \text{Eq. 4}^3$$

To get a sum valid for all indexes, we can substitute k by $(m \operatorname{div} L) \cdot L - n \cdot L$ and get:

$$y[m] = \sum_{n=-\infty}^{\infty} h_L[n \cdot L - m \operatorname{mod} L] \cdot x[m \operatorname{div} L - n] \quad \text{Eq. 5}$$

In the undersampler, we convolve $x[n]$ with the filter $h_M[n]$ and then keep only every M 'th sample. We can pick the first sample for any instant from $n=0$ to $n=M-1$ (initial phase), so:

$$y[m] = \sum_{n=-\infty}^{\infty} h_M[m \cdot M - n + \varphi_0] \cdot x[n] \quad , \varphi_0 = 0, 1, 2, \dots, M-1 \quad \text{Eq. 6}$$

We can from this see that there is M different sets of samples that all represent the same signal, depending on the initial phase.

This is also the case of doing a fixed sample-rate conversion L/M . When decimating we can start at any oversampled value we want and pick out every M 'th value. Thus, in the interpolation-filter from eq.5, m must be replaced with $m \cdot M + \varphi_0$. Doing this substitution we get:

$$y[m] = \sum_{n=-\infty}^{\infty} h_{LM}[n \cdot L + (m \cdot M + \varphi_0) \operatorname{mod} L] \cdot x[(m \cdot M + \varphi_0) \operatorname{div} L - n] \quad \text{Eq. 7}$$

If we do an IFFT of the ideal low-pass filter given, we find the well-known sinc-function. Thus we can give a complete mathematical description of oversampling, undersampling and fixed sample-rate conversion. This is tabulated in table 1.

³ Equation 4 can be written as $y[m] = x[n] \circledast h[m]$ where \circledast is the special convolution operator indicating that the convolved signals have different sampling rates. The zero-padding is the "included" in the convolution. For eq. 4 it is given that $m=n \cdot L$. The convolution is still linear, commutative and associative. See [2] for details.

Table 1: Mathematical description of ideal oversampling, undersampling and fixed SRC

Oversampling	
$y[m] = \sum_{n=-\infty}^{\infty} h_L[n \cdot L - m \bmod L] \cdot x[m \operatorname{div} L - n]$	$h_L[n] = \operatorname{sinc} \left[n \frac{1}{L} \right]$ <p>running at Lf_i</p> $H_L(f) = \operatorname{rect} \left(\frac{f_o}{L \cdot 2} \right)$
Undersampling	
$y[m] = \sum_{n=-\infty}^{\infty} h_M[m \cdot M - n + \varphi_0] \cdot x[n] \quad , \varphi_0 = 0, 1, 2 \dots M - 1$	$h_M[n] = \operatorname{sinc} \left[n \frac{1}{M} \right]$ <p>running at f_i</p> $H_L(f) = \operatorname{rect} \left(\frac{f_i}{M \cdot 2} \right)$
Fractional sample rate conversion	
$y[m] = \sum_{n=-\infty}^{\infty} h_{LM}[n \cdot L + (m \cdot M + \varphi_0) \bmod L] \cdot x[(m \cdot M + \varphi_0) \operatorname{div} L - n]$	$h_{LM}[n] = \operatorname{sinc} \left[n \frac{1}{\max(L, M)} \right]$ <p>running at Lf_i</p> $H_{LM}(f) = \operatorname{rect} \left(\min \left[\frac{f_i}{2}, \frac{f_o}{2} \right] \right)$ <p>, $f_o = f_i \frac{L}{M}$</p>

Since we now know the ideal expressions for fixed sample-rate conversion, we can also calculate the errors it introduces. This will be addressed later.

The arbitrary sample-rate converter

The general approach to AASRC [1] is to oversample the signal heavily to a very dense intermediate signal. This is converted to continuous-time using a hold-circuit. Then this signal is sampled at the new rate f_{so} . In a digital realization this is usually partitioned into two units; a high-order interpolation and/or decimation filter, which limits the signal bandwidth to the lowest of the source and sink frequency, and a frequency tracking unit, which determines the sink phase relative to the source samples. The basic model is shown in figure 2.

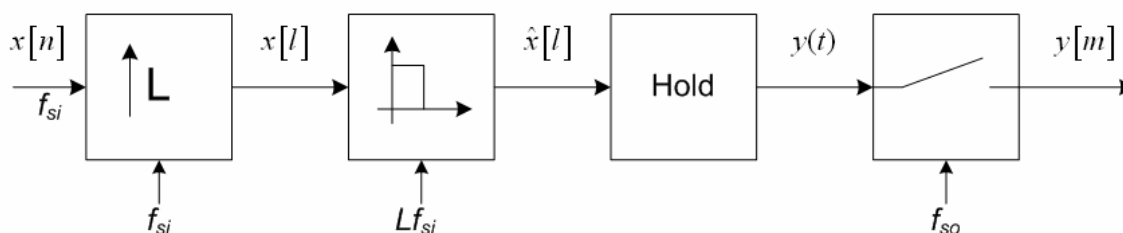


Figure 3: Basic model of arbitrary sample rate conversion.

Figure 3 shows an example of the output $y[m]$. As we can see, the hold operation of the value of the interpolated sample within a time slot T is a convolution between the interpolated samples and a rectangle of width T . This can be expressed as:

$$y(t) = \hat{x}[l] * \text{rect}\left(\frac{1}{L \cdot f_{si}}\right)$$

$$Y(f) = \hat{X}(f) \cdot \text{sinc}\left(\frac{f}{L \cdot f_{si}}\right)$$
Eq. 8

The intermediate signal $y(t)$ can then be resampled at any given time dictated by the output clock f_{so} .

As we can see even from this simple model, there are several error sources we do have to deal with in order to estimate and achieve the desired performance. As mentioned earlier, the interpolation filter will not be a perfect brick-wall, but introduce some error. In addition, we can see from figure 3 that the hold-operation with its sinc frequency response will also introduce error. Fundamental requirements for oversampling ratio and filter order will need to be determined, as well as the effect of limited coefficient resolution and other non-idealities.

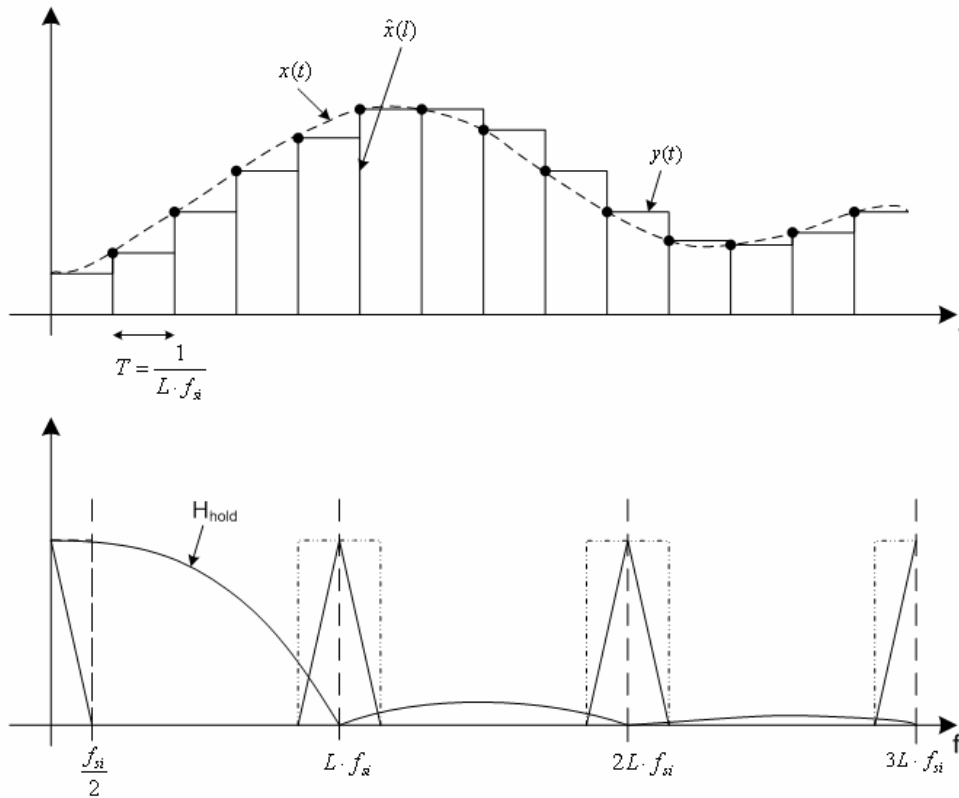


Figure 4: Interpolation and hold, time and frequency view.

In the next section, we will take a look at the separate operations and the requirements for these. Later, we will take a look at the practical implementation of the resampling process itself.

Requirements for the AASRC

In this section, we will take a look at some fundamental performance limitations in the AASRC and try to derive some fundamental requirements for key parameters.

Oversampling ratio L

First we will try to find the necessary intermediate oversampling ratio L. If we look at figure 4, it is apparent that the hold-operation introduces an error around multiples of $L \cdot f_{si}$ that will add and fold into the new baseband ($L \cdot f_{si}/2$).

We will look at the error introduced with a sinusoidal input. In [1] it is shown that a sine-wave close to $f_{si}/2$ produces the most severe error. Looking at the frequency view in figure 4, we can see that for an input with frequency f_{sig} , the error will be introduced at $k \cdot L \cdot f_{si} \pm f_{sig}$ and then fold down since the sinc-function does not have infinite damping at these frequencies. With the maximal input normalized to unity, the error for a full-scale sinewave input will be:

$$E_{hold} = \sum_{k=1}^{\infty} \left[\text{sinc} \left(\frac{k \cdot L \cdot f_{si} \pm f_{sig}}{f_{si}} \right) \right]^2 = \sum_{k=1}^{\infty} \left(\frac{\sin \pi \frac{k \cdot L \cdot f_{si} \pm f_{sig}}{f_{si}}}{\pi \frac{k \cdot L \cdot f_{si} \pm f_{sig}}{f_{si}}} \right)^2 \quad \text{Eq. 9}$$

If $f_{sig} \ll f_{si}$, this simplifies to:

$$E_{hold} \approx \sum_{k=1}^{\infty} 2 \cdot \left(\frac{\pi \frac{f_{sig}}{L \cdot f_{si}}}{\pi \cdot k} \right)^2 = 2 \cdot \left(\frac{f_{sig}}{L \cdot f_{si}} \right)^2 \cdot \sum_{k=1}^{\infty} \frac{1}{k^2} \approx \frac{\pi^2}{3} \cdot \left(\frac{f_{sig}}{L \cdot f_{si}} \right)^2 \quad \text{Eq. 10}$$

If we insert typical numbers for CD audio, input sampling frequency f_{si} at 44.1kHz and a worst-case f_{sig} of 20kHz, and solve for an assumed maximum allowable error of -120dB, we get $L=464094$, or $L=2^{19}$ as the lowest useable interpolation factor.

An oversampling filter with that high L is difficult to implement. To achieve the performance required above, but with less oversampling, the hold-circuit must be replaced with an interpolator. The simplest interpolator is a linear interpolator, drawing a straight line between consecutive samples. Linear interpolation has a sinc^2 frequency-response and we can calculate the error using the same approach as for the hold-circuit:

$$E_{int} = \sum_{k=1}^{\infty} \left[\text{sinc}^2 \left(\frac{k \cdot L \cdot f_{si} \pm f_{sig}}{f_{si}} \right) \right]^2 = \sum_{k=1}^{\infty} \left(\frac{\sin \pi \frac{k \cdot L \cdot f_{si} \pm f_{sig}}{f_{si}}}{\pi \frac{k \cdot L \cdot f_{si} \pm f_{sig}}{f_{si}}} \right)^4 \quad \text{Eq. 11}$$

$$E_{int} \approx \sum_{k=1}^{\infty} 2 \cdot \left(\frac{\pi \frac{f_{sig}}{L \cdot f_{si}}}{\pi \cdot k} \right)^4 = 2 \cdot \left(\frac{f_{sig}}{L \cdot f_{si}} \right)^4 \cdot \sum_{k=1}^{\infty} \frac{1}{k^4} \approx \frac{\pi^4}{45} \cdot \left(\frac{f_{sig}}{L \cdot f_{si}} \right)^4$$

If we now insert the same number for input sampling frequency and signal frequency, we get approximately -120dB error at $L=2^9$ or 512. At $L=2048$ the introduced error is at -143dB. If even higher performance (or lower oversampling ratio) is required, many higher-order interpolation methods can be used. Higher-order Lagrange interpolation is an obvious approach, but there are many other novel algorithms as well, each with different properties and complexity requirements. The reader can read [3], [4], [5], [6] to get some overview of used approaches, they will not be treated in detail in this introductory document.

The oversampling filter

We now addressed the necessity of oversampling and gotten an impression of how high the ratio L has to be. In the ideal case, the oversampling filter is a perfect brickwall, in real life this is of course not the case. It has finite slopes and finite stopband attenuation. The non-ideal filter can be modelled with a few key characteristics, which are shown in figure 5.

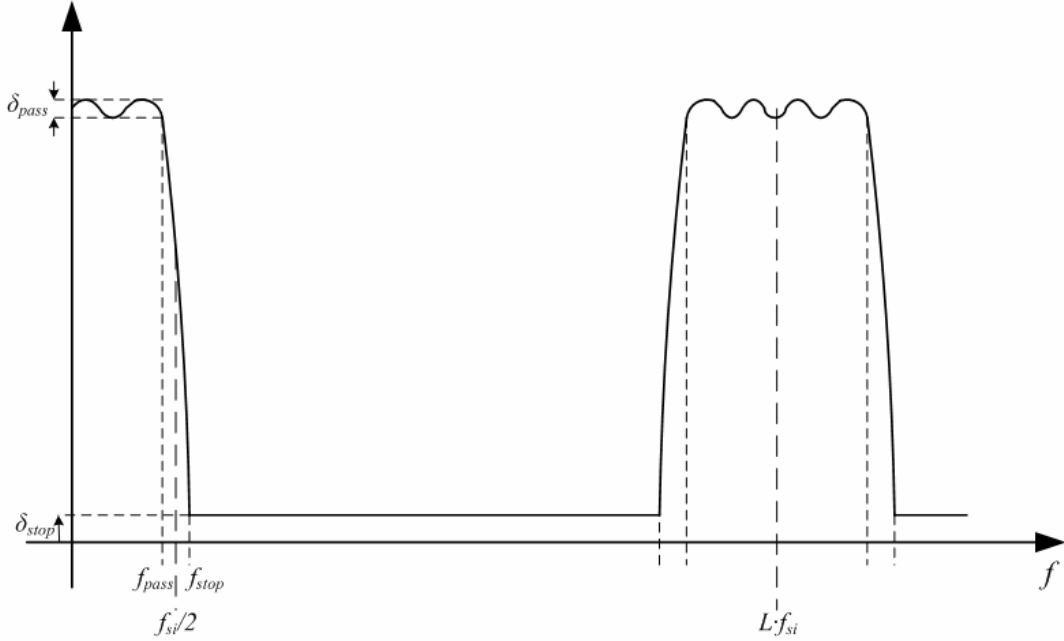


Figure 5: Non-ideal filter characteristics

The filter is characterized by its passband ripple δ_{pass} , the maximum deviation from the norminal (unity) gain within the passband and its finite stopband attenuation δ_{stop} . The upper limit of passband is defined as f_{pass} , while the stopband starts at f_{stop} . In high-quality audio, f_{pass} is usually defined as $(15/32) \cdot f_{si}$ (ca 20kHz for CD) while $f_{stop} = (15/32) \cdot f_{si}$ [1].

Due to the hold (or interpolation) after the oversampling filter, the stopband function will be weighted with the sinc (or sinc² for linear interpolation) response shown in figure 4. Thus, in addition to the error around multiples of $L \cdot f_{si}$ (which we assume not affected by the filter, since the signal is within the baseband), we will also get an unwanted residual in the area between f_{si} and $L \cdot f_{si}$ and multiples thereof, that will also fold into the baseband. For high L, we can assume δ_{stop} to be constant as shown in figure 5 and the signal energy evenly distributed across the stopband. Then we can express the total stopband energy as:

$$E_{stop} = 2 \sum_{k=1}^{\infty} \int_{k \cdot L \cdot f_{si} + f_{si}/2}^{(k+1) \cdot L \cdot f_{si} - f_{si}/2} \frac{\delta_{stop}^2}{f_{si}} \cdot \text{sinc}\left(\frac{f}{L \cdot f_{si}}\right) df \quad \text{Eq. 12}$$

The sinc is for hold-operation. If another interpolation method is used, the frequency response of this should be used instead. If we assume a large L, we can approximate to:

$$E_{stop} \approx 2 \cdot \int_0^{\infty} \frac{\delta_{stop}^2}{f_{si}} \cdot \text{sinc}\left(\frac{f}{L \cdot f_{si}}\right) df = L \cdot \delta_{stop}^2 \quad \text{Eq. 13}$$

When we have the requirement for filter stopband attenuation, we can also find the necessary filter order. This of course depends on the filter topology chosen, if it is IIR or FIR, if it is Butterworth, Chebyshev etc. It must be solved for the given topology.

It must also be noted that if $f_{so} < f_{si}$, the cutoff frequency must be given by $f_{so}/2$, according to table 1. In a real implementation, the filter is adjusted dynamically.

Hold on a moment:

We have now found equations for the errors introduced by the oversampling and the consequent interpolation to continuous time. However, in a fully digital implementation, we cannot convert the signal to real continuous time. The interpolator must calculate a number of samples between two existing (oversampled) ones and the one closest to the sampling instant of the output clock must at any time be chosen. In the case of the second interpolator being a simple hold circuit, we can just pick samples from the output of the oversampler. However, we have shown that this will not give sufficiently good performance until $L=2^{19}$ or more.

When using a second interpolator, we insert a number of samples between the existing ones and pick one of these instead. But, as the observed reader may have deduced by now, *this is equivalent to inserting a second upsampling and hold circuit*. We insert samples, thus interpolate, and from a continuous time perspective have a constant value between the inserted samples, thus hold. In other words, we have just partitioned the oversampling as shown in figure 6.

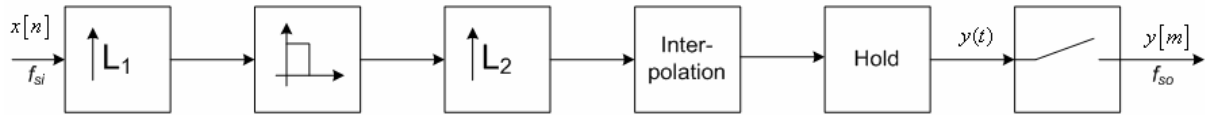


Figure 6: Partitioning of the AASRC

The question is if this partitioning is beneficial. We now have two cases:

- a) A very high L oversampler with simple hold circuit behind:

$$E_{out} = E_{hold} + E_{stop} = \frac{\pi^2}{3} \left(\frac{f_{sig}}{L \cdot f_{si}} \right)^2 + L \cdot \delta_{stop}^2 \quad Eq. 14$$

- b) Or a lower L oversampler L_1 followed by an interpolator and a hold. If we assume the interpolator to be a linear interpolator which inserts L_2 samples, we get:

$$E_{out} = E_{int} + E_{stop} + E_{hold} = \frac{\pi^4}{45} \left(\frac{f_{sig}}{L_1 \cdot f_{si}} \right)^4 + L_1 \cdot \delta_{stop}^2 + \frac{\pi^2}{3} \left(\frac{f_{sig}}{L_1 \cdot L_2 \cdot f_{si}} \right)^2 \quad Eq. 15$$

We can do an example using the two alternatives based on a realistic specification to illustrate how they will differ. If we assume $f_{si}=44.1\text{kHz}$ and require each individual error contribution to be less than -130dB for a 1kHz input signal (design for a little bit less than 130dB), we get:

- a) High L oversampler (FIR Chebyshev) with hold:
- i. Solving eq. 10 with regard to L as done before, we find: $L = \underline{2^{17}}$
 - ii. $\delta_{stop} = 20 \cdot \log_{10}[(E/L)^{1/2}] = \underline{-181\text{dB}}$
 1. If we assume a Chebyshev FIR interpolator and insert 0.001dB allowable passband ripple, $f_{pass} = f_{si} \cdot 15/32$ and $f_{stop} = f_{si} \cdot 17/32$ we get a filter order $N \approx \underline{10e6}$
 - iii. Find total error with eq. 14, $E_{out} = \underline{-128,7\text{dB}}$.

- b) Oversampler with L_1 (FIR Chebyshev) followed by (linear) interpolator:
- i. Solve eq. 11 with regard to L_1 , we find: $L = 49$. We choose $L_1 = \underline{512}$ though as this is a more realistic value (otherwise L_2 will be way too large).
 - ii. $\delta_{\text{stop}} = 20 \cdot \log_{10}[(E/L)^{1/2}] = \underline{-157\text{dB}}$
 1. Under the same assumptions as above, we get $N \approx \underline{1e4}$.
 - iii. Solve L_2 from last part of eq. 15: $L_2 = \underline{256}$.
 - iv. Find total error from eq. 14: $E_{\text{out}} = \underline{-127.0\text{dB}}$

As we can see, there is great savings to be done for the FIR filter. Also, the first approach required it to run at $44.1\text{kHz} \cdot 2^{17} = 5.8\text{GHz}$ which is clearly not feasible. The second approach allows us to run the FIR-filter at a much lower frequency and also use a filter with much lower order. However, the filter order is still in the range of 100.000, which is very large. Also, the computational load from the linear interpolator will be significant. It will have to operate at a sampling frequency of $44.1\text{kHz} \cdot 512 = 22.6\text{MHz}$ and at that rate calculate and store 256 intermediate values that make up the quasi-continuous time signal $y(t)$. The right one must also be chosen for the output $y[m]$ depending on the instantaneous output sample frequency f_{so} , or rather the ratio between f_{so} and f_{si} . This is done through a frequency tracking unit, which is the next part we'll look at.

It must also be added that in addition to the intrinsic distortion mechanisms examined above, there will also be implementation-related non-idealities that have to be considered. First and foremost the resolution of the filter and interpolator coefficients and the intermediate signals. How much limited wordlength compromises performance is however highly dependable on the filter topology and implementation. It will thus not be treated in this document which is intended to give insight on a more general level.

The resampling core of an asynchronous sample rate converter has now been examined through a generic model and its fundamental sources of noise or distortion been derived. It has been shown that for very high resolution, the circuit inevitably becomes very complex.

The frequency tracking unit

We recall equation 7 for fractional sample rate conversion L/M :

$$y[m] = \sum_{n=-\infty}^{\infty} h_{LM}[n \cdot L + (m \cdot M + \varphi_0) \bmod L] \cdot x[(m \cdot M + \varphi_0) \text{div} L - n] \quad \text{Eq. 16}$$

In the case of the AASRC we have, as shown above, a decimation factor that depends on the phase of the sink clock (the sink sampling time relative to the source samples) and is thus variable. In other words:

$$(m \cdot M + \varphi_0) \bmod L = \Delta\varphi_{so}[m] \quad \text{Eq. 17}$$

This means that to be able to compute the correct sink value $y[m]$, the precise sink phase $\Delta\varphi_{so}[m]$ must be known for every m . This is intuitive by viewing the decimation as picking the correct sample depending on the output frequency. If we pick the wrong samples, we will in practice have done a non-uniform resampling and added distortion.

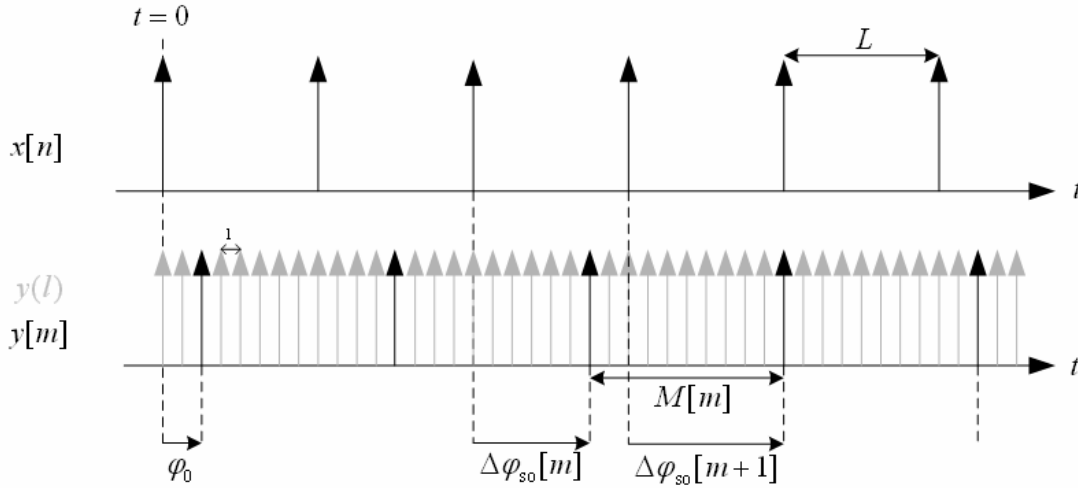


Figure 7: Illustration of the source and sink phase relationship

If the total oversampling factor is L , $\Delta\varphi_{so}[m]$ must be computed to at least $\log_2(L)$ bit precision. A way to do this digitally could be to implement a high-speed counter that measures the distance between source and sink sampling instants, and outputs an address to the location of the corresponding interpolator output sample. However, this counter would have to operate at $L \cdot f_{si}$, already proven to end up in the GHz range. Also, any jitter error in the sampling phase (in other words jitter on either sampling clock) would directly propagate to an error in $\Delta\varphi_{so}[m]$, and thus non-uniform resampling would occur. There would be no jitter attenuation.

However, there is no need to measure each sink sample moment directly. The subsequent sink sample phases $\Delta\varphi_{so}[m]$ can be computed from the previous ones by adding the precisely measured ratio of source and sink sample rate, f_{si}/f_{so} . This ratio in general varies slowly in time, and by doing a long-term averaging, jitter can be suppressed. If we define the average decimation factor $\bar{M} = L \cdot (f_{si}/f_{so})$ and $M[m]$ the ratio at sample instant m , we can use:

$$\Delta\varphi_{so}[m+1] = (\Delta\varphi_{so}[m] + M[m]) \bmod L \quad \text{Eq. 18}$$

To find the sample frequency ratio $M[m]$, we use a *frequency tracking unit*. This is a digital circuit that calculates the ratio for each sample instant. As we will shortly see, the design of the frequency tracking unit decides to what degree jitter in the sampling clocks will affect $M[m]$ and thus, through eq. 18, $\Delta\varphi_{so}$. The frequency tracking unit is thus critical for the desired jitter-rejection.

There are generally two ways to implement the frequency tracking unit digitally. The first is to use a frequency counter. As mentioned the ratio must be found with a precision of at least $\log_2(L)$ bits. One can use a PLL to multiply the source sample frequency to $2^{\log_2(L)-k}$ and use this to measure the sink rate. Averaging k consecutive measurements the required accuracy is obtained and the averaging will suppress, or low-pass filter, incoming jitter. If a $2^{\log_2(L)-k} \cdot f_{si}$ clock is already available, the PLL can be omitted. For instance in SP-dif, a bitclock at $64 \cdot f_{si}$ is available. If $L=512 \cdot 256$, we get $2^{17-k}=64$, or $k=2048$. In other words we can use the bitclock to measure at 6-bit accuracy and then average 2048 consecutive samples to bring it up to the required 17. Then the system will look like figure 8.

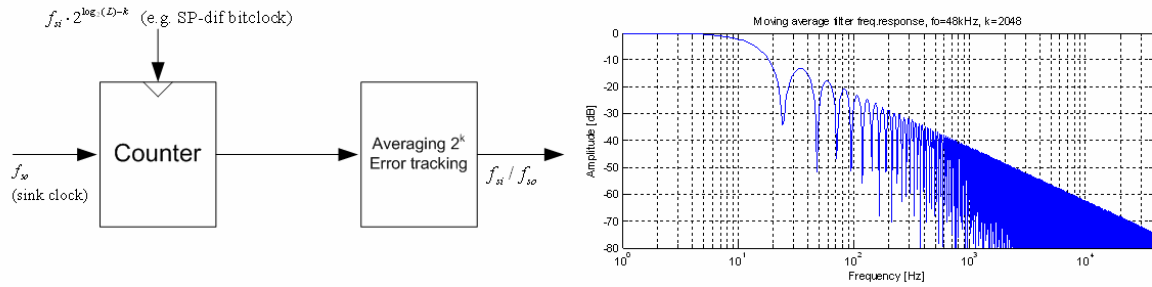


Figure 8: Frequency tracking using frequency counter. With $f_{so}=48\text{kHz}$, $k=2048$

Figure 8 shows the system and the sinc frequency response of a 2048 sample moving average filter running at 48kHz. As we can see, jitter suppression starts at around 10Hz (the first zero is at $48\text{kHz}/2048=23\text{Hz}$) and attenuates at 20dB/dec. This is very good jitter performance.

The second method is to use a digital PLL (DPLL). A DPLL is equivalent to an ordinary PLL; a loop that forces the output to match the phase of the input. It consists of a phase-detector that generates a signal proportionate to the difference between ϕ_{si} and ϕ_{so} , a loop-filter that attenuates short-term jitter in the phase-detector output and a VCO that tracks the difference. In a DPLL, the VCO is not actually a such, but a digital-domain integrator, realized as a accumulator.

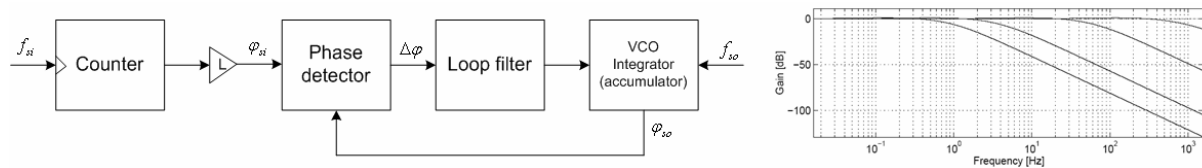


Figure 9: Digital PLL (DPLL) and example of closed-loop response from [1]

We will not go into detail about the implementation of the DPLL, this can be found in [1]. The design approach is however much like the design of a regular PLL. The phase detector can be implemented using a gray-counter run at f_{si} and a register that latches its output value controlled by f_{so} , then ϕ_{si} relative to the output clock is found. The loop-filter is designed as a digital filter of relatively low order. As with a regular PLL, there is a trade-off between the cut-off frequency, which we want low, and the lock-on time, which we want small. The digital integrator, or “VCO”, is as mentioned realized as an accumulator.

As with a regular PLL, the jitter attenuation is determined by the closed-loop transfer function. Since this is a completely digital circuit, the flexibility in this regard is much greater than for a an analog PLL. In [1] a design is shown, which in its narrowest setting attenuates jitter by 70dB at as low as 50Hz, with jitter attenuation starting at below 1Hz.

Both methods mentioned here are used in AASRCs. Studies in [1] indicate that the DPLL solution is significantly easier to implement with the very high precision required for this purpose. It is also a more efficient design as the high-order moving average filter of the frequency counter will be a rather large and complex component.

AASRCs and jitter

As we have stated before, errors in calculating the sample-to-sample sink phase $\Delta\varphi_{so}[m]$ will lead to non-uniform resampling, since the wrong interpolated samples are selected. Thus, deviations in $\Delta\varphi_{so}[m]$ is more or less equivalent to sampling jitter in a DAC. We have also seen that the propagation of jitter from the sampling clocks f_{si} and f_{so} to errors in the calculated ratio $M[m]$, and consequently $\Delta\varphi_{so}[m]$, is dependent on the transfer function of the frequency tracking unit. Very good jitter suppression has been shown.

However, some jitter gets through. To see how it affects the output, we define the continuously running sink phase as $\varphi_{so}[m] = \sum M[m]$ (then, $\Delta\varphi_{so}[m] = (\varphi_0 + \varphi_{so}[m]) \bmod L$ as seen by eq. 17). Then we can express the continuously running output signal $y[m]$ as:

$$y[m] = A \cdot \sin\left(2\pi f_{sig} \frac{\varphi_{so}[m]}{L \cdot f_{si}}\right) \quad \text{Eq. 19}$$

If we divide the calculated sink phase $\varphi_{so}[m]$ into an ideal part $\varphi_{ideal}[m]$ (the actual, non-calculated sink phase) and an error $\varphi_e[m]$, we get:

$$y[m] = A \cdot \sin\left(2\pi f_{sig} \frac{\varphi_{ideal}[m] + \varphi_e[m]}{L \cdot f_{si}}\right) \quad \text{Eq. 20}$$

$$y[m] \approx A \cdot \sin\left(2\pi f_{sig} \frac{\varphi_{ideal}[m]}{L \cdot f_{si}}\right) + \frac{A \cdot 2\pi f_{sig} \cdot \varphi_e[m]}{L \cdot f_{si}} \cos\left(2\pi f_{sig} \frac{\varphi_{ideal}[m]}{L \cdot f_{si}}\right)$$

This approximation is valid for large L . As we can see, we get an ideal signal plus an error term that is proportional to the phase error $\varphi_e[m]$, the signal frequency and the signal amplitude. Note that since multiplication in the time-domain equals convolution in the frequency domain, the spectrum of $\varphi_e[m]$ is shifted to the signal frequency. This means that a sinusoidal phase deviation will produce sidebands, just like sampling jitter in a DAC.

If we assume an ideal phase $\varphi_{\text{ideal}}[m] = m \cdot L \cdot f_{si} / f_{so}$ and sinusoidal phase error with frequency f_e we get:

$$y_e[m] \approx \frac{A \cdot 2\pi f_{sig}}{L \cdot f_{si}} \cdot \hat{\varphi}_e \cos\left(\frac{m \cdot 2\pi}{f_{so}} f_e\right) \cdot \cos\left(\frac{m \cdot 2\pi}{f_{so}} f_{sig}\right) \quad \text{Eq. 21}$$

$$y_e[m] \approx \frac{A \cdot 2\pi f_{sig} \cdot \hat{\varphi}_e}{2 \cdot L \cdot f_{si}} \left[\cos\left(\frac{m \cdot 2\pi}{f_{so}} (f_{sig} - f_e)\right) + \cos\left(\frac{m \cdot 2\pi}{f_{so}} (f_{sig} + f_e)\right) \right]$$

If we use the method in [8] to find the expression for the jitter spuriae in a DAC, with a sampling jitter $J(t) = \hat{J} \cdot \sin(2\pi f_{jit} t)$, we get:

$$y_e(t) \approx \frac{A \cdot 2\pi \cdot f_{sig} \cdot \hat{J}}{2} \sin\left(2\pi (f_{sig} - f_{jit}) t\right) \cdot \cos\left(2\pi (f_{sig} + f_{jit}) t\right) \quad \text{Eq. 22}$$

The equivalence between non-uniform resampling due to deviations in the sink phase and sampling jitter in a DAC should thus be obvious. And just like sampling jitter in a DAC depends on the frequency response of the analog clock recovery circuitry, phase deviations in an AASRC depend on the frequency response of the frequency tracking unit.

However, since the frequency tracking unit is digital, its performance is usually much better than an analog PLL in a DAC, we have the possibility to average a lot of sample instants. Also, the high internal oversampling ratio L helps. Thus, with a high-quality output clock f_{so} , an AASRC can be very effective for “de-jittering” of a noisy f_{si} . But it does not eliminate jitter, and its performance can be compared to a normal analog clock recovery unit by looking at the transfer function of the frequency tracking unit.

Summary:

In this document, the reader has been introduced to asynchronous arbitrary sample-rate conversion or “resampling”. A review of the concepts behind oversampling and undersampling gave the basic understanding necessary to explain fractional and later arbitrary sample-rate conversion. We have seen what key modules are necessary in such a design and given a simple estimate of the performance requirements for these. In addition, we have explained how sample-rate conversion is done totally asynchronous, with separate source and sink clock signals. We have also seen that an AASRC can provide very good jitter rejection, but that it will add some distortion and noise to the signal. AASRCs can be made with extremely high dynamic range and SNR, but will then inevitably be of very high complexity.

Ivar Løkken, 12/4-05.

Reference 1: Rothacher, F.M.: "*Sample-Rate Conversion: Algorithms and VLSI Implementation*", PhD-thesis, Swiss Federal Institute of Technology, Zurich, 1995.

Reference 2: Evangelista, G: "*Zum Entwurf Digitaler Systeme zur Asynchronous Abstratenumsetzung*", PhD-thesis, Ruhr-Universität Bochum, December 2000.

Reference 3: Arnost, V.: "*Comparison of Interpolation Methods in Real-Time Sound Processing*"

Reference 4: Smith, O.J.: "*The Digital Audio Resampling Home-Page*", <http://www-csrma.stanford.edu/~jos/resample/>

Reference 5: Laakso, T.I. et al: "*Splitting the Unit Delay--Tools for Fractional Delay Filter Design*", *IEEE Signal Processing Magazine*, vol. 13, pp. 30-60, January 1996.

Reference 6: J. O. Smith and P. Gossett, "*A flexible sampling-rate conversion method*," Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, San Diego, vol. 2, pp. 19.4.1-19.4.2, IEEE Press, March 1984.

Reference 7: Rabiner, L.R.: "*Approximate Design Relationships for Lowpass FIR Digital Filters*", *IEEE Trans. Audio Electroacoustics*, October 1973

Reference 8: Dunn, J: "*Jitter and Digital Audio Measurements*" AES Preprint 6111, April 1994.