



In multirate signal processing different sample rates are used within a system to achieve the most efficient computation at each stage.

By changing to a lower sample rate, less samples have to be processed so the computation takes less time.

To achieve these efficiencies in practice requires special techniques of multirate filtering.

As an example, take a very narrow lowpass filter:

Sampling frequency 50 kHz
Passband cutoff 800 Hz
Stopband 1 kHz
Passband attenuation 1 dB
Stopband attenuation 60 dB

To implement this filter as a standard Parks-McClellan algorithm design requires 681 taps: in computational terms, 681 multiplies and additions.

This example filter could not be realised on any available general purpose DSP processor.

One approach might be to split the processing between multiple processors, with the signal data distributed between them. But there is a better way, requiring only the application of a little cleverness.

The filter is obviously a narrow band low pass (800 Hz), so a lower sample rate could have been used despite the system-wide requirement for quite a high sample rate (50 kHz).

The high sample rate may be imposed by system constraints such as the need to output signals at the input rate.

If the sample rate could be cut, say to 2500 Hz, the same filter would need only 35 multiplies and additions.

This leads to the idea of reducing the sample rate (decimation) to a lower sample rate, filtering at the lower rate, then changing the sampling rate upward (interpolation) again to the original sampling rate.

Reducing the sampling rate by a factor of N is achieved by discarding every N-1 samples: or equivalently keeping every Nth sample. But to avoid aliasing we have to implement a lowpass antialiasing filter before discarding every N-1 samples.

At first sight the antialiasing filter needs as much computation as we save by working at the lower, downsampled, rate. But some of the samples after the antialiasing low pass filter are going to be discarded.

It is possible to design the filter in such a way that only the retained samples are used in the calculation.

The trick is to embed downsampling into the filter - which can now run at the low sample rate.

To change the sampling rate back to the original system rate (interpolation by factor N), N-1 zeros are inserted into the output stream after every sample from the downsample filtering process. But to interpolate smoothly between the samples a reconstruction filter has to be implemented on the final output sequence.

Again, this extra reconstruction filter at first sight uses up any computational gain, but again we don't need to use the zeroes, so as with the antialiasing filter interpolation is embedded in the reconstruction filter - which can now run at the low rate.

In the example a very worthwhile computational gain of 25 to 1 can be achieved in practice.