Design of Over Sampling Audio DAC

Digital Signal Processing

Notes

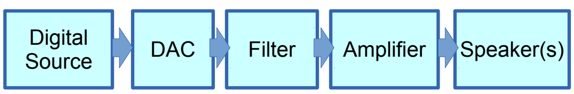
Contents

Notes

The Problem

Reproduction stage of digital audio system includes amplifier and speaker(s).
Preceded by digital-to-analogue conversion (DAC) typically involving:

- The DAC:
  - Takes in clocked sequence of parallel words and
  - Outputs a series of voltage levels which are constant for the duration of the sampling period;
- A low-pass filter:
  - Eliminates spectral copies inherent in any sampled signal and
  - Passes only the original signal, replacing the staircase-like output from the DAC by a much smoother waveform.



Notes

The Problem

- Each part has its own particular problems but not address the DAC in this unit.
- For the low-pass filter, the problem is with:

The guard bands between the original signal spectrum and its first sampling-induced duplicate

- These are very small in typical digital audio systems such as CD:
  - The top of the audio band is at 20 kHz and
  - With a sampling frequency of 44.1 kHz,
  - The bottom end of first duplicate spectrum is only at 24.1 kHz.

This is an extremely small guard band.

- A very narrow guard bands means:

high order analogue reconstruction filters

- Leading to pass-band amplitude ripple & nonlinear phase (and generally temperature and time stability issues).

Notes

The Solution: Oversampling

Solution is to:

Numerically re-sample at a higher rate (oversampling), which eases the constraint on the analogue filter because of the larger guard band.

Interpolation: *two possibilities*

- Re-sampling by inserting additional zero-valued samples between original signal samples does not eliminate frequency components about the harmonics of original sample rate.
  - It just increases the sample clock rate.
- Therefore need to explicitly interpolate i.e. replace inserted zeros with values that "smoothly join the dots".

Notes

The Solution: Oversampling

Goal:

Replace difficult-to-design-and-make high order analogue filter with combination of a digital and analogue filter.

Advantage:

- Low-order analogue filter
- → much easier to design,
- → cheaper to make and more stable etc.

Disadvantage:

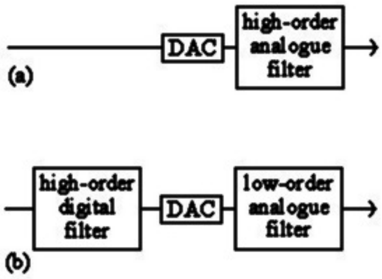
- digital filter and DAC at much higher sampling frequency.

Notes

The Solution: Oversampling

Notes

Replacing difficult-to-design-and-make high order analogue filter with combination of a digital and analogue filter.



The Solution: Oversampling

Notes

Formally, we may define oversampling in a D/A context as:

"*numerically synthesising a sample sequence equivalent to that which would have resulted if the original analogue signal had been sampled at a much higher rate.*"

As such, oversampling consists of two processes:

- **up-sampling**: inserting additional zeros between the existing samples to increase sample rate but not changing spectral content
- **interpolation**: using high-order digital filter to remove intermediate spectral copies and create a large guard-band

The Solution: Oversampling

Notes

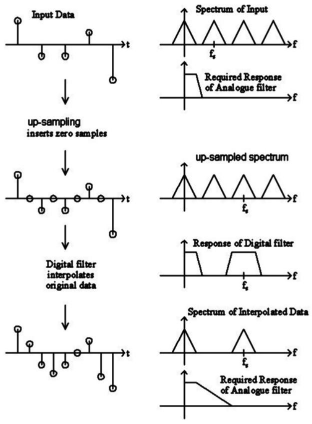
2-times oversampling example
(Next slide)

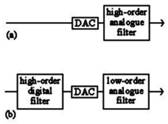
- Left-hand column: time-domain
- Right-hand column: frequency domain.

After the up-sampling process, requirement for analogue filter is unchanged.

Only after the digital filter has removed the intermediate spectral copies that the job of the analogue filter becomers easier.

The Solution: Oversampling





Notes

The Solution: Oversampling

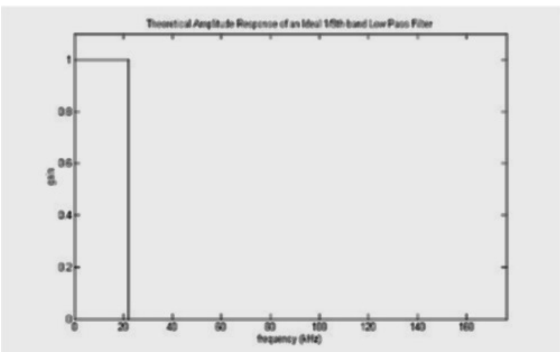
Benefits of oversampling when using realistic numbers:
Example of Butterworth analogue filters with & without 8-times oversampling:

basic f_s	=	44.1 kHz	basic f_s	=	44.1 kHz
(oversampling	=	1)	(oversampling	=	8)
New f_s	=	44.1 kHz	New f_s	=	352.8 kHz
first alias freq	=	24.1 kHz	first alias freq	=	332.8 kHz
filter order	=	**60**	filter order	=	**5**
$f_{3\text{dB}}$	=	20 kHz	$f_{3\text{dB}}$	=	30 kHz
gain @ 20kHz	=	-3 dB	gain @ 20kHz	=	-0.07 dB
gain @ alias	=	-97 dB	gain @ alias	=	-104 dB

Notes

The Solution: Oversampling

Illustration of frequency-domain specification of 1/8th-band digital filter (suitable for 8-times oversampling)



Notes

The Solution: Oversampling

Learning Task
Use the MATLAB program TFA_analogue_filters.m to explore the problem of analogue filter order with varying degrees of oversampling and varying gains @ 20 kHz (typically an audio application would wish for significantly less than 1dB of signal loss at 20 kHz).

- Use the source filter option;
- choose a line-spacing of 11025 (to force a 44.1 kHz basic sampling rate);
- choose a 16 bit DAC.

If you had to design a digital filter for a 16-bit DAC whose analogue filter was only a 3rd-order Butterworth filter, work out what oversampling ratio (power of 2) you would need to use.

Notes

FIR Implementation Issues

FIR digital filter in an oversampling audio DAC is clocked at some integer multiple of the original audio sampling rate, giving it a shorter time in which to carry out its MACs.

If original audio sampled at 44.1 kHz and DAC running at 8 times oversampling then time available to calculate each output from the filter (per channel) is:

$$\frac{1}{44100 \times 8} = 2.834...\mu s$$

If the DSP chip MAC time is 20ns then to implement filtering for a stereo DAC (2 channels) each channel can only use:

$$\frac{1}{44100 \times 8 \times 20 \times 10^{-9} \times 2} = 70.86... \text{MACs per output sample.}$$

Of course we cannot make use of a fraction of a MAC, so that means just 70 MACs per output sample on each channel.

Notes

FIR Implementation Issues

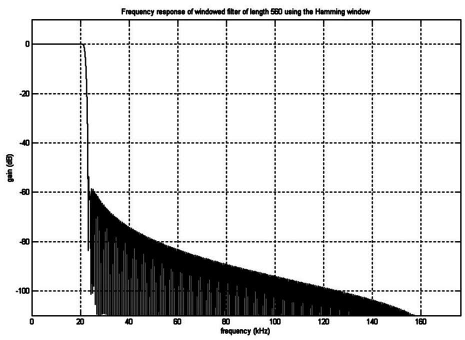
Filters of order 69 (70 coefficients), may not be enough to achieve the sharp cut-off needed.

- However, oversampled to a clock rate of 352.8 kHz means data going into the filter consists of the original audio samples with
  - **seven zero-valued samples between nonzero audio samples.**
- As a result **87.5% of apparent MACs are multiplying by zero,**
  - which therefore contributes nothing to the output samples,
- Thus actual filter design can be order of 559 (560 coefficients) provided DSP chip (or its programmer!) is clever enough to omit the multiply-by-zero terms.

Notes

FIR Implementation Issues

Using MATLAB program oversampling_DAC_filter.m, we can see if this gives us a good enough response. Unfortunately it doesn't:



Notes

FIR Implementation Issues

Notes

- Filter length = 560
- MACs per output sample = 70
- Window selected = Hamming (the design method used and how windows get involved is discussed later)
- Gain at 20kHz = 0 dB / Gain at 24.1kHz = -65.05 dB

The attenuation at 24.1 kHz (the first alias frequency) is insufficient. For a 16-bit DAC we would want an attenuation of at least 96 dB; more for a higher-precision device. This means, for this application, we need a faster DSP chip.

FIR Implementation Issues

An order-N FIR filter obeys the finite convolution sum:

$$y[n] = \sum_{k=0}^{N} h[k]x[n-k]$$

which involves $(N+1)$ MACs.

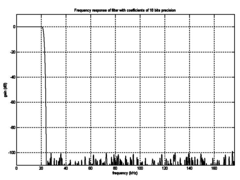
- In a practical implementation, both input signal $x[n]$ and filter's pulse response $h[k]$ represented by finite precision binary numbers,
  - typically integers or a special kind of fractions where the denominator is a power of 2, e.g. 5/64 (these sort of fractions are known as dyadic rationals).
- Important to understand impact of finite precision of the coefficients:

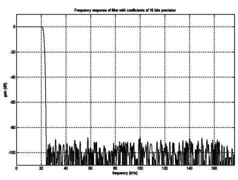
A filter may have a suitable response in simulations when its coefficients are represented as arbitrary precision floating point numbers but may fail to meet its design specification when coefficients are quantised to realistic finite precision.

Notes

FIR Implementation Issues

Consider following two frequency responses, (FIR_coeff_precision.m) using same theoretical response (produced by oversampling_DAC_filter.m).





Left (18-bit precision coefficients), response just satisfies specification (-96 dB from 24.1 kHz upwards) but 16-bit version does not by more than 8 dB (at around 100 kHz).

This means our DSP chip must be capable of at least 18-bit multiplication, in addition to having a fast enough MAC.

Notes

FIR Implementation Issues

Learning Task
Download the MATLAB program FIR_coeff_precision.m and save it to your MATLAB "work" directory. Use it in conjunction with oversampling_DAC_filter.m to check the above information. Then use all the MATLAB programs in this book to do a complete design to replace an 18-bit DAC in a system with a basic sampling rate of 44.1 kHz, assuming you cannot use a Butterworth filter whose order is more than 5 and you want the attenuation at 20 kHz to be no more than 0.1 dB. Remember that TFA_analogue_filters.m only allows oversampling ratios which are powers of 2 (because in that application it is related to the use of the FFT). Assume your DSP chip takes 15ns to carry out one MAC.

Notes

Summary

- Oversampling can help ease stringent analogue filter requirements;
- Increases *digital* computational demand.

Notes