Mark Borgerding

# Turning Overlap-Save into a Multiband Mixing, Downsampling Filter Bank

In this article, we show how to extend the popular overlap-save (OS) fast convolution filtering technique to create a flexible and computationally efficient bank of filters, with frequency translation and decimation implemented in the frequency domain. In addition, we supply some tips for choosing appropriate fast Fourier transform (FFT) size.

Fast convolution is a well-known and powerful filtering technique. All but the shortest finite impulse response (FIR) filters can be implemented more efficiently in the frequency domain than when performed directly in the time domain. The longer the filter impulse response, the greater the speed advantage of fast convolution.

When more than one output is filtered from a single input, some parts of the fast convolution algorithm are redundant. Removing this redundancy increases fast convolution's speed even more. Sample rate change by decimation (downsampling) and frequency translation (mixing) techniques can also be incorporated efficiently in the frequency domain. These concepts can be combined to create a flexible and efficient bank of filters. Such a filter bank can implement mixing, filtering, and decimation of multiple arbitrary channels much faster than direct time-domain implementation.

"DSP Tips and Tricks" introduces practical tips and tricks of design and implementation of signal processing algorithms so that you may be able to incorporate them into your designs. We welcome readers who enjoy reading this column to submit their contributions. Contact Associate Editors Rick Lyons (r.lyons@ieee.org) or Amy Bell (abell@vt.edu).

## SOMETHING OLD AND SOMETHING NEW

The necessary conditions for vanilla-flavored fast convolution are covered pretty well in the literature. However, the choice of FFT size is not. Filtering multiple channels from the same forward FFT requires special conditions not detailed in textbooks. To downsample and shift those channels in the frequency domain requires still more conditions.

The first section is meant to be a quick reminder of the basics before we extend the OS fast convolution technique. If you feel comfortable with these concepts, skip ahead. On the other hand, if this review does not jog your memory, check your favorite DSP book for "Fast Convolution," "Overlap-Add" (OA), "Overlap-Save," or "Overlap-Scrap" [1]–[5].

## REVIEW OF FAST CONVOLUTION

The convolution theorem tells us that multiplication in the frequency domain is equivalent to convolution in the time domain [1]. Circular convolution is achieved by multiplying two discrete Fourier transforms (DFTs) to effect convolution of the time sequences that the transforms represent. By using the FFT to implement the DFT, the computational complexity of circular convolution is approximately $O(N\log_2 N)$ instead of $O(N^2)$, as in direct linear convolution. Although very small FIR filters are most efficiently implemented with direct convolution, fast convolution is the clear winner as the FIR filters get longer. Conventional wisdom places the efficiency crossover point at 25–30 filter coefficients. The actual value depends on the relative strengths of the platform in question (CPU pipelining, zero-overhead looping, memory addressing modes, etc.). On a desktop processor with a highly optimized FFT library, the value may be as low as 16. On a fixed-point DSP with a single-cycle multiply-accumulate instruction, the efficiency value can be over 50.

Fast convolution refers to the block-wise use of circular convolution to accomplish linear convolution. Fast convolution can be accomplished by OA or OS methods. OS is also known as "overlap-scrap" [5]. In OA filtering, each signal data block contains only as many samples as allows circular convolution to be equivalent to linear convolution. The signal data block is zero-padded prior to the FFT to prevent the filter impulse response from "wrapping around" the end of the sequence. OA filtering adds the input-on transient from one block with the input-off transient from the previous block.

In OS filtering, shown in Figure 1, no zero-padding is performed on the input data, thus the circular convolution is not equivalent to linear convolution. The portions that "wrap around" are useless and discarded. To compensate for this, the last part of the previous input block is used as the beginning of the next block. OS requires no addition of transients, making it faster than OA. The OS filtering method is recommended as the basis for the techniques outlined in the remainder of this article. The nomenclature "$\text{FFT}_N$" in Figure 1 indicates that an FFT's input sequence is zero-padded to a length of $N$ samples, if necessary, before performing the $N$-point FFT.

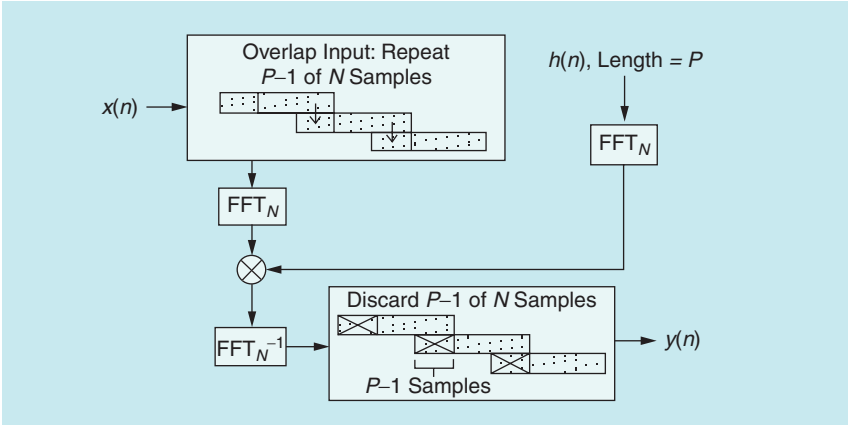
For clarity, the following list defines the symbols used in this article.



1053-5888/06/$20.00©2006IEEE

**SYMBOL CONVENTIONS**

| | |
|---|---|
| $x(n)$ | Input data sequence |
| $h(n)$ | FIR filter impulse response |
| $y(n)=x(n)*h(n)$ | Convolution of $x(n)$ and $h(n)$ |
| $L$ | Number of new input samples consumed per data block |
| $P$ | Length of $h(n)$ |
| $N=L+P-1$ | FFT size |
| $V=N/(P-1)$ | Overlap factor, the ratio of FFT length to filter transient length |
| $D$ | Decimation factor |




**[FIG1]** **Overlap-save (OS) filtering,** $y(n) = x(n) * h(n)$.

## CHOICE OF FFT SIZE: COMPLEXITY IS RELATED TO FILTER LENGTH AND OVERLAP FACTOR

Processing a single block of input data that produces $L$ outputs incurs a computational cost related to $N\log(N) = (L+P-1)\log(L+P-1)$. The computational cost per sample is related to $(L+P-1)\log(L+P-1)/L$. The filter length $P$ is generally a fixed parameter, so choosing $L$ such that this equation is minimized gives the theoretically optimal FFT length. It may be worth mentioning that the log base is the radix of the FFT. The only difference between different-based logarithms is a scaling factor. This is irrelevant to "big O" scalability, which generally excludes constant scaling factors.

Larger FFT sizes are more costly to perform, but they also produce more usable (nonwraparound) output samples. In theory, one is penalized more for choosing too small an overlap factor, $V$, than too large. In practice, the price of large FFTs paid in computation and/or numerical accuracy may suggest a different conclusion.

Some other factors to consider while deciding the overlap factor for fast convolution filtering include:

- FFT speed—It is common for actual FFT computational cost to differ greatly from theory, e.g., due to memory caching. ("In theory there is no difference between theory and practice. In practice there is." — Yogi Berra, American philosopher and occasional baseball manager.)
- FFT accuracy—the numerical accuracy of fast convolution filtering is dependent on the error introduced by the FFT-to-IFFT (inverse FFT) round trip. For floating-point implementations, this may be negligible, but fixed-point processing can lose significant dynamic range in these transforms.
- Latency—Fast convolution filtering process increases delay by at least $L$ samples. The longer the FFT, the longer the latency.

While there is no substitute for benchmarking on the target platform, in the absence of benchmarks, choosing a power-of-two FFT length about four times the length of the FIR filter is a good rule of thumb.

## FILTERING MULTIPLE CHANNELS: REUSE THE FORWARD FFT

It is often desirable to apply multiple filters against the same input sample sequence. In these cases, the advantages of fast convolution filtering become even greater. The computationally expensive operations are the forward FFT, the IFFT, and the multiplication of the frequency responses. The forward FFT needs to be computed just once. This is roughly a "Buy one filter. Get one 40% off" sale.

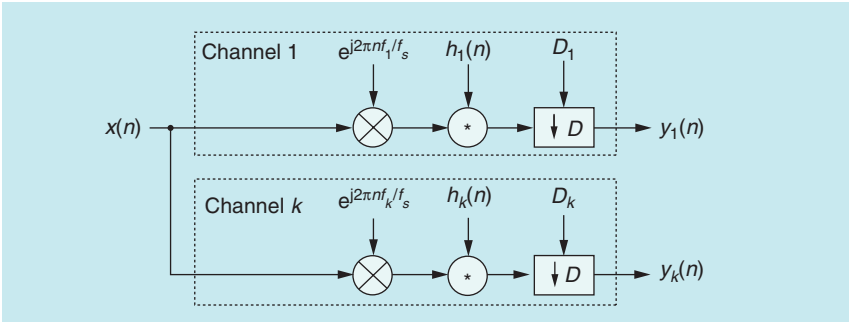
To realize this computational cost savings for two or more filters, all filters must have the same impulse response length. This condition can always be achieved by zero padding the shorter filters. Alternately, the engineer may redesign the shorter filter(s) to make use of the additional coefficients without increasing the computational workload.

## FREQUENCY DOMAIN DOWNSAMPLING: ALIASING IS YOUR FRIEND

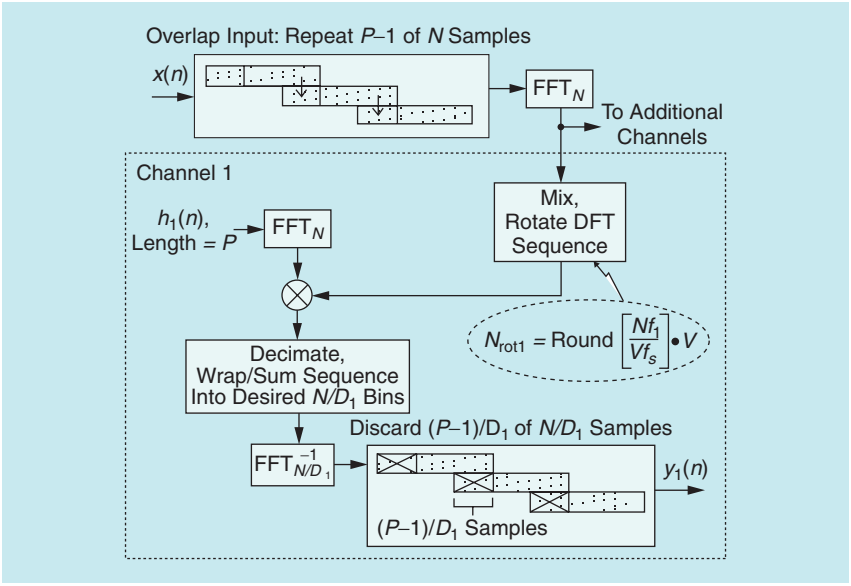
The most intuitive method for reducing sample rate in the frequency domain is to simply perform a smaller IFFT using only those frequency bins of interest. This is not a 100% replacement for time domain downsampling. This simple method will cause ripples (Gibbs phenomenon) at FFT buffer boundaries caused by the multiplication in the frequency domain by a rectangular window.

The sampling theorem tells us that sampling a continuous signal aliases energy at frequencies higher than the Nyquist rate (half the signal sample rate) back into the baseband spectrum (below the Nyquist rate). This is equally true for decimation of a digital sequence as it is for analog-to-digital conversion [6]. Aliasing is a natural part of downsampling. To accurately implement downsampling in the frequency domain, it is necessary to preserve this behavior. It should be noted that, with a suitable antialiasing filter, the energy outside the selected bins might be negligible. This discussion is concerned with the steps necessary for equivalence. The designer should decide how much aliasing, if any, is necessary.

Decimation (i.e., downsampling) can be performed exactly in the frequency domain by coherently adding the frequency components to be aliased [7].




[FIG2] **Conceptual model of a filter bank.**




[FIG3] **OS filter bank.**

The following octave/Matlab code demonstrates how to swap the order of an inverse DFT and decimation.

```
% Make up a completely
random frequency spectrum
Fx = randn(1,1024) +
i*randn(1,1024);

% Time-domain decimation —
inverse transform then
decimate
x_full_rate = ifft(Fx);
x_time_dom_dec =
x_full_rate(1:4:1024);
%Retain every fourth sample

% Frequency-domain
decimation, alias first,
then inverse transform
Fx_alias = Fx(1:256) +
Fx(257:512) + Fx(513:768) +
Fx(769:1024);
x_freq_dom_dec =
ifft(Fx_alias)/4;
```

The sequences x_time_dom_dec and x_freq_dom_dec are equal to each other. The above sample code assumes a complex time-domain sequence for generality. The division by four in the last step accounts for the difference in scaling factors between the IFFT sizes. As various FFT libraries handle scaling differently, the designer should keep this in mind during implementation. It's worth noting that this discussion assumes the FFT length is a multiple of the decimation rate $D$. That is, $N/D$ must be an integer.

To implement time-domain decimation in the frequency domain as part of fast convolution filtering, the following conditions must be met.

1) The FIR filter order must be a multiple of the decimation rate $D$.

$P - 1 = K_1 D$

2) The FFT length must be a multiple of the decimation rate $D$.

$L + P - 1 = K_2 D$

where

- D is the decimation rate or the least common multiple of the decimation rates for multiple channels
- $K_1$ and $K_2$ are integers.

Note if the overlap factor $V$ is an integer, then the first condition implies the second. It is worth noting that others have also explored the concepts of rate conversion in OA/OS [8]. Also note that decimation by large primes can lead to FFT inefficiency. It may be wise to decimate by such factors in the time domain.

## MIXING AND OS FILTERING: ROTATE THE FREQUENCY DOMAIN FOR COARSE MIXING

Mixing, or frequency shifting, is the multiplication of an input signal by a complex sinusoid [1]. It is equivalent to convolving the frequency spectrum of an input signal with the spectrum of a sinusoid. In other words, the frequency spectrum is shifted by the mixing frequency.

It is possible to implement time-domain mixing in the frequency domain by simply rotating the DFT sequence, but there are limitations:

1) The precision with which one can mix a signal by rotating a DFT sequence is limited by the resolution of the DFT.

2) The mixing precision is limited further by the fact that we don't use a complete buffer of output in fast convolution. We use only $L$ samples. We must restrict the mixing to the subset of frequencies whose periods complete in those $L$ samples. Otherwise, phase discontinuities occur. That is, one can only shift in multiples of $V$ bins.

The number of "bins" to rotate is

$$N_{\text{rot}} = \text{round}\left(\frac{Nf_r}{Vf_s}\right) \bullet V, \qquad (1)$$

where $f_r$ is the desired mixing frequency and $f_s$ is the sampling frequency. The second limitation may be overcome by using a bank of filters corresponding to

different phases. However, this increase in design/code complexity probably does not outweigh the meager cost of multiplying by a complex phasor.

If coarse-grained mixing is unacceptable, mixing in the time domain is a better solution. The general solution to allow multiple channels with multiple mixing frequencies is to postpone the mixing operation until the filtered, decimated data is back in the time domain.

If mixing is performed in the time domain:

- All filters must be specified in terms of the input frequency (i.e., nonshifted) spectrum.
- The complex sinusoid used for mixing the output signal must be created at the output rate.

## PUTTING IT ALL TOGETHER

By making efficient implementations of conceptually simple tools, we help ourselves to create simple designs that are as efficient as they are easy to describe. Humans are affected greatly by the simplicity of the concepts and tools used in designing and describing a system. We owe it to ourselves as humans to make use of simple concepts whenever possible. ("Things should be described as simply as possible, but no simpler."—A. Einstein.) We owe it to ourselves as engineers to realize those simple concepts as efficiently as possible.

The familiar and simple concepts shown in Figure 2 may be used for the design of mixed, filtered, and decimated channels. The design may be implemented more efficiently using the equivalent structure shown in Figure 3.

## SUMMARY

In this article, we outlined considerations for implementing multiple OS channels with decimation and mixing in the frequency domain, as well as supplying recommendations for choosing FFT size. We also provided implementation guidance to streamline this powerful multichannel filtering, down-conversion, and decimation process.

## ACKNOWLEDGMENTS

I would like to thank my wife, Elaine, for helping me find the time to write this, and David Evans, for being a DSP mentor and sounding board.

## AUTHOR

*Mark Borgerding* is a principal engineer at 3dB Labs, Inc., a small company specializing in DSP consulting and contract engineering services. He is often found lurking on the comp.dsp newsgroup or tinkering with his KISSFFT library.

## REFERENCES

[1] A. Oppenheimer and R. Schafer, *Discrete-Time Signal Processing*. Upper Saddle River, NJ: Prentice-Hall, 1989.

[2] L. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1975.

[3] R. Lyons, *Understanding Digital Signal Processing, 2/E*. Upper Saddle River, NJ: Prentice-Hall, 2004.

[4] S. Orfanidis, *Introduction to Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[5] M. Frerking, *Digital Signal Processing in Communication Systems*. New York: Chapman & Hall, 1994.

[6] R. Crochiere and L. Rabiner, *Multirate Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[7] M. Boucheret, I. Mortensen, and H. Favaro, "Fast convolution filter banks for satellite payloads with on-board processing," *IEEE J. Select. Areas. Commun.*, vol. 17, no. 2, pp. 238–248, Feb. 1999.

[8] S. Muramatsu and H. Kiya, "Extended overlap-add and -save methods for multirate signal processing," *IEEE Trans. Signal Processing*, vol. 45, no. 9, pp. 2376–2380, Sep. 1997. SP

dsp HISTORY continued from page 157

## REFERENCES

[1] P. Elias, "Predictive coding I," *IRE Trans. Inform. Theory*, vol. IT-1 no. 1, pp. 16–24, Mar. 1955.

[2] N. Wiener, *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. Cambridge, MA: MIT Press, 1949.

[3] C.E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, 623–656, 1948.

[4] P. Elias, "Predictive coding II," *IRE Trans. Inform. Theory*, vol. IT-1 no. 1, pp. 24–33, Mar. 1955.

[5] B.S. Atal and M.R. Schroeder, "Predictive coding of speech," in *Proc. 1967 Conf. Communications and Proc.*, Nov. 1967, pp. 360–361.

[6] B.S. Atal and M.R. Schroeder, "Adaptive predictive coding of speech," *Bell Syst. Tech. J.*, vol. 49 no. 8, pp. 1973–1986, Oct. 1970.

[7] B.S. Atal and S.L. Hanauer, "Speech analysis and synthesis by linear prediction of the speech wave," *J. Acoust. Soc. Amer.*, vol. 50, pp. 637–655, Aug. 1971.

[8] S. Saito, Fukumura, and F. Itakura, "Theoretical consideration of the statistical optimum recognition of the spectral density of speech", *J. Acoust. Soc. Japan*, Jan. 1967.

[9] F. Itakura and S. Saito, "A statistical method for estimation of speech spectral density and formant frequencies," *Electron. Commun. Japan*, vol. 53-A, pp. 36–43, 1970.

[10] T.E. Tremain, "The government standard linear predictive coding algorithm: LPC10," *Speech Technol.*, vol. 1, pp. 40–49, Apr. 1982.

[11] B.S. Atal and J.R. Remde, "A new model of LPC excitation for producing natural-sounding speech at low bit rates," in *Proc. ICASSP'82*, May 1982, pp. 614–617.

[12] B.S. Atal and M.R. Schroeder, "Stochastic coding of speech signals at very low bit rates," in *Proc. Int. Conf. Commun., ICC'84*, May 1984, pp. 1610–1613.

[13] M.R. Schroeder and B.S. Atal, "Code-excited linear prediction (CELP): High-quality speech at very low bit rates," in *Proc ICASSP'85*, Mar. 1985, pp. 937–940. SP