

Fast Convolution (FFT) Filtering: From Basics to Filter Banks**By Mark Borgerding****Comp.DSP Conference****July, 2004****Overview**

Fast convolution filtering is a powerful technique every DSP engineer should know. All but the shortest Finite Impulse Response (FIR) filters can be implemented more efficiently in the frequency domain than directly in the time domain. The longer the filter; the greater the advantage.

If more than one output is filtered from a single input, some parts of the algorithm are redundant. The removal of this redundancy further increases the speed. Efficient techniques for downsampling the filtered output(s) can also be incorporated in the frequency domain. Any mixing of the output sequences must be performed in a manner consistent with the above structures.

These concepts can be combined to create a flexible and efficient bank of filters. This filter bank implements mixing, filtering, and decimation of multiple arbitrary channels much faster than direct implementation can.

FIR Filtering is Convolution

Finite Impulse Response (FIR) filtering is one of the cornerstones of digital signal processing. The FIR filter has many advantages over the Infinite Impulse Response (IIR) filter, including guaranteed stability and linear phase¹. The only disadvantage is the relatively large number of coefficients required. Direct implementation of the FIR filter incurs a computational cost per sample that increases linearly with respect to the length of the filter. In order to halve the transition bandwidth of a filter, a FIR twice as long and twice as costly is required². This complexity is unfortunate, but not surprising.

FIR filtering can easily be called convolution. The two concepts share the same mathematical definition.

$$y(n) = h(n) * x(n) = \sum_m h(n-m)x(m)$$

For:

$$h(n) : 0 \leq n < P$$

$$\text{and } x(n) : 0 \leq n < L,$$

$$y(n) = h(n) * x(n) = \sum_{m=0}^{P+L-2} h(n-m)x(m)$$

The length of the convolution of $h(n)$ and $x(n)$ is the sum of their lengths, minus 1.

$$N = P + L - 1$$

Restated: if $y(n)$ is $x(n)$ filtered by the FIR filter $h(n)$, then $y(n)$ is one sample shorter than the combined length of the sequences x and h .

The complexity of the convolution is $O(LP)$. That is, the work is linearly related to the product of L and P .

1 If the coefficients of the FIR filter are complex-conjugate symmetric

2 With direct computation of the convolution sequence

Conceptualizing the convolution sequence.

The mechanics of convolution can be thought of in terms of an outer product matrix, with the columns indicating the elements of one sequence, and the rows representing the other sequence[Orfan].

	X ₀	X ₁	X ₂	X ₃	X ₄
h ₀	<i>h₀x₀</i>	<i>h₀x₁</i>	<i>h₀x₂</i>	<i>h₀x₃</i>	<i>h₀x₄</i>
h ₁	<i>h₁x₀</i>	<i>h₁x₁</i>	<i>h₁x₂</i>	<i>h₁x₃</i>	<i>h₁x₄</i>
h ₂	<i>h₂x₀</i>	<i>h₂x₁</i>	<i>h₂x₂</i>	<i>h₂x₃</i>	<i>h₂x₄</i>
h ₃	<i>h₃x₀</i>	<i>h₃x₁</i>	<i>h₃x₂</i>	<i>h₃x₃</i>	<i>h₃x₄</i>

Table 1: Convolution Table

The convolution sequence $y_n = h * x = \sum_m h_{n-m} x_m$ is the summation along the antidiagonal lines of the matrix. (i.e. step up one, right one)

y ₀ =	<i>h₀x₀</i>	Input-on Transient
y ₁ =	<i>h₁x₀ + h₀x₁</i>	Input-on Transient
y ₂ =	<i>h₂x₀ + h₁x₁ + h₀x₂</i>	Input-on Transient
y ₃ =	<i>h₃x₀ + h₂x₁ + h₁x₂ + h₀x₃</i>	Steady State
y ₄ =	<i>h₃x₁ + h₂x₂ + h₁x₃ + h₀x₄</i>	Steady State
y ₅ =	<i>h₃x₂ + h₂x₃ + h₁x₄</i>	Input-off Transient
y ₆ =	<i>h₃x₃ + h₂x₄</i>	Input-off Transient
y ₇ =	<i>h₃x₄</i>	Input-off Transient

Table 2: Antidiagonal Sums of Convolution Table

The finite sequences h and x have implicit zeros before and after, e.g. h₋₄₂ = 0. At the beginning and end of the convolution; the sequence h does not fully overlap x, so parts of h are multiplied by zero. These regions are the input-on and input-off transients, respectively[Orfan].

Steady state occurs when the overlapping of the two sequences is the length of the shorter sequence. In graphical terms: the longest lines.

Circular Convolution

The Convolution Theorem tells us that convolution in the time domain is equivalent to multiplication in the frequency domain.

i.e.

$$\begin{aligned} \text{If } & \quad x(n) \quad \xleftrightarrow{F} \quad X(e^{j\omega}) \\ & \quad h(n) \quad \xleftrightarrow{F} \quad H(e^{j\omega}) \\ \text{and } & \quad y(n) = h(n) * x(n) \end{aligned}$$

$$\text{then } Y(e^{j\omega}) = X(e^{j\omega}) H(e^{j\omega}) \quad [\text{OppSch}]$$

So convolution of two sequences is the inverse Fourier transform of the product of their Fourier spectra.

We don't generally know the exact Fourier spectra of the signals. We can only estimate them using the Discrete Fourier Transform (DFT).

Circular convolution is convolution implemented using DFTs. It is therefore limited by the DFT periodicity. Like the DFT, circular convolution acts as if a sequence constitutes one period of a repeating signal.

The circular convolution of x and h is $\text{DFT}^{-1}(\text{DFT}(x) \text{DFT}(h))$

Note that to multiply two DFTs together, they must have the same number of elements. The sequences to be convolved must then be the same length. This is accomplished via zero padding.

The periodicity inherent in the DFT has the effect of combining the input-on and input-off transients into a summation that “wraps around” the edge of the matrix.

The part of the circular convolution sequence that is influenced by both the beginning and end of the sequence has been called “time aliasing” [OppSch], “wraparound difficulties” [RabGold], and “the parts that get wrapped around” [Orfan]. None of these terms is especially descriptive or concise.

The lack of a standard name for this important concept has been the root of at least one misunderstanding on the comp.dsp newsgroup. Lacking a better term, I will use “*wraparound transient response*” to describe the effect.

	X ₀	X ₁	X ₂	X ₃	X ₄
h ₀	h_0x_0	h_0x_1	h_0x_2	h_0x_3	h_0x_4
h ₁	h_1x_0	h_1x_1	h_1x_2	h_1x_3	h_1x_4
h ₂	h_2x_0	h_2x_1	h_2x_2	h_2x_3	h_2x_4
h ₃	h_3x_0	h_3x_1	h_3x_2	h_3x_3	h_3x_4

Table 3: Circular Convolution Table

Notice how the antidiagonal lines at the end wrap around to the beginning of the matrix.

y ₀ =	$h_3x_2 + h_2x_3 + h_1x_4 + h_0x_0$	Wraparound Transient
y ₁ =	$h_3x_3 + h_2x_4 + h_1x_0 + h_0x_1$	Wraparound Transient
y ₂ =	$h_3x_4 + h_2x_0 + h_1x_1 + h_0x_2$	Wraparound Transient
y ₃ =	$h_3x_0 + h_2x_1 + h_1x_2 + h_0x_3$	Steady State
y ₄ =	$h_3x_1 + h_2x_2 + h_1x_3 + h_0x_4$	Steady State

Table 4: Antidiagonals of Circular Convolution Table

The wraparound transient can be eliminated by appending zeros at the end of each sequence to mimic the implicit zeros used in linear convolution³. Zero padding both input sequences to length $N \geq P+L-1$ is sufficient to ensure the circular convolution is equivalent to the linear convolution.

	X ₀	X ₁	X ₂	X ₃	X ₄	0	0	0
h ₀	h_0x_0	h_0x_1	h_0x_2	h_0x_3	h_0x_4	0	0	0
h ₁	h_1x_0	h_1x_1	h_1x_2	h_1x_3	h_1x_4	0	0	0
h ₂	h_2x_0	h_2x_1	h_2x_2	h_2x_3	h_2x_4	0	0	0
h ₃	h_3x_0	h_3x_1	h_3x_2	h_3x_3	h_3x_4	0	0	0

Table 5: Circular Convolution Equivalent to Linear Convolution

³ In truth, the zeros do not need to be at the end of the sequence. Rotating the zeros effects a time shift, which can be useful in some cases.

Overlap-Add Filtering

Consider a FIR filter h , such that

$$y(n) = h(n) * x(n)$$

The input sequence x may be endless. Since circular convolution requires the computation of both input sequences' DFTs, it cannot be performed when one of the sequences is infinite. A solution to this problem is to use **Overlap-Add (OA) filtering** [OppSch]. OA computes a continuous convolution from smaller convolutions in a buffer-by-buffer manner.

The input x can be filtered by the finite length filter h (length P) by splitting x into L -sized chunks and convolving each chunk with the filter. Each iteration consumes L samples of input and produces a convolution $L+P-1$ samples long. We're not changing sampling rates. Where do the extra $P-1$ samples go?

The output buffers must **overlap** each other by $P-1$ samples. In the areas of overlap, the samples must be **added** together. The effect is that the input-on transient of one buffer is complemented by the input-off transient of the previous buffer.

If it seems unclear why the transient responses must be overlapped and added, it may be useful to recall a FIR filter's coefficients $h(n)$ are the **impulse response** of the system. An input consisting of a single delta pulse creates a succession of output samples, whose values are the coefficients of the FIR filter. If the delta pulse is scaled by C , the output response is scaled by C . If the delta pulse is delayed by m , the output is delayed by m . If the input is longer than one sample, the output will be the summation of the delayed, scaled impulse responses. In other words, the convolution of two sequences is the *overlapping of the responses* to the input impulses.

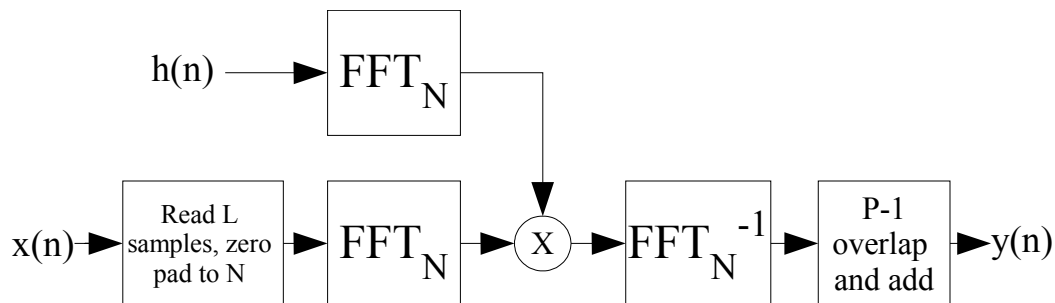


Illustration 1 Overlap-Add Filter

Overlap-Save Filtering

A trick for making fast convolution filtering even faster is to eliminate the addition step for the overlapping region. To do this, **Overlap-Save** OS breaks the rules for linear-circular convolution equivalence. It uses circular convolution *without zero-padding*.

The subsequent wraparound transient is useless and is discarded. OS filtering overlaps the input blocks rather than the output blocks. Overlapping sections of input ensure the full impulse response of each input is produced. This suggests the meaning behind an alternate name for the technique, *Overlap-Scrap* filtering [Frerk]⁴.

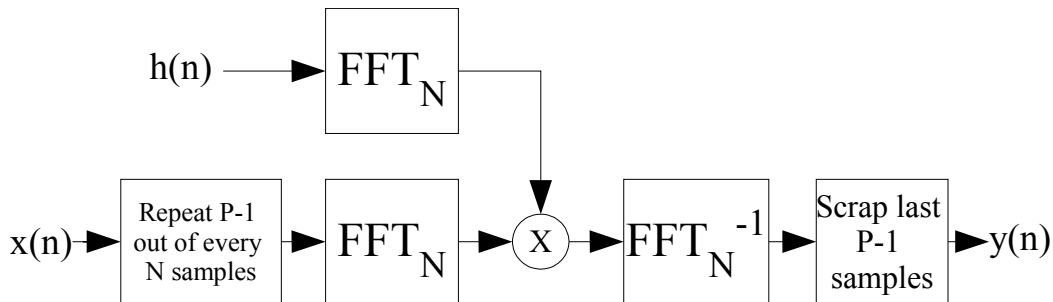


Illustration 2 Overlap-Save Filtering

Note the elimination of the vector addition does not change the fundamental scalability of fast convolution filtering. The complexity is still dominated by the $O(N \log N)$ FFT.

⁴ Overlap-Save is the more well-known name for the technique. I prefer the term Overlap-Scrap, since “Save” is too vague and could easily apply to either technique. Whereas only OS filtering “scraps” anything.

Table 8 reviews the differences between Overlap-Add and Overlap-Save.

	<i>Overlap-Add Filtering</i>	<i>Overlap-Save Filtering</i>
<i>FFT input buffer</i>	L input samples, zero-padded to L+P-1	N input samples, P-1 of which are repeated from the previous buffer
<i>IFFT output buffer</i>	The last P-1 samples from the previous output are added to the first P-1 samples of the current output. The first L (L = N-P+1) samples are then suitable for output.	The first P-1 samples of each IFFT output are discarded. The remaining N-P+1 samples are used as output.
<i>Speed</i>	Faster than direct for filters longer than 25-30 taps	Same scalability as OA, but slightly more efficient.
<i>Name notes</i>	The input-on and input-off transients are overlapped and added .	The input buffers are overlapping , i.e. some of the input is saved between iterations. ⁵

Table 8: Comparison of Overlap-Add and Overlap-Save

⁵ Alternately, the wraparound transient is “scrapped”.

Fast Convolution Filtering Complexity vs. Overlap Factor

The complexity of a single circular convolution is that of the FFT, $O(N \log N)$. However, the entire length of the convolution is not suitable for output.

Using

P = the length of the filter (dictated by design parameters)

L = the length of the input chunks

$N \geq L+P-1$ = the length of the circular convolution

Let us define the ratio of FFT length over the filter order as the **overlap factor**

$$V = \frac{L+P-1}{P-1}$$

How large should the circular convolution (i.e. FFT) be?

Processing a single chunk that produces L outputs incurs a computational cost related to $N \log(N) = (L+P-1) \log(L+P-1)$.

So the cost per sample is $O((L+P-1) \log(L+P-1) / L)$

We can express the cost per sample as $O\left(\frac{V}{V-1} \log(V(P-1))\right) \approx O\left(\frac{V}{V-1} \log(VP)\right)$

The complexity depends on the product of two quantities. As V increases, $\frac{V}{V-1}$ decreases asymptotically toward unity. Conversely, $\log(VP)$ increases with larger V .

In theory, one is penalized more for choosing too small an overlap factor than too large.

“In theory there is no difference between theory and practice. In practice there is.”

-- Yogi Berra

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

- FFT Speed – if the entire sequence does not fit in cache, it will be slow.
- FFT Accuracy – the numerical accuracy of fast convolution filtering is dependent on the error introduced by the FFT->IFFT round trip. For floating point implementations, this may be negligible, but fixed point processing loses significant dynamic range in larger transforms.
- Latency – Fast convolution filtering process extends the group delay by at least L samples. So the longer the FFT, the longer the latency.

There is no substitute for benchmarking the target platform. In the absence of benchmarks, an overlap factor of 4 to 8 is a good rule of thumb.

Parallel Filtering

It is sometimes desirable to apply multiple filters against the same input. In these cases, the advantages of fast convolution filtering become even greater. The three most expensive operations are the forward FFT, the inverse FFT, and the multiplication of the frequency responses. The multiplication step is the least expensive of these, but not negligible. The forward FFT need be computed only once, offering a significant savings. It is roughly a “Buy one filter. Get one 40% off” sale⁶.

In order to realize this cost savings for 2 or more filters, all filters must have the same length. This condition can always be forced by zero padding the shorter filters. Alternately, the engineer may also redesign the shorter filter(s) to make use of the additional taps.

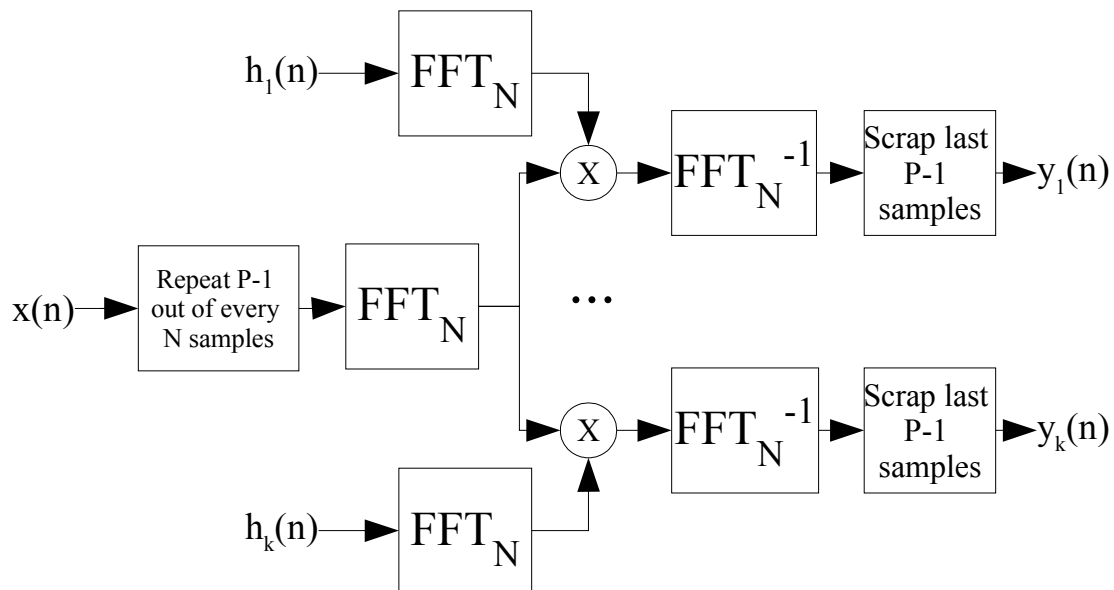


Illustration 3 Parallel Overlap-Save Filters

⁶ Observed FFT sizes: 256 to 2048

Downsampling in the Frequency Domain

(or How I Learned to Stop Worrying and Love the Aliasing)

The sampling theorem tells us that undersampling a signal aliases energy at frequencies higher than the Nyquist rate back into the baseband spectrum. This is just as true for decimation of a digital sequence as it is for analog-to-digital conversion[CroRab]. Decimation (i.e. downsampling) can be performed in the frequency domain by coherently adding the frequency components to be aliased. [Bouch]

The following octave/matlab code demonstrates the case of downsampling by 4:

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

% BASELINE: traditional decimation path -- inverse transform then decimate
x_highrate = ifft( X_freq );
x_baseline = x_highrate(1:4:1024); % every fourth sample

% ALTERNATIVE: decimate in frequency domain, then transform back to time
X_freq_alias = X_freq(1:256) + X_freq(257:512) + X_freq(513:768) + X_freq(769:1024);
x_alt = ifft(X_freq_alias) / 4; % scale

max_error = max( abs( x_alt - x_baseline ) )
```

The extra addition operations required to alias the frequency bins are more than made up for by savings due to the $O(N \log N)$ complexity of the FFT. The speed advantages that come with the smaller inverse FFT becomes more pronounced at higher decimation rates.

Its worth noting this discussion assumes the number of frequency bins is a multiple of the decimation rates. i.e. N/D must be an integer.

Frequency Domain Decimation, as Applied to Fast Convolution

There are some considerations to ensure that we can correctly decimate in the frequency domain. We mentioned that the FFT length must be a multiple of the decimation rate(s) of interest. That, by itself, is not sufficient to implement fast convolution filtering with downsampling. The length of the wraparound transient, $P-1$ (i.e the filter order) must also be a multiple of the decimation rate(s).

To downsample in the frequency domain as part of fast convolution filtering, the following conditions must be met.

1. The filter order must be a multiple of all decimation rates.⁷
 $P-1 = K_1 D$
2. The FFT length must be a multiple of all decimation rates.⁸
 $L+P-1 = K_2 D$
 where
 - D is the least common multiple of the decimation rates
 - K_1 and K_2 are integers

If the overlap factor V is an integer, then the first condition implies the second.

$$V = \frac{L+P-1}{P-1} = \frac{K_2}{K_1}$$

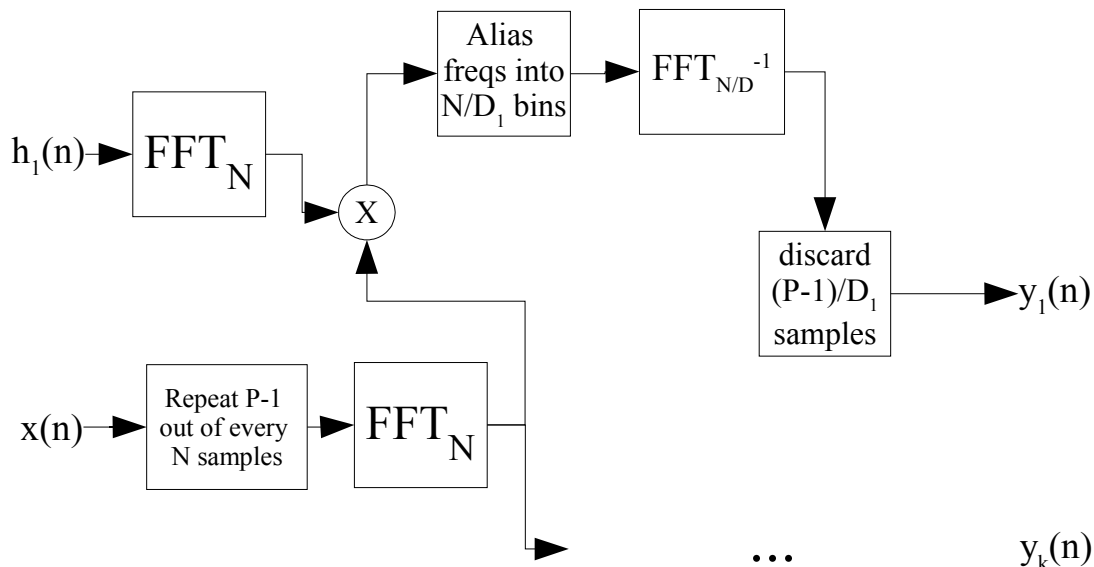


Illustration 4 Decimating Overlap-Save Filter

⁷ Zero-padding can achieve this.

⁸ Decimation by large primes can lead to FFT inefficiency. In such cases, it may be wise to decimate by these factors in the time domain.

Where Does Mixing Fit In?

Mixing, or frequency shifting [OppSch], is the multiplication of an input signal by a complex sinusoid. 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.

$$\begin{aligned} \text{If} \quad & x(n) \leftrightarrow X(e^{j\omega}) \\ \text{then} \quad & e^{j\omega_0} x(n) \leftrightarrow X(e^{j(\omega-\omega_0)}) \end{aligned}$$

It is possible to implement mixing in the frequency domain by 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.

The second limitation may be overcome by careful implementation of a bank of polyphase filters, but the first limitation would still remain. Mixing in the frequency domain is not very flexible. For a general solution, we should mix in the time domain.

We cannot mix the input signal before the forward FFT, since all channels use it, and they can all have different mixing frequencies. Mixing in the frequency domain is undesirable for the above reasons. The remaining option is to postpone mixing until the filtered, decimated data is back in the time domain.

Mixing in the time domain is easy from a complexity viewpoint. It is a fixed amount of work per sample. That is, $O(1)$.

Filtering and downsampling can be applied in parallel structures as described above. The output signal may then be mixed to shift its spectrum as desired.

Care must be taken if mixing is performed only after the signals are filtered and decimated.

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

Note that mixing by a frequency outside $[0, 2\pi)$ is equivalent to mixing by the aliased frequency within the $[0, 2\pi)$ range.

Summary

The familiar and proven concepts shown in Illustration 5 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 Illustration 6.

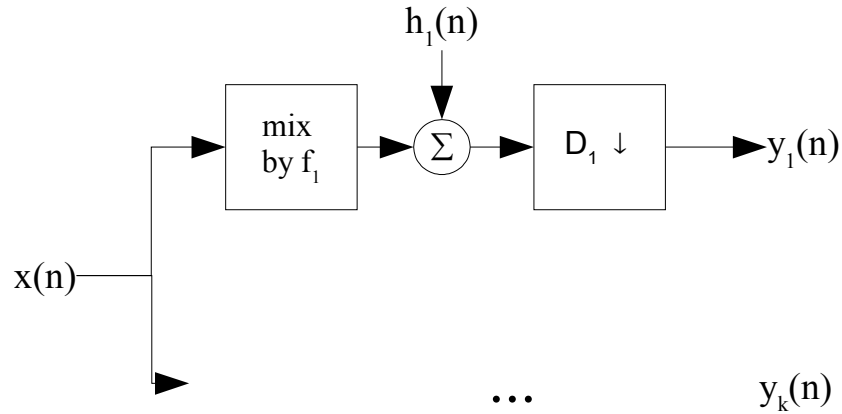


Illustration 5 Classical Filter Bank

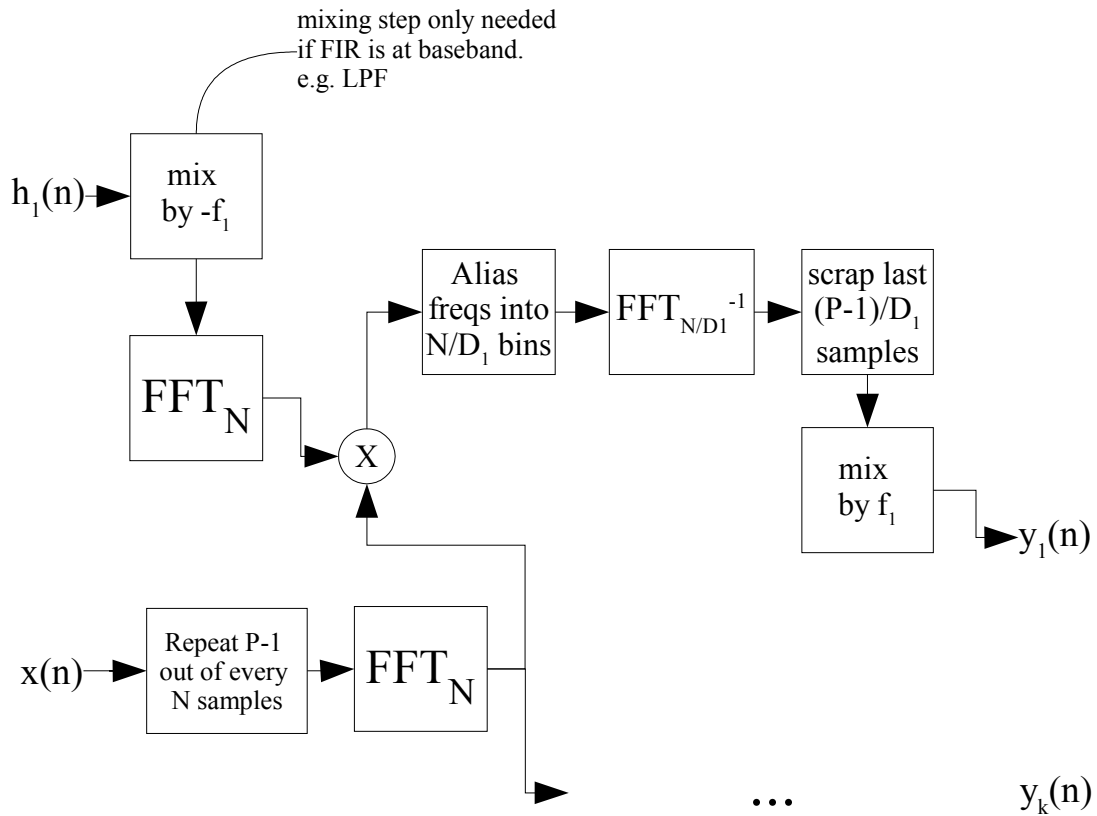


Illustration 6 Overlap-Save Mixing, Decimating Filter Bank

Further Speedups

Highly decimated channels will generally be filtered by a narrow bandwidth filter. Those frequency bins whose combined power falls below a given threshold may be ignored without adversely affecting output. Skipping the multiplications and additions associated with filtering and aliasing those bins can speed processing at the expense of introducing arbitrarily low error energy.

Some researchers have modeled the error created by forcing **all** frequency bins outside the decimated baseband to zero [Bouch]. This error can be modeled as a cyclostationary process.

The hybrid approach of keeping some of the energy outside the baseband in a subset of aliased frequency bins makes modeling and compensation unnecessary. Fast computation is maintained and an arbitrary noise floor may be specified.

Future Direction

How would the above filter bank benefit from multirate/multistage techniques applied to highly decimated channels?

Sources:

[OppSch] Oppenheimer, Alan V.; Schaffer, Ronald W., Discrete-Time Signal Processing . 1989

[RabGold] Rabiner, Lawrence R.; Gold, Bernard, Theory and Application of Digital Signal Processin . 1975

[Orfan] Orfanidis, Sophocles J, Introduction to Signal Processing . 1996

[Lyons] Lyons, Richard G., Understanding Digital Signal Processing . 2004

[Bouch] Boucheret,M; Mortensen,I; Favaro, H, .IEEE Journal on Selected Areas in Communications Feb 1999

[Frerk] Frerking, Marvin, Digital Signal Processing in Communication Systems . 1994

[CroRab] Crochiere, Ronald; Rabiner, Lawrence. Multirate Digital Signal Processing. 1983