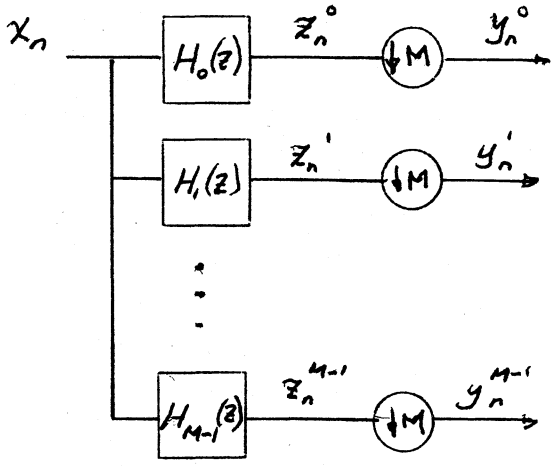
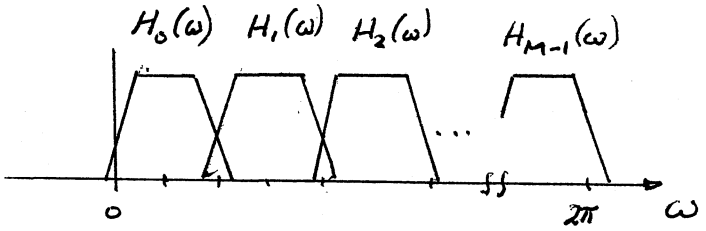
Alternate Derivation of Decimated Filter Bank

Stephen Wilson, with help from Yibin Zheng, 1/9/2003

1 Description of Filter Bank

This note provides a time-domain derivation of the architecture for a decimated uniform filter bank, under study by NRAO. The starting objective is to implement the parallel M-channel filterbank shown below, then decimate the outputs by a factor of M. It is assumed that the bandpass filters form a uniform bandpass response, with centers of the band shifted by $2\pi/M$, where M is the number of filters in the filterbank. In other words the frequency responses appear as shown below





The response $H_0(z)$ is an N-stage FIR filter called the prototype filter, and its impulse response is denoted $\mathbf{h^0} = (h_0^0, h_1^0, ...h_{N-1}^0)$. By the frequency shifting property, all other filter responses are derived from it by

$$h_m^k = h_m^0 e^{j2\pi km/M} \tag{1}$$

Note the impulse responses will be complex in general.

We have that the filter outputs, prior to decimation, are given by

$$z_l^k = h_l^k * x_l \tag{2}$$

where k is a channel index, $k = 0, 1, ...M-1$, and l is a time index. By the impulse response

relationship above we have

$$z_l^k = \sum_{m=0}^{N-1} h_m^0 e^{j2\pi mk/M} x_{l-m} \tag{3}$$

which is the convolution relation involving h_l^k.

Next, consider the M-fold decimation of the parallel sequences. This decimation may introduce aliasing if we were to try to reconstruct the original sequence, although perfect reconstruction filterbanks can be designed to cancel this aliasing if this were the objective. Here, this is not apparently a primary need. Decimation gives

$$y_n^k = z_{Mn}^k = \sum_m h_m^0 e^{j2\pi mk/M} x_{Mn-m} \tag{4}$$

The direct computation using this formula, i.e. doing the convolution over N terms with the modulated impulse response, requires performing N multiply/adds at the input sampling rate, using periodically-loaded filter coefficients, and demultiplexing the outputs to produce the parallel decimated streams. This can be done more efficiently, and avoids need to cyclically load filter coefficients.

2 Polyphase Architecture

To see the better implementation, we break the convolution for y_n^k into M sums using the polyphase decompostion:

$$\begin{aligned} y_n^k &= h_0^0 e^{j0} x_{Mn} + h_M^0 e^{j0} x_{Mn-M} + ... \\ &+ h_1^0 e^{j2\pi k/M} x_{Mn-1} + h_{M+1}^0 e^{j2\pi k/M} x_{Mn-M-1} + ... \\ &+ .... \\ &+ h_{M-1}^0 e^{j2\pi(M-1)k/M} x_{Mn-(M-1)} + h_{2M-1}^0 e^{j2\pi(M-1)k/M} x_{Mn-(M-1)-M} + ... \end{aligned} \tag{5}$$

(Here we have invoked the result that $e^{j2\pi mM/M} = 1$.)

Now define M polyphase filter impulse responses as follows:

$$\begin{aligned} \mathbf{p^0} &= (h_0^0, h_M^0, h_{2M}^0, ...) \\ \mathbf{p^1} &= (h_1^0, h_{M+1}^0, h_{2M+1}^0, ...) \\ &... \\ \mathbf{p^{M-1}} &= (h_{M-1}^0, h_{2M-1}^0, h_{3M-1}^0, ...) \end{aligned} \tag{6}$$

Each of these is a (complex) response with N/M coefficients (we assume that N is a multiple of M.)

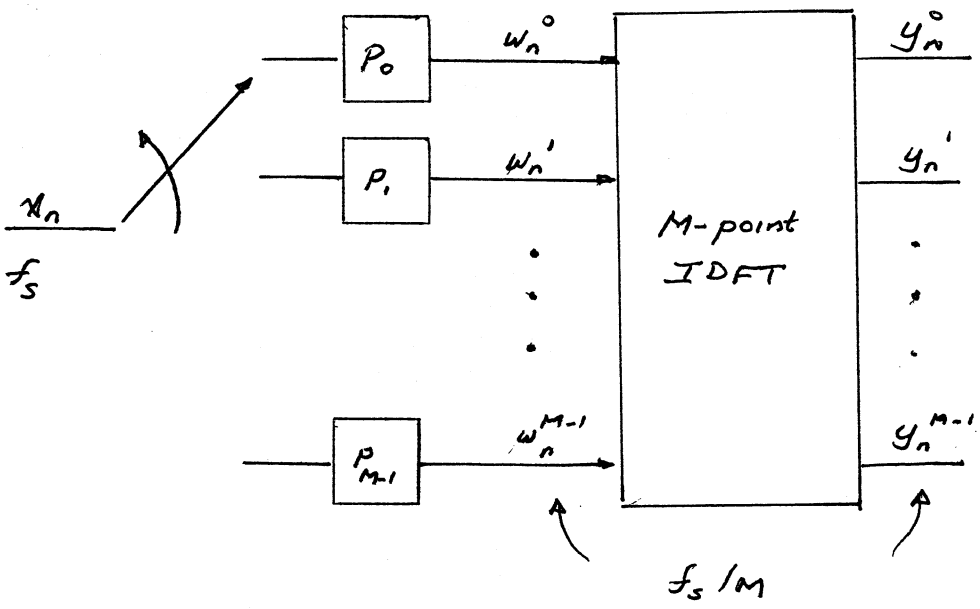
Then we can write

$$y_n^k = [\sum_{l=0}^{N/M-1} p_l^0 x_{M(n-l)}] e^{j0}$$

$$
\begin{aligned}
&+ \quad [\sum_{l=0}^{N/M-1} p_l^1 x_{M(n-l)-1}] e^{j2\pi k/M} \\
&+ \ldots \quad [\sum_{l=0}^{N/M-1} p_l^{M-1} x_{M(n-l)-M-1}] e^{j2\pi k(M-1)/M} \\
&= \quad w_n^0 e^{j0} + w_n^1 e^{j2\pi k/M} + \ldots + w_n^{M-1} e^{j2\pi k(M-1)/M} \qquad (7)
\end{aligned}
$$

where we have defined w_n^k as the output of the kth polyphase channel at time n.

The last expression can be recognized as the kth coefficient of the inverse DFT (IDFT) of the vector $\mathbf{w}_n = (w_n^0, w_n^1, \ldots w_n^{M-1})$. This M-point IDFT must be computed at the subsampled rate, i.e. f_s/M, and at each time instant, the IDFT produces the M parallel outputs required by the filter bank. A block diagram of this process is shown below, where the direction of the multiplexer is crucial as well as the definition of the polyphase coefficients above.



Polyphase Filter Bank, version 1

An alternate realization can be obtained by performing the polyphase decomposition in the "reverse" direction, defining

$$
\begin{aligned}
\mathbf{e^0} &= (h_0, h_M, h_{2M}, \ldots) \\
\mathbf{e^1} &= (h_{-1}, h_{M-1}, h_{2M-1}, \ldots) \\
&\ldots \\
\mathbf{e^{M-1}} &= (h_{-(M-1)}, h_1, h_{M+1}, \ldots)
\end{aligned} \qquad (8)
$$

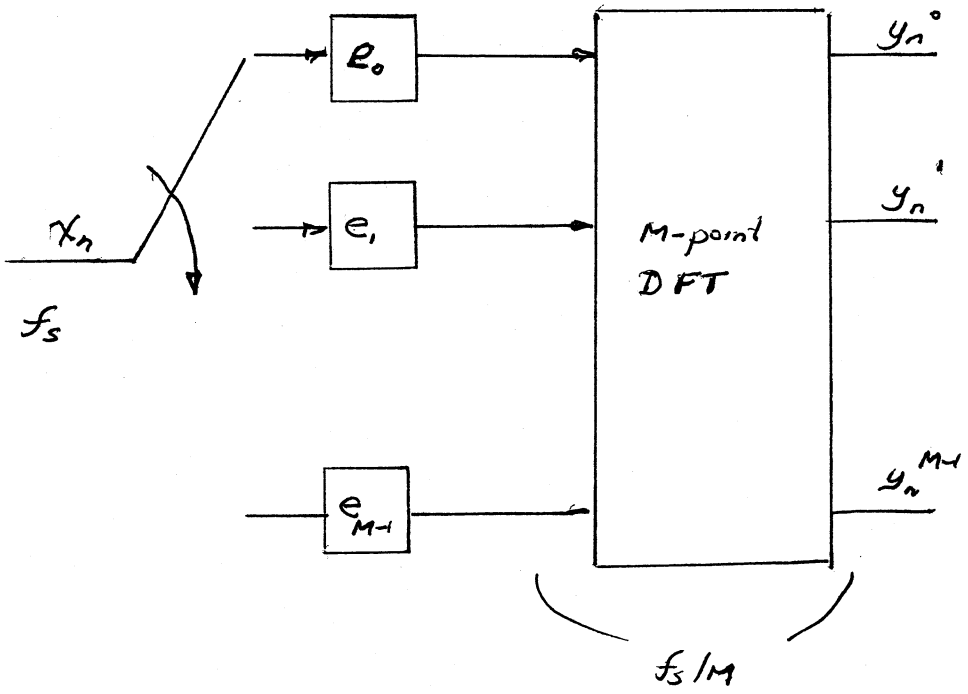
By change of variables we can show that an equivalent realization combines a multiplexer sweeping in the opposite direction, M polyphase filters, and the forward DFT, as shown below.

3 Computation

In the polyphase implementation, the number of complex multiplications (and additions) is $(N/M)M$ for the parallel filters, and $M(log_2 M)$ for the DFT (realized with the FFT). This is the computation *per decimated time step*, so if f_s is the input sample rate, the total number of complex multiplications and additions per second is roughly $C = (f_s/M)(N + M log_2 M) = f_s(N/M + log_2 M)$. The computation of a *single* decimated filter realized with polyphase methods is $f_s(N/M)$ so when the prototype filter is long and the number of channels is not too large, one can say the decimating filter bank can be realized almost as economically as a decimated single filter.

4 Design

The design of the filter bank revolves around choice of N, M and the prototype filter. The standard design would have M a multiple of 2, perhaps up to 32, and N always a multiple of M, say 1024 for filter sharpness. The prototype filter should ba a reasonably sharp FIR filter (for linear phase) with flat response on the interval $[0, f_s/M]$. The N coefficients would be stored in ROM, and the polyphase decomposition of the prototype is easily performed for different M, and loaded into the filter hardware. The required filter sharpness as it relates to aliasing issues needs further discussion. Also, Y. Zheng has commented elsewhere on realizing the filter bank with real filter coefficients. As always, computational tricks exist for efficient DFT computation, particularly for small block sizes anticipated here.



Polyphase Filter Bank, version 2