1 Polyphase filter bank : Maximal Decimation

Consider the following operation on the sequence $x[n]$,

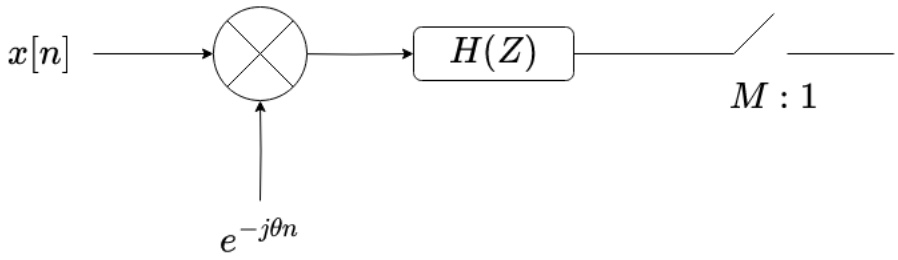


Figure 1: Downconvert, filter and then downsample.

1. The frequency band of the input centered at θ is first downconverted to baseband (this is signal speak for shifting the entire frequency spectrum of $x[n]$ to the left by θ).

2. Then, the downconverted input is passed through a filter $H(Z)$ a notation which simply represents

$$H(Z) = \sum_{n=0}^{\infty} h_n z^{-n}.$$

   In this notation, we multiply the z-transform of $x[n]$ by $H(Z)$ which in time domain simply implies convolution of $x[n]$ by the filter coefficients h_n.

3. Finally, the output of the filter is downsampled/subsampled by a factor of M. This is signalspeak for only keeping every M-th element in the input. Note that the filter is needed, otherwise, a naive subsampling causes aliasing effects in the spectral domain.

This downconvert, filter and then subsample pattern is routinely performed in the case of analyzing isolated frequency bands of the input signals. Therefore,

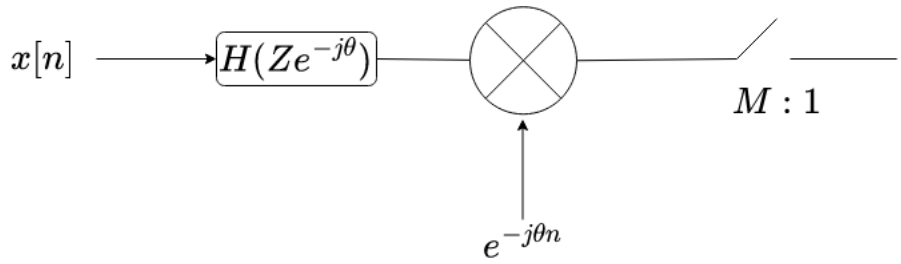


Figure 2: Equivalence theorem

it is of utmost improtance that this be done quickly. We explore the technique of polyphase decomposition of $H(Z)$ to do this. We first note the following:

$$\sum_{k=-\infty}^{\infty} x[k]e^{-j\theta k}h_{n-k} = e^{-j\theta n}\sum_{k} x[k]h_{n-k}e^{j\theta(n-k)}. \tag{1}$$

This is as if, the filter coefficients h_n have been replaced by $h_n e^{j\theta n}$. This implies the rearrangement shown in the figure in Figure 2. A result that is known as the equivalence theorem. We can then shift the multiplication by the complex sinusoid across the downsampler, and we get this: The final

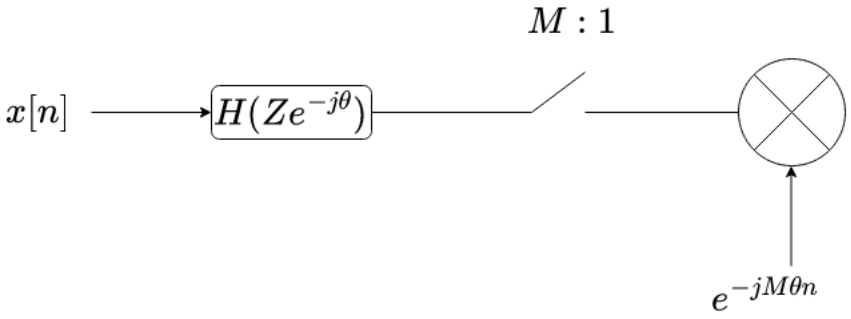


Figure 3: Equivalence theorem with downsample removed

multiplication can be removed with the assumption that

$$M\theta = 2\pi l \tag{2}$$

so that we are only interested in center frequencies that are integer multiples of the downsampling rate. Thus, we need to analyze the situation shown in Figure 4. We next note that as written here, we are computing the convolu-

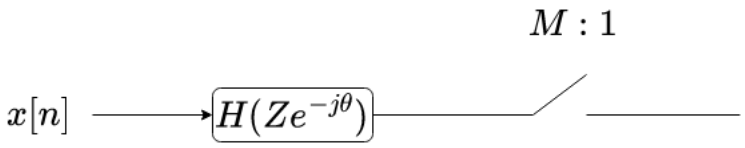


Figure 4: Equivalence theorem with downsample removed

tion of the filter with the input at the full rate, i.e., we are computing all the values of the output, but discarding every M-th value. This is wasteful, and the Polyphase Channelizer algorithm is designed to remedy this. Instead of thinking about delays and polyphase components etc., let us directly compute every M-th output of the filtering operation, and see if we observe any patterns. The Z-transforms of the input $x[n]$ (which we assume starts at time instant $n = 1$) is simply

$$X(z) = x_1 z^{-1} + x_2 z^{-2} + \cdots \tag{3}$$

while the filter Z-transform is (note that we are taking downconversion into account from the get-go, in the following, note $\xi = e^{j\theta}$)

$$H(Ze^{-j\theta}) = h_0 + h_1 \xi z^{-1} + h_2 \xi^2 z^{-2} + \cdots$$

Now, simply multiply $H(Z\xi^{-1})$ with $X(Z)$ to obtain

$$F(Z,\xi) = \left(x_1 Z^{-1} + x_2 Z^{-2} + \cdots \right) \cdot \left(h_0 + h_1 \xi Z^{-1} + h_2 \xi^2 Z^{-2} + \cdots \right). \tag{4}$$

Since we are interested in every M-th output of this product, it makes sense to group together all those filter coefficients whose indices leave the same remainder when divided by M. Thus, we decompose the product $F(Z,\xi)$ as

$$F(Z,\xi) = F_0(Z,\xi) + F_1(Z,\xi) + \cdots + F_{M-1}(Z,\xi) \tag{5}$$

Here, the notation $F_j(Z,\xi)$ indicates the portion of F that depends on filter coefficients h_k such that $k = pM + j$ for some p. Note futher that, at the output due to the downsamplig, we are only interested in terms in the above that involve Z^{-lM} for l an integer. We also note in passing $\xi^M = 1$. Here is the structure of the first few $F_j(Z,\xi)[\xi^{Mk}]$ (a notation that means we only keep terms involving ξ^{Mk}),

$$\begin{aligned}
F_0(Z,\xi)[\xi^{Mk}] &= (h_0 x_M)Z^{-M} + (h_M x_M + h_0 x_{2M})Z^{-2M} \\
&\quad + (h_0 x_{3M} + h_M x_{2M} + h_{2M} x_M)z^{-3M} + \cdots \\
&= G_0(Z) \\
F_1(Z,\xi)[\xi^{Mk}] &= (\xi h_1 x_{M-1})Z^{-M} + \xi(h_{M+1} x_{M-1} + h_1 x_{2M-1})Z^{-2M} \\
&\quad + \xi(h_1 x_{3M-1} + h_{M+1} x_{2M-1} + h_{2M+1} x_{M-1})z^{-3M} + \cdots \\
&= \xi G_1(Z) \\
F_2(Z,\xi)[\xi^{Mk}] &= \xi^2(h_2 x_{M-2})Z^{-M} + \xi^2(h_{M+2} x_{M-2} + h_2 x_{2M-2})Z^{-2M} \\
&\quad + \xi^2(h_2 x_{3M-2} + h_{M+2} x_{2M-2} + h_{2M+2} x_{M-2})z^{-3M}) + \cdots \\
&= \xi^2 G_2(Z)
\end{aligned}$$

We observe,

1. The j-th component F_j depends on ξ^j solely as the factor ξ^j multiplied by a quantity $G_j(Z)$ that does not depend on ξ.

2. To compute $G_j(Z)$, simply take the input samples $x_{M-j}, x_{2M-j}, x_{3M-j}, \cdots$, and perform the convolution of these samples with the filter coefficients $h_j, h_{M+j}, h_{2M+j} \cdots$.

3. Note that before these observations, we were computing convolutions between filter and input at full rate. But now, we only need to perform one convolution computation every M samples of the input. This represents the gains we have made with this observation.

4. Thus, the Z-transform of the output at a given value of the center frequency, set by $\xi = e^{j\theta}$, is simply

$$\sum_{j=0}^{M-1} \xi^j G_j(Z)$$

5. This represents the output at a given value of the center frequency. What if we want all of them? This structure suggests that the answer is to compute $G_j(Z)$ for each j, and expand

$$G_j(Z) = \sum_k f_{jk} Z^{-kM}$$

The actual output as we saw above has the Z-transform $\sum_j \xi^j G_j(Z)$. This suggests that the output time series in each channel is simply given by

$$T_k = \sum_j \xi^j f_{jk}$$

This is nothing but the coloumnwise IFFT of the matrix f_{jk}.

2 Offline Channelizer

This is the computation one needs to perform to get disjoint channels from the input array $x[n]$ given filter coefficients h_n. One first arranges the input array $x[n]$ in the following layout in a buffer array

$$X = \begin{pmatrix} x_M & x_{2M} & x_{3M} & \cdots \\ x_{M-1} & x_{2M-1} & x_{3M-1} & \cdots \\ x_{M-2} & x_{2M-2} & x_{3M-2} & \cdots \\ & \vdots & & \\ x_1 & x_{M+1} & x_{2M+1} & \cdots \end{pmatrix}$$

This step will involve copy of the input into the layout given above. Successive input elements are mapped onto non-contiguous portions of the buffer array in the above layout, and therefore, this step is a time consuming one. Filter coefficients will be stored in the polyphase layout

$$H = \begin{pmatrix} h_0 & h_M & h_{2M} & \cdots \\ h_1 & h_{M+1} & h_{2M+1} & \cdots \\ h_2 & h_{M+2} & h_{2M+2} & \cdots \\ & \vdots & & \\ h_{M-1} & h_{2M-1} & h_{3M-1} & \cdots \end{pmatrix}$$

The output is computed simply as

$$\text{Column-wise IFFT}\Big(\text{Row-wise Convolve}\Big(X, H\Big)\Big)$$

For the off-line version, we use FFT based convolution algorithms as they would be faster than multiplcation based convolution.

3 Online Channelizer

One can also construct a streaming channelizer by a simple modification of the algorithm above. By a streaming channelizer, we mean that the output in each channel is computed one sample at a time. Since there are M channels, the channelizer has to wait for M input samples to compute one output sample in each channel. This is simply a consequence of the down-sampling operation. (One can infact start computing new outputs given $M/2$ samples actually, but we don't do that micro-optimization here). Call the maximum number of (non-zero) filter coefficients per channel as the number of taps per channel (denoted as T). Then, one can construct a streaming channelizer by maintaining M circular buffers of size T each, one for each polyphase component of the filter. The first sample in each batch of M new samples is sent to the last polyphase component, the second to the second-last polyphase component, and so on. The insertion of these new samples updates the buffers. Then, compute the convolution of each buffer with the corresponding polyphase filter component by simple addition and multiplication. This would be a number for each channel. Collect them in a vector and then take the IFFT of the resulting vector. That would be the updated output from the channelizer given the M inputs.

4 Non-maximal decimating Channelizer

In the previous version of the channelizer, the bandwidth of each channel was the same as the spacing between channels. This is only good for analyzing static bands where we know the activity in advance. For bands with dynamic activity, we would like to combine multiple channels together. For this to happen, the filter bank needs to be a near perfect reconstruction filter bank, and this is only possible is channel spacing is smaller than the bandwidth of each channel.

An easy way to get this, is to keep M channels at the output, but increasing the sample rate of each channel by a factor of two, as in the diagram below,

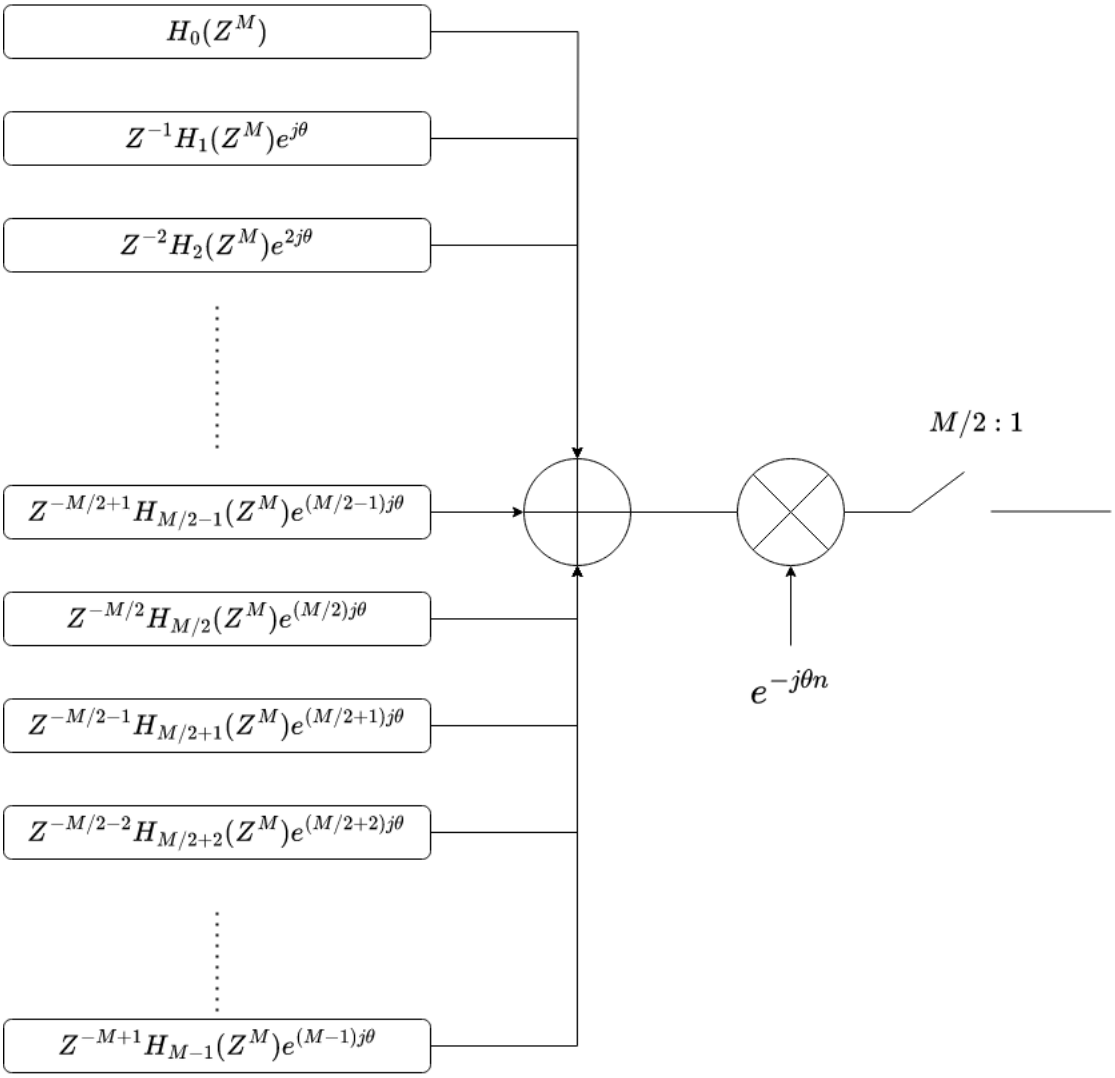


Figure 5: Non-maximal decimated channelizer construction

Note that here we have written down the structure we found in the previous section for the maximally decimated channelizer. We push the downsampler to the left, and use the Noble identity (the previous section can be

thought of as roughly a derivation of the Noble identity for downsampling), and get the following arrangement. The delay and downsample on the left

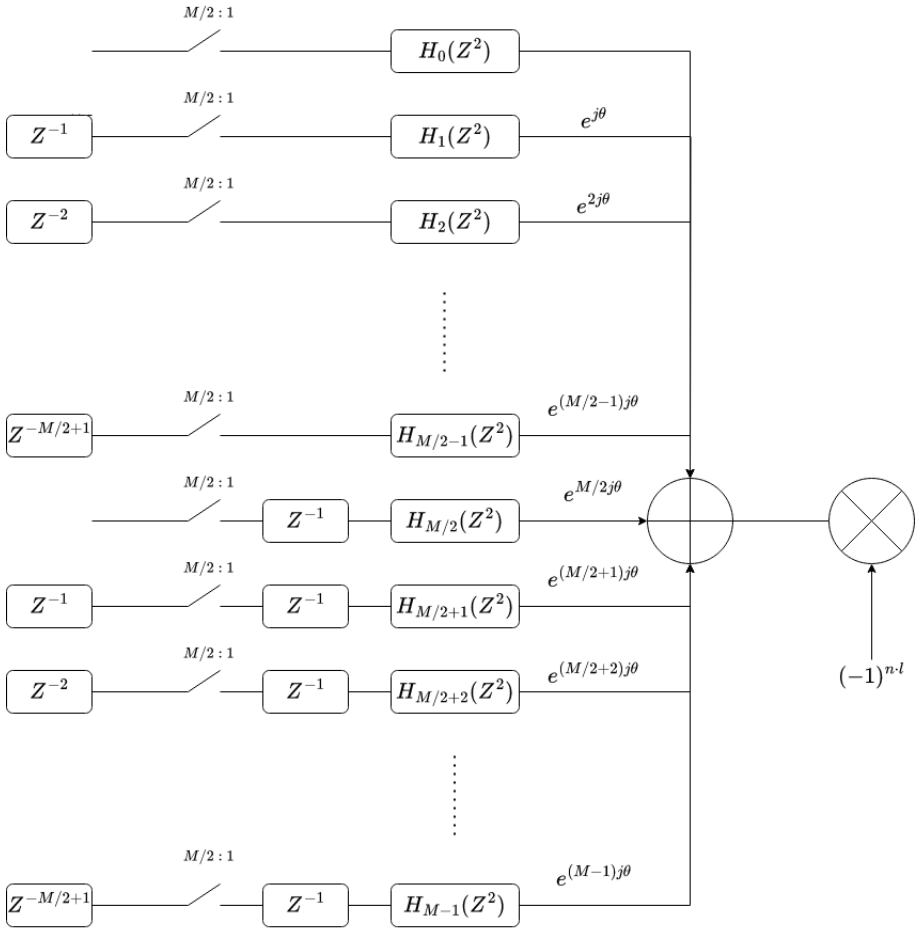


Figure 6: Non-maximal decimated channelizer construction. Note that $\theta = \frac{2\pi l}{M}$, i.e., the center frequency of the output channel is an integer multiple of the downsampled rate of the maximally decimated channelizer. Since the downsampling in this new case is half of the maximal possible, an overall multiplication by $(-1)^n$ remains for odd channels.

side of the figure mean that the input $X = x_0, x_1, \cdots$, needs to be arranged

in the layout

$$X = \begin{pmatrix} x_{M/2} & x_M & x_{3M/2} & \cdots \\ x_{M/2-1} & x_{M-1} & x_{3M/2-1} & \cdots \\ x_{M/2-2} & x_{M-2} & x_{3M/2-2} & \cdots \\ & \vdots & & \\ x_2 & x_{M/2+2} & x_{M+2} & \cdots \\ x_1 & x_{M/2+1} & x_{M+1} & \cdots \\ x_{M/2} & x_M & x_{3M/2} & \cdots \\ x_{M/2-1} & x_{M-1} & x_{3M/2-1} & \cdots \\ x_{M/2-2} & x_{M-2} & x_{3M/2-2} & \cdots \\ & \vdots & & \\ x_2 & x_{M/2+2} & x_{M+2} & \cdots \\ x_1 & x_{M/2+1} & x_{M+1} & \cdots \end{pmatrix}$$

The filter coefficients are in the layout (the zeros in the first $M/2$ rows correspond to taking the filters $H_k(z^2)$), whereas, the delays in the last $M/2$ rows are due to the additional Z^{-1} delay in the last $M/2$ rows with the filters in the diagram above:

$$H = \begin{pmatrix} h_0 & 0 & h_M & 0 & h_{2M} & \cdots & \\ h_1 & 0 & h_{M+1} & 0 & h_{2M+1} & \cdots & \\ h_2 & 0 & h_{M+2} & 0 & h_{2M+2} & \cdots & \\ & \vdots & & & & & \\ h_{M/2-1} & 0 & h_{3M/2-1} & 0 & h_{5M/2-1} & \cdots & \\ 0 & h_{M/2} & 0 & h_{3M/2} & 0 & h_{5M/2} & \cdots \\ 0 & h_{M/2+1} & 0 & h_{3M/2+1} & 0 & h_{5M/2+1} & \cdots \\ 0 & h_{M/2+2} & 0 & h_{3M/2+2} & 0 & h_{5M/2+2} & \cdots \\ & \vdots & & & & & \\ 0 & h_{M-1} & 0 & h_{2M-1} & 0 & h_{3M-1} & \cdots \end{pmatrix}$$

Finally, define the matrix

$$T = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & \cdots \\ 1 & -1 & 1 & -1 & 1 & \cdots \\ 1 & 1 & 1 & 1 & 1 & \cdots \\ 1 & -1 & 1 & -1 & 1 & \cdots \\ & \vdots & & & & \end{pmatrix}$$

Then, the output is computed as

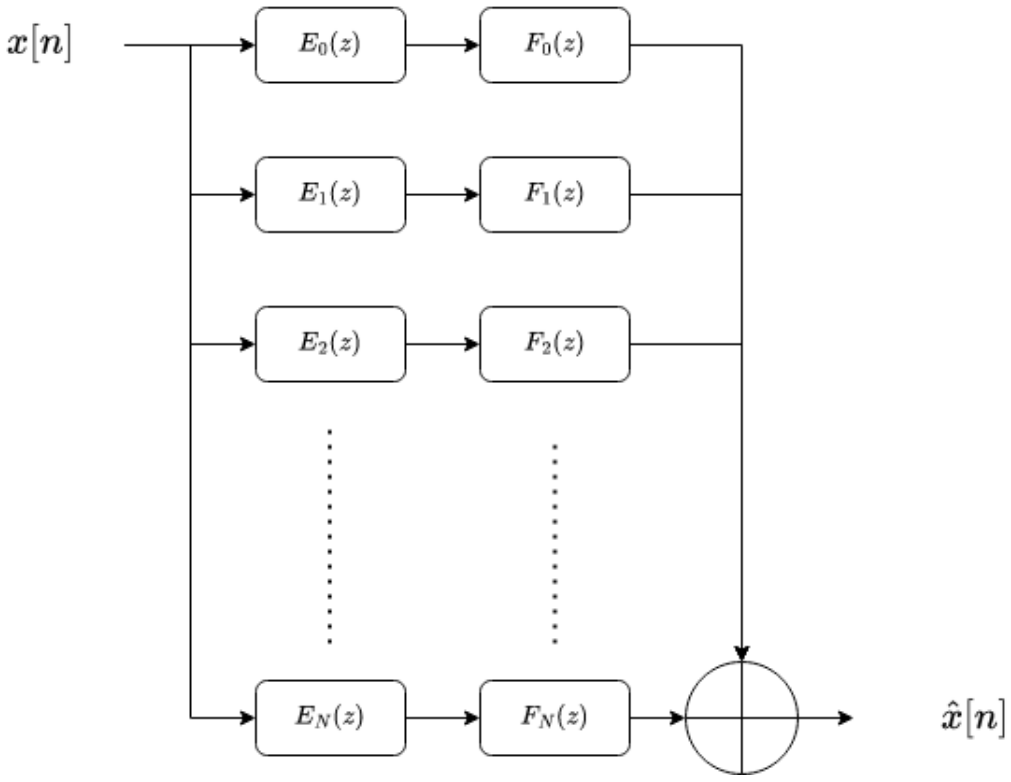
$$Y = T \odot \text{ColumnWise IFFT}(\text{Row-wise Convolve}(X, H)) \tag{6}$$

where $\odot$ represents Hadamard elementwise multiplication. The output Y contains M channels, and each channel is sampled at $\frac{2f_s}{M}$ sample rate, with the spacing between the samples $\frac{f_s}{M}$ if the input sample rate is f_s.

5 IQ Reverts

5.1 General reconstruction problem

Consider the following set of filters acting on an input sequence $x[n]$. Note



that there are no delays/downsamplers in this construction yet. The input is fed to a series of filters $E_j(z)$, and the output of these filters is fed to a

different set of filters $F_j(z)$. The z-transform of the output $\hat{x}[n]$ is

$$\hat{X}(z) = \sum_{k=0}^{N} F_k(z) E_k(z) X(z) \tag{7}$$

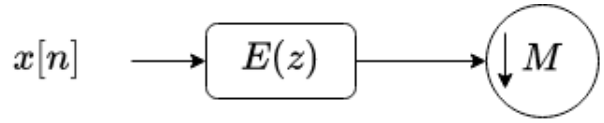
If, we find that

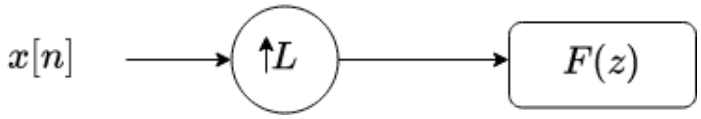
$$\hat{X}(z) = cz^{-L} X(z), \tag{8}$$

for some c and some delay L, then, this set of filters is said to perform perfect reconstruction. We need to study briefly the kind of filter combinations that can help with this. Reference is the textbook by PP Vaidyanathan for this.

5.2 Nyquist(L) filters

Recall the following two protypical cases of filter application





In the first case, we apply an anti-aliasing filter $E(z)$ (to remove aliases due to decimation) and then perform decimation. In the second case, we perform interpolation and then apply a filter $F(z)$ (to remove imaging artifacts due to interpolation).

Note that the output in the second case is simply

$$Y(z) = F(z) X(z^L) \tag{9}$$

Next, decompose the filter $F(z)$ in polyphase fashion

$$F(z) = F_0(z^L) + \sum_{j=1}^{L-1} z^{-j} F_j(z^L) \tag{10}$$

If we arrange that the zeroth polyphase component $F_0(z)$ is a constant, we get the output

$$Y(z) = cX(z^L) + \sum_{j=1}^{L-1} z^{-j} F_j(z^L) X(z^L) \tag{11}$$

The interpretation of this is that while the filter $F(z)$ inserts additional samples, the original signal x is communicated without distortion at every L-th sample of output. The constant zeroth polyphase condition can be written in time domain as

$$f(nL) = \begin{cases} c, & n = 0 \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

Such filters are called Nyquist(L) filters. More generally, we say a filter $F(z)$ is Nyquist(L), if any one of its L polyphase components satisfies

$$F_k(z) = cz^{-n_k} \tag{13}$$

which translates in time domain to

$$f(Ln + k) = \begin{cases} c, & n = n_k \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

Again, if the filter $F(z)$ satisfies the Nyquist filter criterion in its zeroth polyphase component, i.e., $F_0(z^L) = cz^{-kL}$, then we can also write the following equation

$$\sum_{j=0}^{L} F(zW^j) = cL, \qquad W = e^{-2\pi j/L} \tag{15}$$

5.3 Power complementarity

A set of filters $H_k(z)$ is said to be power complementary if

$$\sum_{k=0}^{M-1} |H_k(e^{j\omega})|^2 = c \qquad \forall \omega \tag{16}$$

If $h[n]$ is the impulse response of $H(z)$, we define the paraconjugate $\tilde{H}(z)$ as

$$\tilde{H}(z) = H_*(z^{-1}) = \sum_n h[n]^* z^n = z^N \sum_n h[N-n]^* z^{-n} \tag{17}$$

Then, by analytic continuation, the condition of power complementarity is written as

$$\sum_k H_k(z)\tilde{H}_k(z) = c \qquad \forall z \tag{18}$$

5.4 Perfect reconstruction and Power complementarity

If, in the figure in Section 3.1, the filters $F_j(z)$ are chosen to be $\tilde{E}_j(z)$, then we get

$$\tilde{X}(z) = \sum_k \tilde{E}_k(z)E_k(z)X(z) = cX(z) \tag{19}$$

and we get reconstruction of the input. Note further that $\tilde{E}_j(z)$ are not causal, therefore, we introduce a delay and choose

$$F_j(z) = z^{-N}\tilde{E}_j(z) \tag{20}$$

and get

$$\tilde{X}(z) = cz^{-N}X(z) \tag{21}$$

This is perfect reconstruction.

Next, there is a simple relationship betweek the notions of power complementarity, polyphase decomposition, and Nyquist filters. And it is as follows. Consider the filter $H(z)$. Define the derived filter $G(z) = \tilde{H}(z)H(z)$. Further, suppose

$$H(z) = \sum_{j=0}^{N-1} z^{-j}E_j(z^N) \tag{22}$$

are the polyphase components of the filter $H(z)$. Then,

the set of filters $[E_0(z), \cdots E_{N-1}(z)]$ are power complementary (23)
if and only if $G(z)$ is a Nyquist N-filter. (24)

Before proving this, an aside on notation. If we define the vector $V(z)$ as

$$\mathbf{V}(z) = \begin{pmatrix} V_0(z) \\ V_1(z) \\ \vdots \\ V_{N-1}(z) \end{pmatrix} \tag{25}$$

then, we define

$$\tilde{\mathbf{V}}(z) = V_*^T(z^{-1}) = \left(\tilde{V}_0(z), \cdots, \tilde{V}_{N-1}(z)\right) \tag{26}$$

Define the family of filters

$$H_n(z) = H(zW^{-n}) = \sum_{k=0}^{N-1} z^{-k} W^{-nk} E_k(z^N) \tag{27}$$

Note that we can write

$$\mathbf{H}(z) = \begin{pmatrix} H_0(z) \\ H_1(z) \\ \vdots \\ H_{N-1}(z) \end{pmatrix} = \mathbf{W} \begin{pmatrix} 1 & 0 & \cdots 0 \\ 0 & z^{-1} & \cdots 0 \\ \vdots & & \\ 0 & 0 & \cdots z^{1-N} \end{pmatrix} \begin{pmatrix} E_0(z^N) \\ E_1(z^N) \\ \vdots \\ E_{N-1}(z^N) \end{pmatrix} \tag{28}$$

The definitions mean that

$$\tilde{\mathbf{H}}(z)\mathbf{H}(z) = \sum_{j=0}^{N-1} \tilde{E}_j(z^N) E_j(z^N) = c \tag{29}$$

Here we have used the fact that the $E_j(z)$ are power complementary, that $\mathbf{W}$ is unitary, as also the fact that

$$\mathbf{\Lambda}(z) = \begin{pmatrix} 1 & 0 & \cdots 0 \\ 0 & z^{-1} & \cdots 0 \\ \vdots & & \\ 0 & 0 & \cdots z^{1-N} \end{pmatrix} \tag{30}$$

$$\tilde{\mathbf{\Lambda}}(z)\mathbf{\Lambda}(z) = 1 \tag{31}$$

This shows that $H_n(z)$ are power complementary. Finally this means

$$\sum_{k=0}^{N-1} G(zW^{-k}) = c \tag{32}$$

so that $G(z)$ is a Nyquist N filter. The converse can be proven the same way.

5.5 Nyquist vs Root-Nyquist Filters

Suppose $H(z)$ is a Nyquist-M filter whose impulse response is purely real, so that its polyphase decomposition looks like

$$H(z) = c + \sum_{j=1}^{M-1} z^{-j} E_j(z^{-M}) \tag{33}$$

Then, look at the paraconjugate $\tilde{H}(z)$,

$$\tilde{H}(z) = c + \sum_{j=1}^{M-1} z^{j} E_j(z^{M}) \tag{34}$$

Next, note that

$$E_j(z^M) = z^{(t-1)M} F_j(z^{-M}) \tag{35}$$

where we have assumed the number of filter coefficients per channel is t. The F_j are causal filters. Thus, we get

$$\tilde{H}(z) = z^{tM-1}\left(cz^{-tM+1} + \sum_{j=0}^{M-2} z^{-j} F_{M-1-j}(z^{-M})\right) \tag{36}$$

$$= z^{tM-1}\left(\sum_{j=0}^{M-1} z^{-j} \tilde{E}_j(z^{-M})\right) \tag{37}$$

where we have

$$\tilde{E}_j(z^{-M}) = F_{M-1-j}(z^{-M}) \qquad 0 \le j \le M-2 \tag{38}$$

$$\tilde{E}_{M-1}(z^{-M}) = cz^{-(t-1)M} \tag{39}$$

This shows that if $H(z)$ is a Nyquist filter, then $\tilde{H}(z)$ need not be Nyquist in general. Perfect reconstruction with polyphase channelizer therefore requires a root Nyquist filter, in which case the filter's polyphase components will be power complementary as proven in the previous section.

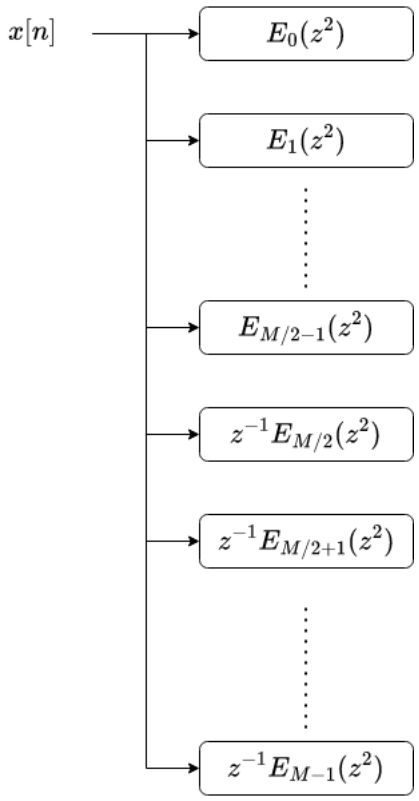


Figure 7: Forward Non-maximal decimating channelizer without delays, downsampling and downconversion. Just the layout of filters is shown here.

6 Reconstruction in the non-maximal channelizer case

6.1 Filters

Suppose $H(z)$ is a filter that satisfies the conditions of the previous section, so that the polyphase component filters for this are power complementary. Question is, if this filter is arranged in the layout of non-maximal decimation, does the resulting filter still allow reconstruction? If so, what is the structure of the synthesis filter? Recall that if the delays, dowsamplings and downconversions have been stripped off, we get the arrangement of filters in Fig. 7, which can be written in matrix form as

$$\mathbf{H}(z) = \begin{pmatrix} \mathbb{I}_{M/2\times M/2} & 0 \\ 0 & z^{-1}\mathbb{I}_{M/2\times M/2} \end{pmatrix} \begin{pmatrix} E(z^2) \\ F(z^2) \end{pmatrix} \tag{40}$$

where we have defined

$$E(z^2) = (E_0(z^2), \cdots, E_{M/2-1}(z^2))^T \tag{41}$$
$$F(z^2) = (E_{M/2}(z^2), \cdots, E_{M-1}(z^2))^T \tag{42}$$

Its then easy to see

$$\tilde{\mathbf{H}}(z)\mathbf{H}(z) = \tilde{E}(z^2)E(z^2) + \tilde{F}(z^2)F(z^2) = c \tag{43}$$

This shows that the set of filters

$$E_0(z^2), \cdots, E_{M/2-1}(z^2), z^{-1}E_{M/2}(z^2), \cdots, z^{-1}E_{M-1}(z^2) \tag{44}$$

is power complementary. The synthesis filters corresponding to this filter bank are therefore

$$\tilde{E}_0(z^2), \cdots, \tilde{E}_{M/2-1}(z^2), z\tilde{E}_{M/2}(z^2), \cdots, z\tilde{E}_{M-1}(z^2) \tag{45}$$

In matrix form, removing some delay units, if A and S are M-channel and t-tap non-maximally decimating analysis and the corresponding synthesis

banks, then,

$$A = \begin{pmatrix} h_0 & 0 & h_M & 0 & h_{2M} & 0 & \cdots & 0 \\ h_1 & 0 & h_{M+1} & 0 & h_{M+1} & 0 & \cdots & 0 \\ h_2 & 0 & h_{M+2} & 0 & h_{M+2} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \\ h_{M/2-1} & 0 & h_{3M/2-1} & 0 & h_{5M/2-1} & 0 & \cdots & 0 \\ 0 & h_{M/2} & 0 & h_{3M/2} & 0 & h_{5M/2} & \cdots & h_{(t-1)M+M/2} \\ 0 & h_{M/2+1} & 0 & h_{3M/2+1} & 0 & h_{5M/2+1} & \cdots & h_{(t-1)M+M/2+1} \\ 0 & h_{M/2+2} & 0 & h_{3M/2+2} & 0 & h_{5M/2+2} & \cdots & h_{(t-1)M+M/2+2} \\ 0 & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & h_{M-1} & 0 & h_{2M-1} & 0 & h_{3M-1} & \cdots & h_{tM-1} \end{pmatrix}$$

$$S = \begin{pmatrix} 0 & h_{(t-1)M} & 0 & h_{(t-2)M} & \cdots & 0 & h_0 \\ 0 & h_{(t-1)M+1} & 0 & h_{(t-2)M+1} & \cdots & 0 & h_1 \\ 0 & h_{(t-1)M+2} & 0 & h_{(t-2)M+2} & \cdots & 0 & h_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots \\ 0 & h_{(t-1)M+M/2-1} & 0 & h_{(t-2)M+M/2-1} & \cdots & 0 & h_{M/2-1} \\ h_{(t-1)M+M/2} & 0 & h_{(t-2)M+M/2} & 0 & \cdots & h_{M/2} & 0 \\ h_{(t-1)M+M/2+1} & 0 & h_{(t-2)M+M/2+1} & 0 & \cdots & h_{M/2+1} & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ h_{tM-1} & 0 & h_{(t-1)M-1} & \vdots & \vdots & h_0 & 0 \end{pmatrix}$$

6.2 Delay and Downsamplers

Consider the system of delay and downsamplers shown in Fig. 8. It is clear from this diagram that the input in arms j and $j + M/2$ arrive at the output at the same clock cycle and therefore get added together. The total delay in the whole system is $M/2$ units. The reconstructed signal $\hat{x}[n]$ is therefore

$$\hat{x}[n] = 2x[n - M/2] \tag{46}$$

6.3 Efficient IQ revert algorithm

6.4 Filter application

The general IQ revert problem is specified thus: on a large input 2D matrix (which can be thought of as a row-major 1D array), we are given time bounds

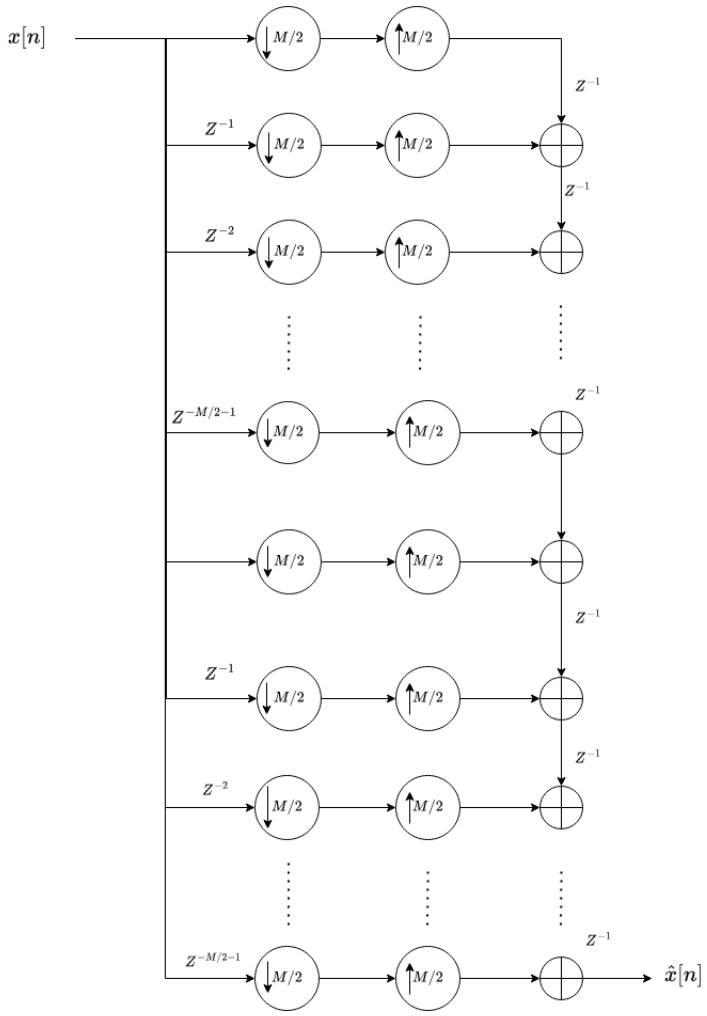


Figure 8: Non-maximal decimation channelizer without the filters. This by itself does perfect reconstruction.

(which are contiguous spans in any one row), and frequency bounds (which span across rows) as boxes. The task is to run the reconstruction algorithm on each such box and get the reverted IQ. This algorithm should conform to CUDA GPU array access patterns. The solution to this problem is as follows:

- Spawn 3D CUDA thread grids, blocks and threads.

- ThreadIdz spans over the channels in each box.
- ThreadIdy spans over the sample time in the output for each box.
- ThreadIdx is used to index into the input in each channel.
- The k-th output element in each channel is given by

$$\sum_j f_{k-j} x_j \tag{47}$$

  Here, f_j are the synthesis filter coefficients, and x is the input. From the theory in the previous sections, given the number of channels that need to be reverted and the forward prototype filters used to channelize, f_{k-j} can be computed during runtime.

- Then, CUDA atomic operations are used to populate the value of this sum at position k in the output. As discussed previously,

$$k = \text{ThreadIdy} + \text{BlockIdy} * \text{DimGrid.y} \tag{48}$$
$$j = \text{ThreadIdx} + \text{BlockIdx} * \text{DimGrid.x} \tag{49}$$

- This does not violate any CUDA access pattern. Therefore, the runtime should be dominated by the kernel launch latency and needs to be optimized.

6.5 Downconversion

For the initial downconversion IFFT, we distinguish between the following two cases

- Revert large number of small boxes
- Revert small number of large boxes

In the first case, we can initialize 1D FFT plans, and run a simple for loop over the box to perform coloumnwise IFFT. This will be fast because boxes are small.

In the second case, we can initialize strided FFT plans, with the size being set by the size of typical large boxes. The upshot is that revert can be done without any runtime allocation of internal memory or fft plan initialization.