

Project 11.7 Running FIR Filter Structure

Abstract:

The computational complexity of running FIR filtering using multirate filter banks is considered. It is presented how to map long running convolutions into smaller ones by using filter banks based on an aperiodic convolution algorithm. With this approach it is possible to achieve good tradeoffs among computational complexity, input-output delay and system architecture.

Introduction:

Computation of FIR (finite impulse response) filters is a really important task in digital signal processing. For example, if the output of the filter is computed for block of samples at once it is possible to reduce computational complexity. However, that also means introducing additional input-output delay of the order of the block size and possibly more complex architecture. Therefore, both the system delay and architectural complexity are increasing function of the block size and the number of operations needed for computing the convolution is a decreasing one. The emphasis of this project is implementing really simple algorithm since the task of speeding up the convolution is explored, with respect to computational complexity, architectural complexity and system delay.

Multirate systems

Processing data blocks that step N samples forward after each computation is a subsampling of N on its internal variables. If system is not an ideal one, aliased versions of the input signal will appear at the output.

A general multirate analysis/synthesis filter bank is shown in Figure 1.

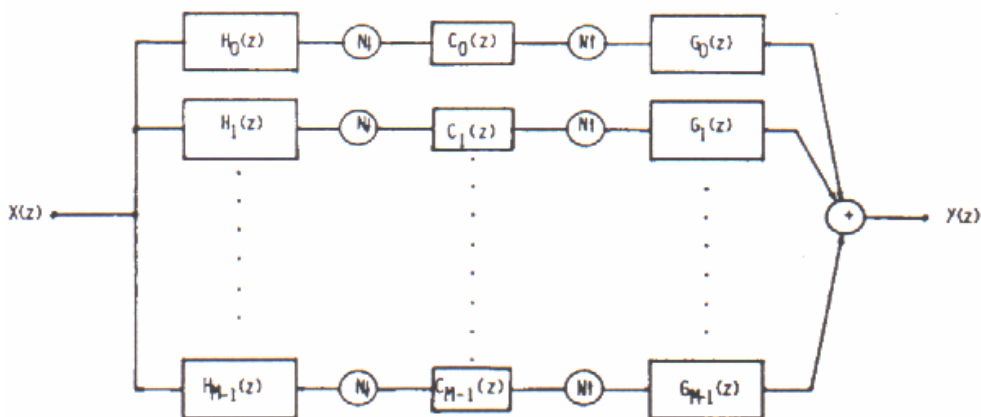


Fig. 1. M channel analysis/synthesis filter bank with subsampling by N and filtering of the channel signals. The subsampling by N is equivalent to moving the input by N samples between successive computations of the output.

Output of the block processing system can be written as:

$$Y(z) = \frac{1}{N} [g(z)^T] \cdot C(z^N) \cdot H_m(z) \cdot x_m(z) \quad (1)$$

where $g(z) = [G_0(z)G_1(z)\dots\dots G_{M-1}(z)]^T$ (2)

is the vector of output filters

$$C(z^N) = \text{diag}[C_0(z^N)C_1(z^N)\dots\dots C_{M-1}(z^N)]^T \quad (3)$$

is a diagonal matrix containing the channel filters

$$H_m(z) = [h(z)h(W_N z)\dots\dots h(W_N^{N-1} z)] \quad (4)$$

is the modulated filter matrix containing the input filters

$$h(z) = [H_0(z)H_1(z)\dots\dots H_{M-1}(z)]^T \quad (5)$$

is the vector of input filters and

$$x_m(z) = [X(z)X(W_N z)\dots\dots X(W_N^{N-1} z)]^T \quad (6)$$

is the vector of modulated input signals

This is a complex characterization of the block processing system.

If all filters in the given system are causal, the block processing system with down/upsampling by N produces a delay of minimum N – 1 samples (if processing time is not taken into account). This is known as *Minimum Delay Property*.

Digital Filter Banks based on Aperiodic Convolution Algorithm

Aperiodic linear or running convolution indicates a noncircular convolution of an infinite signal with a finite or infinite impulse response filter. Efficient aperiodic convolution algorithms are based on a correct way of choosing transfer function of the filters i.e. overall designed system is alias free.

In order to process an infinite signal with a FIR filter of length KN, one can use a filter bank of size M, subsampled by N and have length K channel filters. Length KN convolution is replaced by M-length K convolution at the rate 1/N. If one uses polyphase decomposition i.e.

$$X(z) = \sum_{n=0}^{N-1} z^{-n} \cdot X_n(z^N)$$

$$H(z) = \sum_{n=0}^{N-1} z^{-n} \cdot H_n(z^N)$$

$$Y(z) = \sum_{n=0}^{N-1} z^{-n} \cdot Y_n(z^N)$$

and then evaluate product X(z)H(z) by using the fast aperiodic convolution algorithm, final output is obtained from (8a) and there are only M products to be evaluated.

$$Y_l(z) = \sum_{n=0}^l X_n(z^N) \cdot H_{l-n}(z^N) + \sum_{n=0}^{l+N} X_n(z^N) \cdot H_{l+N-n}(z^N), l = 0, 1, \dots, N-1$$

An optimal aperiodic convolution algorithm for two sequences of length M and N requires $M+N-1$ multiplications but number of additions increases and architecture of the system is more complex one.

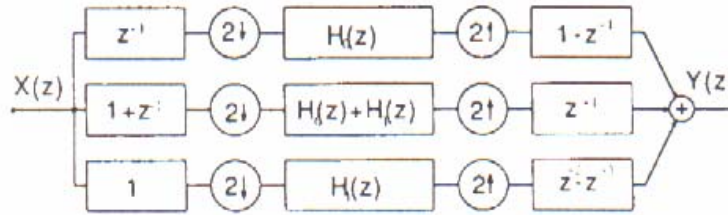


Fig. 2. Fast running convolution based on the two-point aperiodic convolution algorithm. The length- $2K$ running convolution has been replaced by three length- K running convolutions at half speed.

Structure showed in Figure 2 represents a simple three-channel filter bank structure with subsampling by N and with the following filters:

$$h(z) = [z^{-1}, 1 + z^{-1}, 1]^T \quad (7)$$

$$C(z^2) = \text{diag}[H_0(z^2), H_0(z^2) + H_1(z^2), H_1(z^2)]^T \quad (8)$$

$$g(z) = [1 - z^{-1}, z^{-1}, z^{-2} - z^{-1}]^T \quad (9)$$

We're getting overall transfer function: $Y(z) = [z^{-1} \cdot H_0(z^2) + z^{-2} \cdot H_1(z^2)] \cdot X(z)$ (11)

Aliased version of the input (the term with zero transform $X(z)$) has disappeared so alias-free multibank structure is obtained. If a desired filter with z -transform $T(z)$ is given, $H_0(z^2)$ and $H_1(z^2)$ are chosen in the following way by using (11) :

$$\begin{aligned} H_0(z^2) &= 0.5 \cdot (T(z) + T(-z)) \\ H_1(z^2) &= 0.5 \cdot z^* \cdot (T(z) - T(-z)) \end{aligned} \quad (12)$$

This means that impulse response coefficients of $H_0(z)$ are even coefficients of $T(z)$ of length $2K$ and of $H_1(z)$ are the odd ones. We're getting two filters of length K each.

Implementation and Comparison of the proposed algorithms

Transfer function $T(z)$ of even length is given. Implementing the structure shown in Figure 3.

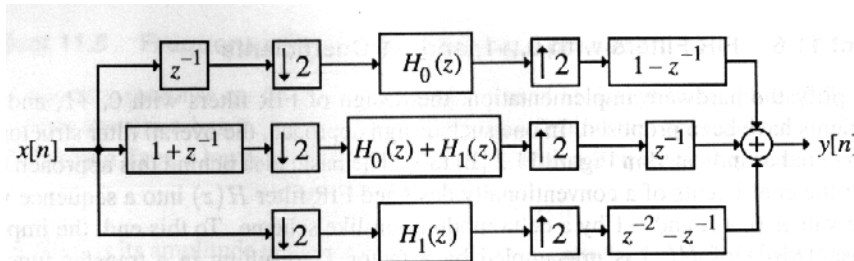


Figure 11.3 A fast running convolution structure.

We're getting alias free realization i.e. aliased version of the input (the term with zero transform $X(z)$) has disappeared), so alias free multibank structure is obtained. Then, $H_0(z^2)$ and $H_1(z^2)$ are computed in the following way, using (12): impulse response coefficients of $H_0(z)$ are even coefficients of $T(z)$ of length $2K$ and of $H_1(z)$ are the odd ones.

We're getting two filters of length K each. Since $Y(z) = z^{-1}T(z)X(z)$, at the cost of one delay, the filtering by $T(z)$ has been replaced by three filterings of half length and at half speed – gain of 25% in the number of multipliers per output sample is achieved. Input filter bank uses one addition per two input samples (downsampling by 2) and the output filter bank uses three additions per two output samples. We can apply this method again to each of the three subfilters if the delay is tolerable. Results are shown in the following table:

TABLE I
COMPUTATION OF THE RUNNING CONVOLUTION WHEN THE FILTER IS A 32-
POINT FIR FILTER. COMPARISON OF THE VARIOUS BREAKUPS INTO SMALLER
FILTERS

Method	Subsampling	Delay	Mult./ Point	Add./ Point
1 32-pt. FIR filter	1	0	32	31
3 16-pt. FIR filters	2	1	24	24.5
9 8-pt. FIR filters	4	3	18	17.5
27 4-pt. FIR filters	8	7	13.5	19.6
81 2-pt. FIR filters	16	15	10.125	21.3
243 1-pt. multiplic.	32	31	7.59	26.4

This method doesn't take advantage of the filter symmetry if original filter is symmetric or antisymmetric.

Structure given in Figure 3 was simulated using MATLAB and correctness of the simulation was verified by using function Structure.m:

```
% Running FIR filter structure - verification of the
structure
format long;
t = input('FIR filter coefficients in ascending order of z
= ');
disp('FIR coefficients:');disp(t);
N = length(t);
x = [1 zeros(1, N-1)];
y = structure(t,x);
disp('Actual FIR coefficients:');disp(y);
```

```

function y = structure(t,x);
% Y = STRUCTURE(T,X) filters input data vector X with
% the FIR filter described by vector P to create the
% filtered data Y. The filter is a computationally
% efficient implementation using multirate techniques.
format long;
k = length(t);
N = length(t);
% Obtaining the two transfer functions
h0 = t(1:2:N-1);
h1 = t(2:2:N);
%first part
a = filter(h0,[1],[0 x(2:2:length(x))]);
y = zeros(1, 2*length(a));
y(1:2:length(y)) = a;
yb = [y 0] - [0 y];

%second part
y = [x 0] + [0 x];
a = filter(h0+h1,[1],y(1:2:length(y)));
y = zeros(1, 2*length(a));
y(1:2:length(y)) = a;
yb = yb + [0 y];

%third part
a = filter(h1,[1], x(1:2:length(x)));
y = zeros(1, 2*length(a));
y(1:2:length(y)) = a;
yb = yb + [0 0 y 0] - [0 y 0 0];
y = yb(2:(length(x)+1));
return;

```

Results – filter is a 32 point FIR filter with randomly chosen coefficients.

```

>>FIR filter coefficients :
7      4      5      0      6      4      8      8      4      3
3     10      5      4      8      4      9      2      4      9
5      1      7      0      6     10      4      7      7     10
2      6

```

```

Actual FIR coefficients:
7      4      5      0      6      4      8      8      4      3
3     10      5      4      8      4      9      2      4      9
5      1      7      0      6     10      4      7      7     10
2      6

```

Further Research

Further research on FIR running structures should go into two directions.

Roundoff error in proposed block processing could lead to aliasing effects because the system is time varying. This problem might lead to tighter precision requirements. Simple example of aliasing effects due to roundoff is presented. Assume that roundoff errors can be modeled as additional noise on the channels of the system. Then, the diagonal matrix containing the channel filters will be (from (8)): $C(z^2) = \text{diag}[H_0(z^2) + N_0(z^2), H_0(z^2) + H_1(z^2) + N_1(z^2), H_1(z^2) + N_2(z^2)]^T$. In this case, the output of the filter bank will contain an aliased component $X(-z)$ of the input:

$$Y(z) = z^{-1}T(z)X(z) + \frac{1}{2}[(z^{-1} - z^{-2})(N_0(z^2) - N_2(z^2)) + (z^{-1} + z^{-2})N_1(z^2)]X(z) + \frac{1}{2}[(z^{-2} - z^{-1})(N_0(z^2) - N_1(z^2)) + N_2(z^2)]X(-z)$$

Running the implemented structure in MATLAB, for high precision original transfer function coefficients, we're getting error due to aliasing effects:

```
format long;
t = rand(1,32);
x = rand(1,32);
% direct computation
y1 = filter(t,1,x);
% running FIR filter structure
y2 = structure(t,x);
disp(y2-y1);
```

Error due to round-off effects:

1.0e-014 *

```
0.00555111512313    0    0    0.01110223024625
-0.02220446049250    0    0.04440892098501    0.02220446049250
-0.04440892098501    0    0.04440892098501    0
0.04440892098501    0    -0.04440892098501    0.08881784197001
-0.08881784197001    0.17763568394003    0.17763568394003    0
0.17763568394003    0    -0.17763568394003    0
0.35527136788005    0.08881784197001    0    -0.08881784197001
0.08881784197001    0.08881784197001    0.26645352591004    0
```

Important FIR filters in practice are one with linear phase. In that case, simple technique of pairing input samples corresponding to equal filter coefficients would reduce multiplicative complexity by half. On the other hand, if $T(z)$ is a linear phase FIR filter its polyphase half-band components are also linear phase FIR transfer functions and there is no phase distortion in two channel quadrature mirror filter bank realization shown in Figure 4.

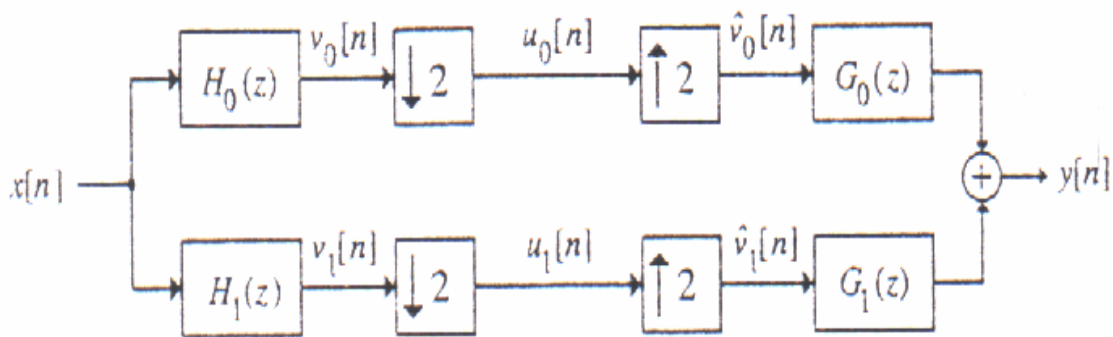


Figure 10.43 The two-channel quadrature-mirror filter (QMF) bank.

Distortion transfer function is zero, i.e.: $G_0(z) = H_1(-z)$ and $G_1(z) = -H_0(-z)$

Amplitude distortion would be minimal if these half band filters are power complementary. In this case, computational complexity is smaller than in proposed three-channel band realization since realization is less complex one.

Conclusion

The use of multirate filter banks for realizing running convolution has been investigated. It was shown how to map long convolutions into smaller ones and aperiodic convolution algorithms for FIR filter has been developed. There are numerous different algorithms for computing running convolution besides the overlap add and overlap save schemes and one of those algorithms was presented.

Acknowledgement

I would like to thank Prof.Mitra for his help and guidance, ECE 258A students for useful hints and to Michael Moore for helping me finding the literature.

References

- S.K.Mitra. *Digital Signal Processing: A computer Based Approach*. McGraw-Hill, New York NY, 1998.
- M.Vetterli. Runnin FIR and IIR Filtering Using Multirate techniques. *IEEE Trans. On Acoustics, Speech and Signal Processing*, 36:730-738, May 1988.