MATRIX FORMULATION: FAST FILTER BANK

Lim Yong Ching

School of Electrical & Electronic Engineering,
Nanyang Technological University,
Singapore 639798

Lee Jun Wei

Department of Electrical and Computer Engineering,
National University of Singapore,
Singapore 119260

ABSTRACT

The Fast Filter Bank (FFB) describes a class of tree-structured filter banks that operate on a frequency response masking principle. Although the structure is highly regular and conveniently implemented in hardware designs, real-time software implementations lead to inefficiencies due to its branching structure. In this paper, an alternative formulation of the FFB is proposed in terms of matrix computations. This allows an efficient approach in its implementation, and significantly reduces the overall buffer memory size required. The matrix operations can be carried out using easily available highly-optimized mathematical software packages, resulting in improvements in computational speed. Savings of up to a factor of 3 in the computer time has been observed during tests on a Pentium 4 computational platform workstation.

1. INTRODUCTION

In [1], the Fast Filter Bank (FFB) was introduced. The FFB splits an input signal into frequency-selective channels. Since its design is based on Frequency Response Masking (FRM) techniques [2], the filter bank functions primarily as a highly efficient single-rate filter bank. Since the overall response of the FFB obtained by summing the outputs of all the channels is a unit delay, it does not introduce any distortion to the input signal. The FFB is a general form of the sliding FFT filter bank [3], and was shown in [4] to have better computational efficiency over other single-rate implementations such as the polyphase implementation.

In many filter bank applications, narrow transition widths between channels and high stopband attenuation are desirable characteristics. Since the design of the FFB is based on the FRM principle, it is able to achieve high selectivity while keeping the order of the prototype subfilters very low. This reduces coefficient sensitivity issues associated with high order filters. The FFB is thus highly suitable for implementation in hardware, and the coefficients can be quantized using integer programming methods [7]-[8].

However, implementing the FFB using software programming methods poses some difficulties. Due to the branching structure of the filter bank, data buffers will have to be implemented prior to each subfilter. For a filter bank with many channels, the number of data buffers required becomes very large. Addressing and manipulating these buffers becomes cumbersome.

In this paper, a matrix formulation is proposed as an alternative representation to the filter bank's operation. The proposed representation in this paper is based on a modified form of the FFB, the node-modulated FFB (nmFFB) [5]-[6], resulting in an efficient and convenient expression. As a result, only one data buffer is required per stage of the filter bank. The data is organized into two-dimensional blocks and are much easier to manipulate. Unlike other types of filter banks, the matrix formulation for the FFB is rather unusual because of its structural properties: 1. It consists of a series of cascaded interpolated subfilters and 2. It has a branching structure.

In modern computers, the limiting factors in processing are usually not due to the rate at which arithmetic operations are performed, but rather due to the efficiency at which data accesses and transfers to and from memory occurs, [9]-[10]. This is especially relevant in our filter bank situation in which a lot of data is often manipulated. By blocking the data in vectors and matrices, data pipelining and efficient memory usage is facilitated within the hardware.

Furthermore, many software mathematical packages are easily available for computing matrix operations. These mathematical packages are highly-optimized for specific processors and make use of the special facilities and architectures inherent in the target processors. One such popular package is the Basic Linear Algebra Subprograms (BLAS), [11]. A further advantage of using standardized packages such as BLAS is to promote structured programming and program portability.

2. BACKGROUND: THE FAST FILTER BANK

The Fast Filter Bank (FFB) was first introduced in [1] as a very efficient single-rate filter bank. It is a generalized form of the sliding Fast Fourier Transform (FFT). The FFB and the FFT is very similar in form and structure when operated as a filter bank. However, whereas the sliding FFT is actually a cascade of first-order subfilters, the FFB is a generalized form consisting of a cascade of higher-order subfilters. As a result, the FFB presents a filter bank with better frequency response characteristics than the FFT at a slightly higher complexity.

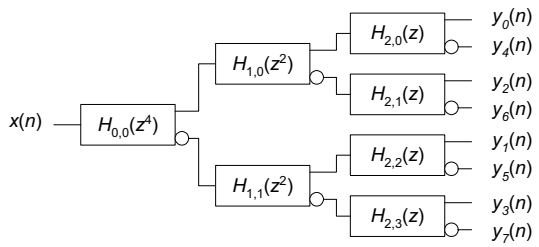

Fig. 1 An 8-channel Fast Filter Bank

Fig. 1 shows a block diagram of an 8 channel FFB. It consists of $K=3$ stages. At each stage k, there are 2^k subfilters which are denoted by $H_{k,i}(z^{L_k})$, and these subfilters are interpolated by a factor of $L_k=2^{K-k-1}$. $H_{k,i}(z)$ for $i>0$ are complex modulated versions of the real-coefficient prototype subfilter $H_k(z) = H_{k,0}(z)$:

$$H_{k,i}(z) = W_N^{\tilde{i}(\Gamma_k-1)/2} H_k\left(W_N^{\tilde{i}} z\right) \tag{1}$$

where $\tilde{i}$ is the bit-reversed version of i in K-1 bits, $N=2^K$ is the number of complex output channels, Γ_k is the length of the prototype subfilter $H_k(z)$ and $W_N = e^{-j2\pi/N}$. The subfilters $H_k(z)$ are assumed to be symmetric linear phase FIR filters. The term $W_N^{-(\Gamma_k-1)/2}$ results in $H_{k,i}(z)$ with conjugate-symmetric coefficients.

In this paper, we adopt the more compact representation of a subfilter as shown in Fig. 2, instead of the butterfly-like structure in [1].

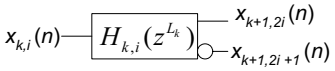

Fig. 2 A subfilter block

The outputs of the subfilter are given by:

$$x_{k+1,2i}(n) = \sum_{m=0}^{\Gamma_k-1} h_{k,i}(m) x_{k,i}(n - L_k m) \tag{2}$$

and:

$$x_{k+1,2i+1}(n) = x_{k,i}\left(n - \frac{\Gamma_k - 1}{2} L_k\right) - x_{k+1,2i}(n) \tag{3}$$

Note that $x_{0,0}(n) = x(n)$ is the input to the FFB and $y_{\hat{i}}(n) = x_{K,i}(n)$, where $\hat{i}$ is the bit-reversed version of i in K bits. $y_{\hat{i}}(n)$ is the $\hat{i}$-th band output of the N-channel FFB centred about frequency $2\pi\hat{i}/N$.

The subfilter $H_{0,0}(z^{L_0})$ plays the biggest role in defining the transition width characteristics of the filter bank, whereas the effect of the remaining subfilters are to mask out unwanted frequency bands. At this point, it is interesting to note how the FFB is a general form of the FFT. If $H_k(z) = 1 + z^{-1}$ for all k, and $H_{k,i}(z^{2^{K-k-1}}) = 1 + W_N^{\tilde{i}} z^{-2^{K-k-1}}$, then the FFB is simply reduced to an N-point FFT structure. Also, note that the filter bank is lossless, ie. the sum of the outputs of all channels is a delayed version of the input.

3. SUBFILTER MATRIX REPRESENTATION

In this paper, we propose a matrix equivalent representation of the subfilters, signal data and the filtering operation in the FFB. As a result, the data is blocked and the implementation is regularized. Furthermore, we can make use of the highly optimized BLAS package, which operates efficiently on vectors and matrices.

Let us first address the matrix representation of the filtering operation in a subfilter. From (2), we can represent this mathematical operation efficiently by blocking the data as shown:

$$\left[\mathbf{x_{k+1,2i}}(n)^T \quad \mathbf{x_{k+1,2i}}(n+L_{k+1})^T\right]^T = \mathbf{X_{k,i}}(n)\mathbf{h_{k,i}} \tag{4}$$

and

$$\begin{aligned}&\left[\mathbf{x_{k+1,2i+1}}(n)^T \quad \mathbf{x_{k+1,2i+1}}(n+L_{k+1})^T\right]^T = \\ &\quad \mathbf{x_{k,i}}\left(n - \frac{\Gamma_k - 1}{2} L_k\right) - \left[\mathbf{x_{k+1,2i}}(n)^T \quad \mathbf{x_{k+1,2i}}(n+L_{k+1})^T\right]^T\end{aligned} \tag{5}$$

for $n = 0, L_k, 2L_k, \ldots$, where

$$\mathbf{x_{k,i}}(n) = \left[x_{k,i}(n) \quad x_{k,i}(n+1) \quad \cdots \quad x_{k,i}(n+L_k-1)\right]^T \tag{6}$$

$$\mathbf{X_{k,i}}(n) = \left[\mathbf{x_{k,i}}(n) \quad \mathbf{x_{k,i}}(n-L_k) \quad \cdots \quad \mathbf{x_{k,i}}\left(n-(\Gamma_k-1)L_k\right)\right] \tag{7}$$

In using this form, we have easily avoided redundant multiplications by zero. The input to the subfilter is organized into a $L_k \times \Gamma_k$ input buffer, $\mathbf{X_{k,i}}$. L_k samples of the input signal, $x_{k,i}(n)$ are shifted into the buffer at a time.

4. FFB MATRIX IMPLEMENTATION

In the previous section, the matrix implementations of the individual subfilters were discussed. In this section, we extend the implementation to cover the FFB as a whole. Observing the tree structure of the FFB in Fig. 1, a straightforward method of implementation would be to allocate 2^k sets of $\mathbf{X_{k,i}}$ and $\mathbf{h_{k,i}}$ matrices per stage of the filter bank. In this section, we propose a method that requires only 1 set of $\mathbf{X_k}$ and $\mathbf{h_k}$ matrices per stage. Furthermore, it is desirable to have the same number of rows for all the $\mathbf{X_k}$ matrices.

In [5]-[6], a method was proposed to reduce the complexity of the FFB by modulating the signal data instead of the subfilter coefficients. We shall use this method to reduce the subfilters so that $\mathbf{h_{k,i}} = \mathbf{h_{k,0}} = \mathbf{h_k}$ for all i. As a result, $\mathbf{h_k}$ will be a real vector. Note that this results in a more efficient implementation by making use of the multiplication between the complex data matrix and the real subfilter vector. As an illustration, the filtering operation for subfilter

$H_{k,i}(z^{L_k})$ can be represented by:

$$x_{k+1,2i}(n) = \sum_{m=0}^{\Gamma_k - 1} e^{\frac{j2\pi[m-(\Gamma_k-1)/2]\tilde{i}}{N}} h_k(m) x_{k,i}(n - L_k m) \qquad (8)$$

where the coefficients of the prototype subfilter are modulated by $2\pi\tilde{i}/N$. An alternative method is to modulate the signal prior to filtering by $-2\pi\tilde{i}/NL_k$ and then demodulating after:

$$\begin{aligned} x_{k+1,2i}(n) &= e^{-\frac{j2\pi\tilde{i}(\Gamma_k-1)/2}{N}} \sum_{m=0}^{\Gamma_k-1} h_k(m) e^{\frac{j2\pi L_k m\tilde{i}/L_k}{N}} x_{k,i}(n - L_k m) \\ &= e^{\frac{j2\pi(n-D_k)\tilde{i}}{NL_k}} \sum_{m=0}^{\Gamma_k-1} h_k(m) e^{-\frac{j2\pi(n-L_k m)\tilde{i}}{NL_k}} x_{k,i}(n - L_k m) \end{aligned} \qquad (9)$$

where $D_k = L_k(\Gamma_k - 1)/2$ is the effective delay of the interpolated prototype subfilter $H_k(z^{L_k})$.

To reduce computations, it was suggested in [5] to modulate only the inputs to every alternate subfilters for each stage, ie. $H_{k,i}(z^{L_k})$ where *i* is odd. The modulating signal is given by:

$$\gamma_k(n) = e^{-j\pi\left(n - \sum_{k'=0}^{k-1} D_{k'}\right)\Big/ 2^{K-k}} \quad \text{for } k = 1, \ldots, K \qquad (10)$$

Then the outputs of the filter bank will be baseband signals. If desired, these signals can be easily modulated back to their original frequencies by applying:

$$y_{\hat{i}}(n) = e^{j2\pi\hat{i}\left(n - \sum_{k=0}^{K-1} D_k\right)\Big/ N} y_{\hat{i}}'(n) \qquad (11)$$

The resultant node-modulated FFB (nmFFB) for *K*=3 and *N*=8 is shown in Fig. 3.

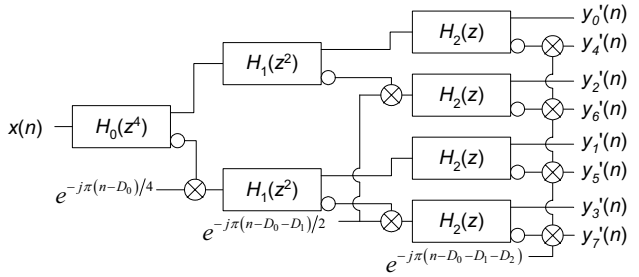

Fig. 3 8-channel nmFFB implementation

Now, let us attempt to arrange the filtering operations for all the subfilters at stage *k*. Since there are 2^k sets of $\mathbf{X_{k,i}}$ and each has L_k rows, we can group all the $\mathbf{X_{k,i}}$ matrices for the same *k* and arrange as follows:

$$\begin{aligned} \mathbf{X_k}(n) &= \left[\mathbf{X_{k,0}}(n)^T \quad \mathbf{X_{k,1}}(n)^T \quad \cdots \quad \mathbf{X_{k,2^k-1}}(n)^T\right]^T \\ &= \left[\mathbf{x_k}(n) \quad \mathbf{x_k}(n-L_k) \quad \cdots \quad \mathbf{x_k}(n-(\Gamma_k-1)L_k)\right] \end{aligned} \qquad (12)$$

$\mathbf{X_k}$ then becomes a $(N/2)\times\Gamma_k$ matrix. Filtering $\mathbf{X_k}$ with with $\mathbf{h_k}$ gives the output:

$$\mathbf{x_k^{out}}(n) = \mathbf{X_k}(n)\mathbf{h_k} \qquad (13)$$

The complementary output should be modulated. In performing the node modulation, and noting that γ_k is periodic with period $4L_k$, let $\boldsymbol{\gamma}_\mathbf{k}(n)$ be a length 2^{k-1} column vector made up of repeating elements of $\gamma_k(n)$:

$$\boldsymbol{\gamma}_\mathbf{k}(n) = \left[\gamma_k(n) \quad \gamma_k(n) \quad \cdots \quad \gamma_k(n)\right]^T \qquad (14)$$

Then, let the modulating column vector be:

$$\boldsymbol{\gamma}_\mathbf{k} = \left[\boldsymbol{\gamma}_\mathbf{k}(0)^T \quad \cdots \quad \boldsymbol{\gamma}_\mathbf{k}(2L_k-1)^T\right]^T \qquad (15)$$

Further noting that $\gamma_k(n) = -\gamma_k(n+2L_k)$, modulation of the complementary output can then be achieved by an element-by-element multiplication of alternating positive and negative versions of the modulating vector with the complementary output vector $\overline{\mathbf{x}}_\mathbf{k}^{\mathbf{out}}(n)$. For the sake of a convenient representation, we would like to denote this operation using matrix multiplications. The modulated vector, $\mathbf{x_{k,mod}^{out}}(n) = (-1)^{n/2L_{k+1}}\left[\boldsymbol{\gamma}_{\mathbf{k+1}}\right]\overline{\mathbf{x}}_\mathbf{k}^{\mathbf{out}}(n)$ where $\left[\boldsymbol{\gamma}_{\mathbf{k+1}}\right]$ is a diagonal matrix with the diagonal elements made up of the column vector $\boldsymbol{\gamma}_{\mathbf{k+1}}$. In summary, the modulated version of the complementary subfilter output is given by:

$$\mathbf{x_{k,mod}^{out}}(n) = (-1)^{n/2L_{k+1}}\left[\boldsymbol{\gamma}_\mathbf{k}\right]\left\{\mathbf{x_k}\left(n - \frac{\Gamma_k - 1}{2}L_k\right) - \mathbf{x_k^{out}}(n)\right\} \qquad (16)$$

The vectors $\mathbf{x_k^{out}}(n)$ and $\mathbf{x_{k,mod}^{out}}(n)$ forms the basis of the inputs to the next stage, but needs to be reordered. The reordering scheme is illustrated in Fig. 4. Fig. 5 shows the resultant block diagram implementation of our scheme.

$$\mathbf{x_k^{out}}(n) = \begin{bmatrix} x_k^{out}[1] \\ x_k^{out}[2] \\ \vdots \\ x_k^{out}[N/2] \end{bmatrix} \xrightarrow{\text{reorder}} \begin{bmatrix} x_k^{out}[1] \\ x_{k,\text{mod}}^{out}[1] \\ \vdots \\ x_k^{out}[N/4] \\ x_{k,\text{mod}}^{out}[N/4] \end{bmatrix} = \mathbf{x_{k+1}}(n)$$

$$\mathbf{x_{k,mod}^{out}}(n) = \begin{bmatrix} x_{k,\text{mod}}^{out}[1] \\ x_{k,\text{mod}}^{out}[2] \\ \vdots \\ x_{k,\text{mod}}^{out}[N/2] \end{bmatrix} \xrightarrow{\text{reorder}} \begin{bmatrix} x_k^{out}[1+N/4] \\ x_{k,\text{mod}}^{out}[1+N/4] \\ \vdots \\ x_k^{out}[N/2] \\ x_{k,\text{mod}}^{out}[N/2] \end{bmatrix} = \mathbf{x_{k+1}}(n+L_{k+1})$$

Fig. 4 Reordering scheme

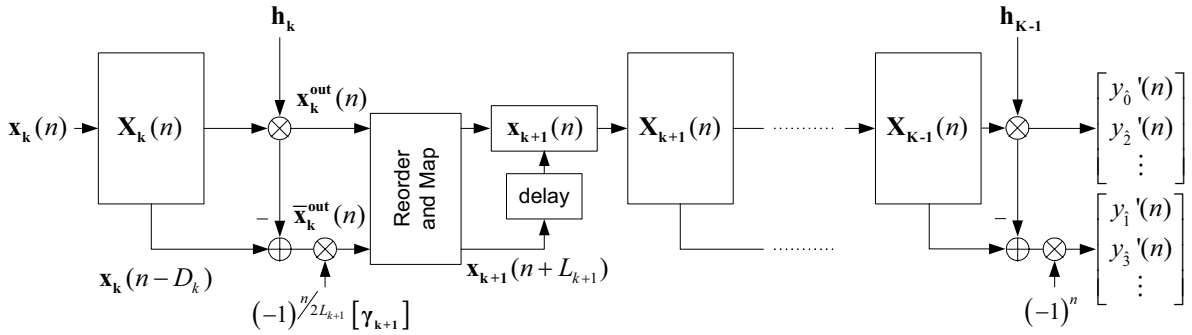

Fig. 5 Matrix allocation scheme for the nmFFB

From Fig. 5, we can observe that there are 2 output vectors for every input vector to a subfilter. Thus, at each subsequent stage, filtering must be processed twice as frequently as the previous stage. A method of achieving this is to make use of recursive functions. A sample algorithm is as follows:

Function Subfilter ($\mathbf{x_k}(n)$, k)
  Filter with $\mathbf{h_k}$
  Reorder to get $\mathbf{x_{k+1}}(n)$ and $\mathbf{x_{k+1}}(n+L_{k+1})$
  Call Subfilter ($\mathbf{x_{k+1}}(n)$, k+1)
  Call Subfilter ($\mathbf{x_{k+1}}(n+L_{k+1})$, k+1)
End of Function

---

5. RESULTS

A Fortran program was used to compare the performances of a FFB using the normal filtering method and our proposed matrix formulation. The computational platform used was an Intel Pentium 4-based workstation. The time taken for a 16-channel FFB sample run with 10K input data samples using the normal filter method was approximately 0.0164s, and the matrix method took approx 0.0139s. For a 256-channel FFB sample run with 10K input data samples, the normal filter method required about 0.28s, and the matrix method took about 0.1s.

6. CONCLUSIONS

The Fast Filter Bank is a generalized form of the sliding FFT filter bank. By using Frequency Response Masking design methods, it is an efficient implementation of single-rate filter banks. Although it is highly suitable for hardware designs, it is much more complicated to realize it in real-time software systems. This is because the number of buffers required to store the intermediate data increase exponentially with the number of stages. Memory allocation, addressing and data handling could result in significant overhead processing.

A matrix representation of the node-modulated Fast Filter Bank was proposed. By blocking the input data, an efficient filtering scheme can be implemented using vectors and matrices. The data buffers are regularly-sized, with equal number of rows. By further making use of easily available mathematical packages such as BLAS, which are highly optimized for specific target architectures, savings by up to a factor of 3 has been observed.

7. REFERENCES

[1] Y.C Lim and B. Farhang-Boroujeny, ``Fast Filter Bank (FFB) '', *IEEE Trans. on Circuits and Systems II*: Vol. 39, No. 5, pp. 316-318, May, 1992.

[2] Y.C Lim, ``Frequency Response Masking Approach for the Synthesis of Sharp Linear Phase Digital Filters'', *IEEE Trans. on Circuits and Systems*: Vol. 33, No. 4, pp. 357-364, Apr, 1986.

[3] B. Farhang-Boroujeny and Y.C Lim, ``A Comment on the Computational Complexity of the Sliding FFT'', *IEEE Trans. on Circuits and Systems II*: Vol. 39, No. 12, pp. 875-876, Dec 1992.

[4] Y.C Lim and B. Farhang-Boroujeny, ``Analysis and Optimum Design of the FFB'', *IEEE ISCAS 1994*: Vol. 2, pp. 509-512, 1994.

[5] J.W. Lee and Y.C. Lim, "Efficient Implementation of Real Filter Banks using Frequency Response Masking Techniques", *IEEE APCCAS 2002*, Vol. 1, pp 69-72, 2002.

[6] J.W. Lee and Y.C. Lim, "Designing the Fast Filter Bank with a Minimum Complexity Criterion", *IEEE ISSPA Paris 2003*, Vol. 2, pp 279-282.

[7] L.R Rabiner, "Linear program design of Finite Impulse Response (FIR) digital filters", *IEEE Trans. On Audio and Electroacoustics*, Vol. AU-20, pp. 280-288, Oct. 1972.

[8] D. Kodek, "Design of optimal finite wordlength FIR digital filters using integer programming techniques", *IEEE Trans. on ASSP*, Vol. ASSP-28, pp.304-308, June 1980.

[9] David R. Wille, *Advanced Scientific Fortran*, John Wiley & Sons, 1995.

[10] C.H. Koelbel, D.B. Loveman, R.S. Schreiber, G.L. Steele Jr., and M.E. Zosel, *The High Performance Fortran Handbook*, The MIT Press, 1994.

[11] *http://www.netlib.org/blas/*

* The authors would like to express their gratitude towards the Singapore Millennium Foundation for their funding and support.