DIGITAL SIGNAL PROCESSING **8,** 126–135 (1998)
ARTICLE NO. SP970309

Low-Complexity Implementation of a Polyphase Filter Bank

Paul D. Fiore*

Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts, and Signal Processing Center, Sanders, a Lockheed–Martin Company, Nashua, New Hampshire 03061

Fiore, P. D., Low-Complexity Implementation of a Polyphase Filter Bank, *Digital Signal Processing* **8** (1998), 126–135.

This paper describes the tradeoffs and design method for a polyphase filter bank implemented in field programmable gate array technology. Consideration is given to the minimization of required hardware resources by careful choice of the FFT approach, multiplier implementation, and wordlength optimization. These choices are made subject to maintaining specified output performance. Finally, the sigma–delta technique is examined to further reduce the computational complexity.

I. INTRODUCTION

Filter banks have found widespread usage in many signal processing applications, including subband speech coding, image compression, and radar electronic support measures (ESM). In this paper, we describe the architecture and design tradeoffs that were performed for a polyphase filter bank for ESM applications. In ESM, filter banks are generally used as a front end for further processing, such as tone detection or channel copy applications [1]. Simple tone detectors, such as instantaneous frequency measures (IFMs) [2–5], rely on only one tone being present within its input bandwidth. Channelizing the entire input bandwidth into multiple channels increases the probability that tones will be isolated into individual channels. Also, low SNR targets can be isolated from high power jammers if they can be resolved into separate channels.

There is a well-developed theory of nonadaptive uniform filter banks [6–8]. Adaptive, nonuniform, and wavelet filter banks have also received great attention in the literature, but will not be considered here.

Recent advances in A/D technology have made high-bandwidth, high-SNR ESM channelizers possible. To process these high bandwidths in real time with a minimal amount of hardware, one must closely examine fixed-point computation strategies. This paper documents the design techniques used to produce an efficient filter bank design, currently being implemented with commercially available filters and field programmable gate arrays (FPGAs).

Modern FPGAs provide a large amount of logic resources (that can be used as either digital logic or lookup tables) and copious routing resources. For this paper, we assume that a full-adder cell is the basic unit by which resource utilization is measured.

II. FILTER BANK REQUIREMENTS AND ARCHITECTURE SELECTION

Our filter bank had the following design requirements:

1. Accept a single 320-MHz 8-bit real data stream.
2. Produce eight 20-MHz complex channels (min).
3. Out-of-band attenuation of 50 dB (min).
4. Stop band corner at center of adjacent channel.
5. Channel crossover point at 1 dB attenuation (max).
6. In-band ripple of 1 dB (max).
7. In-band dynamic range of 60 dB (min).

There are several approaches to implementing the filter bank: parallel demodulate and lowpass filter, short-time Fourier transform (STFT [8]), and polyphase filter bank [6] approaches. For the first ap-

* Fax: (603) 885-0631. E-mail: pfiore@sanders.com.

1051-2004/98 $25.00

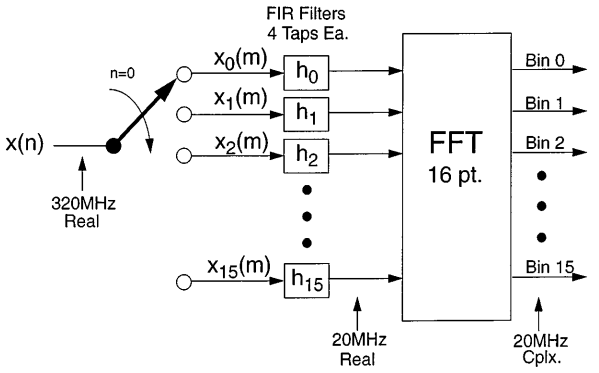

FIG. 1. Polyphase filter bank structure.

proach, one demodulates the center of each desired channel to baseband, then applies a lowpass filter to shape the channel. The outputs are then decimated, resulting in modest aliasing. The second two approaches can provide identical results, but with a tremendous reduction in required computations.

We chose the polyphase filter bank approach (which is essentially the same as the STFT). The basic structure is shown in Fig. 1. The filter bank is composed of a commutator, a set of (polyphase) filters, and a DFT (via FFT). The 16 filters each produce a real output at the rate of 20 MHz. A 16-point FFT is then evaluated at the 20 MHz rate. Since the input was real, only nine bins are actually calculated (the zeroeth and eighth bins are purely real).

To achieve a desired channel shape, we first design a prototype lowpass filter $h(n)$. The polyphase filters will then be derived from $h(n)$ via

$$h_i(n) = h(16n - i), \quad 0, 1, \ldots 15. \tag{1}$$

The prototype lowpass filter must supply 50 dB of out-of-band attenuation. For low risk, we used commercially available FIR filter chips. Each chip can implement a 4-tap FIR filter at the required 20 MHz data rate. This constrains the prototype lowpass filter $h(n)$ to be $16 \cdot 4 = 64$ taps in length. The filters employ 10 bits of precision for both data and coefficients and provide 28 bits of output wordlength.

The prototype lowpass filter was designed using the McClellan–Parks design algorithm [9]. The resulting infinite precision outputs were rounded to 10 bits by employing a nonoptimal "greedy" rounding scheme in which coefficients were successively rounded in the direction that gave the best stopband attenuation. Figure 2 shows the prototype filter response.

III. FFT IMPLEMENTATION

This section describes the chosen implementation of the FFT operation. There are many approaches to

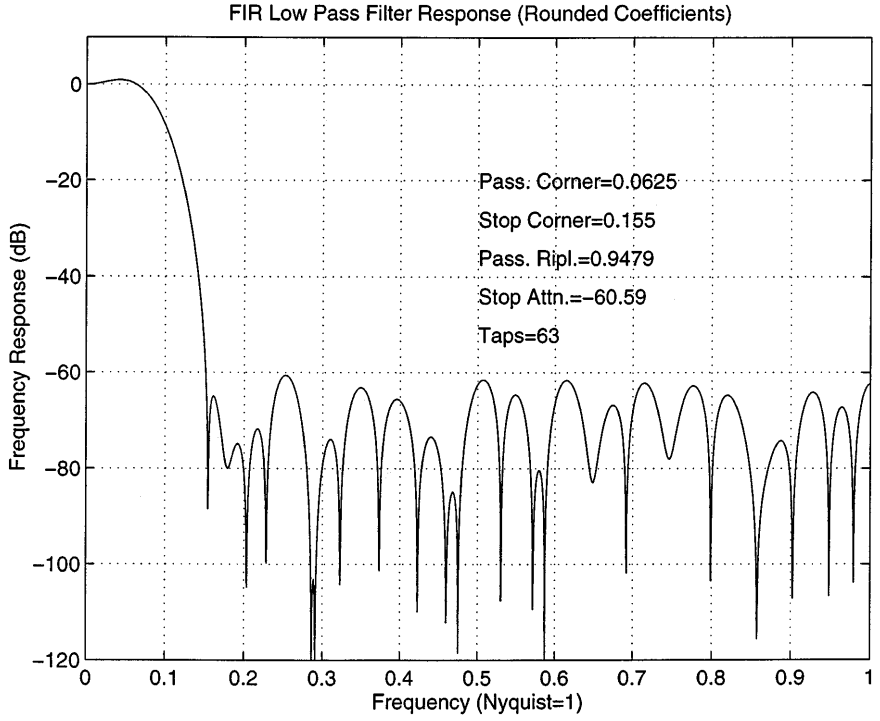

FIG. 2. Prototype lowpass filter frequency response with coefficients rounded to 10 bits.

implementing the FFT that follows the polyphase lowpass filter [10]. The choice of the best algorithm is effected by the implementation technology, latency, and throughput requirements.

The Cooley–Tukey FFT algorithms are perhaps the most well-known methods. These are composed of stages of butterflies, with complex multiplication ("twiddles") between stages. Interstage complex multiplications can also be replaced by CORDIC rotations for the Cooley–Tukey algorithms.

Distributed arithmetic [11] could be used to implement the complex multiplications. This is a sequential ROM-based approach for implementing vector dot products and has been successfully applied to filters and FFT computations in FPGAs [12].

Finally, the Winograd approach [10] also deserves attention for this application. The Winograd FFT algorithms have the minimum number of real multipliers. The structure of the computations is a set of additions, then a set of real multiplications, followed by another set of additions. These properties make the Winograd approach ideal for FPGA implementations, in which a large number of additions can be performed.

The Wingorad FFT decomposes the DFT matrix into the product of three sparse matrices

$$\mathbf{F} = \mathbf{C} \cdot \mathbf{B} \cdot \mathbf{A}, \tag{2}$$

where **F** is the DFT matrix of size 16 whose (i, k)th element is $\exp(-j2\pi ik/16)$. Matrices **A** and **C** only contain elements $-1, 0, 1, j, -j$ and are thus implemented as additions and subtractions. Matrix **B** is a diagonal matrix. It is here that general purpose multiplications appear. The three matrices are given in [10].

In order to perform a Winograd FFT on a data set, form a vector **v** of the 16 data values and then calculate

$$\begin{aligned} \mathbf{a} &= \mathbf{A}\mathbf{v} \\ \mathbf{b} &= \mathbf{B}\mathbf{a} \\ \mathbf{c} &= \mathbf{C}\mathbf{b}. \end{aligned} \tag{3}$$

Of course, the matrix-vector multiplications are not carried out in the usual manner. Rather, the sparse matrices lend themselves to a "straight-line code" implementation where partial sums are formed and shared in forming the outputs of the matrix-vector product. This is equivalent to factoring the **A** and **C** matrices.

A graphical version of the straight-line code given in [10] is shown in Figs. 3 and 4. In these figures, an asterisk represents an addition or subtraction operation. Lines leading into the asterisked nodes represent the data required to form the node. Nodes are formed by either addition or subtraction of two inputs to the node. A dashed line means the input is negated and a solid line means the input is not negated. Bins 9 through 15 are not calculated because they contain the redundant conjugate spectrum of the real-input data.

IV. ROUNDING STUDY

Since we were interested in efficient implementation of the filter bank, minimizing word lengths in all portions of the algorithm would reduce the number of full-adder cells in the FPGA. A study of the effect of rounding the various computed intermediate variables was next conducted.

First, rounding of the **B** coefficients in the Winograd FFT will be addressed. When the infinite precision coefficients are rounded, the overall channel responses are distorted. The main effect will be to degrade the out-of-band attenuation of each channel. Achieving the maximum amount of rounding possible is critical here since the multipliers take up large amounts of FPGA resources.

A "nice number" study was performed on the **B** coefficients. These are numbers that can be represented as sums and differences of powers of two. Since the multiplier values are fixed, dynamic shifting is not required; the required shifts can be hardwired. It was found that a maximum of 4 adds in a 10 bit dynamic range was required to implement the **B** coefficients. The nice coefficients (scaled by 256) are $256 \cdot \mathrm{diag}(\mathbf{B})^T \approx [256, 256, 256, 256, 180, 180, 98, 334, -139, 256, 256, 256, -180, -180, -237, 139, -334]$.

The decomposition of these values is given as

$$\begin{aligned} 256 &= 256 \\ 180 &= 128 + 32 + 16 + 4 \\ 98 &= 64 + 32 + 2 \\ 334 &= 256 + 64 + 16 - 2 \\ -139 &= -128 - 8 - 2 - 1 \\ -180 &= -128 - 32 - 16 - 4 \\ -237 &= -256 + 32 - 8 - 4 - 1 \\ 139 &= 128 + 8 + 2 + 1 \\ -334 &= -256 - 64 - 16 + 2. \end{aligned} \tag{4}$$

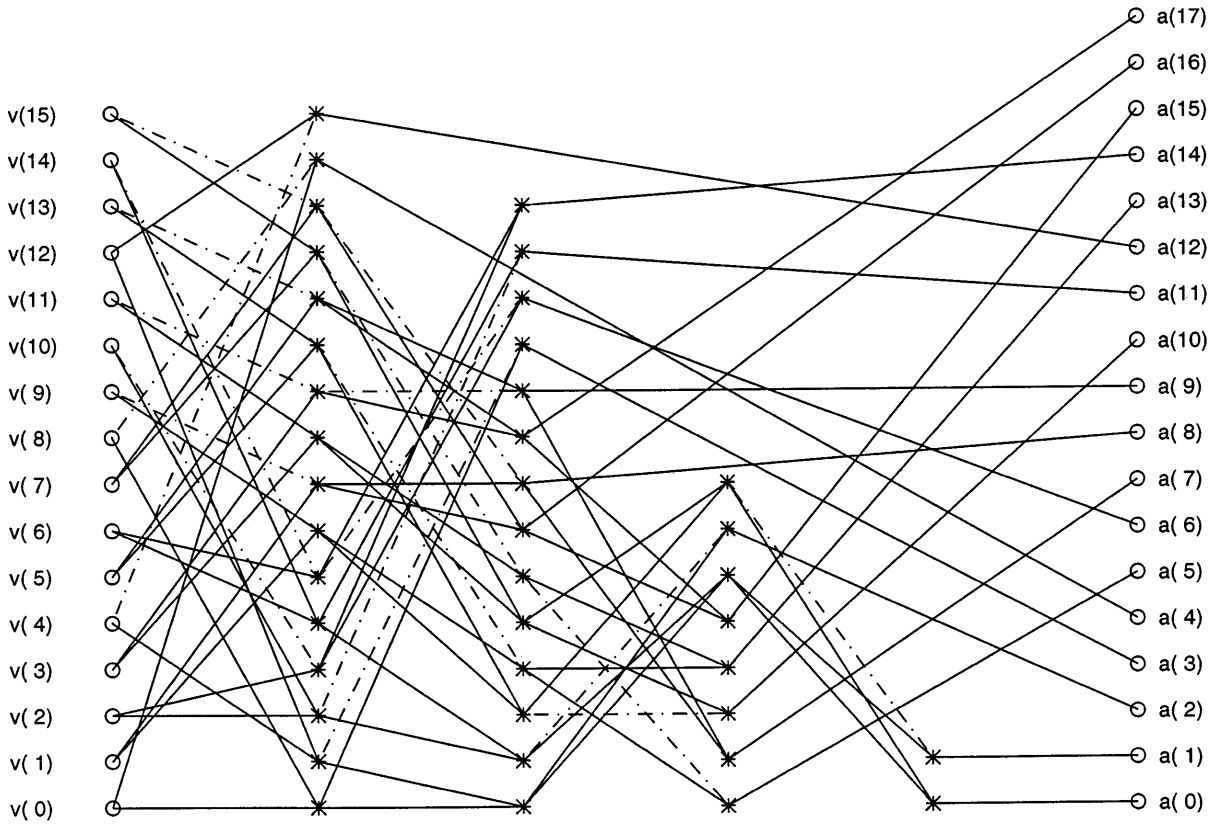

FIG. 3. Flow graph of $\mathbf{a} = \mathbf{A} \cdot \mathbf{v}$ calculation.

Now, the wordlengths required throughout the algorithm must be calculated. Again, minimizing the number of bits required will greatly aid the FPGA implementation.

Rounding at various parts of the algorithm will limit the in-band dynamic range. With fixed-point processing, one must be concerned with bit growth, at both the most and the least significant bit ends (MSB and LSB, respectively) of the words. Growth at the MSB generally occurs with additions and growth at the LSB occurs because of multiplications. Our algorithm here has both of these growths, which must be accounted for.

To determine the appropriate number of bits to round at various stages, we will examine the SNR at the output of the channelizer as a function of various rounding schemes. An optimization technique will be used to tune the rounding amounts. This will minimize the amount of required FPGA resources subject to maintaining a given output SNR.

We must calculate the bit growth (due to additions/subtractions) throughout the algorithm. Four points were selected within the algorithm: FIR output, **A** output, **B** output, and **C** output. In calculating bit growth, one must measure the growth from the A/D output (8 bits) to the point of interest. This is different than measuring the maximum growth of a particular stage, which generally is overly pessimistic.

To calculate the growth from the A/D output to the FIR output, notice that the lowpass prototype FIR filter taps are used to form the 16 small FIR filters. Every 16th tap weight goes to a single small filter. The maximum growth through an FIR filter is the sum of the absolute value of the tap weights (assuming the input is between -1 and 1). For our filter, this value is 1.3594, which can be handled by an implicit divide by 2.

The maximum gain through **A** is 16. Since the maximum value coming into **A** from the FIR filters is $1.3594/2 = 0.6797$, the maximum gain from the A/D output to the **A** output is $16 \cdot 0.6797 = 10.8752$. An implicit shift of 4 bits for the **A** output will be needed.

Now we must calculate the gain through **B**. To do this, consider the combined matrix $\mathbf{B} \cdot \mathbf{A}$ and calculate the gain through this. The maximum gain through the combined matrix is 16. Even though **B**

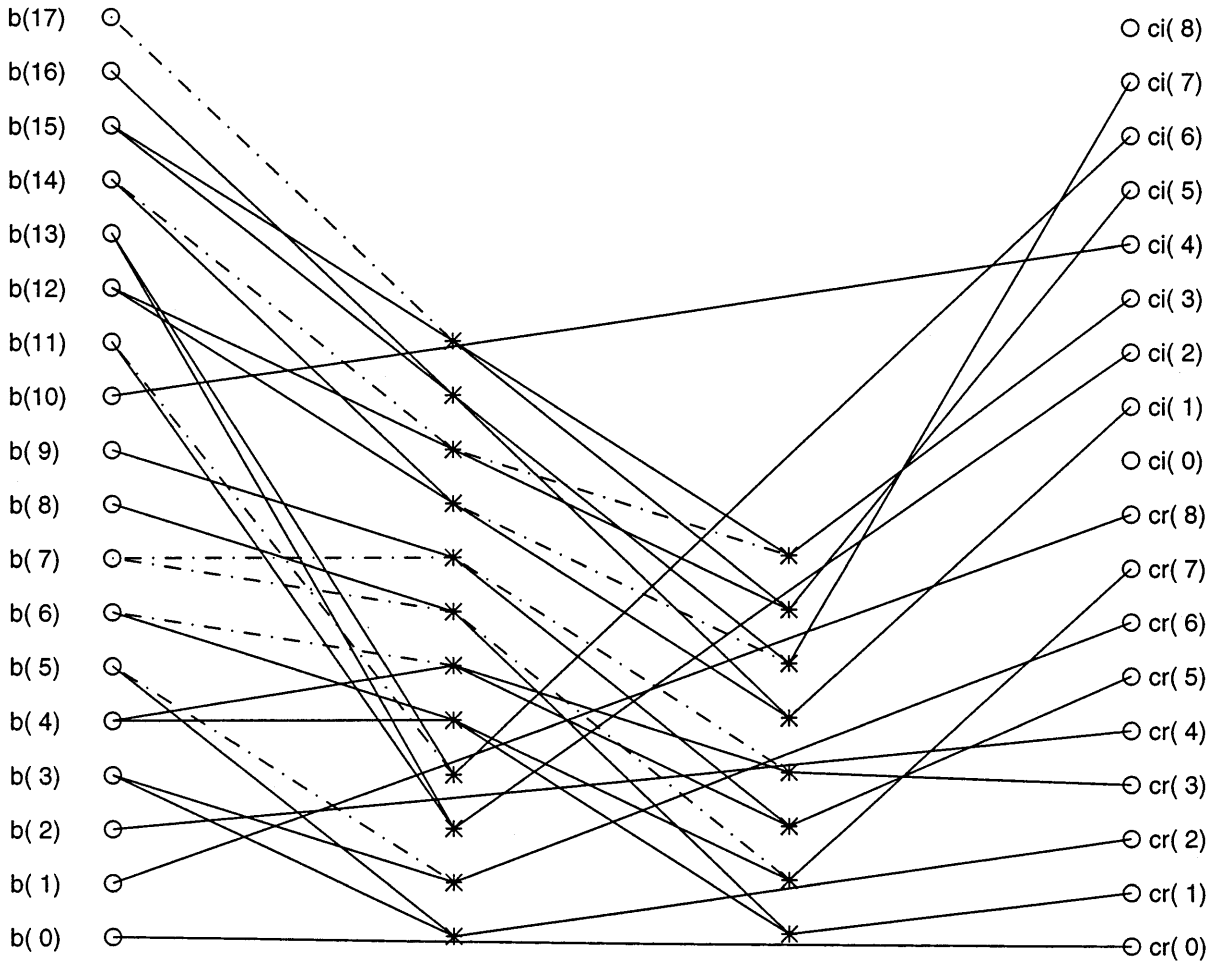

FIG. 4. Flow graph of $\mathbf{c} = \mathbf{C} \cdot \mathbf{b}$ calculation.

has elements greater than 1, they align with elements of **A** that cannot be full scale, so no additional gain is possible. Since we have accounted for the gain of **A** by implicit shifting, no additional implicit shifting is required for **B** for gain reasons. Note that since we generally work with the nice **B** values which are integers, we must perform an implicit shift of 8 bits to divide by 256.

To calculate the gain at the **C** output, recall that $\mathbf{C} \cdot \mathbf{B} \cdot \mathbf{A}$ is a rounded version of **F**. The maximum gain in a real or imaginary component through a DFT matrix is the size of the matrix, in our case 16. Since we have already accounted for a gain of 16 in the **A** output, no additional implicit shifting is needed.

Thus, we will perform implicit shifts of 1, 4, 8, and 0 bits at the FIR, **A**, **B**, and **C**, respectively. Implicit shifts effect the SNR calculation, but do not result in extra hardware.

With the gains understood, we must now calculate the signal variance at the various point in the algorithm. Assume an input signal variance of 1 at the A/D output. The signal variance at the FIR output is the sum of the squares of the tap values of the filter times the input variance. Similarly, the signal variance at the output of the matrices is the sum of the squares of the row elements of the matrices, divided by the square of any implicit scaling factor. For the filter and matrices considered here, the signal variances are

$$\mathbf{sv} = \begin{bmatrix} 0.1631 \\ 0.0040 \\ 0.0034 \\ 0.0068 \end{bmatrix}. \tag{5}$$

Note that the vector of output variances for each matrix-vector product has been averaged for simplicity. Also, note that sv_i represents the cumulative signal variance gain from the A/D output through stage i.

Now we turn to the rounding noise variance calculation. Rounding noise will be modeled as additive white noise. The effect of rounding at a particular stage of the algorithm on the output SNR is of interest. A good approximation is to assume that the noise injected at a particular stage of the algorithm is independent of any other rounding noise. All we must do is calculate the noise power at the output of the (linear) system due to noise injected at an intermediate point. This is easy since the path from any intermediate point to the output is merely a matrix multiplication. It is straightforward to show that the total rounding noise power at the output is

$$\sigma_r^2 = \frac{sv_4}{sv_4}\sigma_4^2 + \frac{sv_4}{sv_3}\sigma_3^2 + \frac{sv_4}{sv_2}\sigma_2^2 + \frac{sv_4}{sv_1}\sigma_1^2, \qquad (6)$$

where σ_i^2 is the noise variance injected at stage i.

The injected noise variance is a function of the number of bits maintained after rounding. Specifically, if b_i bits are maintained after rounding at stage i, the noise variance is well approximated as

$$\sigma_i^2 = \frac{2^{-2b_i}}{12}. \qquad (7)$$

We now have enough information to propose various wordlength combinations and calculate the output SNR. The next question that arises is, how does one optimize the choice of word lengths, while maintaining a fixed SNR? In our case, we would like to minimize the computing resources necessary in the calculation of the 16 point DFT of the filter outputs. Since we have decomposed the Winograd FFT into additions, it is easy to quantify how much logic will be required as a function of wordlengths. For our case, let $[b_1 b_2 b_3 b_4]^T$ represent the number of bits maintained after the FIR filter, **A**, **B**, and **C** calculations, respectively. From (4) and Figs. 3 and 4, the **A** calculation requires 40 additions, the **B** calculation 30 additions, and the **C** calculation 20 additions. The number of required full-adder cells is therefore

$$J = 40b_1 + 30b_2 + 20b_3. \qquad (8)$$

Thus, one would like to minimize the above subject to maintaining some minimum output SNR as defined by Eqs. (6) and (7).

An *ad hoc* greedy algorithm was used to pick the components b_i for a given minimum output SNR. All components are initialized to a lower bound on the number of required bits. At each stage, one of the components will be incremented. This component is chosen as the one that would give the maximum output SNR. If one or more components give an output SNR greater than the required minimum, the algorithm terminates. Table 1 shows the results of running this algorithm. The FPGA was designed for the 60-dB case.

TABLE 1
Channel SNR and Wordlengths (in Bits)

SNR (dB)	FIR	A	B	C
50	9	11	12	12
55	10	12	13	12
60	11	13	13	13
65	11	14	14	15
70	12	15	15	15
75	13	16	16	15
80	14	16	17	17
85	15	17	18	17
90	16	18	18	18
95	16	19	19	20
100	17	20	20	20

V. MINIMUM VARIANCE REQUANTIZATION FILTER

Recently, F. Harris has applied sigma–delta techniques to reduce computational complexity of filtering operations [13]. We can use these techniques to further reduce computational complexity for polyphase filter banks with long prototype filter lengths. Consider the operation of lowpass filtering. One typically implements this with an FIR filter using general purpose multipliers. The wordlengths of the coefficients and data directly effect the size or complexity of the filter. If one requantizes the data so that the data wordlength is reduced, then a savings in hardware can be achieved at the cost of increased noise in the filter output. Harris' paper shows that this noise may be reduced by using error feedback and noise shaping techniques. We will apply this technique to the individual polyphase filters of a filter bank, with frequency responses shown in Fig. 6, to achieve the quantization noise improvement shown in Fig. 7.

Consider the system shown in Fig. 5, which shows a conventional filtering application preceded by a

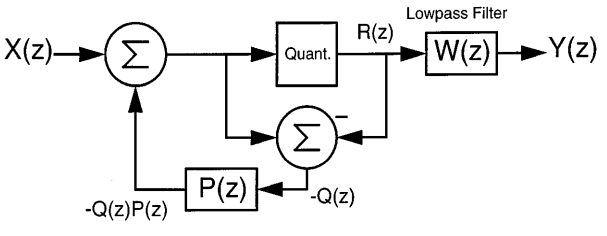

FIG. 5. Requantizer with feedback prior to lowpass filter.

requantizer. The quantization noise enters the data stream by either rounding or truncating bits of the data words. It is modeled as additive white noise. The requantizer works by feeding back a filtered version of the quantization noise $Q(z)$ into the data path in such a way that most of the noise energy will be moved out of the pass band. The response at the output of the requantizer is

$$R(z) = X(z) + Q(z)(1 - P(z)). \tag{9}$$

Let $H(z) = 1 - P(z)$ be the noise transfer function z-transform. We try to make $H(z)$ small in the frequency regions of interest to reduce the output quantization noise. Harris assumes a spectral model for the noise, as well as an additive noise pedestal to improve an ill-conditioning problem in his solution. He then derives the solution in terms of the Normal equations. Here, we take a different approach. We seek to minimize the noise variance at the output of the lowpass filter.

Let $h(n)$ be the noise transfer function impulse response. For causality, we require $h(n) = 0$ for $n < 0$. We also require $h(0) = 1$, because the loop filter z-transform $P(z)$ has no constant term. If $P(z)$ were allowed to have a constant term, the loop in Fig. 5 would contain a path with no delay elements.

Given the filter spectrum $S_{WW}(\omega) = |W(e^{j\omega})|^2$, what is the optimal noise transfer function spectrum $S_{HH}(\omega)$? We will show that the optimal filter is the inverse, or whitening filter of $W(z)$.

Assume the quantization noise $S_{QQ}(\omega)$ is spectrally white. Let the variance be denoted by σ_Q^2. The variance at the output of $W(z)$ due to the quantization noise is

$$\sigma_W^2 = \sigma_Q^2 \int_{-\pi}^{\pi} |H(e^{j\omega})|^2 |W(e^{j\omega})|^2 d\omega. \tag{10}$$

We thus must minimize the value of the integral to minimize the variance. Let the composite transfer function be $Y(z) = H(z)W(z)$. We will show that

$$\int_{-\pi}^{\pi} Y(e^{j\omega}) d\omega \tag{11}$$

is constant and that under this constraint

$$\int_{-\pi}^{\pi} |Y(e^{j\omega})|^2 d\omega \tag{12}$$

is minimized when $|Y(e^{j\omega})|^2$ is a constant (independent of ω). To prove that (11) is constant for all admissible filters $H(z)$, notice that the coefficient of the z^0 term in the z-transform $Y(z)$ is $y(0) = h(0)w(0) = w(0)$. Replacing $Y(e^{j\omega})$ in (11) with its z-transform summation and interchanging the integral and summation gives

$$\int_{-\pi}^{\pi} Y(e^{j\omega}) d\omega = \sum_{n=0}^{N_Y} y(n) \int_{-\pi}^{\pi} e^{-jn\omega} d\omega = 2\pi w(0), \tag{13}$$

thus proving the first assertion (11). A nonrigorous proof of the second assertion can be had by considering a discretized version of the problem. Suppose we wish to minimize the quadratic form $\mathbf{a}^H\mathbf{a}$ subject to the constraint $\mathbf{a}^T\mathbf{1} = 1$. Using a Lagrange multiplier, we differentiate $\mathbf{a}^H\mathbf{a} - \lambda(\mathbf{a}^H\mathbf{1} - 1)$ with respect to each unknown a_i and set these results equal to zero, obtaining $a_i = \lambda/2$ for all i. Thus, we require the solution $a_i =$ Constant. Nonrigorously extending this to an infinite number of variables, the second assertion (12) is proved. We have thus proved that $S_{HH}(\omega) = \alpha/S_{WW}(\omega)$, where α is chosen to rescale $H(z)$ so that $h(0) = 1$.

We can explicitly derive the coefficients of $H(z)$ while imposing a finite impulse response restriction (and $h(0) = 1$). Define

$$C_k = \int_{-\pi}^{\pi} \cos(k\omega) |W(e^{j\omega})|^2 d\omega \tag{14}$$

for $k = 1, \ldots, N$. It is not difficult to show that $h(1), \ldots, h(N)$ satisfy

$$\begin{bmatrix} \vdots & & \vdots \\ \cdots & C_{|r-c|} & \cdots \\ \vdots & & \vdots \end{bmatrix} \begin{bmatrix} h(1) \\ h(2) \\ \vdots \\ h(N) \end{bmatrix} = -\begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_N \end{bmatrix}. \tag{15}$$

The matrix on the left is symmetric and Toeplitz. The (r, c)th element is $C_{|r-c|}$. These results can be obtained by by differentiating (10) with respect to the unknowns $h(k)$ and setting the derivatives equal to zero.

The reduction in noise variance can be easily calculated. Without the feedback loop, the noise variance at the output of $W(z)$ would be $\sigma_Q^2(w^2(0) + \cdots + w^2(N_w))$. With the loop filter, the variance is $\sigma_Q^2(y^2(0) + \cdots + y^2(N_y))$. The variance reduction is simply the ratio of the two variances. The impulse response of the combined filter $Y(z)$ can be had by convolving the impulse responses $h(n)$ and $w(n)$.

We can derive a bound on the achievable variance reduction. As we allow for increasingly longer filters $h(n)$, $y(n)$ becomes a better approximation to an

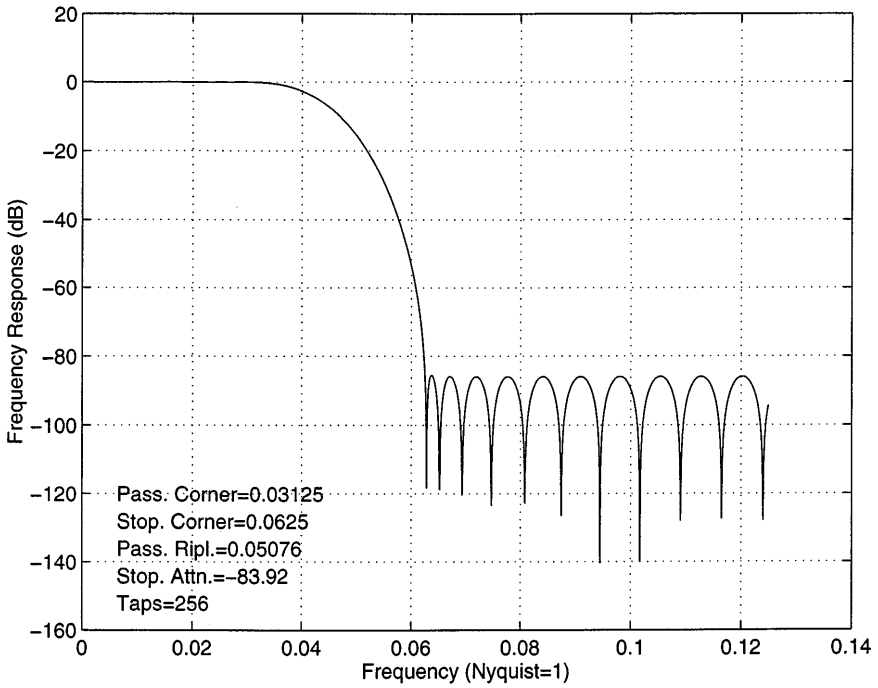

FIG. 6. Frequency response of prototype lowpass filter.

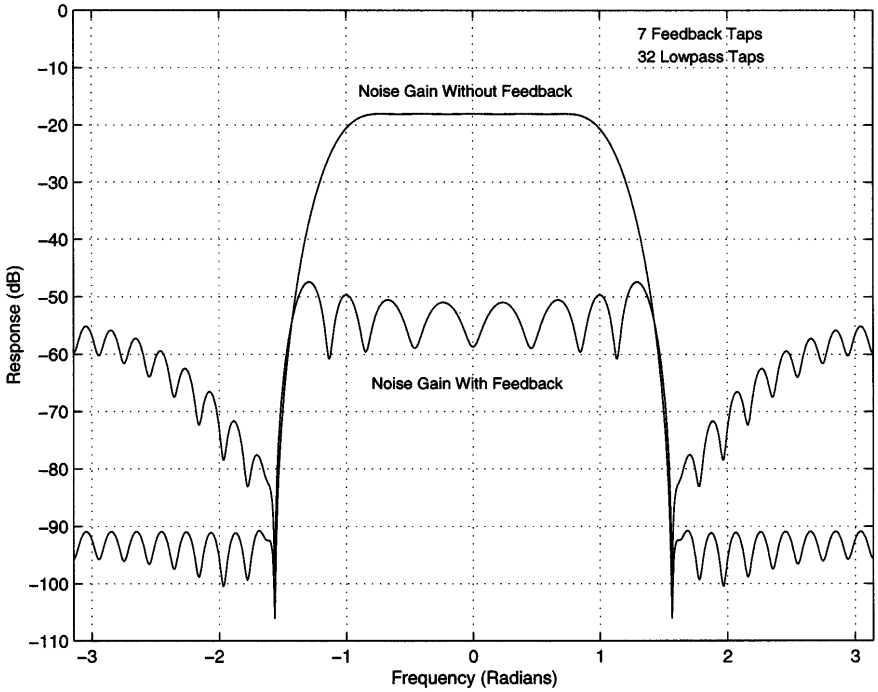

FIG. 7. Effect of requantization error feedback on polyphase filter response.

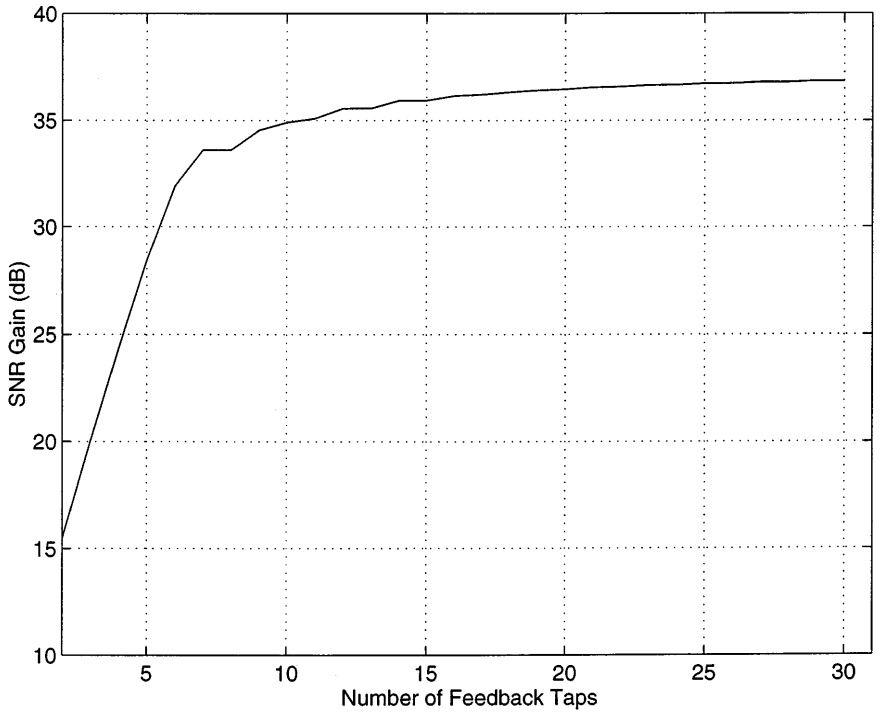

FIG. 8. SNR gain as a function of the number of feedback taps.

impulse, scaled by $w(0)$. Thus, a bound on the achievable variance reduction ratio is $(w^2(0) + \cdots + w^2(N_w))/w^2(0)$.

Now let us apply this technique to a polyphase filter bank. We desire eight output channels, with a prototype filter length of 256 taps. Figure 6 shows the frequency response of the prototype filter.

Since the individual polyphase filter responses also have a considerable stop band attenuation, we can apply the requantization and filtering approach to reduce the wordlength for the filters and the following FFT. Figure 7 shows the effect of requantization error feedback on the polyphase filter responses. Seven taps were used in the error feedback loop. Note that the filter attempted to whiten the output noise, subject to the restricted number of taps. Also note the considerable gain against requantization noise. Figure 8 shows this gain as a function of the number taps allowed in the feedback loop (the values plotted are variance ratios expressed in dB).

VI. SUMMARY

In this paper, we examined several straightforward approaches to minimizing hardware complexity for implementing a filter bank. Choosing the Winograd FFT minimized the number of expensive multiplications. Next, several approximations were introduced to reduce complexity, while maintaining required performance. These included replacing each multiply by a small number of carefully chosen shift-and-adds, and reducing the data wordlength in a controlled fashion throughout the algorithm. Finally, a requantizing technique was presented that promises to significantly reduce the complexity of the FIR filter portion of the filter bank.

ACKNOWLEDGMENTS

This work was supported by ONR Contract N00014-96-C-0389 and Sanders, a Lockheed–Martin Company.

REFERENCES

1. Fields, T. W., Sharpin, D. L., and Tsui, J. B. Digital channelized IFM receiver. *IEEE MTT-S Digest* (1994).
2. Kay, S. Statistically/computationally efficient frequency estimation. In *Proc. Intl. Conf. Acoust., Speech, Signal Processing*, (1988), pp. 2292–2295.
3. Lang, S. W., and Musicus, B. R. Frequency estimation from phase differences. In *Proc. Intl. Conf. Acoust., Speech, Signal Processing*, (1989), pp. 2140–2143.
4. Fiore, P. D., and Lang, S. W. Efficient phase-only frequency estimation. In *Proc. Intl. Conf. Acoust., Speech, Signal Processing*, (1996), Vol. 5, pp. 2809–2812.

5. Tufts, D. W., and Fiore, P. D. Simple, effective estimation of frequency based on Prony's method. In *Proc. Intl. Conf. Acoust., Speech, Signal Processing*, (1996), Vol. 5, pp. 2801–2804.
6. Crochiere, R. E., and Rabiner, L. R. *Multirate Digital Signal Processing*. Prentice Hall, New York, (1983).
7. Vaidyanathan, P. P. Multirate digital filters, filter banks, polyphase networks, and applications: A tutorial. *Proc. IEEE* **78** (1990).
8. Lim, J. S., and Oppenheim, A. V. *Advanced Topics in Signal Processing*. Prentice Hall, New York, (1988).
9. McClellan, J. H., Parks, T. W., and Rabiner, L. R. A computer program for designing optimum FIR linear phase digital filters. *IEEE Trans. Audio Electroacoust.* **21** (1973), 506–526.
10. Blahut, R. E. *Fast Algorithms for Digital Signal Processing*. Addison–Wesley, Reading, MA, (1985).
11. White, S. A. Applications of distributed arithmetic to digital signal processing: A tutorial review. *IEEE ASSP Magazine* **6** (1989).
12. Mintzer, L. Large FFTs in a single FPGA. In *Intl. Conf. on Signal Processing Applications and Technology*, (1995), pp. 895–899.
13. Harris, F., and Dick, C. Design and application of versatile feedback requantizer with large fractional bandwidth, arbitrary center frequency, and real or complex inputs and outputs. In *Proc. Asilomar Conf. on Signals, Systems, and Computers*, (1996).

PAUL D. FIORE received the B.S. degree in computer system engineering from the University of Massachusetts at Amherst in 1986 and the M.S. degree in electrical engineering from Tufts University in 1988; he is currently pursuing a Ph.D in electrical engineering at the Massachusetts Institute of Technology. He is currently employed by the signal processing center of Sanders, a Lockheed–Martin Company in Nashua, NH. His interests are fast DSP algorithms, approximation theory, number theory, computer architecture, and the application of mathematics to the problems of DSP system optimization. He is a member of IEEE Signal Processing Society.