[7] M. R. Petraglia and S. K. Mitra, "Performance analysis of adaptive filter structures based on subband decomposition," in *Proc. ISCAS*, 1993, pp. 60–63.
[8] D. R. Morgan and J. C. Thi, "A delayless subband adaptive filter architecture," *IEEE Trans. Signal Processing*, vol. 43, pp. 1819–1830, 1995.
[9] R. Merched, P. S. R. Diniz, and M. R. Petraglia, "A delayless alias-free subband adaptive filter architecture," in *Proc. ISCAS*, 1997, pp. 2329–2332.
[10] M. Harteneck, J. M. Páez-Borrallo, and R. W. Stewart, "An oversampled subband adaptive filter without cross adaptive filters," *Signal Process.*, pp. 93–101, 1998.
[11] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*. Englewood Cliffs, NJ: Prentice-Hall, 1993.
[12] V. P. Sathe and P. P. Vaidyanathan, "Effects of multirate systems on the statistical properties of random signals," *IEEE Trans. Signal Processing*, vol. 41, pp. 131–146, 1993.
[13] Y. Ono and H. Kiya, "Performance analysis of subband adaptive systems using an equivalent model," in *Proc. ICASSP*, vol. III, 1994, pp. 53–56.
[14] M. J. T. Smith and T. P. Barnwell, III, "Exact reconstruction techniques for tree-structured subband coders," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 434–441, 1986.

Efficient Interpolators and Filter Banks using Multiplier Blocks

A. G. Dempster and N. P. Murphy

***Abstract*—We examine the use of efficient shift-and-add multiplier structures and multiplier blocks to reduce computational complexity in filter banks. This is more efficient than treating each bank filter separately. We also examine the Farrow structure, which is used in interpolators. Applying multiplier blocks makes this structure cheaper than the more recognized Lagrange interpolator.**

I. Introduction

In this section, we review existing work on multiplier blocks and the Farrow structure. We introduce the application of multiplier blocks to filter banks in general in Section II and to the Farrow structure in particular in Section III. Section IV examines an example of applying these methods.

A. Multiplier Blocks

A multiplier block is a network of shifts and adders that performs multiplication by one or more coefficients very efficiently (using few adders) by exploiting redundancy between multipliers. The problem of reducing the number of adders used in a shift-and-add multiplication process has been widely examined. For a single multiplication, we devised a method guaranteeing minimum adders for short wordlengths [1] and the most efficient algorithms for longer wordlengths [2], [3]. Where several products of a single multiplicand are required, multiplier blocks can exploit redundancy between multipliers, requiring far fewer adders [4], [5]. The transposed direct form of the FIR filter benefits greatly from using multiplier blocks: more than for other leading techniques [6]. However, several IIR structures are also suitable for multiplier block use, and significant savings can be made [7]–[10]. In this correspondence, we examine the savings when banks of FIR filters use multiplier blocks to perform coefficient multiplications.

B. Filter Banks and the Farrow Structure

Filter banks (parallel connections of digital filters) are used in many signal processing applications including fractional delay filter design. The "Farrow structure" [11] has been used in this application. An ideal fractional delay filter (which delays its input by a nonintegral number of samples of, say, D) has frequency response

$$H(e^{j\omega}) = e^{-j\omega D} \tag{1}$$

or impulse response

$$h(n) = \text{sinc}(n - D) \qquad \text{for all } n \tag{2}$$

which is infinite and noncausal and, hence, must be approximated. A popular and effective method of approximating this impulse response uses an FIR filter performing Lagrange interpolation, which has coefficients [12]

$$h(n) = \prod_{\substack{k=0 \\ k \neq n}}^{N} \frac{D-k}{n-k} \qquad \text{for } n = 0,\ 1,\ 2,\ \cdots,\ N. \tag{3}$$

This works well for constant delay D, but the coefficients must be recalculated each time D changes. The aim of Farrow's method [11] is to avoid this recalculation. If d is the fractional part of D (i.e., $D = l+d$ for integer l, and $0 \leq d < 1$), then a polynomial of order P in d is used to approximate each coefficient in (2)

$$h_d(n) = \sum_{m=0}^{P} c_m(n) d^m \tag{4}$$

which leads to transfer function

$$\begin{aligned} H_d(z) &= \sum_{n=0}^{N} h_d(n) z^{-n} \\ &= \sum_{n=0}^{N} \left[\sum_{m=0}^{P} c_m(n) d^m \right] z^{-n} \\ &= \sum_{m=0}^{P} \left[\sum_{n=0}^{N} c_m(n) z^{-n} \right] d^m \\ &= \sum_{m=0}^{P} C_m(z) d^m \end{aligned} \tag{5}$$

where $C_m(z) = \sum_{n=0}^{N} c_m(n) z^{-n}$. This structure can be implemented as in Fig. 1, i.e., a filter bank. Importantly, the coefficients of the filters are constant, regardless of the delay d.

The Farrow structure can be used to replace single-filter interpolators such as the Lagrange interpolator. In fact, because the Lagrange interpolator itself uses a polynomial method of approximation, the Farrow structure can exactly replicate the single Lagrange interpolator filter with $N = P$ [13].

Typically, the Farrow structure is only a useful replacement for a single filter, where the delay is different for each output sample [12] because its fixed coefficients save computation complexity. Obviously, for a constant delay, one filter requires less effort than P filters, and therefore, the Farrow structure is less attractive. However, another useful application of the Farrow structure that we propose here is that of a regular interpolator, i.e., a structure that produces interpolated

Manuscript received July 22, 1998; revised June 25, 1999. The associate editor coordinating the review of this paper and approving it for publication was Dr. Sergios Theodoridis.

The authors are with the Department of Electronic Systems, University of Westminster, London, U.K. (e-mail: dempsta@cmsa.wmin.ac.uk).

Publisher Item Identifier S 1053-587X(00)00081-7.

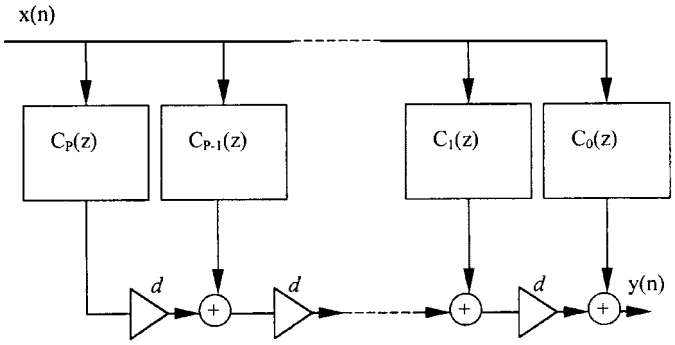


Fig. 1. Farrow structure for polynomial approximation of filter coefficients.

outputs at several (fixed) delay intervals. In multirate signal processing applications, where a signal is being "upsampled" by a factor M, a regular interpolator or "expander" is used to reconstruct signals. In such a structure, the lower part of Fig. 1 (the constant multipliers and sample delay registers) needs to be replicated M times: once for each delay value. However, only one Farrow filter bank is required. Such a structure is shown in Fig. 2. Although the d_i "delays" are fixed, there are several of them, and the extra computations required by the Farrow structure are effectively amortized across the range of delays.

II. Filter Bank Structures Using Multiplier Blocks

In order to use multiplier blocks, we require a structure that performs several multiplications of a single multiplicand. The more multiplications that can be grouped this way, the greater the saving will be [4], [5], [7]–[10]. Fig. 3 shows that for a simple filter bank of FIR filters, all of the coefficients in the filter bank can be incorporated into a single block. Fig. 3(a) is the usual direct-form filter bank. If we remove link A and transpose each separate filter, we get the structure in Fig. 3(b) (without link B). Add link B, and all the coefficients multiply a single data input and can be placed in a single multiplier block, as indicated.

Fig. 3(b) is not canonic in delays. There are NP ($N =$ filter order, $P =$ Farrow polynomial order) delays in the structure, whereas a structure with only N delays is shown in Fig. 4, which has P multiplier blocks, each of $(N+1)$ coefficients. The tradeoff between the number of blocks and the number of delays is analogous to the same tradeoff that occurred when examining direct forms I and II for IIR filters [7], [8]. This structure is the result of starting with the tapped delay line structure for the filter bank (i.e., N delays, each of which is tapped for each filter), transposing, blocking, and transposing again. The final transposition is important in order to produce the outputs $y_i(n)$. However, the structure appears to have i) not all products of a single multiplicand, and ii) the structural adders embedded in the multiplier block. Neither of these facts presents a problem; in fact, each filter in the bank is the same as if it were designed using multiplier blocks as in [4] and [5] and was simply transposed after the design, leaving the basic operation of the filter unchanged. Transposition following the multiplier block design was proposed in the original paper on the subject [16]. In such cases, the total number of adders (i.e., structural adders plus multiplier-block adders) remains the same before and after transposition.

III. Farrow Structures Using Multiplier Blocks

The Farrow structure of (5) is shown in Fig. 1, which is a bank of filters, the outputs of which are summed after multiplication by powers of d and implemented as identical d multipliers. We aim to minimize the number of numerical operations performed in such structures, using

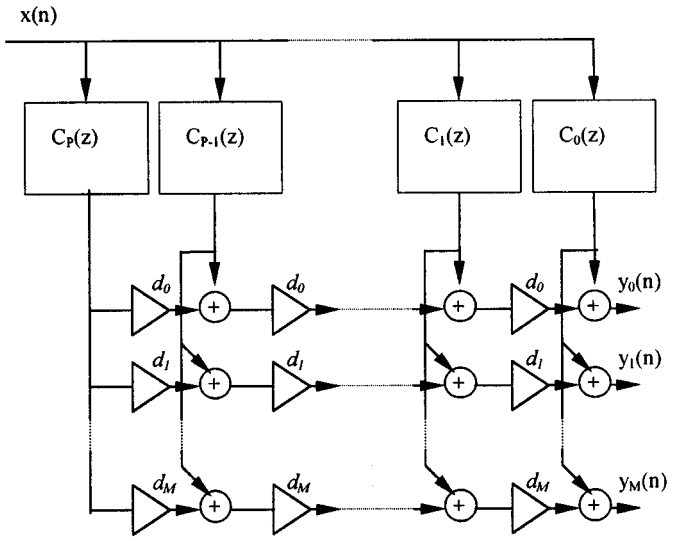


Fig. 2. Farrow structure used to generate several constant delays, as in a regular upsampler or "expander."

the method in the previous section. Previous authors have used similar simplifications for specific simple examples of Farrow filter banks [13], [15].

If the Farrow structure is used to implement a regular grid of delay points, an appropriate structure is that in Fig. 2, where it can be seen that a multiplier block could replace the M products of the output of $C_P(z)$. Further simplifications result from network manipulations. First, the delay value multipliers can be drawn back through the delay registers, as in Fig. 5, to more directly reflect (5). The output of filter p in the bank is multiplied by a coefficient set $\{d_0^p, d_1^p, \cdots, d_{M-1p}^p, d_M^p\}$, which can be implemented as a multiplier block. There is then a large Farrow bank filter multiplier block of $(P+1)(N+1)$ elements plus P blocks [not $P+1$ because the $C_0(z)$ output requires only coefficient value 1], each of $M-1$ coefficients (not M because the zero-delay filter has coefficients of 0, except for one coefficient 1). We call this the *Farrow block I*.

Alternatively, if we transpose just the y_0 filter of Fig. 5, we obtain Fig. 6(a), with a different block of coefficients. Designing the multiplier block and transposing again for all filters gives Fig. 6(b), with structural adders having "disappeared" inside the multiplier blocks, as in Fig. 4. Effectively, output m is attached to a block producing the coefficient set $\{d_m, d_m^2, \cdots, d_m^{P-1}, d_m^P\}$. In this configuration, there is the large Farrow bank filter multiplier block, of $(P+1)(N+1)$ elements, plus $M-1$ blocks, each of P. We call this *Farrow block II*.

IV. Design Examples

A. Experiment Description

In multirate signal processing applications, where a signal is being "upsampled" by a factor M, signals can be reconstructed by a regular interpolator or "expander," which can be implemented as a filter bank. We examined several cases of this type of filter:

In multirate signal processing applications, where a signal is being "upsampled" by a factor M, signals can be reconstructed by a regular interpolator or "expander," which can be implemented as a filter bank. We examined several cases of this type of filter:

i) implementation as single Lagrange interpolator filters, each producing one interpolation point implemented as in Fig. 4 (we examine this type of filter in detail in [14]);

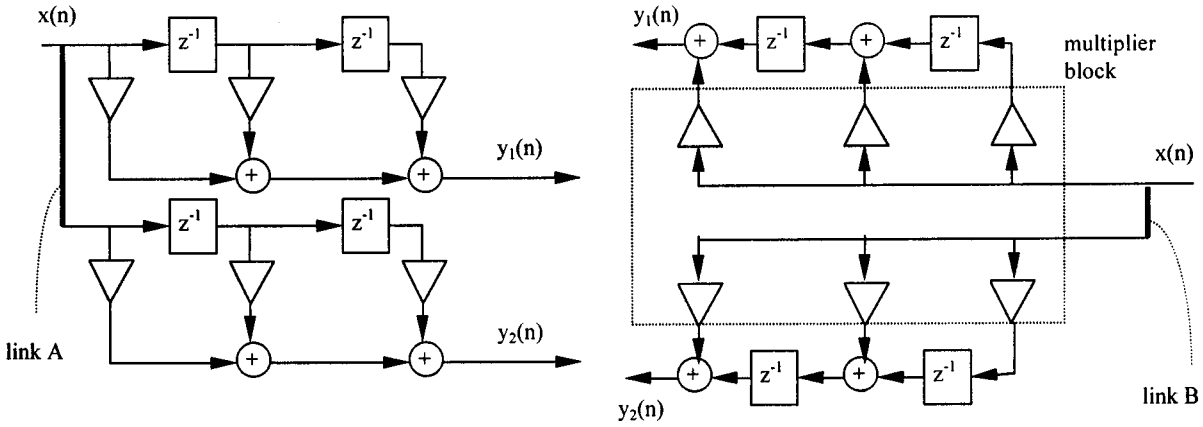


Fig. 3. How to incorporate all coefficients of a filter bank into a single block.

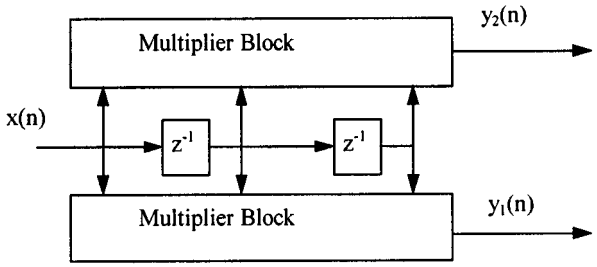


Fig. 4. Alternative multiplier block arrangement for a filter bank.

ii) implementation of Lagrange interpolation filters, with all coefficients implemented as a single multiplier block as in Fig. 3(c);
iii) implementation as Farrow structure filters as in the bank in Fig. 4;
iv) implementation as a Farrow structure filter bank, using both the structures of Fig 3(c)..

For the two Farrow cases iii) and iv), the delay multiplier section, which can be dealt with separately, was designed using Farrow blocks I and II of Figs. 5 and 6(b).

The example has Lagrange order 3, upconversion rate 8, and wordlength 12 bits. The Farrow coefficients for a third-order Lagrange filter are

$$F(n,\,m) = \begin{bmatrix} -0.0625 & 0.0417 & 0.2500 & -0.1667 \\ 0.5625 & -1.1250 & -0.2500 & 0.5000 \\ 0.5625 & 1.1250 & -0.2500 & -0.5000 \\ -0.0625 & -0.0417 & 0.2500 & 0.1667 \end{bmatrix}$$

where $F(n,\,m) = c_m(n)$, as used in (5). In other words, each row of F is used to generate a single coefficient of the Lagrange filter. The delays are spaced at one eighth sample intervals starting at the midpoint (1.5) less half a sample, i.e., the delay values D that are entered into (3) are 1.0000, 1.1250, 1.2500, 1.3750, 1.5000, 1.6250, 1.7500, and 1.8750. The resulting coefficients for the Lagrange filters are

$$L = \begin{bmatrix} 0.0000 & 1.0000 & 0.0000 & 0.0000 \\ -0.0342 & 0.9229 & 0.1318 & -0.0205 \\ -0.0547 & 0.8203 & 0.2734 & -0.0391 \\ -0.0635 & 0.6982 & 0.4189 & -0.0537 \\ -0.0625 & 0.5625 & 0.5625 & -0.0625 \\ -0.0537 & 0.4189 & 0.6982 & -0.0635 \\ -0.0391 & 0.2734 & 0.8203 & -0.0547 \\ -0.0205 & 0.1318 & 0.9229 & -0.0342 \end{bmatrix}$$

where each row of L is a set of filter coefficients. Note that the first row is an impulse response because there is no need for interpolation, and the fifth row matches the first column of F because this is the zero delay case in the middle of the filter.

The coefficients in the Farrow blocks I and II are derived from the values of $d(-0.5000 \cdots 0.3750)$ that correspond to the D values $(1.0000 \cdots 1.8750)$ used to generate L. The coefficients can also be represented in an array with $\mathrm{Del}(m,\,p-1) = d_m^p$ as used in Figs. 5 and 6. Farrow block I coefficients can be read along the columns, and Farrow block II can be read along the rows as in

$$Del = \begin{bmatrix} 1.0000 & -0.5000 & 0.2500 & -0.1250 \\ 1.0000 & -0.3750 & 0.1406 & -0.0527 \\ 1.0000 & -0.2500 & 0.0625 & -0.0156 \\ 1.0000 & -0.1250 & 0.0156 & -0.0020 \\ 1.0000 & 0.0000 & 0.0000 & 0.0000 \\ 1.0000 & 0.1250 & 0.0156 & 0.0020 \\ 1.0000 & 0.2500 & 0.0625 & 0.0156 \\ 1.0000 & 0.3750 & 0.1406 & 0.0527 \end{bmatrix}.$$

Note that columns 1 and row 5 are effectively "free" since no multiplications are required.

In order to get fixed-point coefficients of wordlength w, the relevant coefficients in L and Del, all of which are less than 1 (a coefficient of 1 requires no multiplier), are multiplied by 2^w (in this case 4096) and rounded. The coefficients in F, some of which exceed 1, are multiplied by $2^{w-\lceil \log_2(\max_{n,\,m}(F))\rceil}$ (in this case 2096) and rounded. Multiplier blocks can then be designed using the RAG-n algorithm [4].

We now consider costs of the structural elements not associated with multiplication. For either type of Lagrange filter, there are MN adders. There are N delays when using separate filters and MN delays when all coefficients are in a single block. For the Farrow structure, there are $(P+1)N\{= N^2 + N$ because we are generating Lagrange coefficients$\}$ adders in the filter bank and PM adders in the delay blocks I and II (one adder can be subtracted from this total for each zero-valued coefficient). When using separate filters, there are only N delays, and when all coefficients are in a single block, there are $(P+1)N$ delays.

B. Results

The adder costs are shown in Table I. The total cost for each of the implementations has been broken down into the adders due to multipliers; structural elements (adders and delays); and, for the Farrow structures, the delay blocks I and II. Delay elements cost 0.2 adders,

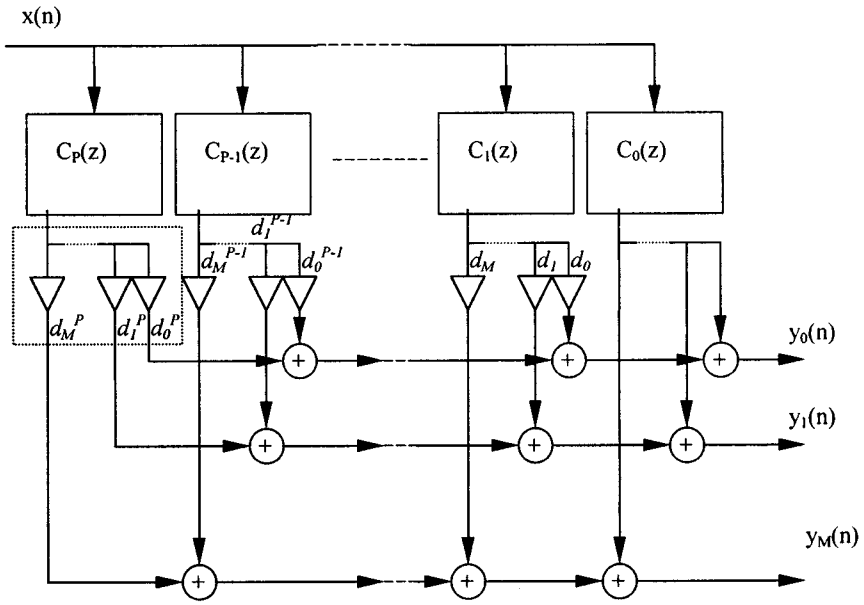
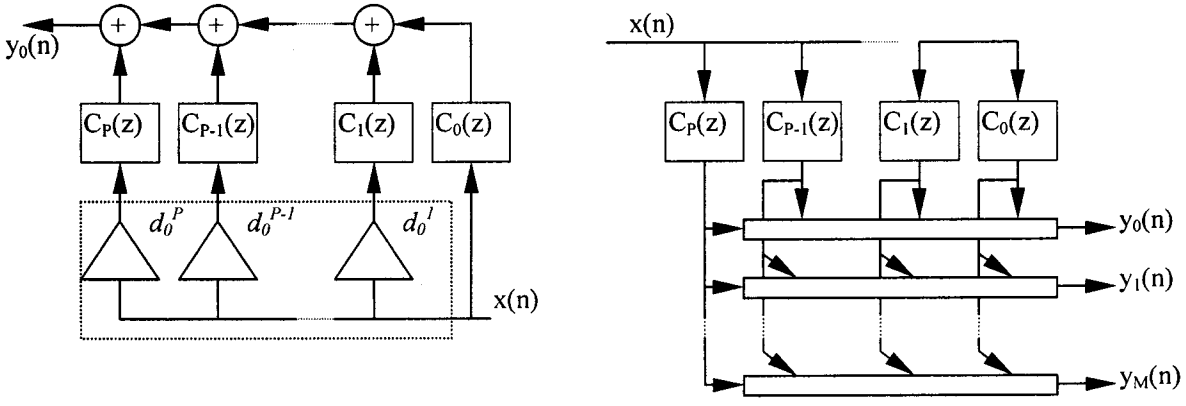


Fig. 5. Farrow structure for several constant delay outputs. Note that the outputs of each bank filter are then multiplied by several coefficients, which can be gathered into multiplier blocks. The highest order coefficient block is shown dotted (Farrow block I).



(a) (b)

Fig. 6. (a) Single delay filter within the Farrow structure transposed so that the coefficients (those within the dotted region) can be implemented in a multiplier block. (b) Whole filter with several of these stages, retransposed (Farrow block II).

TABLE I
ADDER COSTS FOR THE FOUR TYPES OF FILTER DESIGNS USED IN THE EXAMPLE: i) LAGRANGE FILTERS WITH COEFFICIENTS BLOCKED FOR EACH INDIVIDUAL FILTER. ii) LAGRANGE FILTERS WITH ALL COEFFECIENTS IN THE BANK BLOCKED. iii) INDIVIDUALLY BLOCKED FARROW STRUCTURE. iv) BANK-BLOCKED FARROW STRUCTURE

Config.	Multipliers	Struct Adders	Struct Delays	Farrow Block I	Farrow Block II	TOTAL Adder Cost
Lagrange ind	33	24	3	-	-	57.6
Lagrange bank	13	24	24	-	-	41.8
Farrow ind	7	33	3	4	6	44.6
Farrow bank	4	33	12	4	6	43.4

which is consistent with earlier work (e.g., [6]). Of the two delay blocks (Farrow blocks I and II), only the least costly is used for the total.

These results reveal some interesting things.

1) For both the Lagrange and Farrow implementations, the implementation with all coefficients in the bank in a single block is less costly. In both cases, the gains made by including all coefficients in a single block outweigh the penalty of using a few more delay elements. This is consistent with earlier results [4], [7]–[9].
2) For this example, the Lagrange bank filter is the cheapest. However, despite the extra circuitry required by the Farrow structure, it is not far behind. In work yet to be published, we have found that the Farrow structure is better in some circumstances.

REFERENCES

[1] A. G. Dempster and M. D. Macleod, "Constant integer multiplication using minimum adders," in *Proc. Inst. Elect. Eng.—Circuits, Devices, Syst.*, vol. 141, Oct. 1994, pp. 407–413.

[2] ——, "General algorithms for reduced-adder integer multiplier design," *Electron. Lett.*, vol. 31, no. 21, pp. 1800–1802, Oct. 1995.

[3] ——, "Comments on 'Minimum number of adders for implementing a multiplier and its application to the design of multiplierless digital filters'," *IEEE Trans. Circuits Syst. II*, vol. 45, pp. 242–243, Feb. 1998.

[4] ——, "Use of minimum-adder multiplier blocks in FIR digital filters," *IEEE Trans. Circuits Syst. II*, vol. 42, pp. 569–577, Sept. 1995.

[5] ——, "Use of multiplier blocks to reduce filter complexity," in *Proc. ISCAS*, London, U.K., 1994.

[6] ——, "Comparison of fixed point FIR filter design techniques," *IEEE Trans. Circuits Syst. II*, vol. 44, pp. 591–593, July 1997.

[7] ——, "Multiplier blocks and the complexity of IIR structures," *Electron. Lett.*, vol. 30, no. 22, pp. 1841–1842, Oct. 1994.

[8] ——, "Comparison of IIR filter structure complexities using multiplier blocks," in *IEEE Proc. ISCAS*, vol. 2, Apr./May 1995, pp. 858–861.

[9] A. G. Dempster, "The cost of limit-cycle elimination in IIR digital filters using multiplier blocks," in *Proc ISCAS*, vol. 4, June 1997, pp. 2204–2207.

[10] A. G. Dempster and M. D. Macleod, "IIR digital filter design using minimum-adder multiplier blocks," *IEEE Trans. Circuits Syst. II*, vol. 45, pp. 761–763, June 1998.

[11] C. W. Farrow, "A continuously variable digital delay element," in *Proc. Int. Symp. Circuits Syst. (ISCAS)*, pp. 2641–2645.

[12] T. I. Laakso *et al*, "Splitting the unit delay," *IEEE Signal Processing Mag.*, vol. 13, pp. 30–60, Jan. 1996.

[13] V. Valimaki, "A new filter implementation strategy for Lagrange interpolation," in *Proc. ISCAS*, 1995, pp. 361–364.

[14] A. G. Dempster and N. P. Murphy, "Lagrange interpolators and binomial windows," *Signal Process.*, vol. 76, no. 1, pp. 81–91, July 1999.

[15] E. Hermanowicz, "Weighted Lagrangian interpolating FIR filter," in *Proc. EUSIPCO, Signal Process. VIII*, 1996, pp. 1203–1206.

[16] D. R. Bull and D. H. Horrocks, "Primitive operator digital filters," *Proc. Inst. Elect. Eng. G*, vol. 138, no. 3, pp. 401–412, June 1991.

On the Relationship Between the Overlapping Rounding Transform and Lifting Frameworks for Reversible Subband Transforms

Michael D. Adams and Faouzi Kossentini

***Abstract*—Recently, a new framework for reversible subband transforms based on the overlapping rounding transform (ORT) has been proposed as an alternative to the lifting framework. In this correspondence, we show that the ORT framework is, in fact, a special case of the lifting framework with only trivial extensions.**

***Index Terms*—Lifting, overlapped rounding transform, reversible integer-to-integer subband transforms.**

I. INTRODUCTION

In order to efficiently handle lossless coding in subband coding systems, we require transforms that are invertible in finite-precision arithmetic. Such transforms are said to be reversible. In [1], Calderbank *et al.* showed that the lifting scheme [2] forms an effective framework for constructing reversible transforms. Transforms utilizing this framework have since found application in numerous coding systems, including that of the emerging JPEG-2000 standard [3], [4]. More recently, Jung and Prost [5] have proposed an alternative method for constructing reversible transforms based on the overlapping rounding transform (ORT). Although the ORT and lifting frameworks appear quite different at first glance, they are, in fact, intimately related. In what follows, we will show that the ORT framework is, in fact, a special case of the lifting framework with only trivial extensions.

II. NOTATION AND OTHER PRELIMINARIES

Before proceeding further, a short digression concerning the notation used in this correspondence is appropriate. The symbols $\mathbb{Z}$ and $\mathbb{R}$ denote the sets of integer and real numbers, respectively. Matrix and vector quantities are indicated using bold type. For $x \in \mathbb{R}$, the notation $\lfloor x \rfloor$ denotes the largest integer not more than x (i.e., the floor function), and the notation $\lceil x \rceil$ denotes the smallest integer not less than x (i.e., the ceiling function). The z-transform of a sequence $x[n]$ is denoted $X(z)$. For convenience, we also define the quantities

$$\lfloor X(z) \rfloor_z \triangleq \sum_{n \in \mathbb{Z}} \lfloor x[n] \rfloor z^{-n} \quad \text{and} \quad \lceil X(z) \rceil_z \triangleq \sum_{n \in \mathbb{Z}} \lceil x[n] \rceil z^{-n}$$

(i.e., $\lfloor x[n] \rfloor \leftrightarrow \lfloor X(z) \rfloor_z$ and $\lceil x[n] \rceil \leftrightarrow \lceil X(z) \rceil_z$). Lastly, we note some simple yet important properties of floor and ceiling operations. For $x \in \mathbb{Z}$ and $\gamma \in \mathbb{R}$, the following identities hold:

$$\lfloor x + \gamma \rfloor = x + \lfloor \gamma \rfloor \tag{1}$$

$$\lceil x + \gamma \rceil = x + \lceil \gamma \rceil \tag{2}$$

and

$$\lfloor -\gamma \rfloor = -\lceil \gamma \rceil. \tag{3}$$

Manuscript received October 6, 1998; revised June 15, 1999. This work was supported by the Natural Sciences and Engineering Research Council of Canada. The associate editor coordinating the review of this paper and approving it for publication was Dr. Shubha Kadambe.

The authors are with the Signal Processing and Multimedia Group, Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, B.C., Canada V6T 1Z4 (e-mail: mdadams@ece.ubc.ca).

Publisher Item Identifier S 1053-587X(00)00105-7.