One has

$$\sum_{\substack{i=1\\ i\in[j:\, Q(e_j)=0]}}^{t} |e_i| = \sum_{i=1}^{t} |e_i| - \sum_{\substack{i=1\\ i\in[j:\, Q(e_j)=0]}}^{t} |e_i|. \tag{5}$$

For some $r_1 \geq 0, r_2 \geq 0$, let the interval $[-r_1, r_2]$ be the dead zone of the quantizer. Then,

$$-r_1 < e_i < r_2 \text{ if and only if } i \in [j\colon Q(e_j) = 0]. \tag{6}$$

(For a quantizer without a dead zone, $r_1 = r_2 = 0$.) Equations (5) and (6) imply that

$$\sum_{\substack{i=1\\ i\in[j:\, Q(e_j)=0]}}^{t} |e_i| \geq \sum_{i=1}^{t} |e_i| - \max(r_1, r_2)t. \tag{7}$$

Inserting (7) in (4), then

$$\operatorname*{Lim}_{t\to\infty} \frac{1}{t}\sum_{i=1}^{t} |e_i| \leq \max(r_1, r_2) + \frac{n\mu\beta^2 M^2}{2q}. \tag{8}$$

Thus, (3.5) holds with $\delta = \max(r_1, r_2) + (n\mu\beta^2 M^2/2q)$. Equation (8) indicates that the mean absolute error of the QE algorithm increases with the width of the quantizer dead zone, as is intuitively expected. It is worth mentioning that (8) is satisfied by the example given in Section II. Indeed, as shown at the end of this example, $\operatorname{Lim}_{t\to\infty} (1/t)\Sigma_{i=1}^{t} |e_i| = 1/8$. Due to (3) and (6), $r_1 = r_2 = 1/4$. Hence, (8) is satisfied.

References

[1] W. A. Sethares and C. R. Johnson, "A comparison of two quantized state adaptive algorithms," *IEEE Trans. Acoust. Speech Signal Processing*, vol. 37, no. 1, pp. 138–143, Jan. 1989.

[2] P. Xue and B. Liu, "Adaptive equalizer using finite-bit power-of-two quantizer," *IEEE Trans. Acoust. Speech Signal Processing*, vol. ASSP-34, no. 6, pp. 1603–1611, Dec. 1986.

Reply to "Comments on 'A Comparison of Two Quantized State Adaptive Algorithms'"

W. A. Sethares and C. R. Johnson, Jr.

Dr. Eweda is correct in noting that our Theorem 1 does not apply to quantizers that incorporate a dead zone. However, the theorem, as stated in our paper, does not claim to apply to such quantizers.

Theorem 1 begins "Consider the Quantized Error algorithm (1.7)..." Equation (1.7) describes the quantized error algorithm, and the surrounding text explains the notation used in the algorithm. For instance, the regressor vector X_k, the prediction error e_k, and the quantizer $Q(\cdot)$) are all defined here and are used in the theorem and its proof. The first sentence after (1.7) says "In (1.7), $Q(\cdot)$ represents some quantization function: a bounded, discrete valued, element by element monotonic nondecreasing function *which does not change the sign of the argument*" (italics added). Quantization functions with dead zones, such as those in above comments, do change the sign of their arguments. To be specific, the *sign* function is usually defined to have three possible values: $+1$ when its argument is positive, 0 when its argument is zero, and -1 when its argument is negative. Dead zones typically map a region about 0 to 0, thus changing the sign from positive to zero, or from negative to zero. Thus, the quantization functions allowable in Theorem 1 do not include those with dead zones, contrary to the claims in the above comments.

In fact, dead zones are mentioned only once in our paper, inside the proof of Theorem 4, where we note that Theorem 4 also holds for dead zone quantizers. No such comment is made in or about Theorem 1. We welcome Dr. Eweda's result as an extension of our Theorem 1, but we deny that the theorem is in error.

Manuscript received May 25, 1993; revised May 25, 1993. The associate editor coordinating the review of this paper and approving it for publication was Dr. Hong Fan.

W. A. Sethares is with the Department of Electrical Engineering, University of Wisconsin, Madison, WI 53706-1691.

C. R. Johnson, Jr. is with the School of Electrical Engineering, Cornell University, Ithaca, NY 14853.

IEEE Log Number 9214637.

The Recursive Pyramid Algorithm for the Discrete Wavelet Transform

Mohan Vishwanath

Abstract—**The recursive pyramid algorithm (RPA) is a reformulation of the classical pyramid algorithm (PA) for computing the discrete wavelet transform (DWT). The RPA computes the N-point DWT in real time (running DWT) using just $L(\log N - 1)$ words of storage, as compared with $O(N)$ words required by the PA. L is the length of the wavelet filter. The RPA is combined with the short-length FIR filter algorithms to reduce the number of multiplications and additions.**

I. Introduction

The discrete wavelet transform (DWT) can be computed in a efficient manner on general purpose computers using the pyramid algorithm (PA) developed by Mallat [1], [2]. There are a number of applications of the DWT where a "running" (or real time) implementation is desirable. Running implementations of the PA, for an N-point DWT, requires either $O(N)$ storage or $\log N$ filters cascaded together. Both these alternatives are expensive. We present a reformulation of the pyramid algorithm called the recursive pyramid algorithm (RPA). The RPA computes the N-point DWT in real time using just $L\log N - L$ cells of storage, where L is the length of the wavelet filter (QMF) and, generally, $L \ll N$. It computes the DWT in N steps (same as the PA) and the number of operations (multiplications+additions) required is comparable with that for the PA [3]. We show how the RPA can be computed using the short-length FIR algorithms described in [4]. This further reduces the number of operations required.

Manuscript received October 9, 1992; revised February 7, 1993. The associate editor coordinating the review of this paper and approving it for publication was Prof. Sergio D. Cabrera.

The author was with the Department of Computer Science, The Pennsylvania State University, University Park, PA 16802. He is now with the Computer Science Laboratory, Xerox Palo Alto Research Center, Palo Alto, CA 94304.

IEEE Log Number 9214639.

We first present a brief introduction to the DWT. We describe the RPA in Section III, whereas Section IV contains extensions to the RPA that allow it to be computed using the short-length FIR algorithms. Section IV also contains some operation counts that compare the RPA against the direct PA.

II. The Discrete Wavelet Transform

The DWT can be looked at as the multiresolution decomposition of a sequence [1], [2]. It takes a length N sequence $x(n)$ as input and generates a length N sequence as the output. The output has $N/2$ values at the highest resolution and $N/4$ values at the next resolution, and so on, that is, the frequency resolution is low at the high frequencies and high at the low frequencies, whereas the time resolution is high at the higher frequencies and low at the lower frequencies. Let $N = 2^J$, and let the number of frequencies, or resolutions, be J, i.e., we are considering $J = \log N$ octaves. Therefore, the frequency index j varies as $1, 2, \ldots, J$ corresponding to the scales $2^1, 2^2, \ldots, 2^J$. The pyramid algorithm as described by Mallat [1], [2] can be succinctly represented as

$$W(j,n) = \sum_{m=0}^{L-1} W(j-1, 2n-m)w(m) \quad \text{Low Pass o/p} \tag{1}$$

$$W_H(j,n) = \sum_{m=0}^{L-1} W(j-1, 2n-m)h(m) \quad \text{High Pass o/p} \tag{2}$$

where $n = 1, 2, \ldots, 2^{J-j}$, $j = 1, 2, \ldots, J$, $W(0,n) = x(n)$, and $w(m)$ and $h(m)$, $m = 0, \ldots, (L-1)$ are quadrature mirror filters derived from the wavelet. Note that the output of the DWT is actually contained in $W_H(j,n)$. W contains exactly $2N$ elements, where each row corresponds to an octave. Thus, the zeroth row contains N elements (the input), the first row contains $N/2$ outputs, the second contains $N/4$ outputs, and so on. W_H contains N elements, the first row contains $N/2$ outputs, the second contains $N/4$ outputs, and so on. Note that this "matrix" representation is used throughout this paper only as a notational convenience. In practice, the 2-D arrays (W and W_H) will be mapped onto a 1-D array (say, X and X_H), such that

$$W(j,n) = X\left(n + \left(2N - \frac{N}{2^{j-1}}\right)\right) \quad j = 0, \ldots, J$$
$$W_H(j,n) = X_H\left(n + \left(N - \frac{N}{2^{j-1}}\right)\right) \quad j = 1, \ldots, J.$$

This algorithm is pictorially shown in Fig. 1 (note the decimation by 2, which is reflected in (1) and (2) as the factor 2 in the index of W). This can be expressed by the following pseudocode:

```
begin{Direct Pyramid}
  input : x[1..N]
  for(j = 1 to log(N))
      Do the stage j filtering using output of
      stage (j-1) as input
end{Direct Pyramid}
```

A direct implementation of the above algorithm either requires $\log N$ filters (which is too expensive) operating in a pipelined fashion, or it requires one filter and $O(N)$ memory; again, this is too expensive for a 1-D running DWT. It is not practical to need storage proportional to the size of the quasi-infinite input sequence.

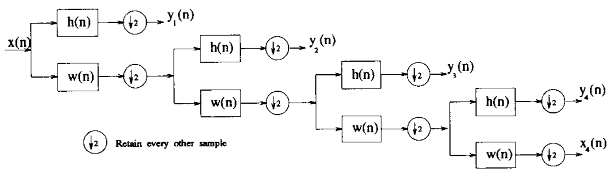

Fig. 1. DWT filter bank $J = 4$.

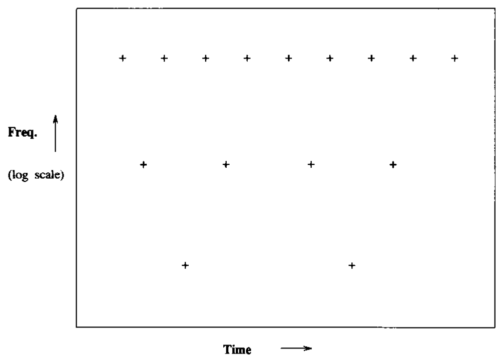

Fig. 2. Dyadic sampling grid for the DWT.

III. The Recursive Pyramid Algorithm

The goals the RPA is set out to satisfy are as follows:

- Real-time performance (running DWT)
- acceptance of input at a uniform, practical rate
- minimization of storage (keep storage independent of input/output size)
- keeping the operation count comparable to the PA.

It is a reformulation of the classical pyramid algorithm for the DWT and allows computation of the DWT in a real-time (running) fashion using just $L \log N - L$ cells of storage. It consists of rearranging the order of the N outputs such that an output is scheduled at the "earliest" instance that it can be scheduled. The "earliest" instance is decided based on a strict precedence relation, i.e., if the "earliest" instance of the ith octave clashes with that of the $(i+1)$st octave, then the ith octave output is scheduled. A simple way of obtaining this output schedule is to consider the sampling grid for the DWT output, which is shown in Fig. 2. Now, push (up or down) all the horizontal lines of samples until they form a single line. The order of the outputs obtained in this manner gives us the output schedule. The basic idea behind the RPA is to linearize the pyramid schedule without increasing the dependencies between stages (octaves in this case).

The pseudocode for the RPA is shown in Fig. 3. Note that the matrices W and W_H are used in the pseudocode for notational convenience and generality. The manner in which the $L(\log N - 1)$ words of storage is managed is dependent on the implementation. It could vary from a simple algorithmic solution of using $\log N - 1$ queues of size L each to a hardware solution of using a routing network of size $L(\log N - 1)$ [5], [6].

Before presenting the output schedule of the RPA, a few observations are in order; only the first octave outputs depend directly on the inputs (x). Due to the decimation by two, a new first octave output can be computed only for every two new inputs. One of the goals is to accept the input at a uniform and practical rate (by practical, we mean that the rate is ck, where k is the precision

```
begin{Recursive Pyramid}

  input: W(0,i)=x(i),  i:[1,N]    /* N is a power of 2 */

  for(i=1 to N)                   /* Once for each output */
     rdwt(i,1)

end{Recursive Pyramid}

rdwt(i,j)
begin{rdwt}

  if(i is odd)
    k=(i+1)/2                     /* Compute output number k of octave j.
    sumL=0                         This is computed using the last L outputs
    sumH=0                         of octave (j-1). */
    for(m=0 to (L-1))
      sumL=sumL+W[j-1,i-m]*w[m];
      sumH=sumH+W[j-1,i-m]*h[m];
    W[j,k]=sumL;                  /* Low pass output */
    H[j,k]=sumH;                  /* High pass output */


  else
    rdwt(i/2,j+1)                /* Recursion to determine correct octave */

end{rdwt}
```

Fig. 3. Recursive pyramid algorithm.

of the inputs/outputs, and c is a small constant). Given the above observations, it is clear that every other output scheduled must be a first octave output. Thus, the overall computation consists of interspersing the computation of all the octaves in between the first octave computations, and the total number of "computation" steps is $2\times$ (number of first octave outputs) $= 2\times\frac{N}{2} = N$. A "computation" step consists of computing one output of an FIR filter. This requires L multiplications and $L-1$ additions, if done directly. This is exactly the same as the number required by the direct implementation of the pyramid algorithm [3]. In the next section, we show how the RPA can be combined with the short-length FIR filtering algorithms to obtain a reduction in the number of operations.

The output schedule generated by the RPA for $N = 16$ and $J = 4$ is shown below; here, $y_i(n)$ is the lowpass output of the ith octave

$$\begin{gathered} y_1(1),\ y_2(1),\ y_1(2),\ y_3(1),\ y_1(3),\ y_2(2),\ y_1(4),\\ y_4(1),\ y_1(5),\ y_2(3),\ y_1(6),\ y_3(2),\ y_1(7),\ y_2(4),\ y_1(8) \end{gathered} \tag{3}$$

The high-pass output schedule is exactly the same. Thus an ith octave output is scheduled once every 2^i cycles. Note that the output schedule relies heavily on the decimation by two at each stage of the pyramid. This would not be possible with a polyphase (biphase, in this case) implementation of the decimated filters [3], [7].

The following are the main advantages of this algorithm:

- Since each output of the jth octave is scheduled at the "earliest" instance, only the latest L outputs of the $(j-1)$st octave need to be stored. Thus, a total of at most $L(\log N - 1)$ words of storage is required. Note that it is $L(\log N - 1)$ and not $L\log N$ since the last octave output need not be stored.
- Due to its structure, it is highly amenable to systolic and pipelined approaches [5], [6], [8].
- It produces the output in an order that is ideal for many applications like subband coding and transmultiplexers.

The main disadvantage of the RPA is that in general-purpose computers, the effort required to keep track of the latest $L\log N - L$ block of outputs might outweigh the reduction in storage requirements. However, the RPA is being presented mainly as an alternative for special-purpose architectures [5], [6], [8] and DSP chips. In these cases, the extra effort to keep track of the latest outputs pays off handsome dividends by allowing the DWT to be implemented on one chip. This means the output of an A/D converter can be fed directly to the DSP chip or DWT chip (special-purpose) without the need for any buffering. Note that there is an order of magnitude difference between on-chip and off-chip speeds.

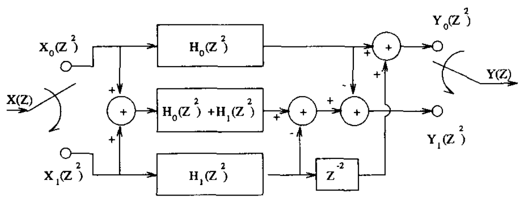

Fig. 4. Short-length filter algorithm $F(2,2)$ from [4].

IV. Incorporating Short-Length FIR Filtering Algorithms

Most "fast" filtering algorithms exploit the redundancy in the computation of a block of outputs to reduce the total number of operations. This implies that a block of input and/or output has to be buffered to utilize these algorithms. In addition, these algorithms do not retain the natural "running" multiply accumulate structure of FIR filtering [4]. The short-length FIR filtering algorithms [4] were introduced to avoid this extra storage requirement and to retain (at least partially) the multiply accumulate structure. We consider these algorithms because, in practice, the QMF filters used in the DWT are short, and these algorithms will allow us to retain the running nature of the RPA and, at the same time, reduce the number of operations per output.

We consider a specific case of the short-length filtering algorithm (the $F(2, 2)$) as described in [4]. This consists of converting the problem of computing two outputs of a length L filter into that of computing one output of three length $L/2$ filters at the cost of four adds per block of two outputs [4]. This is shown in Fig. 4 [4]. If $x(n)$, $y(n)$, and $h(n)$ are the input, output, and filter sequences, respectively, then the quantities in Fig. 4 are defined as follows [4]:

$$\begin{aligned} H_j(Z) &= \sum_{m=0}^{\frac{L}{2}-1} h(2m+j)Z^{-m}; \quad j = 0,1\\ X_k(Z) &= \sum_{m=0}^{\infty} x(2m+k)Z^{-m}; \quad k = 0,1\\ Y_i(Z) &= \sum_{m=0}^{\infty} y(2m+i)Z^{-m}; \quad i = 0,1. \end{aligned}$$

The number of multiplications and additions per output point are $\frac{3}{4}L$ and $\frac{3}{4}L + \frac{1}{2}$, respectively. The modified RPA is shown in Fig. 5. The first outputs of all the octaves have to be handled differently from the other outputs to maintain the recursive structure of the RPA and to prevent any "holes" in the output schedule. Each time the ith octave is scheduled, two outputs are produced, and hence, four outputs of the $(i-1)$st octave should be available. Thus, there is an increase of $2(\log N - 1)$ words in the storage requirement.

```
begin{Recursive Pyramid}

  input: x[i:1,N]                    /* N is a power of 2 */
  mask[i:1,log(N)]=0                 /* Mask to detect first output of
                                        octave i */

  P=(N/2 + log(N)/2)
  for(i=1 to P)                      /* Two outputs produced for each call using
     rdwt(i,1)                          F(2,2), except for the first outputs of
                                        the log(N) octaves. */
end{Recursive Pyramid}

rdwt(i,j)
begin{rdwt}

  if(i is odd)

     if(mask(j)=0)
       compute the first output of octave j.
       mask(j)=1

     else
       Compute outputs numbered (i-1) and i of
       octave j using F(2,2), shown in Figure 4.
       This is computed using the last (L+2) outputs
       of octave (j-1).

  else
    rdwt(i/2,j+1)                    /* Recursion to determine correct octave */

end{rdwt}
```

Fig. 5. Modified RPA.

The number of multiplications per output point (using $F(2,2)$) is given by

$$\begin{aligned}\frac{1}{N}&\left\{\frac{3L}{4}(N-\log N)+\log N\right\}\\ &=\frac{1}{N}\left\{\frac{3LN}{4}-\left(\frac{3L}{4}-1\right)\log N\right\}\\ &=\frac{3L}{4}-\left(\frac{3L}{4}-1\right)\frac{\log N}{N}.\end{aligned}$$

The number of additions per output point (using $F(2,2)$) is given by

$$\begin{aligned}\frac{1}{N}&\left\{\left(\frac{3L}{4}+\frac{1}{2}\right)(N-\log N)\right\}\\ &=\frac{3L}{4}+\frac{1}{2}-\left(\frac{3L}{4}+\frac{1}{2}\right)\frac{\log N}{N}.\end{aligned}$$

Since the first output of each octave depends only on the first output of the previous octave, it needs only one multiplication and no additions. Note that the direct implementation of the RPA requires L multiplications and $(L-1)$ additions per output.

Other short-length FIR algorithms and their nestings [4] can be also be used, and the RPA can be modified in a similar manner. If an $F(M,M)$ algorithm is used or if a nested algorithm ($F'(M,M)$) is used, where the product of the components is M [4], then there is an increase of $(2M-2)(\log N-1)$ words in the storage requirement. Note that $M \le L$ [4]; thus, the amount of storage will at most triple. Assume that the number of multiplications per output point for $F(M,M)$ or $F'(M,M)$ is $kL+p$, where $k<1$. Let the number of additions per output point be $qL+r$. Then, the number of operations required by the modified RPA will be given by

$$\text{Multiplications/output point: } (kL+p)-(kL+p-1)\frac{\log N}{N}$$
$$\text{Additions/output point: } (qL+r)-(qL+r)\frac{\log N}{N}.$$

TABLE I
COMPARISON OF THE RPA AND THE PA BASED ON OPERATION COUNTS ($N = 1024$ and the short-length FIR filter used is $F(2,2)$.)

Filter Length	PA and RPA: Direct		PA: Short-Length		RPA: Short-Length	
	Mults	Adds	Mults	Adds	Mults	Adds
4	4	3	3.00	3.50	2.98	3.47
6	6	5	4.00	4.50	3.97	4.46
8	8	7	6.00	6.50	5.95	6.44
10	10	9	7.00	7.50	6.94	7.43
12	12	11	9.00	9.50	8.92	9.41
16	16	15	12.00	12.50	11.89	12.38
20	20	19	15.00	15.50	14.86	15.35
24	24	23	18.00	18.50	17.83	18.32
30	30	29	22.00	22.50	21.79	22.28
32	32	31	24.00	24.50	23.78	24.26

A comparison of the number of operations required by the RPA and the PA is shown in Table I. For the sake of comparisons, only the $F(2, 2)$ algorithm is considered for all the filter lengths. Note that every output (high pass) has a corresponding low-pass value. Thus, the number of operations per output point in Table I should be doubled to get the correct value (for both the PA and the RPA).

V. CONCLUSION

The recursive pyramid algorithm (RPA) has been proposed as an alternative to the classical pyramid algorithm (PA) for computing the DWT. The RPA computes the DWT in real time using just $L(\log N-1)$ words of storage. It is ideally suited for special-purpose architectures and DSP chips. We showed that whereas on one hand the RPA reduces the storage requirements, it remains competitive with the classical pyramid algorithm as far as the number of operations is concerned. We also showed how the RPA can be combined with the short-length FIR algorithms to reduce the number of operations.

The RPA can be modified to compute a large range of QMF filter bank trees [6] like the Laplacian pyramid. The DWT and the inverse DWT (IDWT) can be computed in a symmetric manner using techniques similar to the RPA [8], i.e., linearizing the pyramid schedule. These algorithms use the same amount of storage and operations as the RPA. In addition, the reconstruction (IDWT) can be initiated with minimal latency [8] (the first output of the IDWT is available $2\log N$ cycles after first input to the DWT).

REFERENCES

[1] S. Mallat, "Multifrequency channel decompositions of images and wavelet models," *IEEE Trans. Acoust. Speech Signal Processing,* vol. 37, no. 12, pp. 2091–2110, Dec. 1989.

[2] ——, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Trans. Patt. Anal. Machine Intell.,* vol. 11, no. 7, pp. 674–693, July 1989.

[3] O. Rioul and P. Duhamel, "Fast algorithms for wavelet transforms," *IEEE Trans. Inform. Theory,* vol. 38, no. 2, pp. 569–586, Mar. 1992.

[4] Z. Mou and P. Duhamel, "Short-length FIR filters and their use in fast nonrecursive filtering," *IEEE Trans. Signal Processing,* vol. 39, no. 6, pp. 1322–1332, June 1991.

[5] M. Vishwanath, R. M. Owens, and M. J. Irwin, "VLSI architectures for the discrete wavelet transform," Tech. Rep., CS-93-05, Dept. of Computer Sci., Penn. State Univ., 1993.

[6] ——, "An efficient systolic architecture for qmf filter bank trees," *VLSI Signal Processing, V,* pp. 175–184, 1992.

[7] P. P. Vaidyanathan, "Quadrature mirror filter banks, m-band extensions and perfect-reconstruction techniques," *IEEE ASSP Mag.,* pp. 4–20, July 1987.

[8] C. Chakrabarti, M. Vishwanath, and R. M. Owens, "Architectures for wavelet transforms," Technical Report CS-93-06, Dept. of Comput. Sci., Penn. State Univ., 1993.