

Scalability of 2-D Wavelet Transform Algorithms: Analytical and Experimental Results on MPPs

Jamshed N. Patel, Ashfaq A. Khokhar, and Leah H. Jamieson, *Fellow, IEEE*

Abstract—This paper studies the scalability of two-dimensional (2-D) discrete wavelet transform (DWT) algorithms on massively parallel processors (MPPs). The principal operation in the 2-D DWT is the filtering operation used to implement the filter banks of the 2-D subband decomposition. This filtering operation can be implemented as a convolution in the time domain or as a multiplication in the frequency domain. We demonstrate that there exist combinations of the machine size, image size, and wavelet kernel size for which the time-domain algorithms outperform the frequency domain algorithms and vice-versa. We therefore demonstrate that a hybrid approach that combines time- and frequency-domain approaches can yield linear scalability for a broad range of problem and machine sizes. Furthermore, we show the effect of processor speed versus communication overhead and the use of separable versus nonseparable wavelets on the crossover points between the algorithm approaches.

Index Terms—Coarse-grained architectures, fine-grained architectures, image processing, intel paragon, Maspar MP, parallel algorithms, scalability analysis, wavelet transform.

I. INTRODUCTION

WAVELET theory provides a unified framework for a number of techniques developed in multiresolution analysis [25], [26], [29], [35] and subband coding [20], [35]. Although wavelets have been studied by mathematicians for many years [4], wavelet transforms have recently generated a great deal of interest as a new form of multiresolution representation for one-dimensional (1-D) signals and two-dimensional (2-D) images [27]. Multiresolution representations are commonly used for the analysis of data and information in signal processing. Thus, the last few years have witnessed an explosion of applications of wavelets in image processing [1], [24], [25], [41], speech processing [17], [22], [36], wideband correlation processing [40], and numerical analysis [2], [3]. Image compression has been cited as potentially the biggest application area for wavelets [1], [27], [35].

The wavelet transform is an operation that transforms a function by integrating it with modified versions of some kernel

function [6]. The kernel function is called the mother wavelet, and the modifications are translations and compressions (or dilations) of the mother wavelet. In this paper, we consider issues dealing with parallel implementations of the 2-D discrete wavelet transform (DWT) for analyzing the information content of 2-D images. We provide a comprehensive treatment of the problem considering algorithmic and implementation issues on fine-grained as well as coarse-grained parallel platforms.

As with many algorithms in signal and image processing, the 2-DDWT is computationally intensive and operates on large data sets. These factors, coupled with the demand for real-time operation in many image processing tasks, have necessitated the use of parallel processing to provide high performance at a reasonable cost. A number of researchers have proposed parallel solutions of the DWT [5], [9], [12], [19], [21], [23], [30], [38]. Most of the proposed solutions focus on parallelizing a given sequential technique and do not study the impact of various machine and problem parameters on the performance. This paper primarily focuses on the scalability of 2-DDWT on coarse- and fine-grained machines. In particular, we analyze the effect of a variety of factors on the performance of the parallel 2-D DWT algorithm. These factors include the data distribution across processors, machine size, problem size, wavelet size, use of separable versus nonseparable wavelets, and choice of parallel algorithm.

The wavelet decomposition can be realized by successive filtering and subsampling of the input signal. The filtering operation can be accomplished either by convolutions with the filter kernel in the time domain or by computing the discrete Fourier transform followed by point-by-point multiplication in the frequency domain. We compare the complexity of the parallel time-domain and frequency-domain techniques used in implementing the filterbanks of the 2-D subband decomposition schemes. We show how the time and frequency domain approaches can be combined to achieve optimal performance for a broad range of problem and machine sizes. The cases of both separable and nonseparable wavelet bases are considered. Experimental results on the MasPar MP-series—a fine-grained parallel system—and on the Intel Paragon—a coarse-grained parallel computer—are presented and compared to the theoretical analyses. These machines can be considered to be representative of the coarse- and fine-grained distributed memory architectures of today since the architectural characteristics (network parameters such as diameter, bisection width, routing algorithms, and computation versus communication latencies, etc.) of these machines are very similar to the contemporary machines. See [37] for a recent survey of different parallel commercial and research machines and how they relate to each other.

Manuscript received December 14, 1998; revised July 20, 2000. This work was supported in part by ARPA under Contract DABT63-92-C-0022 and by NSF Grant CCR-9875662. The content of the information does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred. The associate editor coordinating the review of this paper and approving it for publication was Prof. Chin-Liang Wang.

J. N. Patel is with Intel Architecture Product Division, Oracle Corporation, Redwood Shores, CA 94605 USA.

A. A. Khokhar is with the Department of Electrical Engineering and Computer Science, University of Illinois, Chicago, IL 60607 USA (e-mail: ashfaq@eecs.uic.edu).

L. H. Jamieson is with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA.

Publisher Item Identifier S 1053-587X(00)10151-5.

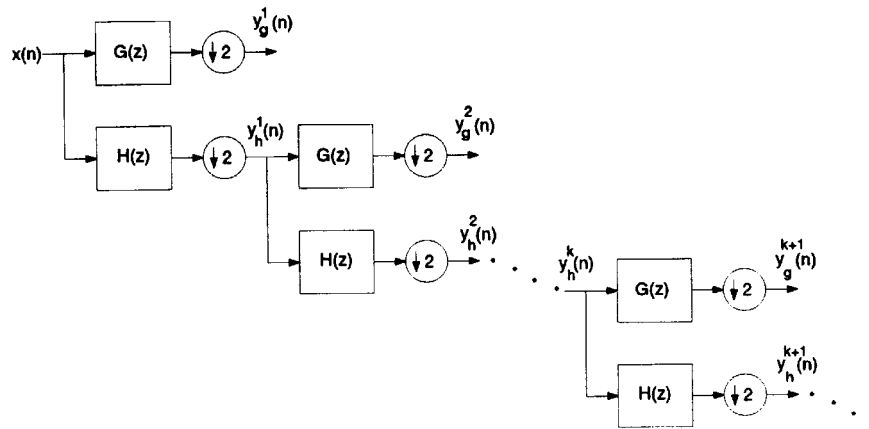


Fig. 1. Analysis filter bank of the 1-D subband decomposition scheme. By repeating in cascade for the low-frequency subsampled approximations, we compute the wavelet representation of the signal $x(n)$ at successively coarser resolutions.

Section II describes the computational structure and algorithms for the 2-D DWT. Section III describes the parallel computational model, scalability measures, and data distributions used for our study. Analytical results for various 2-D DWT algorithms on fine-grained and coarse-grained machines are presented in Section IV. Experimental results on the target parallel platforms are discussed in Section V. Section VI outlines the major conclusions of this work.

II. PARALLEL ALGORITHMS FOR THE 2-D DISCRETE WAVELET TRANSFORM

This section describes the wavelet decomposition of a signal using a quadrature mirror filter (QMF) filter bank. We begin by describing the 1-D subband decomposition scheme and extend these concepts to the case of 2-D wavelet decomposition of image data.

A. 1-D Wavelet Decomposition

Mallat [27] shows that the computation of the wavelet representation can be accomplished with a pyramidal algorithm based on convolutions with quadrature mirror filters. Consider the analysis filter bank of the 1-D subband decomposition scheme shown in Fig. 1. In this figure, $H(z)$ represents a lowpass filter and $G(z)$ represents a highpass filter. One can construct filters $H(z)$ and $G(z)$ which are both orthogonal and converge to continuous functions with compact support. Such filters are called *regular*, and examples can be found in [7], [8], [39].

The orthogonal wavelet representation of a discrete signal $x(n)$ can be computed by convolving with the filters $H(z)$ and $G(z)$ and retaining every other sample of the output. The process of decomposing the sequence into two sequences at half resolution can be iterated on either or both sequences. To achieve better resolution at lower frequencies, the scheme is commonly iterated on the lower band as shown. The output from the lower band of the k th stage $y_h^k(n)$ is the input for stage $k + 1$ of the wavelet decomposition. In general, K stages of wavelet decomposition result in a $(K + 1)$ -band

wavelet decomposition of the original signal $x(n)$. This can be represented as follows:

$$y_h^{k+1}(n) = \sum_{i=-\infty}^{+\infty} h(i)y_h^k(2n-i)$$

$$y_g^{k+1}(n) = \sum_{i=-\infty}^{+\infty} g(i)y_h^k(2n-i).$$

The filters $H(z)$ and $G(z)$ are called analysis filters. The signal can also be reconstructed from a wavelet representation with a similar pyramidal algorithm. The synthesis filters are time-reversed versions of the analysis filters. At the k th stage of wavelet reconstruction, $y_h^k(n)$ can be reconstructed as follows:

$$y_h^k(n) = \sum_{i=-\infty}^{+\infty} y_h^{k+1}(i)h(n-2i) + y_g^{k+1}(i)g(n-2i).$$

B. 2-D Wavelet Decomposition

In order to apply wavelet decompositions to images, 2-D extensions of wavelets are required. This can be achieved by the use of separable or nonseparable wavelets as described below.

Separable 2-D Filter Bank: A separable filter implies that filtering can be performed in one dimension (rows) followed by filtering in another dimension (columns). A 2-D wavelet transform can be computed with a separable extension of the 1-D decomposition algorithm [27], as shown in Fig. 2. We first convolve the rows of $x(n, m)$ with a 1-D filter, retain every other column, convolve the columns of the resulting signals with another 1-D filter, and retain every other row. The filters used in this decomposition are the 1-D QMF filters described in Section II-A. Further stages of the 2-D wavelet decomposition can be computed by recursively applying the procedure to the LL band (see Fig. 2) of the previous stage. In general, K stages of wavelet decomposition result in a $(3K + 1)$ -band wavelet decomposition of the original image $x(n, m)$.

Nonseparable 2-D Filter Bank: In certain cases, it is desirable to use nonseparable subsampling to obtain useful 2-D wavelet representations [27], [35]. For example, nonseparable

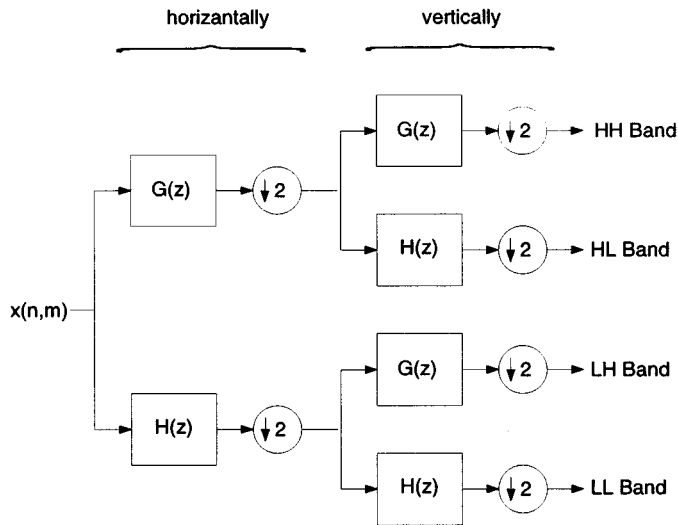


Fig. 2. Analysis filter bank of the separable 2-D subband decomposition scheme. This algorithm is based on iterative 1-D filtering and subsampling (across rows and columns) on the LL band of this filter.

wavelet orthonormal bases can be used for texture discrimination and fractal analysis [28]. In this case, we can use a nonseparable 2-D subband decomposition scheme as shown in Fig. 3. $H(z_1, z_2)$ and $G(z_1, z_2)$ represent the 2-D filters, and D is the matrix characterizing the sampling process. Iteration of the filterbank on the lowpass branch leads to a 2-D DWT.

III. COMPUTATIONAL MODEL AND SCALABILITY

We analyze the scalability of our algorithms and implementations using several architecture and algorithm parameters. This section describes the parameters of the computational model used in our analysis and defines a formal notion of scalability of an algorithm used in this study.

A. Computational Model and Assumptions

For our analysis, we use an explicit network model of the architecture. The terminology is primarily based on the conventions used in [10] and [13].

Let t_f be the time required for one floating-point operation and t_c be the sequential time required to compute one butterfly of an FFT. The time required for the complete transfer of a message containing m words between two processors that are l connections away is given by $t_s + (t_h + t_w m) \times l$, where t_s is the startup time, t_h is the time taken for a message fragment to hop from one processor to the neighboring one, and $t_w = \text{bytes-per-word}/B$, where B is the bandwidth of the communication channel between the processors in bytes per second. The model parameters are summarized in Table I.

B. Scalability of Parallel Algorithms

In our analyses, we define scalability as follows. Consider an algorithm that runs in $T(n, p)$ time on a p processor architecture and where the input problem size is n . The algorithm is considered scalable on the architecture if $T(n, p)$ increases linearly with an increase in the problem size or decreases linearly with an increasing number of processors (machine size) [15], [18].

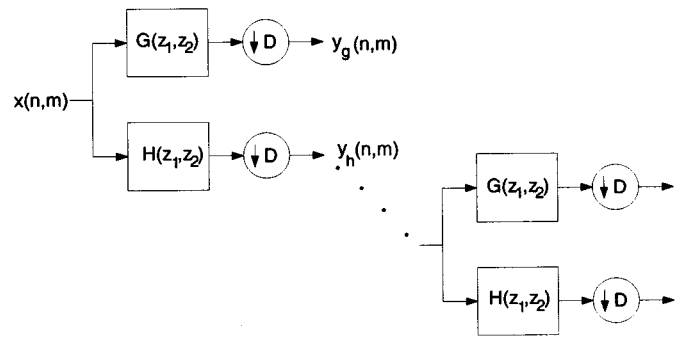


Fig. 3. Analysis filter bank of the nonseparable 2-D subband decomposition scheme. This algorithm is based on nonseparable subsampling and leads to nonseparable wavelets.

TABLE I
DESCRIPTION OF THE MODEL
PARAMETERS

t_s	Startup time for a message
t_h	Time taken for a message to hop from one processor to an immediate neighbor
t_w	Time to transfer one word,
t_f	Time required for one floating point operation
t_c	Time required to compute one butterfly of an FFT

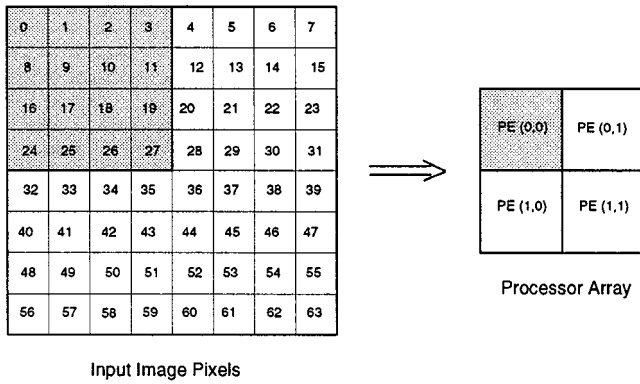
We study the performance of our algorithms by varying machine and problem size, by varying the size of the wavelet filter kernel, and by varying the processor speed. It is likely that no single algorithm is scalable over the entire range of machine and problem sizes. One of the important factors limiting the range of scalability is the sequential component of the parallel algorithm. In order to determine the regions of scalability for different algorithms presented in this paper, we use the analytical methods developed in [16]. We compare the analytical results with the experimental data and discuss any contrast that is observed.

C. Data Distribution Schemes

Image data is usually available as a 2-D array of elements. The data elements correspond to pixel values in the 2-D image plane. In this section, we describe two important data-distribution schemes—*block* distribution and *row* or *column* distribution—used in our implementations. For the sake of simplicity in analysis, in the following, we assume that n and p are powers of 2.

Block distribution: Assume that an input image x of size n such that $n = \sqrt{n} \times \sqrt{n}$ is distributed across p processors. Further, assume that the processors are configured as a square grid of $\sqrt{p} \times \sqrt{p}$ processors and that each processor can be accessed by its row and column indices i, j as $P_{i,j}, 0 \leq i, j, < \sqrt{p}$.

In a block distribution, each processor holds a nonoverlapping block of $\sqrt{n}/\sqrt{p} \times \sqrt{n}/\sqrt{p}$ pixels of the image, and adjacent processors in the $\sqrt{p} \times \sqrt{p}$ processor grid hold adjacent blocks of the input image. An example of a block distribution is shown in Fig. 4.



Input Image Pixels

Fig. 4. Mapping an 8×8 input image to a 2×2 processor array using block distribution.

Row or column distribution: An input image is distributed across p processors P_i , $0 \leq i < p$. In a row distribution, each processor holds \sqrt{n}/p consecutive rows of the input image. Likewise, in a column distribution, each processor holds \sqrt{n}/p consecutive columns of the input image. An example of row distribution is shown in Fig. 5.

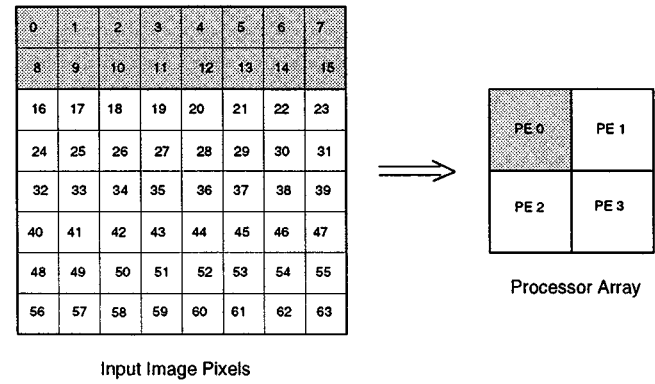
IV. ANALYTICAL RESULTS

In this section, we present analytical results of the 2-D DWT algorithms on fine- and coarse-grained machines. Two commonly used methods for filtering a digital signal are convolution-based filtering in the time domain and point-by-point multiplication in the frequency domain. In the following, we analyze the time complexity of each method on parallel computers. For the analysis, we use the model parameters defined in Section III.

Various computational steps performed in realizing the 2-D-DWT algorithms are outlined in Fig. 6. The analyses presented in this section are based on these steps, assuming the different data distributions discussed in Section III-C. For each step, we assume the best algorithm available on the target platform. For the row (or column) distribution, we assume that $p \leq \sqrt{n}$; for the block distribution, we assume that $p \leq n$. We further assume that the size of each wavelet kernel is L for the separable case and $L \times L$ for the nonseparable case. Let T_1 be the time to compute one level decomposition of a 2-D-DWT algorithm. In terms of the steps in Fig. 6, this corresponds to one iteration of the algorithm. We will show details of the analysis for computing T_1 . Recursion is then applied to achieve the bounds for a K -stage 2D-DWT decomposition. It can be easily shown that the total time for the K stages is bounded by $2T_1$ since the size of the input signal/image reduces to half after each iteration.

A. Analytical Results on Fine-Grained Machines

The MasPar computers used in the experiments are based on a 2-D mesh interconnection network. Hence, we assume a 2-D mesh interconnect between processors, although the analysis can easily be generalized to other topologies. The processors are distributed as a square grid of $\sqrt{p} \times \sqrt{p}$ processors, and each processor is connected to its four nearest neighbors in the north



Input Image Pixels

Fig. 5. Mapping an 8×8 input image to a 2×2 processor array using row distribution.

south, east, and west directions. Since fine-grained machines are characterized by large number of processors, the number of pixels assigned to each processor are assumed to be small in number. For this reason, we have considered only block distribution on fine-grained machines. On fine-grained machines, we assume that model parameters t_s (the startup time) t_h (time per hop), and t_w (time to transfer one word), which are defined in Table I, are constants.

1) *Time-Domain Filtering for Separable 2-D DWT:* The separable 2-D DWT involves 1-D filtering of the rows of the input image with 1-D wavelet filters $H(z)$ and $G(z)$, followed by 1-D filtering of the columns of the image. We assume 1-D wavelet filters of size L , i.e., $h(i)$ and $g(i)$, $0 \leq i < L$. We first consider the convolution of $g(i)$ across the rows of the image. For a 1-D filter of length L , the computation of each output pixel requires $2Lt_f$ floating-point operations (additions and multiplications), where t_f is the time required for one floating-point operation. For a block distribution, the computation of each output pixel also requires data from $L/\sqrt{n/p}$ other processors in the same same row of the $\sqrt{p} \times \sqrt{p}$ processor grid. Since the communication can be accomplished in $L/\sqrt{n/p}$ data movements of one word each between adjacent processors, the time required for communication is $L/\sqrt{n/p}$. The sum of the computation and communication time required for each pixel is $2Lt_f + (L/\sqrt{n/p})$.

Each processor holds n/p pixels. Since the output is down-sampled by two, we need to compute only alternate columns of the image. Thus, the operation described above is repeated for $n/2p$ pixels in each processor. Hence, the time required to convolve $g(i)$ across the rows of the input image is $n/2p[2Lt_f + (L/\sqrt{n/p})]$.

We need to convolve both $g(i)$ and $h(i)$ across the rows of the image, and the entire operation is repeated across the columns of the image. Thus, the total time required for the first stage of the 2-D wavelet decomposition is

$$T_1 = L(n/p)4t_f + 2L(\sqrt{n/p}). \quad (1)$$

Since the successive stages of wavelet decomposition are performed only on the LL-band of the output of the previous stage, the number of pixels per processor at the input of the k th stage of wavelet decomposition is $\sqrt{n/p}/(2^{(k-1)}) \times \sqrt{n/p}/(2^{(k-1)}) =$

Algorithm A: *SeperableTimeDomain(x)*

```

for(i = 0, i < number-of-subbands)
    • Convolve each row in image x with G(z),
      downsample, and store the output in yg.
    • Convolve each row in x with H(z) downsample,
      and store the output in yh.
    • Convolve each column in yg with G(z),
      downsample, and store the output in ygg.
    • Convolve each column in yg with H(z),
      downsample, and store the output in ygh.
    • Convolve each column in yh with G(z)
      downsample, and store the output in yhg.
    • Convolve each column in yh with H(z),
      downsample, and store the output in yhh.
    • Let x = yhh.

```

Next i

Algorithm D: *NonSeperableFrequencyDomain(x)*

```

for(i = 0, i < number-of-subbands)
    • Perform DFT on the image x.
    • Multiply the DFT coefficients of image x with the
      DFT coefficients of filter G(z1,z2), perform IDFT
      on the product, downsample and store the output
      in yg.
    • Multiply the DFT coefficients of image x with the
      DFT coefficients of filter H(z1,z2), perform IDFT
      on the product, downsample and store the output
      in yh.
    • Perform DFT on the image yh.
    • Multiply the DFT coefficients of image yh with the
      DFT coefficients of the filter G(z1,z2) perform an
      IDFT of the product, downsample and store the
      output in yhg.
    • Multiply the DFT coefficients of image yh with the
      DFT coefficients of the filter H(z1,z2) perform IDFT
      on the product, downsample and store the output
      in yhh.
    • Let x = yhh.

```

Next i

Algorithm B: *SeperableFrequencyDomain(x)*

```

for(i = 0, i < number-of-subbands)
    • Perform DFT on each row of the image x.
    • Multiply the DFT coefficients of each row in x with the
      coefficients of G(z), perform IDFT on the output,
      downsample and store the output in yg.
    • Multiply DFT coefficients of each row in x with the
      coefficients of H(z), perform IDFT on the output,
      downsample and store the output in yh.
    • Perform DFT on each column of the image yg.
    • Multiply the DFT coefficients of each column in yg
      with the coefficients of G(z), perform IDFT on the
      output, downsample and store the output in ygg.
    • Multiply the DFT coefficients of each column in yg
      with the coefficients of H(z), perform IDFT on the
      output, downsample and store the output in ygh.
    • Perform DFT on each column of the image yh.
    • Multiply the DFT coefficients of each column in yh
      with the coefficients of G(z), perform IDFT on the
      output, downsample and store the output in yhg.
    • Multiply the DFT coefficients of each column in yh
      with the coefficients of H(z), perform IDFT on the
      output, downsample and store the output in yhh.
    • Let x = yhh.

```

Next i

Algorithm C: *NonSeperableTimeDomain(x)*

```

for(i = 0, i < number-of-subbands)
    • Convolve the image x with G(z1,z2) and store
      every alternate pixel of the output in yg.
    • Convolve the image x with H(z1,z2) and store
      every alternate pixel of the output in yh.
    • Convolve the image yh with G(z1,z2) and store
      every alternate pixel of the output in yhg.
    • Convolve the image yh with H(z1,z2) and store
      every alternate pixel of the output in yhh.
    • Let x = yhh.

```

Next i

Fig. 6. Computational steps for the separable and nonseparable 2-D-DWT algorithms in time and frequency domains.

$(n/p)/(2^{2(k-1)})$. If T_k is the time required for the k th stage of wavelet decomposition, then

$$\begin{aligned}
 T_k &= 4 \times \frac{n/p}{2 \times 2^{2(k-1)}} \left[2Lt_f + 2 \frac{L}{\left(\frac{\sqrt{n/p}}{2^{(k-1)}}\right)} \right] \\
 &< \frac{1}{2^{(k-1)}} \left[L(n/p)(4t_f) + 2L(\sqrt{n/p}) \right] \\
 &< \frac{T_1}{2^{(k-1)}}.
 \end{aligned}$$

If $T(K)$ is the total time required for K stages of the wavelet decomposition, it can be easily shown that $T(K) \leq 2T$.

If T_{conv} denotes the total time required to compute a K -stage separable 2-D-DWT using time-domain convolutions, we then have

$$T_{\text{conv}} \leq L(n/p)8t_f + 4L(\sqrt{n/p}) \quad (2)$$

$$= O(n/p). \quad (3)$$

Based on the scalability definition described earlier, T_{conv} scales linearly with p .

2) *Frequency-Domain Filtering for Separable 2-D-DWT*: To filter a signal in the frequency domain, we first compute the discrete Fourier transform (DFT) of the signal, perform a point-by-point multiplication with the DFT of the filter sequence, and then compute the inverse discrete Fourier transform (IDFT) of the product. A fast parallel computation of the DFT and IDFT can be achieved by parallel implementation of the fast Fourier transform (FFT) algorithms described in [16].

Let T_1 be the time taken to compute the first stage of the 2-D-DWT. We first consider the time taken to filter the rows of the image with $G(z)$ and $H(z)$. Assuming block distribution, where a row of the image consisting of \sqrt{n} pixels is divided among \sqrt{p} processors, the computation of the 1-D FFT of the requires $\log(\sqrt{p}) - 1$ communication steps and the local computation of a $\sqrt{n/p}$ -point 1-D FFT, as described in [11] and [31]. If t_c is the time taken to compute one butterfly of the FFT and 2^i is the distance between communicating processors at step i , the time taken for the $\log(\sqrt{p}) - 1$ communication steps is

$$\begin{aligned}
 &\sum_{i=0}^{\log \sqrt{p} - 1} [\sqrt{n/p}t_c + (t_s + t_w \sqrt{n/p}2^i)] \\
 &\approx \sqrt{n/p} \log(\sqrt{p})t_c + \log(\sqrt{p})t_s + \sqrt{n}t_w.
 \end{aligned}$$

The time to compute a $\sqrt{n/p}$ -point 1-D FFT is $\sqrt{n/p} \log(\sqrt{n/p}) t_c$. Hence, the time to compute a 1-D FFT on one row of the input image is approximately $\sqrt{n/p} \log(\sqrt{n}) t_c + \log(\sqrt{p}) + \sqrt{n}$, assuming $t_s, t_w = 1$.

Point-by-point multiplication with $G(z)$ and $H(z)$ requires $2\sqrt{n/p} t_f$ time. This is followed by an inverse 1-D FFT operation on both the products, where the cost of an inverse FFT equals the cost of a forward FFT. Hence, the total time required to filter one row of the image with $G(z)$ and $H(z)$ is approximately

$$3[\sqrt{n/p} \log(\sqrt{n}) t_c + \log(\sqrt{p}) + \sqrt{n}] + 2\sqrt{n/p} t_f.$$

Since each row of processors holds $\sqrt{n/p}$ rows of the image, the time required to filter all the rows of the image is $\sqrt{n/p}$ times the time required for one row. Thus, the total time required to compute the first stage of the wavelet decomposition, which is the sum of the time required for filtering and downsampling along the rows followed by filtering and downsampling along the columns, is given by

$$T_1 \approx (n/p) [\log(\sqrt{n}) 6t_c + 4t_f] + \sqrt{n/p} [6\log(\sqrt{p}) + 6\sqrt{n}]. \quad (4)$$

As in the case of the convolution-based filtering, the total time for K stages of wavelet decomposition on p processors is bounded by $2T_1$. If T_{FFT} denotes the total time required to compute a K -stage separable 2-D-DWT using frequency-domain filtering, then

$$T_{\text{FFT}} \leq (n/p) [\log(\sqrt{n}) 12t_c + 8t_f] + \sqrt{n/p} [12\log(\sqrt{p}) + 12\sqrt{n}] \quad (5)$$

$$= O\left(\left(\frac{n}{p}\right) \log n\right). \quad (6)$$

Since computing an n -point FFT on a serial computer takes $O(n \log n)$ time, we claim that T_{FFT} scales linearly with p .

The nonseparable 2-D-DWT involves filtering the input image with 2-D wavelet filters $H(z_1, z_2)$ and $G(z_1, z_2)$. Assume 2-D wavelet filters of size $L \times L$, i.e., $h(i, j)$ and $g(i, j)$, $0 \leq i, j < L$. In the following, we analyze the computation complexity of the time-domain and frequency-domain filtering methods to compute the 2-D DWT using nonseparable wavelet bases.

3) *Time-Domain Filtering for Nonseparable 2-D DWT*: Since each processor holds $\sqrt{n/p} \times \sqrt{n/p} = n/p$ adjacent pixels of the input image, every processor requires data from $(L/\sqrt{n/p}) \times (L/\sqrt{n/p})$ other processors. If $(L/\sqrt{n/p}) > 1$, each processor requires data from processors that are more than one hop away. However, efficient pipelined communication patterns can be employed that allow the entire data transfer to be accomplished by $(L/\sqrt{n/p}) \times (L/\sqrt{n/p})$ data movements of size n/p between adjacent processors. Therefore, assuming $t_s, t_h, t_w = 1$, we have

$$T_1 = 2t_f L^2(n/p) + L^2. \quad (7)$$

The successive stages of wavelet decomposition are performed only on the lowpass band of the previous stage.

Assuming that the sampling matrix D has a sampling density of $1/2$, the number of pixels at the input of the k th stage of wavelet decomposition is $(n/p)/(2^{(k-1)})$. If T_k is the time required for the k th stage of wavelet decomposition, then we can show that

$$T_k = \frac{1}{2^{(k-1)}} [2t_f L^2(n/p)] + L^2.$$

If $T(K)$ is the total time required for K stages of wavelet decomposition, then we have

$$T(K) = T_1 + T_2 + \dots + T_k \leq 4t_f L^2(n/p) + KL^2.$$

In practice, only a few stages of decomposition are required. Hence, K can be regarded as a small constant. If T denotes the total time required to compute a K -stage nonseparable 2-D DWT using time-domain convolutions, then we have

$$T \leq 4t_f L^2(n/p) + KL^2 \quad (8)$$

$$= O\left(\frac{n}{p}\right). \quad (9)$$

4) *Frequency-Domain Filtering for Nonseparable 2-D DWT*: To filter a signal in the frequency domain using a nonseparable 2-D filter, we first compute the 2-D FFT of the signal, perform a point-by-point multiplication with the 2-D FFT of the filter sequence, and then compute the 2-D IFFT of the product. The 2-D FFT of the image requires $t_c(n/p) \log \sqrt{n} + 2t_h(n/p) \sqrt{p}$ time. Point-by-point multiplication with $G(z_1, z_2)$ and $H(z_1, z_2)$ requires $2t_f(n/p)$ time. This is followed by a 2-D IFFT operation on both the products, retaining half the points, which requires $t_c(n/p) \log \sqrt{n} + 2t_h(n/p) \sqrt{p}$ time. If T_1 is the time taken to compute the first stage of the 2-D DWT and assuming $t_h = 1$, we have

$$T_1 = 2t_c(n/p) \log \sqrt{n} + 2t_f(n/p) + 4(n/p) \sqrt{p}. \quad (10)$$

As in the case of the separable frequency-domain filtering, we can show that the time taken for the k th iteration T_k is

$$T_k < \frac{T_1}{2^{(k-1)}}.$$

Hence, the total time for K stages of wavelet decomposition on p processors is bounded by $2T_1$. If T denotes the total time required to compute a K -stage nonseparable 2-D DWT using frequency-domain filtering, then

$$T \leq 4t_c(n/p) \log \sqrt{n} + 4t_f(n/p) + 8(n/p) \sqrt{p} \quad (11)$$

$$= O\left(\left(\frac{n}{p}\right) \log n + \frac{n}{\sqrt{p}}\right). \quad (12)$$

B. Analytical Results on Coarse-Grained Machines

For the coarse-grained machines, we present only a summary of the major analytical results. Additional details can be found in [33]. For the row (or column) distribution, we assume that $p \leq \sqrt{n}$; for the block distribution, we assume that $p \leq n$.

TABLE II
 SUMMARY OF THE ANALYTICAL RESULTS

Platform	Algorithm	Row/Column Distribution	Block Distribution
Fine Grain	A		$O(L\frac{n}{p} + L\sqrt{n/p})$
	B		$O(\frac{n}{p} \log \sqrt{n} + \frac{n}{\sqrt{p}})$
	C		$O(L^2\frac{n}{p} + L^2)$
	D		$O(\frac{n}{p} \log \sqrt{n} + \frac{n}{\sqrt{p}})$
Coarse Grain	A	$O(L\frac{n}{p} + t_s p + t_w(n/p))$	$O(L\frac{n}{p} + (t_s + t_w)L\sqrt{n/p})$
	B	$O(\frac{n}{p} \log \sqrt{n} + t_s p + t_w(n/p))$	$O(\frac{n}{p} \log \sqrt{n} + t_s \log \sqrt{p} + t_w(n/p) \log p)$
	C	$O(L^2\frac{n}{p} + (\frac{t_s}{n/p} + t_w)L\sqrt{n})$	$O(L^2\frac{n}{p} + (\frac{t_s}{n/p} + t_w)\sqrt{n/p}L^2)$
	D	$O(\frac{n}{p} \log \sqrt{n} + t_s p + t_w(n/p))$	$O(\frac{n}{p} \log \sqrt{n} + (t_s + t_w n/p) \log \sqrt{p})$

1) *Time-Domain Filtering for Separable 2-D DWT*: This consists of convolution and subsampling of the rows of the image with $g(i)$ and $h(i)$ followed by convolution and subsampling of the columns of the image with $g(i)$ and $h(i)$.

Consider the case of block distribution first. To convolve a 1-D kernel of size L across the rows of the image, the computation of each output pixel requires data from $L/\sqrt{n/p}$ processors in the same row of the processor grid. This can be accomplished by $L/\sqrt{n/p}$ messages of size n/p between adjacent processors. Assuming the parameters listed in Table I, the time required for communication is $(L/\sqrt{n/p})(t_s + t_w(n/p)) = t_s(L/\sqrt{n/p}) + t_w L\sqrt{n/p}$.

The computation of each output pixel requires $2L$ floating-point operations (additions and multiplications). Since each processor holds n/p pixels, the total computation time for row convolution is $2t_f L(n/p)$, where t_f is the time required for one floating-point operation. Thus, the total time required for row convolution on coarse-grained machines is $2t_f L(n/p) + t_s(L/\sqrt{n/p}) + t_w L\sqrt{n/p}$.

The time for 1-D convolution across the columns of the image is the same as the time for row convolution. Thus, the total time taken for separable 2-D DWT using *block distribution* is given by

$$T_{\text{block}} = 8t_f L(n/p) + 4t_s \frac{L}{\sqrt{n/p}} + 4t_w L\sqrt{n/p}. \quad (13)$$

For *row distribution*

$$T_{\text{row}} = 8t_f L(n/p) + t_s Kp + 2t_w(n/p). \quad (14)$$

In a similar manner (for details, see [33]), we obtain expressions for the various data distribution.

2) *Frequency-Domain Filtering for Separable 2-D DWT*: To filter a signal in the frequency domain, we first compute the DFT of the signal, perform a point-by-point multiplication with the DFT of the filter sequence, and then compute the IDFT of the product. A fast parallel computation of the DFT and IDFT can be achieved by parallel implementation of the FFT.

The total time for *block distribution* and *row distribution* is given by the following equations:

$$T_{\text{row}} = 4t_c(n/p) \log \sqrt{n} + 8t_f(n/p) + t_s K \log \sqrt{p} + 8t_w(n/p) \log \sqrt{p}. \quad (15)$$

$$T_{\text{row}} = 4t_c(n/p) \log \sqrt{n} + 8t_f(n/p) + t_s Kp + 2t_w(n/p). \quad (16)$$

3) *Time-Domain Filtering for Nonseparable 2-D DWT*: In the case of the nonseparable 2-D DWT, the time to compute the first stage consists of convolution of the image with $g(i, j)$ and $h(i, j)$, followed by subsampling of the image.

The total time for *block distribution* and *row distribution* is given by the following equations.

$$T_{\text{block}} = 4t_f L^2(n/p) + 2t_s \frac{L^2}{n/p} + 2t_w L^2 \sqrt{n/p}. \quad (17)$$

$$T_{\text{row}} = 4t_f L^2(n/p) + 2t_s \frac{L}{\sqrt{n/p}} + 2t_w L\sqrt{n}. \quad (18)$$

4) *Frequency-Domain Filtering for Nonseparable 2-D DWT*: To filter a signal in the frequency domain using a nonseparable 2-D filter, we first compute the 2-D FFT of the signal, perform a point-by-point multiplication with the 2-D FFT of the filter sequence, and then compute the 2-D IFFT of the product.

The total time for *block distribution* and *row distribution* is given by the following equations.

$$T_{\text{block}} = 4t_c(n/p) \log \sqrt{n} + 4t_f(n/p) + 4t_s K \log \sqrt{p} + 8t_w(n/p) \log \sqrt{p}. \quad (19)$$

$$T_{\text{row}} = 4t_c(n/p) \log \sqrt{n} + 4t_f(n/p) + t_s Kp + 2t_w(n/p). \quad (20)$$

Interpretation of Analytical Results: A summary of the analytical results for both fine- and coarse-grained platforms assuming different data distributions is given in Table II. Each expression consists of at least two parts: one corresponding to the computation complexity and the other corresponding to the communication complexity (the terms involving t_s and t_w in the case of coarse-grained platforms). In this table, these parts have not been merged intentionally to argue the effects of computation/communication granularity on the performance bounds.

The execution time for the convolution-based algorithms (A and C) increases with the size of the wavelet filter (L), whereas the FFT-based algorithms (B and D) are independent of L . The table also shows that the execution time of the convolution-based filtering increases linearly with n , whereas execution time

of the FFT-based filtering increases superlinearly with n . Thus, we would expect the convolution-based algorithms to be faster than FFT-based algorithms for small filter sizes and slower than FFT-based algorithms for large filter sizes. For a given filter length, the convolution-based time domain method is more scalable with respect to n than the frequency domain filtering operation for the separable and nonseparable cases.

Comparing convolution-based algorithms for different distributions, separable wavelet filters are more efficient for the block distribution for small values of L . For nonseparable filters, the block distribution is more efficient if n is large for a practical range of L .

For the frequency domain algorithms, the row-distributed algorithms are faster than the block distributed algorithms when the ratio of the problem size to machine size (n/p) is large.

In the case of coarse-grained platforms, communication time is the combination of the message start-up cost (t_s) incurred in each communication step plus the data transfer time (t_w) dictated by the bandwidth of the links between processors. Usually, the message startup cost is in the order of tens of microseconds in contemporary coarse-grained architectures [14]. Due to this heterogeneity in the machine parameters, the communication-intensive algorithms do not scale linearly in true sense with the increase in the problem size or data size. In our case, this is true for the FFT-based algorithms on coarse-grained machines (B and D). With row/column distribution, the time domain algorithm based on separable kernel performs better in terms of communication when image size is relatively large compared to the machine size.

The computation of inverse wavelet for orthogonal and biorthogonal (such as 9/7) filters is identical to the forward wavelet. In the case of its parallel implementation, the total computation cost depends on the how the coefficient matrix is distributed among processors.

Assume that for a k -level wavelet decomposed image, coefficients in each wavelet subbands are equally partitioned over the processors, i.e., for each subband of size $(\sqrt{n}/2^{i-1}) \times (\sqrt{n}/2^{i-1})$ for $i = 1 \dots k$, each processor contains $(\sqrt{n/p}/2^{i-1}) \times (\sqrt{n/p}/2^{i-1})$ coefficients. This distribution scenario is exactly the same as if you have just finished performing the k -level forward wavelet decomposition using the algorithms and data distributions used in this paper. Now, performing the inverse wavelet on this distribution is equivalent to traversing the wavelet computation graph backward performing convolutions using inverse filters. The manner in which pixel data was collected to perform the forward filter is the same way to perform the inverse filter. For example, for the coarse-grained case, the time required for communication in an inverse wavelet transform is $(L/\sqrt{k/p})(t_s + t_w(k/p)) = t_s(L/\sqrt{k/p}) + t_w L\sqrt{k/p}$ for a subband of $\sqrt{k} \times \sqrt{k}$ coefficients, where L is the size of the inverse filter. This communication time is exactly the same for forward transform on an image of $\sqrt{k} \times \sqrt{k}$ pixels. In addition, note that just like the forward transform, the computation and communication overhead in the final stage of reconstruction will dominate the total computation time T , and it can be easily shown that $T < 2T_1$, where T_1 is the time for the final reconstruction phase.

V. SCALABLE IMPLEMENTATIONS OF 2-D DWT ALGORITHMS

In this section, we discuss implementations of the algorithms outlined in Section IV on fine-grained and coarse-grained machines. To ensure consistent results using both time-domain and frequency-domain methods, we assume that the input image is periodic with period $\sqrt{n} \times \sqrt{n}$. For time-domain filtering based on 1-D or 2-D convolutions, this implies that some of the processors holding the borders of the image will need image border pixels from processors at the opposite end of the $\sqrt{p} \times \sqrt{p}$ processor grid. We also assume that the size of the wavelet filter and the filter coefficients are known in advance. This implies that processors can use locally stored values of the twiddle factors to compute the FFT and access locally stored DFT coefficients of the wavelet filter. The plots in this section show the time taken for the first stage of the 2-D wavelet decomposition, i.e., T_1 . Note that the problem size reduces at subsequent levels of wavelet decomposition; therefore, reduction in the number of processors assigned to compute these decompositions will help maintain processor utilization.

A. Experimental Results on the MasPar MP-1 and MP-2

In this section, we discuss the implementation results on fine-grained SIMD machines. Fine-grained machines, in general, are characterized by a large number of processors with a fairly simple arithmetic logic unit (ALU) in each processor. As discussed in Section IV-A, we use a block distribution of the $\sqrt{n} \times \sqrt{n}$ input image on the p processors. Each processor holds a nonoverlapping block of $\sqrt{n/p} \times \sqrt{n/p}$ pixels of the image. Since the available memory per processor is limited, the number of image pixels per processor n/p is small ($\sqrt{n/p} \leq 32$).

For our experiments, we have used the MasPar MP-1 and MP-2 as being representative of fine-grained machines. Our algorithm implementations use the Xnet primitive for interprocessor communication. Xnet allows data to be sent in parallel in a specified direction using high-speed local routing on the mesh. We present implementation results on both the MP-1 and MP-2 to show the effects of processing power of individual processors on the performance of the algorithms. We have used an extended version of sequential ANSI C, which is the MasPar Programming Language (MPL), to keep our implementations free of machine-dependent software features.

Fig. 8 shows the performance of the time-domain and frequency-domain methods for increasing size of the 1-D wavelet filters $h(i)$ and $g(i)$, $0 \leq i < L$ on 16K processors of the MasPar MP-1. Fig. 8(a) plots the time taken for an input image of size 256×256 , and Fig. 8(b) plots the time taken for an image of size 512×512 pixels. The experimental plots are consistent with the theoretical results stated in (3) and (6). The time for the convolution-based algorithm in the time-domain increases linearly with the filter size L , whereas the time for the FFT-based frequency-domain algorithm is independent of L . Similar results are obtained for larger image sizes ($\sqrt{n} \times \sqrt{n} = 1K \times 1K, 2K \times 2K, 4K \times 4K$).

Fig. 8(a) and (b) also indicates that the crossover point—i.e., the filter-size at which the FFT-based algorithm outperforms the convolution-based algorithm—increases with image size for a fixed number of processors. We can determine the regions of the

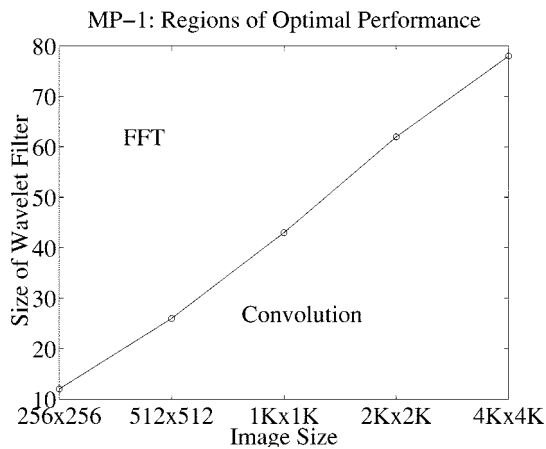
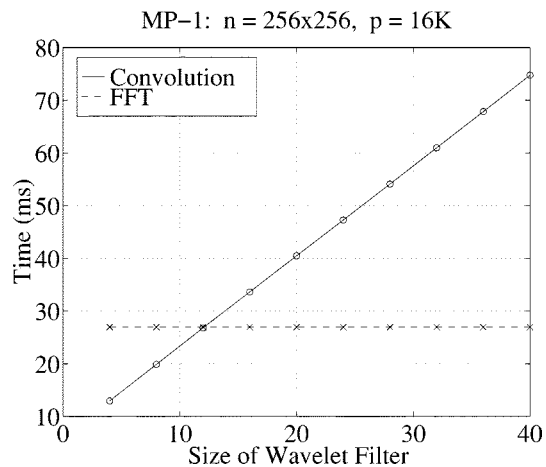
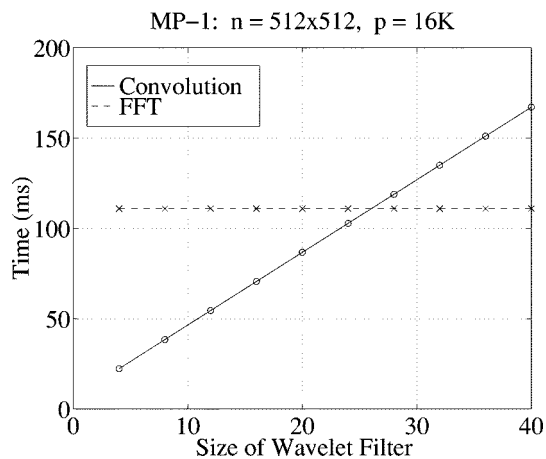


Fig. 7. Experimentally determined regions in the image-size and filter-size space where each method is fastest. The plot is based on execution times for 16 384 processors of the MP-1.



(a)



(b)

Fig. 8. Performance of the algorithms for increasing size of the wavelet filter (L). The plots show the time taken by the time-domain and frequency-domain methods on 16 384 processors of the MP-1 for (a) an image of size 256×256 pixels and (b) an image of size 512×512 pixels.

filter-size/image-size space, where each algorithm is fastest, as shown in Fig. 7. Figs. 7 and 8 clearly indicate that scalability is best achieved by changing the algorithm approach with a change in the problem parameters.

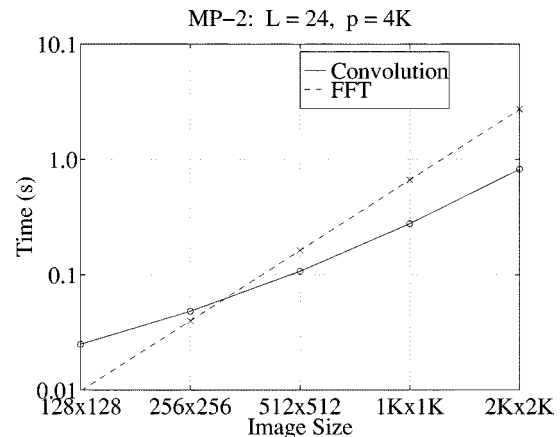
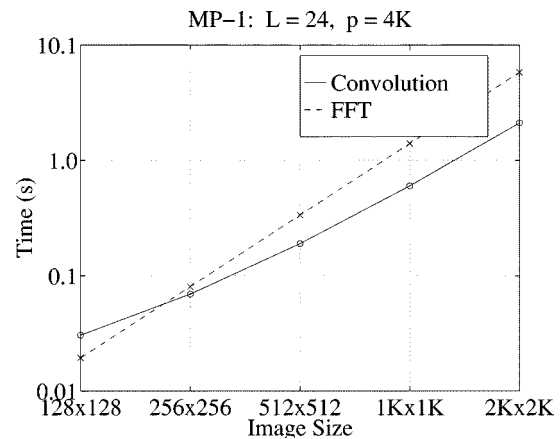


Fig. 9. Comparison of performance on the MP-1 and MP-2: Scalability with respect to image size ($\sqrt{n} \times \sqrt{n}$), for a wavelet filter of size 24 on 4096 processors of (a) the MP-1 and (b) the MP-2.

We have performed scalability experiments on both the MasPar MP-1 and MP-2. Fig. 9 shows the scalability of the algorithms with respect to the problem size n . For a given wavelet filter and a fixed number of processors, the time taken by the FFT-based frequency domain method grows faster with increasing n than the time taken by the time domain method. This is consistent with the analytical results stated in (3) and (6).

Fig. 10 shows the scalability behavior of the algorithms with respect to increasing machine size. We notice that the execution time decreases for both algorithms with respect to p . However, beyond a certain machine size, the execution time for the FFT-based algorithm decreases faster than for the convolution-based algorithm. This can be explained by analyzing (3) and (6). As p increases, n/p decreases. Beyond a certain value of p , the filter size L begins to dominate the execution time of the convolution-based algorithm. Thus the frequency domain method outperforms the time domain operation for large L .

Both the MP-1 and MP-2 use the same Xnet communication network. The only difference between the architectures is the faster processors used by the MP-2. Thus, to study the impact of processor speed, we compare the performance of the algorithms using the same number of processors. It is evident from Figs. 9 and 10 that the execution characteristics of the algorithms are similar on the two machines. However, the crossover

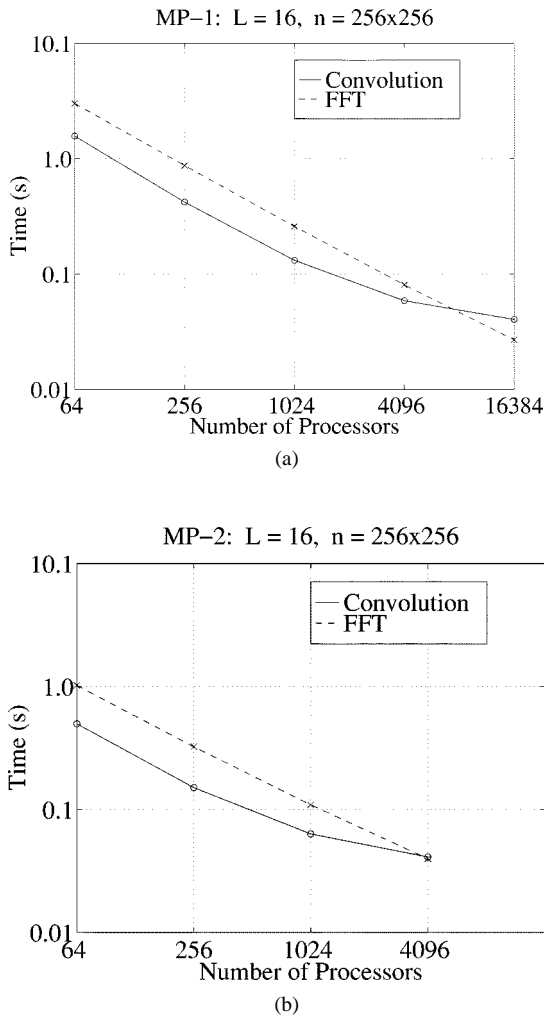


Fig. 10. Comparison of performance on the MP-1 and MP-2: Scalability with respect to processor size (p) for a wavelet filter of size 16 and an input image of size 256×256 pixels on (a) the MP-1 and (b) the MP-2.

point (which is the ratio n/p at which the FFT-based algorithm outperforms the convolution-based algorithm) is higher on the MP-2 relative to the MP-1. This is because the ratio of computation to communication time is larger for the convolution algorithm; hence, it performs better on the MP-2 due to the higher computation-speed/communication-speed ratio of the MP-2. The results obtained on Maspar MP's may be generalized to fine-grained architectures since the communication patterns embedded in the computation are local and require only nearest-neighbor communications.

As described in Section II, the filterbank implementation of the wavelet decomposition involves successive filtering and subsampling of the lowpass band of the previous stage of decomposition. The effective problem size n_k decreases at each stage of the decomposition. For a given machine size and wavelet filter, we can determine the ratio $r = n'/p$ for which the frequency domain method outperforms the time domain method on that machine. If the input image size $n > rp$, we can perform the initial stages of the wavelet decomposition using the frequency domain method. When $n_k \leq rp$, we switch to the time domain method for the remaining stages of the wavelet decomposition. Thus, a polyalgorithmic approach will

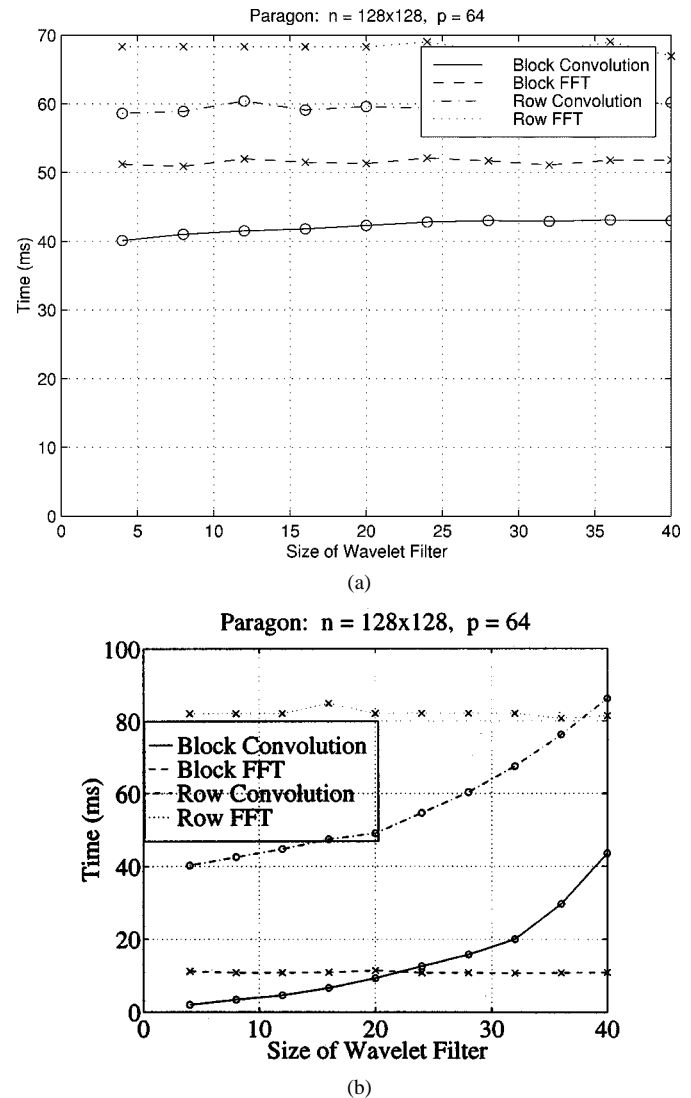


Fig. 11. Performance of the 2-D DWT algorithms for increasing size of the wavelet filter (L). The plots show the time taken to perform a 2-D DWT on 64 nodes of the Intel Paragon on an image of size 128×128 pixels using (a) separable wavelet bases and (b) nonseparable wavelet bases.

provide optimal performance over a broad range of problem and machine sizes.

B. Implementation Results on Coarse-Grained Machines

In this section, we present experimental results for the different 2-D DWT algorithms on coarse-grained machines. The Intel Paragon used in our experiments is an asynchronous (MIMD) coarse-grained mesh architecture with 512 i860 XP computational nodes. The nodes are configured as a 16×32 mesh, each with 32 Mbytes of memory. Each node has a peak speed of 75 MFLOPS (double precision). Communication between nodes is supported by asynchronous message passing across the 2-D mesh.

Fig. 11 compares the performance of the 2-D DWT algorithms using row distribution and block distribution for increasing size of the window kernel. Fig. 11(a) plots the execution time for an image of size 128×128 using separable wavelet bases. Although the execution time for the convolution-based algorithms increases slowly with respect to the filter

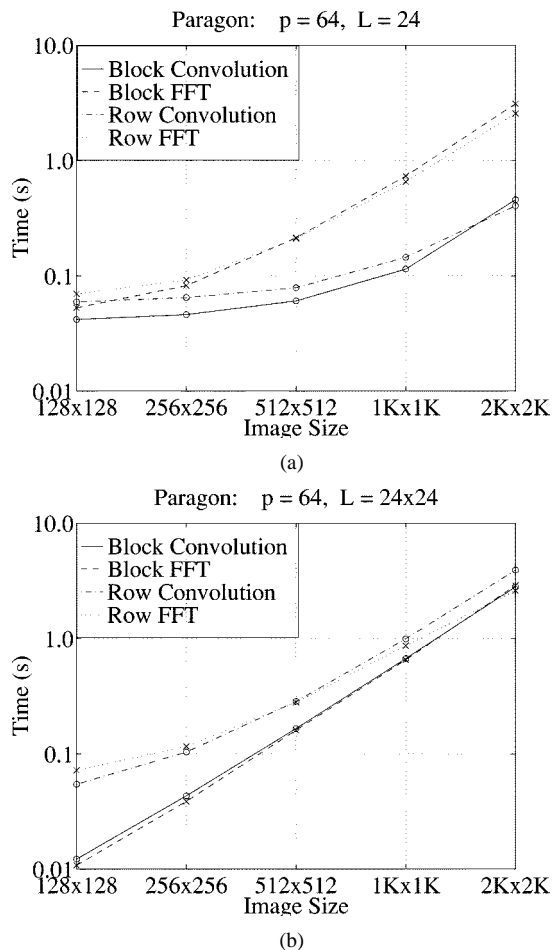


Fig. 12. Scalability with respect to image size $\sqrt{n} \times \sqrt{n}$. The plots show the time taken to perform a 2-D DWT on 64 nodes of the Intel Paragon using (a) separable wavelet bases of size $L = 24$ and (b) nonseparable wavelet bases of size $L \times L = 24 \times 24$.

size, the convolution-based algorithm using block distribution is the most efficient implementation for the 2-D DWT for a practical range of wavelet filter sizes. This is due to the fact that in the case of separable-bases, rows and columns of the input image are operated on independent of each other. Therefore, a row distribution is going to give better performance only in the row-phase of the algorithm. For the column phase, a processor must acquire data from the neighboring processors for all the pixels lying on the boundary row of its assigned subimage. The FFT-based solutions for the separable-bases case suffer from the poor performance of 1-D FFT algorithms [31] for this range of image sizes.

Fig. 11(b) plots the execution time for an image of size 128×128 using nonseparable wavelet bases. The convolution-based algorithm using block distribution is faster than the algorithms using row distribution as in the case of the separable wavelet filters. However, in contrast with the separable case, the execution time of the convolution-based algorithms increases faster than the execution time for FFT-based algorithms with increasing size of the wavelet filter. Thus, the FFT-based algorithm using block distribution is more efficient beyond a certain wavelet filter size. This is consistent with the analytical results summarized in Section IV-B. Similar results are observed for different image sizes and number of processors.

Fig. 12 shows the scalability of the algorithms with respect to the image size $\sqrt{n} \times \sqrt{n}$. Fig. 12(a) plots the execution time for varying image size on 64 processors of the Paragon using separable wavelet bases of fixed size ($L = 24$). As predicted by the theoretical results, the convolution-based algorithms scale better than the FFT-based algorithms with increasing problem size. Fig. 12(b) plots the execution time for varying image size on 64 processors of the Paragon using nonseparable wavelet bases of fixed size ($L \times L = 24 \times 24$). The convolution-based algorithms scale better than the FFT-based algorithms with increasing problem size, although for large image sizes, the difference in execution time is lower than for the separable case. Similar results are observed for different kernel sizes and number of processors.

It is important to note that the results reported in this paper for Intel Paragon are specific to the parameters of the architecture. For example, the relatively large startup time for message passing on the Intel Paragon (40–45 ms) adversely affects the performance of the matrix transpose operation required by the separable 2-D DWT algorithms using row distribution. The results may be different for a machine with a very low message startup time.

VI. CONCLUSIONS

In this paper, we have studied the scalability of 2-D discrete wavelet transform algorithms on coarse-grained parallel architectures. We have compared the analytical complexity of the time domain and frequency domain techniques used to implement the filterbanks of the 2-D subband decomposition schemes on parallel machines. Experiments on the Intel Paragon and MasPar machines have validated the analytical results. We have shown that while one algorithm performs significantly better than the other for a certain combination of the machine size, image size, and wavelet size, the opposite may be true for a different set of problem and machine parameters. Comparing the experimental results on both the platforms, the fine-grained case exhibits much more dramatic cases of crossovers between FFT and convolution based approaches as the wavelet filter size grows, but data distribution plays no significant role in the fine-grained algorithms. Coarse-grained machines are characterized by a larger local memory per node and fewer processors than fine-grained machines. This allows increased flexibility with respect to the choice of the data distribution of the input image on the processors. We have shown that the performance of the algorithms may be quite sensitive to the choice of the data distribution. Furthermore, one data distribution scheme may not be optimal for the entire range of problem-size and machine-size parameters. By contrasting the results on the MasPar MP-1, MP-2, and Paragon, we observe the impact of architectural parameters (e.g., ratio of computation to communication speed) on the relative performance of the algorithms.

ACKNOWLEDGMENT

The authors would like to thank P. N. Srinivasan for many helpful discussions.

REFERENCES

- [1] M. Antonini, M. Barlaud, D. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans. Image Processing*, vol. 1, pp. 205–220, Apr. 1992.
- [2] G. Beylkin, "On the representation of operators in based of compactly supported wavelets," *SIAM J. Numer. Anal.*, vol. 29, no. 6, pp. 1716–1740, Dec. 1992.
- [3] G. Beylkin, R. Coifman, and V. Rokhlin, "Fast wavelet transforms and numerical algorithms I," *Commun. Pure Appl. Math.*, vol. 44, pp. 141–183, 1991.
- [4] A. P. Calderon and A. Zygmund, "On the existence of certain singular integrals," *Acta Math.*, vol. 88, pp. 85–139, 1952.
- [5] C. Chakrabarti and M. Viswanath, "Efficient realizations of the discrete and continous wavelet transform: From single chip implementations to mapping on SIMD array computers," *IEEE Trans. Signal Processing*, vol. 43, pp. 759–771, Mar. 1995.
- [6] J. M. Combes, A. Grossman, and P. Tchamitchian, Eds., *Wavelets: Time-Frequency Methods and Phase Space*, 2nd ed. New York: Springer-Verlag, 1989.
- [7] I. Daubechies, "Orthonormal bases of compactly supported wavelets," *Commun. Pure Appl. Math.*, vol. 41, no. 7, pp. 909–996, 1998.
- [8] —, "The wavelet transform, time-frequency localization and signal analysis," *IEEE Trans. Inform. Theory*, vol. 36, pp. 961–1005, Sept. 1990.
- [9] I. Gertner, R. Peskin, and S. Walther, "Parallel computation of the continuous wavelet transform," *Proc. SPIE—Adaptive Signal Process.*, vol. 1565, pp. 414–422, July 1991.
- [10] A. Y. Grama, A. Gupta, and V. Kumar, "Isoefficiency: Measuring the scalability of parallel algorithms and architectures," *IEEE Parallel Distrib. Syst.*, vol. 1, pp. 12–21, Aug. 1993.
- [11] A. Gupta and V. Kumar, "On the scalability of FFT on parallel computers," presented at the Frontiers Conf. Massively Parallel Comput., Oct. 1990.
- [12] R. W. Hall, S. Kucuk, and M. Hamdi, "Wavelet transform embeddings in mesh architectures," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recogn.*, New York, June 1993, pp. 596–597.
- [13] S. E. Hambrusch and A. A. Khokhar, "C³: A parallel model for coarse-grained machines," *J. Parallel Distrib. Comput.*, vol. 32, pp. 139–154, 1996.
- [14] S. E. Hambrusch, F. Hameed, and A. A. Khokhar, "Communication operations on coarse-grained mesh architectures," *Parallel Comput.*, vol. 21, pp. 731–751, 1995.
- [15] J. JáJá, *An Introduction to Parallel Algorithms*. Reading, MA: Addison-Wesley, 1992.
- [16] L. H. Jamieson, P. T. Mueller Jr., and H. J. Siegel, "FFT algorithms for SIMD parallel processing systems," *J. Parallel Distrib. Comput.*, vol. 3, pp. 48–71, 1986.
- [17] S. Kadambe and G. F. Boudreaux-Bartels, "Applications of the wavelet transform for pitch detection of speech signals," *IEEE Trans. Inform. Theory, Special Issue on Wavelet Transforms and Multiresolution Signal Analysis*, vol. 38, pp. 917–924, Apr. 1992.
- [18] A. A. Khokhar, "Scalable data parallel algorithms and implementations for object recognition," Ph.D. dissertation, Dept. Elect. Eng. Syst., Univ. Southern Calif., Los Angeles, Jan. 1993.
- [19] C. K. Koc, G. Chen, and C. K. Chui, "Complexity analysis of wavelet signal decomposition and reconstruction," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 30, pp. 910–918, July 1994.
- [20] J. Kovacevic and M. Vetterli, "Non-separable multidimensional perfect reconstruction filter banks and wavelet bases for R ," *IEEE Trans. Inform. Theory, Special Issue on Wavelet Transforms and Multiresolution Signal Analysis*, vol. 38, pp. 533–555, Apr. 1992.
- [21] D. Krishnaswamy and M. Orchard, "Parallel algorithms for the two-dimensional discrete wavelet transform," in *Proc. Int. Conf. Parallel Process.*, St. Charles, IL, Aug. 1994, pp. III-47–III-54.
- [22] J. S. Lienard and D. d'Alessandro, "Wavelets and granular analysis of speech," in *Wavelets: Time-Frequency Methods and Phase Space*, 2nd ed, J. M. Combes, A. Grossman, and P. Tchamitchian, Eds. New York: Springer-Verlag, 1989.
- [23] J. Lu, "Parallelizing Mallat algorithm for 2-D wavelet transforms," *Inform. Processing Lett.*, vol. 45, no. 5, pp. 255–259, Apr. 1993.
- [24] S. G. Mallat, "Multifrequency channel decompositions of images and wavelet models," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 2091–2110, Dec. 1989.
- [25] —, "Multiresolution approach to wavelets in computer vision," in *Wavelets: Time-Frequency Methods and Phase Space*, 2nd ed, J. M. Combes, A. Grossman, and P. Tchamitchian, Eds. New York: Springer-Verlag, 1989, pp. 313–327.
- [26] —, "Multiresolution approximations and wavelet orthonormal bases of $L^2(R)$," *Trans. Amer. Math. Soc.*, vol. 315, no. 1, pp. 69–87, 1989.
- [27] —, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 11, pp. 674–693, July 1989.
- [28] Y. Meyer, "Ondelettes et fonctions splines," in *Sem. Equations aux Dérivées Partielles*, Paris, France, Dec. 1986.
- [29] —, "Orthonormal wavelets," in *Wavelets: Time-Frequency Methods and Phase Space*, 2nd ed, J. M. Combes, A. Grossman, and P. Tchamitchian, Eds. New York: Springer-Verlag, 1989, pp. 21–37.
- [30] M. Misra and T. Nichols, "Computation of the 2-d wavelet transforms on the connection machine-2," in *Proc. IFIP WG10.3 Working Conf. Parallel Distrib. Comput.*, Caracas, Venezuela, Apr. 1994, pp. 3–12.
- [31] J. N. Patel and L. H. Jamieson, "Evaluating scalability of the 2-D FFT on parallel computers," in *Proc. Comput. Architectures Machine Perception*, New Orleans, LA, Dec. 1993, pp. 109–116.
- [32] J. N. Patel, A. A. Khokhar, and L. H. Jamieson, "Scalability of 2-D wavelet transform algorithms: Analytical and experimental results on fine-grained parallel computers," in *Proc. Int. Conf. Parallel Processing*, Bloomingdale, IL, Aug. 1996.
- [33] J. N. Patel, "Scalable high performance computing: A poly-algorithmic approach to computer vision and image processing libraries on parallel systems," Ph.D. dissertation, Sch. Elect. Comput. Eng., Purdue Univ., West Lafayette, IN, May 1996.
- [34] O. Rioul and P. Duhamel, "Fast algorithms for discrete and continuous wavelet transforms," *IEEE Trans. Inform. Theory, Special Issue on Wavelet Transforms and Multiresolution Signal Analysis*, vol. 38, pp. 569–586, Apr. 1992.
- [35] O. Rioul and M. Vetterli, "Wavelets and signal processing," *IEEE Signal Processing Mag.*, vol. 8, pp. 14–38, Oct. 1991.
- [36] P. Srinivasan and L. H. Jamieson, "Variable rate speech coding using the discrete time wavelet extrema representation," in *Proc. Asilomar Conf. Signals, Syst., Comput.*, Monterey, CA, Nov. 1995.
- [37] A. J. van der Steen and J. Dongarra. (1999). *Overview of Recent Supercomputers*. [Online]. Available: <http://www.netlib.org/utk/papers/advanced-computers/overview98.html>
- [38] L. R. C. Suzuki, J. R. Reid, T. J. Burns, G. B. Lamont, and S. K. Rogers, "Parallel computation of 3d wavelets," in *Proc. Scalable High-Performance Comput. Conf.*, Knoxville, TN, May 1994, pp. 454–461.
- [39] M. Vetterli and C. Herley, "Wavelets and filter banks: Theory and design," *IEEE Trans. Signal Processing*, vol. 40, pp. 2207–2232, Sept. 1992.
- [40] L. G. Weiss, "Wavelets and wideband correlation processing," *IEEE Signal Processing Mag.*, pp. 13–32, Jan. 1994.
- [41] R. Wilson, A. D. Calway, and E. R. S. Pearson, "A generalized wavelet transform for fourier analysis: The multiresolution fourier transform and its applications to image and audio signal analysis," *IEEE Trans. Inform. Theory, Special Issue on Wavelet Transforms and Multiresolution Signal Analysis*, vol. 38, pp. 674–690, Apr. 1992.



Jamshed N. Patel received the B.E. degree in electrical engineering from the University of Bombay, Bombay, India, in 1988 and the M.S. degree in electrical engineering from Vanderbilt University, Nashville, TN, in 1990. He received the Ph.D. degree in electrical and computer engineering from Purdue University, West Lafayette, IN, in 1996.

Since graduation, he has worked in the Massively Parallel Products Division and the Advanced Technology Division, Oracle Corporation, Redwood Shores, CA. He currently manages the Intel Architecture Product Division at Oracle. His research interests include parallel and distributed computing, database design, image processing, and multimedia technologies.



Ashfaq A. Khokhar received the B.S. degree in electrical engineering from the University of Engineering and Technology, Lahore, Pakistan, in 1985 and the M.S. degree in computer engineering from Syracuse University, Syracuse, NY, in 1989. He received the Ph.D. degree in computer engineering from University of Southern California, Los Angeles, in 1993.

He then spent two years as a Research Assistant Professor with the Department of Computer Sciences and School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN.

He joined University of Delaware, Newark, in 1995 and served first as an Assistant Professor and then as an Associate Professor in the Department of Electrical and Computer Engineering. Recently, he joined the University of Illinois, Chicago, as an Associate Professor with the Department of Electrical and Computer Engineering and Department of Computer Sciences. He has published over 70 technical papers in refereed conferences and journals in the area of parallel computing, image processing, computer vision, and multimedia systems. His research interests include search and retrieval for Internet data, multimedia systems and communication, multidimensional spatial databases, datamining, and high-performance computing.

Dr. Khokhar received the NSF CAREER award in 1998. His paper entitled "Scalable S-to-P broadcasting in message passing MPP's" won the Outstanding Paper award at International Conference on Parallel Processing in 1996.



Leah H. Jamieson (S'75-M'77-SM'91-F'93) received the S.B. degree in mathematics in 1972 from the the Massachusetts Institute of Technology, Cambridge, and the M.A. and M.S.E. degrees in 1974 and the Ph.D. degree in 1977, all in electrical engineering and computer science, from Princeton University, Princeton, NJ.

Since 1976, she has been on the faculty at Purdue University, West Lafayette, IN, where she is currently Professor of Electrical and Computer Engineering and Co-Director of the Engineering

Projects in Community Service-EPICS-Center. Her research interests include speech analysis and recognition and parallel algorithms for digital speech, image, and signal processing. She has authored over 160 journal and conference papers in these areas and has co-edited books on *Algorithmically-Specialized Parallel Computers* (New York: Academic, 1985) and *The Characteristics of Parallel Algorithms* (Cambridge, MA: MIT Press, 1987).

Dr. Jamieson was President of the IEEE Signal Processing Society from 1998 to 1999 and Chair of the IEEE Society's VLSI for Signal Processing Technical Committee from 1993 to 1994. She currently chairs the IEEE Technical Activities Board (TAB) Periodicals Committee and serves on the IEEE Publications Activities Board's Strategic Planning Committee. She has been an Associate Editor for the IEEE TRANSACTIONS ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING and the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED PROCESSING and is a member of the editorial board for the PROCEEDINGS OF THE IEEE. She is a member of the Advisory Committee for the NSF Directorate for Computer and Information Science and Engineering (CISE) and Secretary of the Board of Directors of the Computing Research Association (CRA). She is a past co-chair (1996-1999) of CRA's Committee on the Status of Women in Computing Research. She has been an IEEE Signal Processing Society Distinguished Lecturer and an IEEE Computer Society Distinguished Visitor. She was awarded the 2000 Violet B. Haas Award from Purdue's Council on the Status of Women, the 1997 Chester F. Carlson Award for Innovation in Engineering Education from the American Society for Engineering Education (ASEE), and an IEEE Third Millennium Medal.