Non-RAM-Based Architectural Designs of Wavelet-Based Digital Systems Based on Novel Nonlinear I/O Data Space Transformations

Dongming Peng and Mi Lu, *Senior Member, IEEE*

***Abstract*—The designs of application specific integrated circuits and/or multiprocessor systems are usually required in order to improve the performance of multidimensional applications such as digital-image processing and computer vision. Wavelet-based algorithms have been found promising among these applications due to the features of hierarchical signal analysis and multiresolution analysis. Because of the large size of multidimensional input data, off-chip random access memory (RAM) based systems have ever been necessary for calculating algorithms in these applications, where either memory address pointers or data preprocessing and rearrangements in off-chip memories are employed. This paper establishes and follows novel concepts in data dependence analysis for generalized and arbitrarily multidimensional wavelet-based algorithms, i.e., *the wavelet-adjacent field* and *the super wavelet-dependence vector*. Based on them, a series of novel nonlinear *I/O data space* transformations for variable localization and dependence graph regularization for wavelet algorithms is proposed. It leads to general designs of non-RAM-based architectures for wavelet-based algorithms where off-chip communications for intermediate calculation results are eliminated without preprocessing or rearranging input data.**

***Index Terms*—Dependence graph, discrete wavelet transform, non-RAM-based architectures, zerotree coding.**

I. Introduction

THE DESIGNS of application specific integrated circuits (ASIC) and/or multiprocessor systems are usually required in order to improve the performance of multidimensional applications such as multimedia processing, computer vision, high-definition television, medical imaging, and remote sensing, etc. Wavelet-based algorithms have been found promising among these digital signal processing applications due to the features of hierarchical signal analysis and multiresolution analysis. There have been many useful wavelet-based multidimensional algorithms studied in literature including multi-wavelet transform (MWT) [1], [10], [11], [25], [27], wavelet packet transform (WPT) [2], [15], embedded zerotree wavelet transform (EZW) [13], set partitioning in hierarchical trees (SPIHT) [14], and space frequency quantization (SFQ) [15], [16], etc. In these algorithms, multidimensional data are decomposed into different spectral subbands, and correlations across the subbands are further analyzed and exploited in coding systems. The calculations of these complex algorithms are based on intense and complicated manipulation of multidimensional data. For instance, the algorithms of EZW, SPIHT, and SFQ have three common procedures: 1) hierarchical wavelet decompositions; 2) construction of zerotree data structures; and 3) symbol generation from the wavelet coefficients on zerotrees, quantization of the magnitudes of significant coefficients and entropy coding. The second procedure, i.e., the zerotree construction, is the most important one that efficiently encodes the coefficients with a number of symbols by exploiting the inter-subband correlations of DWT via the zerotree data structure. Because the zerotrees are created from the two-dimensional (2-D) data generated by DWT, in applications it is difficult locating the corresponding parent coefficient for a given child coefficient among the 2-D data [20]. For another instance, the application of 2-D MWT on images involves preprocessing images into 2-D vector-valued data streams, convoluting groups of data from adjacent rows with the matrix-valued wavelet filter taps, and then convoluting groups of data from adjacent columns in the result of row-wise convolution. In calculating these algorithms, off-chip random access memory (RAM) based systems have been necessary where either memory address pointers or data rearrangements in off-chip memories are employed, because of the large size of 2-D input data. As a matter of fact, to the best of our knowledge, all recently proposed special-purpose architectures (e.g., [18]–[20]) for these complex algorithms have to be involved with large off-chip RAMs when calculating and rearranging multidimensional data.

In this paper, we contribute to embedding the main bodies of these algorithms into non-RAM-based architectures leading to the elimination of off-chip communications, and thus, the increase of the processing rates that are especially desirable in image and video coding. For example, one of our main ideas in building zerotrees for the algorithms of EZW, SPIHT, or SFQ, is to rearrange the calculations of wavelet transform and take advantage of parallel as well as pipelined processing, so that **ANY** parent coefficient and its children coefficients in zerotrees are guaranteed to be calculated and output simultaneously during the computation of wavelet transform. In this way, neither locating the parent for a given child coefficient, nor building zerotree data structures is necessary any more after the computation of wavelet transforms. In other words, we combine the procedures of wavelet transform and zerotree construction

Manuscript received April 11, 2001; revised July 16, 2002.

D. Peng is with the Computer and Electronics Engineering Department, University of Nebraska–Lincoln, Omaha, NE 68182 USA (e-mail: dpeng@unl-notes.unl.edu).

M. Lu is with the Electrical Engineering Department, Texas A&M University, College Station, TX 77943 (e-mail: mlu@ee.tamu.edu).

Digital Object Identifier 10.1109/TVLSI.2004.840405

into a single routine without using RAMs. The same philosophy can be applied to other wavelet-based algorithms described above. Principles of non-RAM-based architectural designs for the wavelet based digital systems are proposed in the paper.

Generally, the broadcast or communication of control information is one of the bottlenecks that block the increase of processing rates and hardware efficiency in parallel processing systems. This paper proposes architectures featured with localized control, where algorithms are computed by some processing units, and all devices operate independently with local controls, except when using a single global clock signal.

Prior to presenting architectural designs, this paper establishes and follows two novel concepts in data dependence analysis for generalized and arbitrarily multidimensional wavelet-based algorithms, i.e., *wavelet-adjacent field, and super wavelet-dependence vector*. Based on these, novel nonlinear *I/O data space* transformations for variable localization and dependence graph regularization for wavelet algorithms are proposed which lead to designs of non-RAM-based architectures.

Unlike regular iterative algorithms that can be linearly mapped onto efficient regular architectures by conventional space-time mapping techniques, wavelet-based algorithms are characterized by a rather irregular data dependence structure largely due to sequence decimations, and the efficient space-time maps are bound to employ a certain form of nonlinearity. A thorough design space exploration can be accomplished for the linear synthesis in choosing linear space-time mapping functions, but selecting appropriate nonlinear mapping functions in architectural synthesis is still an open problem. A nonlinear index space transformation applied to synthesizing parallel structures for one-dimensional (1-D) DWT has been reported in [23]. However, the approach in [23] is heuristic but not systematic, and it is ONLY applicable to 1-D cases and cannot be extended to architectural designs for generalized and arbitrarily multidimensional wavelet algorithms. The major contributions of this paper are exploring the computation locality and dependency within general wavelet-based algorithms by representing them with *wavelet-adjacent fields and super dependence vectors* based upon a newly defined model of "I/O data space," then regularizing and merging the dependence graphs via novel nonlinear I/O data space transformations, and finally, proposing non-RAM-based architectures with appropriate space-time mapping techniques, based on the transformed dependence graphs.

Although, the data dependence analysis and dependence graph regularization are proposed in this paper as a theoretical basis of architectural designs for generalized wavelet-based algorithms, with the space limit of this paper, it is impossible here to derive architectural designs for all complex wavelet-based algorithms. In this paper, we use the zerotree construction algorithm as the representative of wavelet-based algorithms for deriving corresponding architectures. Zerotree construction algorithm is the main body of such algorithms as EZW, SPIHT, and SFQ ([13]–[16]). The schemes in architectural designs in this paper form the basis for synthesizing other more generalized wavelet-based digital systems. As such, we mention full wavelet transform [17], M-ary wavelet transform [17], and embedded zerotree coded multi-wavelet [10]. Interested readers may refer to [28]–[31] for our detailed introductions of specific designs.

The rest of the paper is organized as follows. In Section II, we define I/O data space modeling of wavelet based algorithms, conduct a general data dependence analysis of these algorithms based on newly-defined concepts, and propose novel nonlinear I/O data space transformations for regularizing and merging the inter-octave dependence graphs. The non-RAM-based architectural designs for zerotree construction in the class of wavelet zerotree coding algorithms are proposed in Section III. Then, we detail experimental results and performance analysis and finally draw conclusions in Sections IV and V.

II. Novel Nonlinear I/O Data Space Transformations for Regularizing Dependence Graphs of Wavelet-Based Algorithms

A. *I/O Data Space Modeling of Wavelet-Based Algorithms*

The basic equation for any discrete wavelet algorithms is generally represented by

$$X_{j+1}[t] = \sum_{k\in L} C[k]X_j[Mt-k] \tag{1}$$

where $C[k]$ are taps of a wavelet filter, X_j and X_{j+1} are the *sequence* of input data and output data, respectively, at the $(j+1)$th level transform, L is a set that corresponds to the size of the wavelet filter, and M is a constant scalar in the algorithm.

Generally, the algorithm is termed as *M-ary wavelet transform* when $M > 2$. There are M wavelet filters for M-ary wavelet transform. j is always used in this paper to refer to the wavelet transform level. If X_j and X_{j+1} are scalars and C is scalar-valued taps of the wavelet filter, the algorithm is a classical scalar wavelet transform; if X_j and X_{j+1} are vector-valued data and C is matrix-valued taps of the multiwavelet filter, it is an MWT. If t and k are scalars, the algorithm is a 1-D transform; if t and k are n-component vectors, it is an n-D transform. Wavelet-based algorithms are multiresolution algorithms, i.e., the output data at a level of transform can be further transformed at the next level. In the classical DWT, there are two wavelet filters (low-pass filter and high-pass filter) at each level of transform, and only the output of low-pass filter is further transformed at the next level. In the arbitrary wavelet tree expansion of the WPT, the output of either filter may be further transformed at the next level. Note that the domain over which the sum is calculated by (1) is centered at $X_j[Mt]$.

1) Parameter Index Axis: The *parameter index axis* of a signal processing algorithm is the index axis for those data to be broadcasted in the algorithm, i.e., the data used in most computations but not generated by computations. As parameters of the computations, the number of them is fixed.

2) Data Index: The *data index* is the index for the intermediate data, input data, or output data that are generated and/or used by computations. In signal processing algorithms, the input size is variable.

We propose the following definitions for our modeling of wavelet algorithms in the n-D I/O data space.

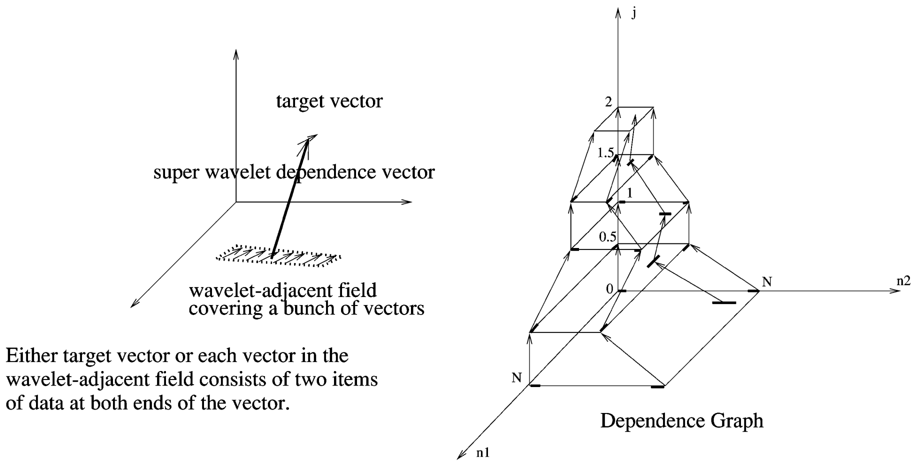

target vector
super wavelet dependence vector
wavelet-adjacent field covering a bunch of vectors
Either target vector or each vector in the wavelet-adjacent field consists of two items of data at both ends of the vector.
j
2
1.5
1
0.5
0
N
n2
N
n1
Dependence Graph

Fig. 1. Example of an dependence graph for the 2-D MWT modeled in I/O data space.

3) I/O Data Space: In an *I/O data space*, the indexed data are only possibly the input data and output data, and the parameters of the algorithm are ignored. The intermediate data are viewed as partial inputs or partial outputs for the intermediate computations.

4) Wavelet-Adjacent Field: In an I/O data space, a *wavelet-adjacent field* is a small domain made up of a group of source data items used by a calculation in (1). Its size is dependent on the wavelet filter.

5) Super Wavelet-Dependence Vector: A *super wavelet-dependence vector* $\overrightarrow{d_{Wb}}$ starts from a wavelet-adjacent field W and ends at the resulted data b. Since the source of the "dependence vector" itself is a domain instead of a single datum, we term such a dependence vector [corresponding to the calculation in (1)] a *super wavelet-dependence vector*. In later analysis, the super wavelet-dependence vectors are generally called dependence vectors and treated similarly as traditional dependence vectors. The length of a super wavelet-dependence vector $|\overrightarrow{d_{Wb}}|$ is defined as the Euclidean distance between a and b_c, where a and b_c are arithmetic centers of W and b, respectively.

We also refine the following concepts which are used throughout this paper.

6) Dependence Graph: Although, there are many versions of the definitions of the dependence graph, in this paper, we make an emphasis on the *dependence graph based on an I/O data space*, where each node in the dependence graph corresponds to a data item, and each edge corresponds to a calculation or a dependence relation between the data used in the calculation and that generated in the calculation.

7) Regular Dependence Graphs: In such dependence graphs, the length of each dependence vector $\mathbf{d}$ is a constant value independent of either the input size or the data positions.

8) Pseudo Regular Dependence Graphs: In such dependence graphs the dependence vectors can be partitioned into a certain number of groups and in each group the length of dependence vectors is a constant value independent of either the input size or the data positions.

As examples, a wavelet-adjacent field, a super wavelet-dependence vector, and the dependence graph including some instances of dependence vectors for the algorithm of *separable* 2-D MWT [11], [27] are shown in Fig. 1, based on these concepts. The term "separable" means that the 2-D wavelet transform can be done row-wise and column-wise, consecutively. In Fig. 1, j is the transform level; n_1 and n_2 are indices for the 2-D input data or 2-D output data. In (1) for the MWT, both X_j and X_{j+1} are vector-valued and C corresponds to the matrix-valued multiwavelet filter. Thus, the wavelet-adjacent field is a bunch of vectors and the end of the super wavelet-dependence vector itself is a vector (X_{j+1}) in Fig. 1. The separable 2-D MWT is performed row-wise and column-wise separately at each level of transform, and the dependence graph shown in Fig. 1 presents the dependence relationships in the I/O data space. Index j corresponds to the multiwavelet transform level. In Fig. 1, Plane $j = 0$ in the 3-D space is the input plane which means the input data is always located at this plane, $j = 2$ is the output plane which means the final output data of the algorithm is always located at this plane, and $j = 1$ is an intermediate data plane. Plane $j = 1/2$ or $j = 3/2$ is the output plane of a row-wise transform and meanwhile the input plane of a column-wise transform. The dependence graph in Fig. 1 is apparently not a regular dependence graph, as the length of dependence vectors is not a constant value yet depends on the positions of the target of the dependence vector.

B. Novel Nonlinear I/O Data Space Transformations for Regularizing Dependence Graphs

As illustrated in Fig. 1, the dependence graph for a wavelet-based algorithm in the I/O data space is irregular in that the lengths of the dependence vectors are dependent on the data positions as well as the input size. Whether it be a classical DWT, a vector-valued transform MWT, an arbitrary wavelet expansion WPT, or other wavelet-based algorithm, this irregularity always exists due to the sequence decimation in the nature of each level wavelet transform. Meanwhile, the output size is always M^n times less than the input size of each level transform (where n is the number of dimensions of the wavelet transforms). However, as proposed later, these irregular dependence graphs in the I/O data space can be generally regularized through a group of novel *nonlinear I/O data space transformations* so that the output data is uniformly distributed and the dependence vectors are regularized. The general dependence regularizations are formulated in the following proofs of Theorem 1 and Theorem 2.

Theorem 1: The dependence graphs of wavelet algorithms modeled in I/O data space can always be regularized through appropriate nonlinear I/O data space transformations.

Proof: The proof can be presented in two cases:

(1) For algorithms of 1-D wavelet transforms and nonseparable n-D wavelet transforms: According to (1), the wavelet adjacent fields in the I/O data space correspond to groups of data $X_j[Mt-k]$ where $k \in L$, and L is a set that depends on the size of the wavelet filter and the number of the dimensions of the wavelet transform. The super dependence vectors represent the dependence relationships between the wavelet adjacent fields and the target data $X_{j+1}[t]$, and the lengths of dependence vectors are the distances between centers of the wavelet adjacent field and the target data. When performing the nonlinear I/O data space transformation Γ_1: $j \mapsto j$, $t \mapsto M^j t$, we can always make the lengths of dependence vectors a constant value, and thus, regularize the dependence graph as analyzed in the following. After rearranging data, the dependence vector in the I/O data space corresponding to the dependence relation between $X_{j+1}[t]$ and the wavelet-adjacent filed $X_j[Mt-k]$ (where $k \in L$) changes to the dependence vector for the dependence relation between $X_{j+1}[M^{j+1}t]$ and the wavelet-adjacent field $X_j[M^{j+1}t - M^j k]$ (where $k \in L$). Meanwhile, the length of this dependence changes to the distance between $X_{j+1}[M^{j+1}t]$ and $X_j[M^{j+1}t]$, or $|(j+1)-j| = 1$. In other words, the dependence graph is regularized by performing the nonlinear I/O data space transformation Γ_1.

(2) For separable n-D wavelet transforms: The multidimensional filter $C[K]$ is separable for the separable n-D wavelet transforms, and (1) is separated to be calculated in each dimension. Generally, (1) is calculated equivalently by the following (2) in separable n-D wavelet transforms:

$$\begin{aligned} X_{j+1}[t_1, t_2, \ldots, t_n] &= \sum_{k_1 \in L_1} C_1[k_1] \sum_{k_2 \in L_2} C_2[k_2] \cdots \sum_{k_n \in L_n} C_n[k_n] \\ &\quad \times X_j[Mt_1 - k_1, Mt_2 - k_2, \ldots, Mt_n - k_n] \end{aligned} \tag{2}$$

where $(t_1, t_2, \ldots, t_n)$ are components of the n-component vector $t, (k_1, k_2, \ldots, k_n)$ are the components of $k, (Mt_1 - k_1, Mt_2 - k_2, \ldots, Mt_n - k_n)$ are the components of $Mt - k$ in (1), $C_1, C_2, \ldots, C_n$ are the n 1-D filters separated from the n-D filter C, and $L_1, L_2, \ldots, L_n$ correspond to n 1-D wavelet-adjacent fields separated from the original n-D wavelet-adjacent field. In order to give the dependence graph for (2) in the I/O data space, we draw the index j [which represents the level of multiresolution transforms and can only be integers in case (1)] in fractional numbers to represent the intermediate calculations in every level of transform. For example, the plane $j = 0.5$ in Fig. 1 corresponds to the output of the first level row-wise wavelet transform and the input of the first level column-wise wavelet transform. In the $(s+1)$ level (where s is a nonnegative integer) of an n-D wavelet transform, we thus have intermediate planes $j = s + 1/n, j = s + 2/n, \ldots, j = s + (n-1)/n$ between the plane $j = s$ and $j = s+1$. Accordingly, (2) can be calculated in the order of

$$\begin{aligned} &\sum_{k_1 \in L_1} C_1[k_1] \sum_{k_2 \in L_2} C_2[k_2] \cdots \sum_{k_n \in L_n} C_n[k_n] \\ &\quad \times X_s[Mt_1 - k_1, Mt_2 - k_2, \ldots, Mt_n - k_n] \\ &= \sum_{k_2 \in L_2} C_2[k_2] \cdots \sum_{k_n \in L_n} C_n[k_n] \\ &\quad \times X_{s+1/n}[t_1, Mt_2 - k_2, \ldots, Mt_n - k_n] \\ &= \sum_{k_3 \in L_3} C_3[k_3] \cdots \sum_{k_n \in L_n} C_n[k_n] \\ &\quad \times X_{s+2/n}[t_1, t_2, Mt_3 - k_3, \ldots, Mt_n - k_n] \\ &= \cdots \\ &= \sum_{k_n \in L_n} C_n[k_n] X_{s+(n-1)/n}[t_1, t_2, \ldots, t_{n-1}, Mt_n - k_n] \\ &= X_{s+1}[t_1, t_2, \ldots, t_n]. \end{aligned} \tag{3}$$

When performing the nonlinear I/O data space transformation $\Gamma_2 : j \mapsto j, t_i \mapsto M^{\lceil j-(i/n)+(1/2n) \rceil} \times t_i$, for $i = 1, 2, \ldots, n$, we can make the lengths of dependence vectors a constant value, and thus, regularize the dependence graph corresponding to (3) as analyzed in the following. Here, $(1/2n)$ is used in the expression for adjusting the value of ceiling function. Since $\lceil j - (i/n) + (1/2n) \rceil = s+1$ when $j \in [s + (i/n), s + 1 + (i/n) - (1/2n)]$, where s is an integer, the calculation of the ith step in the calculation of (3) (for the $(s+1)$ level of the separable wavelet transform)

$$\begin{aligned} &\sum_{k_i \in L_i} C_i[k_i] \sum_{k_{i+1} \in L_{i+1}} C_{i+1}[k_{i+1}] \cdots \sum_{k_n \in L_n} C_n[k_n] \\ &\quad \times X_{s+(i-1)/n}[t_1, t_2, \ldots, Mt_i - k_i, \\ &\quad\quad Mt_{i+1} - k_{i+1}, \ldots, Mt_n - k_n] \\ &= \sum_{k_{i+1} \in L_{i+1}} C_{i+1}[k_{i+1}] \cdots \sum_{k_n \in L_n} C_n[k_n] \\ &\quad \times X_{s+i/n}[t_1, t_2, \ldots, t_i, Mt_{i+1} - k_{i+1}, \ldots, Mt_n - k_n] \end{aligned}$$

changes to

$$\begin{aligned} &\sum_{k_i \in L_i} C_i[k_i] \sum_{k_{i+1} \in L_{i+1}} C_{i+1}[k_{i+1}] \cdots \sum_{k_n \in L_n} C_n[k_n] \\ &\quad \times X_{s+(i-1)/n}[M^{s+1}t_1, M^{s+1}t_2, \ldots, M^s(Mt_i - k_i) \\ &\quad\quad M^s(Mt_{i+1} - k_{i+1}), \ldots, M^s(Mt_n - k_n)] \\ &= \sum_{k_{i+1} \in L_{i+1}} C_{i+1}[k_{i+1}] \cdots \sum_{k_n \in L_n} C_n[k_n] \\ &\quad \times X_{s+i/n}[M^{s+1}t_1, M^{s+1}t_2, \ldots, M^{s+1}t_i \\ &\quad\quad M^s(Mt_{i+1} - k_{i+1}), \ldots, M^s(Mt_n - k_n]) \end{aligned}$$

after the mapping of Γ_2.

Thus, after the transformation Γ_2, the dependence vector, starting from the wavelet-adjacent field which corresponds to L_i and is centered at data

$$\begin{aligned} &X_{s+(i-1)/n}[M^{s+1}t_1, M^{s+1}t_2, \ldots, M^{s+1}t_i \\ &\quad\quad M^s(Mt_{i+1} - k_{i+1}), \ldots, M^s(Mt_n - k_n)] \end{aligned}$$

is targeted to data

$$X_{s+i/n}[M^{s+1}t_1, M^{s+1}t_2, \ldots, M^{s+1}t_i \\ M^s(Mt_{i+1}-k_{i+1}), \ldots, M^s(Mt_n-k_n)].$$

Thus, the length of the dependence vector is $|(s+i/n)-(s+(i-1)/n)| = 1/n$. In other words, the dependence graph for the calculation of the separable wavelet transform is regularized by performing the nonlinear I/O data space transformation Γ_2. $\square$

Note that we have not assumed that the data calculated in the wavelet transforms are scalar-valued, so the proof is also applicable to MWT.

In the algorithm of WPT, the computation has a structure of an arbitrary wavelet tree expansion, and the algorithm is calculated by iterating the wavelet branches of the filter bank to give a finer resolution to the wavelet decomposition, where not only the coarse component but also the detailed component is possibly further decomposed. The classical DWT can be taken as a special instance of WPT, where only the coarse component is recursively decomposed at each level of transform. The calculation for any wavelet filter in the computation of WPT follows (1), with C replaced by different wavelet filters. We can construct the dependence graphs corresponding to the calculations of WPT, and regularize them similarly by nonlinear I/O data space transformations Γ_1 and Γ_2 as introduced above. Nevertheless, each dependence graph represents only one path in the arbitrary expansion of wavelet tree, and the whole dependence graph of WPT should be the combination of dependence graphs corresponding to all paths in the expanded wavelet tree of WPT. Such process of combining the dependence graphs via nonlinear I/O data space transformations is formulated in the following Proof of Theorem 2.

Theorem 2: The dependence graphs of wavelet-packet based algorithms modeled in I/O data space can always be merged and regularized to a pseudo regular dependence graphs via appropriate nonlinear I/O data space transformations.

Proof: The symbols j, t, X, L, $t_1, t_2, \ldots, t_n$, L_1, $L_2, \ldots, L_n$ have the same meanings as in the previous paragraphs. The proof can be presented in three cases:

(1) For algorithms of 1-D transforms: There are *M* wavelet filters $(f_1, f_2, \ldots, f_M)$ at each level of 1-D M-ary wavelet transform, and each level of transform can decompose a certain subband into *M* components in wavelet-packet based algorithms. One of the filters (f_1) is for generating coarse component, others for detailed components. Assume *M* functions $F_i(x) = Mx + i - 1$ for $i = 1, 2, \ldots, M$.

Suppose that a subband Π is calculated in l levels of wavelet-packet based transform consecutively with wavelet filters $p_1, p_2, \ldots, p_l$, where $p_u = f_i$ for $u = 1, 2, \ldots, l$ and i is any integer $\in [1, M]$. Since the calculation of each level of transform follows the same format in (1), the corresponding dependence graph of Π can be constructed and regularized via the I/O data space transformation similarly as in the Proof of Theorem 1. However, considering that there are many subbands generated together by l levels of wavelet-packet based transform, and their corresponding dependence graphs should be merged as well as regularized to get a whole dependence graph for the algorithm, the nonlinear I/O data space transformation Γ_3 is presented as follows. Without loosing generality, for the dependence graph corresponding to subband Π, Γ_3 is: $j \mapsto j; t \mapsto t$ if $j = 0; t \mapsto P_1(P_2(\ldots(P_j(t))\ldots))$ otherwise, where $P_u = F_i$ if $p_u = f_i$ for $u = 1, 2, \ldots, j$, and $j \le l, i \in [1, M]$. Note that here j corresponds to the level of transform and can be only integers.

Consider another subband Π_1 different from Π generated in the algorithm. Suppose that Π_1 is calculated in l levels consecutively with wavelet filters $p'_1, p'_2, \ldots, p'_l$, where $p'_u = f_i$ for $u = 1, 2, \ldots, l$ and i is any integer $\in [1, M]$. Since there exits at least one $p'_u \ne p_u$ where $u \in [1, l], \Gamma_3$ maps the data for Π and Π_1 to different positions in the I/O data space. In other words, Γ_3 can combine all dependence graphs of the subbands into a single I/O data space without conflicts.

Consider a dependence vector in the I/O data space corresponding to a calculation of (1) at data position t_0 at the $(u+1)$ level of transform

$$X_{u+1}[t_0] = \sum_{k \in L} p_{u+1}[k] X_u[Mt_0 - k]$$

where p_{u+1} represents the wavelet filter used at this level. After the mapping of Γ_3, the calculation changes to

$$X_{u+1}[P_1(P_2(\ldots(P_u(P_{u+1}(t_0)))\ldots))] \\ = \sum_{k \in L} p_{u+1}[k] X_u[P_1(P_2(\ldots(P_u(Mt_0 - k))\ldots))]$$

where $P_v = F_i$ if $p_v = f_i$ for $v = 1, 2, \ldots, u+1$, and $i \in [1, M]$. The dependence vector, starting from the wavelet-adjacent field which corresponds to L and is centered at data $X_u[P_1(P_2(\ldots(P_u(Mt_0))\ldots))]$, is targeted to data $X_{u+1}[P_1(P_2(\ldots(P_u(P_{u+1}(t_0)))\ldots))]$. The length of the dependence vector can be resolved by the difference between their coordinates along index j and t. The difference along index j is $|u+1-u| = 1$. The difference along t is

$$|P_1(\ldots(P_u(P_{u+1}(t_0)))\ldots)) - P_1(\ldots(P_u(Mt_0))\ldots))| \\ = |P_1(\ldots(P_u(Mt_0 + w))\ldots)) - P_1(\ldots(P_u(Mt_0))\ldots))| \\ = M^u w$$

where w is an integer $\in [1, M-1]$. Here the length of the dependence vector is independent of t_0's value, and the number of transform levels and the number of wavelet filters (M) remain constant in the algorithm. In other words, the lengths of the dependence vectors in the I/O data space after the mapping of Γ_3 are bounded and independent of the data positions and input size. Moreover, the dependence vectors can be partitioned into a finite number of groups (according to the possible values of w and u), and the lengths of the dependence vectors in each group are the same. That is, the dependence graphs for 1-D wavelet-packet based algorithm are combined and regularized to be a pseudo regular dependence paragraph via the nonlinear I/O data space transformation Γ_3.

(2) For nonseparable n-D transforms: There are $Q = M^n$ different wavelet filters $(f_1, f_2, \ldots, f_Q)$ at each level of n-D M-ary wavelet transform. Assume *Q* functions $F_i(x) = Mx + q$, where x and q are n-component vectors. The components of q are $q_1, q_2, \ldots, q_n$, and q_v is an integer $\in [0, M-1]$ for $v = 1, 2, \ldots, n$, and $i = \sum_{u=1}^{n} M^u q_u$. So $i \in [1, Q]$.

Suppose that a subband Π is calculated in l levels of n-D wavelet-packet based transform consecutively with wavelet filters $p_1, p_2, \ldots, p_l$, where $p_u = f_i$ for $u = 1, 2, \ldots, l$ and $i \in [1, Q]$. Without loosing generality, for the dependence graph corresponding to subband Π, a nonlinear I/O data space transformation Γ_4 is presented as: $j \mapsto j; t \mapsto t$ if $j = 0; t \mapsto P_1(P_2(\ldots(P_j(t))\ldots))$ otherwise, where $P_u = F_i$ if $p_u = f_i$ for $u = 1, 2, \ldots, j$, and $j \leq l, i \in [1, Q]$. Note that here j corresponds to the level of transform and can be only integers, and t represents n-component vectors.

For other subbands different from Π generated in the algorithm, since there exits at least one filter used in the calculation of l levels of transform different from that of Π, Γ_4 maps the data of them to different positions. In other words, Γ_4 can combine all dependence graphs of the subbands into a single I/O data space without conflicts.

Similar to the case (1), a calculation corresponding to the dependence vector changes to

$$\begin{aligned} X_{u+1}&[P_1(P_2(\ldots(P_u(P_{u+1}(t)))\ldots))] \\ &= \sum_{k \in L} p_{u+1}[k] X_u[P_1(P_2(\ldots(P_u(Mt-k))\ldots))] \end{aligned}$$

where $P_v = F_i$ if $p_v = f_i$ for $v = 1, 2, \ldots, u+1$, and $i \in [1, Q]$. The difference between the coordinates of the source and the target of the dependence vector along index j is $|u+1-u| = 1$. The difference along t is

$$\begin{aligned} P_1(&P_2(\ldots(P_u(P_{u+1}(t)))\ldots)) - P_1(P_2(\ldots(P_u(Mt))\ldots)) \\ &= P_1(P_2(\ldots(P_u(Mt+w))\ldots)) \\ &\quad - P_1(P_2(\ldots(P_u(Mt))\ldots)) \\ &= M^u w \end{aligned}$$

where w *is an* n*-component vector* whose components are integers $\in [1, M-1]$. Thus, the length of the dependence vector is independent of t's value. We have the similar conclusion that the lengths of the dependence vectors in the I/O data space after the mapping of Γ_4 are bounded and independent of the data positions and input size, and the dependence vectors can be partitioned into a finite number of groups (according to the possible values of w and u), and the lengths of the dependence vectors in each group are the same.

(3) For separable n*-D transforms*: As in case (2) of the proof for Theorem 1, the n-D separable transforms are calculated separately and consecutively in every dimension. The index j is drawn in fractional numbers to represent the intermediate calculations in each level of transform. In the $(s+1)$ level (where s is a nonnegative integer) of a separable n-D wavelet transforms, we have $(n-1)$ intermediate I/O data planes $j = s+1/n, j = s+2/n, \ldots, j = s+(n-1)/n$ between the planes $j = s$ and $j = s+1$. In the calculations for every dimension, there are M wavelet filters $f_1, f_2, \ldots, f_M$, and a subband may be decomposed into M components on each dimension. So after each level of transform, a subband can be decomposed into M^n components. In addition, we assume M functions $F_v(x) = Mx + v - 1$ for $v = 1, 2, \ldots, M$.

Suppose that a certain subband Π is calculated in l levels of n-D separable wavelet-packet based transform consecutively with wavelet filters $p_{1,1}, p_{1,2}, \ldots, p_{1,n}, p_{2,1}, \ldots, p_{2,n}, \ldots, p_{l,n}$, where $p_{u,i} = f_v$ for $u = 1, 2, \ldots, l$ and $i = 1, 2, \ldots, n$, and $v \in [1, M]$. $p_{u,i}$ represents the wavelet filter used for the calculation of the uth level transform on the ith dimension in generating Π. In order to regularize the dependence graphs, we present the nonlinear I/O data space transformation Γ_5 as follows. Without loosing generality, for the dependence graph corresponding to subband Π, Γ_5 is: $j \mapsto j, t_i \mapsto t_i (i = 1, 2, \ldots, n)$ if $j = 0$; $t_i \mapsto P_{1,i}(P_{2,i}(\ldots(P_{s+1,i}(t_i))\ldots))$ otherwise, with $j \in [s+i/n, s+1+i/n)$, s being an integer $\in [0, l-1]$, $P_{u,i} = F_v$ for $p_{u,i} = f_v$ ($u = 1, 2, \ldots, s+1; i = 1, 2, \ldots, n$; and $v \in [1, M]$).

For other subbands different from Π generated in the algorithm, since there exits at least one filter used in the calculation of l levels of transform different from that of Π, Γ_5 maps the data of them to different positions.

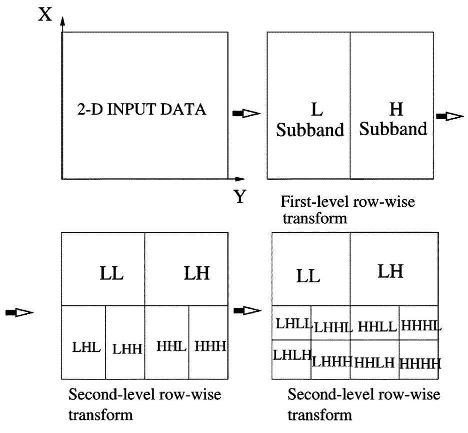
The calculation for the ith dimension in (3) (at the $(s+1)$ level of transform) for Π

$$\begin{aligned} &\sum_{k_i \in L_i} p_{s+1,i}[k_i] \sum_{k_{i+1} \in L_{i+1}} p_{s+1,i+1}[k_{i+1}] \cdots \sum_{k_n \in L_n} p_{s+1,n}[k_n] \\ &\quad \times X_{s+(i-1)/n}[t_1, t_2, \ldots, Mt_i - k_i, \\ &\quad Mt_{i+1} - k_{i+1}, \ldots, Mt_n - k_n] \\ &\qquad = \sum_{k_{i+1} \in L_{i+1}} p_{s+1,i+1}[k_{i+1}] \cdots \sum_{k_n \in L_n} p_{s+1,n}[k_n] \\ &\qquad \times X_{s+i/n}[t_1, t_2, \ldots, t_i, Mt_{i+1} \\ &\qquad - k_{i+1}, \ldots, Mt_n - k_n] \end{aligned}$$

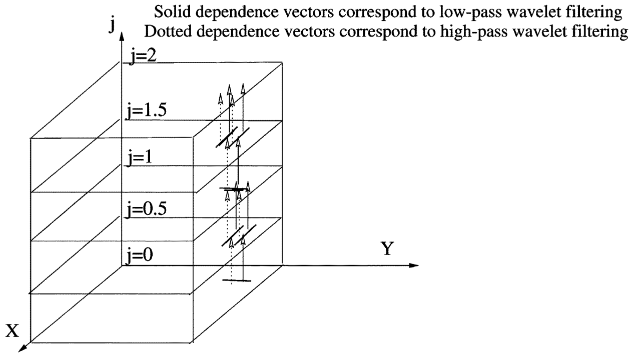
changes to calculating

$$\begin{aligned} &\sum_{k_i \in L_i} p_{s+1,i}[k_i] \sum_{k_{i+1} \in L_{i+1}} p_{s+1,i+1}[k_{i+1}] \cdots \sum_{k_n \in L_n} p_{s+1,n}[k_n] \\ &\quad \times X_{s+(i-1)/n}[P_{1,1}(P_{2,1}(\ldots(P_{s+1,1}(t_1))\ldots)) \\ &\quad P_{1,2}(P_{2,2}(\ldots(P_{s+1,2}(t_2))\ldots)), \ldots \\ &\quad P_{1,i}(P_{2,i}(\ldots(P_{s,i}(Mt_i - k_i))\ldots)) \\ &\quad P_{1,i+1}(P_{2,i+1}(\ldots(P_{s,i+1}(Mt_{i+1} - k_{i+1}))\ldots)), \ldots \\ &\quad P_{1,n}(P_{2,n}(\ldots(P_{s,n}(Mt_n - k_n))\ldots))] \\ &= \sum_{k_{i+1} \in L_{i+1}} p_{s+1,i+1}[k_{i+1}] \cdots \sum_{k_n \in L_n} p_{s+1,n}[k_n] \\ &\quad \times X_{s+i/n}[P_{1,1}(P_{2,1}(\ldots(P_{s+1,1}(t_1))\ldots)) \\ &\quad P_{1,2}(P_{2,2}(\ldots(P_{s+1,2}(t_2))\ldots)), \ldots \\ &\quad P_{1,i}(P_{2,i}(\ldots(P_{s+1,i}(t_i))\ldots)) \\ &\quad P_{1,i+1}(P_{2,i+1}(\ldots(P_{s,i+1}(Mt_{i+1} - k_{i+1}))\ldots)), \ldots \\ &\quad P_{1,n}(P_{2,n}(\ldots(P_{s,n}(Mt_n - k_n))\ldots))] \end{aligned}$$

after the mapping of Γ_5, where $P_{u,i} = F_v$ if $p_{u,i} = f_v$ for $u = 1, 2, \ldots, s+1; i = 1, 2, \ldots, n$; and $v \in [1, M]$.

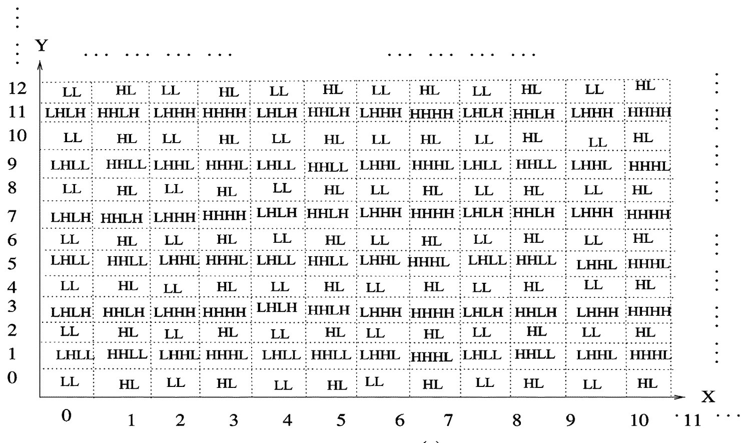

X
Y
2-D INPUT DATA
L Subband
H Subband
First-level row-wise transform
LL
LH
HL
HH
First-level column-wise transform
LHL
LHH
HHL
HHH
Second-level row-wise transform
LHLL
LHHL
HHLL
HHHL
LHLH
LHHH
HHLH
HHHH
Second-level row-wise transform

(a)


Solid dependence vectors correspond to low-pass wavelet filtering
Dotted dependence vectors correspond to high-pass wavelet filtering
j
j=2
j=1.5
j=1
j=0.5
j=0
Y
X

(b)

Row-column coordinates for data in different subbands after the nonlinear I/O data space transformation:
LL: (2x,2y)
LHLL: (4x,4y+1)
LHLH: (4x,4y+3)
LHHL: (4x+2,4y+1)
LHHH: (4x+2,4y+3)
HHLL: (4x+1,4y+1)
HHLH: (4x+1,4y+3)
HHHL: (4x+3,4y+1)
HHHH: (4x+3,4y+3)
HL: (2x+1,2y)
where x and y are non-negative integers.


Y
X
12 LL HL LL HL LL HL LL HL LL HL LL HL
11 LHLH HHLH LHHH HHHH LHLH HHLH LHHH HHHH LHLH HHLH LHHH HHHH
10 LL HL LL HL LL HL LL HL LL HL LL HL
9 LHLL HHLL LHHL HHHL LHLL HHLL LHHL HHHL LHLL HHLL LHHL HHHL
8 LL HL LL HL LL HL LL HL LL HL LL HL
7 LHLH HHLH LHHH HHHH LHLH HHLH LHHH HHHH LHLH HHLH LHHH HHHH
6 LL HL LL HL LL HL LL HL LL HL LL HL
5 LHLL HHLL LHHL HHHL LHLL HHLL LHHL HHHL LHLL HHLL LHHL HHHL
4 LL HL LL HL LL HL LL HL LL HL LL HL
3 LHLH HHLH LHHH HHHH LHLH HHLH LHHH HHHH LHLH HHLH LHHH HHHH
2 LL HL LL HL LL HL LL HL LL HL LL HL
1 LHLL HHLL LHHL HHHL LHLL HHLL LHHL HHHL LHLL HHLL LHHL HHHL
0 LL HL LL HL LL HL LL HL LL HL LL HL
0 1 2 3 4 5 6 7 8 9 10 11

(c)

Fig. 2. An example for the applications of the nonlinear I/O data space transformation on a 2-D wavelet packet transform. (a) A 2-D wavelet packet transform, (b) Some instances of super wavelet dependence vectors after I/O data space transformation, (c) A projection of data distribution along axis j.

Thus, after the transformation Γ_5, the dependence vector, starting from the wavelet-adjacent field which corresponds to L_i and is centered at data

$$X_{s+(i-1)/n}[P_{1,1}(P_{2,1}(\ldots(P_{s+1,1}(t_1))\ldots))$$

$$P_{1,2}(P_{2,2}(\ldots(P_{s+1,2}(t_2))\ldots)),\ldots$$
$$P_{1,i}(P_{2,i}(\ldots(P_{s,i}(Mt_i))\ldots)),$$
$$P_{1,i+1}(P_{2,i+1}(\ldots(P_{s,i+1}(Mt_{i+1}-k_{i+1}))\ldots)),\ldots$$
$$P_{1,n}(P_{2,n}(\ldots(P_{s,n}(Mt_n-k_n))\ldots))]$$

is targeted to data

$$\begin{aligned}X_{s+i/n}[&P_{1,1}(P_{2,1}(\ldots(P_{s+1,1}(t_1))\ldots))\\&P_{1,2}(P_{2,2}(\ldots(P_{s+1,2}(t_2))\ldots)),\ldots\\&P_{1,i}(P_{2,i}(\ldots(P_{s+1,i}(t_i))\ldots))\\&P_{1,i+1}(P_{2,i+1}(\ldots(P_{s,i+1}(Mt_{i+1}-k_{i+1}))\ldots)),\ldots\\&P_{1,n}(P_{2,n}(\ldots(P_{s,n}(Mt_n-k_n))\ldots))].\end{aligned}$$

The difference between the coordinates of the target and the source of the dependence vector along index j is $|(s+i/n)-(s+(i-1)/n)|=1/n$. The difference along t is

$$\begin{aligned}&P_{1,i}(P_{2,i}(\ldots(P_{s,i}(P_{s+1,i}(t_i)))\ldots))\\&\quad-P_{1,i}(P_{2,i}(\ldots(P_{s,i}(Mt_i))\ldots)\\&=P_{1,i}(P_{2,i}(\ldots(P_{s,i}(Mt_i+w))\ldots))\\&\quad-P_{1,i}(P_{2,i}(\ldots(P_{s,i}(Mt_i))\ldots))\\&=M^s w\end{aligned}$$

where w is an integer $\in[1,M-1]$. Thus, the length of the dependence vector is independent of t's value. We have the similar conclusion that the lengths of the dependence vectors in the I/O data space after the mapping of Γ_5 are bounded and independent of the data positions and the input size, and the dependence vectors can be partitioned into a finite number of groups (according to the possible values of w and s), and the lengths of the dependence vectors in each group are the same.

To sum up, the dependence graphs for wavelet-packet based algorithms are combined and regularized to be a pseudo regular dependence paragraph via the nonlinear I/O data space transformation Γ_3, Γ_4, or Γ_5. □

An example of nonlinear I/O data space transformation Γ_5 is illustrated in Fig. 2, where an original 2-D signal is decomposed into various subband images, and each element (or wavelet coefficient) in the subbands is relocated in the I/O data space by Γ_5.

Although, nonlinear input formats of multidimensional data such as random access or pseudofractal scan [26] also exist, the paper only considers linear input format of a multidimensional data set. This is to prevent systems from potentially involving data preprocessing and rearrangement in RAMs before computation. Suppose that the input data is n-dimensional. A group of n-component unitary orthogonal vectors $\{S_1, S_2, \ldots, S_n\}$ is used to describe a linear input format. Assume that *A* and *B* are any two samples indexed by n-component Cartesian vectors *X* and *Y*, respectively, in the multidimensional data set. (*X*, *Y*) is the inner product of two vectors *X* and *Y*. The linear input format can be described as: 1) if $(X,S_1)\neq(Y,S_1)$, the sample corresponding to the less of the two inner products will be input to the system earlier; 2) else if $(X,S_2)\neq(Y,S_2)$, the sample corresponding to the less of the two inner products will be input to the system earlier; 3)...; n) else if $(X,S_n)\neq(Y,S_n)$, the sample corresponding to the less of the two inner products will be input to the system earlier.

Now consider the dependence graphs regularized by the nonlinear I/O data space transformations proposed in this subsection. The n-dimensional input data is always located on the super plane $j=0$ shown as in Fig. 2, and it is scanned in a linearly indexed order when the data is input to the system. The I/O data space is $(n+1)$-dimensional. On super planes $j=c+i/n$ (where c and i are integers, $c>0$ and $n>i\geq 0$) located are intermediately calculated data which makes up wavelet-adjacent fields for the next level calculation. Dependence vectors starting from the wavelet-adjacent fields are located between every two neighboring super planes.

A scheme of *free schedule* is used to optimize the system performance, which schedules a calculation in the I/O data space to be executed as soon as its operands (i.e., the data in wavelet-adjacent field) are ready. Due to the dependence graph regularization via nonlinear I/O data space transformations, dependence vectors and wavelet-adjacent fields are uniformly distributed on super planes which are orthogonal to axis j. When the input data items are fed to the system one by one according to a linear input format, the wavelet-adjacent fields on plane $j=0$ are scanned one by one, and the corresponding dependence vectors which start from this plane and take these wavelet-adjacent fields as sources are ready to be processed one by one. Since dependence vectors are along the orientation of axis j due to I/O data space transformations, the calculation results on the next super plane, or the targets of these dependence vectors, can be produced one by one in a linear order which is the same as the system's linear input format, resulting in that the wavelet-adjacent fields in this super plane are also "scanned" one by one, and dependence vectors between this plane and the further next plane are ready to be processed in the same linear order, and so on. Thus, if we assign each level of algorithm computation, or each layer of dependence vectors in I/O data space, to a separate processor, have all these processors execute simultaneously, and let the processing rates match the system's data-feeding rate, we can finish the algorithm computation as soon as the feeding of input data is finished. In this scheme, we avoid using RAMs to rearrange or manipulate multidimensional input data, which was necessary in the complex computation structure of wavelet algorithms. Since any wavelet-based algorithm consists of wavelet transforms that follow (1), and basically uses data structures on the results of wavelet transforms that can be modeled and reformulated in I/O data space as proposed in this section, we can apply the above scheme to general wavelet-based algorithms without using RAMs. The following two sections give more detailed explanations in design examples for specific wavelet-based algorithms.

Because the dependence vectors and wavelet-adjacent fields are uniformly distributed on super planes which are orthogonal to axis j due to I/O data space transformations $\Gamma_1-\Gamma_5$, when the input data items are fed to the system in linear order at a constant rate, the processors, each of which is to perform calculations corresponding to each layer of dependence vectors in I/O data space, will operate periodically according to the scheme described in the above paragraph. For instance, when 2-D input data which is located at plane $j=0$ in Fig. 2, is fed to the system in row-major format, the processor that needs to perform calculations for dependence vectors between plane $j=0$ and $j=1/2$ will alternate performing low-pass and high-pass wavelet filtering periodically. The other processor that performs

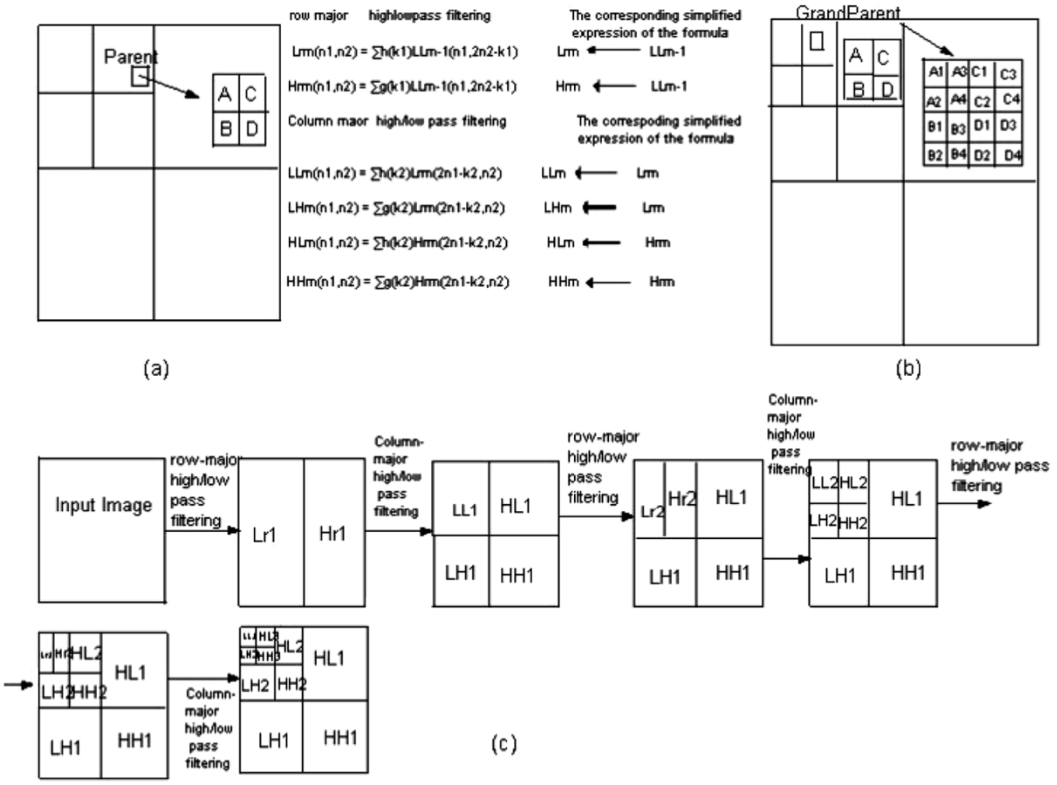

Parent
A C
B D
row major highlowpass filtering
Lrm(n1,n2) = ∑h(k1)LLm-1(n1,2n2-k1)
Hrm(n1,n2) = ∑g(k1)LLm-1(n1,2n2-k1)
Column maor high/low pass filtering
LLm(n1,n2) = ∑h(k2)Lrm(2n1-k2,n2)
LHm(n1,n2) = ∑g(k2)Lrm(2n1-k2,n2)
HLm(n1,n2) = ∑h(k2)Hrm(2n1-k2,n2)
HHm(n1,n2) = ∑g(k2)Hrm(2n1-k2,n2)
The corresponding simplified expression of the formula
Lrm ← LLm-1
Hrm ← LLm-1
The correspoding simplified expression of the formula
LLm ← Lrm
LHm ← Lrm
HLm ← Hrm
HHm ← Hrm
(a)
GrandParent
A1 A3 C1 C3
A2 A4 C2 C4
B1 B3 D1 D3
B2 B4 D2 D4
(b)
Input Image
row-major high/low pass filtering
Lr1
Hr1
Column-major high/low pass filtering
LL1
HL1
LH1
HH1
Lr2
Hr2
LL2
HL2
LH2
HH2
(c)

Fig. 3. The zerotree construction in the EZW algorithm. (a) The Relation of parent-children in two level DWT. (b) The Relation of parent-children in three-level DWT. (c) The three-level separable 2-D DWT.

calculations for dependence vectors between planes $j = 1/2$ and $j = 1$ has the similar feature of periodic operations. This feature of periodic operations of parallel processors, which is independent of input data or intermediate calculation results, leads to control-localized parallel processing architectures.

III. Design Example: Non-RAM-Based Architectures for Zerotree Construction in Wavelet Zerotree Coding Systems

A. Zerotree Coding Algorithm

Zerotree coding is the common and the most important part of algorithms EZW, SPIHT, and SFQ. With the limit of the length of this paper, we briefly review EZW as the typical scheme of the class of wavelet zerotree coding algorithms, and make our designs of zerotree construction based on the scheme without loosing generality. Hereby we quote several definitions like parent, child, descendent, root, zerotree, significance, dominant pass, subordinate pass, etc. from the [13] to describe the EZW coding scheme. As illustrated in Fig. 3, the input image is transformed and subsampled using the hierarchical DWT to obtain a collection of $3S + 1$ subband images, where S is the number of transform levels. As wavelet coefficients in the subband images have some correlations along the same orientation (horizontal, vertical, or diagonal), the dependencies can be well exploited by building a quadtree structure called "zerotrees," according to which any coefficient at a given band in Fig. 3 has four children coefficients at its lower-level subband (corresponding to the four-time larger subband image) in the same orientation. Suppose that the parent's position is (i, j) with i and j standing for the row and column number in its subband image, then its children's positions are $(2i, 2j), (2i, 2j + 1), (2i + 1, 2j)$, and $(2i + 1, 2j + 1)$ in their corresponding subband image. There are two types of passes performed in EZW coding. The dominant pass finds significant coefficients (greater than a given threshold), and its following subordinate pass refines the magnitudes of all significant coefficients found in the dominant pass. A ZTR symbol (meaning "zerotree root") is used for an insignificant coefficient (less than a given threshold) that has no significant descendents. An isolated zero symbol (named IZ) is used when a coefficient is insignificant but has some significant descendents. Because a child coefficient is probably also insignificant when its parent coefficient is insignificant, and thus, many insignificant coefficients can be represented as ZTRs, and furthermore a ZTRs descendents may be cut off from the final code stream, the use of ZTR and IZ symbols informs the locations of significant coefficients quite efficiently. Interested readers may refer to [13] for the details of zerotree coding algorithm.

B. The Application of Nonlinear I/O Data Space Transformations on Zerotree Construction

In the zerotree coding algorithm one first computes 2-D DWT and then constructs zerotree data structures on the results of wavelet transform. At each level of the 2-D separable DWT, the band is decomposed into L_r and H_r (where L_r is the result of low-pass row-wise filtering and H_r is the result of high-pass row-wise filtering); L_r is decomposed into LL and LH by low-pass and high-pass column-wise filters; H_r is decomposed into HL and HH by low-pass and high-pass

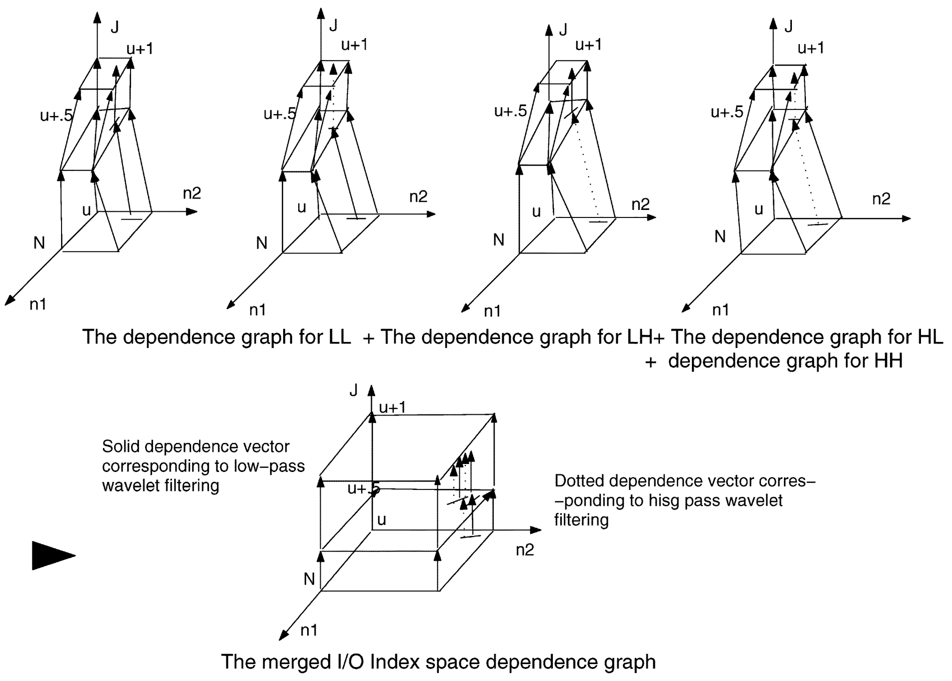

J
u+1
u+.5
u
N
n1
n2
The dependence graph for LL + The dependence graph for LH+ The dependence graph for HL + dependence graph for HH
Solid dependence vector corresponding to low–pass wavelet filtering
Dotted dependence vector corres––ponding to hisg pass wavelet filtering
The merged I/O Index space dependence graph

Fig. 4. Regularization of dependence graphs for zerotree construction.

column-wise filters. Subband LL is recursively decomposed in higher levels of transforms, as shown in Fig. 3. The subscript number of a subband in Fig. 3 refers to the wavelet transform level. For instance, LH_2 refers to the LH subband in the second-level wavelet transform. The dependence graphs in I/O data space and the nonlinear I/O data space transformations are illustrated in Fig. 4. According to Γ_5 in Section II, the nonlinear I/O data space transformation for 2-D DWT is rewritten as follows. When $j = u + 1/2$ (u is supposedly a positive integer), for $L_r : n_1 \mapsto 2^{\lfloor j \rfloor} n_1, n_2 \mapsto 2^{\lceil j \rceil} n_2, j \mapsto j$; for $H_r : n_1 \mapsto 2^{\lfloor j \rfloor} n_1, n_2 \mapsto 2^{\lceil j \rceil}(n_2 + (1/2)), j \mapsto j$; when j is an integer, for LL: $n_1 \mapsto 2^{\lfloor j \rfloor} n_1, n_2 \mapsto 2^{\lceil j \rceil} n_2, j \mapsto j$; for LH: $n_1 \mapsto 2^{\lfloor j \rfloor}(n_1 + (1/2)), n_2 \mapsto 2^{\lceil j \rceil} n_2, j \mapsto j$; for HL: $n_1 \mapsto 2^{\lfloor j \rfloor} n_1, n_2 \mapsto 2^{\lceil j \rceil}(n_2 + (1/2)), j \mapsto j$; for HH: $n_1 \mapsto 2^{\lfloor j \rfloor}(n_1 + (1/2)), n_2 \mapsto 2^{\lceil j \rceil}(n_2 + (1/2)), j \mapsto j$. Considering the zerotree data structures among the results of wavelet transforms, we can find that the nonlinear I/O data space transformation Γ_5 relocate the parent-children relationships and put each parent and its children together in terms of their coordinates of n_1 and n_2 in the I/O data space. For instance, let there be a parent data item belonging to subband HL_2 with the coordinates of $(n_1 = a, n_2 = b, j = 2)$ in I/O data space before the nonlinear I/O data space transformation. According to the zerotree coding algorithm, it has four children belonging to subband HL_1 with coordinates of $(n_1 = 2a, n_2 = 2b, j = 1), (2a, 2b+1, 1), (2a+1, 2b, 1)$, and $(2a+1, 2b+1, 1)$, respectively. After the nonlinear I/O data space transformation is applied, the parent's new coordinates are $(4a, 4b+2, 2)$, and the children's new coordinates are $(4a, 4b+1, 1), (4a+2, 4b+1, 1), (4a, 4b+3, 1)$ and $(4a+2, 4b+3, 1)$, respectively. In other words, the parent-children relationships are restricted in local domain if we get a projection of the dependence graphs in I/O data space along axis j.

To give a brief presentation, we suppose that there are two levels of wavelet transforms ($j \leq 2$ for the dependence graphs in I/O data space), and assume a row-major input format to scan a 2-D image and feed the image pixels to the system. Following the scheme of free schedule at the end of the last section, we can assign the computation corresponding to each layer of dependence vectors in Fig. 4, to a separate processor, and have all processors perform calculations in parallel when the input data are fed to the system in real time. The particular challenge in architectural designs of wavelet zerotree coding algorithms stems from locating children given any parent data item. To avoid using RAMs for data rearrangement when constructing zerotrees and generating symbols as designated in the zerotree coding algorithm, we adjust the scheme of free schedule so that the calculation of any parent data and the calculation of its children are scheduled to be on the same time. In other words, we reorder the 2-D DWT computation so that the result of the DWT computation (as the outputs of parallel processors) itself follows the zerotree structures. This has been made possible by nonlinear I/O data space transformations proposed in the paper. When the input image pixels are fed to the system in a row-major order, the wavelet-adjacent fields on plane $j = 0$ are row-wise scanned, and the calculations corresponding to the dependence vectors between planes $j = 0$ and $j = 1/2$ are ready to be performed in the same row-wise order. Due to I/O data space transformation, the calculation results on each plane (orthogonal to axis j) above $j = 0$ can also be produced in row-wise major by parallel processors in real time. Since any parent data and its children on zerotree structure are put together via Γ_5 in terms of their coordinates of n_1 and n_2 in I/O data space, we can adjust the schedules of parallel processors only a little so that the children are calculated by a processor when their parent is calculated by another processor simultaneously. This adjustment is

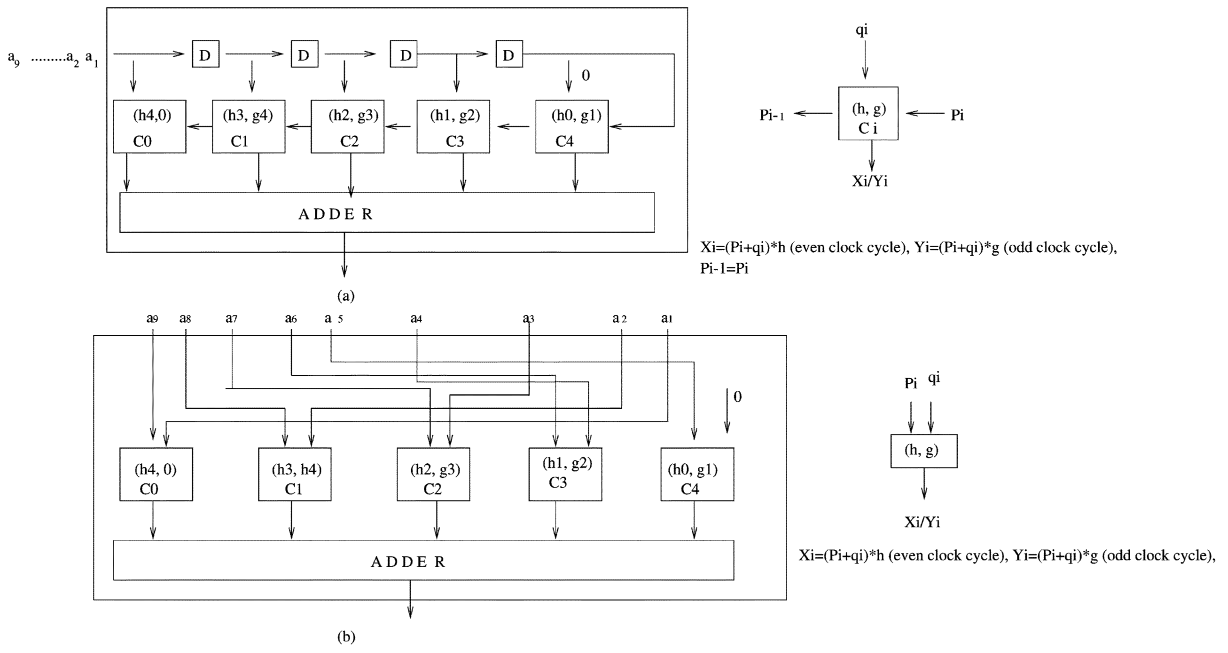


Fig. 5. Systolic and parallel wavelet filters integrating low-pass and high-pass filtering. (a) Systolic filter. (b) Parallel filter.

accomplished by using a small *transpose unit* in which only systolic data flow is allowed, and meanwhile a large off-chip RAM is avoided. The architectures are proposed in the following.

C. The Architecture for Rearranging 2-Level 2-D DWT

We assume the width of wavelet filters is L and the size of 2-D input data is $N \times N$. For typical applications in practice, let $L = 9$ and $N = 512$ in the introduction of this subsection. We introduce a simple structure of wavelet filters whose architecture is detailed in [22]. Its structure is used in our design as a module of processing unit (PU) that computes wavelet filtering and decimation. As illustrated in Fig. 5 for a 9-point wavelet filter, it is made up of four registers, five multipliers, and six adders. It rearranges the calculation of wavelet filtering such that the filter is cut to half taps based on the symmetry between the negative and positive wavelet filter coefficients. a, X, and Y are the input sequence, low- and high-pass filtering output sequence, respectively. While a datum of input sequence a is fed and shifted into the PU every clock cycle, a datum of X is calculated every even clock cycle and a datum of Y is calculated every odd clock cycle. Such calculations are possible because of the wavelet dyadic downsampling. The connections between computation units (multipliers or adders) are restricted local. The PU in Fig. 5(a) is extended to a parallel format as illustrated in Fig. 5(b) where if a number of data from sequence $a_{k+8}, a_{k+7}, \ldots, a_k$ are fed to the PU in parallel at a certain clock cycle, then $a_{k+9}, a_{k+8}, \ldots, a_{k+1}$ are fed at the next cycle. The calculations of X and Y are the same as in Fig. 5(a). The PU actually takes a wavelet-adjacent field as the input.

We propose a module called transpose unit (TU) that can partially transpose a matrix on the fly. When a matrix is input to the TU in a row-wise indexing way with one element per clock cycle, the TU gives out the elements in a partially column-wise indexing way as explained in the following. The structure of

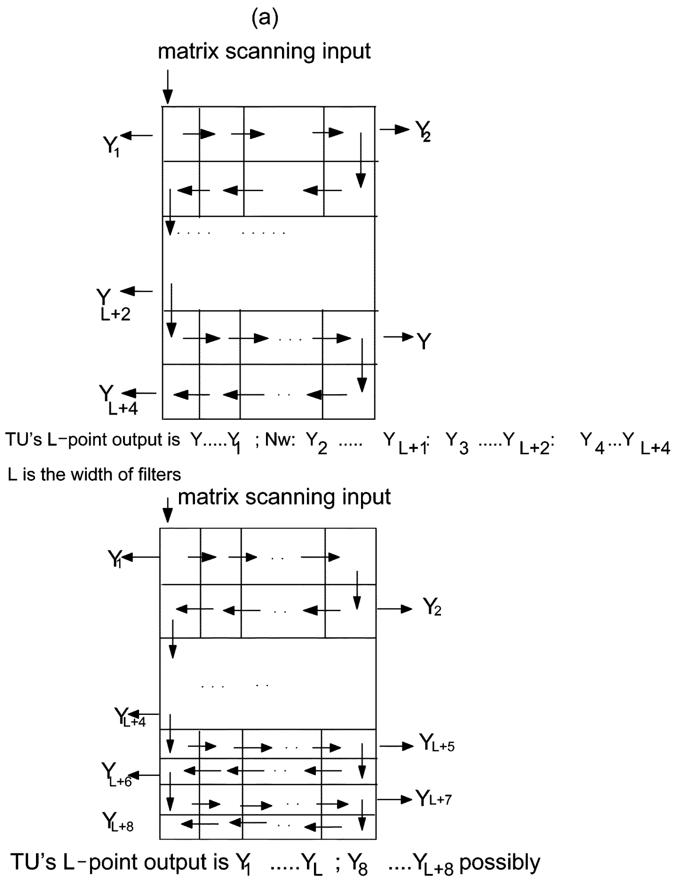
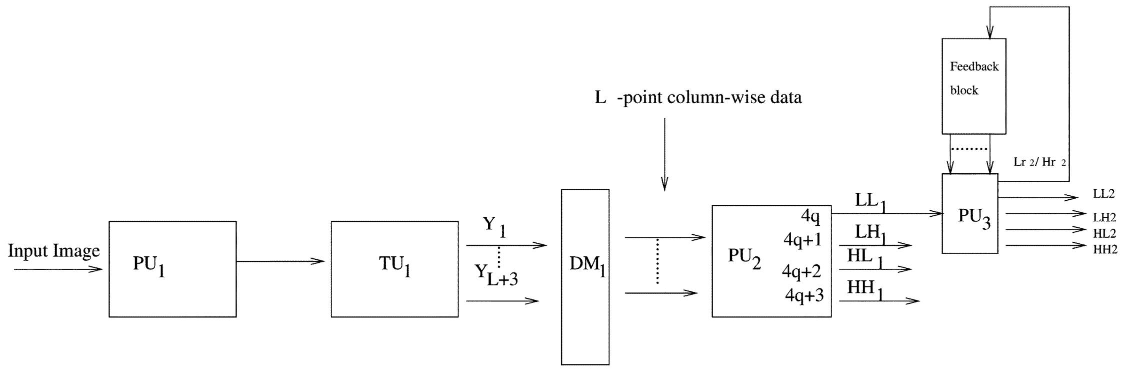

(a)
matrix scanning input
Y1
Y2
YL+2
Y
YL+4
TU's L-point output is Y.....Y1 ; Nw: Y2 ..... YL+1; Y3 .....YL+2; Y4...YL+4
L is the width of filters
matrix scanning input
Y1
Y2
YL+4
YL+5
YL+6
YL+7
YL+8
TU's L-point output is Y1 .....YL ; Y8 ....YL+8 possibly

Fig. 6. Architecture of Transpose Unit (TU).

TU is illustrated in Fig. 6. It is made up of $(L + 4)$ or 13 concatenated modules of first in first out (FIFOs), with each FIFO having N or 512 cells. The top FIFO takes as its input the row-wise indexed matrix with one element per clock cycle. Because the distance between the outputs of all FIFOs Y_i and



1) DM1 is a multiplexer that selects L-point data from L+3 outputs of TU1.

2) PU2 is a parallel filter as in Figure 2(b) and has four output ports active at different time.

3) PU3 is the hybrid version of PU that can take either sequential or parallel inputs.

4) Feedback block consists of 2 separate TUs and multiplexers to select Lr2 / Hr2 into respective TU and to select outputs from 2 TUs into PU3.

Fig. 7. Architecture for zerotree construction.

```
real-time Alg. DWT amenable to EZW
{
  for i=0 to N-1    / * row */
    for j=0  to N-1  / * column */
      Do DWT(i,j)
}

DWT(i,j)     / * q,s are any non-negative integers */
{
                      what PU1 does                  what PU2 does                          what PU3 does
 if  i=4q
   if j=even          $Lr_1 \xleftarrow{r} x$        $LL_1 \xleftarrow{c} Lr_1$
   if j=odd           $Lr_1 \xleftarrow{r} x$        $LL_1^1 \xleftarrow{c} Lr_1^1$
   if j=4s                                                                                  $Lr_2 \xleftarrow{r} LL_1$
   if j=4s+1                                                                                $Hr_2 \xleftarrow{r} LL_1$
   if j=4s+2                                                                                $Lr_2^1 \xleftarrow{r} LL_1^1$
   if j=4s+3                                                                                $Hr_2^1 \xleftarrow{r} LL_1^1$
 if i=4q+1
   if j=even          $Lr_1 \xleftarrow{r} x$        $LH_1 \xleftarrow{c} Lr_1^+$
   if j=odd           $Lr_1 \xleftarrow{r} x$        $LH_1^1 \xleftarrow{c} Lr_1^{1+}$
   if j=4s                                                                                  $LL_2 \xleftarrow{c} Lr_2$
   if j=4s+2                                                                                $LH_2 \xleftarrow{c} Lr_2^+$
 if i=4q+2
   if j=even          $Lr_1 \xleftarrow{r} x$        $HL_1 \xleftarrow{c} Hr_1$
   if j=odd           $Hr_1 \xleftarrow{r} x$        $HL_1^1 \xleftarrow{c} Hr_1^1$
   if j=4s+2                                                                                $HL_2 \xleftarrow{c} Hr_2$
 if i=4q+3
   if j=even          $Lr_1 \xleftarrow{r} x$        $HH_1 \xleftarrow{c} Hr_1^+$
   if j=odd           $Hr_1 \xleftarrow{r} x$        $HH_1^1 \xleftarrow{c} Hr_1^{1+}$
   if j=4s+2                                                                                $HH_2 \xleftarrow{c} Hr_2^+$
```

Fig. 8. Operation timing for 2-level zerotree construction.

$Y_{i+1}(0 < i < (L+3))$ is always *N*, the TUs 13 outputs belong to the same column in the input matrix. The FIFOs transfer the input data step by step in clock cycles, and meanwhile provide a new group of column-wise data (i.e., a new column-wise wavelet-adjacent field) to the next wavelet filter in each clock cycle.

Keeping in mind the dyadic downsamplings in both row-major and column-major filtering of 2-D DWT. In the following, we present our design that rearranges the computation of the DWT so that a parent and its children are calculated at the same time. The structure for the non-RAM-based zerotree construction is proposed in Fig. 7, and the timing of operations is regulated in Fig. 8.

The coefficients generated in the first decomposition level can be separated into groups each of which contains four coefficients having the same parent (which is generated in the next decomposition level). Now we have the restriction that these four sibling coefficients be calculated together for the purpose of calculating children and parent simultaneously. More exactly, we calculate the four siblings consecutively at a rate

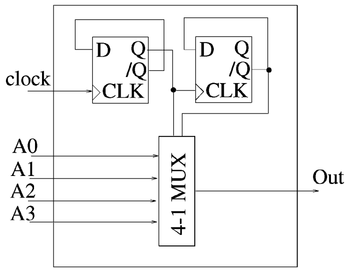


Fig. 9. Example of local-controlled 4-1 multiplexer.

with one coefficient per clock cycle, and at the same time generate their parent coefficient at a four-time less rate via another output port. Due to the row-major dyadic downsampling, the row-major high- low-pass filtering is alternatively executed by PU_1 in Fig. 7 point by point in each row. Based on similar column-major dyadic downsampling, PU_2 takes turns to execute column-major high-pass/low-pass filtering and selects appropriate inputs from TU_1 to generate four sibling coefficients consecutively. The order to calculate the siblings is as A, then B, C, and D in the example of four siblings illustrated in Fig. 3(a). After PU_2's calculation of point A by taking $\mathrm{Y}_1, \ldots, \mathrm{Y}_L$ as inputs, PU_2 has to calculate B by taking $\mathrm{Y}_3, \ldots, \mathrm{Y}_{L+2}$ as inputs, then PU_2 comes back to take $\mathrm{Y}_1, \ldots, \mathrm{Y}_L$ to calculate C, then PU_2 takes $\mathrm{Y}_3, \ldots, \mathrm{Y}_{L+2}$ again to calculate D. This is for the case that A–D are in the column-major low-pass subband (HL in Fig. 3). If they are in the column-major high-pass subband (LH or HH in Fig. 3), PU_2 will alternate using $\mathrm{Y}_2, \ldots, \mathrm{Y}_{L+1}$ and $\mathrm{Y}_4, \ldots, \mathrm{Y}_{L+3}$ as inputs and take turns on the calculations in similar ways. Note that PUs consume one clock cycle for each calculation and the TU cells consume one clock cycle to transfer a datum so that the above calculations and the corresponding data transfers are performed step by step. In a similar way, PU_3 follows the timing in Fig. 8, and takes turns generating the parents as the outputs of the second level of DWT. In summary, any four siblings are always generated in turns by PU_2 and their parent is calculated by PU_3 at the same time. The control signal for the switch DM is internal and simple according to the operation timing in Fig. 8. There is neither external nor complex control for any device in Fig. 7. Only the clock signal is global to synchronize the system. We call this scheme of internal control as "self-controlled" device. Fig. 9 has demonstrated an example that a multiplexer selects data based on internal control signals which are generated from the input clock periodically. It is the rearranged periodical operations for zerotree construction derived from our novel nonlinear I/O data space transformations in Section II that make such designs of "self-controlled" devices possible.

D. Extension to Arbitrary Number of Levels in DWT and Zerotree Construction

In this section, we extend our design to general cases where any levels of wavelet decompositions are possible. All coefficients in the first level and in the intermediate levels which correspond to the same ancestor in the last level decomposition have to be calculated together to satisfy the restriction that any parent and its children be calculated simultaneously. As in the former subsection, by the word "together" we mean successively calculating the children at a four-time higher frequency than their parent and simultaneously outputting children and the parent via two ports, respectively.

Because the input image is fed into the system in the same way as before, the first level row-major high-pass/low-pass filtering is still performed in PU_1 alternatively as designated in Fig. 8. Regarding the first level column-major high-pass/low-pass filtering performed in PU_2, we note that there are 4^{m-1} coefficients in the first-level decomposition corresponding to the same ancestor in the last level (level m) decomposition. To satisfy the restriction of generating parent and children simultaneously, it is required that these 4^{m-1} "kindred" coefficients be calculated together. Meanwhile, these coefficients' parents in the intermediate levels of decomposition should be calculated together as well. Note that these 4^{m-1} coefficients are located in 2^{m-1} adjacent rows and 2^{m-1} adjacent columns in their subband. PU_2 should alternatively select appropriate inputs among 2^{m-1} different groups of parallel column-major data from TU_1, and perform column-major filtering to generate the 4^{m-1} kindred coefficients in turn, where the coefficients calculated with the same group of input belong to the same row (see the following paragraph, for example). Accordingly, TU_1 is an extended version in Fig. 6 and is supposed to have output ports $\mathrm{Y}_1, \ldots, \mathrm{Y}_{L+M}$ with M equal to 2^m, so TU_1 has $(L+M) \times N$ cells with a structure similar to Fig. 6.

For instance, there are 16 "kindred" coefficients in HL_1 subband illustrated in Fig. 3(b) to be calculated successively in the first level of the three-level DWT. The order to calculate these coefficients is from A1 to A2, A3, A4, B1, B2, B3, B4, C1, C2, C3, C4, D1, D2, D3, and at last D4. This calculation order is according to the requirement that siblings be calculated successively, e.g., the siblings A1, A2, A3, and A4 should be calculated as a group (meanwhile their parent A is calculated in PU_3). Four parallel column-major data are selected among TU_1's output ports $\mathrm{Y}_1, \ldots, \mathrm{Y}_{L+8}$ as: $\mathrm{Y}_1, \ldots, \mathrm{Y}_L$ (named $\mathbf{E}_1$ for brevity); $\mathrm{Y}_3, \ldots, \mathrm{Y}_{L+2}$ (named $\mathbf{E}_2$); $\mathrm{Y}_5, \ldots, \mathrm{Y}_{L+4}$ (named $\mathbf{E}_3$); and $\mathrm{Y}_7, \ldots, \mathrm{Y}_{L+6}$ (named $\mathbf{E}_4$). Based on the systolic data transfer in TU_1 and column-major dyadic subsamplings in $\mathrm{PU}_2, \mathrm{PU}_2$ first takes $\mathbf{E}_1$ as input to calculate A1, then uses $\mathbf{E}_2$ for calculating A2 in the next clock cycle; after that, it comes back to take $\mathbf{E}_1$ for A3, then $\mathbf{E}_2$ for A4. PU_2's following operations are: $\mathbf{E}_3$ for B1, $\mathbf{E}_4$ for B2, $\mathbf{E}_3$ for B3, $\mathbf{E}_4$ for B4, then $\mathbf{E}_1$ for C1, $\mathbf{E}_2$ for C2, etc.

Now we analyze the operations in PU_3. PU_3 carries out the rest of the computations in DWT. The second level decomposition is achieved as follows. In the first quarter of the period when 2^m rows of input image are fed to the system, PU_3 gets its inputs, i.e., the coefficients in LL_1 subband [see Fig. 3(d)] from TU_1, and alternatively performs the second level low-pass/high-pass row-major convolution. The calculated results, or the coefficients in Lr_2 and Hr_2 are stored in two TUs in PU_3's feedback block. In the second quarter, the Lr_2 coefficients are fed back to PU_3 to be column-major filtered to get the results in LL_2 and LH_2. In the third and fourth quarter, the Hr_2 points are fed back to PU_3 to be used to calculate out HL_2, and HH_2, respectively. There are some idling intervals during PU_3's performing the second

```
real-time Alg. for 3-level DWT amenable to EZW
{
  for i=0 to N-1    / * row */
    for j=0  to N-1  / * column */
      Do 3-level-DWT(i,j)
}

3-level-DWT(i,j)     / * q,s are any non-negative integers */
{
```

	what PU_1 does	what PU_2 does	what PU_3 does depending on j's value	
if i=8q				
if j=even	$Lr_1 \xleftarrow{r} x$	$LL_1 \xleftarrow{c} Lr_1$	$Lr_2 \xleftarrow{r} LL_1$ (j=4s)	$Lr_2^1 \xleftarrow{r} LL_1^1$ (j=4s+2)
if j=odd	$Lr_1 \xleftarrow{r} x$	$LL_1^1 \xleftarrow{c} Lr_1^1$	$Hr_2 \xleftarrow{r} LL_1$ (j=4s+1)	$Hr_2^1 \xleftarrow{r} LL_1^1$ (j=4s+3)
if i=8q+1				
if j=even	$Lr_1 \xleftarrow{r} x$	$LL_1^2 \xleftarrow{c} Lr_1^2$	$Lr_1^2 \xleftarrow{r} LL_1^2$ (j=4s)	$Lr_2^3 \xleftarrow{r} LL_1^3$ (j=4s+2)
if j=odd	$Hr_1 \xleftarrow{r} x$	$LL_1^3 \xleftarrow{c} Lr_1^3$	$Hr_2^2 \xleftarrow{r} LL_1^2$ (j=4s+1)	$Hr_2^3 \xleftarrow{r} LL_1^3$ (j=4s+3)
if i=8q+2				
if j=even	$Lr_1 \xleftarrow{r} x$	$LH_1 \xleftarrow{c} Lr_1^+$	$LL_2 \xleftarrow{c} Lr_2$ (j=4s)	$LL_2^1 \xleftarrow{c} Lr_2^1$ (j=4s+2)
if j=odd	$Hr_1 \xleftarrow{r} x$	$LH_1^1 \xleftarrow{c} Lr_1^{1+}$	$Lr_3 \xleftarrow{r} LL_2$ (j=8s+1)	$Lr_3^1 \xleftarrow{r} LL_2^1$ (j=8s+3)
			$Hr_3 \xleftarrow{r} LL_2$ (j=8s+5)	$Hr_3^1 \xleftarrow{r} LL_2^1$ (j=8s+7)
if i=8q+3				
if j=even	$Lr_1 \xleftarrow{r} x$	$LH_1^2 \xleftarrow{c} Lr_1^{2+}$	$LH_2 \xleftarrow{c} Lr_2^+$ (j=4s)	$LH_2^1 \xleftarrow{c} Lr_2^{1+}$(j=4s+2)
if j=odd	$Hr_1 \xleftarrow{r} x$	$LH_1^3 \xleftarrow{c} Lr_1^{3+}$	$LL_3 \xleftarrow{c} Lr_3$ (j=8s+1)	$LH_3 \xleftarrow{c} Lr_3^+$ (j=8s+5)
if i=8q+4				
if j=even	$Lr_1 \xleftarrow{r} x$	$HL_1 \xleftarrow{c} Hr_1$	$HL_2 \xleftarrow{c} Hr_2$ (j=4s)	
if j=odd	$Hr_1 \xleftarrow{r} x$	$HL_1^1 \xleftarrow{c} Hr_1^1$		
if i=8q+5				
if j=even	$Lr_1 \xleftarrow{r} x$	$HL_1^2 \xleftarrow{c} Hr_1^2$	$HL_2^1 \xleftarrow{c} Hr_2^1$ (j=4s)	
if j=odd	$Hr_1 \xleftarrow{r} x$	$HL_1^3 \xleftarrow{c} Hr_1^3$	$HL_3 \xleftarrow{c} Hr_3$ (j=8s+5)	
if i=8q+6				
if j=even	$Lr_1 \xleftarrow{r} x$	$HH_1 \xleftarrow{c} Hr_1^+$	$HH_2 \xleftarrow{c} Hr_2^+$ (j=4s)	
if j=odd	$Hr_1 \xleftarrow{r} x$	$HH_1^1 \xleftarrow{c} Hr_1^{1+}$		
if i=8q+7				
if j=even	$Lr_1 \xleftarrow{r} x$	$HH_1^2 \xleftarrow{c} Hr_1^{2+}$	$HH_2^1 \xleftarrow{c} Hr_2^{1+}$ (j=4s)	
if j=odd	$Hr_1 \xleftarrow{r} x$	$HH_1^3 \xleftarrow{c} Hr_1^{3+}$	$HH_3 \xleftarrow{c} Hr_3^+$ (j=8s+5)	

```
}   / * the notation in Fig(c),(d) and Fig 5 applies to this algorithm */
```

Fig. 10. Operation timing for 3-level zerotree construction.

level transform due to less computation in the second level of DWT. Thus, we can insert the computation of further levels of decomposition into those available intervals. For example, once an LL_2 point is generated in the second quarter, it will be fed back to PU_3 via the feedback block to be row-major filtered in available intervals. Then the generated Lr_3 and Hr_3 coefficients are also stored in TUs in PU_3's feedback block. In the next available intervals Lr_2 and Hr_2 are fed to PU_3 for column-major convolution to calculate LH_3, HL_3, and HH_3 coefficients. Due to the exponentially decreasing number of parents, PU_3 can similarly proceed to more levels of decomposition in the intervals between lower-level calculations. Fig. 10 has given an illustration of the operation timing for 3-level zerotree construction.

IV. Experimental Results and Performance Analysis

As part of our study, various experimental results have been obtained regarding the implementation of architectures proposed in this paper. We have achieved the gate-level synthesis of the architecture with the Cadence Verilog HDL simulation package. The simulation results have shown advantages in our designs in contrast with those RAM-based architectures for implementing the algorithm of zerotree construction. We also compare the gate-level synthesis of our architecture for zerotree construction with the results of the architecture for basic wavelet transforms (without zerotree construction). The experimental results have shown that the execution time and structure cost are comparable although the algorithm of zerotree construction has a much higher complexity than the algorithm of basic wavelet transform.

In simulation of the module processing unit for PU_1, PU_2, and PU_3, we chose to implement the computation of Harr-basis [17] integral wavelet filtering using shifters and adders based on calculations of integers. The computation of other wavelet-basis filterings can be implemented similarly when using general structures of floating-point adders and multipliers. All devices are locally controlled and the hardware connections are localized. The size of input image is $N \times N$, with each pixel represented in 8 bit. The input square image is fed into the system with one pixel per clock cycle.

To evaluate the performance, we also present a gate-level synthesis of a typical RAM-based design for zerotree construction [18] as a contrast with the architectures proposed in this

TABLE I
GATE-LEVEL IMPLEMENTATION OF NON-RAM-BASED ZEROTREE CONSTRUCTION

The width of input square image	The number of gates in the system	Execution clock cycles
64	67232	4576
128	125712	17344
256	248592	67456
512	491832	265984
1024	934480	1124380

TABLE II
GATE-LEVEL IMPLEMENTATION OF RAM-BASED ZEROTREE CONSTRUCTION

The width of input square image	The number of gates in the system	Execution clock cycles
64	360448B RAM cells + 35488	10304
128	131072B RAM cells + 68256	41088
256	524288B RAM cells + 133792	164096
512	2097152B RAM cells + 264864	655872
1024	8388608B RAM cells + 520462	2619844

TABLE III
GATE-LEVEL IMPLEMENTATION OF BASIC DWT

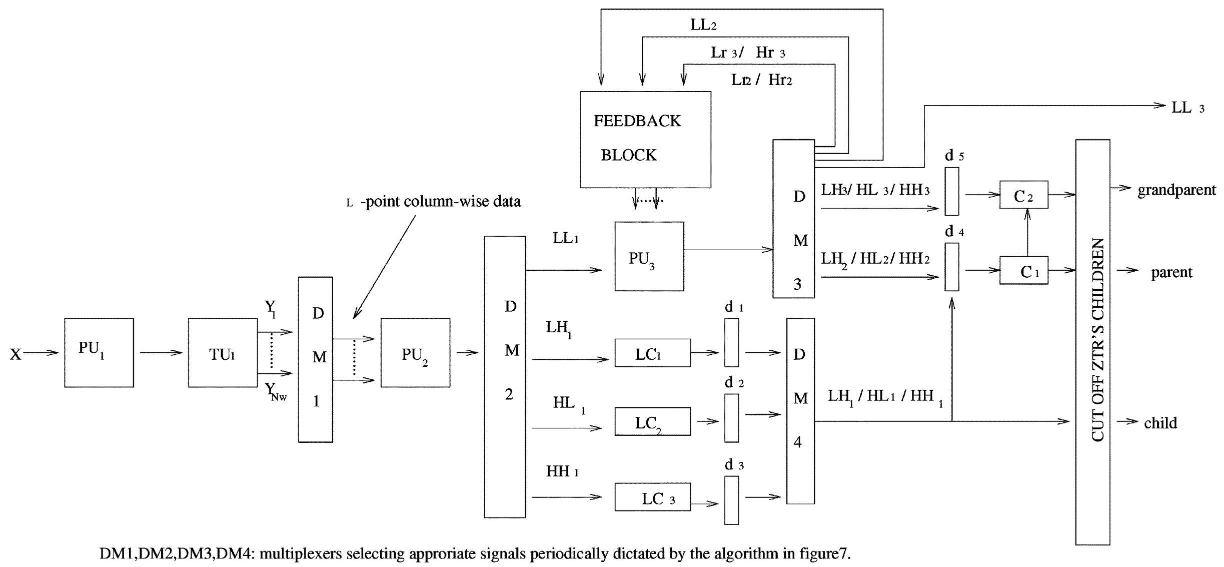
The width of input square image	The number of gates in the system	Execution clock cycles
64	26736	4288
128	51312	16768
256	100464	66304
512	198768	263680
1024	410992	1087642

paper. In this design, the image is stored in an off-chip RAM and accessed by an ASIC processing structure for the wavelet transforms and zerotree construction. The processing structure performs the calculation of hierarchical wavelet transforms on the image, and constructs the zerotree data structure based on the results of wavelet transforms by locating the parent-children relationships with calculation of address pointers.

In literature, researchers have proposed many RAM-based architectural designs for zerotree coding ([18]–[20]). All of these designs use off-chip RAMs for data preprocessing and/or calculation since the sizes of input images are much bigger than the size of what on-chip caches can hold. There exist bottlenecks in nature limiting the processing rate in these designs—the frequent data transfers between the processor and the RAM, the limitation on the off-chip RAM bus bandwidth and on the size of the data bus. The general difference between access rates from off-chip RAMs and access rates from on-chip storage can be referenced from [32], [33]. Generally, access from off-chip RAMs is at least tens of times slower than access from on-chip FIFOs. The comparison in Tables I and II between our design of zerotree construction and a typical RAM-based implementation shows advantages of the non-RAM-based design of zerotree construction against the RAM-based implementation of zerotree construction. The numbers of gates used in the implementation and the execution time in clock cycles are shown in Tables I– III. The clock rate in the circuit in Fig. 7 is limited by the longest latch to latch delay which is independent of the width of input square image. In the structure, the longest delay path is located within PU_3's feedback block which contains two multiplexers, and has 14 gate delay time units. The clock rate in Table II is limited by the rate of off-chip RAM buses.

In the following, we present the complexity analysis of our architectural designs for arbitrary level of wavelet transforms, generic wavelet filters and data precisions. Since *L* (the width of wavelet filters) is far less than *N* (the width or length of input image) and the size of boundary effect of wavelet transforms is only dependent on *L*, in the analysis we ignore the boundary effect to simplify the expressions. The area of the architecture in Fig. 7 is dominated by PUs and TUs since all connections are restricted in locality. A PU contains pL multiplier and accumulator cell MACs, where p is the number of precision bits of data, thus, three PUs contain $3pL$ MACs, and the area for these MACs is $O(pL)$. Because a TU is necessary for the column-major filtering in every level tran sform, and the number of cells in TU at the ith level transform is $N(L+2^i)$, the area for TUs is $N(L+2^{m+1}-1)$, where m is the number of levels in DWT. Assuming 2^{m+1} is a constant *C*, the whole area of the architecture for m level DWT is $\mathbf{A} = O(\mathrm{pNL})$. Noting that a new datum is calculated and outputted as a pixel arrives every cycle, and the output size is not more than input size if the boundary effect is disregarded, we have the system's latency (execution time) $\mathbf{T}$ as N^2 clock cycles. Thus, the product of $\mathbf{A}$ and $\mathbf{T}$ for the system is $\mathrm{O(pN^3L)}$, where pN^2 is the input size of the algorithm. The hardware utilization of PU_1 and PU_2 is 100%. The utilization of PU_3 for *m* levels of DWT can be figured out by the comparison between the computation tasks of PU_3 and PU_1 or PU_2. PU_1 and PU_2, respectively, processes a half amount of the computation for the first level of 2-D DWT; on the other hand, PU_3 with the same hardware structure takes all computation tasks for the other levels of 2-D DWT. Considering that the computation amount for every level of 2-D DWT is four times less than that of the previous level, we have the utilization of PU_3 equal to $(\sum_{i=1}^{m-1}(T/4^i))/(T/2)$, where T is the computation amount for the first level of 2-D DWT. Thus, the total hardware utilization of three processors is $(2+2\sum_{i=1}^{m-1}/1/4^i)/3$, which is around 90% for the 2-D DWT of more than 4 levels of transform. The utilization of TUs is 100%. The result of performance analysis has been summarized in Table IV.

The proposed architectures can be extended to the computation of more complex algorithms. For instance, with the coefficients grouped in zerotrees, it is easy to generate the zerotree-coding symbols (as the third step of EZW algorithm) based on the architecture in Fig. 7. Put detectors for insignificant coefficients and registers at output ports and connect the registers at those output ports corresponding to the relation of parent and children. A symbol of IZ is generated and kept in the pertinent register if one coefficient is detected to be insignificant. According to the scheme of EZW, this register is then reset to the symbol of ZTR either if all of its corresponding children are ZTRs or if the children are leaves of the zerotree and IZs. Then the registers of the ZTRs children are set as *bubbles* so that nothing is output to the generated symbol stream. Also the symbols POS and NEG [13] can be generated by trivially detecting the signs of the significant coefficients. Thus, the zerotree-coding symbols can be streamed out via system output



DM1,DM2,DM3,DM4: multiplexers selecting approriate signals periodically dictated by the algorithm in figure7.
LC1 , LC2 , LC3 : linear storage (FIFO) containning N celts ------2 rows of LH / HL / HH .1
d 1, d2 , d 3 , d 4 , d5 : zero-detector thresholding the coefficients
C 1 : ZTR-detector setting the parent ZTR if its children are zeros.
C 2 : ZTR-detector setting the grandparent ZTR if its children are ZTR.
FEEDBACK BLOCK: store data and feed them in PU3
TU : the Figure 6(b) version

Fig. 11. Extension of the zerotree-building architecture for symbol generation in real time.

TABLE IV
PERFORMANCE ANALYSIS FOR THE ZEROTREE CONSTRUCTION ARCHITECTURE

Device	Area	Execution Time	Hardware Utilization
PU_1	$O(pL)$	N^2	100%
PU_2	$O(pL)$	N^2	100%
PU_3	$O(pL)$	N^2	$2\sum_{i=1}^{m-1}\frac{1}{4^i}$
ALL PUs	$O(pL)$	N^2	$(2+2\sum_{i=1}^{m-1}\frac{1}{4^i})/3$
TU(FIFO)	$O(pNL)$	N^2	100%

ports in real time when the input image is fed to the system. The extension of the architecture (Fig. 7) is shown in Fig. 11.

V. CONCLUSION

This paper has proposed a methodology for non-RAM-based architectural designs of wavelet algorithms based on novel nonlinear I/O data space transformations. Exploiting common features of computation locality and multirate signal processing within general wavelet-based algorithms, this paper proposes a series of novel nonlinear transformations in I/O data space analysis and obtains regularized and/or merged structures of dependence graphs for any wavelet-based algorithms. Such nonlinear transformations for newly-modeled data dependence graphs lead to non-RAM-based architectures for hardware implementations of general wavelet-based algorithms.

The series of nonlinear I/O data space transformations are proposed as a theoretical basis of architectural designs for generalized wavelet-based algorithms, but it is infeasible to introduce designs for all complex wavelet-based algorithms due to the space limit in this paper. We use the zerotree construction algorithm as the representative and propose a non-RAM-based design in which the input image is recursively decomposed by DWT and the zerotrees are constructed simultaneously. In contrast with the reported architectures ([18]–[20]) for wavelet zerotree constructions that use large off-chip RAMs in building zerotrees and employ either memory address pointers or data re-arrangement, our architectures only need much smaller on-chip FIFOs leading to the elimination of off-chip communications and the increase of processing rates.

The philosophy underlying our proposed design is a full exploitation of the locality of the computation. The computation of wavelet-based zerotree coding is strongly featured by the computation locality in that the calculations of coefficients on a certain zerotree only depend on the same local subarea of the 2-D inputs. This desirable feature has been fully exploited in this paper by calculating children and their parent simultaneously in the rearranged DWT, so that the necessary storage for intermediately calculated data is reduced to a great extent—some items of intermediate data need not be held for future calculations if the coefficients on the corresponding zerotrees are scheduled to be calculated together and earlier. This primary approach based on our novel nonlinear I/O data space transformations is not only used to derive designs for the algorithm of zerotree construction, but for many other general complex wavelet-based digital systems. As such, interested readers are referred to [28]–[31] for our detailed architectural designs of MWT, FWT, etc.

REFERENCES

[1] M. Cotronei, L. B. Montefusco, and L. Puccio, "Multiwavelet analysis and signal processing," *IEEE Trans. Circuits Syst. II*, vol. 45, pp. 970–987, Aug. 1998.
[2] F. G. Meyer, A. Z. Averbuch, and J. O. Stromberg, "Fast adaptive wavelet packet image compression," *IEEE Trans. Image Processing*, vol. 9, pp. 792–800, May 2000.
[3] H. Sava, M. Fleury, A. C. Downton, and A. F. Clark, "Parallel pipeline implementation of wavelet transforms," *IEE Proc. Vision, Image, and Signal Processing*, vol. 144, pp. 355–360, Dec. 1997.
[4] K. K. Parhi and T. Nishitani, "VLSI architectures for discrete wavelet transforms," *IEEE Trans. VLSI Syst.*, vol. 1, pp. 191–202, Jun. 1993.

[5] A. Grzeszczak, M. K. Mandal, and S. Panchanathan, "VLSI implementation of discrete wavelet transform," *IEEE Trans. VLSI Syst.*, vol. 4, pp. 421–433, Dec. 1996.
[6] S.-K. Pack and L.-S. Kim, "2D DWT VLSI architecture for wavelet image processing," *Electron. Lett.*, vol. 34, pp. 537–538, Mar. 1998.
[7] G. Knowles, "VLSI architecture for the discrete wavelet transform," *Electron. Lett.*, vol. 26, pp. 1184–1185, Jul. 1990.
[8] T. C. Denk and K. K. Parhi, "VLSI architectures for lattice structure based orthonormal discrete wavelet transforms," *IEEE Trans. Circuits Syst. II*, vol. 44, pp. 129–132, Feb. 1997.
[9] C. Chakrabarti and M. Vishwanath, "Efficient realizations of the discrete and continuous wavelet transforms: From single chip implementations to mappings on SIMD array computers," *IEEE Trans. Signal Processing*, vol. 43, pp. 759–771, Mar. 1995.
[10] M. Cotronei, D. Lazzaro, L. B. Montefusco, and L. Puccio, "Image compression through embedded multiwavelet transform coding," *IEEE Trans. Image Processing*, vol. 9, pp. 184–189, Feb. 2000.
[11] G. Lin and Z.-M. Liu, "The application of multiwavelet transform to image coding," *IEEE Trans. Image Processing*, vol. 9, pp. 270–273, Feb. 2000.
[12] F. Kurth and M. Clausen, "Filter bank tree and M-band wavelet packet algorithms in audio signal processing," *IEEE Trans. Signal Processing*, vol. 47, pp. 549–554, Feb. 1999.
[13] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Processing*, vol. 41, pp. 3445–3462, 1993.
[14] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 243–250, Jun. 1996.
[15] Z. Xiong, K. Ramchandran, and M. T. Orchard, "Wavelet packet image coding using space-frequency quantization," *IEEE Trans. Image Processing*, vol. 7, pp. 892–898, Jun. 1998.
[16] ——, "Space-frequency quantization for wavelet image coding," *IEEE Trans. Image Processing*, vol. 6, pp. 677–693, May 1997.
[17] M. Vetterli and J. Kovacevic, *Wavelets and Subband Coding*. Englewood Cliffs, NJ: Prentice-Hall, 1995.
[18] L.-M. Ang, H. N. Cheung, and K. Eshraghian, "VLSI architecture for significance map coding of embedded zerotree wavelet coefficients," in *Proc. 1998 IEEE Asia-Pacific Conf. Circuits and Systems*, 1998, pp. 627–630.
[19] J. Bae and V. K. Prasanna, "A fast and area-efficient VLSI architecture for embedded image coding," in *Proc. Int. Conf. Image Processing*, vol. 3, 1995, pp. 452–455.
[20] J. M. Shapiro, "A fast technique for identifying zerotrees in the EZW algorithm," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, vol. 3, 1996, pp. 1455–1458.
[21] J. Vega-Pineda, M. A. Suriano, V. M. Villalva, S. D. Cabrera, and Y.-C. Chang, "A VLSI array processor with embedded scalability for hierarchical image compression," in *Proc. IEEE Int. Symp. Circuits and Systems*, vol. 4, 1996, pp. 168–171.
[22] T. Acharya and P.-Y. Chen, "VLSI implementation of a DWT architecture," in *Proc. 1998 IEEE Int. Symp. Circuits and Systems*, vol. 2, 1998, pp. 272–275.
[23] J. Fridman and E. S. Manolakos, "Discrete wavelet transform: Data dependence analysis and synthesis of distributed memory and control array architectures," *IEEE Trans. Signal Processing*, vol. 45, pp. 1291–1308, May 1997.
[24] J. N. Patel, A. A. Khokhar, and L. H. Jamieson, "On the scalability of 2-D wavelet transform algorithms on fine-grained parallel machines," in *Proc. 1996 Int. Conf. Parallel Processing*, vol. 2, pp. 24–28.
[25] K.-W. Cheung, C.-H. Cheung, and L.-M. Po, "A novel multiwavelet-based integer transform for lossless image coding," in *Proc. 1999 Int. Conf. Image Processing*, vol. 1, pp. 444–447.
[26] G. Lafruit, F. Catthoor, J. P. H. Cornelis, and H. J. De Man, "An efficient VLSI architecture for 2-D wavelet image coding with novel image scan," *IEEE Trans. VLSI Syst.*, vol. 7, pp. 56–68, Mar. 1999.
[27] V. Strela, P. N. Heller, G. Strang, P. Topiwala, and C. Heil, "The application of multiwavelet filterbanks to image processing," *IEEE Trans. Image Processing*, vol. 8, pp. 548–563, Apr. 1999.
[28] D. Peng and M. Lu, "Systolic and load balanced structure for full wavelet transform and wavelet packet transform," in *Proc. 5th Joint Conf. Information Sciences*, vol. 1, Atlantic City, NJ, Feb./Mar. 2000, pp. 378–381.
[29] ——, "Non-memory-based and real-time zerotree building for wavelet zerotree coding systems," in *Proc. 15th Int. Parallel and Distributed Processing Symp. Workshops*, Cancun, Mexico, May 2000, pp. 469–475.
[30] ——, "Optimally embedding discrete wavelet transform into TESH connected parallel processors via I/O index space data dependence analysis," in *Proc. 1st Int. Conf. Parallel and Distributed Computing Applications and Technologies*, Hong Kong, May 2000, pp. 125–132.
[31] ——, "The efficient and real-time structures for 2-D constrained multiwavelet transforms derived from nonlinear I/O index space transformations," in *Proc. 7th Australia Conf. Parallel and Real-Time Systems*, Sydney, Australia, Nov. 2000, pp. 192–201.
[32] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 3rd ed. San Mateo, CA: Morgan Kaufmann, 2002.
[33] B. S. Amrutur and M. A. Horowitz, "Speed and power scaling of SRAM's," *IEEE J. Solid-State Circuits*, vol. 35, pp. 175–185, Feb. 2000.

Dongming Peng received the B.A. and M.A. degrees in electrical engineering from Beijing University of Aeronautics and Astronautics, China, in 1993 and 1996, respectively, and the Ph.D. degree in computer engineering from Texas A&M University, College Station, in 2003.

From 1996 and 1997, he was a faculty member at Beijing University. In 2002, he joined the University of Nebraska–Lincoln as an Assistant Professor. His research interests include digital image processing, computer architectures, parallel and distributed computing, and VLSI architectures. He has published more than 10 technical papers in these areas.

Dr. Peng served as the Vice Chair of the Computer Society, IEEE Nebraska Chapter, in 2004. He has also served as a referee and program committee member for several conferences and journals.

Mi Lu (SM'94) received the M.S. and Ph.D. degrees in electrical engineering from Rice University, Houston, TX, in 1984 and 1987, respectively.

She joined the Department of Electrical Engineering, Texas A&M University, College Station, in 1987 and is currently a Professor. Her research interests include parallel computing, distributed processing, parallel computer architectures and algorithms, computer networks, computer arithmetic, computational geometry, and VLSI algorithms. She has published over 100 technical papers in these areas. She is the Chair of 60 research advisory committees for Ph.D. and M.A. students

Dr. Lu was the Stream Chair of the Seventh International Conference of Computing and Information and the Conference Chair of the Fifth and Sixth International Conferences on Computer Science and Informatics. She has served on the panels of the National Science Foundation and on the 1992 IEEE Workshop on Imprecise and Approximate Computation, as well as on many conference program committees. She has also served as an Associate Editor of the *Journal of Computing and Information* and the *Information Sciences Journal*. She is a registered professional engineer, and is recognized in *Who's Who in the World*, 2001, and *Who's Who in America*, 2002.