An OCA-Based fast Algorithm for 2-D Discrete Periodized Wavelet Transform

King-Chu Hung, Jyh-Horng Jeng, Yu-Jung Huang, and Chi-Wave Hung
College of Electrical and Information Engineering, I-Shou University

Abstract

This paper presents a fast algorithm to perform the 2-D discrete periodized wavelet transform based on the operator correlation algorithm (OCA). The OCA-based algorithm needs half of the multiplications and bits required by the classical algorithm. The OCA-based algorithm is modular inherent. It can be easily mapped to VLSI design.

I. INTRODUCTION

The 2-D discrete wavelet transform (DWT) [1] is a powerful tool for digital image analysis [2-3] in which perfect reconstruction (PR) result is usually required for the inverse process. By this means, the reconstructed image from the inverse 2-D DWT is identical to the original image to the 2-D DWT. Thus, images at each resolution level (or octave) should be regarded to be periodic. This periodicity implies that periodized wavelets [4] should be used in the 2-D DWT, i.e., the 2-D DPWT.

The classical pyramid algorithm (PA) [5] and the recursive PA (RPA) [7] associated with the short-length FIR filtering algorithm [6] are essentially based on separate computation to perform the 2-D DPWT, i.e., using the 1-D DPWT for row and column transformations. Since the classical 1-D DPWT has time lag between the low-pass and high-pass filtering, the non-synchronization is an adverse factor for 2-D fast algorithm design. Moreover, long-bit representation of separate computation eventually results in bad efficiency for special-purpose hardware and VLSI design.

Applying a homeomorphic high pass filter, an operator correlation algorithm (OCA) is presented to implement the modified 2-D DPWT. Based on the OCA, this paper also proposes the 2-D OCA-based fast algorithm that needs only half of the multiplication counts (MCs) and bits representation required by the classical 2-D PA.

In next section, the 2-D DPWT is briefly reviewed. The 2-D OCA is presented in Section III. The 2-D OCA-based fast algorithm, a comparison, and finite precision analysis are presented in Section IV. Finally, Section V includes the conclusion and remark.

II. Review of the 2-D Discrete Periodized Wavelet Transform

Periodized wavelet is generally consisted of periodically shifted wavelets such as the compactly supported wavelets, which have reasonable decay. These periodized wavelets form an orthonormal basis in the Hilbert space $L^2([0,1])$. The theorem of 1-D DPWT is based on the multi-resolution analysis [1]. Given a negative integer J, $-J$ is referred to as the decomposition levels. Let $N = 2^{-J}$, N denotes the number of original sampled data of a 1-D finite signal. Let the tap vectors $\mathbf{h}$, $\mathbf{g}$ be the N-dimensional column vectors defined by $\mathbf{h} = [h_0, h_1, \ldots, h_{N-1}]^t$ and $\mathbf{g} = [g_0, g_1, \ldots, g_{N-1}]^t$ which represent the low band and high band discrete filter coefficients of the 1-D DPWT, respectively. The tap vectors $\mathbf{h}$, $\mathbf{g}$ satisfy

$$\sum_{i=0}^{N-1} h_i^2 = 1,\quad \sum_{i=0}^{N-1} h_i = \sqrt{2},\ \text{and}\ \sum_{i=0}^{N-1} (-1)^i h_i = 0, \tag{1}$$

$$g_i = (-1)^i h_{(1-i)_N}, \tag{2}$$

where $(1-i)_N$ denotes the residual of $(1-i)$ mod N.

Let $\mathbf{T}$ be the $N \times N$ matrix defined by $\mathbf{T} = \begin{bmatrix} 0 & 1 \\ \mathbf{I}_{N-1} & 0 \end{bmatrix}$, where $\mathbf{I}_{N-1}$ is the $(N-1)\times(N-1)$ identity matrix. Thus, the low-pass filter $\mathbf{H}$ and the high-pass filter $\mathbf{G}$ for the 1-D DPWT can be expressed by $\mathbf{H} = [\mathbf{T}^0\mathbf{h}, \mathbf{T}^2\mathbf{h}, \ldots, \mathbf{T}^{N-2}\mathbf{h}, \mathbf{T}^N\mathbf{h}, \mathbf{T}^{N+2}\mathbf{h}, \ldots, \mathbf{T}^{2N-2}\mathbf{h}]$ and $\mathbf{G} = [\mathbf{T}^0\mathbf{g}, \mathbf{T}^2\mathbf{g}, \ldots, \mathbf{T}^{N-2}\mathbf{g}, \mathbf{T}^N\mathbf{g}, \mathbf{T}^{N+2}\mathbf{g}, \ldots, \mathbf{T}^{2N-2}\mathbf{g}]$, respectively. Note that both $\mathbf{H}$ and $\mathbf{G}$ are $N \times N$ matrices.

For $\mathbf{H}$ and $\mathbf{G}$ defined above, let $\mathbf{s}_J = [s_1, s_2, \ldots, s_N]$ in V_J, a finite-dimensional multi-resolution approximation subspace, be the original sampled data of a finite 1-D signal For $j \in \{J, J+1, \ldots, -1\}$, the elements $\mathbf{s}_{j+1}$ and $\mathbf{d}_{j+1}$ can be obtained from $\mathbf{s}_j$. That is

$$\mathbf{s}_{j+1} = \mathbf{s}_j\mathbf{H} \text{ and } \mathbf{d}_{j+1} = \mathbf{s}_j\mathbf{G}. \tag{3}$$

The elements $\mathbf{s}_{j+1}$ and $\mathbf{d}_{j+1}$ are the 1-D DPWT coefficients corresponding to the projection of the 1-D signal onto subspaces V_{j+1} and W_{j+1}, respectively. Eq.(3) is the iterative form of 1-D DPWT.

The 2-D DPWT is usually performed by a separable approach using two transforms of 1-D DPWT. For $J < j < 0$, let $\mathbf{ss}_{j+1}$, $\mathbf{sd}_{j+1}$, $\mathbf{ds}_{j+1}$, and $\mathbf{dd}_{j+1}$ denote the projection of a finite 2-D signal (an image) onto the orthonormal bases in subspace V_{j+1} and its orthogonal complement W_{j+1}, respectively. The iterative form of 2-D DPWT can be represented by

$$\mathbf{ss}_{j+1} = \mathbf{H}^t\mathbf{ss}_j\mathbf{H},\quad \mathbf{sd}_{j+1} = \mathbf{H}^t\mathbf{ss}_j\mathbf{G},\quad \mathbf{ds}_{j+1} = \mathbf{G}^t\mathbf{ss}_j\mathbf{H},$$

$$\text{and}\quad \mathbf{dd}_{j+1} = \mathbf{G}^t\mathbf{ss}_j\mathbf{G}. \tag{4}$$

where $\mathbf{H}^t$ and $\mathbf{G}^t$ denote the transposes of $\mathbf{H}$ and $\mathbf{G}$, respectively.

From Eq. (2), it is clear that the low-pass filter relating to the high-pass filter has l-2 points of time delay at the beginning process. The classical 2-D PA based on Eq. (4) needs $l^2 + l$ MCs and $l^2 - 1$ addition counts (ACs) to compute per 2-D DPWT coefficient. For converting row transform to column

This work was supported by the National Science Council, R.O.C., under Grant NSC 87-2213-E-214-008.

transform, the 2-D PA needs N^2 words to store the intermediates of row-transformed data.

III. The OCA Algorithm for the Modified 2-D DPWT

To eliminate the time lag, one defines a linear space $\mathbf{C(G)}$ constituted of the column vectors in the non-periodic part of matrix $\mathbf{G}$. Obviously, the operations of addition and scalar multiplication are continuous for an adequate norm definition in the space $\mathbf{C(G)}$. Given an integer $\lambda \in \mathbf{Z}$, one defines a new continuous linear space $\mathbf{C(G^{2\lambda})}$ constituted of the column vectors defined in the matrix $\mathbf{G}^{2\lambda} = \mathbf{T}^{2\lambda}\mathbf{G}$. As defined above, there exists a continuous function f from $\mathbf{C(G)}$ into $\mathbf{C(G^{2\lambda})}$ such that for each element $\mathbf{d}_{j+1} \in \mathbf{C(G)}$, $f(\mathbf{d}_{j+1}) = \mathbf{d}_{j+1}\mathbf{T}^{\lambda}_{j+1} = \mathbf{d}_{j}\mathbf{T}^{2\lambda}_{j}\mathbf{G} = \mathbf{d}^{\lambda}_{j+1} \in \mathbf{C(G^{2\lambda})}$. Suppose there exists another element $\mathbf{d'}_{j+1} \in \mathbf{C(G)}$ and $\mathbf{d'}_{j+1} = f^{-1}(\mathbf{d}^{\lambda}_{j+1})$, one obtains $\mathbf{d'}_{j+1} = f^{-1}(\mathbf{d}^{\lambda}_{j+1}) = \mathbf{d}^{\lambda}_{j+1}\mathbf{T}^{-\lambda}_{j+1} = \mathbf{d}_{j+1}\mathbf{T}^{\lambda}_{j+1}\mathbf{T}^{-\lambda}_{j+1} = \mathbf{d}_{j+1}$. This result implies that the function f is injective. Thus, for a given $\lambda \in \mathbf{Z}$, the two spaces $\mathbf{C(G)}$ and $\mathbf{C(G^{2\lambda})}$ are linearly homeomorphic [8].

Let $2\lambda = l-2$, the modified 2-D DPWT using $\mathbf{G}^{l-2}$ is computational efficiency because both the low-pass and high-pass filters can deal with the common data synchronously. The modified 2-D DPWT can be implemented by an OCA process described as follows:

Let $a(x,y)$ denote an $N \times M$ image and $w(k,\ell)$ denote an $K \times L$ operator, where $0 < K \le N$ and $0 < L \le M$. The $m-shift$ operator correlation processes ($\mathbf{o_m}$) of $a(x,y)$ and $w(k,\ell)$ is defined by

$$w(k,\ell)\ \mathbf{o_m}\ a(x,y) \equiv \sum_{\ell=0}^{L-1}\sum_{k=0}^{K-1} a((i\cdot m+k)_K,(j\cdot m+\ell)_L)\cdot w(k,\ell)$$

$= b(i,j)$, where $b(i,j)$ is a $(\mathbf{int}(\frac{N-1}{m})+1)\times(\mathbf{int}(\frac{M-1}{m})+1)$ image and $\mathbf{int}(x)$ denotes the integer part of x. Let $m=2$, the 2-D modified DPWT can be then rewritten by

$$\mathbf{ss}_{j+1} = \mathbf{W}_{LL}\ \mathbf{o_2}\ \mathbf{ss}_j,\quad \mathbf{sd}_{j+1} = \mathbf{W}_{LH}\ \mathbf{o_2}\ \mathbf{ss}_j,$$

$$\mathbf{ds}_{j+1} = \mathbf{W}_{HL}\ \mathbf{o_2}\ \mathbf{ss}_j, \text{ and } \mathbf{dd}_{j+1} = \mathbf{W}_{HH}\ \mathbf{o_2}\ \mathbf{ss}_j, \qquad (5)$$

where $\mathbf{W}_{LL}$, $\mathbf{W}_{LH}$, $\mathbf{W}_{HL}$, and $\mathbf{W}_{HH}$ are called the 2-D DPWT operators defined by

$$\mathbf{W}_{LL} = [w_{m,n}] = [h_m \cdot h_n],\ \mathbf{W}_{LH} = [w_{m,n}] = [(-1)^n h_m \cdot h_{l-1-n}],$$

$$\mathbf{W}_{HL} = [w_{m,n}] = [(-1)^m h_{l-1-m} \cdot h_n], \text{ and}$$

$$\mathbf{W}_{HH} = [w_{m,n}] = [(-1)^{m+n} h_{l-1-m} \cdot h_{l-1-n}], \qquad (6)$$

where $m,n \in \{0,1,\cdots,l-1\}$. The elements $w_{m,n}$ in the operators are called the 2-D DPWT filter coefficients. To compute per 2-D DPWT coefficient, the 2-D OCA needs l^2 MCs and l^2-1 ACs without storage to save the intermediates. Note that in the 2-D OCA, l-2 columns on the left side of $\mathbf{ss}_j$ should be duplicated on the right side of $\mathbf{ss}_j$ so that the 2-D DPWT coefficients of l/2-1 columns on the right side of $\mathbf{ss}_{j+1}$ can be undistorted. Similarly, the duplication of l-2 rows on the topside of $\mathbf{ss}_j$ is needed. These data are called boundary data of $\mathbf{ss}_j$ and $\mathbf{ss}_{j+1}$.

IV. The OCA-Based Fast Algorithm

The basic idea behind the OCA-based fast algorithm is to remove the redundancy of 2-D DPWT filter coefficients ($|w_{m,n}|$) and make the process suit for sequential data input of row scan. The fast algorithm depicted in Fig.1 is principally composed of three functional processes: a parallel multiplication and two orientated data accumulations called row-accumulator and column-accumulator. Firstly, each one of the input $\mathbf{ss}_j$ will be multiplied by all the non-duplicate $|w_{m,n}|$ to produce the weighted data ($a_{m,n}$). Then, in terms of an adequate orientation arrangement for the weighted data accumulation, the four bands 2-D DPWT coefficients at the resolution level j+1 can be derived simultaneously. For this purpose, l row-accumulators and two column-accumulators are needed. In each one of the row and column accumulators, data are accumulated in two orientations, which are mutually opposite for low-pass and high-pass filtering. Essentially, both the row and column accumulators are based on a similar algorithm of pipeline processing. The l row-accumulators accumulate the weighted data of input in row orientation synchronously. The column-accumulator accumulates the row-accumulated data in column orientation sequentially. Hence, for each band processing, the column-accumulator needs (l-1) buffers to store row-accumulated data of (l-1) rows. Each buffer has N/2 cells of storage. Due to the mirror-reflection properties of the four 2-D DPWT operators, (l-2) buffers in each column-accumulator can be shared for two-band processing. For sharing buffer, the I/O paths are controlled by row coordinate. The desired outputs appear only at even coordinate of the row and column of current input. One goal of the algorithm design is to make the last input and the production of final output can be simultaneous. To prevent from double input of the boundary data of $\mathbf{ss}_j$, the semi-manufacture of accumulated boundary data should be preserved in the algorithm to produce undistorted 2-D DPWT coefficients on the right side and down side boundary of $\mathbf{ss}_{j+1}$. Hence, the boundary data of $\mathbf{ss}_{j+1}$ can be generated in parallel at $x = N_1$ or $y = N_1$, where $N_1 = 2^{-j}$, x and y denote row and column coordinates of the current input of $\mathbf{ss}_j$, respectively.

Let N=16 and l=10, An example of the OCA-based fast algorithm is illustrated in Tables I and II. The symbol $\rightarrow$ denotes that the right-side output of accumulated data will be produced at the current left-side input with coordinate x or y The example clearly reveals the whole process. The algorithm has the synchronism property, and independence from the input data and filter length. It is also obvious that the fast algorithm has capability to decompose an image to any desired terminate level.

A comparison of several 2-D DPWT algorithms is shown in Table III. Essentially, the RPA is a 1-D method, it can not be directly applied to the 2-D case by a simple RAM transposer. The main idea behind the RPA is to derive a specific output order by the earliest instance schedule. This idea can be applied to the 2-D case based on interspersing the octave data of other stages into the octave data of first stage [9]. In principle, the 2-D

RPA is a separate computation algorithm. It needs $lC+C$ counts of both multiplication and addition to compute each 2-D DPWT coefficient, where C is the needed counts of corresponding 1-D operation. To convert row transform to column transform, row-transformed data of all stages should be preserved for earliest instance scheduling. Hence, the 2-D RPA needs $(l-1)N\sum_{j=-1}^{-J+1} 2^{j+1} \approx 2(l-1)N$ words to store row-transformed data. From Fig.3, it is clear that the 2-D OCA-based fast algorithm needs $lN+B$ words to store row-transformed data, where $B=(l-2)\ (2N+1)$ denotes the words of storage requirement of boundary data for PR desire.

Four Daubechies' wavelets of filter length l=4,6,8,and 10 are used to analyze the finite precision performance of separate computation and non-separate computation i.e., directly using 2-D filter coefficients. The round off error of finite-bits representation of 2-D filter coefficients is shown in Figs. 2 and 3 (including one sign bit). It reveals that for a desired accuracy of finite-bits representation, non-separate computation can save nearly half of the bits needed by the separate computation.

V. Conclusion

Based on a homeomorphic high pass filter, an operator correlation algorithm (OCA) has been presented in this paper to implement the modified 2-D DPWT. Moreover, a fast algorithm of OCA-based suiting for special-purpose hardware was also proposed. The 2-D OCA-based algorithm needs only half of the multiplications and bits required by the classical algorithm to compute one 2-D DPWT coefficient and requires $lN+B$ cells of storage to compute $N\times N$ points of 2-D DPWT.

The 2-D OCA-based algorithm inhered in simple modular can be easily mapped to VLSI design [10]. For VLSI design, the original data should be interleaved in order to perform the decompositions of all stages simultaneously. In this case, nearly double storage shown in table I is needed to save all the row-accumulated data. The storage cell is implemented by shift-register, which is not expensive.

REFERENCES

[1] S. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," IEEE Trans. on Pattern Analysis and Machine Intell., vol.11, no 7, pp.674-693, July 1989.

[2] A. Averbuch, D. Lazar, and M. Israeli, "Image compression using wavelet transform and multiresolution decomposition," IEEE Trans. on Image Processing, vol.5, no.1, pp.4-15, Jan. 1996.

[3] M. R. Banham and A. K. Katsaggelos," Spatially adaptive wavelet-based multiscale image restoration," IEEE Trans on Image Processing, vol.5, no.4, pp.619-634, Apr. 1996.

[4] I. Daubechies, Ten Lectures on Wavelets, No.61 in CBMS-NSF Series in Applied Mathematics, SIAM, Philadelphia, 1992.

[5] O. Rioul and P. Duhamel, "Fast algorithms for wavelet transforms," IEEE Trans. on Inform. Theory, vol.38, no.2, pp.569-586, March 1992.

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

[7] M. Vishwanath, "The recursive pyramid algorithm for the discrete wavelet transform," IEEE Trans. on Signal Processing, vol.42, no.3, pp.673-677, Mar. 1994

[8] W. Rudin, Functional analysis, McGraw-Hill, Inc., New York, 1991.

[9] M. Vishwanath, R. M. Owens, and M. J. Irwin, "VLSI architectures for the discrete wavelet transform," IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing, vol.42, no.5, pp.305-316, May 1995.

[10] K. C. Hung, T. K. Truong, and Y. J. Huang "A Non-Separate VLSI Architecture for 2-D Discrete Periodized Wavelet Transform," Tech. Rep, EE-97-02, Dept. of Electronic Eng., I-Shou Univ., 1997.

Table I. Accumulation results of the row accumulator in the OCA-based fast algorithm (N=16). (for LL band)

$N_1=2^{-j}$	x: Column Coordinate of $\mathbf{ss}_j$, $m=1,\cdots,l$	
$N_1=16$	x=2 $\rightarrow bs_{m,1}=ts_{m,10}$	x=14 $\rightarrow s_{m,3}=ts_{m,10}$
	x=4 $\rightarrow bs_{m,2}=ts_{m,10}$	x=16 $\rightarrow s_{m,4}=ts_{m,10}$
	x=6 $\rightarrow bs_{m,3}=ts_{m,10}$	x=16 $\rightarrow s_{m,5}=bs_{m,1}+ts_{m,8}$
	x=8 $\rightarrow bs_{m,4}=ts_{m,10}$	x=16 $\rightarrow s_{m,6}=bs_{m,2}+ts_{m,6}$
	x=10 $\rightarrow s_{m,1}=ts_{m,10}$	x=16 $\rightarrow s_{m,7}=bs_{m,3}+ts_{m,4}$
	x=12 $\rightarrow s_{m,2}=ts_{m,10}$	x=16 $\rightarrow s_{m,8}=bs_{m,4}+ts_{m,2}$
$N_1=8$	x=2 $\rightarrow bs_{m,1}=ts_{m,10}$	x=8 $\rightarrow s_{m,1}=bs_{m,1}+ts_{m,8}$
	x=4 $\rightarrow bs_{m,2}=ts_{m,10}$	x=8 $\rightarrow s_{m,2}=bs_{m,2}+ts_{m,6}$
	x=6 $\rightarrow bs_{m,3}=ts_{m,10}$	x=8 $\rightarrow s_{m,3}=bs_{m,3}+ts_{m,4}$
	x=8 $\rightarrow bs_{m,4}=ts_{m,10}$	x=8 $\rightarrow s_{m,4}=bs_{m,4}+ts_{m,2}$
$N_1=4$	x=2 $\rightarrow bs_{m,1}=ts_{m,10}$	x=4 $\rightarrow s_{m,1}=bs_{m,1}+ts_{m,8}$
	x=4 $\rightarrow bs_{m,2}=ts_{m,10}$	$+ts_{m,4}$
		x=4 $\rightarrow s_{m,2}=bs_{m,2}+ts_{m,6}$

Table II. Accumulation results of the column accumulator in the OCA-based fast algorithm (N=16). (for LL band)

$N_1=2^{-J}$	y: Row Coordinate of $\mathbf{ss}_j$, $q=1,\cdots,N_2$,	
$N_1=16$	y=2 $\rightarrow bss_{1,q}=t_{10,q}$	y=14 $\rightarrow ss_{3,q}=t_{10,q}$
	y=4 $\rightarrow bss_{2,q}=t_{10,q}$	y=16 $\rightarrow ss_{4,q}=t_{10,q}$
	y=6 $\rightarrow bss_{3,q}=t_{10,q}$	y=16 $\rightarrow ss_{5,q}=bss_{1,q}+t_{8,q}$
	y=8 $\rightarrow bss_{4,q}=t_{10,q}$	y=16 $\rightarrow ss_{6,q}=bss_{2,q}+t_{6,q}$
	y=10 $\rightarrow ss_{1,q}=t_{10,q}$	y=16 $\rightarrow ss_{7,q}=bss_{3,q}+t_{4,q}$
	y=12 $\rightarrow ss_{2,q}=t_{10,q}$	y=16 $\rightarrow ss_{8,q}=bss_{4,q}+t_{2,q}$
$N_1=8$	y=2 $\rightarrow bss_{1,q}=t_{10,q}$	y=8 $\rightarrow ss_{1,q}=bss_{1,q}+t_{8,q}$
	y=4 $\rightarrow bss_{2,q}=t_{10,q}$	y=8 $\rightarrow ss_{2,q}=bss_{2,q}+t_{6,q}$
	y=6 $\rightarrow bss_{3,q}=t_{10,q}$	y=8 $\rightarrow ss_{3,q}=bss_{3,q}+t_{4,q}$
	y=8 $\rightarrow bss_{4,q}=t_{10,q}$	y=8 $\rightarrow ss_{4,q}=bss_{4,q}+t_{2,q}$
$N_1=4$	y=2 $\rightarrow bss_{1,q}=t_{10,q}$	y=4 $\rightarrow ss_{1,q}=bss_{1,q}+t_{8,q}+$
	y=4 $\rightarrow bss_{2,q}=t_{10,q}$	$t_{4,q}$
		y=4 $\rightarrow ss_{2,q}=bss_{2,q}+t_{6,q}+$

		$+ts_{m,2}$
$N_1 = 2$	x=2 $\rightarrow$ $bs_{m,1} = ts_{m,10}$	x=2 $\rightarrow$ $s_{m,1} = bs_{m,1} + ts_{m,8}$ $+ts_{m,6} + ts_{m,4} + ts_{m,2}$

		$t_{2,q}$
$N_1 = 2$	y=2 $\rightarrow$ $bss_{1,q} = t_{10,q}$	y=2 $\rightarrow$ $ss_{1,1} = bss_{1,1} + t_{8,1}$ $+t_{6,1} + t_{4,1} + t_{2,1}$

Table III. Comparison of different 2-D DPWT algorithms for per 2-D DPWT coefficient (N=1024).

Algorithms	MCs.	ACs	Storage for Row-Column Conversion	Usage of Filter Coeffs.	Undistorted Coefficients
Classical 2-D PA	$l^2 + l$	$l^2 - 1$	N^2	1-D Coeffs.	Yes
Short-Length 2-D RPA	$lC+C$ (C $\approx$ 3/4*l)	$lC+C$ ($C \approx$ 3/4*l+1/2)	$2(l-1)N$	1-D Coeffs.	No
2-D OCA	l^2	$l^2 - 1$	None	2-D Coeffs.	Yes
2-D OCA-Based	$(l^2 + l)/2$	$l^2 - 1$	$lN + B$	2-D Coeffs.	Yes

```
Begin(2-D DPWT)
 for(j=J to -1) {
  for( $x, y = 1$ to $2^{-j}$ ) {
  input $\mathbf{ss}_j$ .
  for(m, n=1 to l) {   // weighted data
   if (m<n)  $a_{m,n} = a_{n,m}$ ;
   else     $a_{m,n} = ss_j * |w_{m-1,n-1}|$ ; }
  for(m=1 to l) {
   Row_Accumulator(); }
// obtain the $\mathbf{ss}_{j+1}$ and $\mathbf{ds}_{j+1}$
  input $s_{m,q}$ ;
  Column_Accumulator();
// obtain the $\mathbf{sd}_{j+1}$ and $\mathbf{dd}_{j+1}$
  input $d_{m,q}$ ;
  Column_Accumulator(); }}
//--------------------------------------------------
Column_Accumulator()
```

The process is similar in algorithm to the **Row_Accumulator** except that the arguments *ts* and *td* are replaced by *buffers* ($t_{i,q}$), which are controlled by y to store row-accumulated data If y is even, *buffers* of i=even are used to store low band octaves while *buffers* of i=odd are used to store high band octaves. If y is odd, the correspondence of buffers to store octaves is changed.

```
//----------------------------------------------------------------------------------
Row_Accumulator()
 if ( $x == 1$ ) { $q$=0;          // initialization
  for( $n$=1 to $l$, $step$=2 ) {
   $ts_{m,n} = a_{m,n}$;             // low-pass filtering
   $td_{m,n} = a_{m,l-n+1}$; }}      // high-pass filtering
 else { for( $n$=1 to $l$ ) {
   $ts_{m,l-n+1} = ts_{m,l-n} + a_{m,l-n+1}$ ;   // low-pass filtering,  $ts_{m,0} = 0$
   $td_{m,l-n+1} = td_{m,l-n} + (-1)^n a_{m,n}$ }}   // high-pass filtering,  $td_{m,0} = 0$
 if ($x$ is even) { if ( $1 < x < l$ ) { $bs_{m,x/2} = ts_{m,l}$ ;  $bd_{m,x/2} = td_{m,l}$ ; }
  if ( $l \le x$ ) {$q$=$q$+1;  $s_{m,q} = ts_{m,l}$ ;  $d_{m,q} = td_{m,l}$ ; }
  if ( $x = N_1$ ) { for( $n = q$+1 to $N_2$ ) {  $s_{m,n} = bs_{m,n-q} + \sum_{n'\ge 2} ts_{m,n'}$ ;
     $d_{m,n} = bd_{m,n-q} + \sum_{n'\ge 2} td_{m,n'}$ ; }}} //  $n' = l - 2(n-q) - k \cdot N_2$ ,
                                  $k \in \{0,1,2,\cdots\}$
```

Fig. 1. The OCA-based fast algorithm for the modified 2-D DPWT.

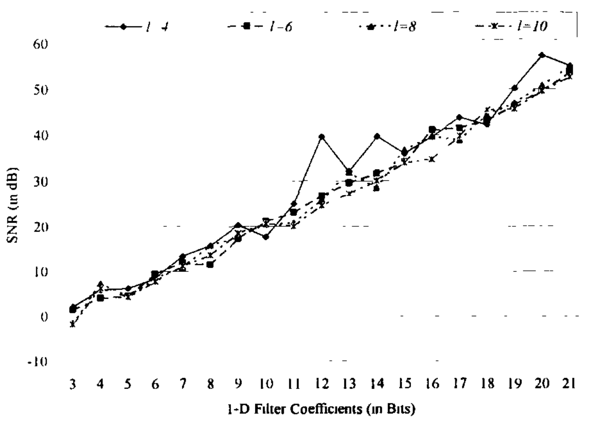

Fig.2. Round off error analysis of finite-bits 2-D filter coefficients using two transformed 1-D filter coefficients.

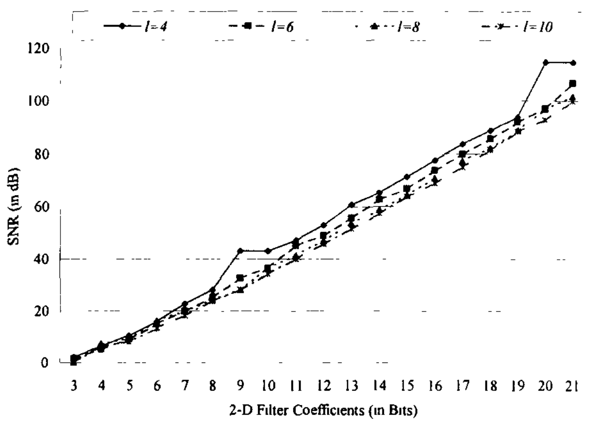

Fig.3. Round off error analysis of finite-bits 2-D filter coefficients directly using 2-D filter coefficients.