A MEMORY EFFICIENT ALGORITHM FOR MULTI-DIMENSIONAL WAVELET TRANSFORM BASED ON LIFTING

Zeinab Taghavi, Shohreh Kasaei

Sharif University of Technology, Tehran, Iran
ztaghavin@mehr.sharif.edu, skasaei@sharif.edu

ABSTRACT

Conventional implementation of multi-dimensional wavelet transform (e.g. 3-D wavelet) requires whether a high amount of "in access" memory or a continual access to slow memory of a processor which makes it infeasible for most applications. In this paper, we proposed a novel algorithm for computation of an n-D discrete wavelet transform (DWT) based on lifting scheme. In addition to benefits of lifting scheme (which causes a major reduction in computational complexity and performs the total computations in time domain), our real-time approach computes the coefficients for all kinds of 1st and 2nd generation wavelets with short delay and optimized utilization of the slow and fast memories of a processor.

1. INTRODUCTION

Multi-dimensional discrete wavelet transform has been considered to be used in many fields such as image and video processing applications. In these applications one needs a fast and memory efficient algorithm to compute the transform. In traditional n-D DWT algorithms, the signal is first loaded on the memory and then transform coefficients are computed. This method is simple, but its main drawback is that the processor should have access to the whole signal simultaneously, and further none of the coefficients are ready before the end of the whole process.

When implementing, one should consider the process time in addition to necessary memory size. The number of "reads from" and "writes on" the slow memory of a processor is one of the main parameters that affects the process time. If an algorithm is designed in such a way that while using a reasonable amount of the fast memory of a processor is able to have at most one read and write for each sample of the signal, the process speed will increase considerably.

One of the schemes used to compute DWT is the *lifting* scheme. Suppose that $P(z)$ is the polyphase matrix of analysis quadrature mirror filters (QMF) of a wavelet, which can be of any type; minimum phase (orthonormal) or linear phase (biorthogonal). The lifting scheme expresses that $P(z)$ can be decomposed into elementary matrices as:

$$P(z)=\prod_{i=1}^{m}\begin{bmatrix}1 & s_i(z)\\ 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0\\ t_i(z) & 1\end{bmatrix}\begin{bmatrix}K & 0\\ 0 & 1/K\end{bmatrix},$$

where $s_i(z)$ and $t_i(z)$ for $1\le i\le m$ are Laurent polynomials and K is a nonzero constant [1]. This method is FFT free and in [6] has been shown that it reduces the computational complexity to nearly 1/2 and for long filters even to 1/4, relative to the standard algorithm.

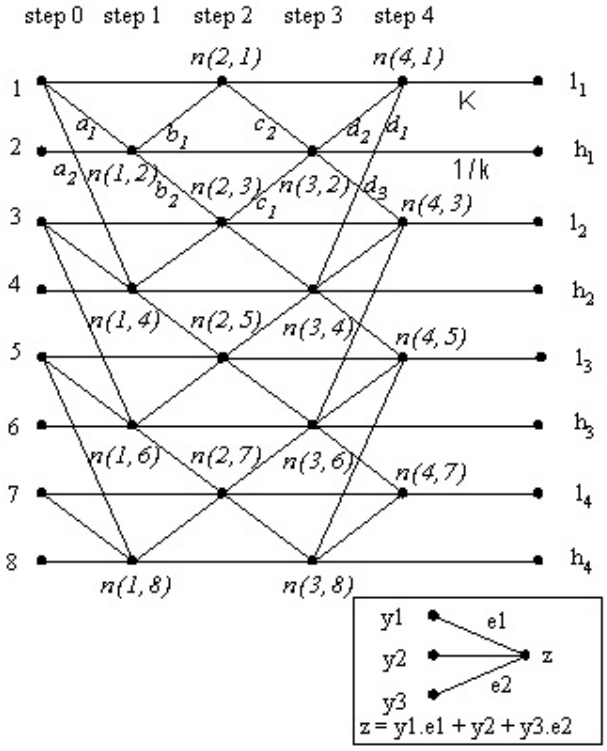


Figure 1. An example of lifting network for a lifting polyphase matrix decomposition.

Figure 1 illustrates an example of a lifting network for a general parametric lifting polyphase matrix decomposition, as:

$$P(z)=\begin{bmatrix}1 & a_1+a_2z\\ 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0\\ b_1+b_2z & 1\end{bmatrix}\begin{bmatrix}1 & c_1z^{-1}+c_2\\ 0 & 1\end{bmatrix}$$
$$\begin{bmatrix}1 & 0\\ d_1z^{-1}+d_2+d_3z & 1\end{bmatrix}\begin{bmatrix}K & 0\\ 0 & 1/K\end{bmatrix}.$$

Note that based on lifting theory the decomposition structure can be symetric/asymetric for both orthonormal and biorthogonal filters. In this figure each edge between two nodes (at two

consequent steps) and the corresponding weights indicates that the node in the lower step should be multiplied to the weight and be added to the amount of the node in the higher step. It is worth noting that every parallel edge between two specific steps has the same weight. Also, all horizontal edges have the weight of one except for those that are ended to wavelet coefficients. In this figure l_is and h_is indicate the low frequency and high frequency wavelet coefficients, respectively.

A method for implementing lifting scheme is to compute each step thoroughly for every sample and then move to the next step [7]. This method has two main drawbacks:

I) either all of the signal should be available in a fast memory or each sample of signal should be read from and written on a slow memory at least as many times as the number of steps in lifting, and

II) the procedure even takes the first transformation coefficient only when it has the whole signal data, and therefore this implementation can not be used in real-time.

In this paper we suggest an implementation algorithm of lifting scheme that overcomes these drawbacks.

2. THE PROPOSED ALGORITHM

It is more proper to first analyze the algorithm for one level of 1-D DWT, and then generalize it to a multilevel and multi-dimensional case.

To overcome the drawbacks explained in the previous section, here we propose an algorithm such that as each sample enters the fast memory (a constant length buffer), all possible computations on samples in buffer (in different steps) will be performed, through which a wavelet coefficient is computed. Then the procedure outputs the prepared coefficient and inputs the next sample to the buffer. It is worth mentioning that the length of the used buffer is in the order of the wavelet QMFs half-length.

As can be seen in figure 1, all nodes in the i^{th} row are intermediate values of a unique variable *i*. We say a node *n(s,i)* is *active*, if the computation of the variable *i* is in step *s*, therefore it should be in the buffer in fast memory of processor. By inspection, one can find that the nodes in step *s+1* which are connected to node *n(s,i)*, can be updated from it only when the computation of *n(s,i)* has been finalized (*i.e., n(s,i)* is *complete*). When these nodes have received all of the information from *n(s,i)* (*i.e., n(s,i)* is *free*), computation of variable *i* can be transferred to node *n(s+2,i)* (*i.e., n(s+2,i)* is activated instead of *n(s,i)*). These principles can be realized by two sub-procedures:

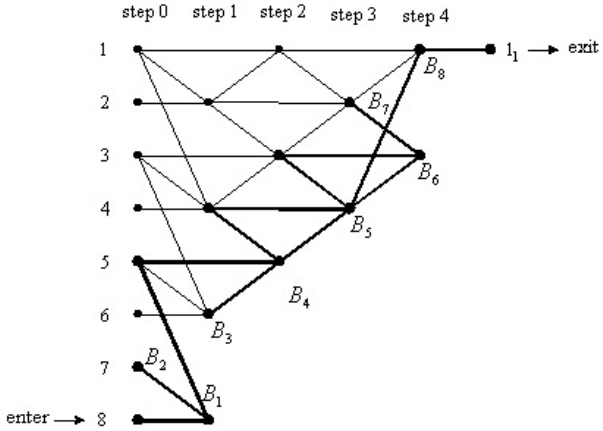
I. If node *n(s,i)* is complete, active nodes connected to it in step *s+1* can update their values from *n(s,i)*.

II. If *n(s,i)* is free, activate *n(s+2,i)*. So it can update itself from complete nodes of step *s+1*, which are connected to it.

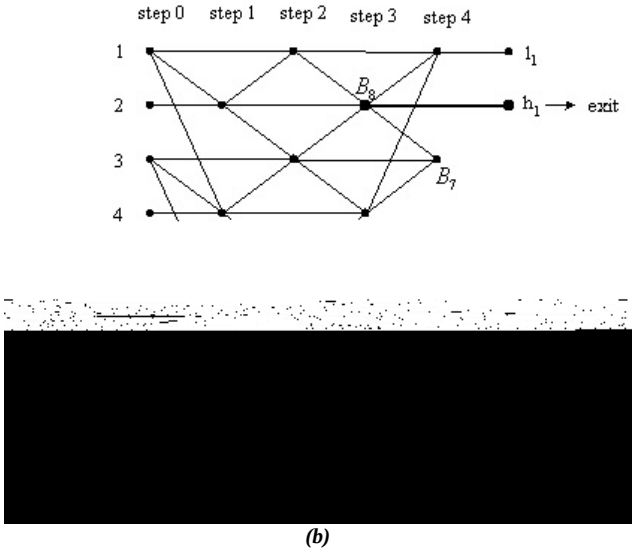
Regarding the method of polyphase matrix decomposition, in lifting network (as can be seen in figure 1) in step 1 even numbered nodes can always be updated from odd numbered nodes. As a result, we can execute the first sub-procedure if an odd sample enters, and the second sub-procedure if an even sample inputs. Besides, when entering a new sample, each step of the lifting network needs to call at most one of these sub-procedures for nodes available in the buffer. So the order of calling these sub-procedures could be such that:

- by entering an odd sample, the sub-procedure I will be executed at most once for each step on active nodes (the *oddupdate* procedure), and
- by entering an even sample the sub-procedure II will be run for each step, if necessary (the *evenupdate* procedure).

Figures 2-a and 2-b illustrate the progress of the computation in lifting network of figure 1 after the execution of *evenupdate* and *oddupdate* procedures, respectively. The bolded edges in each figure are computed during the last procedure execution. In these figures B_is are the elements of the 8-length buffer so the corresponding nodes are active nodes in buffer, and the omitted edges have not been computed yet. In these procedures, first one sample of signal is entered and then after the completion of the procedure, one wavelet coefficient is computed.



(a)



(b)

Figure 2. Figures *a* and *b* show the progress of computation in lifting network of figure 1, after the execution of *evenupdate* and *oddupdate* procedures,respectively.

As stated above, this algorithm needs a buffer with a length of N_B, that is nearly as long as the half-length of QMFs. To extend the algorithm to multi-levels, when a low frequency coefficient of one level is ready (one l_i), it immediately enters to the next level as a new sample. Figure 3 shows the 2-level wavelet computation of figure 1. Thus for the J-level case, the size needed for fast memory will be just N_BJ.

In order to employ the advantages of our algorithm for multi-dimensional signals such as video, it is proper to use its structure only for the longest dimension (*e.g.*, the time dimension for video) and use the conventional implementation of lifting for other dimensions. In this case, we need a memory of size JN_BN_I where N_I is the amount of memory needed for one sample of the signal in the mentioned dimension (*e.g.*, an image for video). It is obvious that using this amount of memory is completely feasible for video signals with normal resolutions.

3. PERFORMANCE COMPARISON

There are only few researches done on n-D wavelets with memory restriction considerations. Some of them have not considered a true n-D wavelet transform. For instance they study it on signals decomposed to groups of pictures (GOP), groups of frames (GOF), or even perform independent transforms on different dimensions (*e.g.*, they implement a 3-D wavelet transform by performing a 2-D wavelet on each frame and an unrelated shorter 1-D wavelet on the temporal direction) [2, 4]. Another class like [3] makes use of parallel implementations on parallel processors to speed up the process.

Reichel in [6] suggested a method of 2-D wavelet to reduce memory requirements. In his work he needs a 40-length buffer for the 9-7 Daubechies wavelet (the only example stated in his algorithm). It is worth mentioning that with the same filter our algorithm requires only a 5-length buffer. Also, his algorithm needs multi reads and writes per pixel while ours needs only one.

In [5] a memory-constrained routine for just the 3-D 9-7 Daubechies wavelet transform is introduced. In fact this routine can be viewed as a special case of our method while our algorithm is also capable of dealing with all kinds of mother wavelets.

Also in [8] Jiang and Ortega proposed a line-based system in which they use GOP scheme (as appose to our algorithm) followed by a boundary postprocessing approach. In addition, compared to our algorithm, they need more total memory requirements in computational process, for both parallel and sequential schemes. Therefore to the best knowledge of the authors, the novelty of the proposed algorithm is that it simultaneously satisfies the following interesting properties:

1. Is based on lifting scheme, and therefore:
   a. is approximately 2 to 4 times faster than standard DWT algorithm for sufficient long filters (as can be seen in table 1),
   b. is an FFT free scheme and performs in time domain,
   c. can be used for all types of wavelet filters, containing 1st and 2nd generations,
   d. computes wavelet coefficients in-place, and
   e. has capability of implementing in parallel.
2. Requires few fast memories independent from signal length. It needs only JN_BN_I byte of fast memory for a J-level wavelet transform, where N_B is the buffer length and N_I is the memory requirement for each frame of the signal. Table 1 shows the N_B for some wavelet filters. It can be seen that N_B is approximately as long as half-length of the filter.
3. Needs only one read and write per each sample of a signal from the slow memory of processor.
4. Uses the locality property of wavelet, therefore for computing a wavelet coefficient it needs only necessary samples of signal, and thus can be implemented in real-time.

Table 1 shows the performance of the proposed algorithm for a number of orthonormal and biorthogonal wavelets. Columns of this table show the wavelet type, the wavelet name, length of the QMFs, length of the buffer required in the fast memory (N_B), and the computational complexity rate relative to the standard algorithm (R), respectively. As can be seen in this table, the computational complexity of the proposed algorithm for sufficiently long filters is about 1/2 and even in some cases near 1/4 of the standard algorithm. We have also proposed a wavelet that offers R of 0.292 (longer filter can be designed to offer R of less than this amount up to 0.25). Also, as shown in this table, N_B is approximately as long as half-length of the QMFs (for sufficiently long filters).

Table 1. Performance of the proposed algorithm for a number of orthonormal and biorthogonal wavelets. [N_B: length of buffer required in fast memory, R: rate of computational complexity relative to the standard algorithm].

Wavelet Type	Wavelet Name	Filters Length	N_B	R
Biortogonal	Daubechies	7-9	5	0.6
	CDF(2, 8)	3-17	9	0.579
	Proposed	75-75	38	0.292
Orthonormal	Coiflet (1)	6	5	0.636
	Daubechies	20	11	0.447
	Vaidyanathan	24	13	0.53

4. CONCLUSION

In this paper we have explained a novel real-time algorithm for implementing n-D wavelet transform. The algorithm is based on lifting scheme and therefore is faster than standard algorithm. It uses the fast and slow memory of a processor efficiently, which makes it memory efficient and fast. We have also given a performance comparison among other available algorithms and have shown that the proposed algorithm is the only feasible and general algorithm capable of implementing all types of n-D wavelets (especially in 3-D).

5. REFERENCES

[1] I. Daubechies and W. Sweldens. "Factoring Wavelet Transforms into Lifting Steps," *J. fourier Anl. Appl.*, Vol. 4, pp. 247-269, 1998.

[2] E. Moyano, L. Orozco-Barbosa, F. J. Quiles, and A. Garrido, "A New Fast 3D Wavelet Transform Algorithm for Video Compression," *IEEE Pacific Rim Conference on Communications, Computers and signal Processing (PACRIM)*, pp.156-159, 2001.

[3] R. Kutil and A. Uhl. “Hardware and Software Aspects for 3-D Wavelet Decomposition on Shared Memory MIMD Computers”, *Proceedings of ACPC’99*, volume 1557 of Lecture Notes on Computer Science, pp. 347-356, Springer-Verlag, 1999.

[4] K.H. Goh, J.J. Soraghan, and T.S. Durrani, “New 3-D Wavelet Transform Coding Algorithm for Image Sequences,” *Electronics letters*,Vol. 29(4), pp. 401-402, 1993.

[5] J. Xu, Z. Xiong, S. Li, and Y.-Q. Zhang, “Memory-Constrained 3-D Wavelet Transform for Video Coding without Boundary Effects,” *IEEE Trans. Circuits and Systems for Video Tech.*, vol. 12, pp. 812-818, September 2002.

[6] J. Reichel. *Complexity Related Aspect of Image Compression*. PHD. Thesis, Signal processing Lab, Swiss Federal Institute of Technology , Switzerland , Febraury 2001.

[7] G. Fernandez, S. Periaswamy, and W. Sweldens, “LIFTPACK: a Software Package for Wavelet Transforms using Lifting,” *Proceedings of the SPIE - The International Society for Optical Engineering 2825*, pp. 396-408, 1996.

[8] W. Jiang, A. Ortega, “Lifting Factorization-Based Discrete Wavelet Transform Architecture Design,” *IEEE Trans. Circuits and Systems for Video Tech.*, vol. 11, pp. 651-657, May 2001.

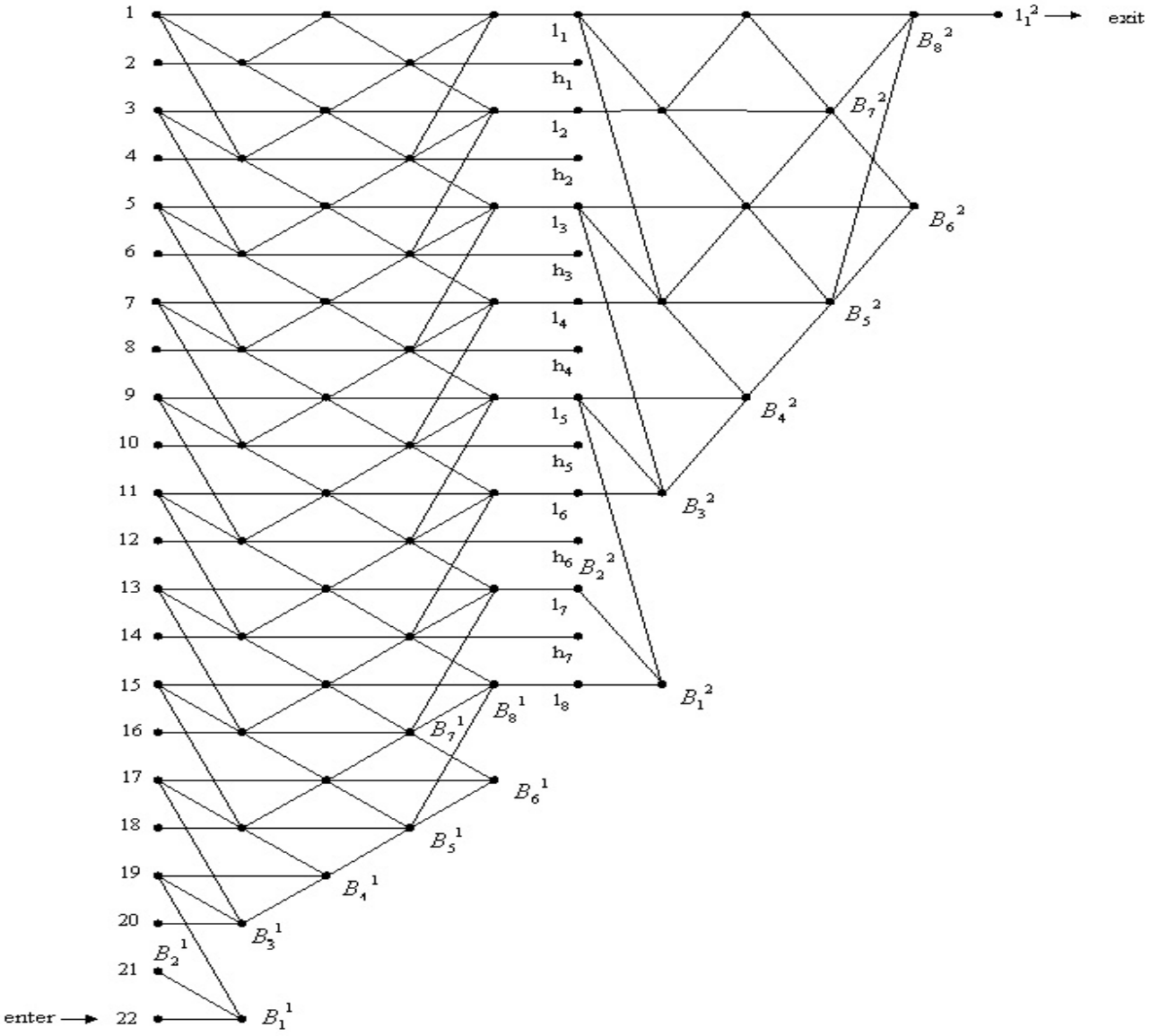


Figure 3. 2-level wavelet computation of figure 1 (in this example the size needed for fast memory is only 2*8=16 unit).