SYSTOLIC ARRAY FOR REALIZATION OF DISCRETE WAVELET TRANSFORM

Dr.Waleed A. Mahmoud,
University of Baghdad
College of Eng.\ Electrical Dept.

Ahmed S. Hadi
University of Baghdad
Al_Khawarizmy College of Eng.\Information Dept.

ABSTRACT:

This paper introduce a new method for using a systolic array to perform the one, and two dimension discrete wavelet transform (1-D DWT, and 2-D DWT).

The 1-D needs only a semi-systolic array for its realization. However, it was found that the 2-D method needs the combination of two types of semi-systolic array into one systolic array to achieve its implementation.

الخلاصه:

في هذا البحث تم تقديم طريقة جديدة لايجاد تحويل ال"wavelet" المتقطع احادي البعد (1-D DWT) وثنائي البعد (2-D DWT) باستخدام مصفوفة ال" systolic ".

في تحويل ال"wavelet" المتقطع احادي البعد (1-D DWT) استخدمنا شبه مصفوفة ال"systolic". بينما في تحويل ال"wavelet" المتقطع ثنائي البعد (2-D DWT) تم دمج مصفوفتان من شبه ال"systolic" لانتاج مصفوفة "systolic" كاملة.

KEYWORDS:
1-D DWT, 2-D DWT , Semi-systolic array, and Systolic array.

INTRODUCTION

One of the signal-processing problems is the efficient implementation in VLSI of the discrete wavelet transform (DWT). The DWT is important in many applications such as spectral analysis, digital filtering, image processing . . . etc. There is a much interest in two dimension discrete wavelet transform (2-D DWT), to gain a better understanding for pattern analysis [Martin 1999].

For VLSI implementation, systolic arrays are attractive because they are constructed with a repetitive array of identical cells. The use of identical cells simplifies the design process. Further, if the interconnection between cells can be restricted to the nearest neighbor connections (i.e., only adjacent cells communicate with each other), high speed is possible [Swartzlander 1987].

The DWT is better suited to realization with a systolic array, due to the high density, and high speed processors of the systolic array that result from the short and systematic wire routing and concurrency [Lee 1992, O'Brien 1989].

This paper is organized as follows: Section II presents a computation of 1-D DWT and 2-D DWT. Section III describes semi-systolic array for matrix multiplication. Section IV presents a systolic array for computing 1-D DWT, and 2-D DWT. Section V presents conclusions.

COMPUTATION OF DWT:

The structure of a one dimensional DWT is shown in fig. (1). X(n) is the 1-D input signal. H(n) and g(n) are the analysis lowpass and high pass filters which, split the input signal into two subbands: lowpass and highpass. The lowpass and highpass subbands are then downampled generating $X_L(n)$ and $X_H(n)$ respectively [Goswami 1999]. The 1-D DWT can compute as follows:

1. Checking input dimensions: input vector should be of length N, where N must be power of two.
2. Construct a transformation matrix T: using transformation such as Haarr, or Daubechies Db4 [Burrus 1998]. As shown below for N=4:

$$T_{Haar} = \begin{bmatrix} h_0 & h_1 & 0 & 0 \\ 0 & 0 & h_0 & h_1 \\ h_1 & -h_0 & 0 & 0 \\ 0 & 0 & h_1 & -h_0 \end{bmatrix}$$

$$T_{Db4} = \begin{bmatrix} h_0 & h_1 & h_2 & h_3 \\ h_2 & h_3 & h_0 & h_1 \\ h_3 & -h_2 & h_1 & -h_0 \\ h_1 & -h_0 & h_3 & -h_2 \end{bmatrix}$$

3. Transformation of input vector: which can be done by multiplying the N×N constructed transformation matrix by the N×1 input vector.

$$Y_{N\times 1} = T_{N\times N} . X_{N\times 1} \qquad [1]$$

There are two main types of methods for computing DWT for 2-D signals, which are separable and non-separable algorithms. Separable methods simply work on each dimension in series. The typical approach is to process each of the rows in order and then process each column of the result. Non-separable methods work in both signal dimensions at the same time [Strela 1996]. Because the non-separable method can save time and computation, it will be use here.

The 2-D DWT can compute using non-separable method as follows:

1. Checking input dimensions: input matrix, X, should be of length N×N, where N must be power of two.
2. For an N×N matrix input 2-D signal, X construct an N×N transformation matrix, T, using Haarr, or Daubechies Db4 [Burrus 1998].
3. Apply transformation by multiplying the transformation matrix by the input matrix by the transpose of the transformation matrix.

$$Y_{N\times N} = T_{N\times N} . X_{N\times N} . T^T_{N\times N} \qquad [2]$$

This multiplication of the three matrices result in the final discrete wavelet transformed matrix.

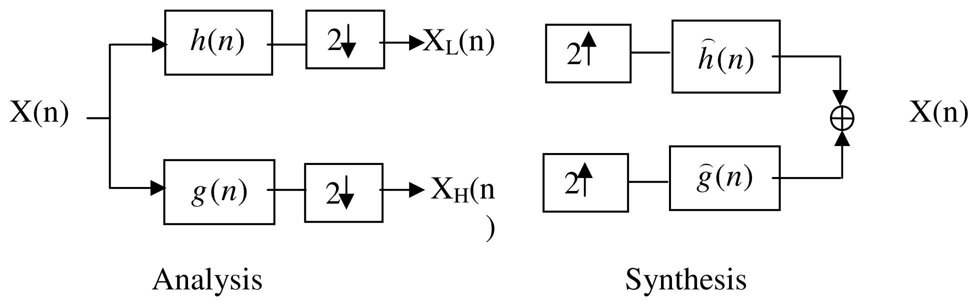


Fig. (1): Analysis and synthesis stages of a 1-D DWT

SEMI-SYSTOLIC ARRAY FOR MATRIX MULTIPLICATION:

Since matrix multiplication is the basic operation for the DWT it will describe here. Our approach is based on systolic arrays for matrix multiplication. Kung [Kung 1988] has identified two types of semi-systolic arrays for the multiplication of two N × N matrices. An array is semi-systolic if the output data is not produced in the boundary cells of the array (type 1) or if the input data has to be preloaded into the array (type 2).

The type 1 array and the type 2 array and their cell structure are shown for N=4 on fig.(2) for matrix multiplication C=AB. Where H_{in}, H_{out}, V_{in}, and V_{out} represent the horizontal input, the horizontal output, the vertical input, and the vertical output respectively. R_{ij} is a value saved in a register of each semi-systolic cell.

In the type 1 array shown in fig. (2 a). Each semi-systolic cell perform a multiply-accumulator operation. Where the horizontal and vertical inputs are are multiplied and added to the register in semi-systolic cell. In N cycles, each semi-systolic cell computes an inner product of row of A and column of B. Here type 1 computation is done as follows from type 1 array in fig. (2 a):

$$H_{out} \leftarrow H_{in}$$

$$V_{out} \leftarrow V_{in}$$

$$R_{ij} \leftarrow R_{ij} + H_{in}V_{in}$$

The computational sequence of this array is shown in table (1 a), where $x \leftarrow y$ indicates that the value of x is replaced by the value of y. this array is semi-systolic since the output data is produced throughout the array and not at in the boundary cells. This requires overhead for the output to be shifted out of the array.

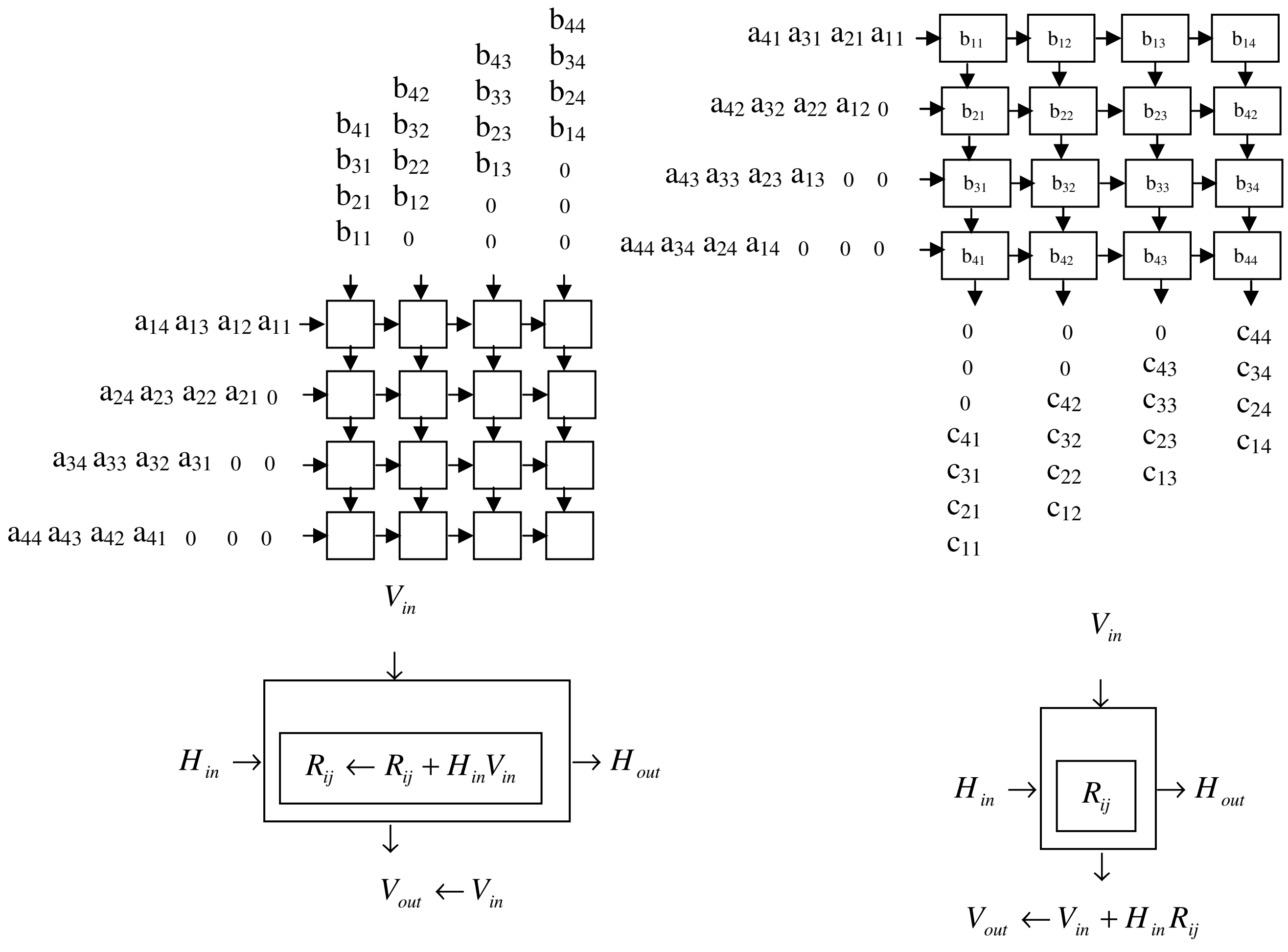
In the type 2 semi-systolic array shown in fig. (2 b), to compute C=AB, the components of matrix B are preloaded into register in the semi-systolic cells, and matrix A stream into the array. Each cell multiplies the horizontal input times the register value and adds this to the vertical input to produce the vertical output. The inner product of a row of input matrix A and a column of the array (a column of matrix B) is computed in N cycles. Type 2 semi-systolic computation is defined as follows from the type 2 array in fig. (2 b):

$$H_{out} \leftarrow H_{in}$$

$$V_{out} \leftarrow V_{in} + R_{ij}H_{in}$$

The computational sequence of this array is given in table (1 b). The value of C_{ij}(i=1, 2, 3, 4) is generated by semi-systolic cell (4, j) for j=1, 2, 3, 4.

Inputs for type 1 matrix multiplication array are stream inputs, whereas one of the inputs of the type 2 matrix multiplication array is preloaded input. On the other hand, in the type 1 matrix multiplication array, each element of matrix C is saved in a semi-systolic cell, where as in type 2 matrix multiplication array, each element of matrix C is generated in the bottom boundary semi-systolic cell's of the array as stream outputs.


b_{44} b_{34} b_{24} b_{14} 0 0 0
b_{43} b_{33} b_{23} b_{13} 0 0
b_{42} b_{32} b_{22} b_{12} 0
b_{41} b_{31} b_{21} b_{11}
a_{14} a_{13} a_{12} a_{11}
a_{24} a_{23} a_{22} a_{21} 0
a_{34} a_{33} a_{32} a_{31} 0 0
a_{44} a_{43} a_{42} a_{41} 0 0 0
V_{in}
H_{in}
$R_{ij} \leftarrow R_{ij} + H_{in}V_{in}$
H_{out}
$V_{out} \leftarrow V_{in}$
a_{41} a_{31} a_{21} a_{11}
a_{42} a_{32} a_{22} a_{12} 0
a_{43} a_{33} a_{23} a_{13} 0 0
a_{44} a_{34} a_{24} a_{14} 0 0 0
b_{11} b_{12} b_{13} b_{14}
b_{21} b_{22} b_{23} b_{42}
b_{31} b_{32} b_{33} b_{34}
b_{41} b_{42} b_{43} b_{44}
0 0 0 c_{41} c_{31} c_{21} c_{11}
0 0 c_{42} c_{32} c_{22} c_{12}
0 c_{43} c_{33} c_{23} c_{13}
c_{44} c_{34} c_{24} c_{14}
V_{in}
H_{in}
R_{ij}
H_{out}
$V_{out} \leftarrow V_{in} + H_{in}R_{ij}$

(a) Type 1 array

Fig. (2): Semi-systolic arrays for matrix multiplication.

(b) Type 2 array.

Table 1

Coputational sequences for type 1 and type 2 semi-systolic arrays. (a) for the type 1 semi-systolic array. (b) for the type 2 semi-systolic array.

	cell(1,1)	cell(2,1)	cell(3,1)	cell(4,1)
1	$R_{11} \leftarrow a_{11}b_{11}$			
2	$R_{11} \leftarrow R_{11} + a_{12}b_{21}$	$R_{12} \leftarrow a_{21}b_{11}$		
3	$R_{11} \leftarrow R_{11} + a_{13}b_{31}$	$R_{21} \leftarrow R_{21} + a_{22}b_{21}$	$R_{31} \leftarrow a_{31}b_{11}$	
4	$R_{11} \leftarrow R_{11} + a_{14}b_{41}$	$R_{21} \leftarrow R_{21} + a_{23}b_{31}$	$R_{31} \leftarrow R_{31} + a_{32}b_{21}$	$R_{41} \leftarrow a_{41}b_{11}$
5		$R_{21} \leftarrow R_{21} + a_{24}b_{41}$	$R_{31} \leftarrow R_{31} + a_{33}b_{31}$	$R_{14} \leftarrow R_{41} + a_{42}b_{21}$
6			$R_{31} \leftarrow R_{31} + a_{34}b_{41}$	$R_{14} \leftarrow R_{41} + a_{43}b_{31}$
7				$R_{14} \leftarrow R_{41} + a_{44}b_{41}$

(b)

	cell(1,1)	cell(2,1)	cell(3,1)	cell(4,1)
1	$V_{out} \leftarrow a_{11}b_{11}$			
2	$V_{out} \leftarrow a_{21}b_{11}$	$V_{out} \leftarrow V_{in} + a_{12}b_{21}$		
3	$V_{out} \leftarrow a_{31}b_{11}$	$V_{out} \leftarrow V_{in} + a_{22}b_{21}$	$V_{out} \leftarrow V_{in} + a_{13}b_{31}$	
4	$V_{out} \leftarrow a_{41}b_{11}$	$V_{out} \leftarrow V_{in} + a_{32}b_{21}$	$V_{out} \leftarrow V_{in} + a_{23}b_{31}$	$V_{out} \leftarrow V_{in} + a_{14}b_{41}$
5		$V_{out} \leftarrow V_{in} + a_{42}b_{21}$	$V_{out} \leftarrow V_{in} + a_{33}b_{31}$	$V_{out} \leftarrow V_{in} + a_{24}b_{41}$
6			$V_{out} \leftarrow V_{in} + a_{43}b_{31}$	$V_{out} \leftarrow V_{in} + a_{34}b_{41}$
7				$V_{out} \leftarrow V_{in} + a_{44}b_{41}$

COMPUTATION OF DWT USING SYSTOLIC ARRAY:

The 1-D DWT can be implement by using type 2 semi-systolic array. Where Rij will content the elements of the transformation matrix T after transposition (Which is either Haar, or Db4 transformation matrix). X is the input signal. As shown in fig. (3). Which implement equation 1. Where Fig. (3 a) implement this equation by using the Haar transformation. While Fig. (3 b) implement this equation by using the Db4 transformation.

$$\begin{bmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \end{bmatrix} = \begin{bmatrix} h_0 & h_1 & 0 & 0 \\ 0 & 0 & h_0 & h_1 \\ h_1 & -h_0 & 0 & 0 \\ 0 & 0 & h_1 & -h_0 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} X_0h_0 + X_1h_1 \\ X_2h_0 + X_3h_1 \\ X_0h_1 - X_1h_0 \\ X_2h_1 - X_3h_0 \end{bmatrix}$$

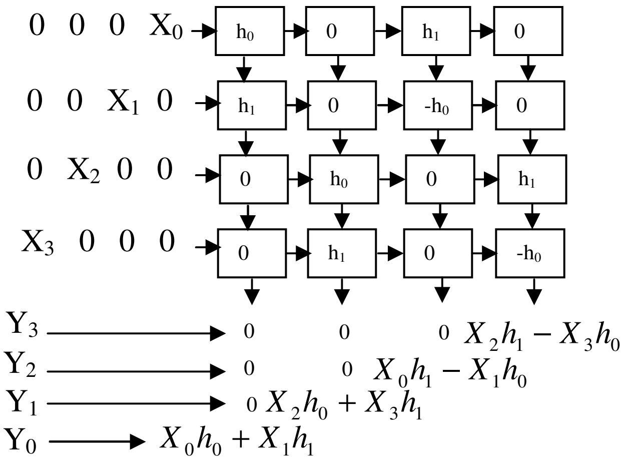


Fig. (3, a): Implementation of 1-D DWT using Haar.

$$\begin{bmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \end{bmatrix} = \begin{bmatrix} h_0 & h_1 & h_2 & h_3 \\ h_2 & h_3 & h_0 & h_1 \\ h_3 & -h_2 & h_1 & -h_0 \\ h_1 & -h_0 & h_3 & -h_2 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} X_0h_0 + X_1h_1 + X_2h_2 + X_3h_3 \\ X_0h_2 + X_1h_3 + X_2h_0 + X_3h_1 \\ X_0h_3 - X_1h_2 + X_2h_1 - X_3h_0 \\ X_0h_1 - X_1h_0 + X_2h_3 - X_3h_2 \end{bmatrix}$$

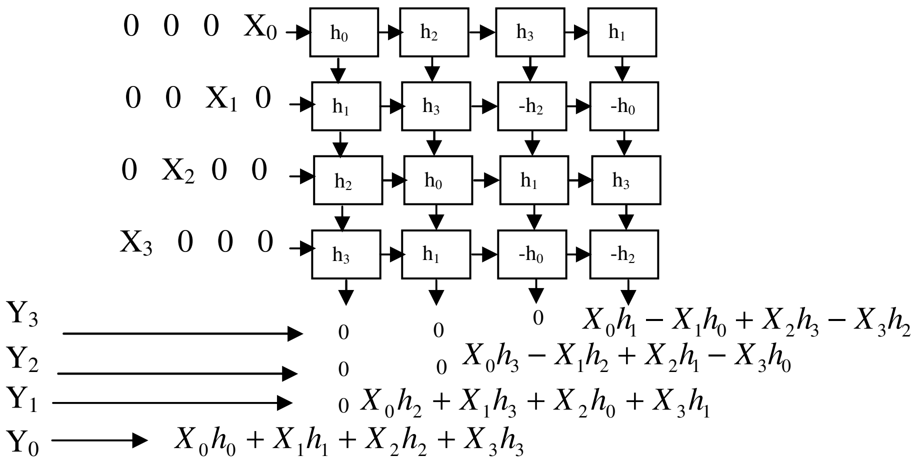

0 0 0 X0
0 0 X1 0
0 X2 0 0
X3 0 0 0
h0
h2
h3
h1
h1
h3
-h2
-h0
h2
h0
h1
h3
h3
h1
-h0
-h2
Y3
0 0 0 $X_0h_1 - X_1h_0 + X_2h_3 - X_3h_2$
Y2
0 0 $X_0h_3 - X_1h_2 + X_2h_1 - X_3h_0$
Y1
0 $X_0h_2 + X_1h_3 + X_2h_0 + X_3h_1$
Y0
$X_0h_0 + X_1h_1 + X_2h_2 + X_3h_3$

Fig. (3, b): The implementation of 1-D DWT using Db4.

Fig. (3): Implementation of 1-D DWT using type 2 semi-systolic array.

Equation 2, which represent the 2-D DWT can be implemented by using type 1, and type 2 semi-systolic array. As shown in fig. (4), which represent a fully systolic array. Where first a type 1 semi-systolic array is used to implement the first part of equation 2 (Z=T.X), and store it inside it. Then type 2 is used to implement the second part of equation 2 ($Y=Z.T^T$), but with some change as shown below:

$$H_{out} = H_{in} + V_{in} R_{ij} \quad \text{instead of} \quad V_{out} = V_{in} + H_{in} R_{ij}$$

$$V_{out} = V_{in} \quad \text{instead of} \quad H_{out} = H_{in}$$

Note that due the rearrangement of T in type 2, the result obtained in reverse order. The overall fully systolic array is shown in fig. (4 c).

$$R_{ij} \leftarrow R_{ij} + H_{in} V_{in}$$
$$V_{out} \leftarrow V_{in}$$
$$H_{out} \leftarrow H_{in}$$

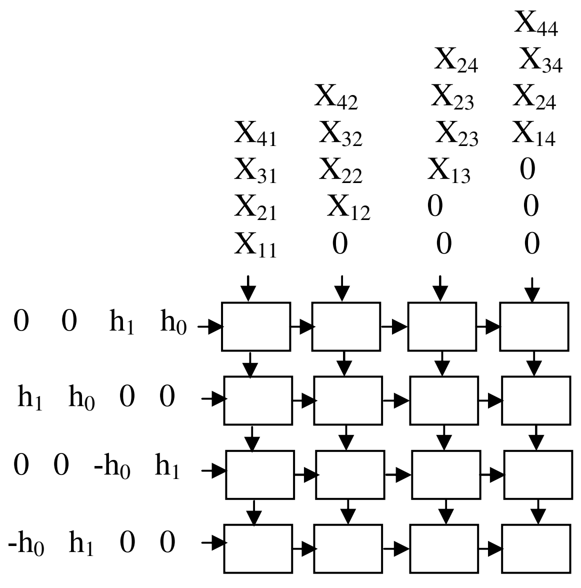


Fig. (4, a): Represent type 1 semi-systolic array to implement Z=T.X

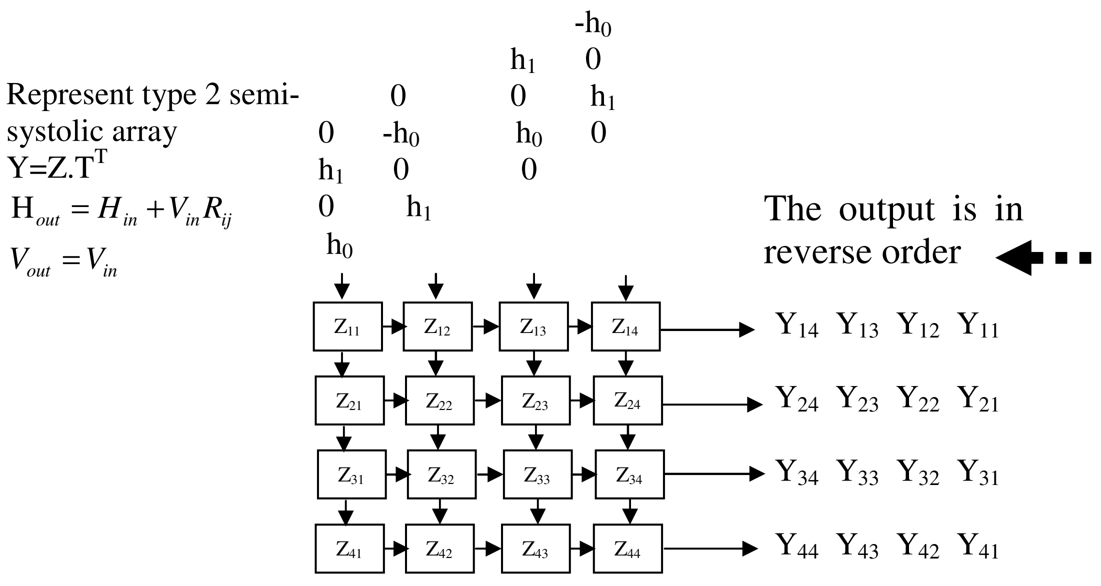


Fig. (4, b): Represents type 2 semi-systolic array to implement $Y=Z.T^T$.

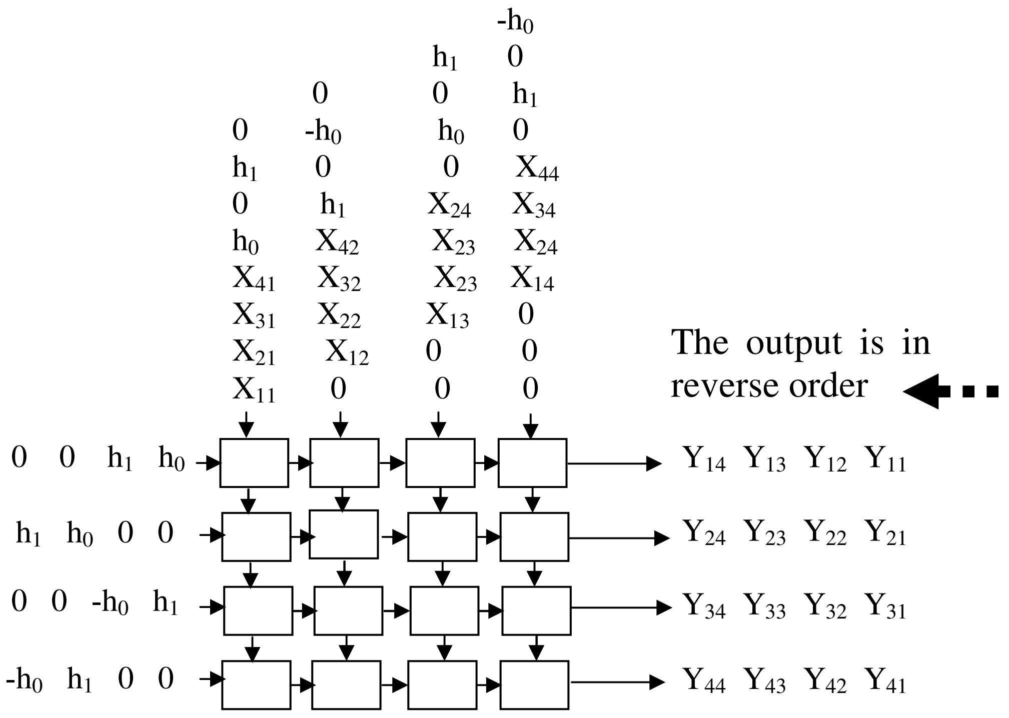

0 h1 0 h0 X41 X31 X21 X11
0 -h0 0 h1 X42 X32 X22 X12 0
h1 0 h0 0 X24 X23 X23 X13 0 0
-h0 0 h1 0 X44 X34 X24 X14 0 0 0
0 0 h1 h0
h1 h0 0 0
0 0 -h0 h1
-h0 h1 0 0
Y14 Y13 Y12 Y11
Y24 Y23 Y22 Y21
Y34 Y33 Y32 Y31
Y44 Y43 Y42 Y41
The output is in reverse order

Fig. (4, c): The overall fully systolic array, to implement $Y=T.X.T^T$.

Fig. (4): The implementation of 2-D DWT using systolic array with Haar Transformation.

CONCLUSION

This paper presents a systolic array realization of the 1-D DWT, and 2-D DWT. The 1-D needs only a semi-systolic array for its realization, which need a preloading of input data. While the 2-D use a regular array of simple processors that operate in the two semi-systolic manners to produce a fully systolic system. The resulting systolic array accepts stream of input data without any preloading.

This open the way for further study of image and speech processing as well as in other DSP applications that based on Wavelet transform.

REFERNCES

[1] C. S. Burrus, R. A. Gopinath, and H. Guo, “Introduction to Wavelets and Wavelet Transforms”, Prentice hall, 1998.

[2] E. E. Swartzlander, Jr., "Systolic FFT processors", in Systolic Arrays, W. Moore, A. McCabe, and R. Urquhart, Eds. Boston, MA: Adam Hilger, 1987, pp. 133-140.

[3] J. C. Goswami and A. K. Chan, “ Fundamentals of Wavelets: Theory Algorithms, and Applications”, John Wiley & Sons Inc., 1999.

[4] J. O'Brien, J. Mather, an B. Holland, "A 200 MIPS single-chip 1K FFT processor," in IEEE Inst. Solid-State Circuits Conf. Rec., 1989, pp. 166-167.

[5] M. H. Lee, "High speed multidimentional systolic arrays for discrete Fourier transform", IEEE Trans. Circuits Syst. II, vol. 39, pp. 876-879, 1992.

[6] M. B. Martin, “Applications of Multiwavelets to Image Compression”, M.Sc. thesis, Virginia Polytechnic Institute and State University, Virginia, June, 1999.

[7] S. Y. Kung, VLSI Array Processors. Englewood Cliffs, NJ: Prentice-Hall, 1988.

[8] V. Strela, “Multiwavelets: Theory and Applications” Ph.D thesis, Massachusetts Institute of Technology, 1996.