Arab J Sci Eng (2014) 39:6125–6135
DOI 10.1007/s13369-014-1208-2

RESEARCH ARTICLE - COMPUTER ENGINEERING AND COMPUTER SCIENCE

High-Speed Systolic VLSI Architecture for 2-D Forward Lifting-Based DWT

N. Usha Bhanu · A. Chilambuchelvan

Received: 30 March 2013 / Accepted: 15 July 2013 / Published online: 8 June 2014

Abstract In this paper, an efficient approach to design a 2-D systolic array for high-speed implementation of block-based lifting lossy 9/7 wavelet filter is proposed. The inherent advantage of the in place computation of the lifting-based discrete wavelet transform over the conventional convolution method makes it suitable for efficient hardware implementation with lower computational complexity. The row processor consists of processing elements arranged in a systolic manner, and for column processing, the lifting steps are computed concurrently, by mapping the coefficients to the same systolic arrays, using the cyclic symmetry property of the block of input image coefficients. The advantage of the discussed architecture is that it does not require additional memory for storing the intermediate coefficients. The functionality of the processing element in the systolic array improves the speed, by having the critical path delay of one multiplier and two adders computational time. The area efficient, high-speed design of the lifting algorithm is coded in hardware description language and implemented in Altera cyclone II field programmable gate arrays (FPGA). The implemented results show that the systolic architecture achieves a high speed of 260 MHz with a lower accessing time of 0.246 μs, when compared to other existing architectures, and reaches a speed performance suitable for real-time multimedia applications. This conceptual design of systolic arrays can be used as IP core in FPGA-based reconfigurable coprocessors.

Keywords Lifting-based discrete wavelet transform · Systolic arrays · Block processing · Field programmable gate arrays (FPGA)

الخلاصة

تم في هذه الورقة العلمية-عرض طريقة فعالة لتصميم مصفوفة انقباضية ثنائية الأبعاد لمرشح موجات رافع للفقد 9\7 قائم على كتلة تنفيذ عالية السرعة. إن الميزة الأصيلة المتمثلة في الحساب في الموقع لتحويل الموجات المتقطع القائم على الرفع مقارنة بطريقة الالتواء التقليدية تجعلها مناسبة لتضمين المكونات المادية بشكل فعال وبتعقيدات حسابية أقل. يتكون صف المعالج من عناصر معالجة منظمة بطريقة انقباضية، وتُحسب خطوات الرفع لمعالجة العمود بشكل متزامن عن طريق تعيين المعاملات لنفس المصفوفات الانقباضية، ويتم ذلك باستخدام خاصية التماثل الدوري لكتلة معاملات الصورة المدخلة. إن الميزة للتصميم الذي تمت مناقشته هو أنه لا يتطلب ذاكرة إضافية لتخزين المعاملات الوسيطة. وكذلك يحسن الأداء الوظيفي لعنصر المعالجة في المصفوفة الانقباضية من السرعة من خلال وجود تأخير المسار الحرج لمضاعف واحد والوقت الحسابي لمضيفين اثنين. ويتم ترميز تصميم عالي السرعة ذي منطقة فعالة لخوارزمية الرفع بلغة وصف المكونات المادية، وتم تنفيذه باستخدام مصفوفة البوابات المنطقية القابلة للبرمجة (Altera Cyclone II FPGA). وقد أظهرت نتائج التنفيذ أن التصميم الانقباضي يحقق سرعة عالية مقدارها 260 ميغاهيرتز مع وقت وصول أقل مقداره 0.246 ميكروثانية مقارنة بتصاميم أخرى موجودة، وكذلك يصل إلى سرعة أداء مناسبة لتطبيقات الوسائط المتعددة ذات الزمن الحقيقي. وكذلك يمكن استخدام تصميم المفاهيم هذا الخاص بالمصفوفات الانقباضية كجوهر بروتوكول إنترنت في المعالجات المشتركة القابلة لإعادة التشكيل والقائمة على مصفوفة البوابات المنطقية القابلة للبرمجة.

1 Introduction

The advantages of the discrete wavelet transform over other transforms make it suitable for image compression JPEG2000 standards [1]. It overcomes the blocking artifacts

N. Usha Bhanu (✉)
Department of Electronics and Communication Engineering,
Saveetha School of Engineering, Saveetha University,
Chennai, 602105 Tamil Nadu, India
e-mail: bhanu.usha@rediffmail.com

A. Chilambuchelvan
Department of Electronics and Instrumentation Engineering,
R.M.D.Engineering College, Anna University,
Chennai, 601206 Tamil Nadu, India
e-mail: Chill97@gmail.com

problems in the DCT, using the sub-band decomposition [2] technique and achieves a higher compression ratio with multiresolution capability. The convolutional method of implementing the DWT has its own limitations in real-time image /video applications, such as computational complexity and memory for storing the image coefficients.

The lifting scheme-based wavelet transform was proposed by Daubechies and Sweldens [3] based on a spatial construction of the second-generation wavelet. The lifting scheme has many advantages over the convolutional DWT. All the interesting properties of the wavelets, such as biorthogonality and regularity, are defined by the linear relationships between the filter bank coefficients. The lifting scheme does not depend on the Fourier transform of the wavelets. Therefore, the wavelets can be designed on arbitrary lattices in the spatial domain and used as a multiresolution analysis tool in signal/image, texture classification [4], and in compression for real-time multimedia applications.

Various efficient VLSI architectures were proposed to meet the real-time implementation of the DWT. Parhi and Nishitani [5] proposed folded, digit serial architectures for the implementation of 1-D and 2-D discrete wavelet transforms. Denk and Parhi [6] presented a systolic mapping techniques and dependence graph to derive the DWT architectures. Wu and Chen [7] proposed a polyphase decomposition technique and the coefficient-folding technique of decimation filters for the implementation of the 2-D DWT. Andra et al. [8] presented a lifting-based forward and inverse wavelet transform for the set of seven filters proposed in the JPEG2000. Dillen et al. [9] proposed a combined line-based architecture for the 5/3 and 9/7 wavelet transform of JPEG2000, using lifting schemes by incorporating pipelining for optimization. Shi et al. [10] presented an efficient folded architecture (EFA) with lower computational complexity. The critical path delay affects the performance of this architecture. Huang et al. [11] proposed a flipping structure for lifting the DWT using five-stage pipelining, where the critical path is reduced to one multiplier delay, but it requires a large temporal buffer. Usha and Chilambuchelvan [12] proposed polyphase decomposition and the coefficient-folding technique for the implementation of convolutional DWT. Xiong et al. [13] presented an efficient architecture by exploiting the embedded decimation technique, working in parallel and pipeline way without using multipliers. Cheng and Parhi [14] proposed a systematic high-speed VLSI implementation of the DWT, based on hardware-efficient parallel FIR filter structures. Tian et al. [15] presented an efficient multi-input/multi-output VLSI architecture (MIMOA) with computing time as low as N^2/M for an $N \times N$ image, with controlled increase in hardware cost, where 'M' is the throughput rate. Kilic and Yilmaz [16] presented a coding scheme in combination with hierarchical vector quantization, and zero tree wavelet to encode the quantized wavelet coefficients for the MPEG-4 video encoder.

The survey of various architectures using lifting schemes have been presented in [17–19], taking into consideration the required numbers of multipliers, adders and registers, and the amount of memory required for their implementation. The data movement and transfer play a key role in determining the efficiency of VLSI implementation of lifting-based DWT. The systolic arrays have regular computational structures [20] having less complex data routing and control for efficient VLSI implementation, when the length of the filter is large. Systolic arrays [21–25] represent an appropriate architectural design for constructing wavelet-lifting schemes. Alwan [26] proposed the numerical computation of sinusoidal sequences from truncated trigonometric series, in a systolic manner to improve the speed, in parallel over conventional Von Neumann architectures. Mohanty and Meher [27] proposed hardware-efficient systolic-like modular design using a new data-access scheme and a novel folding technique for the implementation of the DWT. Mohanty et al. [28] proposed block processing and systolic array for lifting-based 2-D DWT.

The motivation of this work is to propose a high-speed systolic VLSI architecture for the given image coefficients for the efficient implementation of lifting-based DWT. The lifting algorithm for 1-D and 2-D systolic arrays is coded in VHDL, and the implementation results in Altera cyclone II FPGA show that the hardware implementation is suitable for resource constrained high-speed embedded multimedia applications.

The rest of the paper is structured as follows: Sect. 2 reviews the basics and mathematics of the lifting algorithm involved in deriving the systolic array for the implementation of the Cohen Daubechies Feauveau 9/7 lossy wavelet filters. In Sect. 3, the construction of 1-D and 2-D systolic arrays, for effective hardware utilization, with their flow diagram is given. Section 4 provides the implementation results of the proposed systolic architecture and its comparison with the previous existing architectures in FPGA. Finally, in Sect. 5, the concluding remarks and the possibility of extending this work in future are made.

2 Lifting Scheme for Discrete Wavelet Transform

2.1 Basics of Lifting Scheme

The lifting scheme has been developed as a flexible tool suitable for constructing second-generation wavelets. It is composed of three basic operation stages: split, predict and update. Figure 1 shows the lifting scheme of the wavelet filter for computing a 1-D signal. The three basic steps in lifting-based DWT are:

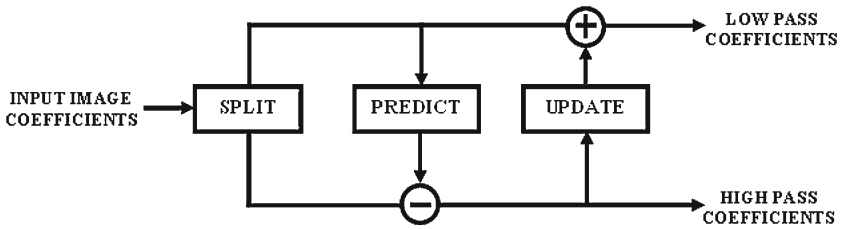

Fig. 1 Block diagram of the forward lifting scheme

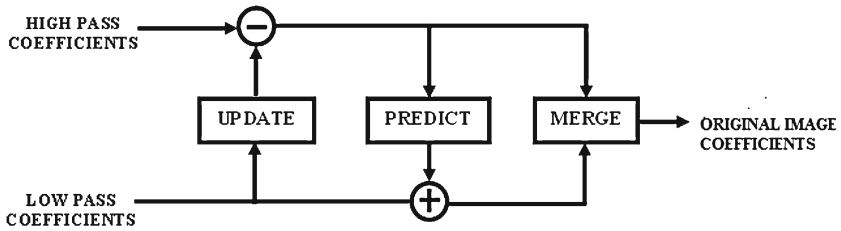

Fig. 2 Block diagram of the inverse lifting scheme

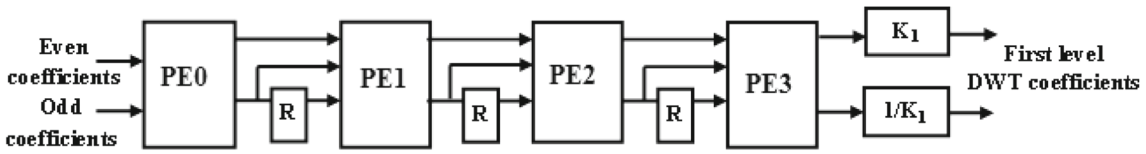

Fig. 3 Representation of the systolic array for 9/7 lifting

Split Step: where the signal is split into even and odd points, because the maximum correlation between adjacent pixels can be utilized for the next predict step. For each pair of given input, samples $x\,(n)$ split into even $x\,(2n)$ and odd coefficients $x\,(2n+1)$.

Predict Step: The even samples are multiplied by the predict factor, and then, the results are added to the odd samples to generate the detailed coefficients (*dj*). Detailed coefficients result in high-pass filtering.

Update Step: The detailed coefficients computed by the predict step are multiplied by the update factors, and the results are added to the even samples to get the coarse coefficients (*sj*). The coarse coefficients give a low-pass filtered output.

The inverse transform could easily be found by exchanging the sign of the predict step and the update step, and applying all the operations in the reverse order, as shown in Fig. 2. The implementation of the lifting-based inverse transform (IDWT) is simple and involves the order of operations in the DWT to be reversed. Hence, the same resources can be reused to define a general programmable architecture for forward and inverse DWT.

2.2 Mathematics in Lifting DWT for the Implementation of the Systolic Array

The JPEG2000 9/7 lossy standard for DWT consists of four lifting steps. The constants a, b, c and d are called the lifting coefficients; K_1 and $1/K_1$ are called the scaling normalization coefficients. The basics of the lifting steps are used for the representation of the systolic array with four processing elements, connected as shown in Fig. 3. Each of the processing elements is connected to the next element with a register delay. The samples Xo (n) and Xe (n) represent the odd and even input image coefficients.

The processing element PE0 computes the predict stage P1 given by,

$$D(n) = \mathrm{Xo}(n) + a[\mathrm{Xe}(n) + \mathrm{Xe}(n+1)] \tag{1}$$

PE1 computes the update stage U1 given by,

$$S(n) = \mathrm{Xe}(n) + b\,[D(n-1) + D(n)] \tag{2}$$

PE2 computes the predict stage P2 resulting in high-pass filtering and is given by,

$$\mathrm{HP}(n) = D(n) + c\,[S(n) + S(n-1)] \tag{3}$$

PE3 computes the update stage U2 resulting in desired low-pass filtering,

$$\mathrm{LP}(n) = S(n) + d[\mathrm{HP}(n-1) + \mathrm{HP}(n)] \tag{4}$$

After N lifting steps, the scaling coefficients of $K1$ and $\frac{1}{K1}$ are applied to the horizontal and vertical coefficients of steps (3) and (4) to obtain the low-pass and high-pass coefficients and is given by,

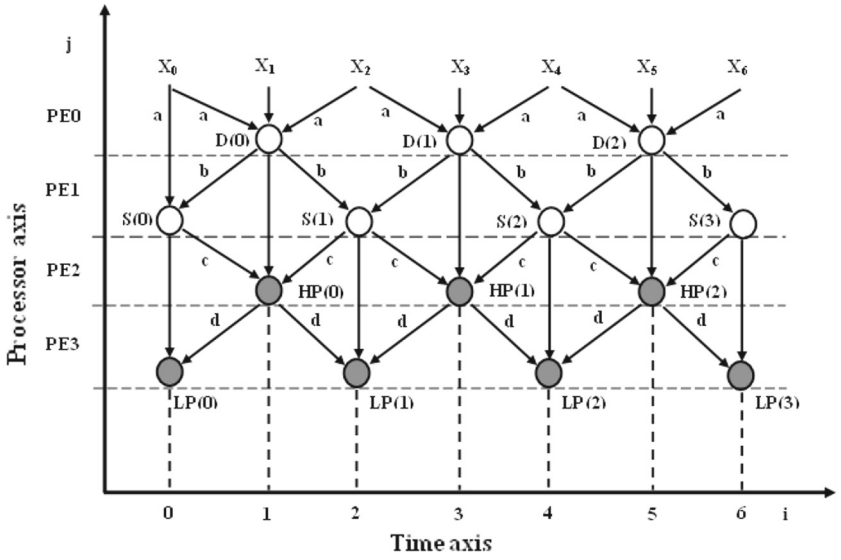

Fig. 4 Processor space time representation of the 9/7 lifting systolic array

$$\mathrm{HP}(n) = K1\mathrm{HP}(n) \tag{5}$$

$$\mathrm{LP}(n) = \frac{1}{K1}\mathrm{LP}(n) \tag{6}$$

This scaling is done for normalizing the magnitude. The values of the constants are $a = -1.586134$, $b = 0.0529801185$, $c = 0.882911076$, $d = -0.443506852$ and $K_1 = 1.149604398$ [11].

The systolic array architecture discussed is B1 design with broadcast inputs, move results and weight stay in the processing element. The data dependencies, data operations and the control complexity used to derive 1-D and 2-D systolic arrays for 9/7 lifting filter are derived from the processor space time representation of the lifting algorithm, as shown in Fig. 4.

$X_0, X_1, X_2, \ldots, X_7$ represent original image block of coefficients, open circle represents intermediate coefficients and filled circle represents final high- and low-pass coefficients

The input image coefficients $(X_0, X_1, \ldots, X_N)$ are available at all the processors at the same time. The input data are being broadcast to the adjacent processors with a delay. The weighting coefficients a, b, c, d appear at the same spatial coordinates. The outputs appear in the processor in different space and time and move from one processing element to another, yielding high- and low-pass coefficients with a delay of four clock pulses. The architecture has a regular data flow with 100 % hardware utilization. The projection space vector (P^{T}), processor space vector (S^{T}), scheduling vector $(S^{\mathrm{T}}d)$ and Hardware Utilization Efficiency (HUE) for the proposed 9/7 lifting algorithm are:

$$d = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad P^{\mathrm{T}} = (0\ 1\ 0\ 0) \quad S^{\mathrm{T}} = (1\ 0\ 0\ 0)$$

(i) **Projection Space Vector** Two adjacent processing nodes are displaced by d or multiples of d and executed by the same processor. Any node with index $I^{\mathrm{T}} = (i, j)$ is mapped to the same processor.

$$P^{\mathrm{T}}I = (0\ 1\ 0\ 0)\begin{pmatrix} i \\ j \end{pmatrix} = j \tag{7}$$

Therefore, all nodes on a horizontal line are mapped to the same processor.

(ii) **Processor Space Vector** The scheduling of the processor with index $I^{\mathrm{T}} = (i, j)$ is executed at the time.

$$S^{\mathrm{T}}I = (1\ 0\ 0\ 0)\begin{pmatrix} i \\ j \end{pmatrix} = i \tag{8}$$

(iii) **Scheduling vector** Any node with index I would be executed at time, $S^{\mathrm{T}}d$

$$S^{\mathrm{T}}d = (1\ 0\ 0\ 0)\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = 1 \tag{9}$$

(iv) **Hardware Utilization Efficiency (HUE) = $\mathbf{S^{T}d}$,**

$$\frac{1}{S^{\mathrm{T}}d} = 1 \tag{10}$$

To estimate the speed performance and parallelism involved in the computation of the lifting steps, the critical path is computed, which represents the longest path necessary for the coefficients to move from the input to the output, by considering the individual node in the processor space time scheduling, as shown in Fig. 5.

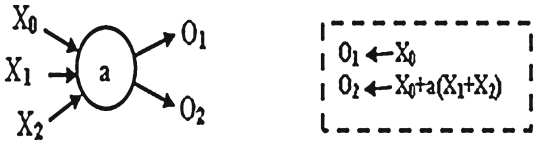

Fig. 5 Representation of the single processing node in the systolic array

Each of the processing elements (PE) consists of 2 adders, 1 register and 1 multiplier. The individual processing element processes two blocks of samples for every clock period. The delay in the execution of the processing module is $Tm+2Ta$, which represent the critical path delay for the architecture, where Tm and Ta are the execution time of the multipliers and adders, respectively.

3 Hardware Implementation of the Proposed Lifting Algorithm

3.1 One-Dimensional Systolic Array Architecture

The first-level representation of the systolic array is done, based on the lifting steps as discussed in lifting steps [1–4] in Sect. 2.2, and from the dependence graph shown in Fig. 4. The construction of the 1-D systolic processing array requires one multiplier, two adders and one delay register, as shown in Fig. 6. The architecture can be pipelined by adding the pipeline registers to speed up the execution time.

Each processing element performs two additions and one multiplication, as shown in Fig. 7. The lifting steps have similar computing steps. The 1-D systolic array architecture consists of L rows and each row consists of four processing elements, where L represents the required DWT levels. The pixel coefficients of an image are read from the external static random access memory. The image coefficients can be read either by line-wise or block-wise [29]. The advantage of the block-based scanning of the image coefficients is that the memory required for computing the intermediate coefficients is reduced.

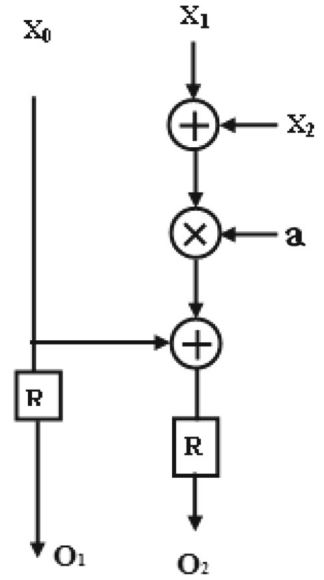

Fig. 7 Representation of the single processing element in the systolic array

In this paper, the image coefficients are read in a block or group of pixel coefficients and used for computing the lifting steps P1, U1, P2 and U2 as discussed in [1–4] of Sect. 2.2 are computed by the processing element, as shown in Fig. 8.

At each clock pulse, the processing element 1 takes a pair of odd and even coefficients and computes the predict stage (P1). This output is given to the next processing element to compute the update stage (U2) with a delay of one clock pulse. The processing element 3 receives the coefficients from PE2 to compute the predict stage (P2). The processing element 4 computes the fourth lifting step to compute the update stage U2. The output coefficients are scaled by a factor of $K1$ and $\frac{1}{K1}$ to give the final first-level high- and low-pass coefficients. A similar idea can be extended to the N dimensional input image coefficients, by properly schedul-

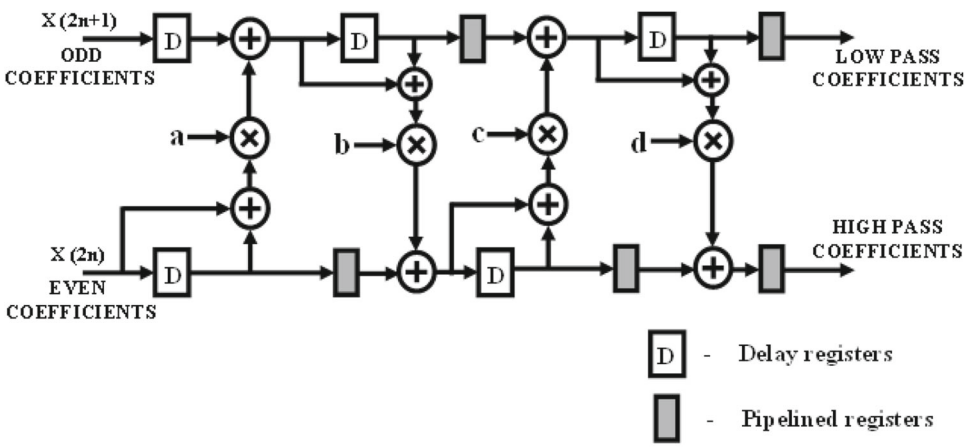

Fig. 6 Pipelined architecture for the 9/7 lifting DWT

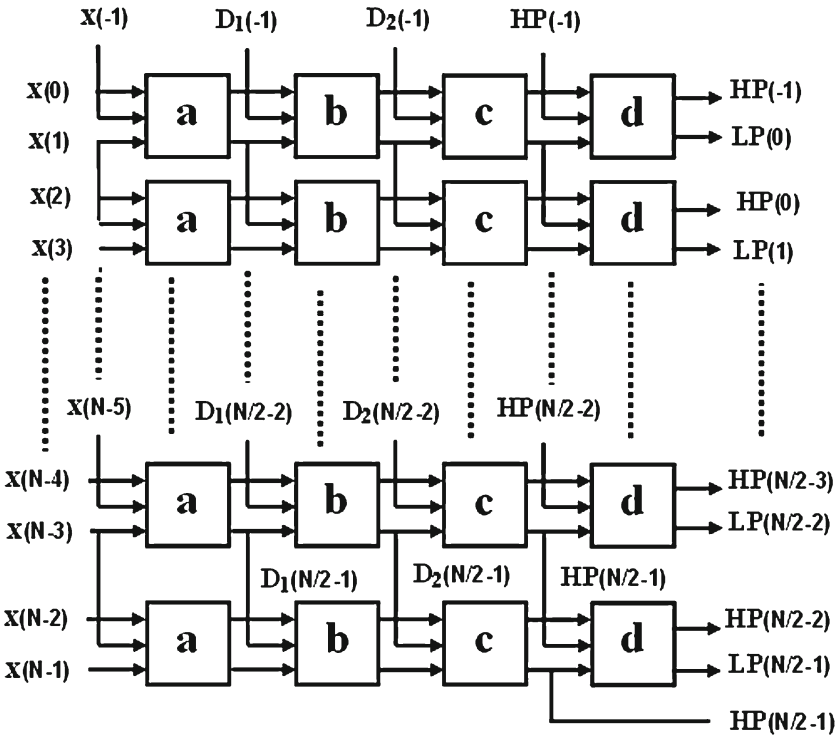

Fig. 8 Construction of 1-D systolic array

ing the input coefficients. The arrays of processing elements are arranged in such a way, as to achieve pipelining with efficient hardware utilization and high speed. After L level of decomposition of an N-dimensional image, the desired filter output is obtained.

3.2 Systolic Array Representation for the 2-D 9/7 Lifting DWT

The paper proposes an approach to construct a 2-D systolic array for lifting schemes, as shown in Fig. 9. The row processor consists of $M \times N/2$ processing elements arranged in a systolic manner. For column processing using the cyclic symmetry property of the input image coefficients, the last row of the systolic array for the given levels of decomposition is overlapped with the first row of the array of processing elements resulting in efficient hardware utilization.

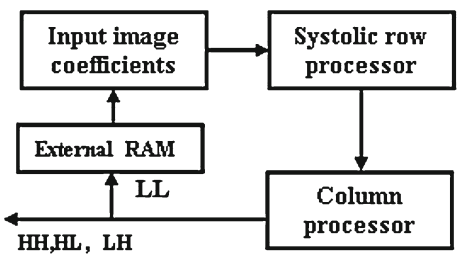

Fig. 9 Block diagram of the 2-D Systolic architecture

The 2-D systolic array architecture for the lifting DWT is shown in Fig. 10. The first block of image coefficients is given to the first row of processing elements. The systolic row processing is done for the input image coefficients; and it produces the intermediate matrices of $M \times N/2$; and they are stored in the intermediate buffer where M and N are dimensions of the image. Using these intermediate coefficients, column processing is done and the coefficients are mapped to each of the PEs in such a manner as to schedule the PEs to give a 2-D array. The cyclic symmetry property is used for the input block of image coefficients $\{x(0)_p, x(1)_p, x(2)_p, \ldots, x(N-1)_p, \ldots x(N-1)_p, x(N-2)_p \ldots\}$ where p is the given block size. The first term of the input image coefficients is overlapped with the last term of the image coefficients, in the same systolic array processing element. A 2-parallel structure of the architecture is derived for the image with a block size of $P = M \times N$ with $M = N$ with a factor of $N^2/2$. This is done to reduce the average computing time and the number of processing elements used for the systolic array.

For computing the jth level DWT, it requires $M*N/2^{2j-1}$ samples in every clock cycle where $1 \leq j \leq L$ where $L = [\log_4^{(M \times N)}]$. In each clock cycle, the PE receives N^2 input samples, and a row of first-level sub-bands is obtained with a gap of four clock cycles. Each row of matrix A^{j-1} is overlapped and fed to the PE_{-j} in $2^{j-1} * 4$ cycles.

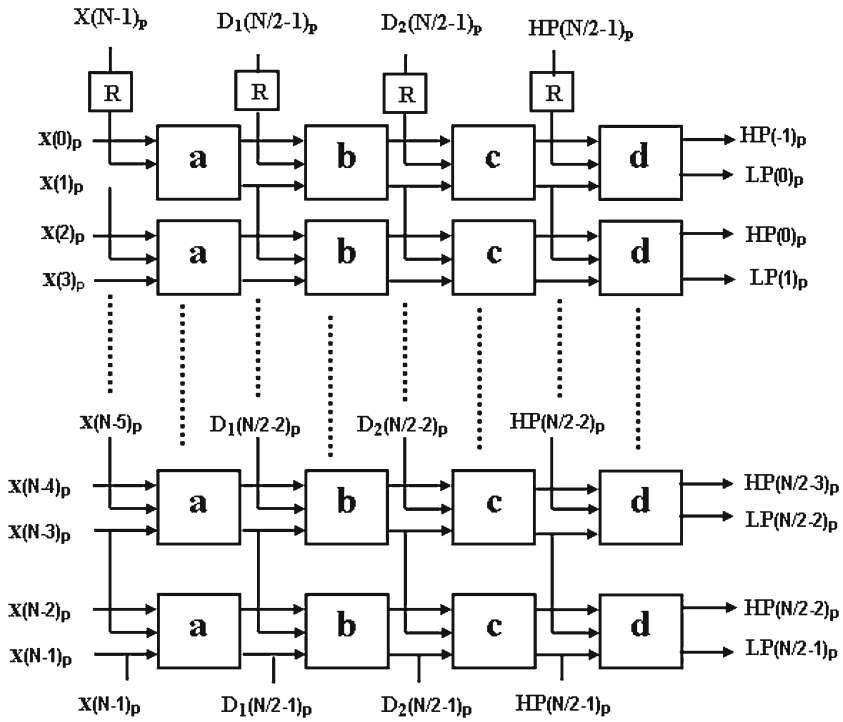

Fig. 10 Representation of the 2-D systolic array

Fig. 11 Sample image

Table 1 First level of decomposed coefficients for the given image

Approximation coefficients		Detailed coefficients	
LL	LH	HL	HH
171	6	98	−46
172	6	128	−47
161	8	184	−48
174	10	191	−64

3.3 Data-Access Scheme

The images to be analyzed are read from MATLAB and loaded into the systolic VHDL file. The image size is 128 × 128 as shown in Fig. 11. The architecture is generic for any square 2-D input image.

Considering a sample of 8 × 8 coefficients with a block size of $P = 4$, the block of coefficients is fed parallel to the systolic array. The corresponding sample image coefficients with $M = N = 8$ with the block size of $P = 4$ of the given image coefficients are shown in Fig. 12.

$$P1=\begin{bmatrix}108 & 107 & 102 & 105\\ 107 & 109 & 108 & 108\\ 107 & 109 & 108 & 109\\ 107 & 108 & 108 & 110\end{bmatrix}\quad P2=\begin{bmatrix}111 & 123 & 192 & 188\\ 126 & 135 & 151 & 157\\ 111 & 116 & 131 & 184\\ 109 & 106 & 129 & 197\end{bmatrix}\quad P3=\begin{bmatrix}108 & 107 & 107 & 108\\ 146 & 109 & 108 & 109\\ 205 & 156 & 113 & 108\\ 209 & 217 & 153 & 111\end{bmatrix}\quad P4=\begin{bmatrix}109 & 106 & 112 & 146\\ 110 & 107 & 106 & 108\\ 109 & 107 & 107 & 107\\ 110 & 108 & 105 & 106\end{bmatrix}$$

Fig. 12 Block of image coefficients for the sample image of 8 × 8 with block size, $P = 4$

Table 2 Data flow of the systolic array architecture

Clock cycle	0	1	2	3	4	5	6	7	8	9
Input data	X_{01}	X_{02}	X_{03}	X_{04}	X_{05}	X_{06}	X_{07}	X_{08}	X_{11}	X_{12}
Predict stage 1 (P1)	–	D (0)	–	D (1)	–	D (2)	–	D (3)	–	D (4)
Update stage 1 (U1)	–	S (0)	–	S (1)	–	S (2)	–	S (3)	–	S (4)
Predict stage 2 (P2)				HP (0)	HP (1)					
Update stage 2 (U2)				LP (0)	LP (1)					

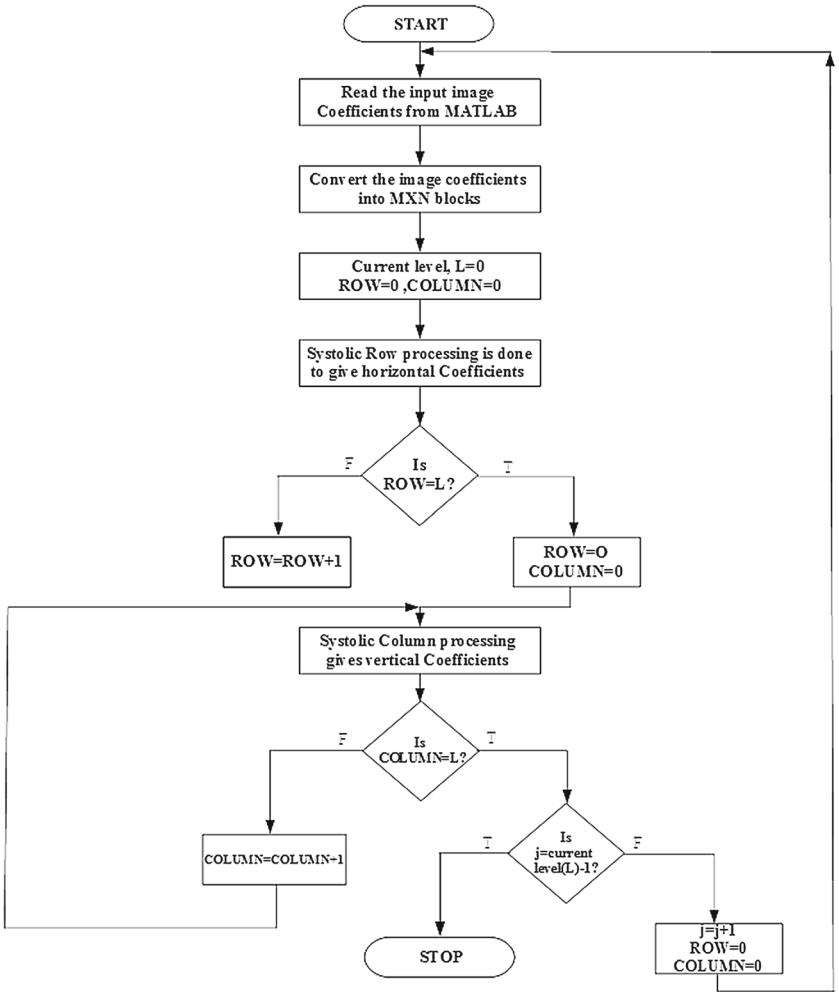

Fig. 13 Flowchart of the block-based systolic array architecture as implemented

L level decomposition has to be done for the given image of $N * N$ with $M = N$. The first-level works on all N^2 pixels of the original image, each successive level works on the approximation and detail coefficients produced by the previous level, and uses only one-fourth of the pixels of the previous level, resulting in fewer coefficients of the original image. Table 1 shows the first level of decomposed image coefficients, after passing through the processing elements in parallel for the blocks P1, P2, P3 and P4.

Each processing element in the systolic array requires two samples to compute one lifting step. Four pipeline stages are required for the efficient implementation of 9/7 lifting filters. From the data flow Table 2, it is clear that when the block of input image coefficients is loaded into the systolic processor, it requires four clock cycles to compute the first level of lifting coefficients. The flowchart for the block-based systolic array as implemented in this architecture is shown in Fig. 13. In this, the ROW represents the current row, and COLUMN represents the current column for the given j level

Table 3 Synthesis report of the systolic architecture

Synthesis report		
FPGA	Altera cyclone II EP2C35F672C6	
	One dimensional systolic array	Two dimensional systolic array
Total logic elements	248/33, 216 (<1 %)	31/33,216 (<1 %)
Total registers	200	200
Total pins	241/475 (50 %)	281/475 (59 %)
Total memory bits	0/483,840 (0 %)	0/483,840 (0 %)
Embedded multipliers	8/70 (11 %)	2/70 (3 %)
Clock set up period (ns)	4.398	3.846
Operating frequency (MHz)	227.38	260.01

of coefficients. The maximum level of decomposition is L of DWT.

4 Results and Comparison

4.1 Implementation Results

The performance analysis of the 1-D and 2-D systolic arrays is being coded in VHDL and is generic for any square input block of image coefficients. The architectures are implemented using EP2C35F672C6 cyclone II Altera FPGA. The performance of the proposed architecture is compared in terms of the number of multipliers, number of adders, storage size, computing time, control complexity and hardware utilization. The computing time has been normalized to the internal clocking rate. Every two input block of coefficients generate two output blocks of coefficients, with a delay of four clock cycles. From the results shown in Table 3, it is clear that the hardware implementation of the lifting schemes is suitable for real-time high-speed multimedia applications.

4.1.1 Comparison of Architectures

The comparison of the 2-D lifting systolic architecture is compared with other existing architectures. The parameters taken for comparison are hardware complexity (measured by the number of multipliers, adders and registers); the DWT scheme used, and operating frequency of the architectures. Table 4 indicates that the proposed design method provides high speed when compared to the other existing architectures.

4.1.2 FPGA Implementation

The synthesizable hardware description model, developed to implement the proposed systolic array architecture, is implemented for the given image coefficients with a block size of $P = 4$. The Quartus II IDE with Altera cyclone II EP2C35F672C6 FPGA hardware is used for the implementation. The input image coefficients are represented in integer data width to overcome the finite word length effects. To validate the results of the proposed method, the EFA [10] and multiple input multiple output architecture [15] are redesigned, with the same specifications of input conditions.

The FPGA implementation results, in terms of the dedicated logic registers, combinational LUTs, critical path and operating frequency achieved for the architectures, are presented in Table 5. From the implementation results, it is clear that the critical path of the folded architecture is long, because of the latency involved in the output coefficients reaching the input. The MIMOA architecture processes two input/two out-

Table 4 Comparison of the performance analysis of various architectures

Architectures	DWT scheme	Adders	Multipliers	Frequency (MHz)
Mohanty and Meher [27]	Lifting	8P	4.5P	112.892
Salehi and Amirfattahi [29]	Lifting	–	–	97
Tian et al. [15]	Lifting	8P	6P	64.25
Cheng and Parhi [14]	FIR	16	12	58.73
Meher et al. [25]	Convolution	4 $(K-1)$	4K	230.3
Proposed 2-D systolic architecture	Lifting	8 N	4 N	260.01

Table 5 FPGA implementation results of the various architectures

Architecture	Combinational LUT's	Dedicated logic registers	Critical path latency (ns)	Operating frequency (MHz)
Folded [10]	438	170	8.359	119.63
MIMOA [15]	235	235	5.763	173.52
Proposed systolic 1-D	248	200	4.398	227.38
Proposed systolic 2-D	248	200	3.846	260.06

puts per cycle, reducing the critical path and area required for the logic implementation. The proposed systolic array architecture achieves high speed due to the regular data flow scheduling and is suitable for high-speed multimedia applications.

4.1.3 Hardware and Timing Complexity of the Systolic Array

The systolic array representation for the first-level decomposition of an image consists of ten delay registers, four multipliers and eight adders. For N dimensional arrays, the hardware requirement is given below:

(i) **Number of adders**

$$4N + N/2 \times 8 = 8N$$

(ii) **Number of multipliers**

$$2N + N/2 \times 4 = 4N$$

For N element input image vector, the number of the computation level $L = \log_4 N$. At each level, computations are performed through multiply and accumulate units in a pipelined fashion. The pipeline delay exists in registers for calculating the L level DWT; then, for the delay, T_d, $L * T_d$ is the latency for the given input, to reach the output with the specified clock cycle. The processing time in the clock cycle for L-stage N dimensional DWT is $N + (L * T_d)$. The computation time required for the individual PE in the systolic array is given by;

$$\text{Computation time} = M \times N \times 1/L \times 1/F_{\max}$$

For 8×8 image; $T = 8 \times 8 \times 1/L \times 1/(260 \times 10^6) = 0.246\,\mu\text{s}$.

With $L = 1$ for the first stage, the speed can be increased with the added stages and is NT for the given stages. High speed can be achieved by suitably combining multiple levels of inputs, which is desirable for high-speed applications in real-time multimedia applications.

5 Conclusion

Systolic array architecture is used to achieve high degree of parallelism capable of handling multiple blocks of input image coefficients to implement the lifting DWT. The proposed architecture has regular and modular interconnections between the arrays of processing elements, with pipelining and parallel processing. The implemented results show that the systolic architecture achieves a high speed with a lower accessing time, when compared to the other existing architectures, and reaches a speed performance suitable for real-time multimedia applications. The advantage of the discussed architecture is that it does not require additional memory for storing the intermediate coefficients. The systolic architecture can be extended to varying block sizes, with an overlapped scanning of the image coefficients, thus reducing the accessing time and memory requirements, for storing the intermediate coefficients.

References

1. Christopoulos, C.; Skouras, A.; Ebrahimi, T.: The JPEG2000 still image coding system an overview. IEEE Trans. Consum. Electron. **46**, 1103–1127 (2000)
2. Mallat, S.: A theory for multiresolution signal decomposition: the wavelet representation. IEEE Trans. Pattern Anal. Mach. Intell. **11**, 674–693 (1989)
3. Daubechies, I.; Sweldens, W.: Factoring wavelet transform into lifting steps. J. Fourier Anal. Appl. **4**, 247–269 (1998)
4. Singer, A.: Colour texture classification using wavelet transform and neural networks ensembles. Arab. J. Sci. Eng. **34**(2B), 145–152 (2009)
5. Parhi, K.K.; Nishitani, T.: VLSI architectures for discrete wavelet transforms. IEEE Trans. VLSI Syst. **1**(2), 191–202 (1993)
6. Denk, T.C.; Parhi, K.K.: Systolic VLSI architectures for 1-D discrete wavelet transforms. In: Proceedings of Asilomar Conference on signals, systems and Computers, vol 2, pp 1220–1224. Pacific Grove, CA (1998)
7. Wu, P.-C.; Chen, L.-G.: An efficient architecture for two-dimensional discrete wavelet transform. IEEE Trans. Circuits Syst. Video Technol. **11**(4), 536–545 (2001)
8. Andra, K.; Chakrabarti, C.; Acharya, T.: A VLSI architecture for lifting-based forward and inverse wavelet transform. IEEE Trans. Signal Process. **50**(4), 966–977 (2002)
9. Dillen, G.; Georis, B.; Legat, J.-D.; Cantineau, O.: Combined line-based architecture for the 5/3 and 9/7 wavelet transform of JPEG2000. IEEE Trans. Circuits Syst. Video Technol. **13**(9), 944–950 (2003)
10. Shi, G.; Liu, W.; Zhang, L.: An efficient folded architecture for lifting based discrete wavelet transform. IEEE Trans. Circuits Syst II Express Briefs **56**(4), 290–294 (2009)
11. Huang, C.-T.; Tseng, P.-C.; Chen, L.-G.: Flipping structure—an efficient VLSI architecture for lifting-based discrete wavelet transform. IEEE Trans. Signal Process. **52**(4), 1080–1089 (2004)
12. Usha Bhanu, N.; Chilambuchelvan, A.: Efficient VLSI architecture for discrete wavelet transform. IJCSI Int. J. Comput. Sci. Issues, Special Issue, ICVCI-2011, **1**(1), 2011 ISSN (Online), 1694–0814 (2011)
13. Xiong, C.; Tian, J.; Liu, J.: Efficient architectures for Two dimensional wavelet transform using lifting scheme. IEEE Trans. Image Process. **16**(3), 607–614 (2007)
14. Cheng, C.; Parhi, K.K.: High-speed VLSI implementation of 2-D discrete wavelet transform. IEEE Trans. Signal Process. **56**(1), 393–403 (2008)
15. Tian, X.; Wu, L.; Tan, Y.-H.; Tian, J.-W.: Efficient multi-input/multi-output VLSI architecture for two-dimensional lifting-based DWT. IEEE Trans. Comput. **60**(8), 1207–1211 (2011)
16. Kilic, I.; Yilmaz; R.: A hybrid video compression based on zero tree wavelet structure. Arab. J. Sci. Eng. **34**(1B), 187–196 (2009)
17. Salehi, S.A.; Sadri, S.: Investigation of lifting-based hardware architectures for DWT. J. Circuits Syst. Signal Process. **28**, 41–55 (2009)
18. Acharya, T.; Chakrabarti, C.: A survey on lifting-based discrete wavelet transforms architectures. J. VLSI Signal Process. **42**, 321–339 (2006)

19. Usha Bhanu, N.; Chilambuchelvan, A.: A detailed survey on VLSI architectures for lifting based DWT for efficient hardware implementation. Int. J. VLSI Des. Commun. Syst. (VLSICS) **3**(2), 143–164 (2012)
20. Parhi, K.K.: VLSI Digital Signal Processing Systems: Design and Implementation. Wiley, New York (1999)
21. Kung, H.T.: Why systolic architectures. Comput. Mag. **15**, 37–45 (1982)
22. Pan, S.B.; Park, R.H.: Unified systolic array for computation of DCT/DST/DHT. IEEE Trans. Circuits Syst. Video Technol. **7**, 413–419 (1997)
23. Chiper, D.F.; Swamy, M.N.S.; Ahamad, O.; Stouraitis, T.: A systolic array architecture for discrete sine transforms. IEEE Trans. Signal Process. **50**(9), 2347–2354 (2002)
24. Meher, P.K.; Chandrasekaran, S.; Amira, A.: FPGA realization of FIR filters by efficient and flexible systolization using distributed arithmetic. IEEE Trans. Signal Process. **56**(7), 3009–3017 (2008)
25. Meher, P.K.; Mohanty, B.K.; Patra, J.C.: Hardware efficient systolic like modular design for 2D DWT. IEEE Trans. Circuits Syst. II Express Briefs **55**(2), 151–155 (2008)
26. Alwan, N.A.S.: Efficient systolic sinusoidal sequence generation. Arab. J. Sci. Eng. **33**(1B), 179–186 (2008)
27. Mohanty, B.K.; Meher, P.K.: Memory efficient modular VLSI architecture for high-throughput and low-latency implementation of multilevel lifting 2-D DWT. IEEE Trans. Signal Process. **59**(5), 2072–2084 (2011)
28. Mohanty, B.K.; Mahajan, A.; Meher, P..K.: Area and power efficient architecture for high throughput implementation of lifting 2D DWT'. IEEE Trans. Circuits Syst. II Express Brief. **59**(7), 434–438 (2012)
29. Salehi, S.A.; Amirfattahi, R.: A block based 2D discrete wavelet transform structure with new scan method for overlapped sections. In: Proceedings of the First Middle East Conference on Biomedical Engineering, pp 126–129. Sharjah (2011)