VHDL Simulation of DWT for Low Level Image Processing Application

M.Gayathri[1], E. Madhan Kumar[2]

PG Scholar[1], Assistant Professor[2],
St.Peter's University, TN, India

Abstract- Discrete wavelet transform (DWT) is the one of the main approaches used for image compression. A new discrete wavelet transform (DWT) architecture is proposed in this paper to realize a memory-efficient 2D DWT unit. The main goal of the proposed system is to change the processing unit in the pipelined DWT. Processing unit is modified in such a way that total number of processing required in the system will be reduced. The hardware is efficiently reduced by using the concept of intra-stage parallelism and inter-stage parallelism. The intra-stage parallelism is obtained by dividing the 2D filtering operation into four tasks. The multi decomposition levels in the stage of pipeline are mapped by computational task in inter-stage parallelism. To maintain the critical path delay serially concatenated additions are optimized by changing computation topology and applying arithmetic optimization. The Proposed architecture computes DWT efficiently with less clock cycles. The hardware complexity of 2D DWT is significantly reduced.

Keywords:discrete wavelet transforms, Computational parallelism, inter-stage parallelism, intra-stage parallelism, multi resolution filtering.

I INTRODUCTION

The multi-resolution decomposition approach is an effective approach to analyze the information present in the content of the image.2D discrete wavelet transform gives the multi-resolution decomposition capability [1]. The 2D discrete wavelet transform involves computation of large volumes of data and processing them in various decomposition levels is overhead. Earlier, many studies have been done to improve the performance of 2-D DWT computation which effectively utilizes the hardware resources. The architectures proposed earlier can be broadly classified into separable [2]–[16] and non-separable architectures [17]–[27]. Earlier can be broadly classified into separable [2]–[16] and non-separable architectures [17]–[27].

ICGPC 2014
St.Peter's University, TN, India.

A separable architecture is one where a 2-D filtering operation is divided into two 1-D filtering operations, one for processing the data row-wise and the other column-wise. A low-storage short-latency separable architecture where the row-wise operations are performed by systolic filters and the column-wise operations are performed in parallel filters has been proposed in [2]. This architecture requires complex control units to facilitate the interleaved operations of the output samples of different decomposition levels by employing a recursive pyramid algorithm (RPA) [3].

A scheme which leads to low complexity architecture with large latency have been proposed by Liao et al. [3] in which each of the row- and column-wise filtering operations are decomposed using the so called lifting operations [29] into a cascade of sub-filtering operations. As the 2D transforms are computed directly by using 2D filters in non-separable architectures, they do not have this problem. Two non-separable architectures based on a modified RPA have been proposed by Chakrabarti et al. [17]. One using parallel 2-D filters where high degree of computational parallelism is achieved at the expense of less efficient hardware utilization. Second using SIMD 2-D architecture requires a reconfigured organization of the array as the processing moves on to higher decomposition levels. Cheng et al. [18] have proposed an architecture which improves the processing speed at the expense of increased hardware by using a number of parallel FIR filters with a polyphase structure. Hung et al. [19] have proposed an architecture that is a pipeline of one stage of parallel multipliers and two stages of accumulators to perform the accumulation tasks of the filters in each of the two directions. This architecture provides a reduced count of multipliers and to facilitate the processing of the boundary data. The processing speed of this architecture is low as same architecture is utilized recursively to perform the tasks of successive decomposition levels. Marino [21] has proposed a two-stage pipeline architecture which provides short computation time where first stage performs the task of the first decomposition level and the second one that of all the remaining levels. The complexity of the hardware resources is high and design is complicated as the processing units employed in this architecture differ from one another.

Chengjun Zhang et.al [20] have proposed a scheme for the design of a high-speed pipeline VLSI architecture for the computation of the 2-D discrete wavelet transform (DWT). The main focus in the development of the architecture is on providing a high operating frequency and a small number of clock cycles along with efficient hardware utilization by maximizing the inter-stage and intra-stage computational parallelism for the pipeline. The inter-stage parallelism is enhanced by optimally mapping the computational task of multi decomposition levels to the stages of the pipeline and synchronizing their operations. The intra-stage parallelism is enhanced by dividing the 2-D filtering operation into four subtasks that can be performed independently in parallel and minimizing the delay of the critical path of bit-wise adder networks for performing the filtering operation

Most existing non-separable architectures aim at providing fast computation of the DWT by using pipeline structures and a large number of parallel filters.
However, these existing architectures have not exploited the computational parallelism inherent in the DWT operation to the extent possible in order to provide a high speed.

II Formulations for the Computation of the 2-D DWT

The 2-D DWT is an operation through which a 2-D signal is successively decomposed in a spatial multi resolution domain by low pass and high pass FIR filters along each of the two dimensions. The four FIR filters, denoted as high pass-high pass (HH), high pass-low pass (HL), low pass-high pass (LH) and low pass- Low pass (LL) filters, produce, respectively, the HH, HL, LH and LL sub band data of the decomposed signal at a given resolution level. The samples of the four sub bands of the decomposed signal at each level are decimated by a factor of two in each of the two dimensions. For the operation at the first level of decomposition, the given 2-D signal is used as input, whereas for the operations of the succeeding levels of decomposition, the decimated LL sub band signal from the previous decomposition level is used as input. Let a 2-D signal be represented by a matrix, with its element denoted by, where is chosen to be, being an integer. Let the coefficients of a 2-D FIR filter, be represented by a matrix. The coefficient of the filter P is denoted by. The decomposition at a given level can be expressed as

$$A^{(j)}(m,n) = \sum_{k=0}^{L-1}\sum_{i=0}^{M-1} H^{(\mathrm{HH})}(k,i) \cdot S^{(j-1)}(2m-k,2n-i),$$

$$B^{(j)}(m,n) = \sum_{k=0}^{L-1}\sum_{i=0}^{M-1} H^{(\mathrm{HL})}(k,i) \cdot S^{(j-1)}(2m-k,2n-i),$$

It is seen from that the four decomposed sub bands at a level are obtained by performing four 2-D convolutions. Each 2-D convolution can be seen as

$$C^{(j)}(m,n) = \sum_{k=0}^{L-1}\sum_{i=0}^{M-1} H^{(\mathrm{LH})}(k,i) \cdot S^{(j-1)}(2m-k,2n-i),$$

$$S^{(j)}(m,n) = \sum_{k=0}^{L-1}\sum_{i=0}^{M-1} H^{(\mathrm{LL})}(k,i) \cdot S^{(j-1)}(2m-k,2n-i),$$

a sum of the products of the filter coefficients and the elements contained in an window sliding on a 2-D data. The decimation by a factor of two in both the horizontal and vertical dimensions can be accomplished by sliding the window by two positions horizontally and vertically for the computation of two successive samples. Only the LL sub band data of decomposition are used as input for the decomposition at the next level. After iterations, the 2-D signal is transformed into resolution levels, with HH, HL and LH sub bands from each of the first levels and HH, HL, LH and LL sub bands from the last level. Since, the number of samples that need to be processed at each level is one quarter of that at the preceding level.

III. Pipeline Structure for 2D DWT

In a pipeline structure a number of stages are used for the computation of decomposition levels. Each levels of decomposition are mapped to the various stages of pipeline. A tap filtering is carried out for all samples of the input. The operation of filtering is divided into four types of filters as high-high pass filter, high-lowpass filter, low-high pass filter and low-lowpass filter. Following the operation of filter division the filtering operation are carried out. In FIR filter at first stage L*M multiplication are done then L*M accumulation. Pipeline structure is designed in such a way that computation is done at a faster rate with less usage of hardware resources and with easy design complexity. The number of filter units used by each stage should be minimum. Too many usage of filter will lead to complex computation. In order to reduce the complexity the straight forward mapping can be used.

One level to one stage mapping is carried out for overall computation. The usage of hardware should be one fourth of used by earlier stage. The number of filter usage will decrease when compared to one stage to next stage .The number of filter usage will decrease when compared to one stage to next stage in pipeline architecture causing the synchronization process difficult. To reduce such synchronization problem more multiplexers and registers are used which causes higher complexity of hardware. Hence $\lambda<1$ is used to make the synchronization process easier.

The λ value can be increased by dividing large stages into small stages and combining various small stages to merge them and form large stages. The number of filter units can be reduced by using merger.

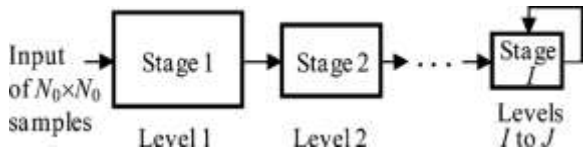

Fig.1 Pipeline structure with stages for -level computation

IV. DESIGN OF THE ARCHITECTURE

In the previous section, a three-stage pipeline structure for the computation of the 2-D DWT is advocated. The pipeline solution realizes an optimal combination of the parameters for the hardware utilization and pipeline synchronization. As in any pipeline structure in the proposed three-stage structure the operations in a given stage depend on the data produced by the preceding stage. The computational load of the various decomposition levels of the 2-D DWT computation has been distributed among these three stages. The operations in third stage depend on the data produced by it, whereas first and second stages of the pipeline do not depend on the data produced by them. The operations on the three stages need to be synchronized such that each stage performs multiple computations in multiple levels within a minimum possible time period while using the available hardware resources maximally. In this section, we present the design of the proposed 3-stage pipeline architecture, which is focusing on synchronization of the operations of the stages as well as details of the intra-stage design so as to provide an optimal performance.

A. Stage Synchronization

Distribution of the computational load among the three stages, and the hardware resources made available to them are in the ratio 8:2:1 as discussed in Section III. The synchronization of the operations between the stages is carried out under above constraint of the distribution of the computational load and hardware resources. The computation of a decomposition level *j* depends on the data computed at its previous level *j-1*, as it's the nature of DWT. And so, the number of computations is four times of that at the decomposition level *j*. Once the operation started in pipeline, the stages of pipeline need to be synchronized in such a way that each stage starts the operation at an earliest possible time and ensured that the required data become available for its operation and it must be continued until the task assigned to it is fully completed.

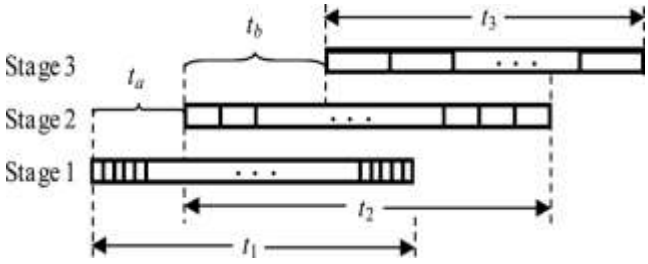

Fig. 2: Timing diagram for the operations of three stages

Consider the time taken for the operations of the three stages are t_1,t_2 and t_3 individually by stages 1, 2 and 3 respectively, to complete their assigned tasks. The elapsed times between stage 1 and 2 and stage 2 and 3 is ta and t_b respectively as shown in Fig . Note that the lengths of the times t_1 , t_2 and t_3 to complete the tasks by individual stages are approximately the same, since the ratios of the tasks assigned and the resources made available to the three stages are the same. The average times to compute one output sample by stages 1, 2 and 3 are in the ratio 1:4:8. The relative widths of the slots in the three stages are shown to reflect this ratio 1:4:8. The main objective of computation is to minimize the total computation $t_a+t_b+t_3$ time by minimizing t_a , t_b and t_3 individually. With the assumption the 2-D output samples for a decomposition level are computed row-by-row starting from the upper-left corner sample. The operations in stage 1 are independent of other two stages, so it can operate continuously to compute all the samples of level 1. The value of t_1 is equal to $T_sN^2_1$, where T_s is the average time taken by stage 1 to compute one output sample. The operations of stages are delayed certain amount time of since the stages 2 and 3 require the output data computed by stages 1 and 2, respectively. The operations of the three stages can be done in following manner. Stage 1 computes level -1 output by continuously operating on the input signal. Stage 2 starts its operation immediately and continues its operation of all other level-2 output samples in a sequential manner. Stage 3 starts its operation for the computation of level-3 samples immediately after stage 2 completes the computation of the level-2 output sample.

B. Design of Stages

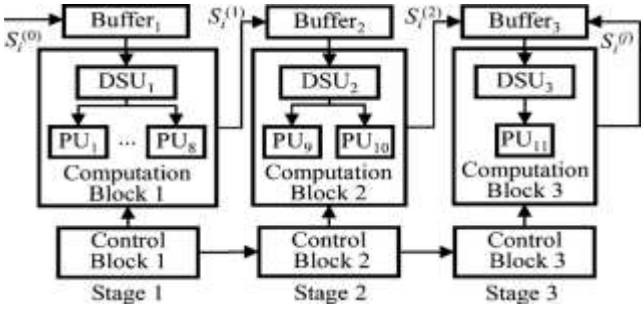

Fig.3. Block diagram of the three-stage architecture.

The block diagrams for pipeline with all the components required by the three stages are shown in Fig 3. Note that the data flow shown in this figure comprises only the LL-subband data necessary for the operations of the stages. The HH, HL and LH subband data are outputted directly to an external memory. Now, we give details on the structure of the data scanning unit to scan the 2-D data and establish four distinct sub-windows, as well as on the distribution of the filtering operations to the processing units in each stage.

V. DESIGN OF THE PROCESSING UNIT

In each stage, a processing unit carries out an -tap filtering operation using the samples of an sub-window at a time to produce the corresponding output. Since the sub-windows cannot be fed into a processing unit at a rate faster than the rate at which these sub-windows are processed by the processing unit, the processing time to process a sub-window (one time unit) is critical in determining the maximum clock frequency at which the processing units can operate. Each physical link from a given bit of the input to an output bit of the processing unit gives rise to a data path having a delay that depends on the number and the types of operations being carried out along that path. Therefore, it is crucial to aim at achieving the shortest possible delay for the critical path when designing a processing unit for our architecture. The filtering operation carried out by a processing unit, as described above, can be seen as parallel multiplications followed by an accumulation of the products. If the input samples and the filter coefficients have the word lengths of and bits, respectively, then the processing unit produces an array of bits simultaneously in one clock cycle. In order to obtain the output sample corresponding to a given sub-window, the bits of the partial products must be accumulated vertically downward and from right to left by taking the propagation of the carry bits into consideration. The task of this accumulation can be divided into a sequence of layers. The shortest critical data path can be achieved by minimizing number of layers and the delay of the layers. In each layer, a number of bits consisting of the partial product bits and/or the carry bits from different rows need to be added. This can be done by employing in parallel as many bit-wise adders as needed in each layer. The idea behind using bit-wise adder is to produce to the extent possible the number of output bits from a layer is smaller than the number of input bits to that layer. This can be done by using full adders and specifically designed double adders, in which the full adder consumes 3 bits and produces 2 bits (one sum and one carry bits) whereas the double adder consumes two pairs of bits from neighboring columns and produces 3 bits (one sum and two carry bits/two sum and one carry bits). The two types of adders have equal delay, and are efficient in generating carry bits and compressing the number of partial products. With this structure of the layers, the number of layers becomes minimum possible and the delay of a layer is equal to that of a full adder or equivalently to that of a double adder, thereby providing the shortest critical path for the accumulation network.

Since the two rows of bits produced by the accumulation network still remain un accumulated, they finally need to be added to produce one row of output bits in the final phase of the task of a processing unit by using a carry propagation adder. Note that tasks of the accumulation network and the carry propagation adder can be made to have some partial overlap, since the latter can start its processing as soon as the rightmost pairs of bits becomes available from the former. In the block diagram shown in Fig .4 of processing unit in proposed system the squaring circuit are used in order to increase the processing speed of the processing unit. Instead of using carry propagation adder carry save adder are used. A carry-save adder is a type of digital adder, used in computer micro architecture to compute the sum of three or more *n*-bit numbers in binary.

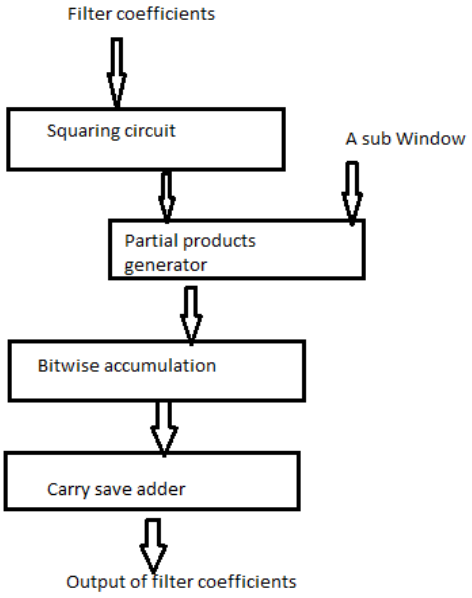

Fig 4.Block diagram of a processing unit in proposed system

It differs from other digital adders in that it outputs two numbers of the same dimensions as the inputs, one which is a sequence of partial sum bits and another which is a sequence of carry bits. This will increase the speed of a processing unit and gives more efficiency.

VI .PERFORMANCE RESULTS AND COMPARISONS

In order to evaluate the performance of a computational architecture, one needs to make use of certain metrics that characterize the architecture in terms of the hardware resources used and the computation time.

Device utilization summary			
Logic utilization	Used	Available	Utilization
No.of slices	86	768	11%
No.of LUT'S	132	1536	10%
No.of Bonded IOB's	58	63	15%

Table 1: Device Utilization for existing system design.

In this paper, the hardware resources used for the filtering operation are measured by the number of multipliers and the number of adders, and that used for the storage of data and filter coefficients are measured by the number of registers. The computation time, in general, is technology dependent. However, a metric that is technology independent and can be used to determine the computation time is the number of clock cycles elapsed between the first and the last samples inputted to the architecture.

Device utilization summary			
Logic utilization	Used	Available	Utilization
No.of slices	58	768	7%
No.of LUT'S	101	1536	6%
No.of Bonded IOB's	24	63	38%

Table 2: Device Utilization for Proposed system design.

Comparison of existing and proposed system design utilization for processing unit is shown in the table 1 and table 2. It shows that the area covered in our proposed system is very less when compared to existing system and computation speed is high.

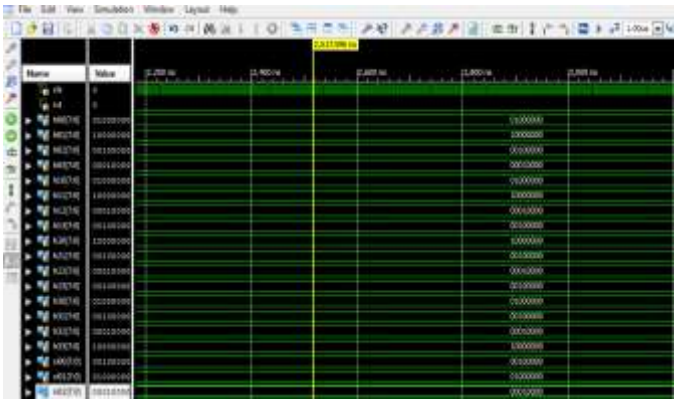

Fig.5. Simulation Results- High hig-pass filtering

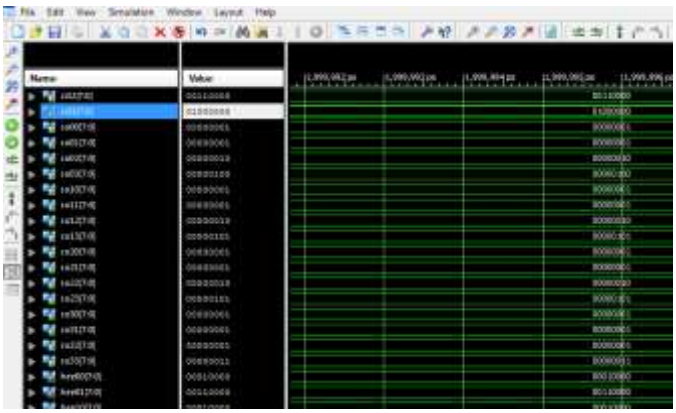

Fig.6. Simulation Results -High low-pass filtering

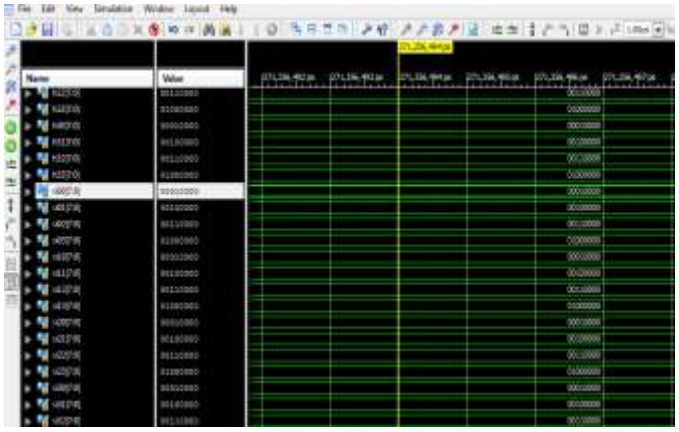

Fig.7.Simulation Results- Low high –pass filtering

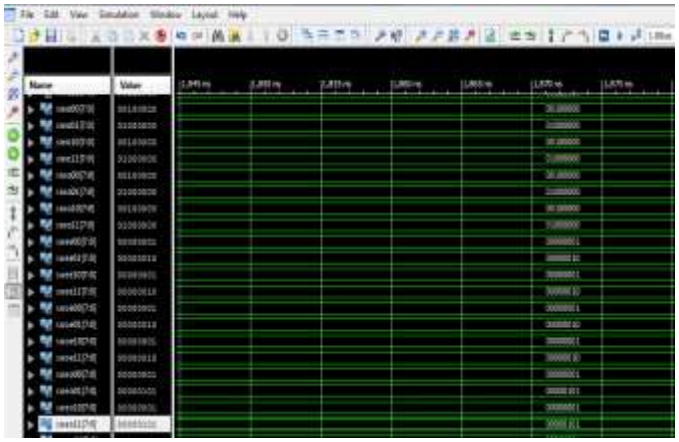

Fig.8. Simulation Results- Low low-pass filtering

The simulation results are taken for an image and image pixel representations are divided into four filtering operation as high high-pass, high low-pass, low high-pass, low low-pass filtering. The calculations are done based on four equations listed in Section II and squaring coefficient is used for filter coefficient. This gives better performance, high speed and hardware utilization of 2D –DWT computation is efficiently reduced.

VII. CONCLUSION

In pipelined DWT our contribution in the paper is modifying the processing unit (PU) of the pipelined DWT in such a way that computation speed of the DWT will be increased. The main modules in the PU are truncated multiplier and square calculation. Truncated multiplier and square cal modules are designed using verilog and simulated using Xilinx 12.3 screen shots of simulated outputs are attached. The filtering operation is done in the data storage unit. In the proposed system the filtering operations calculates the result directly in the last stage thus it eliminates the first two stages causing the computation speed to be high.

REFERENCES

[1] G.Eason,B.Noble,andI.N.Sneddon,"OncertainintegralsofLipschitz-HankeltypeinvolvingproductsofBesselfunctions," Phil.Trans.Roy.Soc.London,vol.A247,pp.529-551, April 1955.

[2] J.ClerkMaxwell,ATreatiseonElectricityandMagnetism,3rded.,vol.2.Oxford:Clarendon,1892,pp.68–73.

[3] I.S.JacobsandC.P.Bean,"Fineparticles,thinfilmsandexchangeanisotropy,"inMagnetism,vol.III,G.T.RadoandH.Suhl,Eds.NewYork:Academic,1963,pp.271–350.

[4] K.Elissa, "Titleofpaperifknown," unpublished.

[5] R.Nicole, "Title of paper with only first word capitalized," J.NameStand.Abbrev.,in press.

[6] Y.Yorozu,M.Hirano,K.Oka,andY.Tagawa,"Electronspectroscopystudiesonmagneto-opticalmediaandplasticsubstrateinterface," IEEETransl.J.Magn.Japan,vol.2,pp.740–741,August1987 [Digests9thAnnualConf.MagneticsJapan,p.30,1982].

[7] P. K. Meher, B. K. Mohanty, and J. C. Patra, "Hardware-efficient systolic- like modular design for two-dimensional discrete wavelet transform," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 55, no. 2, pp. 151–155, Feb. 2008.

[8] A. Benkrid, D. Crookes, and K. Benkrid, "Design and implementation of a generic 2-D orthogonal discrete wavelet transform on an FPGA," in *Proc. IEEE 9th Symp. Field-programming Custom Computing Machines (FCCM)*, Apr. 2001, pp. 190–198.

[9] P. McCanny, S. Masud, and J. McCanny, "Design and implementation of the symmetrically extended 2-D wavelet transform," in *Proc. IEEE Int. Conf. Acoustic, Speech, Signal Process. (ICASSP)*, 2002, vol. 3, pp. 3108–3111.

[10] S. Raghunath and S. M. Aziz, "High speed area efficient multi-resolution 2-D 9/7 filter DWT processor," in *Proc. Int. Conf. Very Large Scale Integration (IFIP)*, Oct. 2006, vol. 16–18, pp. 210–215.

[11] M. Angelopoulou, K. Masselos, P. Cheung, and Y. Andreopoulos, "A comparison of 2-D discrete wavelet transform computation schedules on FPGAs," in *Proc. IEEE Int. Conf. Field Programmable Technology (FPT)*, Bangkok, Tailand, Dec. 2006, pp. 181–188.

[12] C. Chrysytis and A. Ortega, "Line-based, reduced memory, wavelet image compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 3, pp. 378–389, Mar. 2000.

[13] M. Ravasi, L. Tenze, andM.Mattavelli, "A scalable and programmable architecture for 2-D DWT decoding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 8, pp. 671–677, Aug. 2002.

[14] K. G. Oweiss, A. Mason, Y. Suhail, A. M. Kamboh, and K. E. Thomson, "A scalable wavelet transform VLSI architecture for real-time signal processing in high-density intra-cortical implants," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 6, pp. 1266–1278, Jun. 2007.

[15] G. Shi, W. Liu, L. Zhang, and F. Li, "An efficient folded architecture for lifting-based discrete wavelet transform," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 56, no. 4, pp. 290–294, Apr. 2009.

[16] M. Alam,W. Badawy, V. Dimitrov, and G. Jullien, "An efficient architecture for a lifted 2-D biorthogonal DWT," *J. VLSI Signal Process.*, vol. 40, pp. 333–342, 2005.

[17] C. Chakrabarti and M. Vishwanath, "Efficient realizations of the discrete and continuous wavelet transforms: From single chip implementations to mapping on SIMD array computers," *IEEE Trans. Signal Process.*, vol. 43, no. 3, pp. 759–771, Mar. 1995.

[18] C. Cheng and K. K. Parhi, "High-speed VLSI implementation of 2-D discrete wavelet transform," *IEEE Trans. Signal Process.*, vol. 56, no. 1, pp. 393–403, Jan. 2008.

[19] K. C. Hung, Y. S. Hung, and Y. J. Huang, "A nonseparable VLSI architecture for two-dimensional discrete periodized wavelet transform," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 9, no. 5, pp. 565–576, Oct. 2001.

[20] I. S. Uzun and A. Amira, "Rapid prototyping—framework for FPGAbased discrete biorthogonal wavelet transforms implementation," *IEE Vision, Image Signal Process.*, vol. 153, no. 6, pp. 721–734, Dec. 2006.

[21] F. Marino, "Efficient high-speed low-power pipelined architecture for the direct 2-D discrete wavelet transform," *IEEE Trans. Circuits Syst II, Analog. Digit. Signal Process.*, vol. 47, no. 12, pp. 1476–1491, Dec. 2000.

[22] R. J.C. Palero, R. G. Gironez, andA. S. Cortes, "Anovel FPGAarchitecture of a 2-D wavelet transform," *J. VLSI Signal Process.*, vol. 42, pp. 273–284, 2006.

[23] Q. Dai, X. Chen, and C. Lin, "A novel VLSI architecture for multidimensional discrete wavelet transform," *IEEE Trans. Circuits Syst.Video Technol.*, vol. 14, no. 8, pp. 1105–1110, Aug. 2004.

[24] M. H. Sheu,M. D. Shieh, and S. W. Liu, "A low cost VLSI architecture design for nonseparable 2-D discrete wavelet transform," in *Proc. 40th Midwest Symp. Circuits Syst*, 1997, vol. 2, pp. 1217–1220.

[25] C. Y. Chen, Z. L.Yang, T. C.Wamg, and L.G. Chen, "A programmable VLSI architecture for 2-D discrete wavelet transform," in *Proc. IEEE Int. Symp. Circuits Systems (ISCAS)*, Geneva, Switzerland,May 28–31 2000, vol. 1, pp. 619–622.

[26] B. K. Mohanty and P. K. Meher, "Bit-serial systolic architecture for 2-D non-separable dircrete wavelet transform," in *Proc. Int. Conf. Intell. Adv. Syst. (ICIAS)*, Kualalampur, Malaysia, Nov. 2007.

[27] C. Yu and S.-J. Chen, "VLSI implementation of 2-D discrete wavelet transform for real-time video signal processing," *IEEE Trans. Consum. Electron.*, vol. 43, no. 4, Nov. 1997.

[28] M. Vishwanath, "The recursive pyramid algorithm for the discrete wavelet transforms," *IEEE Trans. Signal Process.*, vol. 42, no. 3, pp.673–677, 1994.

[29] K. A. Kotteri, S. Barua, A. E. Bell, and J. E. Carletta, "A comparison of hardware implementations of the biorthogonal 9/7 DWT: Convolution versus lifting," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 52, no. 5, pp. 256–260, May 2006.

[30] M. Ferretti and D. Rizzo, "Handling borders in systolic architectures for the 1-D discrete wavelet transform for perfect reconstruction," *IEEE Trans. Signal Process.*, vol. 48, no. 5, pp. 1365–1378, May 2000.

[31] D.Guevorkian, P. Liuha, A. Launiainen, and V. Lappalainen, "Architectures for Discrete Wavelet Transforms," U.S. 6976046, Dec. 13,2005.

[32] J. Song and I. Park, "Pipelined discrete wavelet transform architecture scanning dual lines," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 56, no. 12, pp. 916–920, Dec. 2009.

[33] C. Zhang, C.Wang, and M. O. Ahmad, "A VLSI architecture for a fast computation of the 2-D discrete wavelet transform," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2007.

[34] C. Zhang, C. Wang, and M. O. Ahmad, "An efficient buffer-based architecture for on-line computation of 1-D discrete wavelet transform," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Process. (ICASSP)*,May 2004, vol. 5, pp. 201–204.

[35] C. Zhang, C. Wang, and M. O. Ahmad, "A VLSI architecture for a high-speed computation of the 1-D discrete wavelet transform," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2005, vol. 2, pp. 1461–1464.

[36] C. Zhang, C. Wang, and M. O. Ahmad, "A pipeline VLSI architecture for high-speed computation of the 1-D discrete wavelet transform," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 10, pp. 2729–2740, Oct. 2010.