A Pipeline VLSI Architecture for Fast Computation of the 2-D Discrete Wavelet Transform

Chengjun Zhang, Chunyan Wang, *Senior Member*, *IEEE*, and M. Omair Ahmad, *Fellow*, *IEEE*

***Abstract* — In this paper, a scheme for the design of a high-speed pipeline VLSI architecture for the computation of the 2-D discrete wavelet transform (DWT) is proposed. 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 an 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. To validate the proposed scheme, a circuit is designed, simulated, and implemented in FPGA for the 2-D DWT computation. The results of the implementation show that the circuit is capable of operating with a maximum clock frequency of 134 MHz and processing 1022 frames of size 512×512 per second with this operating frequency. It is shown that the performance in terms of the processing speed of the architecture designed based on the proposed scheme is superior to those of the architectures designed using other existing schemes, and it has similar or lower hardware consumption.**

***Index Terms* — Discrete wavelet transform, FPGA implementation, image processing, parallel architecture, pipeline architecture, real-time processing, VLSI architecture, multi-resolution filtering, non-separable approach, computational parallelism.**

I. INTRODUCTION

THE 2-D discrete wavelet transforms (DWT) have been widely used in many engineering applications because of their multi-resolution decomposition capability [1]. However, processing large volumes of data of various decomposition levels of the transform makes their computation computationally very intensive. In the past, many architectures have been proposed aimed at providing high-speed 2-D DWT computation with the requirement of utilizing a reasonable amount of hardware resources. These architectures can be broadly classified into separable [2]−[16] and non-separable architectures [17]−[27]. In a separable architecture, 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. Vishwanath *et al.* [2] have proposed a low-storage short-latency separable architecture in which the row-wise operations are performed by systolic filters and the column-wise operations by parallel filters. 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) [28]. Liao *et al*. [3] have introduced an architecture 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. The scheme leads to a low-complexity architecture with a large latency. The separable architectures, in which a 1-D filtering structure is used to perform the 2-D DWT, have an additional requirement of transposing the intermediate data between the two 1-D filtering processes. This increases the memory size as well as the latency of the architectures. The non-separable architectures do not have this problem, since in these architectures, the 2-D transforms are computed directly by using 2-D filters. Chakrabarti *et al.* [17] have proposed two non-separable architectures, one using parallel 2-D filters and the other an SIMD 2-D array, both based on a modified RPA. In the former architecture, a high degree of computational parallelism is achieved at the expense of less efficient hardware utilization, whereas the latter 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 in which a number of parallel FIR filters with a polyphase structure are used to improve the processing speed at the expense of increased hardware. Hung *et al*. [19], in an effort to provide a reduced count of multipliers and to facilitate the processing of the boundary data, 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. But the processing speed of this architecture is low

Manuscript received January 16, 2011, revised July 12, 2011. This work was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada and in part by the Regroupement Stratégique en Microélectronique du Québec (ReSMiQ).

The authors are with the Center for Signal Processing and Communications, Department of Electrical and Computer Engineering, Concordia University, Montréal, QC, H3G 1M8 Canada (e-mail: z_chengj@ece.concordia.ca; chunyan@ece.concordia.ca; omair@ece.concordia.ca).

in view of the fact that the same architecture is utilized recursively to perform the tasks of successive decomposition levels. Marino [21] has proposed a two-stage pipeline architecture in which the first stage performs the task of the first decomposition level and the second one that of all the remaining levels, and has aimed at providing a short computation time. As the processing units employed in this architecture differ from one another, the complexity of the hardware resources is high and the design of the architecture is complicated. 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.

In this paper, a non-separable pipeline architecture for fast computation of the 2-D DWT with a reasonable low cost for the hardware resources is proposed. The high-speed computation is achieved by efficiently distributing the task of the computations of multiple decomposition levels among the stages of the pipeline, and by optimally configuring the data and synchronizing the operations of pipeline so as to maximize the inter-stage and intra-stage computational parallelism. The paper is organized as follows. In Section II, a mathematical formulation of the 2-D DWT computation necessary for the development of the proposed architecture is presented. In Section III, a study is conducted to determine the number of stages of a pipeline necessary for optimally mapping the task of the DWT computation onto the stages of the pipeline. Based on this study, in Section IV, a three-stage pipeline architecture is developed with an efficient structure of the 2-D input data and an optimal organization of the processing units in each of the stages. In Section V, the performance of the proposed architecture is assessed and compared with that of other existing architectures and validated by an FPGA implementation. Section VI summarizes the work of this paper by highlighting the salient features of the proposed architecture.

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 multiresolution domain by lowpass and highpass FIR filters along each of the two dimensions. The four FIR filters, denoted as highpass-highpass (HH), highpass-lowpass (HL), lowpass-highpass (LH) and lowpass-lowpass (LL) filters, produce, respectively, the HH, HL, LH and LL subband data of the decomposed signal at a given resolution level. The samples of the four subbands 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 subband signal from the previous decomposition level is used as input.

A. Formulation for the 2-D DWT Computation

Let a 2-D signal be represented by an $N_0 \times N_0$ matrix $\mathbf{S}^{(0)}$, with its (m,n)th element denoted by $S^{(0)}(m,n)$ $(0 \le m, n \le N_0-1)$, where N_0 is chosen to be 2^J, J being an integer. Let the coefficients of a 2-D FIR filter P (P=HH, HL, LH, LL) be represented by an $L \times M$ matrix $\mathbf{H}^{(\mathrm{P})}$. The (k,i)th coefficient of the filter P is denoted by $H^{(\mathrm{P})}(k,i)$ $(0 \le k \le L-1,\ 0 \le i \le M-1)$. The decomposition at a given level $j = 1, 2, \cdots, J$ can be expressed as

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

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

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

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

where $A^{(j)}(m,n)$, $B^{(j)}(m,n)$, $C^{(j)}(m,n)$ and $S^{(j)}(m,n)$ $(0 \le m, n \le N_j-1)$ denote the (m,n)th elements of the four $N_j \times N_j$ $(N_j = N_0/2^j)$ matrices, $\mathbf{A}^{(j)}$, $\mathbf{B}^{(j)}$, $\mathbf{C}^{(j)}$ and $\mathbf{S}^{(j)}$, respectively, representing the HH, HL, LH and LL subbands of the 2-D input signal at the jth level. It is seen from (1) that the four decomposed subbands at a level are obtained by performing four 2-D convolutions. Each 2-D convolution can be seen as a sum of the products of the $L \times M$ filter coefficients and the elements contained in an $L \times M$ 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 $L \times M$ window by two positions horizontally and vertically for the computation of two successive samples. Only the LL subband data of decomposition are used as input for the decomposition at the next level. After J iterations, the 2-D signal $\mathbf{S}^{(0)}$ is transformed into J resolution levels, with HH, HL and LH subbands from each of the first $J-1$ levels and HH, HL, LH and LL subbands from the last (Jth) level. Since $N_j = N_0/2^j$, the number of samples that need to be processed at each level j is one quarter of that at the preceding level.

B. Formulation for a Four-channel Filtering Operation

In order to facilitate parallel processing for the 2–D DWT computation, the $L \times M$ filtering operation needs to be divided into multi-channel operations, each channel processing one part of the 2-D data. It is seen from (1) that the even and odd indexed elements are always operated on the even and odd indexed filter coefficients, respectively. The matrix $\mathbf{S}^{(j)}$ representing the LL subband at the jth level can, therefore, be divided into four $(N_j/2+L/2) \times (N_j/2+M/2)$ sub-matrices, $\mathbf{S}_{ee}^{(j)}, \mathbf{S}_{oe}^{(j)}, \mathbf{S}_{eo}^{(j)}$ and $\mathbf{S}_{oo}^{(j)}$, whose (m,n)th $(0 \le m \le N_j/2+L/2-1,\ 0 \le n \le N_j/2+M/2-1)$ elements are given by

$$\begin{aligned} S_{ee}^{(j)}(m,n) &= S^{(j)}(2m, 2n) \\ S_{oe}^{(j)}(m,n) &= S^{(j)}(2m+1, 2n) \\ S_{eo}^{(j)}(m,n) &= S^{(j)}(2m, 2n+1) \\ S_{oo}^{(j)}(m,n) &= S^{(j)}(2m+1, 2n+1) \end{aligned} \quad (2)$$

taking into consideration the periodic padding samples at the boundary [30]. It is seen from (2) that the data at any decomposition level are divided into four channels for processing by first separating the even and odd indexed rows of $\mathbf{S}^{(j)}$, and then separating the even and odd indexed columns of the resulting two sub-matrices. The data in each channel can then be computed by an $(L/2\times M/2)$-tap filtering operation. In order to facilitate such a 4-channel filtering operation, the filter coefficients, as used in (1), need to be decomposed appropriately. Accordingly, the matrix $\mathbf{H}^{(\mathrm{P})}$ needs to be decomposed into four $(L/2\times M/2)$ sub-matrices, $\mathbf{H}_{ee}^{(\mathrm{P})}, \mathbf{H}_{oe}^{(\mathrm{P})}, \mathbf{H}_{eo}^{(\mathrm{P})}$ and $\mathbf{H}_{oo}^{(\mathrm{P})}$, whose (k,i)th $(0\le k\le L/2-1, 0\le i\le M/2-1)$ elements are given by

$$\begin{aligned} H_{ee}^{(\mathrm{P})}(k,i) &= H^{(\mathrm{P})}(2k,2i) \\ H_{oe}^{(\mathrm{P})}(k,i) &= H^{(\mathrm{P})}(2k+1,2i) \\ H_{eo}^{(\mathrm{P})}(k,i) &= H^{(\mathrm{P})}(2k,2i+1) \\ H_{oo}^{(\mathrm{P})}(k,i) &= H^{(\mathrm{P})}(2k+1,2i+1) \end{aligned} \tag{3}$$

respectively. By using (2) and (3) in (1), any of the four subband signals, $\mathbf{A}^{(j)}$, $\mathbf{B}^{(j)}$, $\mathbf{C}^{(j)}$ and $\mathbf{S}^{(j)}$, at the jth decomposition level, can be computed as a sum of four convolutions using $(L/2\times M/2)$-tap filters. For example, the LL subband given by (1d) can now be expressed as

$$\begin{aligned} S^{(j)}(m,n) &= \sum_{k=0}^{L/2-1}\sum_{i=0}^{M/2-1} H_{ee}^{(\mathrm{LL})}(k,i)\cdot S_{ee}^{(j-1)}(m+k,n+i) \\ &+ \sum_{k=0}^{L/2-1}\sum_{i=0}^{M/2-1} H_{oe}^{(\mathrm{LL})}(k,i)\cdot S_{oe}^{(j-1)}(m+k,n+i) \\ &+ \sum_{k=0}^{L/2-1}\sum_{i=0}^{M/2-1} H_{eo}^{(\mathrm{LL})}(k,i)\cdot S_{eo}^{(j-1)}(m+k,n+i) \\ &+ \sum_{k=0}^{L/2-1}\sum_{i=0}^{M/2-1} H_{oo}^{(\mathrm{LL})}(k,i)\cdot S_{oo}^{(j-1)}(m+k,n+i) \end{aligned} \tag{4}$$

At any decomposition level, the separation of the subband processing corresponding to even and odd indexed data as given by (4) is consistent with the requirement of decimation of the data in each dimension by a factor of two in the DWT computation. It is also seen from (4) that the filtering operations in the four channels are independent and identical, which can be exploited in the design of an efficient pipeline architecture for the 2-D DWT computation.

III. Pipeline for the 2-D DWT Computation

In a pipeline structure for the DWT computation, multiple stages are used to carry out the computations of the various decomposition levels of the transform [31]. The computation corresponding to each decomposition level needs to be mapped to a stage or stages of the pipeline. It is seen from the formulation in Section II that the task of computing the jth decomposition level in a J-level DWT computation consists of computing $N_0^2/4^{j-1}$ samples, where $N_0=2^J$. The computation of each sample actually performs an $(L\times M)$-tap HH, HL, LH or LL FIR filtering operation that comprises the operations of $(L\times M)$ multiplications followed by $(L\times M)$ accumulations. Assuming that these operations for the computation of one sample are carried out by a unit of filter processor, the overall task of the DWT computation would require a certain number of such filter units. In order to design a pipeline structure capable of performing a fast computation of the DWT with low expense on hardware resources and low design complexity, an optimal mapping of the overall task of the DWT computation to the various stages of the pipeline needs to be determined. Any distribution of the overall task of the DWT computation to stages must consider the inherent nature of the sequential computations of the decomposition levels that limit the computational parallelism of the pipeline stages, and consequently the latency of the pipeline. The key factors in the distribution of the task to the stages are the maximization of the inter-stage and intra-stage computational parallelism and the synchronization of the stages within the constraint of the sequential nature of the computation of the decomposition levels. The feature of identical operations associated with the computations of all the output samples irrespective of the decomposition levels in a DWT computation can be exploited to maximize the intra-stage parallelism of the pipeline. Further, in order to minimize the expense on the hardware resources of the pipeline, the number of filter units used by each stage ought to be minimum and proportional to the amount of the task assigned to the stage.

A straightforward mapping of the overall task of the DWT computation to a pipeline is one-level to one-stage mapping, in which the tasks of J decomposition levels are distributed to J stages of the pipeline. In this mapping, the amount of hardware resources used by a stage should be one-quarter of that used by the preceding stage. Thus, the ratio λ of the hardware resource used by the last stage to that used by the first stage has a value of $1/4^{J-1}$. For images of typical size, this parameter would assume a very small value. Hence, for a structure of the pipeline that uses identical filter units, the number of these filter units would be very large. Further, since the number of such filter units employed by the stages would decrease exponentially from one stage to the next in pipeline, it will make their synchronization very difficult. The solution to such a difficult synchronization problem, in general, requires more control units, multiplexers and registers, which results in a higher complexity of the hardware resources. A reasonably large value of $\lambda<1$ would be more attractive for synchronization. In this respect, the parameter λ can be seen as a measure of difficulty in that a smaller value of this parameter implies a greater design effort and more hardware resources for the pipeline.

The parameter λ can be increased from its value of $1/4^{J-1}$ in the one-level to one-stage pipeline structure by dividing the large-size stages into a number of smaller stages or merging the small-size stages into larger ones. However, dividing a stage of the one-level to one-stage pipeline into multiple stages would require a division of the task associated with the corresponding decomposition level into sub-tasks, which in turn, would call for a solution of even a more complex problem of synchronization of the sub-tasks associated with divided stages. On the other hand, merging multiple small-size stages of the pipeline into one stage would not create any additional synchronization problem. As a matter of fact, such a merger

could be used to reduce the overall number of filter units of the pipeline.

In view of the above discussion, the synchronization parameter λ can be increased by merging a number of stages at tail end of the pipeline. Fig. 1 shows the structure of a pipeline in which the stages I to J of the one-level to one-stage pipeline have been merged. In this structure, the tasks of the decomposition level from $j=1$ to $j=I-1$ are mapped to stage 1 to $I-1$, respectively, whereas those of the decomposition levels $j=I, \cdots, J$, are mapped all together to the Ith stage. Note that the total amount of computations performed by stage I is less than one-half of that performed by stage $I-1$. Considering the fact that the number of filter units employed by each stage of the pipeline is an integer, it is reasonable to have the ratio of the numbers of filter units used by the last two stages (i.e., stages $I-1$ and I) to be 2:1. The value of the parameter λ is now increased from $1/4^{J-1}$ to $1/4^{I-1.5}$. However, now the resources employed by stage I would not be fully utilized, which would lower the efficiency of the hardware utilization of the pipeline of Fig. 1. Assume that the parameter η represents the hardware utilization efficiency defined as the ratio of the resources used to that employed by the pipeline. The hardware utilization efficiency η of the pipeline in Fig. 1 can be shown to be equal to $(1-4^{-J})/(1+4^{-I+0.5})$. Since for images of typical size, 4^{-J} is negligibly small compared to one, the expression for η can be simplified as $1/(1+4^{-I+0.5})$. As the number of stages I employed by the pipeline increases, the hardware utilization efficiency increases with the parameter η approaching unity for a maximum efficiency. On the other hand, the difficulty in synchronizing the stages gets worse as the parameter λ decreases with increasing value of I. A variation in the value of I results in the values of λ and η that are in conflict from the point of view of stage synchronization and hardware utilization efficiency. Therefore, a value of I needs to be determined that optimizes the values of λ and η jointly.

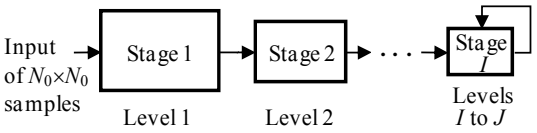

Fig. 1. Pipeline structure with I stages for J-level computation.

Considering an example of an image of size $2^8 \times 2^8$, in which case $J=8$. Table I gives the values of the parameters λ and η for the pipeline structures with $I=2$, 3 and 4. It is seen from this table that the 2-stage and 3-stage pipelines have acceptable values of λ, whereas the synchronization of the 4-stage pipeline would be very difficult because of its very low value of λ=1/32. On the other hand, the 3-stage and 4-stage pipelines have more desirable values of η in comparison to that for the 2-stage pipeline. Therefore, a 3-stage pipeline with an acceptable value for the synchronization parameter and high hardware utilization efficiency would be the best choice of a pipeline. Note that the size of the images used in typical applications would have little bearing on the conclusion thus reached regarding the number of stages employed in the pipeline. Also, note that a 3-stage pipeline can perform the DWT computation for a variable number of decomposition levels from 3 to J. With three as the optimal choice of the number of stages in a pipeline, one can now choose the minimum numbers of filter units as 8, 2 and 1 for the stages 1, 2 and 3 in order to perform the tasks associated with the decomposition levels 1, 2 and 3 to J together, respectively. The next section is concerned specifically with a detailed design of the 3-stage pipeline structure.

TABLE I
VALUES OF THE PARAMETERS λ AND η FOR A PIPELINE WITH NUMBER OF STAGES AS TWO, THREE AND FOUR ($J=8$)

Parameter	$I=2$	$I=3$	$I=4$
λ	1/2	1/8	1/32
η	89%	96%	99%

IV. DESIGN OF THE ARCHITECTURE

In the previous section, we advocated a three-stage pipeline structure for the computation of the 2-D DWT to realize an optimal combination of the parameters for the hardware utilization and pipeline synchronization. In this three-stage structure, like in any pipeline architecture, the operations in a given stage depend on the data produced by the preceding stage. However, because of the way that the computational load of the various decomposition levels of the 2-D DWT computation has been distributed among the three stages, the operations in the first and second stages of the pipeline do not depend on the data produced by themselves, whereas that in stage 3 does depend on the data produced by itself. The operations of the three stages need to be synchronized in a manner so that the three stages perform the computation of multiple decomposition 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, starting with the synchronization of the operations of the stages, and then focusing on the details of the intra-stage design so as to provide an optimal performance.

A. Synchronization of Stages

Recall from Section III that the distribution of the computational load among the three stages, and the hardware resources made available to them are in the ratio 8:2:1. Accordingly, the synchronization of the operations between the stages needs to be carried out under this constraint of the distribution of the computational load and hardware resources. According to the nature of the DWT, the computation of a decomposition level j depends on the data computed at its previous level $j-1$, in which the number of computations is four times of that at the decomposition level j. Therefore, the stages of pipeline need to be synchronized in such a way that each stage starts the operation at an earliest possible time when the required data become available for its operation. Once the operation of a stage is started, it must continue until the task assigned to it is fully completed.

Consider the timing diagram given in Fig. 2 for the operations of the three stages, where t_1, t_2 and t_3 are the times taken individually by stages 1, 2 and 3, respectively, to complete their assigned tasks, and t_a and t_b are the times elapsed between the starting points of the tasks by stages 1 and 2, and that by stages 2 and 3, respectively. 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. In Fig. 2, the relative widths of the slots in the three stages are shown to reflect this ratio. Our objective is to minimize the total computation time $t_a+t_b+t_3$ by minimizing t_a, t_b and t_3 individually.

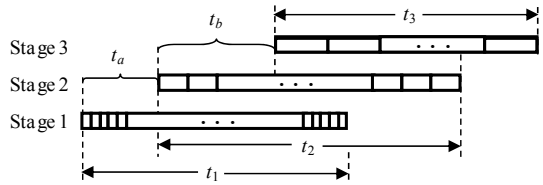

Fig. 2. Timing diagram for the operations of three stages.

Assume that 2-D output samples for a decomposition level are computed row-by-row starting from the upper-left corner sample. Since the operations in stage 1 are independent of those in the other two stages, it can operate continuously to compute all the samples of level 1. The value of t_1 is equal to $T_s N_1^2$, where T_s is the average time taken by stage 1 to compute one output sample. Since the operations of stages 2 and 3 require the output data computed by stages 1 and 2, respectively, their operations must be delayed by certain amount of times so that they can operate continuously with the data required by them becoming available. We now give the lowest bound on t_a and t_b so that once stages 2 and 3 start their operations they could continue their operations uninterruptedly. Since the operation of stage 2 starts at time t_a, the (i,k)th output sample of level 2, denoted by $S^{(2)}(i,k)$, will be computed starting at the time instant $t_x=t_a+4T_s(i\cdot N_2+k)$, where $4T_s$ is the average time taken by stage 2 to compute one output sample. Using (1), among the level-1 samples required for the computation of $S^{(2)}(i,k)$, the $(2i+L-1, 2k+M-1)$th level-1 sample, denoted by $S^{(1)}(2i+L-1, 2k+M-1)$, is the latest output sample computed at the time instant $t_y=T_s[N_1(2i+L-1)+2k+M-1]+T_s$. Now, if at the time of starting the calculation of the output sample $S^{(2)}(i,k)$, i.e. t_x, the sample $S^{(1)}(2i+L-1,2k+M-1)$ has already been calculated by stage 1, all the leve-1 samples necessary to calculate this level-2 output sample would be available. This requires us to impose the constraint $t_x>t_y$, for all i and k, i.e. $0\le i$, $k\le N_2-1$. This condition implies that

$$t_a > T_s(N_1 L - N_1 + M - 2k) \tag{5}$$

The minimum value of t_a is given by

$$t_{a\min} = T_s[N_1(L-1)+M] \tag{6}$$

Assume that stage 3 computes all the output samples of all remaining levels (i.e. level 3 to level J) in a sequential manner. We only need to consider the requirement of the data availability for the computation of level-3, which uses the level-2 samples computed by stage 2. Then, in a way similar to that obtaining $t_{a\min}$, by imposing the condition that at the time instant of starting the calculation of a level-3 output sample by stage 3, all the samples in the window of the level-2 output samples are available, it can be shown that the minimum value of t_b is given by

$$t_{b\min} = 4T_s[N_2(N_2/2+L-2)+M] \tag{7}$$

Based on the above discussion, the operations of the three stages can be arranged in the following manner:

Step 1. Stage 1 operates continuously on the input signal to compute the level-1 output samples sequentially.

Step 2. Stage 2 starts its operation immediately following the computation of the $(L-1, M)$th level-1 output sample, $S^{(1)}(L-1,M)$, and then continues its operation of all other level-2 output samples in a sequential manner.

Step 3. Stage 3 starts its operation for the computation of level-3 samples immediately after stage 2 completes the computation of the $(N_2/2+L-2, M-1)$th level-2 output sample, $S^{(2)}(N_2/2+L-2, M-1)$, and then continues the computation of other level-3 output samples sequentially. Computations of the output samples of levels 4 to J are carried out sequentially by the stage 3 following the computation of level-3 output samples.

B. Design of Stages

As discussed in Section III, in the proposed three-stage architecture, stages 1 and 2 perform the computations of levels 1 and 2, respectively, and stage 3 that of all the remaining levels. Since the basic operation of computing each output sample, regardless of the decomposition level or the subband, is the same, the computation blocks in the three stages can differ only in the number of identical processing units employed by them depending on the amount of the computations assigned to the stages. As seen from (4), an $(L\times M)$-tap filtering operation is decomposed into four independent $(L/2\times M/2)$-tap filtering operations, each operating on the 2-D $L/2\times M/2$ data resulting from the even or odd numbered rows and even or odd numbered columns of an $L\times M$ window of an LL-subband data. A unit consisting of $L/2\times M/2$ MAC cells can now be regarded as the basic *processing unit* to carry out an $(L/2\times M/2)$-tap filtering operation. An $L\times M$ window of the raw 2-D input data or that of an LL-subband data must be decomposed into four distinct $L/2\times M/2$ sub-windows in accordance with the four decomposed terms given by the right side of (4). This decomposition of the data in an $L\times M$ window can be accomplished by designing for each stage an appropriate data scanning unit (DSU) based on the way the raw input or the LL-subband data is scanned. The stages would also require memory space (buffer) to store the raw input data or the LL-subband data prior to scanning. Since stages 1 and 2 need to store only part of a few rows of raw input or LL-subband data at a time, they require a buffer of size of O(N), whereas since stage 3 needs to store the entire LL-subband data of a single

decomposition level, it has a buffer of size of O(N^2). Fig. 3 gives the block diagram of the pipeline showing all the components required by the three stages. 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 $L/2\times M/2$ sub-windows, as well as on the distribution of the filtering operations to the processing units in each stage.

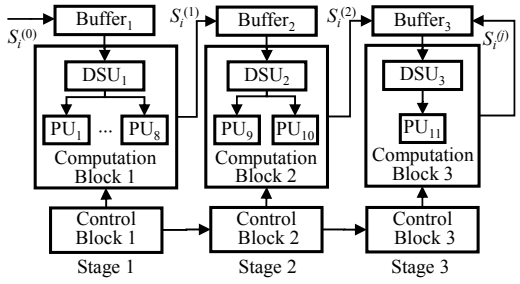

Fig. 3. Block diagram of the three-stage architecture.

1) Structure of the Data Scanning Unit

In accordance with (4), an $L\times M$ window of the raw 2-D input data stored in Buffer_1 or an LL-subband data stored in Buffer_2 or Buffer_3 must be partitioned into four $L/2\times M/2$ sub-windows, and stored into the DSU of the corresponding stage. Further, this same equation also dictates that a 2-D input data must be scanned in a sequential manner shown in Fig. 4(a). According to this sequence of scanning, the samples in a set of data comprising L rows of a 2-D input data are scanned starting from the top-left corner. Once the scanning of all the samples of L rows is completed, the process is repeated for another L rows after shifting down by two row positions. The objective is then to design a structure for a DSU so that samples scanned with this sequential mode get partitioned into the four sub-windows (Fig. 4(b)).

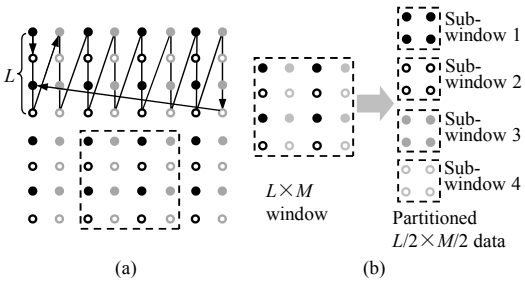

Fig. 4. (a) Scanning of an $N_j\times N_j$ 2-D data. (b) Partitioning of an $L\times M$ window into four $L/2\times M/2$ sub-windows. The solid and empty circles represent the samples in even-indexed and odd-indexed rows, respectively, whereas the black and grey circles represent the samples in even-indexed and odd-indexed columns, respectively.

In order to partition an $L\times M$ window into four $L/2\times M/2$ sub-windows, the structure of the DSU must first partition the samples of the window into two parts depending on whether a sample belongs to an even-indexed or odd-indexed row; then the samples in each part must be partitioned further into two parts depending on whether a sample belongs to an even-indexed or odd-indexed column. The first partition can be achieved by directing scanned samples alternatively to two sets of $L/2$ shift registers. The second partition can be achieved by reorganizing the samples stored in the shift registers of the two sets depending on whether a sample belongs to even-indexed or odd-indexed column by employing demultiplexers. Finally, the samples of the four sub-windows can be stored, respectively, into four units of $L/2\times M/2$ parallel registers. Fig. 5 shows a structure of the DSU to accomplish this task. This data scanning scheme automatically incorporates the downsampling operations by two in the vertical and horizontal directions (as required by the transform), and thus no additional peripheral circuits and registers are required for the downsampling operations by the architecture. As a result, the data scanning scheme, in comparison to the other schemes [32], requires less hardware resources for the control units and fewer registers for the stages.

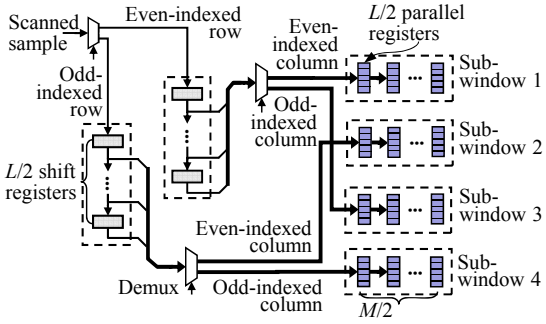

Fig. 5. Structure of the data scanning unit (DSU).

2) Distribution of filtering operations among the processing units employed by stages

In accordance with (1) and (4)), decomposing input data into four subbands requires four $L\times M$ filtering operations, and each of the four filtering operations requires four ($L/2\times M/2$)-tap filtering operations. Thus, a total of 16 ($L/2\times M/2$)-tap filtering operations are involved for the computation of the samples for the four subbands using an $L\times M$ window of the input data. Now, for each stage, these 16 types of filtering operations must be assigned to the processing units available to the stage using four sub-windows of data from its DSU. Given the available resources of the stages, the objective here is to process the 16 types of filtering operations with maximized computational parallelism and with priority given to the computation of the samples of LL subband.

In stage 1, since eight processing units are available, the processing task can be distributed among them so that one processing unit carries out the subtask of ($L/2\times M/2$)-tap filtering operations corresponding to a pair of subbands from the LL, LH, HL and HH using the data of one sub-window. One such distribution of the task is shown in Fig. 6, from which it is seen that each of the processing units PU_1 to PU_4 carries out the LL and LH filtering operations sequentially using the sub-windows 1 to 4, respectively, whereas each of the processing units PU_5 to PU_8 carries out the HH and HL filtering

operations using the same sub-windows. In stage 1, the LL and HH subband samples are produced in parallel in one clock cycle, whereas the LH and HL subband samples are produced in parallel in the next.

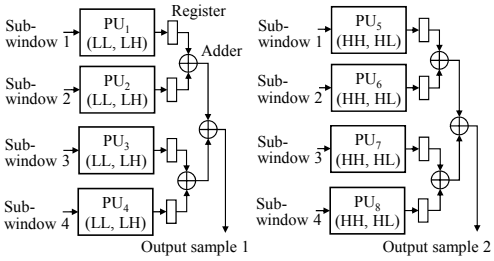

Fig. 6. The structure of eight processing units employed by stage 1.

Since stage 2 employs two processing units, each must perform the task of all the four subbands using two sub-windows. As the data of the four sub-windows, 1 to 4, become available in a sequential manner, sub-windows 1 and 3 are sequentially assigned to PU_9, whereas sub-windows 2 and 4 in a similar manner are assigned to PU_{10}. This distribution of the task for stage 2 is shown in Fig. 7, from which it is seen that each of the processing units, PU_9 and PU_{10}, carries out the (L/2×M/2)-tap filtering operations. In stage 2, PU_9 and PU_{10} operating in parallel produce the LL, LH, HH and HL subband samples sequentially in eight consecutive clock cycles.

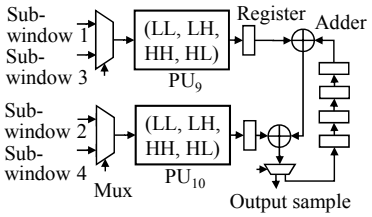

Fig. 7. The structure of two processing units employed by stage 2.

Since only one processing unit, PU_{11}, is employed by stage 3, it has to carry out all the filtering operations for each of the four sub-windows, as shown in Fig. 8. In this figure, the four sub-windows, 1 to 4, are chosen successively, as input to PU_{11}. For each sub-window, the processing unit PU_{11} then carries out the (L/2×M/2)-tap filtering operations. In this stage, PU_{11} produces sequentially the LL, LH, HH and HL subband samples in 16 consecutive clock cycles.

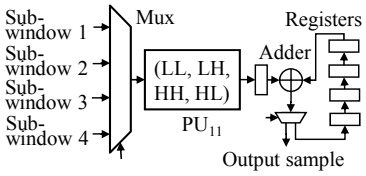

Fig. 8. The structure of one processing unit employed by stage 3.

Note that one processing unit at a time processes the samples of only one sub-window corresponding to one of the four subbands. Assume that such a processing time by a processing unit to be one time unit. Now, since stages 1, 2 and 3 have 8, 2 and 1 processing units, respectively, they can process sub-windows at the rates of 2, 1/2 and 1/4 sub-windows per unit time. This coupled with the fact that the processing loads (i.e. the number of sub-windows) assigned to the three stages are in the ratio 8:2:1, lets us to conclude that the operations of the three stages are mutually synchronized.

C. Design of the Processing Unit

In each stage, a processing unit carries out an (L/2×M/2)-tap filtering operation using the samples of an L/2×M/2 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 [33]−[36].

The filtering operation carried out by a processing unit, as described above, can be seen as L/2×M/2 parallel multiplications followed by an accumulation of the L/2×M/2 products. If the input samples and the filter coefficients have the wordlengths of A and B bits, respectively, then the processing unit produces an array of (B*L*M/4)×A 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 the 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 (2×2) from neighbouring 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 [36]. 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 unaccumulated, 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. Fig. 9 depicts a block diagram of a processing unit based on the above discussion.

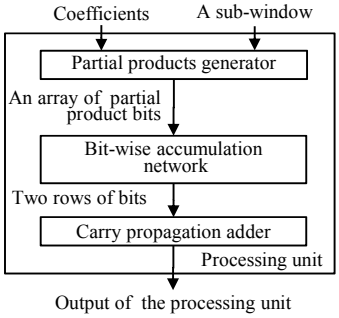

Fig. 9. Block diagram of a processing unit.

V. Performance Results and Comparisons

A. Performance of the Proposed Architecture

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. In this paper, the hardware resources used for the filtering operation are measured by the number of multipliers (N_{MUL}) and the number of adders (N_{ADD}), and that used for the storage of data and filter coefficients are measured by the number of registers (N_{REG}). The computation time, in general, is technology dependent. However, a metric that is technology independent and can be used to determine the computation time T is the number of clock cycles (N_{CLK}) elapsed between the first and the last samples inputted to the architecture. Assuming that one clock period is T_c, the total computation time can then be obtained as $T=N_{CLK}T_c$.

For a J-level 2-D DWT computation of an $N\times N$ image using $(L\times L)$-tap filters, the expressions for the metrics mentioned above for the proposed 3-stage architecture are given in Table II. It is seen from this table that the numbers of multipliers, adders and registers in the DSUs employed by the architecture depend only on the filter length, whereas the number of the registers of the buffers depends also on the image size.

In order to evaluate the performance of the proposed architecture in terms of T_c, we consider an example of designing a circuit for the DWT computation of an image of size N=512. For this purpose, we use 2-D filters of size $L=M$=4, wordlength for the filter coefficients as 8-bit, and the number of decomposition levels J=6. The input samples are encoded by using a radix-4 booth encoder and used as one of the two operands for the multiplication operation. All the carry propagation adders of the architecture have a 16-bit wordlength and use a structure that combines the carry-skip and carry-select adders [36]. The circuit is synthesized in RTL by using Synopsys with 0.18-μm CMOS technology. The synthesized results show that the circuit can operate with a minimum clock period of 6.5 ns (i.e. at a maximum clock frequency of 153 MHz). The circuit has a core area of 4.95×3.84 mm^2, and consists of 850K logic gates and a 24.5K-RAM. The power consumed by the circuit is obtained as 214 mW at 100 MHz clock frequency.

In order to validate the circuit design based on the proposed architecture, the circuit is implemented on a typical FPGA board, Virtex-II Pro XC2VP30-7. The board is capable of operating with a clock frequency of up to 400 MHz at a core voltage of V_{DD}=1.5 V. The resources utilized by the FPGA implementation in terms of the numbers of configuration logic block (CLB) slices, flip-flop slices, 4-input look-up tables (LUTs), input/output blocks (IOBs) and block RAMs (BRAMs) are given in Table III. The circuit implemented is found to perform well with a clock period as short as 7.4 ns (i.e. a maximum clock frequency of 134 MHz). The time for the DWT computation of an image of size 512×512 is 0.97 ms. In other words, the circuit is able to process motion pictures with a speed of 1022 frames per second (FPS). The power consumption of the FPGA device on which the circuit is implemented is measured to be 303 mW at 100 MHz clock frequency. This measured value for the power consumption compares reasonably well with the simulated value of 214 mW, considering that the measured value also includes the power dissipated by the unused slices within the FPGA device.

In order to validate the proposed architecture further, various circuits, which are designed based on the proposed architecture for the values of N=128, 256, 512, 1024, 2048 and J=3, 6, are implemented on the same type of FPGA board as used above. The implementation results for the various circuits are shown in Fig. 10. It is seen from this figure that the number of CLB slices (N_{CLB}) changes very slightly with the image size N or the number of decomposition levels J (Fig. 10(a)), while the number of BRAMs (N_{BRAM}) increases rapidly (Fig. 10(b)). These results are consistent with the performance evaluation results provided in Table II, and also demonstrate that the

TABLE II
Expressions for Metrics of the Proposed Architecture

N_{CLK}	N_{MUL}	N_{ADD}	N_{REG}	
			DSUs	Buffers
$N^2/2$	$11L^2/4$	$11\mathrm{Log}_2(L^2/2)+9$	$3L^2+3L$	$3NL/4+3N^2/128$

TABLE III
Resources Utilized in FPGA Device for the Circuit Implementation for the DWT Computation when N=512, $L=M$=4 and J=6

Resource	Number used	Percentage used
CLB Slices	2842	20%
Flip-flop Slices	1059	3%
4-input LUTs	4989	18%
Bonded IOBs	130	23%
BRAMs	8	5%

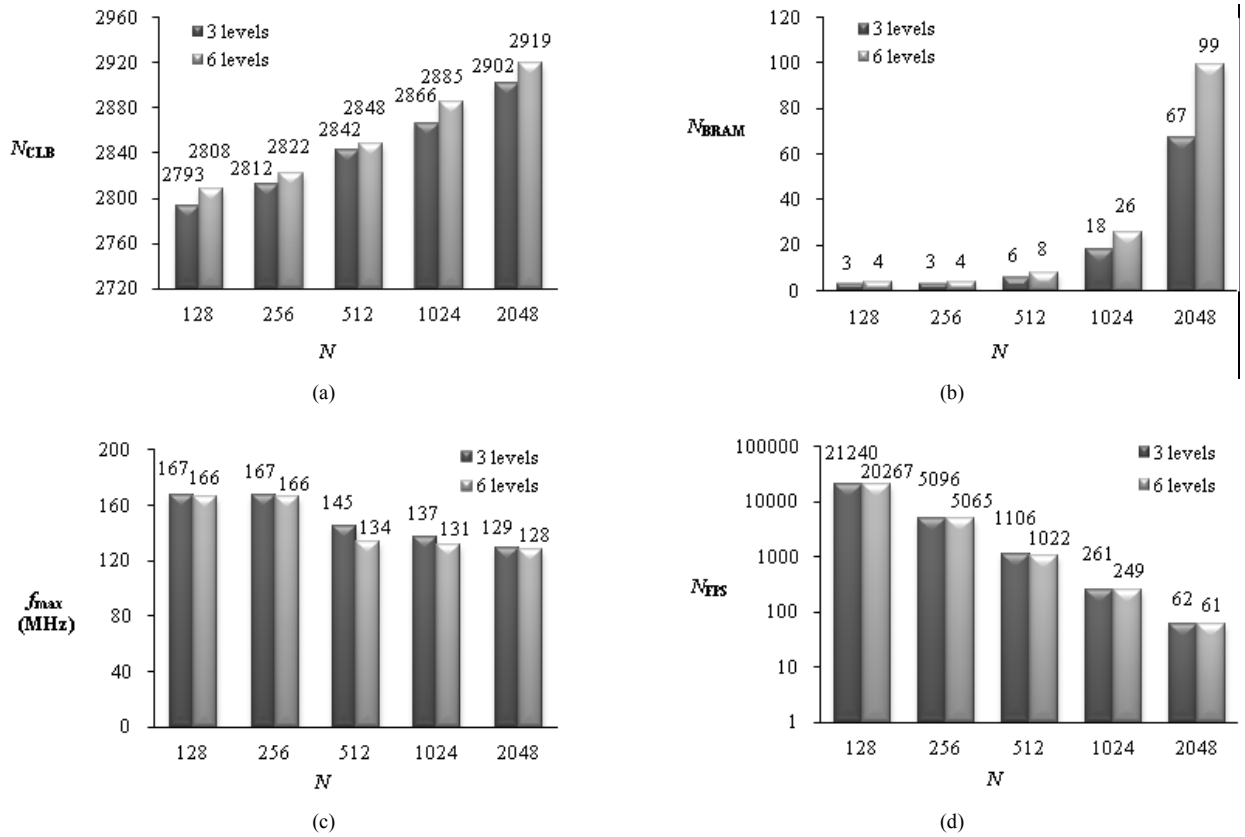

Fig. 10. Results of various FPGA implementations with N=128, 256, 512, 1024, 2048, and J=3, 6. (a) The numbers of CLB slices vs. N, (b) the numbers of BRAMs vs. N, (c) the maximum clock frequencies vs. N, and (d) the numbers of frames per second vs. N.

circuits for the DWT computation of images of different size and with different number of decomposition levels can be implemented essentially by varying the size of the buffer used. The performance of only a slight decrease in the maximum clock frequency (f_{max}) and that of a logarithmic decrease in the number of frames per second (N_{FPS}), as the image size increases (Fig. 10(c) and (d)), are in conformity with the normal expectation.

B. Comparisons of Various Architectures

In order to compare the hardware utilization and computation time of the proposed and other architectures, expressions for the relevant performance metrics for a J-level DWT computation of an $N\times N$ image using $(L\times L)$-tap filters for the various architectures are given in Table IV. It is seen from this table that the architecture of Prop. 4 in [18] and that of [21], require, respectively, $N^2/12$ and $N^2/4$ clock cycles, which are smaller than $N^2/2$ clock cycles required by the proposed architecture. This performance of [21] is achieved by utilizing the hardware resources of adders and multipliers that is more than twice of that required by the proposed architecture. Also, it is to be noted that in [18] the amount of the hardware resources (adders, multipliers and delay units) is larger than that required by the proposed architecture. Indeed, a smaller value of N_{CLK} does not necessarily mean a smaller computation time T, since the clock period T_c may significantly differ from one architecture to another. It is also seen from Table IV that the hardware utilization of the proposed architecture is higher than that of the pipeline architectures in [3], [9], [21] and [22], and it is only slightly lower than that of [18], in which 100% hardware utilization is achieved by using a much larger number of adders. Furthermore, the proposed architecture provides a shorter latency compared with the architectures in [3] and [8]-[10] that use 1-D type filters. On the other hand, the architectures in [18] and [21] provide smaller latencies, but employ proportionally larger hardware resources.

The performance of the proposed architecture is now compared with various other architectures in terms of the FPGA implementation results available in the literature. The FPGA implementation results for the architectures presented in [3], [8]–[11], [20] and [22] are listed in Table V. It is seen from this table that the implemented circuit for the proposed architecture requires a time of 0.97 ms to compute a 6-level DWT of an image of size 512×512, which is about one-half and one-third of the closest computation times offered by the implementations of the architectures of [20] and [10], respectively. In comparison to the architecture of [10], the proposed architecture provides this 3 times increase in the speed of computation at the expense only about 67% increase in the hardware. In comparison to the architecture of [20], the proposed architecture provides an improvement of 50% in the speed of computation while at the same times consumes about 35% less hardware resources. In order to have a fair

TABLE IV
EXPRESSIONS FOR METRICS OF VARIOUS ARCHITECTURES

Architecture	No. of multipliers	No. of adders	Storage size	Filter type	No. of clock cycles	Hardware utilization	Latency
Recursive architecture [3]	12	16	$4N$	1-D (9/7)	N^2+N	50%-70%	$T_c N^2$
Generic folded [8]	$6J$ (L/2)	$6J(1+\log_2(L/2))$	$4(L-1)N/3$	1-D	N^2	N/A*	$T_c N^2$
Symmetrically extended [9]	$L/2+L/4+L/8$	$2(L/2+L/4+L/8)$	$(L+0.5)N$	1-D	$1.5N^2$	87.5%	$1.5T_c N^2$
Parallel FDWT [10]	12	16	$3N/2$	1-D (9/7)	N^2	N/A	$T_c N^2$
Line-based [11]	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Parallel Prop. 4 [18]	96	240	$[4N+32J+256]$ (on chip delay units) $[8N+128(J-1)]$ (off chip buffer)	2-D (L=4)	$N^2/12$	100%	$T_c N^2/12$
Arch2D-II [20]	$L^2/2$	$L^2/2+L$	N/A	2-D	$2N^2/3$	N/A	$2T_c N^2/3$
Pipeline [21]	$6L^2$	$6L^2$	$2NL$	2-D	$N^2/4$	66.7%	$T_c N^2/4$
Parallel structure [22]	48	24	$6N/2+6N/4$ (J=3)	2-D (4×4)	$L^2N^2/16+L^2N/8$	5.6%	N/A
Proposed	$11L^2/4$	$11\log_2(L^2/2)+9$	$3L+3L^2$ (on chip delay units) $3NL/4+3N^2/128$ (off chip buffer)	2-D	$N^2/2$	96%	$T_c N^2/2$

*Not available

comparison with the non-separable architecture of [20], whose computation time is next best to that of the proposed architecture, we have implemented the latter also on Virtex 2000E. The implementation of the proposed architecture on this device results in a computation time of 1.4 ms and in 3430 used CLB slices. Thus, with the architecture of [20] and the proposed architecture implemented on the same FPGA device, the latter gives a 17% gain in the computational speed and 21% reduction in the hardware resources. Overall, the area-time product of the proposed architecture has a value that is at least 33% smaller than that of the other architectures.

VI. CONCLUSION

In this paper, a 3-stage pipeline architecture for a real-time computation of the 2-D DWT has been proposed. The objective has been to achieve a short computation time by maximizing the operational clock frequency ($1/T_c$) and minimizing the number of clock cycles (N_{CLK}) required for the DWT computation by developing a scheme for enhanced inter-stage and intra-stage computational parallelism for the pipeline architecture.

To enhance the inter-stage parallelism, a study has been undertaken that suggests that, in view of the nature of the DWT computation, it is most efficient to map the overall task of the DWT computation to only three pipeline stages for performing the computation tasks corresponding to the decomposition level 1, level 2, and all the remaining levels, respectively. Two parameters, one specifying the synchronization of the operations of the stages and the other representing the utilization of the hardware resources of the pipeline, have been defined. It has been shown that the best combination for the value of these parameters is achieved when the pipeline is chosen to have three stages. In order to enhance the intra-stage

TABLE V
COMPARISON OF VARIOUS FPGA IMPLEMENTATIONS

Architecture	Image size (N)	No. of CLB slices	RAM size (bits)	f_{max} (MHz)	Time (ms)	Area×Time*	Device
Recursive architecture [3]	512	879	$10N$	50	5.3	4659	XC2V250
Generic folded [8]	256	4720	10×(4K)	75	0.874	4125	Virtex 600E-8
Symmetrically extended [9]	512	2559	17×(18K)	44.1	9	23031	XC2V500
Parallel FDWT [10]	512 (J=5)	1700	$3N/2$	171.8	3.1	5270	Virtex 2
Line-based [11]	512 (J=6)	2950	4×(18K)	113.6	5.2	15340	XC4VLX15
Parallel Prob. 4 [18]	Implementation results not available						
Arch2D-II [20]	512	4348	24×(18K)	105	1.7	7392	Virtex 2000E
Pipeline [21]	Implementation results not available						
Parallel structure [22]	512	3580	2304	45	5.9	21122	XCV600E
Proposed	512 (J=6)	2842	8×(18K)	135	0.97	2757	XC2VP30

*The value of area in the calculation of area-time product is replaced by the No. of CLB slices since the former is proportional to the latter.

parallelism, two main ideas have been employed for the internal design of each stage. The first idea is to divide the 2-D filtering operation into four subtasks that perform independently and simultaneously on the elements of even or odd indexed rows and columns of the 2-D input data. This idea stems from the fact that for each consecutive decomposition level, the input data are decimated by a factor of two along the rows and columns of the 2-D data. Each subtask of the filtering operation is performed by a processing unit. The second idea employed is in organization of the array of bit-wise adders, which is the core of the processing unit, in a way so as to minimize the delay of the critical path from a partial product input bit to a bit of an output sample through this array. In this paper, this has been accomplished by minimizing the number of layers of the array while at the same time minimizing the delay of each layer.

In order to validate the proposed scheme, a circuit for the DWT computation has been designed, simulated and implemented in FPGA. The circuit is designed for a filter length $L=M=4$ and simulated for the number of the decomposition levels $J=6$ and data size $N\times N=512\times512$. The simulation results have shown that the circuit designed based on the proposed scheme is able to operate at a maximum clock frequency $f_{max}=153$ MHz. The results of the FPGA implementation have shown that the circuit can process a 512×512 image in 0.97 ms, which is at least two times faster than that of the other FPGA implementations, and in some instances, even with less hardware utilization. Finally, it is worth noting that the architecture designed in this paper is scalable in that its processing speed can be adjusted upward or downward by changing the number of MAC cells in each of the processing units by a factor equal to that of the reduction required in the processing speed.

REFERENCES

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

[2] M. Vishwanath, R. Owens, and M. J. Irwin, "VLSI architectures for the discrete wavelet transform," *IEEE Trans. Circuits Syst. II*, vol. 42, no. *5,* pp. 305−316, May 1995.

[3] H. Y. Liao, M. K. Mandal, and B. F. Cockburn, "Efficient architectures for 1-D and 2-D lifting-based wavelet transforms," *IEEE Trans. Signal Process*., vol. 52, no. 5, pp. 1315−1326, May 2004.

[4] P. Wu and L. Chen, "An efficient architecture for two-dimensional discrete wavelet transform," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 4, pp. 536–545, Apr. 2001.

[5] S. Masud and J.V. McCanny, "Reusable silicon IP cores for discrete wavelet transform applications," *IEEE Trans. Circuits Syst. I*, vol. 51, no. 6, pp. 1114−1124, Jun. 2004.

[6] T. Huang, P. C. Tseng, and L. G. Chen, "Generic RAM-based architectures for two-dimensional discrete wavelet transform with line-based method," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 7, pp. 910–920, Jul. 2005.

[7] P. K. Meher, B. K. Mohanty, and J. Chandra Patra, "Hardware-efficient systolic-like modular design for two-dimensional discrete wavelet transform," *IEEE Trans. Circuits Syst. II*, 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, and M. 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*, 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: Express 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 2D biorthogonal DWT", *Journal of VLSI Signal Processing*, vol. 40, pp. 333−342, 2005.

[17] C. Chakrabarti and M. Vishwanath, "Efficient realizations of the discrete and continuous wavelet transforms: from single chip impelmentations 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 Integration* (*VLSI*) *Systems*, vol. 9, no. 5, pp. 565–576, Oct. 2001.

[20] I. S. Uzun and A. Amira, "Rapid prototyping -- framework for FPGA-based discrete biorthogonal wavelet transforms implementation," *IEE Vision, Image and Signal Processing*, 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*, vol. 47, no. 12, pp. 1476–1491, Dec. 2000.

[22] R. J. C. Palero, R. G. Gironez, and A. S. Cortes, "A novel FPGA architecture of a 2-D wavelet transform", *Journal of VLSI Signal Processing*, 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.*, vol. 2, 1997, 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 and 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. Intelligent & Advanced Systems* (*ICIAS*), Kualalampur, Malaysia, Nov. 2007, pp.1355−1358.

[27] C. Yu and S.-J. Chen, "VLSI implementation of 2-D discrete wavelet transform for real-time video signal processing," *IEEE Trans. Consumer Electronics*, vol. 43, no.4, pp. 1270−79, 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*, 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, U.S. Patent 6976046, Architectures for discrete wavelet transforms, December 13, 2005.
[32] J. Song and I. Park, "Pipelined discrete wavelet transform architecture scanning dual lines," *IEEE Trans. Circuits Syst. II: Express 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 and Systems* (*ISCAS*), May 2007, pp. 3980−3983.
[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 1D discrete wavelet transform," in *Proc. IEEE Int. Symp. Circuits and Systems* (*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*, vol. 57, no. 10, pp. 2729−2740, Oct. 2010.

Chengjun Zhang received the B.S degree and M.S. degree in Physics from Nanjing University, Nanjing, Jiangsu, China, in 1994 and 1997, respectively. He is working toward the Ph.D. degree in the Department of Electrical and Computer Engineering at Concordia University, Montreal, QC, Canada.

His research interests include signal processing, architecture design, and VLSI implementation of digital systems.

Chunyan Wang received the B. Eng. degree in electronics from JiaoTong University, Shanghai, China, and the M. Eng. and Ph.D. degrees from Universite´ Paris Sud, Paris, France.

She joined Concordia University, Montreal, QC, Canada, in 1997, as an Assistant Professor, where she is presently an Associate Professor of Electrical and Computer Engineering.

Her current research areas are low-power analog- and mixed-signal VLSI design, CMOS sensor integration, and VLSI implementation of digital signal processing systems.

M. Omair Ahmad (S'69-M'78-SM'83-F'01) received the B.Eng. degree from Sir George Williams University, Montreal, QC, Canada, and the Ph.D. degree from Concordia University, Montreal, QC, Canada, both in electrical engineering.

From 1978 to 1979, he was a member of the Faculty of the New York University College, Buffalo. In September 1979, he joined the Faculty of Concordia University as Assistant Professor of Computer Science. Subsequently, he joined the Department of Electrical and Computer Engineering, Concordia University, where he was the Chair of the department from June 2002 to May 2005 and is presently a Professor. He holds the Concordia University Research Chair (Tier I) in Multimedia Signal Processing. He has published extensively in the area of signal processing and holds four patents. His current research interests include the areas of multidimensional filter design, speech, image and video processing, nonlinear signal processing, communication DSP, artificial neural networks, and VLSI circuits for signal processing. He was a Founding Researcher at Micronet from its inception in 1990 as a Canadian Network of Centers of Excellence until its expiration in 2004. Previously, he was an Examiner of the Order of Engineers of Quebec.

Dr. Ahmad was an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS PART I: FUNDAMENTAL THEORY AND APPLICATIONS from June 1999 to December 2001. He was the Local Arrangements Chairman of the 1984 IEEE International Symposium on Circuits and Systems. During 1988, he was a member of the Admission and Advancement Committee of the IEEE. He has also served as the Program Co-Chair for the 1995 IEEE International Conference on Neural Networks and Signal Processing, the 2003 IEEE International Conference on Neural Networks and Signal Processing, and the 2004 IEEE International Midwest Symposium on Circuits and Systems. He was General Co-Chair for the 2008 IEEE International Conference on Neural Networks and Signal Processing. Presently, he is the Chair of the Montreal Chapter IEEE Circuits and Systems Society. He is recipient of numerous honors and awards, including the Wighton Fellowship from the Sandford Fleming Foundation, an induction to Provost's Circle of Distinction for career achievements, and the award of Excellence in Doctoral Supervision from the Faculty of Engineering and Computer Science of Concordia University.