An Efficient VLSI Architecture and FPGA Implementation of High-Speed and Low Power 2-D DWT for (9, 7) Wavelet Filter

A. Mansouri, A. Ahaitouf, and F. Abdi.
UFR SSC, LSSC, Electrical Engineering Department Faculty of sciences & techniques BP: 2202 FES MOROCCO

Summary
This paper presents an efficient VLSI architecture of a high speed, low power 2-D Discrete Wavelet Transform computing. The proposed architecture, based on new and fast lifting scheme approach for (9, 7) filter in DWT, reduces the hardware complexity and memory accesses. Moreover, it has the ability of performing progressive computations by minimizing the buffering between the decomposition levels. The system is fully compatible with JPEG2000 standard. Our designs were realized in VHDL language and optimized in terms of throughput and memory requirements. The implementations are completely parameterized with respect to the size of the input image and the number of decomposition levels. The proposed architecture is verified by simulation and successfully implemented in a Cyclone II and Stratix III FPGAs, and the estimated frequency of operation is 350 MHz. The resulting computing rate is up to 48 frames (4096x2160) per second with 24 bpp. The architecture has regular structure, simple control flow, small embedded buffers and low power consumption. Thus, it is very suitable for new generation image compression systems, such as JPEG2000.

Key words:
JPEG2000; 2D-DWT; VLSI architecture; FPGA implementation.

1. Introduction

Over the past several years, the wavelet transform has gained widespread acceptance in signal processing in general and in image compression research in particular. In applications such as still image compression, Discrete Wavelet Transform (DWT) based schemes have outperformed other coding schemes like the ones based on Discrete Cosine Transform (DCT). The DWT has been introduced as a highly efficient and flexible method for sub band decomposition of signals [1]. The two-dimensional DWT (2D-DWT) is nowadays established as a key operation in image processing. This is due to the fact that DWT supports features like progressive image transmission (by quality, by resolution), ease of compressed image manipulation, region of interest, etc. In addition to image compression, the DWT has important applications in many areas, such as computer graphics, numerical analysis, radar target distinguishing and so forth. The high algorithmic performance of the 2D DWT in image compression justifies its use as the kernel of both the JPEG2000 still image compression standard [1] and the MPEG-4 texture coding standard [2]. It is widely recognized that the LeGall (5, 3) and the Daubechies (9, 7) filters are among the best filters for DWT-based image compression [3]. In fact, the JPEG2000 image coding standard [1] employs the (5, 3) and the (9, 7) filters as the default wavelet filters for respectively loss and lossy compression. The JPEG2000 can compress images 100 times smaller than the original image. With this compression ratio, the reconstructed image of the JPEG2000 still provides good visual quality. The coding efficiency of the JPEG2000 comes with the cost. Several years passed by since the JPEG2000 standard was approved in 2002. However, there are not many consumer products that support most features of the JPEG2000 available today. The real-time constraint and cost effectiveness are still major issues for the realization of the JPEG2000 into consumer products. The 2D-DWT is one of the main resources intensive components of JPEG2000; it demands massive computations and represents one of the critical parts in the design and implementation of the JPEG2000 standard. Hence, it requires a parallel and pipelined architecture to perform real-time or on-line video and image coding and decoding, and to implement high-efficiency application-specific integrated circuits (ASIC) or field programmable gate array (FPGA).

Up to now, much work has been performed on DWT theory and many VLSI architectures have been proposed. Mallat combined the Wavelet transform and filter bank into a single transformation [3]. Doubechies applies DWT to image coding and proposed many famous wavelet filters [4] including the (9, 7) filter. Swendens proposed the Lifting Scheme [5] making DWT more computationally efficient. Calderbank, Doubechies and Swendens later proposed the Integer Wavelet Transform (IWT) [6] that is more efficient without scarifying the performance. In addition, several VLSI architectures have been proposed for computing the 2D-DWT. They are mainly based on convolution scheme and lifting scheme. The lifting scheme can reduce the computational complexity by exploiting the similarities between high and low pass filters and it usually requires fewer multipliers and adders than the convolution scheme. Some architecture [7], [8], [9] and [10] for DWT have been

Manuscript received March 5, 2009
Manuscript revised March 20, 2009

proposed to meet the real time requirement in many applications for convolution scheme. The first architecture, presented by Knowles [7], uses many large multiplexers for storing intermediate results. Wu and Chen proposed a 2-D architecture that employs a folding technique [8]. Yu et al. proposed a 2-D architecture that employs a computation-schedule table [9]. Masud et al. proposed an efficient architecture implemented by filter banks [10]. Xixin Cao et al. presented an efficient VLSI implementation of Distributed Architecture for DWT in order to minimize area requirement, but they have a computation time which is proportional to input data N [11]. More recently, several architectures [12] [13] [14] [15]-[19] have been proposed for efficient computation of DWT based on the lifting scheme. Andra et al. proposed a 2-D DWT architecture which composes of simple processing units and computes one stage of DWT at a time [12]. Dillen et al. presented a combined architecture for the (5,3) and (9, 7) transforms with minimum area [13]. Wu and Lin proposed the fast pipeline architecture by merging two equations into one equation [14]. Huang et al. proposed a flipping structure by using an efficient VLSI architecture for lifting-based discrete wavelet transform [15]. S. V. Silva and S. Bampi presented Area and throughput trade-offs in the design of pipelined discrete wavelet transform architectures [16]. Martina, and Masera proposed the low-complexity and efficient (9, 7) wavelet filters VLSI implementation [17].

In this paper, we propose a pipeline, high performance; low power and lifting based architecture design for the 2D-DWT. The default lossy filter of the latest image compression standard JPEG2000, Daubechies (9, 7) is implemented on FPGA-based platforms and compared in terms of performance, area and power consumption. The advantages of the proposed architectures are 100% hardware utilization, small embedded buffers, regular structure, simple control flow and low power.

The rest of the paper is structured as follows. Section 2 summarizes the discrete wavelet transform and the three main important design architectures of 2D-DWT are discussed in detail along with a comparison among them in terms of the number of access to the external memory and the size of the local memories. In section 3, the high efficient architecture for the (9, 7) filter 2-D lifting-based DWT is proposed, followed by the implementation and performance analysis in section 4, and section 5 concludes the work.

2. Discrete Wavelet Transform

One of the prominent features of JPEG2000 standard, providing it the resolution scalability [3], is the use of the two-dimensional Discrete Wavelet Transform (2D-DWT) to convert the image samples into a more compressible form. It is considered as the key difference between JPEG and JPEG2000 standards. Since there is no need to divide the input image into non-overlapping 2-D blocks and its basis functions have variable length, wavelet-coding schemes at higher compression ratios avoid blocking artifacts. Hence the compression artifacts are dispersed over a correspondingly larger area, and reducing the visual impact.

2.1 One-Dimensional Discrete Wavelet Transform

Two main methods exist for the implementation of 1D-DWT: the traditional convolution-based implementation [18] and the lifting-based implementation [5, 12].

2.1.1 Convolution Based DWT

In the traditional implementation of DWT, a pair of finite impulse response filters (FIR) is applied in parallel, high-pass and low-pass filter. Each filtering operation is shown in Fig. 1.

Mallat's pyramid algorithm [18] computes the one dimensional (1-D) convolution based DWT at different levels of resolution. The first level decomposition can be represented by using the block diagram illustrated in Fig.1.

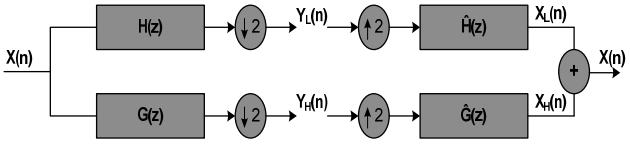

Fig. 1 Single 1D-DWT Block.

The input sequence X(n) in Fig. 1 is convolved with the quadrature mirror filters H(z) and G(z) and the outputs obtained are decimated by a factor of two. After down-sampling, alternate samples of the output sequence from the low pass filter and high pass filter are dropped. This reduces the time resolution by half and conversely doubles the frequency resolution by two. The 1D-DWT is a two-channel sub-band decomposition of an input signal X(n) that produces two sub-band coefficients YL(n) and YH(n) for one-stage of decomposition [18] according to the following equations.

$$Y_L(n) = \sum_{i=0}^{\tau_L - 1} H(i)x(2n-1) \qquad (1)$$

$$Y_H(n) = \sum_{i=0}^{\tau_H - 1} G(i)x(2n-1) \quad (2)$$

In the synthesis stage, scaling and wavelet coefficients $Y_L(n)$ and $Y_H(n)$ are treated inversely by up-sampling and filtering with low pass $\hat{H}(z)$ and high pass $\hat{G}(z)$ filters to perform reconstruction. This stage is also called Inverse Discrete Wavelet Transform (IDWT). Original and reconstructed signals are generally different, unless the two filters H and G satisfy some relationships [18]. The perfect reconstruction condition consists in ensuring no distortion and no aliasing of the reconstructed data. Early research on filter-bank design proved that the execution of 1D-DWT can be accelerated by using the polyphase matrix of the filter-bank, instead of the conventional filtering and down-sampling structure of Fig. 1. As Fig. 2 shows, the signal is split into two signals (polyphase components) at half of the original sampling rate. The polyphase components of the signal are filtered in parallel by the corresponding filter coefficients, producing the same result as if the down-sampling was performed as described in [18].

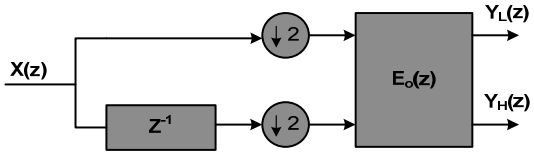

Fig. 2 The convolution-based implementation of the 1D-DWT by using the polyphase matrix.

The analysis polyphase matrix $E_0(Z)$ in Fig. 2 is defined (in the Z-domain) as:

$$E_0(z) = \begin{bmatrix} H_e(z) & H_o(z) \\ G_e(z) & G_o(z) \end{bmatrix} \quad (3)$$

Where $H_e(z)$ and $H_o(z)$ denote the even and odd polyphase components of the corresponding low-pass analysis filter, and $G_e(z)$ and $G_o(z)$ denote the even and odd polyphase components of the corresponding high-pass analysis filter. The wavelet decomposition can be written using Eq.3 (in the Z-domain) as:

$$\begin{bmatrix} Y_L(z) \\ Y_H(z) \end{bmatrix} = E_o(z) \begin{bmatrix} X_e(z) \\ X_o(z) \end{bmatrix} \quad (4)$$

Where $Y_L(z)$ denotes the approximation at the coarser resolution, $Y_H(z)$ denotes the detail signal, and $X_e(z)$ and $X_o(z)$ denote the even and odd polyphase components of the signal $X(z)$.

2.1.2 Lifting based DWT

The convolution-based 1-D DWT requires both a large number of arithmetic computations and a large memory for storage. Such features are not desirable for either high-speed or low-power image processing applications. Recently, a new mathematical formulation for wavelet transformation has been proposed by Swelden [5] as a light-weighted computation method for performing wavelet transform. The main feature of the lifting-based wavelet transform is to break-up the high pass and the low pass wavelet filters into a sequence of smaller filters. The lifting scheme requires fewer computations compared to the convolution-based DWT. Therefore the computational complexity is reduced to almost a half of those needed with a convolution approach [3] [5]. As a result, lifting has been suggested for implementation of DWT in JPEG2000 standard. The lifting-based wavelet transform basically consists of three steps, which are called split, lifting, and scaling, respectively, as shown in Fig. 3.

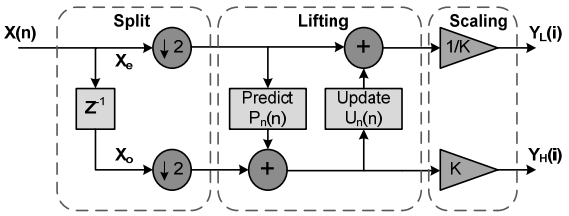

Fig. 3 The lifting scheme implementation of the 1D-DWT.

The basic idea of lifting scheme is first to compute a trivial wavelet (or lazy wavelet transform) by splitting the original 1-D signal into odd and even indexed subsequences, and then modifying these values using alternating prediction and updating steps. The lifting scheme algorithm can be described as follow:

- *Split step:* The original signal, *X(n)*, is split into odd and even samples (lazy wavelet transform).

- *Lifting step:* This step is executed as N sub-steps (depending on the type of the filter), where the odd and even samples are filtered by the prediction and update filters, $P_n(n)$ and $U_n(n)$.

- *Normalization or Scaling step:* After N lifting steps, a scaling coefficients K and 1/K are applied respectively to the odd and even samples in order to obtain the low-pass band ($Y_L(i)$), and the high-pass sub-band ($Y_H(i)$). Fig. 4 illustrates how the lifting scheme can be implemented using these steps. The diagram shows the lifting scheme for Daubechies (9, 7) biorthogonal filter adopted in JPEG2000 standard for lossy compression [1]. The lifting scheme algorithm to the (9,7) filter is applied as:

Split step:

Xe ← X(2i) *Even Samples*

Xo ← X(2i+1) *Odd Samples*

Lifting Steps:

For (9, 7) filter, N=2

Predict P1: $D(i) = X_o(i) + a [X_e(i) + X_e(i+1)]$

Update U1: $S(i) = X_e(i) + b [D(i-1) + D(i)]$

Predict P2: $Y_H(i) = D(i) + c [S(i) + S(i+1)]$

Update U2: $Y_L(i) = S(i) + d [Y_H(i-1) + Y_H(i)]$

Scaling Step:

$$Y_H(i) = K\, Y_H(i)$$

$$Y_L(i) = 1/K\, Y_L(i)$$

Where a=-1.586134342, b=-0.0529801185, c=0.882911076, d=-0.443506852, and K=1.149604398 [1].

These mathematical equations can be illustrated by the scheme in Fig. 4.

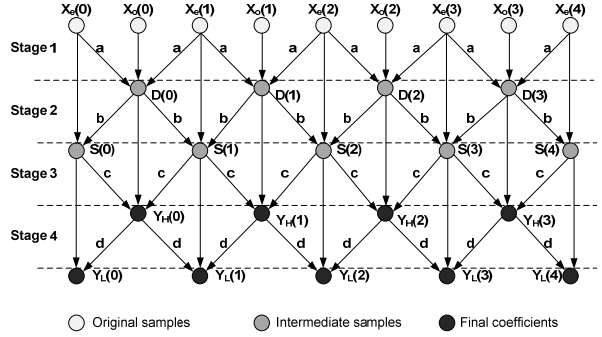

Fig. 4 The diagram of 1-D DWT using lifting scheme for (9, 7) filter.

To compare the complexity of the convolution and the lifting approaches, we have performed a software implementation in C++ language of both methods for (5, 3) and (9, 7) filters. Results of the simulation are presented in table 1. This table shows the number of multiplications, additions and shifts needed for (5, 3) and (9, 7) for both methods. Simulation was performed by using Lena image (512x512).

Table 1 Complexity comparison of convolution and lifting-based implementation.

		Multiplication	Addition	Shift
(9, 7) filter	Convolution	3670216	4194504	None
	Lifting	1579108	2109540	None
(5, 3) filter	Convolution	490	2890	3340
	Lifting	None	1940	1420

This comparison as well as others works [19] reveals that lifting-based DWT requires less computation than convolution-based one. Consequently, convolution-based DWT is computationally extensive and resulting to be area, power, and memory hungry. Lifting scheme reduces the computations up to 50%, which affect directly the memory, surface and the power consumption of the system. To sum up, lifting scheme will be more suitable for hardware implementation with limited on-ship memory, lower computational complexity, small area and low power.

2.2 Two-Dimensional Discrete Wavelet Transform

The basic idea of 2-D architecture is similar to 1-D architecture. A 2-D DWT can be seen as a 1-D wavelet transform along the rows and then a 1-D wavelet transform along the columns, as illustrated in Figure 5. The 2-D DWT operates in a straightforward manner by inserting array transposition between the two 1-D DWT. The rows of the array are processed first with only one level of decomposition. This essentially divides the array into two vertical halves, with the first half storing the average coefficients, while the second vertical half stores the detail coefficients. This process is repeated again with the columns, resulting in four sub-bands (see Fig. 5a) within the array defined by filter output. Fig. 5b shows a three-level 2-D DWT decomposition of the Lena image.

The LL sub-band represents an approximation of the original image, the LL1 sub-band can be considered as a 2:1 sub-sampled (both horizontally and vertically) version of the original image. The other three sub-bands HL1, LH1, and HH1 contain higher frequency detail information (mostly local discontinuities in the edges of the image). This process is repeated for as many levels of decomposition as are desired. The JPEG2000 standard specifies five levels of decomposition [1], although three are usually considered acceptable in hardware. In order to extend the 1-D filter to compute 2-D DWT in JPEG2000, two points have to be taken into account:

Firstly, the 1-D DWT generates the control signal memory to compute 2-D DWT and manages the internal memory access. Secondly; we need to stores temporary results generated by 2-D column filter. The amount of the external memory access and the area occupied by the embedded internal buffer are considered the most critical issues for the implementation of 2D-DWT. In other words, the design trade-off mainly comes from the external memory access bandwidth and the internal buffer size. As the cache is used to reduce the main memory access in the general processor architectures, in similar fashion, the internal buffer is used to reduce the external

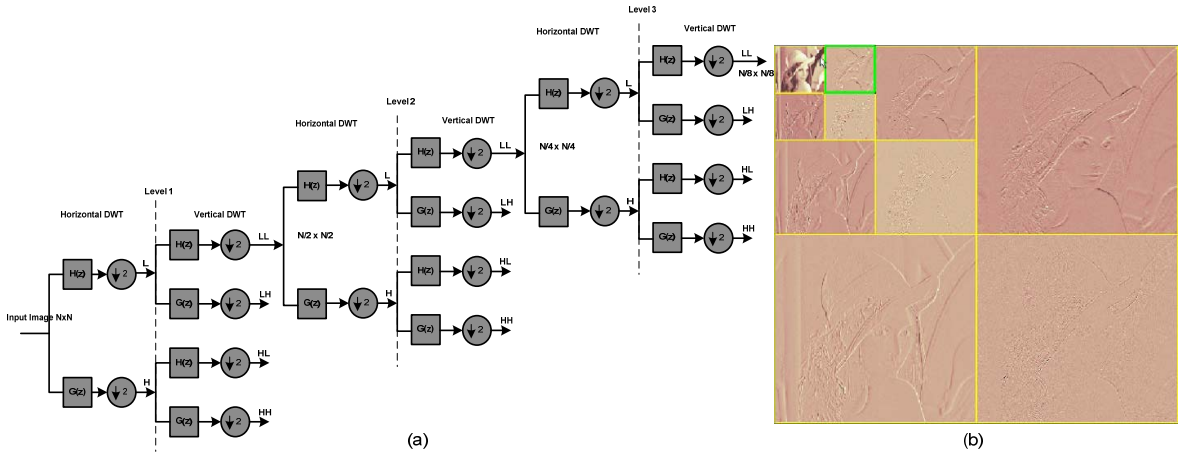

Fig. 5 Three-level decomposition algorithm for 2-D DWT and Lena image decomposition.

memory access for 2D-DWT However, the internal buffer would occupy much area. Three main architecture design approaches were proposed in the literature with the aim to implement efficiently the 2D-DWT [20]: level by level, block-based, and line-based architecture. These architectures address this difficulty in different ways. A typical level-by-level architecture uses a single processing module that first processes the rows, and then the columns. Intermediate values between row and column processing are stored in memory; since this memory must be large enough to keep wavelet coefficients for the entire image, external memory is usually used. Access to the external memory is sometimes done in row-wise order, and sometimes in column-wise order, so high-bandwidth access modes cannot be used. As a result, external memory access can become the performance bottleneck of the system. In block-based architecture, the image is broken into blocks small enough to fit in an embedded memory that is processed separately. A typical Block-based architecture scans the external memory block-by-block, and the DWT coefficients are also computed block-by-block. To perform the block-based wavelet transform, it is necessary to store into memory an additional row of coefficients and one additional column. The additional row is located at the top of the block, and the additional column is located to the left of the block, as illustrated in Fig. 6. As a result, the input block has a size of (N +1) x (N +1) pixels. At the top and to the left of the image, the additional row or column is extracted from the extended signal [20]. In a block-based approach, the filtering of the image boundaries should be taken into account. The treatment of this filtering has a potential impact on the visual artifacts near the boundaries.

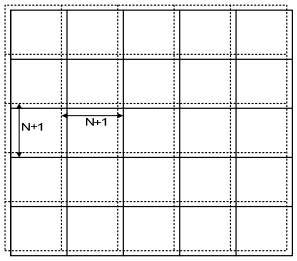

Fig. 6 Image divided into input blocks.

Both line-based and block-based approaches have been proposed to improve the issues of memory usage and memory-access of the conventional level-by-level approach. A line-based architecture scan input image row-by-row manner to produce the wavelet coefficients. However, a block-based architecture scans the input image block-by-block and produces the wavelet coefficients for each block. Consequently, the main difference between the two methods is the selected image traversal method (based on complete image rows or on blocks). The line-based architecture needs only few lines of the image to be stored, whereas traditional methods almost need the whole image (or tile) to be memorized. Thus, this technique does not require extra memory or external memory to store the intermediate data. Instead, some internal buffers are used to store the intermediate data, and the required memory size is proportional to image width or height [20]. There have been several line-based architectures proposed for the convolution-based hardware implementation of 2-D DWT, such as [8], and also for lifting-based 2-D DWT implementation [12] [21] [22]. The line-based architecture uses local memories whereas it increases the processing speed, reduces the memory access, the complexity of the control system and the address generator. In the present work, we propose an architecture which adopt the line-based structure, it is similar to the level by level

architecture. We use an efficient line buffer scheme to store intermediate data and respect the real-time processing constraint. Our goal mainly focuses on the high-performance, low power consumption and hardware size.

3. Proposed architecture

The proposed architecture uses the popular Daubechies (9, 7) filter used in JPEG2000 [1] [3]. This architecture is a pipeline and memory efficient based on the lifting scheme. It improves the implementation of the 2D-DWT by adopting an efficient usage of hardware resources, low control complexity; reducing the embedded memory requirements and external memory access. We exploit the data dependency of the lifting scheme technique and propose a new design of 1D-DWT architecture. The key idea consists of pipelining and interleaving the operations between row and column processing to increase throughput and reduce latency. The architecture minimizes the number of external memory access, reduces the power consumption and employs small embedded memory for intermediate data storage. The proposed hardware architecture is shown in Fig. 7. It calculates the 2D-DWT in row-column fashion on the input image. The row filter calculates the DWT of each row of the external memory image data. Then, the resulting decomposed high-pass and low-pass coefficients are stored in intermediate buffers, and the column filter calculates the vertical DWT as soon as there are sufficient coefficients generated by the row filter. The architecture framework is composed of the following parts: two 1D-DWT blocks, internal buffers, LL FIFO used for multilevel decomposition, Address generator block and Controller block.

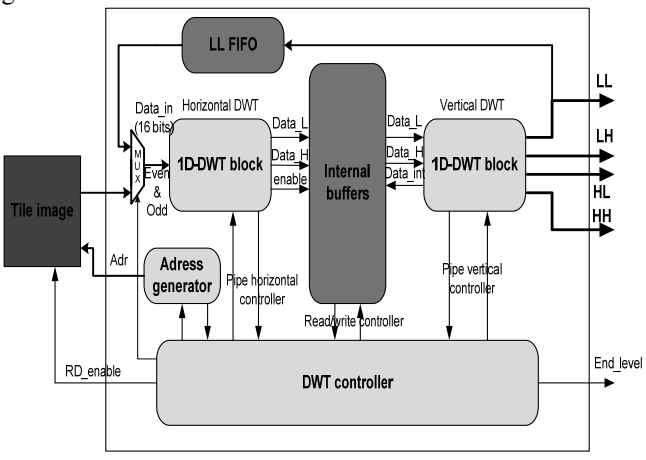

Fig. 7 The block diagram of the proposed 2D-DWT architecture.

3.1 1D-DWT block

In the beginning of the transform, let's focus on how many bits are needed to preserve the precision.

3.1.1 The bit precision

The input to the 1D-DWT block is the YCrCb component stored in the tile memory input. Each pixel of a YCrCb component is about 8 bits or 9 bits. The (9, 7) filter is called "irreversible" because the filter is defined using irrational numbers [3]. In general it is not possible within a finite precision floating-point architecture to guarantee reversibility. Thus the (9, 7) filter is suitable only for lossy compression applications [1] [3]. By tracing the expression of the (9, 7) filter forward DWT, we find that the output could be 8.265234 times larger than the input in the worst case. Therefore, before prototyping the 2D-DWT onto FPGA, intensive simulations have been done to verify the design. Floating point implementation proves very useful in helping us to conceptualize our design. It also provides a solid base to compare the results of the fixed point and hardware implementations. Hence, it is worthwhile to develop the architecture in floating point implementation using a high level language such as C language, then translated into fixed-point for precision analysis before RTL implementation. The development of the 2-D DWT floating point algorithm uses a configuration with the flowing parameters: (9, 7) lossy filter, five levels wavelet decomposition and the size of the test image (Baboon, Jet and Lena) is 512 x 512 pixels. For an 8-bit decompressed image, PSNR is defined as:

$$10\log_{10}\left(\frac{255^2}{MSE}\right) \quad (5)$$

Where MSE refers to the Mean Squared Error between the original image and the reconstructed image. As shown in Fig. 8, the PSNR (dB) value of three test images under different filter coefficient word lengths is presented.

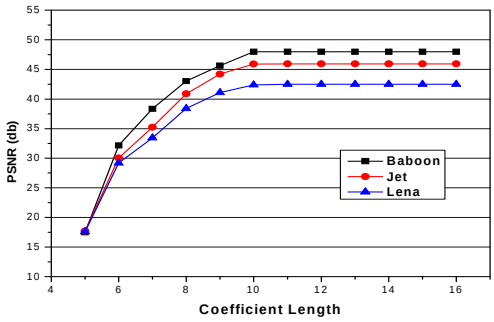

Fig. 8 The coefficient length of Daubechies (9, 7) filters versus PSNR for three commonly used images.

We learned from it that using 8 bits for the fraction bits has great quality and concluded that the PSNR saturates at 10-bit. The filter coefficients for the (9, 7) filter considered range from 0.05298 ~ 1.58613. In order to convert the filter coefficients to integer, the coefficients are multiplied with 256 (i.e. shift left 8 bits). The range of the coefficients is now 14 to 407, which implies that the coefficients require 10 bits available to be represented in 2's complement form. Then we consider the product after the end of multiplication, the product will round off the eight significant bits. Hence, we select 12 bits to represent filter coefficients. The input signals are shifted left to decrease the normalization errors. An extension bit is introduced as the number of shifting bits. Fig. 9 shows that the 16-bit signal width with extension bit equal to 5 satisfies both the required precision.

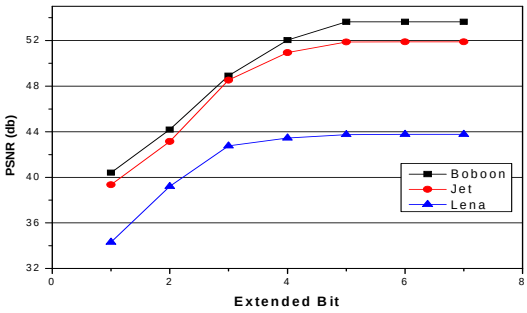

Fig. 9 The Extended bit of the input signal versus PSNR for three commonly used images.

Experiments show that the average mean square error of the reconstructed images equals 1.54E-5, compared with the result in [13], 4.83E-3. Therefore, our design improves the data image accuracy.

3.1.2 Symmetric Extension

The extension of the signal is needed to enable filtering at both boundaries of the signal. This problem can be solved by extending the signal at the boundaries as much as needed to complete the filtering operation. In the proposed filter's design the symmetric extension is adopted. The symmetric extension suggested in JPEG2000 is a simple method for extending a finite length signal [1] [3]. In this method the signal is extended so as it becomes periodic and symmetric. An example of the one row signal symmetric extension is shown in Fig. 10.

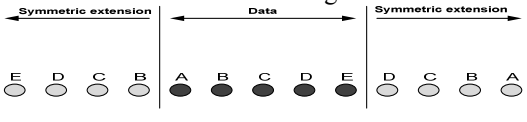

Fig. 10 The signal symmetric extension.

The filter's operation consists of three phases:

Initialization phase: initially, the filter loads its inputs with the appropriate number of samples in order to start the filtering. Since these samples are present in the input, symmetric extension takes place.

Filtering phase: the filter processes the input signal samples present in the internal registers.

Finalization phase: at the end of filter's operation, when the input samples are fully consumed, the input signal must be extended to perform the remaining filtering operations imposed by the algorithm.

3.1.3 1D-DWT architecture

The input signals in 1-D DWT are transformed into a high (H) and a low (L) component signals by a 1-D filter bank. Referring to the lifting scheme using the filter (9, 7) presented in Fig.4, four lifting steps are required to compute the high pass coefficients and the low pass coefficients. So, to map the lifting scheme of the filter (9, 7) to the architecture, each lifting step can be assigned to a computing module, as shown in Fig. 11.

Based on the operation of the row processing illustrated in the Fig. 4, the straightforward implementation is to apply the first step of the Lifting Calculation (LC1) to all the input samples and store the transformed coefficients, and then apply the second step (LC2). The same process is repeated for the third and fourth steps. However, this approach requires high embedded memory usage (to store intermediates coefficients), large amount of access to external memory (to access the even index input samples for LC2), and will results in huge latency. To overcome all these issues, we propose to start the computation of LC2, LC3 and LC4 as soon as enough data are available (2 coefficients are produced) in order to reduce the LC2, LC3 and LC4 latency. The register block is also used between each processor to locally store the intermediate results computed by the previous step and the even data.

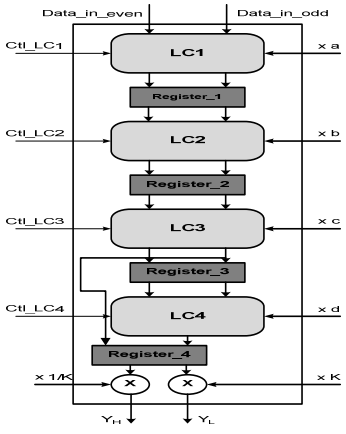

Fig. 11 The block diagram of 1D-DWT architecture.

The 1D-DWT block reads two inputs data in one clock cycle, because the even and odd image data are stored in

one memory case. Fig. 11 shows the internal architecture of 1D-DWT block. It consists of four instances of the Lifting Calculation blocks LC1, LC2, LC3 and LC4 and the intermediate registers blocks denoted by Register_1, Register_2, Register_3 and Register_4 to store the intermediate data needed between the lifting steps. In general, all the lifting steps are essentially in the form:

$$Y_i = X_i + a(X_{i-1} + X_{i+1}) \quad (6)$$

Where a is multiplication factor. So we need a different configuration of adders, and multipliers that are connected in a manner that will support the computational structure of the lifting steps. For the (9, 7) filter, there are four multiplication factors, the Fig. 12 illustrates the architecture of LC block, it composed from one multiplier one adder and two delay registers.

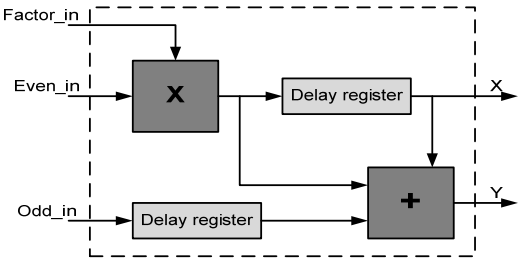

Fig. 12 Basic architecture of each Lifting Calculation (LC).

Based on the bit precision analysis, the system data path width is fixed at 16 bits. The adder and registers are designed for 16-bit data. The adder computes the sum of three input vectors in one clock cycle. The multiplier multiplies a 16-bit number by a 12-bit number (filter coefficient) and then rounds the product with twelve LSBs and four MSBs to form a 16-bit output.

Pipelining the data path requires that values passed from one pipe stage to the next pipe stage must be placed in registers. After an initial latency, the 1D-DWT block generates two coefficients (high pass and low pass) in every clock cycle. As a result, delay registers are used between the pipe stages in LC blocks. The register placed between two LC blocks also work as pipeline registers to store the values that will be used by the next LC block.

The same 1D-DWT block is used to compute the data coming from the embedded buffers and performs vertical filtering on the columns of the row-transformed image.

3.2 The embedded buffers block

When performing the 2D-DWT, the 1D-DWT block reads the row data from tile input memory in row-wise order and performs the horizontal filtering. The resulted high coefficients and the low pass coefficients are written into the embedded buffers (internal buffers). Once a half of three rows have been processed and stored in the intermediate buffers, the 1D-DWT block for columns starts processing and performs the column-wise filtering along the odd rows and generates the four sub-bands LL, LH , HL and HH into the output. Within this solution, the size of the embedded buffers is reduced and the power consumption becomes lower. To better improve the speed of the proposed 2D-DWT architecture and to keep 1D-DWT block for columns active continuously, the rows have to be processed in a non sequential fashion as presented in [12]. Fig. 13 shows the buffers organization of internal buffer block. It composed from seven instances of Altera buffers [23] (buff_1, buff_2, buff_3, buff_4, buff_5, buff_6, and buff_7). Seven buffers are required for the calculation of the column filtering along the rows. All buffers have one read and one write port. The buffers from buff_1 to buff_6 contain a quarter row of data (i.e. size is 1 x N/4, where N is the width of the input image).

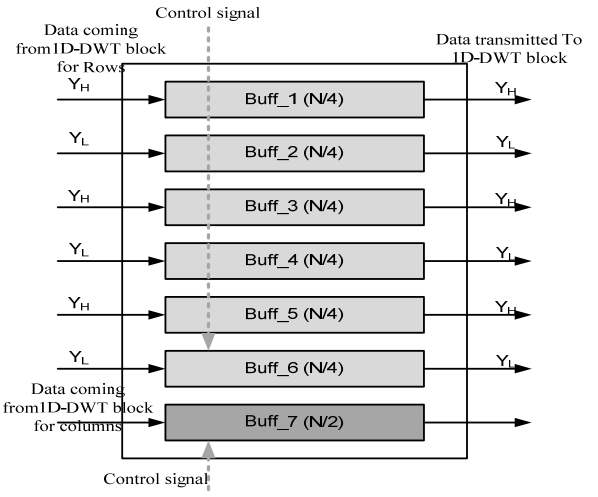

Fig. 13 Internal buffers structure for (9, 7) filter.

The high pass coefficients coming from the 1D-DWT block are write to the buff_1, buff_3, and buff_5 buffers, while the low pass coefficients are write to the buff_2, buff_4 and buff_6 buffers. Finally, the 1D-DWT block for columns writes to buff_7, which has size of a half row of data (1 x N/2). This arrangement of these buffers is due to the nature of the computation system for vertical filtering to calculate the HH, LH, HL, and LL sub-bands. So, only 2N ((6xN/4) + N/2) size of internal buffers are used to store the intermediate data between horizontal filtering and vertical filtering. The raison that the data is stored into separate buffers is to perform read and write operations at the same clock cycle and to reduce the complexity of FIFOs control.

For multiple decompositions, The LL sub-band FIFO will be read by the 1D-DWT block. The proposed architecture

works until five levels of decomposition, and the multiplexer component is used to select the appropriate data input between tile image data or the LL_FIFO data. Beyond one level of decomposition, the LL1 sub-band is read and we apply the 2D-DWT transform in order to produce HH2, LH2, HL2 and LL2 sub-bands. Similarly, for the third decomposition, the LL2 sub-band is read to produce HH3, LH3, HL3 and LL3. For other higher levels of decomposition, the same procedure is repeated.

4. Implementation and performance analysis

The proposed architecture is described with VHDL language (VHSIC Hardware Description Language). All of the system components have been described with structural architecture using generic parameters which allow changing the number of stages of the filter, the word length or the number of decomposition as desired. The ModelSim HDL was used to simulate separately each component and the top of the proposed architecture. The 2D-DWT algorithm for (9, 5) filter was also developed using C language (ISO/IEC 15444-1 [1] compatibility) to validate the architecture by comparison with the hardware behavioral. The architecture has been synthesized for an ALTERA Cyclone II and Stratix III [24] [25] FPGAs. In table 2, we shown the area results and the maximum clock frequency for the proposed architecture. We can see that our design needs only 8K bits memory of intermediate buffers and it also needs 512K bits size memory for LL buffer in multi levels decomposition. The architecture system spent 1.1 K gates which represents around 1% of the total Stratix slices and performs at 350 MHz clock frequency.

Table 2 Synthesis results of EBCOT block implemented in a Altera Cyclone II and stratix III.

	Cyclone II	Stratix III
Total logic elements	1,112 (2%)	1,112 (1%)
Total registers	286	163
Total memory bits	8190 (7%)	8190 (3%)
CLK (2-DDWT)	290 MHz	350 MHz

4.1 Performance comparison

The previous 2-D DWT architectures have the average N^2 computation time for NxN image. For performance analysis, we compare the number of multipliers, adders and memory size and computation time. The performance comparisons of our architecture and other similar architectures are listed in Table 3.

Table 3 Performance of the proposed architecture compared with those of other works.

Architectures	Adders	Multipliers	Internal memory size	Computation time
Conventional Lifting	16	12	$N^2 + N^2/4$	$\frac{N^2}{2} + 2\sum_{j=0}^{L-1}\frac{N}{2^j}$
Andra [12]	8	4	4N	$\frac{N^2}{2} + 2\sum_{j=0}^{L-1}\frac{N}{2^j}$
Wu [14]	36	36	9N	$\frac{N^2}{2} + 2N$
Huang [15]	12	10	14N	$N^2 + 4N$
Jung [21]	12	9	12N	$\frac{N^2}{2} + 2\sum_{j=0}^{L-1}\frac{N}{2^j}$
Mei [26]	10	4	10N	$\frac{N^2}{2} + 2\sum_{j=0}^{L-1}\frac{N}{2^j}$
McCanny [27]	32	19	17N	$N^2 + 2N$
Proposed	8	4	2N	$\frac{N^2}{2} + N$

From Table 3, the proposed architecture may need the same hardware cost than Andra architectures [12]. However, our architecture needs fewer memories size and has shorten critical path than the above one. In addition, the proposed architecture does not waste any clock cycle to handle the boundary side. In order to compare our architecture with the Andra's one, we have implemented Andra's architecture on the same FPGA Altera Stratix III. The total number of the used gate and the estimated work frequency for the (9,7) lifting architecture is provided in Table 4.

Table 4 The total number of the used gate and the estimated work frequency.

Architecture	Total gates	Clock frequency
Andra [12]	2,103 (2%)	260 MHz
Proposed	1,112 (1%)	350 MHz

A high processing frequency is achieved with a suitable number of the clock cycles/image, thus the proposed architecture is faster, with low number of the required clock cycles. This increasing in the speed of the 2D-DWT is mainly due to a pipeline structure, memory efficient and reducing the size of the embedded buffers.

4.2 Power consumption

In this section, we provide energy measurements for the implementations. We provide on-chip energy results, excluding the image memory. To estimate the on-chip power consumption of each schedule, we have used the Stratix-power-estimator tool offered by Altera Quartus v.8.1. We need multiple voltage supplies to power our FPGA. We have used the default values in the Stratix-power-estimator: Vccint 1.2V and Vccpd 3.3V. The IO pins consume a large power since these are designed in a larger geometry than the core to support sinking currents for all of the IO standards. However, since all our designs have the same IO pins, the power consumed would be similar. IO power is approximately 0.02W for our designs. The formulas used for power calculations in the program are based on the intended behavior of our digital hardware design.

$$P = CV^2EF \quad (7)$$

Where, P is the power in mW, C is the load capacitance in Farads, E is the switching activity for the element and F is the frequency of operation in Hz. The Stratix-power-estimator tool requires design information such as the resource utilization (LUTs, IOs, registers, DSPs), clock frequency, clock fanout and the toggle rates to estimate the dynamic power for our design.

Fig. 14 presents the energy consumed for the execution of the proposed 2D-DWT architecture and conventional lifting, relative to different image size N and 5 levels of decomposition. Thus, the power consumption of the proposed architecture starts at 135 mW for image size N= 256 and stretches to 487 mW for N = 1024. The power consumption of conventional lifting is at relatively higher levels, starting at 435 mW (N=256) and reaching 723 mW (N=1024). This is due to the large number of cycles associated with access to memory and the size of the embedded buffer. The number of cycles is once more the dominant factor in the energy calculations. That is, the significantly smaller computation time (see table 3) results in lower power consumption than others works.

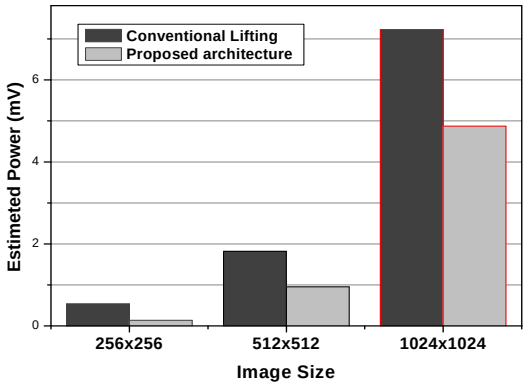

Fig. 14 Power consumption comparison between conventional Lifting and the proposed architecture for (9, 7) filter.

Finally, the proposed architecture is faster than others previous works. Our design has two important advantages than others works:

- It only uses 2xN size of embedded buffer and employs an adder block to calculate the sum of three inputs vectors in one clock cycle.
- It has almost lower power consumption compared with the conventional lifting.

5. Conclusion

In this paper, we have proposed an efficient VLSI architecture for the 2D-DWT to meet the requirements of real-time image and video processing. The advantages of the proposed architecture are saving embedded memories, fast computing time, low power consumption, and low control complexity. This hardware is designed to be used as part of a complete high performance and low power JPEG2000 encoder system for digital cinema applications. The proposed architecture has been correctly verified by the VHDL Language. It routed in Altera Stratix III to work at 350 MHz and Cyclone II FPGA at 290 MHz. The FPGA implementation can code 48 frames (4096 x 2160) per second with 24 bpp. Moreover, it can be applied very well to the implementation of the coder used in real time video processing, such as MPEG-4.

References

[1] ISO/IEC FCD15444-1: 2000, “JPEG 2000 image coding system,” 2000.
[2] ISO/IEC JTC1/SC29/WG11, FCD 14496-1, “Coding of moving pictures and audio,” 1998.
[3] D. S. Taubman and M. W. Marcellin, JPEG2000: “Image Compression Fundamentals, Standards and Practice”, Norwell, MA: Kluwer, 2002.

[4] A. Cohen, I. Daubechies, and J. Feauveau, "Bi-othogonal bases of compactly supported wavelets", Comm. Pure Appl. Math., vol. 45, pp. 485-560, 1992.
[5] W. Sweldens, "The Lifting Scheme: A Custom-Design Construction of Biorthogonal Wavelets", Applied and Computational Harmonic Analysis, Vol. 3, NO. 15, pp. 186-200, 1996.
[6] A. R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo, "Wavelet transforms that map integers to integers", Technical report, Deportment of Mathematics, Princeton University, 1996.
[7] G. Knowles, "VLSI architecture for the discrete wavelet transform", Electronic Letters, vol. 26, no.5, pp. 1184-1185, July 1990.
[8] P.-C. Wu and L.-G. Chen, "An efficient architecture for two-dimensional discrete wavelet transform", IEEE Trans. Circuits and Syst. Video Tech., vol. 11, no. 4, pp. 536-545, April 2001.
[9] C. Yu and S.-J. Chen, "Design of an efficient VLSI architecture for 2D discrete wavelet transforms", IEEE Trans. Consumer electronics, vol. 45, no. 1, February 1999.
[10] S. Masud, J. V. McCanny, "Rapid design of biorthogonal wavelet transforms," in Proc. IEE Circuits, Devices and Systems, vol. 147, pp. 293-296, Oct. 2000.
[11] Xixin Cao, Qingqing Xie, Chungan Peng, Qingchun Wang, Dunshan Yu, "An Efficient VLSI Implementation of Distributed Architecture for DWT", IEEE Trans. Circuits and Syst. Video Tech., 0-7803-9752-5/06. 2006.
[12] K. Andra, C. Chakrabarti, and T. Acharya, "A VLSI architecture for lifting-based forward and inverse wavelet transform", IEEE Trans. Signal Processing, vol. 50, no. 4, pp. 966-977, April 2002.
[13] G. Dillen, B. Georis, J.-D. Legat, and O. Cantineau, Combined line-based architecture for the 5-3 and 9-7 wavelet transform of JPEG2000," IEEE Trans. Circuits and Syst. Video Tech., vol. 13, no. 9, September 2003.
[14] B.-F. Wu and C.-F Lin, "A rescheduling and fast pipeline VLSI architecture for lifting-based discrete wavelet transform," in Proc. IEEE int. Symp. Circuits and Systems, vol. 2, May 2003.
[15] C.-T. Huang, P.-C. Tseng, L.-G. Chen, "Flipping structure: an efficient VLSI architecture for lifting-based discrete wavelet transform", IEEE Trans. Signal Processing, vol. 52, no. 4, April 2004.
[16] S. V. Silva and S. Bampi, "Area and throughput trade-offs in the design of pipelined discrete wavelet transform architectures," in Proc. IEEE Design, Automation and Test in Europe, 2005.
[17] Maurizio Martina and Guido Masera, "Low-Complexity, Efficient 9/7 Wavelet Filters VLSI Implementation", IEEE Trans on circuits and systems_II: Express Briefs, Vol. 53, No. 11, November 2006.
[18] S. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," IEEE transactions on Pattern Analysis and Machine Intelligence, vol. 11, no. 7, 1989, pp. 674–693.
[19] Omid Fatemi Sara Bolouki, "Architecture for the 2- D Discrete Wavelet Transform using Lifting Scheme", Department of Electronics and Computer Engineering University of Tehran, Iran, Proc. SPIE, Vol. 5150, 1121 (2003).
[20] N. D. Zervas, G. P. Anagnostopoulos, V. Spiliotopoulos, Y. Andreopoulos, and C. E.Goutis, "Evaluation of Design Alternatives for the 2D-Discrete Wavelet Transform", IEEE Trans. Circ. and Syst. for Video Tech, 2001.
[21] G. C. Jung, D. Y. Jin, and S. M. Park, "An efficient line based VLSI architecture for 2-D lifting DWT", in IEEE Int. Symp. Circuits and Systems, July 2004.
[22] C. Zhang, C. Wang, and M. O. Ahmad, "A VLSI architecture for a high-speed computation of the 1D discrete wavelet transform", in Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium, pp. 461-1464, May 2005.
[23] Altera Megawizard buffers : Complete Data Sheet. ALTERA. [Online]. Available: http://www.altera.com.
[24] Cyclone-II platform FPGAs: Complete Data Sheet. ALTERA. [Online]. Available: http://www.altera.com.
[25] Stratix-III platform FPGAs: Complete Data Sheet. ALTERA. [Online]. Available: http://www.altera.com.
[26] K. Z. Mei, N. N. Zheng, C. Huang, Y. Liu, and Q. Zeng "VLSI Design of a High-Speed and Area-Efficient PEG2000 Encoder", IEEE Trans. Circuits Syst. Video Technol., vol. 17, no. 8, pp. 1065–1078, Agu. 2007.
[27] P. McCanny, S. Masud, and J. McCanny, "An efficient architecture for the 2-D biorthogonal discrete wavelet transform", in Proc. IEEE Image Processing, Oct. 2001.

Author Biographies

Anass MANSOURI received the B.S. and M.S. degrees in electrical engineering from Faculty of sciences & techniques, Fes, MOROCCO, in 2003 and 2005, respectively, where he is currently working toward the Ph.D. degree in the Department of Electrical Engineering. His major research interests include VLSI architecture design and algorithms for Image, Audio and video processing, reconfigurable computing for multimedia systems. He is a member of the LSSC laboratory.

Ali AHAITOUF received the Ph.D. degrees in electronics from the Metz University in France 1992. He is a Professor in electrical engineering department at Faculty of sciences & techniques, Fes, MOROCCO, when he obtained the Doctor Title in Physics at 1998. His major research interests include Digital and Analog VLSI architecture, EMC Simulation and Physics of Semiconductor Components. He is managing the Microelectronics and Components research group.

Farid ABDI received the Ph.D. degrees in Physics from the Metz University in France 1992. He is a professor in electrical engineering department at Faculty of sciences & techniques, Fes, MOROCCO. His major research interests include Optical Components, Image, Audio and video processing. He is managing the optical and image processing research group.