AN OPTIMIZED ARCHITECTURE FOR 2D DISCRETE WAVELET TRANSFORM ON FPGAs USING DISTRIBUTED ARITHMETIC

PROJECT REPORT

Written by
Patrick Longa

Professor: Miodrag Bolic
Digital Signal Processing:
Microprocessors, Software and Applications

School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa
2006

An Optimized Architecture for 2D Discrete Wavelet Transform on FPGAs using Distributed Arithmetic: Project Report

1. Introduction

In the last few years, there has been a growing trend to implement DSP functions in Field Programmable Gate Arrays (FPGAs), which offer a balanced solution in comparison with traditional devices. Although ASICs and DSP chips have been the traditional solution for high-performance applications, now the technology and the market are imposing new rules. On one hand, high development costs and time-to-market factors associated with ASICs can be prohibitive for certain applications and, on the other hand, programmable DSP processors can be unable to reach a desired performance due to their sequential-execution architecture. In this context, FPGAs offer a very attractive solution that balance high flexibility, time-to-market, cost and performance.

Following this trend, the research community has focused in evaluating DSP functions on FPGAs to take advantage of the high level of parallelism that can be reached with these devices. In this sense, Discrete Wavelet Transform (DWT), a relatively new computing-intensive signal transform, has been explored and several architectures have been proposed to achieve a performance level that is difficult to reach with traditional PDSP devices.

Although wavelets have been around for a while in the area of mathematics, it has been until very recently that they were formally formulated and began to be extensively used. Just in 1986, a joint effort between Mallat and Meyer [2] gave birth to multiresolution analysis with wavelets, and in 1988, based on this study, Daubechies [3] discovered the most widely used and known family of wavelets, called after her the Daubechies Wavelets. Nowadays, wavelet transform is intensively used in speech, image and video processing, and in signal processing in general because of its attractive characteristics to represent non-stationary signals in both frequency and time domains. Researchers have switched from STFT (Short-Time Fourier Transform) to DWT because the former uses a fixed resolution at all times while the latter provides a variable resolution following the observed pattern on most applications: high frequency components of signals have a short duration while low frequency components have a long duration [4]. Furthermore, DWT has been adopted by recent still image and video coding standards, JPEG2000 [5] and MPEG-4 [6], given its high performance for image and video compression showing superior results when compared to the traditionally used Discrete Cosine Transform (DCT).

For the present project, we have developed an innovative two-dimensional Discrete Wavelet Transform (2D-DWT) architecture for image compression applications based on the very

well known Distributed Arithmetic (DA) technique, which exploits the LUT-based FPGA structure to build multiplier-less filterbank, the main component in a DWT structure. With help of a special memory arrangement, the fully parallel DA-based 2D-DWT has been time multiplexed in such a way that the downsampling stage is seamlessly realized inside the filterbank structure. With this scheme, our implementation achieves twice the speed of traditional proposals because a resultant sample outputs every clock cycle. Furthermore, the latter is achieved with the input samples running at the same clock rate than the rest of the circuit.

The proposed DA-based architecture for the 2D-DWT has been implemented using 8-tap Daubechies filterbanks in Simulink (DSP Builder) and Altera Quartus II. Simulations with several images have been done in Matlab to evaluate performance. This new architecture is intended for image compression applications, specifically for JPEG2000, the new image-coding standard that has adopted DWT as main component [5].

2. Previous work

In the last few years, there have appeared several efforts to propose optimized architectures for 1D and 2D-DWT on ASICs and more recently on FPGAs, with focus in achieving superior levels of throughput necessary in highly intensive image and video processing applications, which frequently demand real-time performance.

The basic structure of a 1D-DWT is originally based on filterbank for sub-band decomposition as presented in figure 1, where the number of cascaded units determines the level of transformation in the Discrete Wavelet Transform. Figure 1 shows a two-level 1D-DWT.

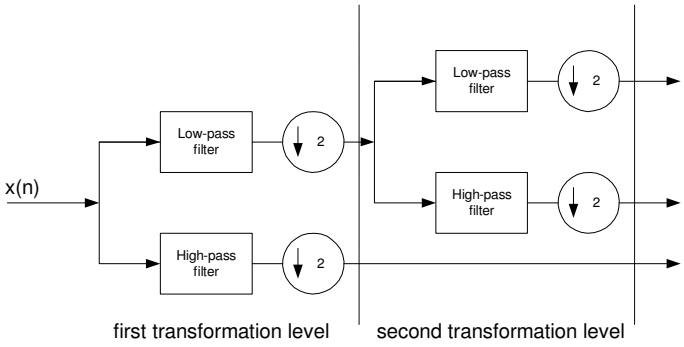

Fig. 1. Two-level 1D Discrete Wavelet Transform.

From the previous structure, a separable 2D-DWT with N levels of transformation can be easily achieved by concatenation of 1D-DWT units, with the first stage processing N transformation levels on rows and the second one with N transformation levels on columns. However, that

scheme is still not the most efficient one. For image compression purposes, JPEG 2000 recommends an alternate row/column-based structure as the one presented in Figure 2.

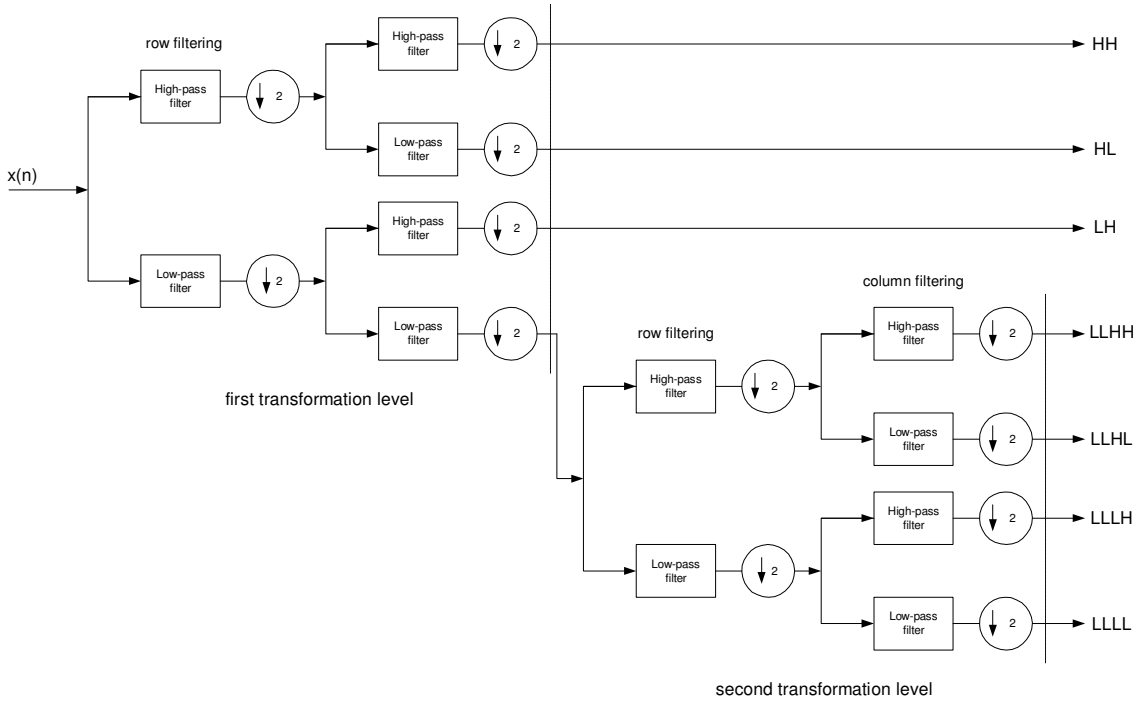

Fig. 2. Standard 2D-DWT with two transformation levels.

The standard filterbank shown in figure 2 processes alternatively rows and columns in every stage, with iteration only in the low-pass filter sub-band. Figure 3 shows the sub-band decomposition of an image when the standard 2D-DWT with three transformation levels is applied. In figure 3, “h” and “l” correspond to high and low-pass filter stages, respectively.

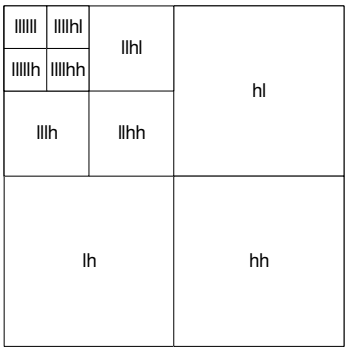

Fig. 3. Sub-band decomposition using a 2D-DWT with three transformation levels.

Based on these previous basic structures for 1D and 2D-DWT, several approaches have been proposed to overcome the area/bandwidth limitations imposed by direct implementation of the filterbank structure. Among them, Distributed Arithmetic (DA) presents an attractive solution to design multiplier-less filterbanks especially suited for LUT-based FPGA architectures. DA, first proposed by Croisier et al. [11], is a multiplier-less architecture that is based on an efficient partition of the function in partial terms using 2's complement binary representation of data. The partial terms can be pre-computed and stored in LUTs. The flexibility of this algorithm on FPGAs permits everything from bit-serial implementations to pipelined or full-parallel versions of the scheme, which can greatly improve the design performance. It is relevant to note that for each case the designer has to trade between bandwidth and area.

Some approaches have been proposed to apply DA technique to the design of multiplier-less DWT filterbanks. Alam et al. [10] presented a fully parallel LUT-less DA-based architecture to implement a 1D-DWT. The inner products traditionally pre-computed and stored in LUTs were mapped to arrangements of adders called Adder Butterflies. The advantage of this scheme is that it avoids the increase in LUT usage by exploiting the redundancy in the partial terms and using an optimal implementation with adders. This way, it achieves to process an N by N image in $N^2/2$ clock cycles. That shows a superior performance in relation to the N^2 clock cycles needed in traditional DWT schemes. However, this is not optimal since the extra decimation step after filtering still increases the computing effort.

In [11], Al-Haj used the traditional LUT-based DA scheme to implement the filterbank in a 1D-DWT. As expected, the results show a high throughput when the filter bank is implemented in a fully parallel way but with a higher resource requirement, while the serial version shows a lower throughput with improved savings in area. Again, the disadvantage of this scheme is that the extra decimation step after filtering increases the computing effort.

For the present project, we have focused in wavelets based on orthonormal basis. We will show that applying Distributed Arithmetic and time-multiplexing techniques to build the DWT filterbank it is possible to achieve twice the speed reached by previous proposals. Furthermore, important area savings can be achieved given that the delay line is shared by high and low-pass filters.

3. Proposed architecture

For the proposed architecture based on orthonormal basis, we will take advantage of the polyphase nature of the sub-band decomposition in the DWT. Equation (1) describes the general structure of an FIR filter:

$$y[n] = \sum_{k=0}^{K-1} a_k x_{[n-k]} \tag{1}$$

With transfer function:

$$H(z) = \sum_{n=0}^{N-1} h_{(n)} z^{-n} \tag{2}$$

Equation (2) can be decomposed and expressed as filters operating at half rate on odd and even samples, respectively. This is shown in equation (3), and it is applicable to both high- and low-pass filters in the wavelet filterbank.

$$H(z) = H_0(z^{-2}) + z^{-1} H_1(z^{-2}) = \sum_{n=0}^{N-1} h_{(2n)} z^{-2n} + z^{-1} \sum_{n=0}^{N-1} h_{(2n+1)} z^{-2n} \tag{3}$$

Because we are dealing with images where samples are available normally in memory before processing, it is possible to take advantage of the partition in equation (3) and to process two input samples concurrently to obtain an output sample every clock cycle. This scheme is shown in figure 4.

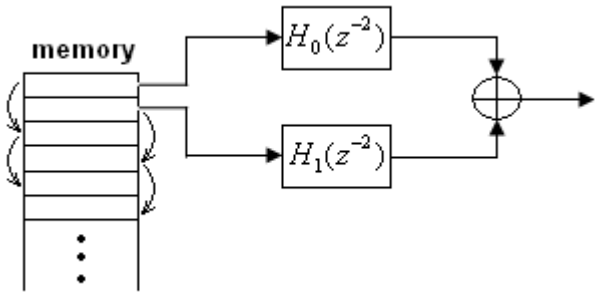

Fig. 4. Decimated filter with two input samples processing simultaneously.

Now we will apply the previous concepts in a DA scheme. Assuming that the input *x* to the filter is represented in L-bit 2's complement binary numbers with the sign bit to the left of the radix point, we have:

$$x_{[n-k]} = -b_{k,0} + \sum_{l=1}^{L-1} b_{k,l} 2^{-l} \tag{4}$$

Replacing this result in equation (1), we obtain:

$$y[n] = \sum_{k=0}^{K-1} a_k (-b_{k,0} + \sum_{l=1}^{L-1} b_{k,l} 2^{-l})$$

$$y[n] = -\sum_{k=0}^{K-1} a_k b_{k,0} + \sum_{l=1}^{L-1} (\sum_{k=0}^{K-1} a_k b_{k,l}) 2^{-l} \quad (5)$$

In equation (5), we observe that the term in parenthesis may take one of 2^K possible values, given that $b \in \{0,1\}$, and that those values correspond to all possible sum combinations of filter coefficients a_k. These values can be pre-computed and stored in LUTs or memories, and addressed by $b_{k,l}$. This way, the Multiply-and-Accumulate (MAC) algorithm of FIR filters is reduced to LUT accesses and summations.

A straightforward approach to decimate the result given in equation (5) is partitioning the filter in half-rate filters as shown in equation (3). This time, however, with every two samples (odd and even samples processed simultaneously as in figure 4), we would obtained a complete added result every other clock cycle. Then, the left clock cycle should be discarded in some way. A special arrangement of interlaced input registers as shown in the next section seamlessly discard the unnecessary response and present the output at the same rate than input samples.

4. Circuit description

Applying the described approach, the proposed DA-based 2D-DWT on an FPGA would consist of three main components (see figure 5): DA-based filterbank units, memory blocks and the control unit.

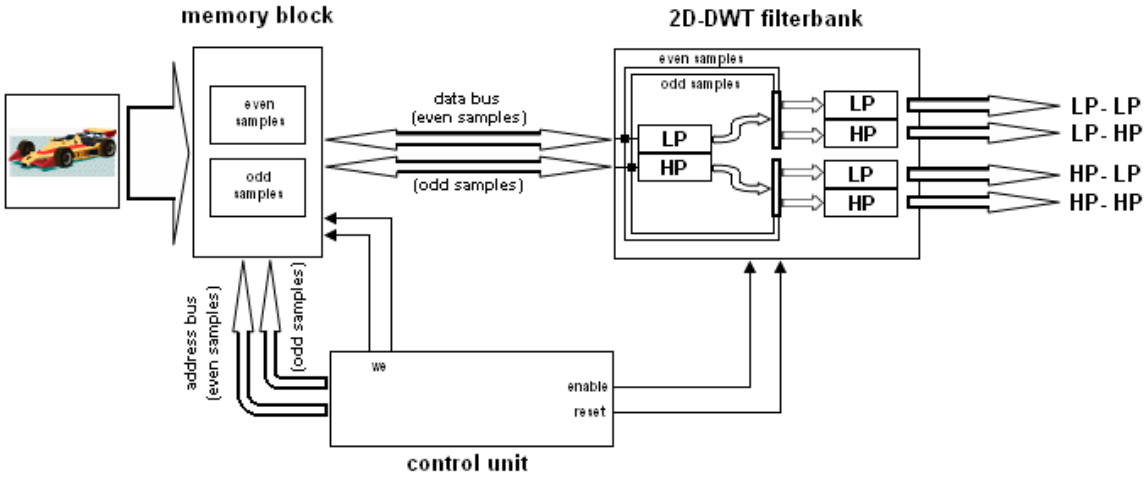

Fig. 5. Basic architecture of the 2D-DWT showing one transformation level.

DA-based filterbank:

The main component of the DWT architecture is the filterbank, which is basically an array of high- and low-pass filters as illustrated in figure 2. In the presented DA-based approach, every filter in the filterbank consists of the next components:

1. Input registers:

It consists of an array of 8 input registers that are shared by every high- and low-pass filter pair. The array has been interlaced as shown in figure 6 to produce an output sample every two concurrent input samples. Furthermore, our scheme permits to achieve reduced area requirement with the delay sharing described previously.

2. LUT unit:

One of the problems with LUT-based implementations is that the LUT requirement increases exponentially with the order of the filter. To alleviate this problem, the main strategy is to take advantage of the 4-input LUT-based architecture of FPGAs. Partitioning the input into 4-bit units reduces the Look-Up Table from 2^K – words to $(K/4 \times 2^4)$ - words at a cost of approximately 4-bit $(K-4)/4$ adders. As it is shown in figure 6, in our scheme every 8-input LUT has been partitioned into two 4-input LUT units. To have the complete response, the two partial values from each 4-input LUT are added together.

To implement the LUT values, we created two .hex files (one for the low-pass filters and another for the high-pass filters) containing all possible sum combinations of 8-tap Daubechies filter coefficients. These files are used to configure the LUT component in DSP Builder.

3. Adder-tree structure:

It consists of a tree of adders that perform addition and multiplication with the appropriated scaling factor on partial results of each 4-input filter pair (see figure 6).

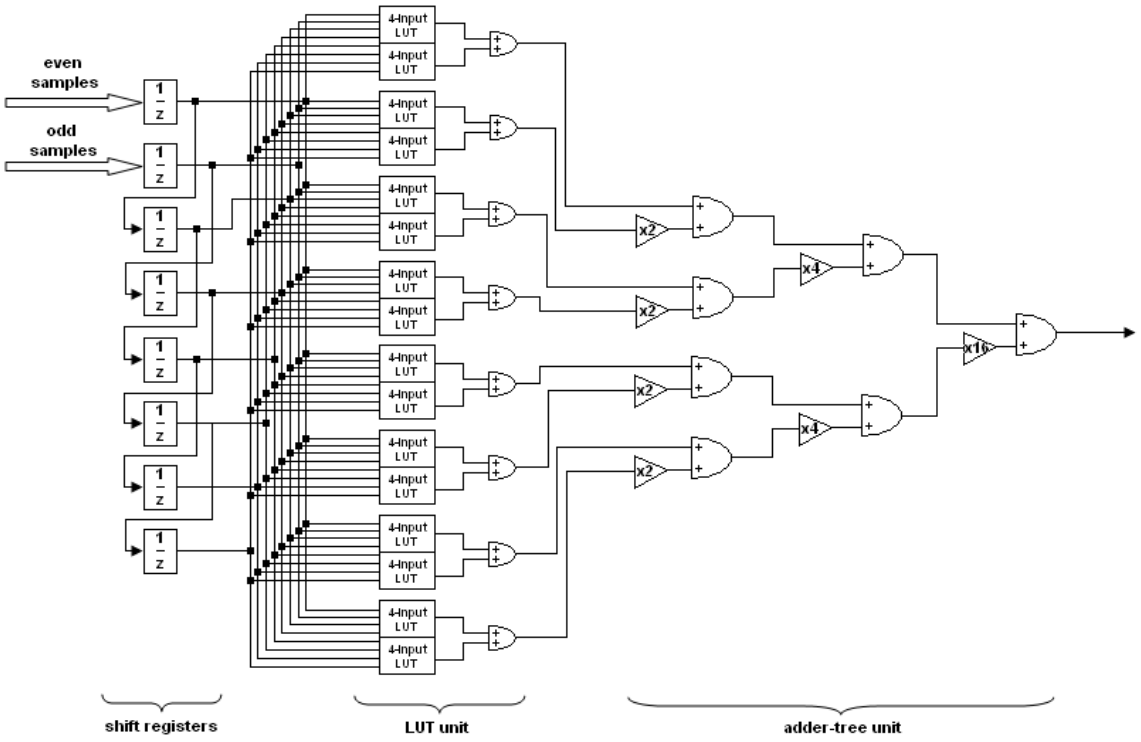

Fig. 6. Fully-parallel DA-based filter structure with interlaced shift registers.

Memory block:

The memory blocks consist of dual-port RAM units that store the partial and final transformed samples in a convenient way to allow the dual input access in each computation. For this purpose, the memory unit has been divided in two blocks: one to store even samples and another to store odd samples (see figure 5). The address generation and manipulation of write enable signals for even and odd sample blocks is done by the control unit.

Control unit:

This unit controls the whole circuit behaviour from address generation, data storage and access to reset and set up of filterbank operation before every row or column filter operation. The internal components of this unit were individually designed and simulated using VHDL in Altera software Quartus II, and compiled via HDL Import to instantiate them in Simulink as DSP Builder components.

5. Implementation and results

To evaluate the performance of the proposed scheme, the DA-based 2D-DWT described in previous sections was implemented using Simulink (DSP Builder) and Altera software Quartus II on a Stratix device. The precision for inputs and coefficients was 8 bits.

Firstly, the wavelet filter design was done using the Wavelet Toolbox from Matlab 7r14. The standard 8-tap Daubechies filter was used for this purpose. The resultant filter coefficients that were later used to implement the LUT units in our design are shown in Appendix A.

A floating to fixed-point analysis was carried out to determine the appropriated fixed-point representation. In our case, we focused in grayscale images for the wavelet transformation, where every input sample is represented in [0, 255] range and consequently every output sample must belong to the same range to represent the transformed image appropriately. Given that, we tested several coefficient representations by using the Fixed-point Toolbox from Matlab, and finally determined that 8-bit coefficients gave close enough results in the final image representation.

The logic flow of one transformation level of the presented 2D-DWT circuit can be summarized as follows:

A 64 by 64 image is divided in two blocks: one storing the even samples and another storing the odd samples. The first part of the process involves the filtering of each row in two different filters (high- and low-pass filters). To partially compensate the transformation in the edges, we include five zeros at the end of each row and one zero at the beginning, i.e. in total the first part involves 64 iterations filtering 70 input samples, which produces 32 x 35 output samples. To filter the 70 input samples two samples are injected at a time in every clock cycle and filtered out by the high- and low-pass filters. Finally, we get the 35 output samples per row in each filter. Simultaneously to the first filtering process, the output samples are stored in the memory blocks. Again, to prepare samples for the next stage, even and odd samples are stored separately and in the respective addresses as computed by the address generator of the control unit. Because of the initial padding process on rows, three samples per row are generated that not correspond actually to the filtered row. Then, these samples are eliminated and not stored in memory. The manipulation of rows in this first stage is shown in figure 8.

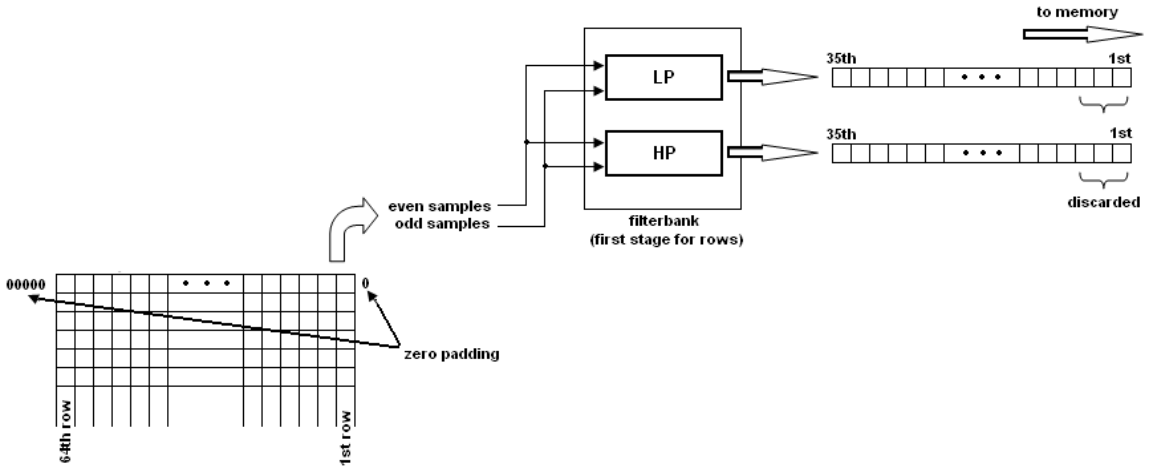

Fig. 8. Row filtering process in the first stage of the 2D-DWT.

In the second part of the process, we deal with the filtering of columns, which were conveniently stored in the block memories to allow the dual process over two input samples. This time the number of filters is increased to four two high-pass and two low-pass filters). Again, to partially compensate the transformation in the edges, we include five zeros at the end of each row and one zero at the beginning, i.e. in total this second part involves 32 iterations filtering 70 input samples, which produces 32 x 35 output samples per filter. To filter the 70 input samples per column two samples are injected at a time in every clock cycle and filtered out by the four high- and low-pass filters. Finally, we get the 35 output samples per column in each filter. Simultaneously to this process, the output samples are stored in the memory blocks. Again, to prepare samples for the next stage, even and odd samples are stored separately and in the respective addresses as computed by the address generator of the control unit. Because of the initial padding process on columns, three samples per column are generated that not correspond actually to the filtered column. Then, these samples are eliminated and not stored in memory.

The resultant values of each stage are conveniently truncated to 8-bit numbers to allow their final transformation to the grayscale range from 0 to 255. The final transformed image consists of four images which correspond to each of the sub-band transformations: hh, hl, lh and ll, where again “h” and “l” represents high- and low-pass filter stages, respectively.

To validate the correct functionality of the implemented circuit, tests were done in Simulink using several images. In figure 7, we can see the results for a 64 by 64 grayscale image. Four images that represent the first level of transformation were generated. They represent all combinations of high- and low-pass filters: hh, hl, lh and ll (where “h” and “l” represents high- and low-pass filter stages, respectively). Also, it is shown the 32 x 64 image that is generated in the first stage of row filtering.

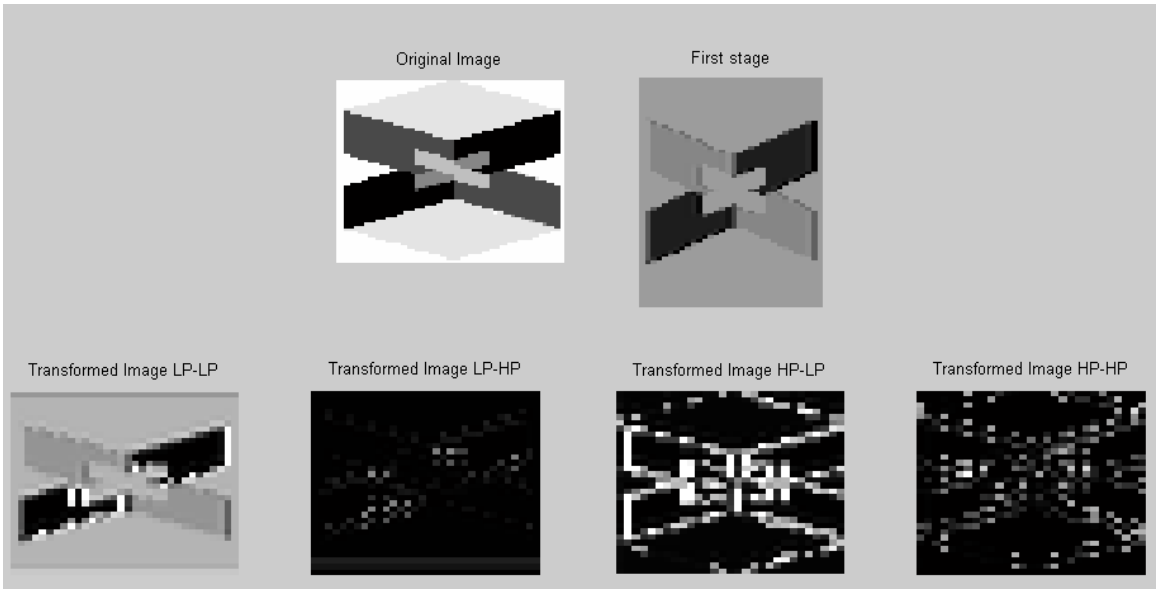

Fig. 7. Results obtained with a 64 by 64 grayscale image.

6. Summary

We have successfully implemented a fully parallel DA-based 2D-DWT with one level of transformation. A special arrangement in the memory structure and the interleaving manipulation in the DA-based filtering process applying time-multiplexing techniques have permitted to reduce the number of clock cycles needed for image transformation. From $N^2/2$ clock cycles needed to process an N by N image in previous DA-based schemes, we have reduced the computational time to $N^2/4$ clock cycles.

Furthermore, important area savings have been introduced with the shared utilization of the delay line in the filter structure among high- and low-pass filter pairs, and the efficient partition of LUT into 4-input LUT units following the 4-input structure of FPGAs.

On the other hand, some improvements can still be accomplished. For instance, a more efficient resource usage can be achieved in the filterbank if scheduling techniques were used and part of the filters of a given stage was used by following stages. Also a more efficient and complex treatment for the edge filtering can be used. That would reduce the error introduced in the edges when zero padding is applied.

In addition, our implementation presents a fully parallel scheme to achieve maximum throughput. If area savings are the most important factor, then a bit serial version of the DA-based filter structure would be the best option.

Finally, the proposed scheme has been applied for one transformation level on images. Appropriate modifications in the memory block and the control unit could permit more levels of transformations by cascading several of the implemented filterbank units.

References:

[1] "Reusable Silicon IP Cores for Discrete Wavelet Transform Applications", S. Masud, J. McCanny, in *IEEE Transactions on Circuits and Systems*, 2004**.**

[2] "A Theory for Multiresolution Signal Decomposition: the Wavelet Representation", S. Mallat, in *IEEE Transactions on Pattern Anal. Machine Intell.*, 1989.

[3] "Orthonormal bases of Compactly Supported Wavelets", I. Daubechies, in *Comm. Pure Appl. Math*, 1988.

[4] "The Wavelet Tutorial", R. Polikar, *Rowan University*, 2001 (last update). Website: http://users.rowan.edu/~polikar/WAVELETS/WTtutorial.html

[5] "JPEG2000: Image Compression Fundamentals, Standards and Practice", D. Taubman, M. Marcellin, *Kluwer Academic Publishers*, 2001.

[6] "The MPEG-4 Book", T. Ebrahimi, F. Pereira, *Prentice Hall*, 2002.

[7] "The Lifting Scheme: a New Philosophy in Biorthogonal Wavelet Construction", W. Sweldens, in *Proc. SPIE*, 1995.

[8] "Low-Power and High-Speed VLSI Architecture for Lifting-based Forward and Inverse Wavelet Transform", X. Lan, N. Zheng, Y. Liu, in *IEEE Transactions on Consumer Electronics*, 2005.

[9] "A VLSI Architecture for Lifting-based Forward and Inverse Wavelet Transform", K. Andra, C. Chakrabarti, in *IEEE Transactions on Signal Processing*, 2002.

[10] "Efficient Distributed Arithmetic based DWT Architecture for Multimedia Applications", M. Alam, C. Rahman, W. Badawy, G. Jullien, in *Proc. IEEE International Workshop on System-on-Chip for Real-Time Applications*, 2003.

[11] "An FPGA-based Parallel Distributed Arithmetic Implementation of the 1-D Discrete Wavelet Transform", Ali M. Al-Haj, in *International Journal of Computing and Informatics (Informatica), Volume 29, Number 2*, 2005.

[12] "A High-Throughput and Memory Efficient 2-D Discrete Wavelet Transform Hardware Architecture for JPEG2000 Standard", G. Dimitroulakos, M. Galanis, A. Milidonis, C. Goutis, in *IEEE International Symposium on Circuits and Systems (ISCAS 2005)*, 2005.

[13] "A Comparison between Lattice, Cascade and Direct Form FIR Filter Structures by using a FPGA Bit-Serial DA Implementation", Martinez-Peiro, J. Valls, T. Sansaloni, A.P. Pascual, E.I. Boemo, in *Proc. IEEE International Conference on Electronics, Circuits and Systems*, 1999.

[14] "Implementing FIR Filters in FLEX Devices", *ALTERA Application Note*, 1998.

[15] "Transposed Form FIR Filters", V. Pasham, A. Miller, K. Chapman, *XILINX Application Note*, 2001.

[16] "A Guide to Using Field Programmable Gate Arrays (FPGAs) for Application-Specific Digital Signal Processing Performance", G. R. Goslin, *XILINX*, 1995.