EFFICIENT IMPLEMENTATIONS OF DISCRETE WAVELET TRANSFORMS USING FPGAS

D. U. Shah[1], C. H. Vithlani[2]
[1]Assistant Prof., EC Department, School of Engineering, RK University, Rajkot, India.
[2]Associate Professor, Department of EC Engineering, GEC, Rajkot, India.

ABSTRACT

Recently the Wavelet Transform has gained a lot of popularity in the field of signal and image processing. This is due to its capability of providing both time and frequency information simultaneously, hence giving a time-frequency representation of the signal. The traditional Fourier Transform can only provide spectral information about a signal. Moreover, the Fourier method only works for stationary signals. In many real world applications, the signals are non-stationary. One solution for processing non-stationary signals is the Wavelet Transform. Currently, there is tremendous focus on the application of Wavelet Transforms for real-time signal processing. This leads to the demand for efficient architectures for the implementation of Wavelet Transforms. Due to the demand for portable devices and real-time applications, the design has to be realized with very low power consumption and a high throughput. In this paper, different architectures for the Discrete Wavelet Transform filter banks are presented. The architectures are implemented using Field Programmable Gate Array devices. Design criteria such as area, throughput and power consumption are examined for each of the architectures so that an optimum architecture can be chosen based on the application requirements. In our case study, a Daubechies 4-tap orthogonal filter bank and a Daubechies 9/7-tap biorthogonal filter bank are implemented and their results are discussed. Finally, a scalable architecture for the computation of a three-level Discrete Wavelet Transform along with its implementation using the Daubechies length-4 filter banks is presented.

KEYWORDS: *Daubechies wavelet, discrete wavelet transform, Xilinx FPGA.*

I. INTRODUCTION

In general, signals in their raw form are time-amplitude representations. These time-domain signals are often needed to be transformed into other domains like frequency domain, time-frequency domain, etc., for analysis and processing. Transformation of signals helps in identifying distinct information which might otherwise be hidden in the original signal. Depending on the application, the transformation technique is chosen, and each technique has its advantages and disadvantages.

The properties of Wavelet Transform allow it to be successfully applied to non-stationary signals for analysis and processing, e.g., speech and image processing, data compression, communications, etc. [5]. Due to its growing number of applications in various areas, it is necessary to explore the hardware implementation options of the Discrete Wavelet Transform (DWT).

An efficient design should take into account aspects such as area, power consumption, throughput, etc. Techniques such as pipelining, distributed arithmetic, etc., help in achieving these requirements. For most applications such as speech, image, audio and video, the most crucial problems are the memory storage and the global data transfer. Therefore, the design should be such that these factors are taken into consideration.

In this paper, Field Programmable Gate Arrays (FPGAs) are used for hardware implementation of the DWT [3, 4]. FPGAs have application specific integrated circuits (ASICs) characteristics with the advantage of being reconfigurable. They contain an array of logic cells and routing channels (called interconnects) that can be programmed to suite a specific application. At present, the FPGA based

ASIC market is rapidly expanding due to demand for DSP applications. FPGA implementation could be challenging as they do not have good arithmetic capabilities when compared with the general purpose DSP processors. However, the most important advantage of using an FPGA is because it is reprogrammable. Any modifications can be easily accomplished and additional features can be added at no cost which is not the case with traditional ASICs.

II. DIFFERENT WAVELET FILTER BANK ARCHITECTURES

There are various architectures for implementing a two channel filter bank. A filter bank basically consists of a low pass filter, a high pass filter, decimators or expanders and delay elements. We will consider the following filter bank structures and their properties, specifically with reference to DWT [1, 2].

2.1. Direct Form Structure

The direct form analysis filter consists of a set of low pass and high pass filters followed by decimators. The synthesis filter consists of up samplers followed by the low pass and high pass filters as shown in figure 1.

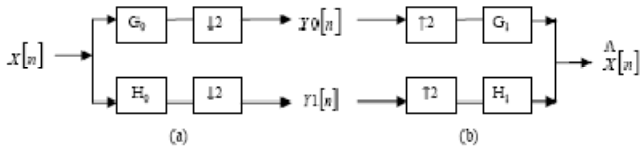


Figure 1: Direct form structure (a) Analysis filter bank (b) Synthesis filter

In the analysis filter bank, x[n] is the discrete input signal, G_0 is the low pass filter and H_0 is the high pass filter. ↓2 represents decimation by 2 and ↑2 represents up sampling by 2. In the analysis bank, the input signal is first filtered and then decimated by 2 to get the outputs Y_0 and Y_1. These operations can be represented by equations1 and 2.

$$Y_0[k] = \sum_n X[n].G_0[2k \quad n] \tag{1}$$

$$Y_1[k] = \sum_n X[n].H_0[2k - n] \tag{2}$$

The output of the analysis filter is usually processed (compressed, coded or analyzed) based on the application. This output can be recovered again using the synthesis filter bank. In the synthesis filter bank, Y_0 and Y_1 are first up sampled by 2 and then filtered to give the original input. For perfect output the filter banks must obey the conditions for perfect reconstruction.

2.2. Poly phase Structure

In the direct form analysis filter bank, it is seen that if the filter output consists of, say, N samples, due to decimation by 2 we are using only N/2 samples. Therefore, the computation of the remaining unused N/2 samples becomes redundant. It can be observed that the samples remaining after down sampling the low pass filter output are the even phase samples of the input vector X_{even} convoluted with the even phase coefficients of the low pass filter G_{0even} and the odd phase samples of the input vector X_{odd} convoluted with the odd phase coefficients of the low pass filter G_{0odd}. The poly phase form takes advantage of this fact and the input signal is split into odd and even samples (which automatically decimates the input by 2), similarly, the filter coefficients are also split into even and odd components so that X_{even} convolves with G_{0even} of the filter and X_{odd} convolves with G_{0odd} of the filter. The two phases are added together in the end to produce the low pass output. Similar method is

applied to the high pass filter where the high pass filter is split into even and odd phases H_{0even} and H_{0odd}. The poly phase analysis operation can be represented by the matrix equation 3.

$$\begin{bmatrix} G_{0_{even}} G_{0_{odd}} \\ H_{0_{even}} H_{0_{odd}} \end{bmatrix} \times \begin{bmatrix} X_{even} \\ z^{-1} X_{odd} \end{bmatrix} = H_p \begin{bmatrix} X_{even} \\ z^{-1} X_{odd} \end{bmatrix} = \begin{bmatrix} Y_0 \\ Y_1 \end{bmatrix} \qquad (3)$$

The filters with G_{0even} and G_{0odd} are half as long a G_0, since they are obtained by splitting G_0. Since, the even and odd terms are filtered separately, by the even and odd coefficients of the filters, the filters can operate in parallel improving the efficiency. The figure 2 illustrates poly phase analysis and synthesis filter banks.

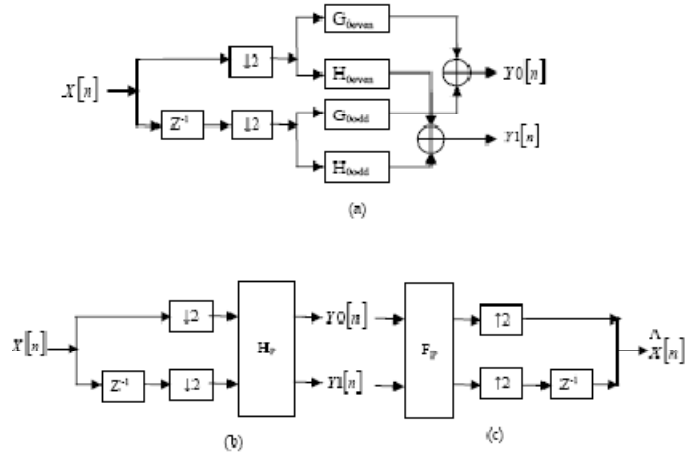


Figure 2: Polyphase structure of (a) Analysis filter bank (b) Equivalent representation of Analysis filter bank (c) Synthesis Filter bank

In the direct form synthesis filter bank, the input is first up sampled by adding zeros and then filtered. In the poly phase synthesis bank, the filters come first followed by up samplers which again, reduces the number of computations in the filtering operations by half. Since, the number of computations is reduced by half in both the analysis and synthesis filter banks; the overall efficiency is increased by 50%. Thus, the poly phase form allows efficient hardware realizations.

2.3. Lattice Structure

In the above structure, the poly phase matrix, $H_P(z)$ can be replaced by a lattice structure. The filter bank, $H_P(z)$ can be obtained if the filters $G_0(z)$ and $H_0(z)$ are known. Similarly, if $H_P(z)$ is known, the lattice structure can be derived by representing it as a product of simple matrices. The wavelet filter banks have highly efficient lattice structures which are easy to implement. The lattice structure reduces the number of coefficients and this reduces the number of multiplications. The structure consists of a design parameter *k* and a single overall multiplying factor. The factor *k* is collected from all the coefficients of the filter. For any *k*'s, a cascade of linear phase filters is linear phase and a cascade of orthogonal filters is orthogonal. The complete lattice structure for an orthogonal filter bank is shown in figure 3, where β is the overall multiplying factor of the cascade.

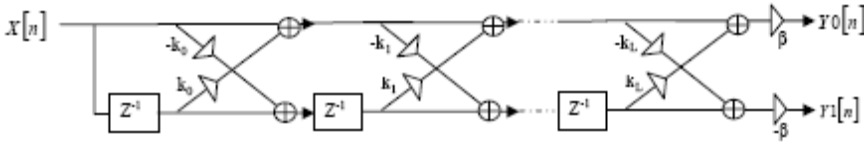


Figure 3. Lattice structure of an orthogonal filter bank

The lattice structure improves the filter bank efficiency as it reduces the number of computations performed. If the direct form requires 4L multiplications, the poly phase requires 2L multiplications, and the lattice requires just L+1 multiplications. The number of additions is also reduced in the lattice form.

2.4. Lifting Structure

The lifting scheme proposed independently by Herley and Swelden is a fast and efficient method to construct two-channel filter banks. It consists of two steps: lifting and dual lifting. The design starts with the Haar filter or the Lazy filter which is a perfect reconstruction filter bank with $G_0(z) = H_1(z)=1$ and $H_0(z) = G_1(z) = z^{-1}$. The lifting steps are:

Lifting: $H'(z) = H(z) + G(-z)\, S(z^2)$ for any $S(z^2)$.

Dual Lifting: $G'(z) = G(z) + H(-z)\, T(z^2)$ for any $T(z^2)$.

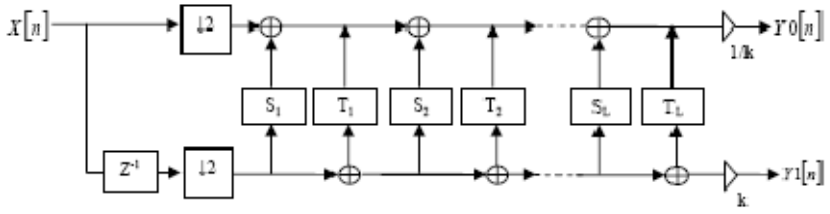


Figure 4. Lifting implementation

The lifting implementation is shown in figure 4. The lifting and dual lifting steps are alternated to produce long filters from short ones. Filters with good properties which satisfy the perfect reconstruction properties can be built using this method [18, 19].

III. COMPARISON OF IMPLEMENTATION OPTIONS

For hardware implementation, the choice of filter bank structure determines the efficiency and accuracy of computation of the DWT. All structures have some advantages and drawbacks which have to be carefully considered and based on the application, the most suitable implementation can be selected. It is observed that the direct form is a very inefficient method for DWT implementation. This method is almost never used for DWT computation. The poly phase structure appears to be an efficient method for DWT computation. But the lattice and lifting implementations require fewer computations than the poly phase implementation and therefore are more efficient in terms of number of computations. However, the poly phase implementation can be made more efficient than the lattice and lifting schemes in case of long filters by incorporating techniques like Distributed Arithmetic. Also, the lattice structure cannot be used for all linear phase filters and imposes restrictions on the length of the filters. In the case of the lattice and lifting schemes, the filtering units cannot operate in parallel as each filtering unit depends on results from the previous filtering unit. In the case of convolution poly phase implementation, the units can operate in parallel, and therefore the filtering operations have less delay. However, pipelining can be used in the other schemes to reduce the delay. Often, for implementation purposes, the real number filter coefficients are quantized into binary digits. This introduces some quantization error. In the lifting scheme, the inaccuracy due to quantization is accumulated with each step. Thus, the lifting scheme constants must be quantized with better accuracy than the convolution filter constants i.e., the lifting constants need to be represented by more number of bits.

IV. DISTRIBUTED ARITHMETIC TECHNIQUE

4.1 DA-based approach for the filter bank

Distributed Arithmetic (DA) has been one of the popular techniques to compute the inner product equation in many DSP FPGA applications [8, 11]. It is applicable in cases where the filter coefficients

are known a priori. The inner sum of products is rearranged so that the multiply and accumulate (MAC) operation is reduced to a series of look-up table (LUT) calls, and two's complement (2C) shifts and adds. Therefore, the multipliers which occupy large areas are replaced by small tables of pre-computed sums stored on FPGA LUTs which reduce the filter hardware resources.
Consider the following inner product calculation shown in 4(a) where c[n] represents an N-tap constant coefficient filter and x[n] represents a sequence of B-bit inputs:

$$y = \sum_{n=0}^{N-1} c[n].x[n] \qquad 4\ (a)$$

$$= \sum_{n=0}^{N-1} c[n].\sum_{b=0}^{B-1} x_b[k].2^b \qquad 4\ (b)$$

$$= \sum_{b=0}^{B-1} 2^b.\sum_{n=0}^{N-1} c[n].x_b[n] \qquad 4\ (c)$$

In equation 4(a), the inputs can be replaced as in 4(b) where $x_b[k]$ denotes the b^{th} bit of k^{th} sample of x[n]. Rearranging equation 4(b) gives 4(c). All the possible values of the inner function in (c) can be pre-computed and stored in an LUT. Now, the equation can be implemented using an LUT, a shifter and an adder. The architectures for the conventional MAC operation, represented by equation 4(a), and the DA-based shift-add operation, represented by equation 4(c) are shown in figure 5 for a 4-tap filter.

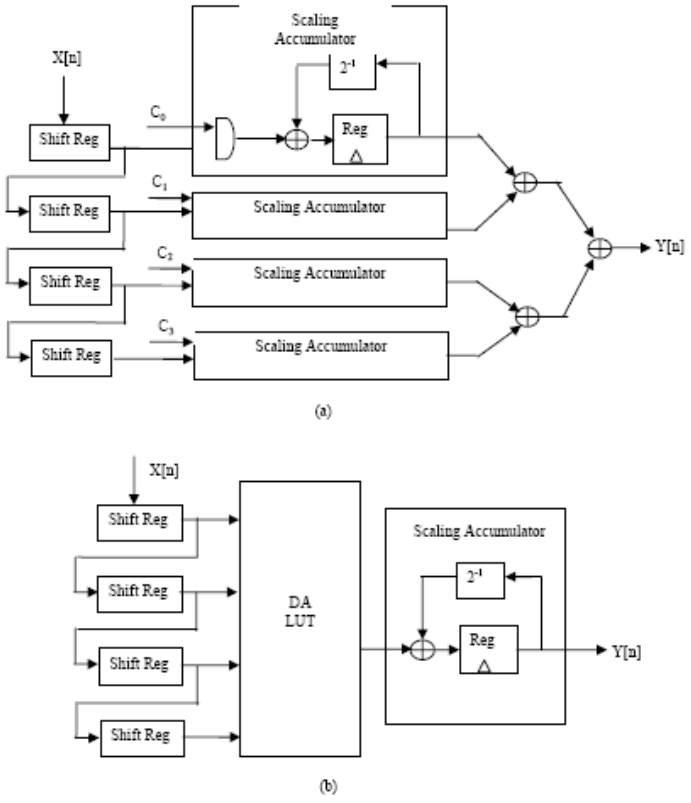


Figure 5. (a) Conventional MAC and (b) shift-add DA architectures.

In the DA architecture, the input samples are fed to the parallel-to-serial shift register cascade. For an N-tap filter and B-bit input samples, there are N shift registers of B-bits each. As the input samples are shifted serially through the B-bit shift registers, the bit outputs (one bit from each of N registers) of the shift register cascade are taken as address inputs by the look-up table (LUT). The LUT accepts the N bit input vector x_b, and outputs the value which is already stored in the LUT. For an N-tap filter

a 2^N word LUT is required. The LUT output is then shifted based on the weight of x_b and then accumulated. This process is followed for each bit of the input sample before a new output sample is available. Thus for a B-bit input precision a new inner product *y* is computed every B clock cycles.
Consider a four-tap serial FIR filter with coefficients C_0, C_1, C_2, C_3. The DA-LUT table is as shown in table 1. The table consists of the sums of the products of the N bit input vector x_b (N = 4 in this case) and the filter coefficients for all possible combinations.

Table 1. DALUT FOR 4 Tap Filter

Address	Data
0000	0
0001	C_0
0010	C_1
0011	C_0+C_1
:	:
:	:
:	:
1110	$C_1+C_2+C_3$
1111	$C_0+C_1+C_2+C_3$

In conventional MAC-based filter, the throughput is based on the filter length. As the number of filter taps increase, the throughput decreases. In case of DA-based filter, the throughput depends on the input bit precision as seen above and is independent of the filter taps. Thus the filter throughput is de-coupled from the filter length. But when the filter length is increased, the throughput remains the same while the logic resources increase. In case of long filters, instead of creating a large table, it can be partitioned into smaller tables and their outputs can be combined. With this approach, the size of the circuit grows linearly with the number of filter taps rather than exponentially.
For a DWT filter bank, the equation 4(c) can be extended to equation 5(a) and 5(b) to define the low pass and high pass filtering operations.

$$y_l = \sum_{b=0}^{B-1} 2^b \cdot \sum_{n=0}^{N-1} G_0[n].x_b[n] \qquad 5\ (a)$$

$$y_h = \sum_{b=0}^{B-1} 2^b \cdot \sum_{n=0}^{N-1} H_0[n].x_b[n] \qquad 5\ (b)$$

The poly phase form of the above filters can be obtained by splitting the filters and the input, x[n] into even and odd phases to obtain four different filters. Since the length of each filter is now halved they require much smaller LUTs [13, 14].

4.2 Parallel Distributed Arithmetic for Increased Speed

DA-based computations are inherently bit-serial. Each bit of the input is processed before each output is computed [9]. For a B-bit input, it takes B clock cycles to compute one output. Thus, this serial distributed arithmetic (SDA) filter has a low throughput. The speed can be increased by partitioning the input words into smaller words and processing them in parallel. As the parallelism increases, the throughput increases proportionally, and so does the number of LUTs required. Filters can be designed such that several bits of the input are processed in a clock period. Partitioning the input word into M sub-words requires M-times as many memory LUTs and this increases the storage requirements. But, now a new output is computed every B/M clock cycles instead of every B cycles. A fully parallel DA (PDA) filter is achieved by factoring the input into single bit sub-words which achieves maximum speed. A new output is computed every clock cycle. This method provides exceptionally high-performance, but comes at the expense of increased FPGA resources. Figure 6 shows a parallel DA architecture for an N-tap filter with 4-bit inputs.

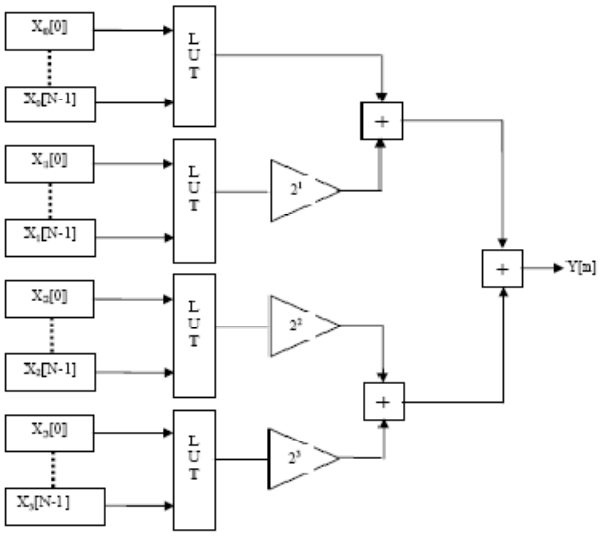


Figure 6. Parallel DA Architecture

In some applications, the same filter is applied to different inputs. In this case, instead of using two separate filters, a single filter can be shared among the different inputs. Sharing of filters decreases the filter sample rate but this method is very efficient in terms of the logic resources consumed. A multi-channel filter can be realized using virtually the same amount of logic resources as a single channel version of the same filter. The trade-off here is between the logic resources and filter sample rate.

4.3 A Modified DA-based approach for the filter bank

Unlike in the conventional DA method where the input is distributed over the coefficients, in this case the coefficient matrix is distributed over the input. It is seen that in the previous architecture, as the input bit precision increases there is an exponential growth in the LUT size and this increases the amount of logic resources required. The advantage of the present architecture over the previous one is that, in this method we do not require any memory or LUT tables. This reduces the logic resources consumed tremendously [10].

Consider the following inner product equation 6(a) where c[n] represents the M-bit coefficients of an N-tap constant coefficient filter and x[n] represents the inputs.

$$y = \sum_{n=0}^{N-1} c[n].x[n] \qquad 6\ (a)$$

$$= \sum_{m=0}^{M-1} c_m[k] 2^m \sum_{n=0}^{N-1} x[n] \qquad 6\ (b)$$

$$= \sum_{m=0}^{M-1} 2^m . \sum_{n=0}^{N-1} c_m[n].x[n] \qquad 6\ (c)$$

In equation 6(a) the coefficients can be replaced as in equation 6(b) where $c_m[k]$ denotes the m^{th} bit of k^{th} coefficient of c[n]. Rearranging equation 3.6(b) gives 6(c). The inner function, in 6(c) can be designed as a unique adder system based on the coefficient bits consisting of zeros and ones. The output, *y*, can then be computed by shifting and accumulating the results of the adder system accordingly based on the coefficient bit weight. Thus the whole equation can be implemented using just adders and shifters [20, 21].

V. IMPLEMENTATION OF DWT FILTER BANKS WITH FIELD PROGRAMMABLE GATE ARRAYS

Field Programmable Gate Arrays (FPGAs) are used to synthesize and test the architectures in this paper [7, 12]. FPGAs are programmable logic devices made up of arrays of logic cells and routing channels. They have ASIC characteristics such as reduced size and power dissipation, high throughput, etc., with the added advantage that they are reprogrammable. Therefore, new features can be easily added and they can be used as a tool for comparing different architectures. Currently, Altera Corporation and Xilinx Corporation are the leading vendors of programmable devices. The architecture of the FPGAs is vendor specific. Among the mid-density programmable devices, Altera's FLEX 10K and Xilinx XC4000 series of FPGAs are the most popular ones[6]. They have attractive features which make them suitable for many DSP applications. FPGAs contain groups of programmable logic elements or basic cells. The programmable cells found in Altera's devices are called Logic Elements (LEs) while the programmable cells used in Xilinx's devices are called the Configurable Logic Blocks (CLBs). The typical design cycle for FPGAs using Computer Aided Design (CAD) tools is shown in figure 7.

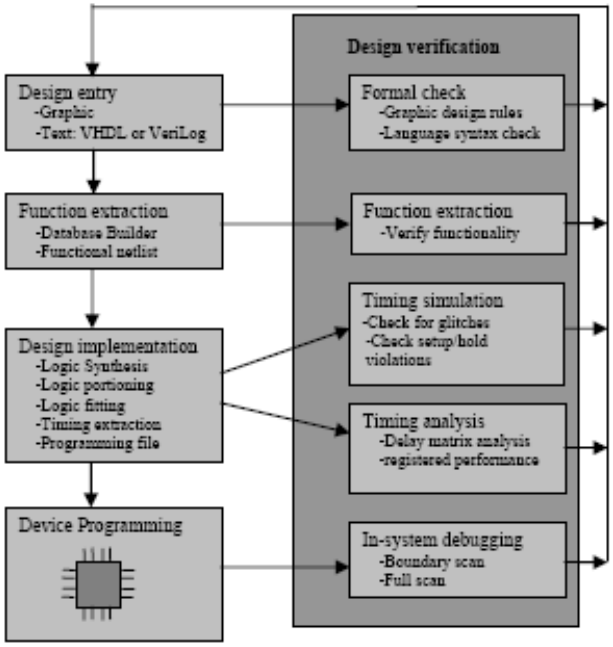


Figure 7. CAD Design Cycle

The design is first entered using graphic entry or text entry. In the next stage the functionality of the design is extracted. Then the design is targeted on a selected device and its timing is extracted. Finally the actual hardware device is programmed. At every stage the appropriate verification is done to check the working of the design. For design entry, text is preferred as it allows more control over the design compared to graphic design entry.

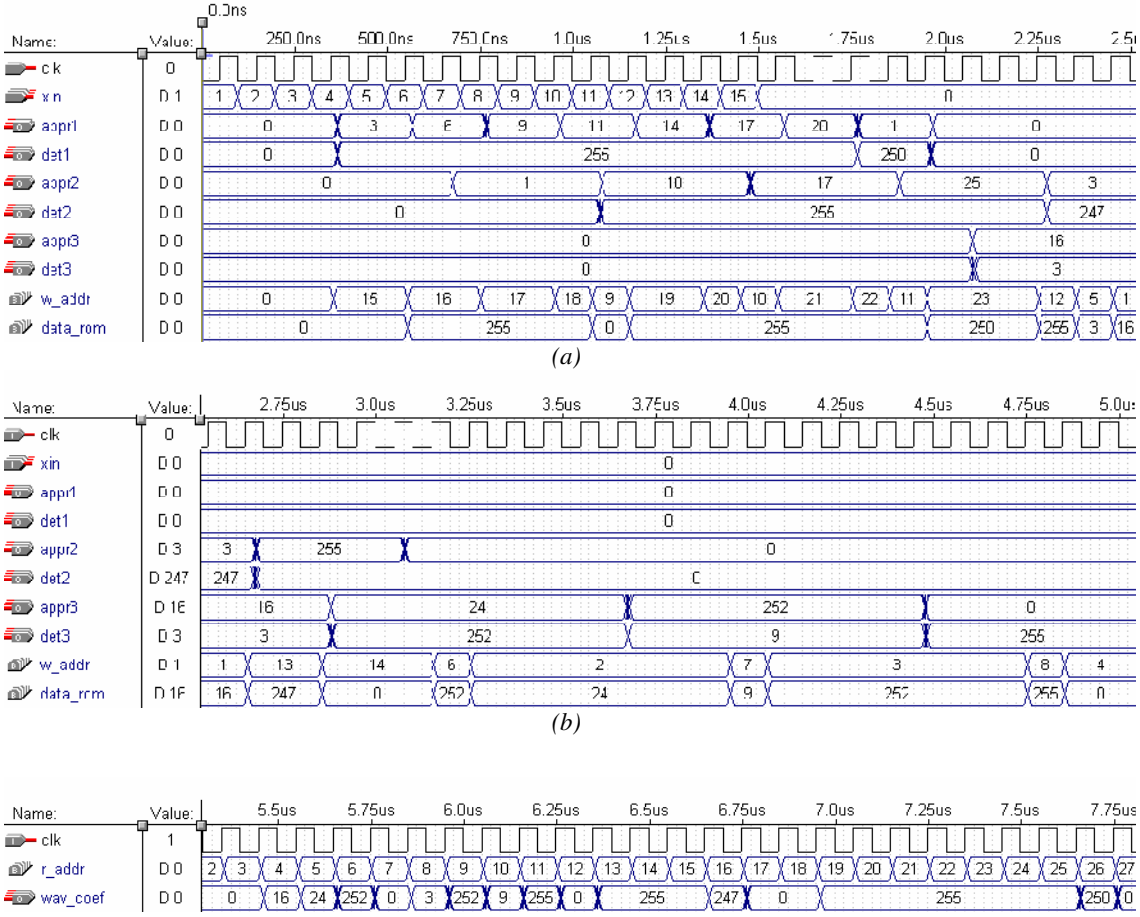
VI. IMPLEMENTATION AND RESULTS

The Altera device EPF10K70RC240 with speed grade 2 is chosen for implementation purpose so that the whole design can fit into one device. It is a 5V device and some of its features are listed in Table 2.

Table 5.1 Features of EPF10K70 devices

Feature	EPF10K70
Typical gates (logic and RAM)	70,000
Logic Elements (LEs)	3,744
Logic Array blocks (LABs)	468
Embedded Array Blocks (EABs)	9
Total RAM bits	18,432

The architecture is implemented for an input signal of 15 samples using the orthogonal Daubechies length-4 filter. The simulation waveforms generated by the Quartus simulator to verify the functionality of the design. Figure 8 shows the simulation results of the implemented architecture. Input samples of 8-bit precision are used. The coefficients at every level are scaled to have the same number of bits as the input. This allows the use of the same PEs for different levels of computation of the DWT. Thus, the architecture is modular and is easily scalable to obtain higher level of octaves.



(a)

(b)

(c)

Figure 8. Simulation results of the 3-level DWT architecture.

The hardware resources required for the implementation can be derived from the report file generated by Quartus software. The number of logic cells (LCs) used was found to be 2794, which corresponds to 74% of the total LCs available in the device. The maximum operating frequency was found to be 20.83 MHz. The power consumption calculated was 3094.32mW. The supply voltage, V_{cc}, of the

EPF10K70 device is 5V, the standby current, $I_{CCSTANDBY}$ is 0.5 mA and its I_{CC} coefficient, K is 85. The average ratio of logic cells toggling at each clock, tog_{LC}, is taken to be the typical value of 0.125.

VII. CONCLUSION

The Discrete Wavelet Transform provides a multiresolution representation of signals. The transform can be implemented using filter banks. In this paper, different architectures for the Discrete Wavelet Transform have been discussed [16, 17]. Each of them can be compared on the basis of area, performance and power consumption. Based on the application and the constraints imposed, the appropriate architecture can be chosen. For the Daubechies length-4 orthogonal filter, three architectures were implemented, i.e., the poly phase architecture, the poly phase with fully parallel DA architecture, and the poly phase with modified DA architecture. It is seen that, in applications which require low area and power consumption, e.g., in mobile applications, the poly phase with modified DA architecture is most suitable and for applications which require high throughput, e.g., real-time applications, the poly phase with DA architecture is more suitable.

The biorthogonal wavelets, with different number of coefficients in the low pass and high pass filters, increases the number of operations and the complexity of the design, but they have better SNR than the orthogonal filters. For the Daubechies 9/7 biorthogonal filter, two different architectures were implemented, i.e., the poly phase architecture, and the poly phase with modified DA architecture. It is seen that the poly phase architecture has better throughput while the poly phase with modified DA architecture has lower area and lower power consumption.

A scalable architecture for computation of higher octave DWT has been presented. The architecture was implemented using the Daubechies length-4 filter for a signal length of 15. The simulation results verify the functionality of the design. The proper scheduling of the wavelet coefficients written to the RAM ensures that, when the coefficients are finally read back from the RAM, they are available in the required order for further processing. The proposed architecture is simple since further levels of decomposition can be achieved using identical processing elements. It is easily scalable to different signal lengths and filter orders for use in different applications. he architecture enables fast computation of DWT with parallel processing [22]. It has low memory requirements and consumes low power.

VIII. FUTURE WORK

Synthesis filter banks to compute the inverse DWT, i.e., IDWT can be implemented using similar architectures for the corresponding analysis filter banks.

The architectures of the filter banks can be further improved using techniques such as Reduced Adder Graph, Canonic Signed Digit coding and Hartley's common sub expression sharing among the constant coefficients. Also, in the case of orthogonal filters with mirror coefficients, the transpose form of the filters yields a good architecture; this can be implemented and compared with the others.

The proposed higher octave DWT architecture can be extended to include symmetric signal extension. The use of symmetric extension in image compression applications reduces the distortion at boundaries of reconstructed image and provides improved SNR.

In memory intensive applications such as image and video processing, memory accesses could be the dominant source of power dissipation, as reading and writing to memory involves switching of highly capacitive address busses. Methods such as gray code addressing can be incorporated into the architecture to reduce this power dissipation.

As the DWT hierarchy increases, the required precision of the wavelet coefficients also increases. In the proposed architecture, the coefficients at all levels are scaled to have the same precision. While this reduces the hardware requirements, the accuracy of the coefficients is compromised as the number of levels increases. Therefore, the architecture can be modified to allow increased precision as the DWT level increases so as to achieve higher accuracy.

The proposed architecture can also be extended to 2-dimensional DWT computation. This can be achieved by computing the 1-dimensional DWT along the rows and columns separately. This operation requires large amount of memory and involves extensive control circuitry.

REFERENCES

[1].Gilbert Strang and Truong Nguyen, Wavelets and Filter Banks, Wellesley-Cambridge Press,1997.
[2].C. Sydney Burrus, Ramesh A. Gopinath, Haitao Guo, Introduction to Wavelets and Wavelet Transforms: A Primer, Prentice Hall, 1997.
[3].Kaushik Roy, Sharat C. Prasad, Low-Power COMS VLSI Circuit Design, John Wiley and Sons, Inc., 2000.
[4].Uwe Meyer-Baese, Digital Signal Processing with Field Programmable Gate Arrays, Springer-Verlag, 2001.
[5].http://engineering.rowan.edu/~polikar/WAVELETS/WTtutorial.html ; the Wavelet Tutorial by Robi Polikar.
[6].Robert D. Turney, Chris Dick, and Ali M. Reza, Multirate Filters and Wavelets: From Theory to Implementation, Xilinx Inc.
[7].V. Spiliotopoulos, N.D. Zervas, C.E. Androulidakis, G. Anagnostopoulos, S. Theoharis, Quantizing the 9/7Daubechies Filter Coefficients for 2D DWT VLSI Implementations, 14th International Conference on Digital Signal Processing, pages 227 -231, vol.1, July 2002.
[8].J.Ramirez, A. Garcia, U. Meyer-Baese, F. Taylor, P.G. Fernendez, A. Lloris, Design of RNS-Based Distributed Arithmetic DWT Filterbanks, Proceedings of 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pages 1193 -1196, vol.2, May 2001.
[9].Xilinx Incorporation, The Role of Distributed Arithmetic in FPGA-based Signal Processing, Xilinx application notes, San Jose, CA.
[10]. M. Alam, C.A. Rahman, W. Badawy, G. Jullien, Efficient Distributed Arithmetic Based DWT Architecture for Multimedia Applications, Proceedings of the 3rd IEEE International Workshop on System-on-Chip for Real-Time Applications, pages 333 -336, June 2003.
[11]. Ali, M., 2003. Fast Discrete Wavelet Transformation Using FPGAs and Distributed Arithmetic. International Journal of Applied Science and Engineering, 1,2:160-171
[12]. Mansouri, A. Ahaitouf, and F. Abdi. An Efficient VLSI Architecture and FPGA Implementation of High-Speed and Low Power 2-D DWT for (9, 7) Wavelet Filter, IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.3, March 2009
[13]. Mountassar Maamoun, VLSI Design for High-Speed Image Computing Using Fast Convolution-Based Discrete WaveletTransform, WCE 2009, July 1 - 3, 2009, London, U.K.
[14]. Patrick Longa, Ali Miri And Miodrag Bolic, A Flexible Design Of Filterbank Architectures For Discrete Wavelet Transforms, ICASSP 2007
[15]. Chao-Tsung Huang, Po-Chih Tseng And Liang-Gee Chen, "VLSI Architecture for Forward Discrete Wavelet Transform Based on B-spline Factorization", Journal of VLSI Signal Processing, 40, 343–353, 2005.
[16]. Chao-Tsung Huang, Po-Chih Tseng, and Liang-Gee Chen, "Analysis and VLSI Architecture for 1-D and 2-D Discrete Wavelet Transform", IEEE Transactions On Signal Processing, Vol. 53, No. 4, APRIL 2005.
[17]. Xixin Cao, Qingqing Xie, Chungan Peng, Qingchun Wang, Dunshan Yu, "An Efficient VLSI Implementation of Distributed Architecture for DWT", Multimedia Signal Processing, 2006 IEEE 8th Workshop on, pp. 364 - 367, Oct. 2006.
[18]. Kai Liu, Ke-Yan Wang, Yun-Song Li and Cheng-Ke Wu, "A novel VLSI architecture for real-time line-based wavelet transform using lifting scheme", Journal of Computer Science and Technology, Vol. 22, no. 5, September 2007.
[19]. Wang Chao and Cao Peng, "Efficient Architecture for 2-Dimensional Discrete Wavelet Transform with Novel Lifting Algorithm", Chinese Journal of Electronics, Vol.19, No.1, Jan. 2010.
[20]. Mohsen Amiri Farahani, Mohammad Eshghi, "Implementing a New Architecture of Wavelet Packet Transform on FPGA", Proceedings of the 8th WSEAS International Conference on Acoustics & Music:Theory & Applications, Vancouver, Canada, June 19-21, 2007.
[21]. Maria A. Trenas, Juan Lopez, Emilio L. Zapata, “FPGA Implementation of Wavelet Packet transform with Reconfigurable Tree Structure”, Euro micro Conference, 2000. Proceedings of the 26th Volume 1, 5-7 Sept. 2000, pp. 244 - 251 Vol.1.
[22]. Mountassar Maamoun, Abderrahmane Namane, Mehdi Neggazi, Rachid Beguenane, Abdelhamid Meraghni and Daoud Berkani, "VLSI Design for High-Speed Image Computing Using Fast Convolution-Based Discrete Wavelet Transform", Proceedings of the World Congress on Engineering, Vol I, WCE 2009, July 1 - 3, 2009, London, U.K.

Authors

D. U. Shah received the M. E. degree in Microprocessor Systems Application from The M. S. University of Baroda in the year 2008. Currently, he is working as Asst. Professor in the Department of Electronics Communication Engineering, R. K. University, Rajkot, India and simultaneously pursuing his Ph.D in EC from the Kadi Vishwavidyalaya University, Gandhinagar, India. His areas of interests are Microprocessor, Embedded Systems, VLSI, Digital Image Processing, MATLAB, etc.

C. H. Vithlani received the Ph. D. degree in Electronics Communication from Gujarat University in the year 2006. Currently, he is working as Associate Professor in the Department of Electronics Communication Engineering, Govt. Engineering College, Rajkot, India. He has published number of papers in National and International conferences and journals. His areas of interests are Microprocessor, Embedded Systems, Digital Signal and Image Processing, MATLAB, etc.