Military Technical College
Kobry El-Kobba
Cairo, Egypt

12-th International Conference
on
Aerospace Sciences &
Aviation Technology

FPGA BASED IMPLEMENTATION OF DISCRETE WAVELET TRANSFORM

Bohra[*] A., Izharuddin[**], Farooq[***] O. and Ghani[****] F.

ABSTRACT

This paper proposes a Lifting based wavelet transform implementation of Haar mother wavelet from image compression. Discrete Wavelet Transform (DWT) algorithm using the Haar wavelet is implemented and some modifications are proposed without the increasing in hardware requirements. The Peak Signal to Noise Ratio of the reconstructed image is found to be 6.409 dB better when compared to Conventional Haar Transform.

KEY WORDS

Discrete Wavelet Transform, Lifting Scheme, Image Compression, JPEG 2000

INTRODUCTION

Wavelet Transform has been successfully applied in different fields, ranging from pure mathematics to applied sciences. Numerous studies carried out on Wavelet Transform have proven its advantages in image processing and data compression. Recent progress has made it the basic encoding technique in data compression standards. 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 coding, etc. Pure software implementations of the Discrete Wavelet Transform,

however, appears to be the performance bottleneck in real-time systems. Therefore, hardware acceleration of the Discrete Wavelet Transform has become a topic of interest. Recently, a lifting-based scheme that often requires far fewer computations has been proposed for the DWT [1, 2].

The main feature of the lifting based DWT scheme is to break up the high pass and low pass filters into a sequence of upper and lower triangular matrices and convert the filter implementation into banded matrix multiplications [1, 2]. Such a scheme has several advantages, including "in-place" computation of the DWT, integer-to-integer wavelet transform (IWT), symmetric forward and inverse transform, etc. Therefore, it comes as no surprise that lifting has been chosen in the upcoming JPEG2000 standard [3].

The rest of paper is organized as follows. Section 2 describes Lifting Scheme. Section 3 describes the architecture in detail and its hardware utilization. Section 4 gives the results, and Section 5 draws conclusions.

LIFTING

Forward Transform

The algorithm can be described in three phases, namely: Split phase, Predict Phase and Update Phase, as illustrated in Figure 1. The scheme starts at data set of $\lambda_{0,k}$ where k represents the data element and zero signifies the original data level [4, 5, 6].In the first stage the data set $\lambda_{0,k}$ is split into two other sets (see Figure 1): the $\lambda_{-1,k}$. and the $\gamma_{-1,k}$. The negative indices have been used according to the convention that the smaller the data set, the smaller the index. The new data at level 1 corresponding to the data at level 0 are given as:

$$\lambda_{-1,k} = \lambda_{0,2k} \tag{1}$$

$$\gamma_{-1,k} = \lambda_{0,2k+1} \tag{2}$$

Separating the set of even samples and the odd samples does the splitting. The next step is to use $\lambda_{-1,k}$ subset to predict $\gamma_{-1,k}$ subset with the use of prediction function *P*, see Figure 1, independent of the data, so that

$$\lambda_{-1,k} = P(\lambda_{-1,k}) \tag{3}$$

Now, the set $\gamma_{-1,k}$. will be replaced by the difference between itself and its predicted value $P(\lambda_{-1,k})$. Thus,

$$\gamma_{-1,k} = \lambda_{0,2k+1} - P(\lambda_{-1,k}) \tag{4}$$

In this stage the coefficient $\lambda_{-1,k}$. is lifted with the help of the neighboring wavelet coefficients so that a certain scalar quantity *Q.*

$$Q(\lambda_{-1,k}) = Q(\lambda_{0,k})$$

For this reason a new operator *U* is applied update $\lambda_{-1,k}$. see Fig. 1

$$\lambda_{-1,k} = \lambda_{-1,k} + U(\gamma_{-1,k}) \tag{5}$$

In this phase, a *scaling function* is calculated from the previously calculated wavelet coefficients to maintain some properties among all the λ coefficients throughout every level.

Inverse Transform

The inverse transform for lifting scheme is very clear and trivial. It is just the reverse data flow in the setup of forward transform with small changes like switching additions and subtractions and also switching divisions and multiplications. Hence, the algorithm for inverse transform becomes as depicted in Fig.2.

Proposed Method

The simplest wavelet is the Haar. It is defined for a input consisting of two numbers $\lambda_{j,2k}$ and $\lambda_{j,2k+1}$. The transform is:

$$\lambda_{j,k} = (\lambda_{j,2k} + \lambda_{j,2k+1})/2 \tag{6}$$

$$\gamma_{j,k} = \lambda_{j,2k+1} - \lambda_{j,2k} \tag{7}$$

The inverse of this transformation is

$$\lambda_{j,2k} = \lambda_{j,k} + (\gamma_{j,k}/2) \tag{8}$$

$$\lambda_{j,2k+1} = \lambda_{j,k} - (\gamma_{j,k}/2) \tag{9}$$

Discrete wavelets are traditionally defined on a line of values:

$$\ldots, x_{-2}, x_{-1}, x_0, x_1, x_2, \ldots$$

For the Haar wavelet pairs $<x_{2i}, x_{2i+1}>$ are formed, such that every even value is represented as $\lambda_{j,2k}$, and the odd value as $\lambda_{j,2k+1}$.

The divisions can be skipped, or the remainder of the division can be extracted and stored elsewhere. Whatever the rounding function [.] or truncation may be, this transform is highly lossy when using integer instead of floating point number which in turns increases the error. The integer Transform is basically used as it requires less memory and hardware complexity as compare to floating point which in turns saves lot of hardware. In case of hardware implementation the rounding function requires more hardware as compare to truncation hence here truncation is used but with modification to reduce the error. At the end of the transform the even samples are replaced by the smooth coefficients and the odd by the detail coefficients.
The divisions can be skipped, or the remainder of the division can be extracted and stored elsewhere. The code for the modified version of this wavelet is:

$$\lambda_{j,k} = (\lambda_{j,2k} + \lambda_{j,2k+1} + \xi)/2 \tag{10}$$

$$\gamma_{j,k} = \lambda_{j,2k+1} - \lambda_{j,2k} \tag{11}$$

ξ is ***1*** .The inverse of this transform is.

$$\lambda_{j,2k} = \lambda_{j,k} + (\gamma_{j,k} + \xi/2) \tag{12}$$

$$\lambda_{j,2k+1} = \lambda_{j,k} - (\gamma_{j,k} + \xi/2) \tag{13}$$

This is really a wavelet that yields three components: Smooth, Detail and some ξ*'s* needed for approx lossless construction. ξ cannot be compressed much, because it contains bits with lesser significance and are random in nature. For perfect constant region ξ tends to be zero, so for lossy compression ξ can be set to zero.

Architecture of Lifting Based Haar DWT

The unit is meant to be integrated in *custom-computing machine* (CCM), as a reconfigurable functional unit. This means that the unit will co-exist with a general-purpose processor and will execute wavelet transform operations in hardware, upon occurrence. In an ideal hardware implementation, we prefer the inputs and results are integers to keep the computations simple. So ideal wavelet transform for a hardware implementation are those that map integer to integer.

A modified form of the Bi-orthogonal Haar wavelet filter is used. If there are *2k* elements in the row or column then these produce *k* low pass and *k* high pass wavelet transform coefficients. The analysis filter equations are shown below:

High pass coefficients:

$$g(k) = x(2k+1) - x(2k) \tag{14}$$

Low pass coefficients:

$$f(k) = \{x(2k) + [x(2k+1) + 1]\}/2 \tag{15}$$

Where *g(k)* is the k_{th} high pass coefficient and *f(k)* is the k_{th} low pass coefficient and *x(k)* represents the input pixel value in the k_{th} position.

The boundary conditions are handled by symmetric extension of the coefficients as shown below:

$$x[2], x[1], (x[0], x[1], \cdots, x[n-1], x[n],), x[n-1], x[n-2]$$

The synthesis filter equations are shown below:

Even samples:

$$x(2k) = f(k) - g(k)/2 \tag{16}$$

Odd samples:

$$x(2k+1) = f(k) + g(k)/2 \tag{17}$$

The modules for Forward and Inverse DWT uses shifting operation in place of division in order to reduce complexity and hardware as division by 2 is equivalent of shifting the element by one to right. The current hardware implementation processes the 512 by 512 pixel input image frame with one level of wavelet transform. In this, 512 pixels of each row are used to compute 256 high pass coefficient *g* and 256 low pass coefficients *f*. Then these coefficients are written back in a rearranged manner such that the low frequency ones are in one part of the memory and high frequency ones are in other part. Once all the 512 rows are processed, the filters are applied in the Y direction i.e. column wise, these modules are parallel and hence speed of the processing.

Forward DWT Block

To implement one level of FDWT using the lifting method the following steps are necessary:
• Split the input into coefficients at odd and even positions
• Perform a predict-step that is the operation given in Equation (13)
• Perform an update-step that is the operation given in Equation (14)

The implementation is spitted into the elementary pieces, which are additions, subtractions, and shifts as bottom-up approach is being used. These blocks can be connected for parallel operation, which is shown in Figure 3. And hence the speed of operation increases. Few registers are to be used in order to latch the data for synchronization

Inverse DWT Block

To implement one level of Inverse DWT using the lifting method the following steps are necessary (as shown in Fig.4.):
• Perform an undo update-step, which is the operation given in Eqn. (15)
• Perform an undo predict-step, that is the operation given in Eqns. (16,17)
• Merge odd and even coefficients at the input into positions

SIMULATION RESULTS

The MODELSIM tool provided the necessary environment for complete functional simulation of the target hardware, i.e., Xilinx Virtex FPGAs in this case. The Modified DWT performance is better then the existing Haar DWT. This can be seen in the Fig. 5, which shows the Peak Signal to Noise Ratio. Test has been done on three images each of size 512x512 for 1-level decomposition. The PSNR metrics are computed as per the following relation.

$$MSE=\frac{1}{512\times 512}\sum_{x=1}^{x=512}\sum_{y=1}^{y=512}\left[\left(p(x,y)-p'(x,y)\right)\right]^2 \qquad (18)$$

$$PSNR = 10\log_{10}\left(\frac{255^2}{MSE}\right) \tag{19}$$

Subsequently, the implementation tool of the Xilinx ISE 8.1i is used to implement the design using the target family VIRTEX-XC50 PQ 240. Equivalent MATLAB programs are coded to evaluate the execution time taken by the software implementation. Tests are performed on three images of size 512x512. Figure 6 shows the comparison of the speedup achieved by the hardware implementation in comparison to the software methods for images with 8 bit depth and for 2D wavelet transform.

CONCLUSIONS

This paper discusses the basic Lifting based wavelet transform from the image compression point of view. DWT algorithm using the Haar wavelet is studied and implemented in software. Since the Haar based DWT is the simplest, it is selected for hardware implementation. The Haar algorithm is studied and mapped for hardware implementation with some modification without increasing the hardware requirement; the performance is 6.409 dB better when compared to Conventional Haar Transform. Performance analysis of the hardware design with respect to software is evaluated. It is seen that the hardware speed up obtained is 52.667 times compared to software implementations. The results are promising when compared to software; however, further work needs to be done towards the extension of the system to handle color images, different wavelet analysis and synthesis schemes along with different architectures.

REFERENCES

[1] Daubechies, I. and Sweldens, W., “Factoring wavelet transforms into lifting schemes,” J. Fourier Anal. Appl., vol. 4, pp. 247–269, (1998).

[2] Sweldens, W., “The lifting scheme: A new philosophy in biorthogonal wavelet constructions,” in Proc. SPIE, vol. 2569, pp. 68–79, (1995).

[3] JPEG 2000 FINAL COMMITTEE DRAFT VERSION 1.0, 16 MARCH 2000

[4] Chen, P. Y., “VLSI Implementation for One-Dimensional Multilevel Lifting-Based Wavelet Transform”, IEEE Transactions on Computers, Vol.53 no. 4, pp- 386-398, April (2004).

[5] Vishwanath, M. and Owens, R. M., “A common architecture for the DWT and IDWT”, Proc. Int. Conf. on Application Specific Systems, Architectures and Processors (ASAP), pp. 193-198, (1996).

[6] Kolev, V., “Multiplierless Modules for Forward and Backward Integer Wavelet Transform”, International Conference on Computer Systems and Technologies CompSysTech, (2003).

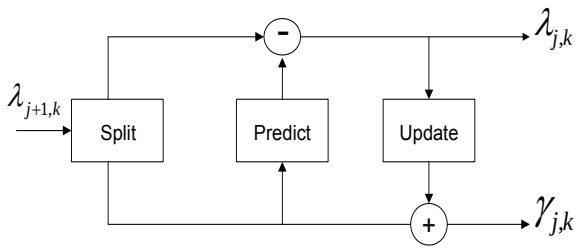

Fig.1 The Lifting Scheme: Split, Predict and Update

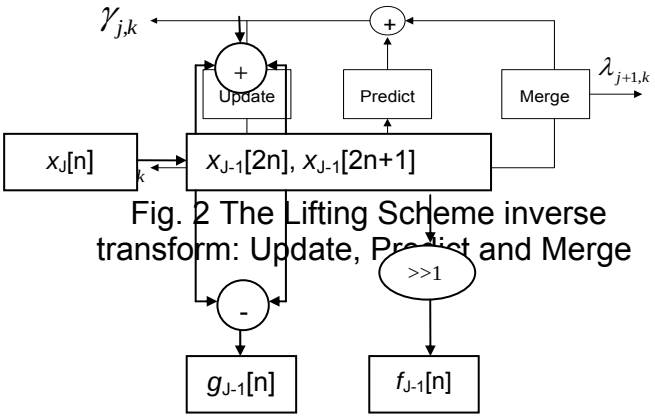

Fig. 2 The Lifting Scheme inverse transform: Update, Predict and Merge

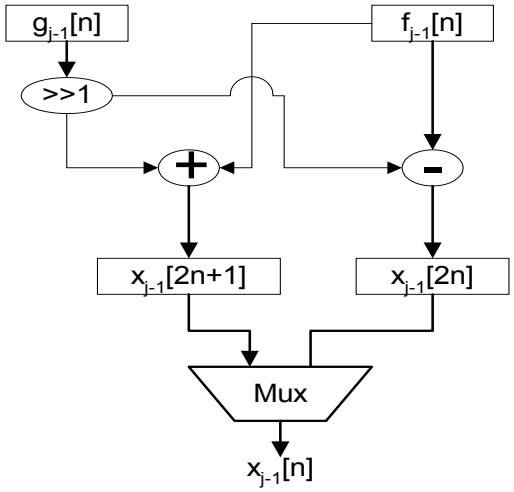

Fig.4 Inverse DWT Block

Fig.3 Forward DWT Block

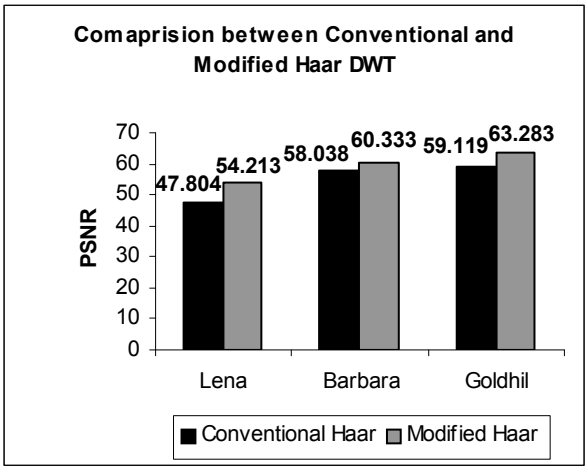

Fig.5 PSNR Comparison between Conventional and Modified Haar DWT for 1Level decomposition

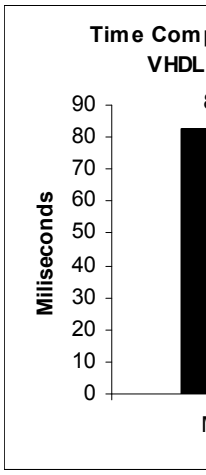

Fig.6 Perform
different pictu
between