2nd International Conference on Autonomous Robots and Agents
December 13-15, 2004 Palmerston North, New Zealand

VLSI Implementation of Discrete Wavelet Transform (DWT) for Image Compression

Abdullah Al Muhit, Md. Shabiul Islam and Masuri Othman*

Faculty of Engineering
Multimedia University (MMU)
Jalan multimedia, Cyberjaya, Selangor 63100,Malaysia.
E-mail:muhit@mmu.edu.my
Phone : +6-03-83125119
Fax : +6-03-83183029

*Institute of Microengineering and Nanoelectronics(IMEN)
University Kebangsaan Malaysia (UKM)
43600 UKM, Bangi, Selangor DE, Malaysia.

Abstract

This paper presents an approach towards VLSI implemention of the Discrete Wavelet Transform (DWT) for image compression . The design follows the JPEG2000 standard and can be used for both lossy and lossless compression. In order to reduce complexities of the design, linear algebra view of DWT and IDWT has been used in this paper. This design can be used for image compression in a robotic system.

Key words: Discrete wavelet transform(DWT), image compression, thresholding, VLSI design, JPEG 2000.

1 Introducation

With the use of more and more digital still and moving images, huge amount of disk space is required for storage and manipulation purpose. For example, a standard 35-mm photograph digitized at 12μm per pixel requires about 18Mbytes of storage and one second of NTSC-quality color video requires 23 Mbytes of storage [1]. That is why image compression is very important in order to reduce storage need. Digital images can be compressed by eliminating redundant information present in the image, such as spatial redundancy, spectral redundancy and temporal redundancy. The removal of spatial and spectral redundancy is often accomplished by transform coding, which uses some reversible linear transform to decorrelate the image data [2]. JPEG is the most commonly used image compression standard in today’s world. Joint Photographic Experts Group (JPEG) is an ISO standard committee with a mission on “Coding and compression of still images”. It’s jointly supported by ISO and ITU-T. But researchers have found that JPEG has many limitations. In order to overcome all those limitations and to add on new improved features, ISO and ITU-T has come up with new image compression standard, which is JPEG2000. The JPEG2000 is intended to provide a new image coding/decoding system using state of the art compression techniques, based on the use of wavelet technology. Its wide range of usage includes from portable digital cameras through pre-press, medical imaging. Right now, JPEG2000 is composed of 6 main parts [3]. The two important parts in the coding/decoding processes of JPEG2000 are Wavelet Transform and Arithmetic Coding. Since the later is widely implemented, this paper focuses on the hardware implementation of discrete wavelet transform (both FDWT and IDWT), which will provide the transform coefficients for later stage and is one key part of JPEG2000 implementation.

2 Overview of DWT Algorithm

In this section the theoretical background and algorithm development is discussed. The first recorded mention of what is now called a "wavelet" seems to be in 1909, in a thesis by Alfred Haar. An image is represented as a two dimensional (2D) array of coefficients, each coefficient representing the brightness level in that point. When looking from a higher perspective, it is not possible to differentiate between coefficients as more important ones, and lesser important ones. But thinking more intuitively, it is possible. Most natural images have smooth color variations, with the fine details being represented as sharp edges in between the smooth variations. Technically, the smooth variations in color can be termed as low frequency components and the sharp variations as high frequency components. The low frequency components (smooth variations) constitute the base of an image, and the high frequency components (the edges which give the detail) add upon them to refine the image, thereby giving a detailed image. Hence, the averages/smooth variations are demanding more importance than the details [4].

In wavelet analysis, A signal can be separated into approximations or averages and detail or coefficients. Averages are the high-scale, low frequency components of the signal. The details are the low-scale, high frequency components.

If we perform forward transform on a real digital signal, we wind up with twice as much data as we started with. That's why after filtering downsampling has to be done. The inverse process is how those components can be assembled back into the original signal without loss of information. This process is called reconstruction or synthesis. The mathematical manipulation that effects synthesis is called the inverse discrete wavelet transform. The original signal, is reconstructed from the wavelet coefficients. Where wavelet analysis involves filtering and downsampling, the wavelet reconstruction process consists of up sampling and filtering.

The DWT algorithm consists of Forward DWT (FDWT) and Inverse DWT (IDWT) which are shown in fig. 1 and 2 respectively.

The FDWT can be performed on a signal using different types of filters such as db7, db4 or Haar. The Forward transform can be done in two ways, such as matrix multiply method and linear equations. In the FDWT, each step calculates a set of wavelet averages (approximation or smooth values) and a set of details. If a data set s_0, s_1, ... s_{N-1} contains N elements, there will be N/2 averages and N/2 detail values. The averages are stored in the upper half and the details are stored in the lower half of the N element array.

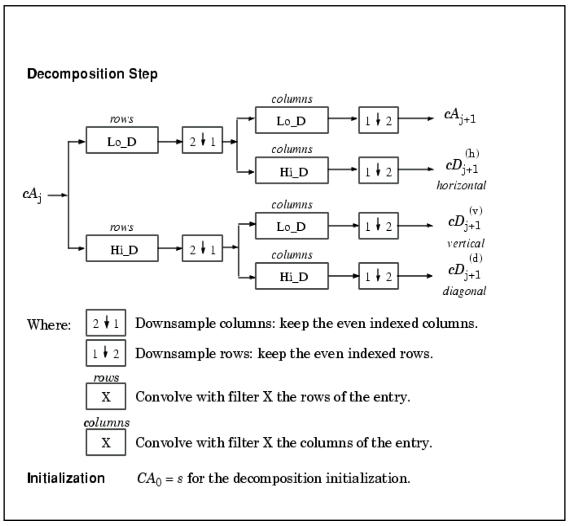
Decomposition Step
rows
Lo_D
2↓1
columns
Lo_D
1↓2
cA_{j+1}
columns
Hi_D
1↓2
$cD^{(h)}_{j+1}$
horizontal
cA_j
rows
Hi_D
2↓1
columns
Lo_D
1↓2
$cD^{(v)}_{j+1}$
vertical
columns
Hi_D
1↓2
$cD^{(d)}_{j+1}$
diagonal
Where:
2↓1 Downsample columns: keep the even indexed columns.
1↓2 Downsample rows: keep the even indexed rows.
rows X Convolve with filter X the rows of the entry.
columns X Convolve with filter X the columns of the entry.
Initialization $CA_0 = s$ for the decomposition initialization.

Figure 1: Block diagram of FDWT.

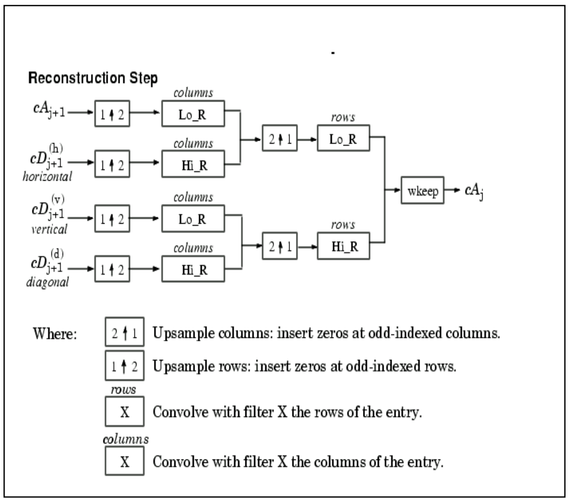
Reconstruction Step
cA_{j+1}
1↑2
columns
Lo_R
$cD^{(h)}_{j+1}$
horizontal
1↑2
columns
Hi_R
2↑1
rows
Lo_R
$cD^{(v)}_{j+1}$
vertical
1↑2
columns
Lo_R
$cD^{(d)}_{j+1}$
diagonal
1↑2
columns
Hi_R
2↑1
rows
Hi_R
wkeep
cA_j
Where:
2↑1 Upsample columns: insert zeros at odd-indexed columns.
1↑2 Upsample rows: insert zeros at odd-indexed rows.
rows X Convolve with filter X the rows of the entry.
columns X Convolve with filter X the columns of the entry.

Figure 2: Block diagram of IDWT.

In the matrix multiply method [6] of the forward transform, the first average is calculated by the inner product of the signal [s_0, s_1, ... s_{N-1}] and the scaling vector, of the same size, [h_0, h_1, 0, 0,...,0]. The first detail is calculated by the inner product of the signal and the wavelet vector [g_0, g_1, 0, 0, …., 0]. The next average and detail values are calculated by shifting the scaling and wavelet vectors by two and then calculating the inner products. In the case of the wavelet transform using Haar filter, the Scaling function coefficients are $h_0 = 0.5$, $h_1 = 0.5$ and Wavelet function coefficients are $g_0 = 0.5$, $g_1 = -0.5$. The scaling and wavelet values for the forward transform are shown in the fig. 3 in matrix form.

$$\begin{array}{ccccc} h_0 & h_1 & 0 & 0 & \cdots \\ g_0 & g_1 & 0 & 0 & \cdots \\ 0 & 0 & h_0 & h_1 & \cdots \\ 0 & 0 & g_0 & g_1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{array}$$

Figure 3: Transform matrix of vector coefficients.

On the other hand, the following linear equations no (1) and (2) can be used to calculate an average (a_i) and a detail (d_i or c_i) from an odd and even element in the input data set :

$$a_i = (s_i + s_{i+1})/2 \qquad (1)$$

$$d_i = (s_i - s_{i+1})/2 \qquad (2)$$

After the FDWT stage, the resulting average and detail values can be compressed using thresholding method [7].

In the IDWT process, to get the reconstructed image, the wavelet details and averages can be used in the matrix multiply method and linear equations. For the matrix multiply method, the Scaling function coefficients are $h_0 = 1$, $h_1 = 1$ and Wavelet function coefficients are $g_0 = 1$, $g_1 = -1$.

The IDWT process can be performed using the following linear equations (3) and (4).

$$s_i = a_i + d_i \qquad (3)$$

$$s_{i+1} = a_i - d_i \qquad (4)$$

A single wavelet transform step using a matrix algorithm involves the multiplication of the signal vector by a transform matrix, which is an N X N operation (where N is the data size for each transform step). In contrast, linear equations need only N operations. In practice matrices are not used to calculate the wavelet transform. The matrix form of the wavelet transform is both computationally inefficient and impractical in its memory consumption.

The first step of the forward transform (FDWT) for an eight element signal is shown in figure 4. Here signal is multiplied by the forward transform matrix with haar filter coefficients.

For the inverse transform (IDWT), here the signal is reconstructed by the following multiplication. It is shown in figure 5.

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \Leftarrow \begin{bmatrix} a_0 \\ c_0 \\ a_1 \\ c_1 \\ a_2 \\ c_2 \\ a_3 \\ c_3 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & -\frac{1}{2} \end{bmatrix} \bullet \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix}$$

Figure 4: FDWT using an eight element signal.

$$\begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \bullet \begin{bmatrix} a_0 \\ c_0 \\ a_1 \\ c_1 \\ a_2 \\ c_2 \\ a_3 \\ c_3 \end{bmatrix} \Leftarrow \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Figure 5: IDWT using an eight element signal.

3 Implementation of DWT and IDWT algorithm

In this section we will discuss how to implement 2D FDWT and IDWT together with thresholding in MATLAB. In the FDWT part the input data will be transferred from time domain to scale domain. Then in thresholding part some of the coefficients will be set to zero and in the IDWT part the coefficients will be transferred back into time domain. The block diagram of MATLAB implementation is shown in the figure 6.

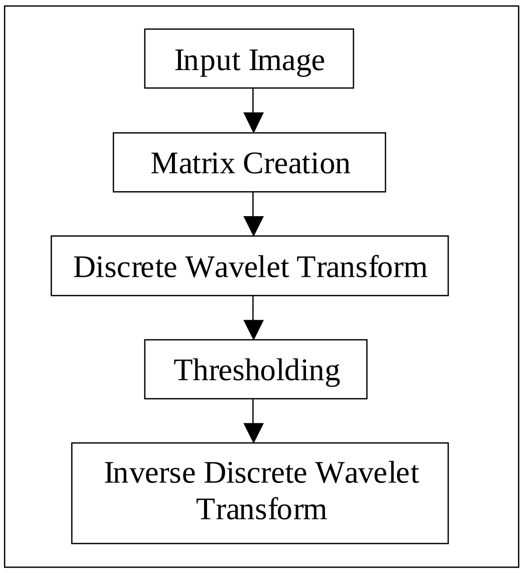
Input Image
Matrix Creation
Discrete Wavelet Transform
Thresholding
Inverse Discrete Wavelet Transform

Figure 6: Steps in MATLAB implementation.

While implementing the algorithm in MATLAB the matrix multiplication method has been used. We have tested the 'barbara.png'[8] as the image input file and also 8 randomly chosen image co-efficients for MATLAB simulation.

After we have achieved satisfactory result in MATLAB we proceed to the next stage where we translate the code into VHDL. The development of algorithm in VHDL is different in some aspects. The main difference is unlike MATLAB, VHDL does not support many built in functions such as convolution, max, mod, flip and many more. So while implementing the algorithm in VHDL, linear equations of FDWT and IDWT is used. The floating point operations have been avoided here. The VHDL code is compiled and simulated using Aldec Active HDL 3.5 software. 8 image co-efficients that have been used in MATLAB were also used in VHDL simulation.

Next, the VHDL codes were synthesized using the synthesis tool "Synplify" which have produced "gate-level architecture" for VLSI implementation. Finally, the design codes of DWT have been downloaded into FPGA board for verifying the functionality of the design. The simulation results and also the synthesis results are presented in following sections 3 and 4.

3 Simulation results

"barbara.png" was selected as the 2D image that will be used as input for the MATLAB simulation. The simulation results are given in figure 7.

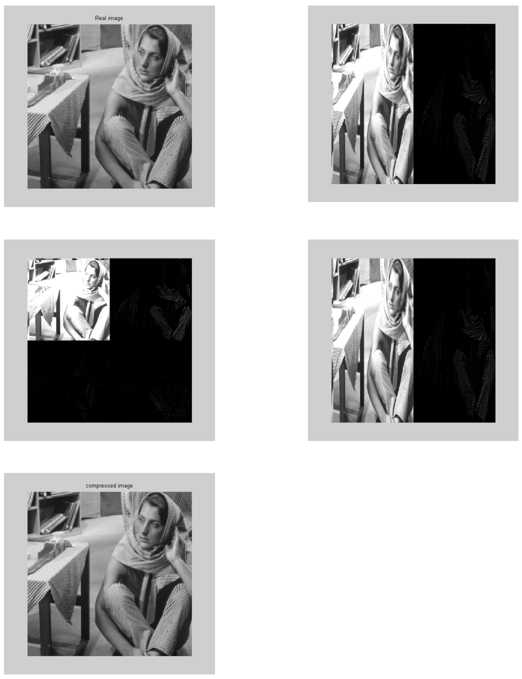

Figure 7: MATLAB Simulation results on "barbara.png" image.

8 randomly chosen image co-efficients have been simulated using both MATLAB and VHDL and the result is shown in table 1. From the table we can see that input co-efficients and the reconstructed co-efficients are exactly similar while the threshold value is very small.

Table 1: Comparison of MATLAB and VHDL simulation

Input Co-efficients	MATLAB simulation result	VHDL simulation result
1100	1100	1100
1200	1200	1200
1000	1000	1000
1200	1200	1200
1400	1400	1400
1200	1200	1200
1400	1400	1400
1200	1200	1200

4 Synthesis Results

Synthesis is performed to transform the VHDL code into logic gate level using Synplify 7.0 by Synplicity. The physical hardware layout is generated using synthesis tools. It is a great design approach to take the VHDL code as a basis and translate it automatically into a netlist. After synthesis, 3 RTL views and 106 Technology views have been achieved. One sample Technology view is given in figure 8. The achieved system frequency is 10.8 MHz. In this design 64 bit registers have been used.

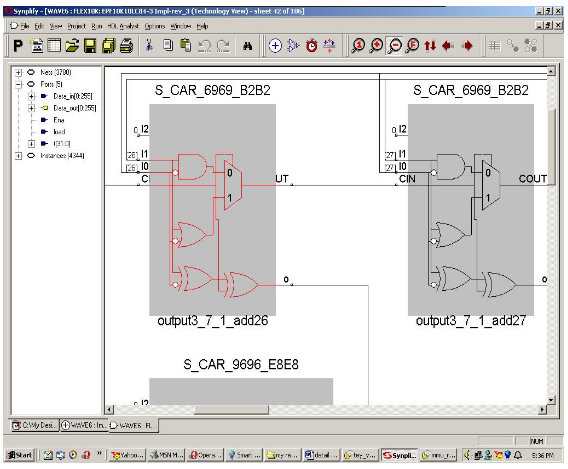

Figure 8: Zoomed Technology View of the Design

5 Discussion

This work focuses on the digital VLSI implementation of DWT (FDWT and IDWT) algorithm using VHDL. In order to increase the performance and reduce computational complexities many modifications have been made. ‘db7’ filter can be used in MATLAB for better performance while ‘haar’ can be used in VHDL for simplification. We have found that higher decomposition is expected to cause higher compression ratio. In order to reduce the complexity and increase computation speed, linear algebra equations of DWT are used while implementing the algorithm in VHDL. This linear algebra approach gives almost the same output. But it takes very less resources. Also in order to balance between the time needed by serial input and resources needed by parallel input, a more suitable approach has been taken where 8 image co-efficients are processed at a time.

6 Conclusion

In this paper an improved and efficient Discrete Wavelet Transform algorithm has been developed and tested in MATLAB. Then the algorithm has been modified for implementation in VHDL using Aldec Active HDL 3.5. From the satisfactory simulation results we can conclude that algorithm works properly. This VHDL code was also synthesized to achieve the gate level architecture of the design which is now ready to be implemented in hardware.

7 References

[1] “Compressing still and moving images with wavelets” by M L Hilton, B D Jawerth and Ayan Sengupta, published in the journal “Multimedia Systems” vol – 2 no-3,1994.

[2] Rabbani M, Jones P (1991) Digital Image Compression Techniques.(SPIE tutorial texts in optical engineering, vol TT7) SPIE press, Washington.

[3] JPEG official website, www.jpeg.org/jpeg2000.html , visited on 18/7/03

[4] Tutorial by Satish http://www.debugmode.com/imagecmp/ , visited on 18/7/03

[5] Wavelets and Filter Banks by Gilbert Strang and Truong Nguyen, Wellesley-Cambridge Press, 1997

[6] Ripples in Mathematics: the Discrete Wavelet Transform by Arne Jense and Anders la Cour-Harbo, Springer, 2001

[7] “Image Compression Using The Haar Wavelet Transform” by Colm Mulcahy, Spelman College Science & Mathematics Journal, Vol 1, No 1, April 1997, 22-31

[8] A website for Public-Domain Test Images for Homeworks and Projects http://homepages.cae.wisc.edu/~ece533/images/, visited on 18/7/03