HARDWARE & SOFTWARE CODESIGN OF A JPEG2000 WATERMARKING ENCODER

Jose Antonio Mendoza, B.S.

Thesis Prepared for the Degree of

MASTER OF SCIENCE

UNIVERSITY OF NORTH TEXAS

December 2008

APPROVED:

Elias Kougianos, Major Professor
Saraju P. Mohanty, Co-Major Professor
Robert B. Hayes, Committee Member
Nourredine Boubekri, Chair of the Department of Engineering Technology
Costas Tsatsoulis, Dean of the College of Engineering
Sandra L. Terrell, Dean of the Robert B. Toulouse School of Graduate Studies

Mendoza, Jose Antonio. Hardware and software codesign of a JPEG2000 watermarking encoder. Master of Science (Engineering Systems) December 2008, 77 pp., 6 tables, 59 figures, references, 38 titles.

Analog technology has been around for a long time. The use of analog technology is necessary since we live in an analog world. However, the transmission and storage of analog technology is more complicated and in many cases less efficient than digital technology. Digital technology, on the other hand, provides fast means to be transmitted and stored. Digital technology continues to grow and it is more widely used than ever before. However, with the advent of new technology that can reproduce digital documents or images with unprecedented accuracy, it poses a risk to the intellectual rights of many artists and also on personal security. One way to protect intellectual rights of digital works is by embedding watermarks in them. The watermarks can be visible or invisible depending on the application and the final objective of the intellectual work.

This thesis deals with watermarking images in the discrete wavelet transform domain. The watermarking process was done using the JPEG2000 compression standard as a platform. The hardware implementation was achieved using the ALTERA DSP Builder and SIMULINK software to program the DE2 ALTERA FPGA board. The JPEG2000 color transform and the wavelet transformation blocks were implemented using the hardware-in-the-loop (HIL) configuration.

ACKNOWLEDGMENTS

I would like to thank my thesis advisor Dr. Elias Kougianos and Dr. Saraju P. Mohanty for their support and advice. I really appreciate all the help and guidance they gave me with my research and I thank them for providing me with the necessary resources to successfully accomplish my research objectives. Without the kind help of my thesis advisors and members of my committee, it would have been impossible for me to successfully complete my thesis.

I would also like to thank my dear host family who has been there for me at all times. Thank you Howard and Sarah Stone for always believing in me. I am very fortunate to have met people like Silvia, Magy and her family, Mark, Bri, Bobby, and all the professors in the department of engineering technology who made my life in college much more pleasant and bearable.

I thank God for giving me the opportunity of coming to this country and finishing my bachelor's and master's degree at UNT. I dedicate this thesis to my beloved parents, family and host family and friends.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

For years, financial institutions and big companies have struggled to protect themselves against crimes that involve fraudulent documents and forgery. As the ability to reproduce documents using computer technology and scanners increases, so does the incidence of forgery and similar crimes (1). In order to hinder criminals from these illegal activities, government agencies and scientific communities have joined efforts to create ways to encrypt special data or symbols into documents that can be set apart from imitations (2)(3). Unfortunately, many of these encryption methods can be cracked or tempered by having the right tools and proper training. Many of these methods are not very effective because criminals are able to find the data encrypted on the material that is being forged. Heat-sensitive ink and UV light technologies have been widely used by banks and industry to hide data. Thus, criminals know what to look for or when to be suspicious of certain documents. A good approach to encrypt data is to hide it even from criminals. This approach is a science by itself and it is called steganography. A procedure like this can be achieved in different ways, one of which is by inserting watermarks using frequencies above or below the human sensorial system (HSS) frequency spectrum (4).

1.1 Border Security and Intellectual Property Protection

Not only can this approach be used to keep banks and financial institutions from losing money, but it can also be used to enhance national border security. During times of war, higher border security is very important and necessary to avoid a possible enemy attack. In the United States, for example, millions people cross the U.S borders illegally every year (3). National security is severely jeopardized since illegal intruders are able to obtain fake identifications and

other important documents once inside the U.S. Border security is increasing and better technology is being used to keep illegal intrusions through the Canadian and Mexican borders. However, having more secure passports and proper documentation could make it even harder for the enemy to enter U.S soil. This approach could be used to watermark names, social security numbers, and even phone numbers on IDs, passports or visas more effectively (4). Equipment to encode watermarks and decode such information could be assigned only to the homeland security department to control it and manage it. Verifying and ensuring the authenticity of documents is a step towards protecting governmental institutions, banks and even individuals.

Another reason to use hidden watermarks is to protect intellectual property. A professional photographer, for example, is likely to invest thousands of dollars on high class equipment to provide good quality pictures. The investment made by photographers and sometimes artists is jeopardized when pictures and computer artwork are replicated with amazing accuracy using image processing software and scanning devices. Pictures and artwork are very vulnerable to forgery attacks when used on websites in the Internet. Once an image or picture has been placed in the internet, it is nearly impossible to protect it from being downloaded by internet users. The image is unprotected and false ownership claims can be made if the image is modified enough from its original. The infringement of intellectual property rights is very common nowadays and it is hard to keep control over who owns the rights to what. However, it is essential to keep finding ways to protect the intellectual work of designers and artists to avoid further economic losses that could affect more areas of the media industry (5). Watermarking provides a useful and elegant way to deter copy right infringement. By inserting watermarks in images and artwork, the original designer or creator can claim legal ownership over them (4). In addition to adding watermarks, intellectual work can be further protected by having the quality

of the work be significantly damaged if the watermarks are removed. This helps original designers make the necessary claims or changes to protect their intellectual work.

CHAPTER 2

THE NEED FOR COMPRESSION

Digital technology is growing and becoming more widely used than it was a few decades ago. With the invention of computers, cell phones, and internet, digital representation of data has become imperative and so the need to create ways to store it more efficiently (6). Digital technology has occupied nearly all areas of the everyday life for most people nowadays. Kitchens, cars, appliances and even furniture have now digital technology embedded inside. The transition from analog to digital is now quite evident and it is primarily due to the many advantages that can be found in digital technology. As compared with analog data, digital data is more robust and less sensitive to noise due to electromagnetic fields and other external factors. Analog data, on the other hand, can be easily lost or degraded if an analog channel is located near magnetic fields or extreme temperature changes.

2.1 Reasons for Compression Data

In addition to being more resistant to noise, digital data is also more easily reproduced and stored. The reproduction of analog data requires expensive equipment and its storage does require large amounts of material. An audio tape, for example, requires several meters of magnetic tape in order to store a dozen songs. On the other hand, the same tape can be digitalized and compressed to reduce the amount of material needed to reproduce the same amount of data. However, even as data becomes digitalized, it does require some compression in order to be stored and transmitted more efficiently. Memory space and bandwidth are two of the most common reasons for compressing digital data (6). Prices for memory space and devices have gone down; nevertheless, the use of bandwidth for telecommunication purposes continues to

increase, thus the need for more efficient compression algorithms is great. A sample of a low-resolution video file of 30 frames per second, containing 640x480 pixels per frame, for example, requires storage space of about 95 gigabytes. The transmission of this video file through a regular communication channel is impossible in its original form. By compressing large files, not only is it possible to send it through a transmission channel, but it is also easier to store it more efficiently (6).

2.2 Data Compression and CODEC

Data compression is a method to compress an input signal and represent it with less number of elements or bits. Figure 1 shows a simplified diagram of a compression system and a decompression system. When connected both are connected as a single system they form a CODEC.

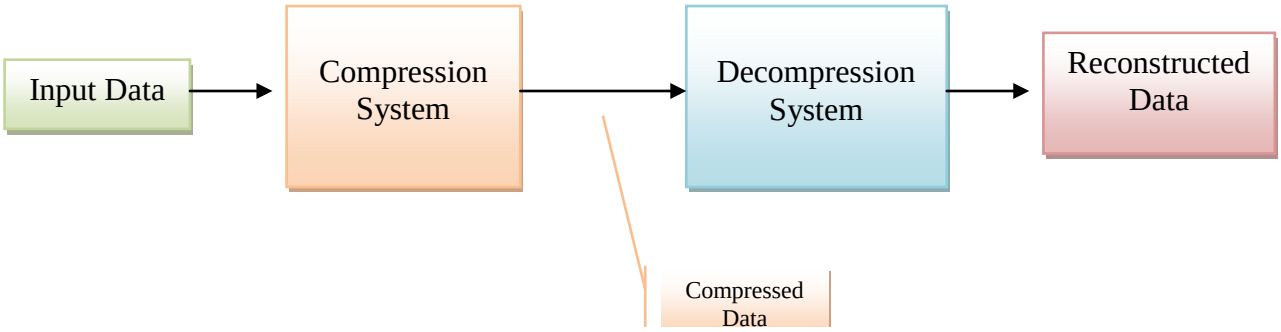

Figure 1 - CODEC

A CODEC that outputs data with identical characteristics of the input data it is said that the systems is operated using “lossless” compression techniques. However, if the output and input data are not exactly alike, then “lossy” techniques are being used (7)(8).

2.3 Different Media & Data Compression Model

Data compression can be applied to several types of media. Video, audio, and still images

are good examples of media being compressed nowadays to optimize its transmission or storage requirements. Figure 2 shows a simplified block diagram of a data compression model.

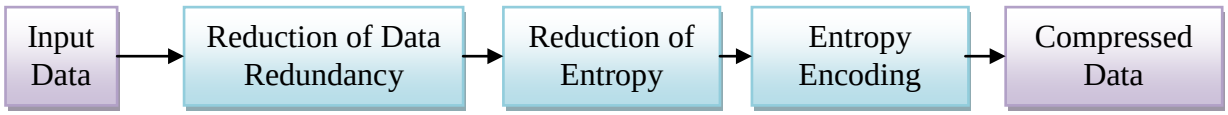

Figure 2 - Data Compression Model for Compressions Systems

The amount of redundant data found on still images, video and audio is usually quite large. The higher the digital quality of the media, the more redundant data is contained in it. The amount of redundancy can be reduced greatly if only human senses are involved on the quality appreciation of the media to process. Human perception, such as eyes or ears, is quite limited to a certain frequency spectrum. Redundancy contained in an image above the human sensorial frequency threshold will be completely invisible to our senses. This type of redundancy is called "perceptual" redundancy (9). Redundancy in still images, for example, is mostly due to the pixel correlation on their frequency and spatial values. The variation on these values is so small that our eyes cannot detect it. Hence it is not "perceived" and it can be taken out without visually affecting the quality of the picture. By taking out redundant data in an image, further compression is achieved.

2.4 Image Compression Framework

Figure 3 shows a general compression framework and the two types of compression schemes which are lossy and lossless data compression. If human sensing is involved on the quality appreciation of the reconstructed data, then further reduction of redundancy can be achieved by using quantization (9). Quantization is an irreversible process used in lossy

compressions and it should be avoided if loss of information on media could cause undesired results. Quantization is part of the “additional preprocessing” block in Figure 3.

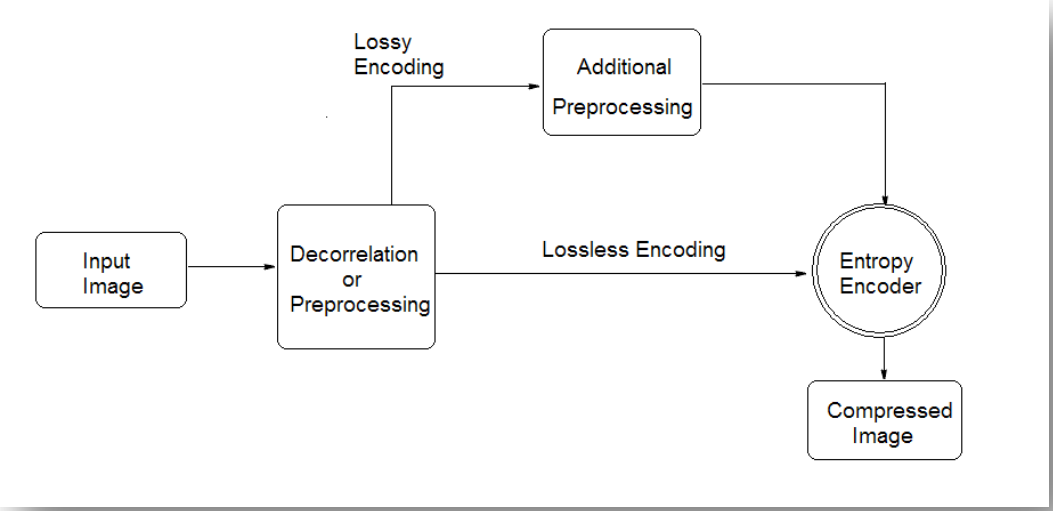

Figure 3 - General Image Compression Framework

In cases in which data are text or numbers, one must decide carefully if lossy or lossless compression techniques should be used. Data is composed of pertinent information and redundancy. In some cases, redundancy is desired and even essential when decompressing data (6). For example, whenever data compression takes place, redundancy is taken out and only the compressed data is processed for transmission or storage. However, when decompression takes place, redundancy must be re-inserted into the data in order to represent it in its original form. Some amount of redundancy in text and numerical data bases is necessary in order to make sense of what the information is about after decompression has taken place. For numerical data bases it would be impossible to determine the order of data properly if a number was missing after a lossy compression took place. A document would be hard to read if letters “i" and “e” were left out because they are the most redundant letters in the English language (6). Careful analysis of the data is essential when compressing different types of images. Before proceeding to compress

any image, one must decide whether to use lossy or lossless compression techniques. In some cases when the information conveyed in a picture is not critical, further compression of the image can take place by using lossy compression techniques. By quantizing and reducing the number of bits representing the image even further, the image will lose perceptual information and provide efficient storage and transmission capabilities (9). However, if an important image such as a mammography is being compressed, it is obvious that lossless compression must be used and quantization must be avoided. An improper analysis of these types of images could be disastrous for the patient being examined.

In summary, image compression is a method to represent an image with less numbers of bits without damaging the visual and information quality of the image itself. Depending on the compression technique, whether lossy or lossless, the reduction of redundancy can be small or great. If lossy compression is used, the amount of redundancy eliminated is greater than using lossless compression. However, using lossy compression introduces quantization errors on the final decompressed image. Thus, depending on the application and the final user to analyze the compressed file, the appropriate compression technique should be used.

2.5 Redundancy Types

In images, compression is accomplished by removing one or more of the three most common types of redundancies which are coding redundancy, interpixel redundancy and psychovisual redundancy (9).

2.5.1 Coding Redundancy

Coding redundancy is the result of coding symbols with short and long lengths using the

same code length for all coding regardless of their probability to occur. The gray levels of a gray-scale image, for example, generate symbols of different length and with different probabilities of occurring. By assigning short codes to the gray levels that occur the most, and long codes to the gray levels that occur less frequently, redundancy is reduced and the encoding process is optimized. Coding redundancy is present when less than optimal symbols or words are encoded and no consideration is given to the length of the codes (6).

2.5.2 Interpixel Redundancy

Interpixel redundancy occurs when the pixel values of an image are correlated. In most cases, this type of redundancy can be reduced by applying coding techniques and mathematical transformations to better represent pixel information (10). An example of a coding technique to reduce interpixel redundancy is the variable-length Huffman coding technique. Figure 4 shows two gray scaled images with different pixel alignment. Image A is composed of several pixels that show no specific pattern, whereas Image B shows a more adjacent or similar pattern among pixels. The variable-length Huffman coding technique was used to analyze the amount of entropy contained in the A and B images in Figure 4.

The images entropy analysis was done using MATLAB. See Table 1.

Table 1 - MATLAB Random Entropy and Coding Analysis

```
f2=imread('Aligned Matches.tif');     %Read image B
c2=mat2huff(f2);                      %Apply coding technique
entropy(f2)                           %Display amount of information
ans =  7.3505
 imratio(f2,c2)                       % Compare ratios between coded and original images
ans = 1.0821
```

(table continues)

```
f1=imread('Random Matches.tif');    %Read Image A
c1=mat2huff(f1);                    %Apply coding technique
entropy(f1)                         %Display amount of information
 ans = 7.4253

 imratio(f1,c1)                     %Compare ratios between coded and original
images
 ans = 1.0704
```

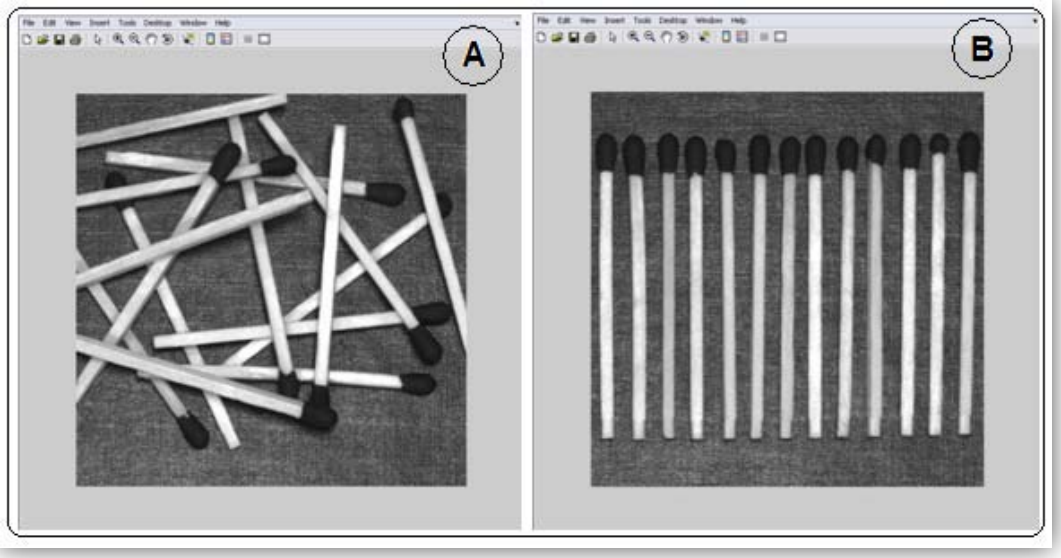

Figure 4 - Interpixel Redundancy Example

Even though Image B has a more predictable pixel pattern than the Image A, the entropy level remains almost the same after using Huffman coding. No much advantage using the Huffman coding technique is found with these two images. In pictures where the majority of pixels have similar values and are adjacent to each other, the difference between adjacent pixels can be used to represent the image. Transformations that use this type of analysis are called mappings (9). An example of predictive mapping is given in Figure 8. When mapping can be

reversed, that is when input data can be reconstructed from a given transformation, it is called lossless. Figure 5 shows a lossless mapping predictive coding model.

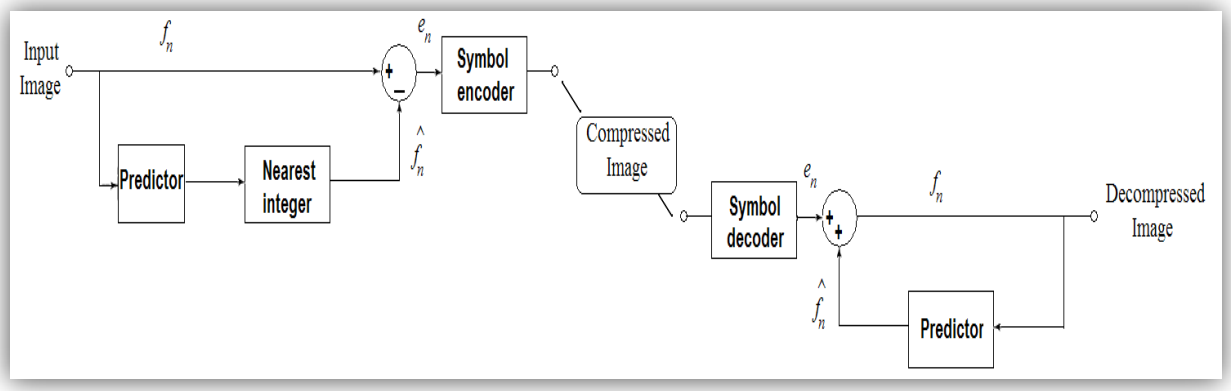

Figure 5 – Predictive Coding Model

The model shown in Figure 5 consists of an encoder and decoder, both containing the same predictor block. As conforming pixels from the input image (*fn*) flow successively and are processed by the encoder, the predictor creates an anticipated value based on past values of pixels previously processed. The values generated from the predictor are then rounded up to the nearest integer and denoted as $\hat{f}_n$. The difference between the predicted and actual value of the single pixel been analyzed is called “new information” or prediction error.

Equation 1 - Prediction Error e_n

$$e_n = f_n - \hat{f}_n$$

The prediction error is subsequently coded using a variable-length code by the symbol encoder. Subsequently, the symbol encoder generates the next element of compressed data stream. The output is a series of bits representing a compressed version of the input image.

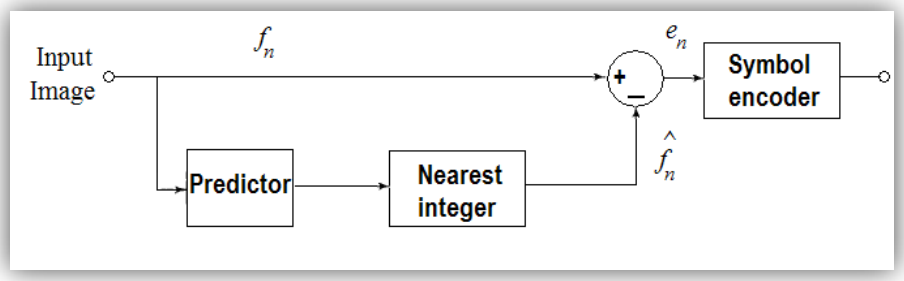

Figure 6 - Lossless Predictive Coding Model

The lossless predictive coding is able to reduce interpixel redundancy of closely spaced pixels by taking out only new information in every pixel and encoding it (9). To decode a compressed image, the inverse process is used by using a predictive decoder. Figure 7 shows the decoder stage of the lossless predictive coding model.

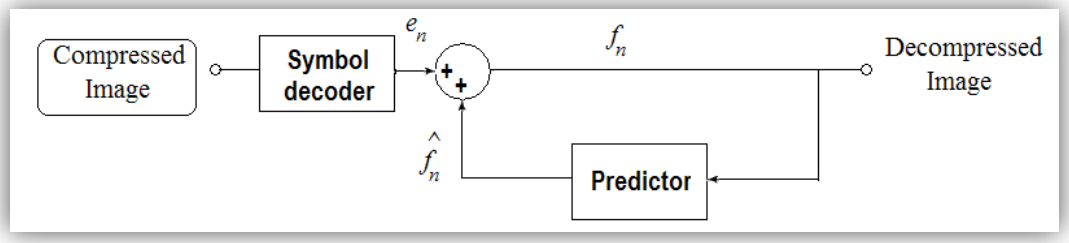

Figure 7 - Lossless Predictive Decoding Model

The compressed data stream representing the compressed image is fed into the symbol decoder to reconstruct the prediction error e*n*. In order to reconstruct the pixels from the original image *fn*, the inverse operation from Equation 1 is performed

Equation 2 – Pixel Reconstruction

$$f_n = e_n + \hat{f}_n$$

$\hat{f}_n$ is found by using the Equation 3:

Equation 3 – Prediction

$$\hat{f}_n = round\left[\sum_{i=1}^{m} \alpha_i f_{n-i}\right]$$

Where α_i = Prediction coefficients

The prediction of pixels is achieved by taking into account the values of previous pixels. The variable *m* represents the order of the linear predictor, the *round* is a function used to "round" up the coefficients to the nearest integer and α_i contains *i=1,2,3..m* prediction coefficients.

This mapping technique was analyzed and simulated using MATLAB. The results of the image compression process are shown in Figure 8. The inter-pixel redundancy was further reduced using the predictive mapping coding rather than using Huffman encoding. Thus, it is necessary in some cases to try different mapping techniques to achieve greater reduction of inter-pixel redundancy.

The following MATLAB code was used to compute the mapping of a gray level image. Note the reduction of entropy from the original amount of entropy in the original image in Figure 8.

```
>> f=imread('Aligned Matches.tif');
>> e=mat2lpc(f);
>> imshow(mat2gray(e));
>> entropy(e)
     ans = 5.9727
```

Original image = 7.3505 bits/pixel

Coded image = 5.9727 bits/pixel

This reduction of entropy allows for a more efficient encoding of the prediction error image in Figure 8 .

```
>> c=mat2huff(e);
>> cr=imratio(f,c)
     cr =   1.3311
```

The compression ratio is much better using this mapping coding technique than the Huffman encoding one.

Mapping CR=1.3311

Huffman CR=1.0821

The histogram of prediction error was obtained to analyze the amount of entropy reduced.

```
>> [h,x]=hist(e(:)*512, 512);
>> figure; bar(x, h, 'k');
```

The high peaks around 0 and small value variations on the histogram graph show a lot of interpixel redundancy been removed by using the prediction and differencing mapping process. For images when patterns are aligned and pixel values are predictable, mapping techniques are more effective on reducing redundancy than variable length coding techniques.

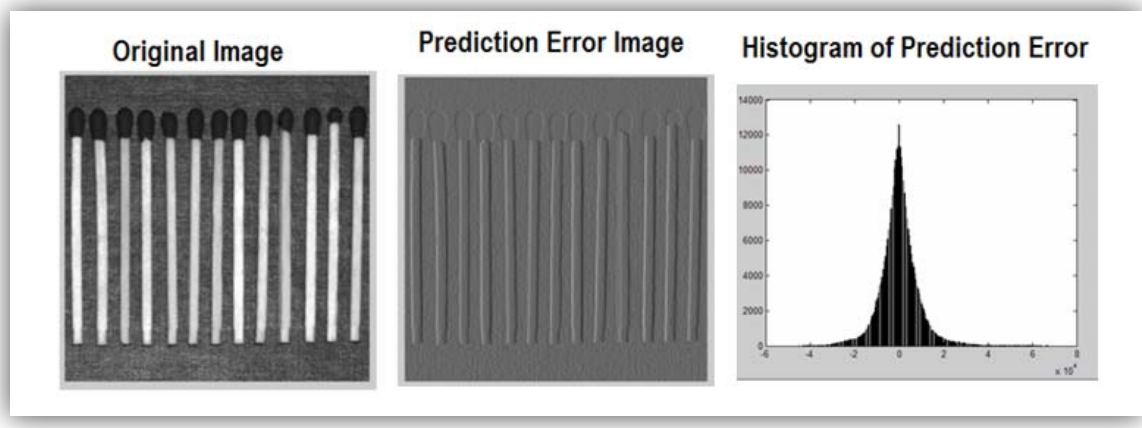

Figure 8 - Mapping Example

2.5.2 Psychovisual Redundancy

As mentioned before, image compression is a method that reduces the number of required bits to represent the information of a given image. In image compression, there two fundamental

concepts which are redundancy reduction, which aims to reduce duplicate information, and irrelevant reduction, which seeks to eliminate information not perceived by the human visual system (H.V.S)(11)(12).

In contrast with coding and interpixel redundancy, psychovisual redundancy is highly dependent on the H.V.S sensitivity. The reduction of psychovisual redundancy belongs to the irrelevant reduction concept and it is in most cases desired for the simple visual processing of an image. However, the reduction of psychovisual redundancy results in information loss due to the quantization process necessary to eliminate this type of redundancy. The reduction of psychovisually redundant data results in lossy data compression (9).

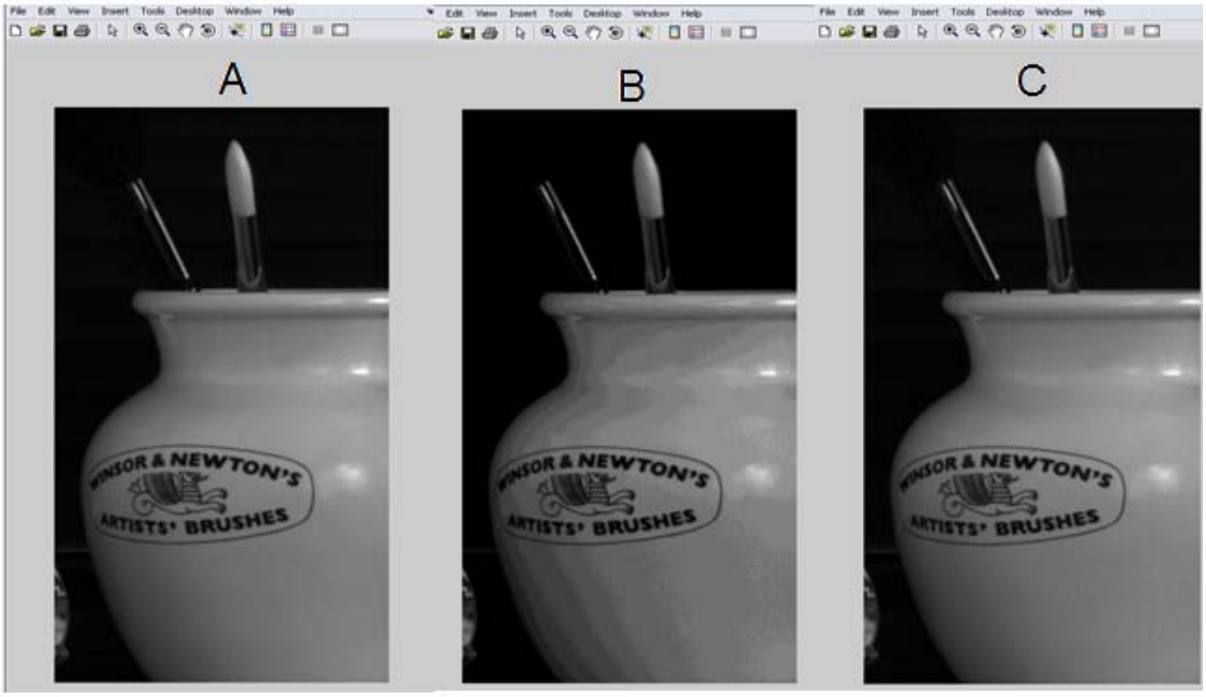

Figure 9 - Psychovisual Data – Figures A,B,C with different quantization ratios

The image A in Figure 9, is the original image to process using quantization techniques to reduce the amount of redundancy. The images B and C are the result of the psychovisual redundancy reduction of image A. Images A and B have large quantization steps. The

quantization effects in B are noticeable on the texture of the base, whereas the image C has almost no signs of being quantized at all. The image C uses a filter that allows for compensation when large quantization steps are used.

CHAPTER 3

THE JPEG2000 COMPRESSION STANDARD

JPEG2000 is the new standard for image compression and it is the successor to the former standard JPEG. The word JPEG stands for joint photographic experts group and it was first considered as a compression standard in 1992 (7)(13). Both JPEG and JPEG2000 standards were jointly developed by the international electrotechnical commission (IEC) and the international organization for standardization (ISO). The main purpose of these two standards is to achieve a good compression performance with variable compression ratios and excellent decompressions with great quality on reconstructed images (6).

As well as the former JPEG standard, the JPEG2000 standard provides several compression techniques and different modes of operation. Depending on the application, the proper technique and mode are used. Nevertheless, JPEG2000 has more modes of operations and other advantages over base-line JPEG which are mainly due to its compression algorithm based on the discrete wavelet transform (13).

3.1 How It Works

The standard JPEG2000 relies on the idea that the coefficients obtained from wavelet transformations to decorrelate pixels on an image are coded more efficiently than the values of the pixels themselves (14). The basic function of the wavelet transformation is to concentrate the pixels with the most pertinent visual information into a small number of coefficients. The rest of the coefficients are then quantized or truncated to zero if lossy compression is being performed (9).

The basic blocks involved in the compression of the image using the JPEG2000 standard

are shown in Figure 10 (15)(9).

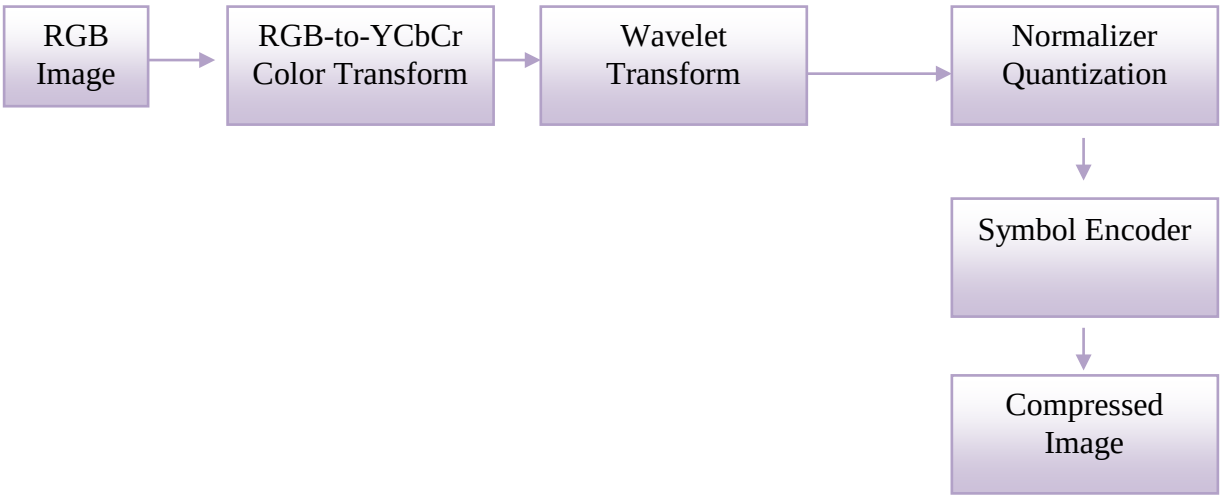

Figure 10 - Basic Blocks for JPEG2000 Compression

Before applying the wavelet transformation to an RGB image, for example, the pixels of the image are converted from the RGB domain to the YCbCr color domain. Applying a color transformation to an image is an excellent way to decorrelate the color information contained in the image. Once this is done, the wavelet transformation of the image can then be computed (9). The initial decomposition of the image after one-level wavelet transformation results in 4 sub bands. The first band is a low-resolution approximation of the image. The remaining bands are the image's horizontal, vertical, and diagonal frequency characteristics of the image, see Figure 11. The number of coefficients is then quantized for lossy compressions or encoded if lossless compression is desired. The result is a set of coefficients that contain image information regarding its frequency and space characteristics (16)(9).

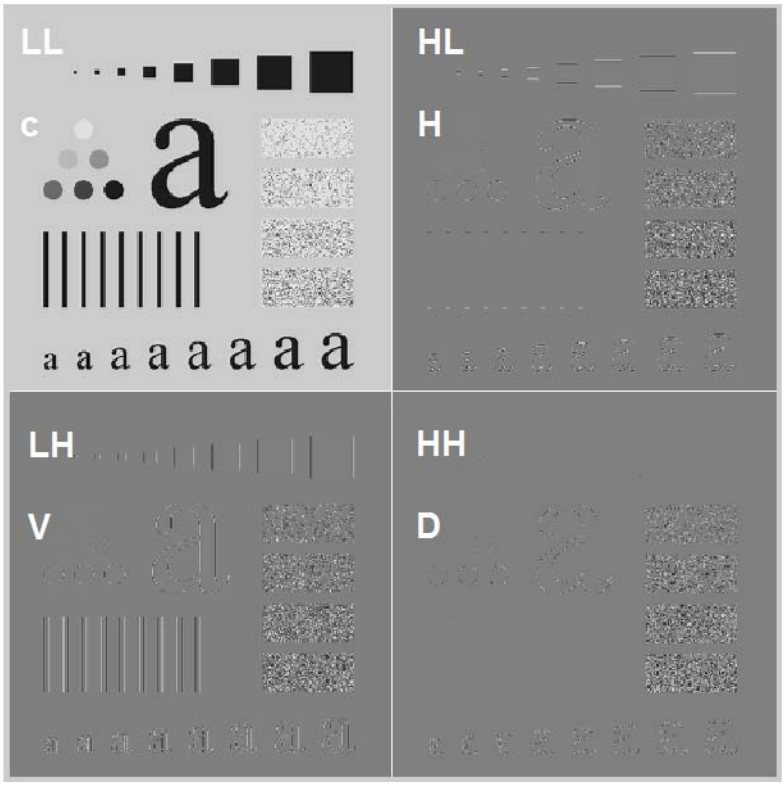

Figure 11 - Wavelet Sub Bands

The JPEG2000 decompression of an image is achieved by performing the inverse steps taken to compress it. Figure 12 shows the basic steps to decompress a JPEG2000 image (9).

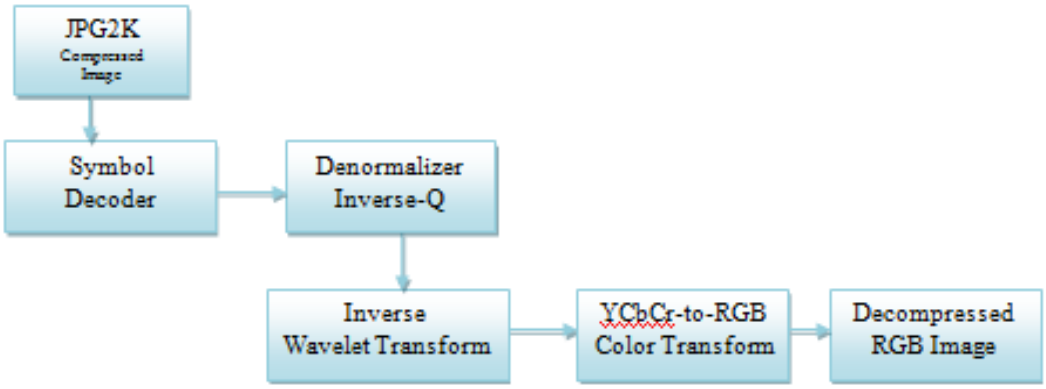

Figure 12 - Basic Blocks for JPEG2000 Decompression

Figure 13 shows the original image and the compressed one using the JPEG2000 compression simulation in MATLAB. The image decompressed by JPEG2000 decoder in Figure 12 was a grayscale image and it was one plane only. If an RGB image is to be compressed and decompressed, three planes (red, green, blue) must be processed, each plane for each color component of the image.

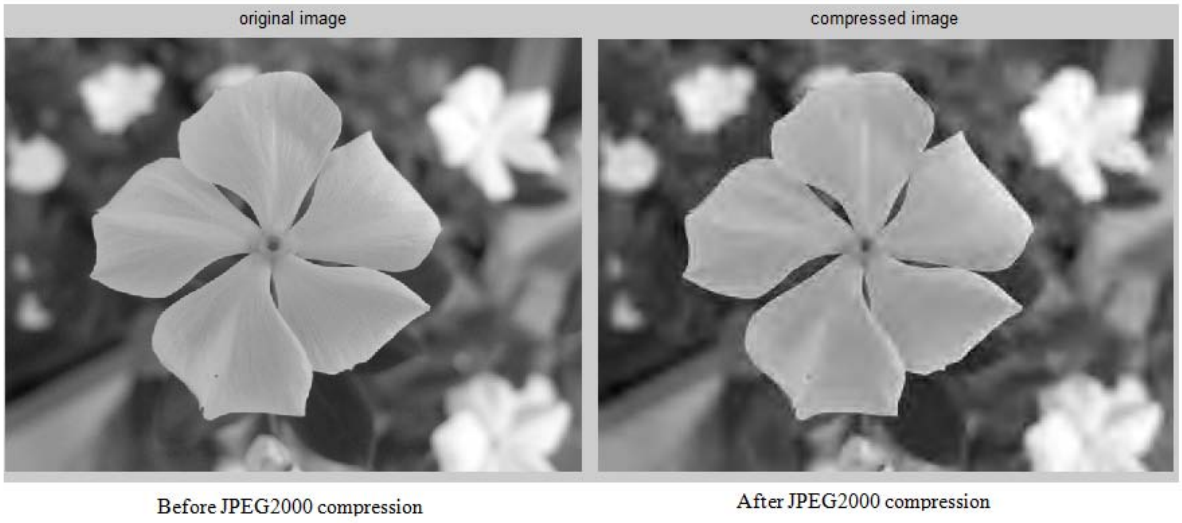

Figure 13 - - Before and After JPEG2000 Compression Images

The following MATLAB code was used to generate a 42:1 compression ratio. The JPEG2000 compression of the image was achieved using a five-scale wavelet transformation.

Table 2- MATLAB Commands to JPEG2000 Compression/Decompression

```
>> f=imread('flower.tif');          % Read image from directory
>> c1=im2jpeg2k(f, 5, [8 7.5]);     %Perform JPEG2000 compression (46:1)
>> f1=jpeg2k2im(c1);                %Perform JPEG2000 decompression
>> imshow(f1)                       %Show compressed image
>> imshow(f)                        %Show original image
```

Notice how the edges of the flower become blurred due to the high compression level. If regular JPEG had been used to compress the image, there would be “blocking” on the picture due to the DCT transform that it uses. JPEG2000 uses wavelet transformations and no

subdivisions of the image are necessary to decorrelate data, thus there is no “blocking” artifact (7)(17).

3.1.2 Analyzing JPEG2000 Compression Ratios

Using the root-mean-square error (RMSE) is one way to calculate how much difference exists between the original image and the compressed one.

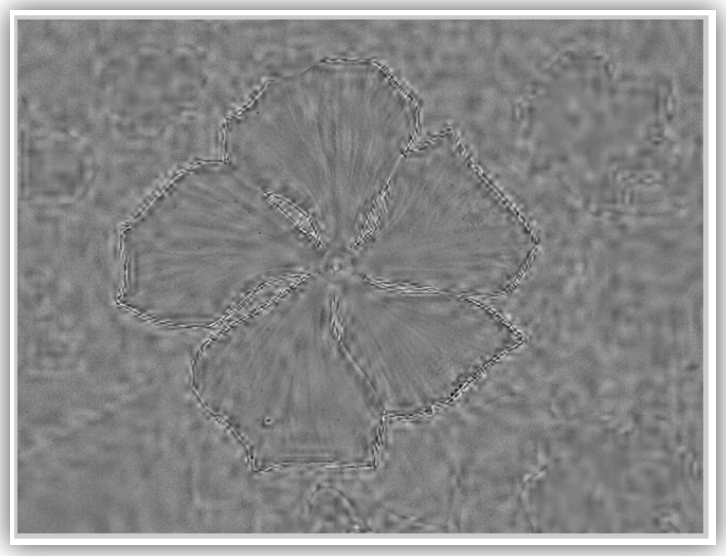

Figure 14 - Error Calculation On JPEG2000 Compressed Image

The error image in Figure 14 show the difference between the original image and the compressed one. The higher the compression ratio, the more visible the image becomes. This effect is primarily due to the quantization errors introduced to the compressed image. Quantization implies that a further process to reduce data redundancy has taken place. When using quantization, some data is lost due to the rounding of coefficients from the transformations previously taken. The data loss generates an error that can then be easily appreciated on the figures above generated from the differences between the original and compressed images.

The following histograms in Figure 15 show the amount of interpixel redundancy removed from the two compressions shown above:

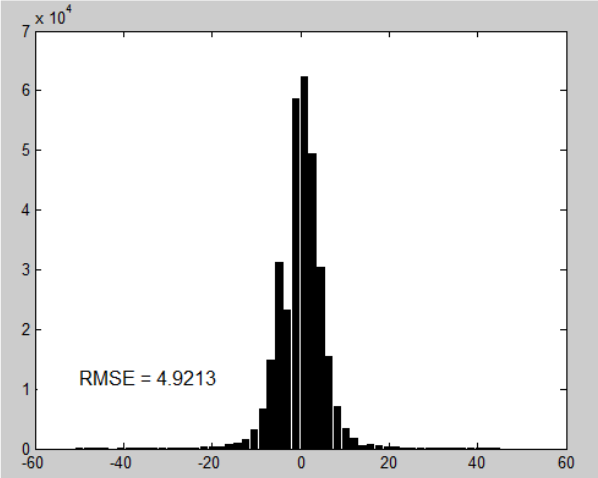

Figure 15 - Analysis Of JPEG2000 Coefficient Distribution

The high peaks around zero and small variations reflect the effects on the data redundancy removal. The sharper the peaks and less variation, the higher the amount of redundant data that has been removed.

3.2 Why JPEG2000 Over Regular JPEG

The idea behind the development of JPEG2000 is to compress an image only once and decompress it in many different ways to suit different requirements (7). JPEG2000 was developed based on the principles of the discrete wavelet transform (DWT). Regular JPEG uses a different type of transformation called discrete cosine transform (DCT) and lacks the scalable image compression features achieved by JPEG2000 wavelet transformations (6).The JPEG2000 standard compression has a few other advantages over regular JPEG which are:

- Superior compression performance

JPEG2000 has no perceptible artifacts or “blocking” when bit ratio compressions are performed. The compression gains of JPEG2000 over regular JPEG are due to the use of

sophisticated entropy encoding schemes and the use of discrete wavelet transformations. (7)

- Multiple resolution representation

During the compression process using JPEG2000, the image gets decomposed into a multiple resolution representation. This multiple resolution representation can serve for further image analysis beyond just compression. (7)(18)

- Random code stream access and processing

Also known as region of interest (ROI), random code stream access allows for storage or processing of certain parts of an image. JPEG2000 code streams can be processed separately using different quality for areas that are more relevant in an image; thus, storage space is optimized. More advantages are listed on (6)(7).

CHAPTER 4

STEGANOGRAPHY IN GENERAL

4.1 Definition

Steganography is derived from the Greek words “steganos,” which means secret or covered, and “graphy,” which means drawing or writing. The main purpose of steganography is to insert hidden information in messages (or other mediums) in such manner that no one, besides the sender and final recipient, will even realize there is information hiding in the message (11). This type of technology is quite ancient. The use of steganography can be traced back as far 440 years BC. Demeratus warned his countrymen about a forthcoming attack to Greece by writing a message on a wooden panel and concealing it by covering the panel with wax. In those days, wax tablets were commonly used for writing, so no one suspected there could be information hiding underneath the wax. Only the sender and recipient had knowledge of such hidden message in the wax tablet. Other examples of steganography used in those days were invisible ink that became visible with heat, usually milk and similar organic substances were used as ink. Another way to conceal information was to tattoo a message on the shaved head of a person and send the message only when the hair had grown back. A steganography message always appears to be hiding on elements that seem inconspicuous such as a shopping list, an ad on a newspaper or even a picture. The advantage of steganography over other techniques to covert data is that messages don’t attract any attention to themselves, the messenger or the receiver (4).

For example cryptology, a technique that scrambles data to conceal its meaning, is not as effective as steganography because the message is visible even if it is not decipherable. The fact that the message can be seen but not understood can arouse suspicion and may incriminate the

messenger as well as the receiver. In some cases a combination of cryptology and steganography is used to improve the effectiveness of hiding data from other people.

4.2 How It Works

Figure 16 shows a basic block diagram showing the steps used in steganography to hide a message.

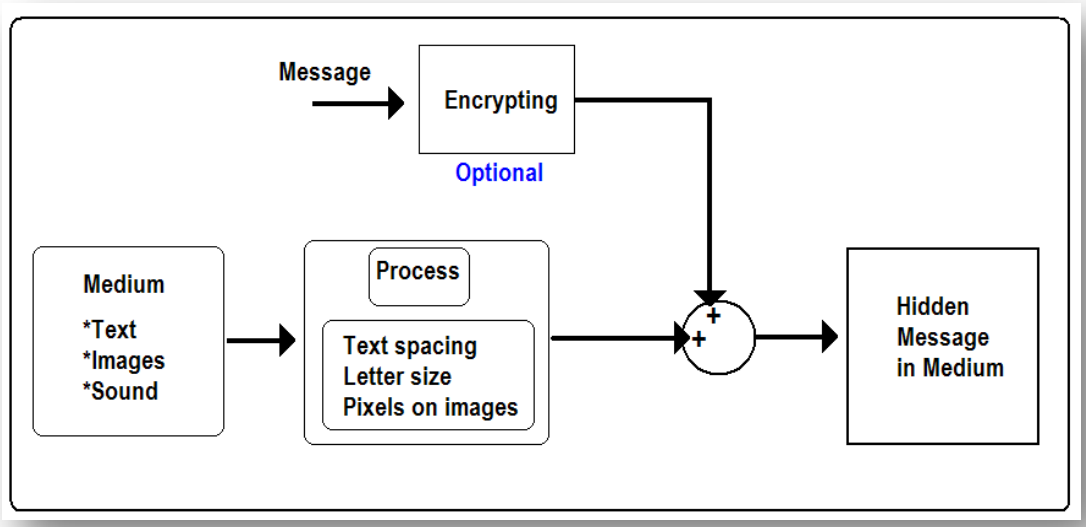

Figure 16 - Steganography Process Block Diagram

A steganography message is first encrypted using traditional means in order to create cipher-message. The encrypting of the message is optional and it is only done to further protect the message. Before the message is inserted into the medium, a careful analysis is done to decide where or how to hide the message. The characteristics of the medium, such as letter size and spacing in text or even pixel values in images, are manipulated in such a way as to effectively conceal the message to be sent (4). Only the recipient will then recover the message by uncovering it using the same technique and by applying the reverse steps shown in the Figure 16.

4.3 Applications

Today, steganography is widely used in many areas of the public or private sectors and it is often associated with high technology where the message to conceal is hidden in an electronic file rather than in paper. Whereas missing letters on a text document are easy to perceive, it is harder to find discrepancies on more complex mediums such as audio and images (4). It is very feasible to slightly change the characteristics of an electronic image or audio file without it been noticeable for the viewer or listener. In images, for example, redundant bits in pixels can be removed and still have an image that looks intact to the viewer. The same applies with audio files in which sounds above the human audible frequency can be removed and still reproduce the audio without any discrepancies to the listener. In these missing bits, a stego algorithm hides data by embedding it into an image or audio file. The bigger the file size and quality, the higher the amount of redundancy that can be used to hide data using steganography (4).

An image, for example, containing a 24-bit bitmap, 8 bits representing red, green and blue (RGB) for each pixel, will have 2^8 different values of hue for each color. The number of possible bit combinations is very big and small changes in color hues are not easily perceived by the human eye. The average human eye can’t detect a small difference in hue values such as 11111110 and 11111111. This undetectable difference in the least significant bit can be used to hide images or ASCII characters for every three pixels in a picture (4). The same principle can also be applied to gray scale pictures.

Steganography could be used to protect intellectual property in movies, pictures and audio. The movie industry, for example, loses money every year due to piracy programs that can copy DVDs and internet sites that allow individuals to illegally download movies and other related files. The appropriate use of steganography can provide a way to deter some of the

unlawful activities that hurt the intellectual work and rights of many movie makers and media companies (4).

Another media in which steganography can be used is in audio files. Figure 17 shows the basic principles to hide data in audio.

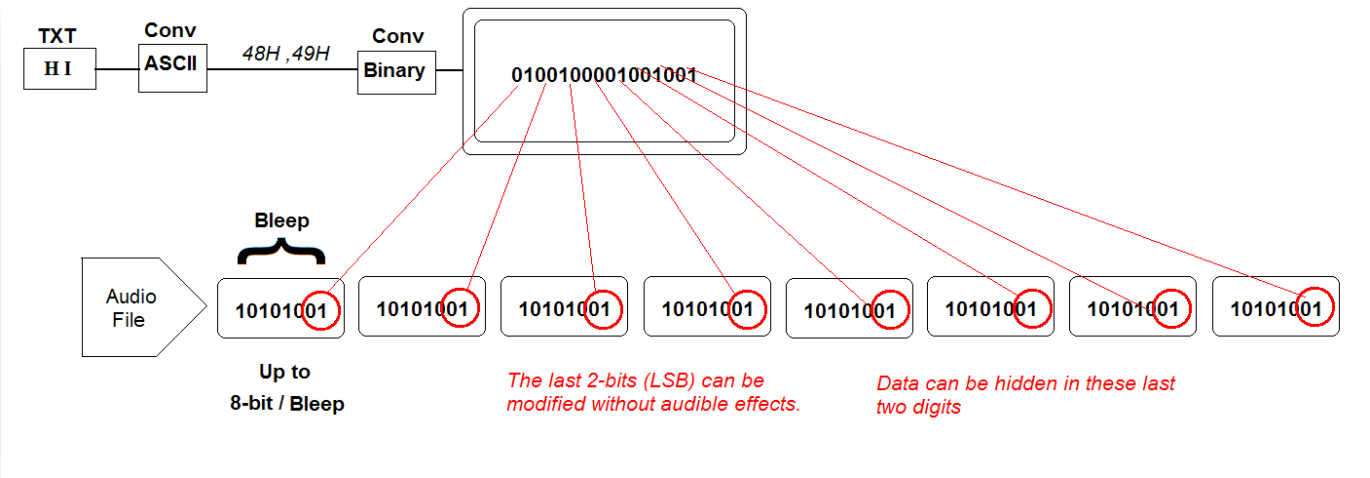

Figure 17 - Steganography In Audio

As already discussed, a stego message requires a covert file (the word “HI”) and an overt file (audio file). The covert file to be hidden (text) is converted to ASCII and then translated into binary. The word to be hidden is “HI” and the respecting values for H is 48 hex and I is 49 hex. The binary conversion of 48H is 01001000 and 49H is 01001001. By connecting the two together, the covert message is obtained. The series of bits “0100100001001001” correspond to the letters H and I (4).

The overt file in Figure 17 is an audio file. An audio file in this figure consists of a series of bleeps. A bleep is a sound of a given pitch that lasts for a less than a second. By having enough of these bleeps connected together music and sounds can be reproduced. Songs and audio files are created by connecting a bunch of individual bleeps, each containing up 8-bits. The

most-significant-bit (MSB) is the left-most bit in the bleep and it is also the most audible to the human ear, and the least-significant-bit is the right-most bit on the 8-bits string and the least audible. Modifying the left-most significant bit will have a significant an audible impact on the pitch of the bleep. However, if the modification is done in the right-most bit, which is the least-significant-bit (LSB), the effect will be minimal and have no audible impact (4).

The two least significant bits can be modified and have almost no impact, or at least not an audible one, on the final sound. This basic stego technique would require the 2 least significant bits from each audio bleep and replace it with 2 bits from the hidden message. Therefore in order to hide a 16-bit message in an audio file, 8 bleeps are necessary so that their least significant 2 bits can be overwritten with the bits of the message to be concealed.

Another possible use for steganography is to protect the security of the national borders. Steganography is so versatile that it could be used to improve homeland security by watermarking information on visas and other important documents. It is primordial to protect the U.S national borders especially when the nation is at war. With the advent of outstanding image scanning and sophisticated printing technology, important documents and personal identifications are in jeopardy of being forged with unprecedented accuracy. Hundreds of thousands of people enter the borders of the United States daily showing fake visas and IDs (2). . The flow of undocumented people crossing the U.S border and living here illegally can have a negative impact on the nation's security and commonwealth. By using steganography to hide information on documents such as visas and passports, a better control on who comes or leaves the U.S territory could be achieved. The names and dates at which people entered or left the U.S can help to keep better track and prevent illegal entrance to the nation more effectively. People with fake passports or visas can be more easily detected and stopped from recurring on this type

of criminal activity. Personal information such as phone numbers, address, and other pertinent data can be inserted into photo IDs to prevent forgeries and thefts from occurring less frequently.

4.4 Types of Steganography

Steganography techniques can be categorized in two general ways, which are by type of host file and by how the data has been hidden (4).

4.4.1 File Type

The type of file is the host or overt file in which the data is hidden. Depending on the file format the data is hidden in the file. This is due to the fact that different file formats have different characteristics that control how the data is concealed within the file. In .bmp images, for example, the data is hidden by placing the information in the least significant bits of each pixel. The choice of bits to alter varies, yet the technique used to hide the information remains the same. Therefore, knowing the host file type is useful to understand how and where the data might be hidden.

4.4.2 Methods of Hiding Data

Steganography uses three methods to conceal data: substitution, injection and generation (4).

4.4.2.1 Substitution

The substitution method replaces insignificant information in the host file with the desired covert data. In audio files, for example, several bytes are used to construct sounds that together form music. By modifying the least significant bits, the sounds are modified slightly, yet

the modification is very small and the difference cannot be perceived by the average human ear.

4.4.2.2 Injection

The injection method looks for areas in a given file that can be ignored and puts the covert data in such areas. In most audio files an end-of-file marker (EOF) can be found. This EOF helps the device playing the audio file to determine when to stop playing. When the device reaches the marker EOF, it stops playing because it knows that is has reached the end of the file. The “injection” of the data takes place after the EOF marker has been reached and the covert data has no effect on the sound of the audio file.

4.4.2.3 Generation

The generation method “generates” a completely different overt file based on the information contained in the message to be hidden. A picture can be generated based on the bits contained in the message to be hidden. For example, a patch of green is built for every bit 0 and a patch of red for every bit 1 found in the covert message. The colors forming the picture are then based solely on the bit sequence of the covert message.

4.5 Digital Watermarking

Digital watermarking is the method to be used for this thesis to insert data in images using watermarking algorithms. By the fact that digital watermarking can insert and hide data; it becomes a form of steganography. Digital watermarking is very useful protecting copyrights and proving ownership, yet it is not very effective on transmitting information. Digital watermarking can be used as a measure to protect intellectual property and ensure authenticity of important

documents such as passports and other IDs. Digital watermarking works by inserting information in small amounts throughout a given file in such a way that the file can be analyzed and the watermark removed if desired (19). Figure 18 shows how visible digital watermarking looks on an image. The watermark can either be visible or invisible. In some cases it is desired to have the watermark show on the picture as long as the watermark cannot be removed without damaging the image. This is done as a way to discourage copy right offenders from stealing the image.

Figure 18 - Visible Watermarking

The goal of invisible digital watermarking is to modify information in a file without having a noticeable impact on the quality of the image been processed. One must keep in mind that when a watermark is added to an image file, errors at the bit level are being introduced to the image. These bit level errors are fine as long as they don't cause a visible impact on the image (4).

CHAPTER 5

HARDWARE IMPLEMENTATION OF JPEG2000 WATERMARKING ENCODER

5.1 MATLAB & SIMULINK Architecture For The JPEG2000 Watermarking Encoder

A base-line JPEG2000 SIMULINK model was developed along with an additive watermarking algorithm. The process steps to develop a JPEG2000 standard compression algorithm were carefully analyzed while implementing the JPEG2000 watermark model in SIMULINK and the ALTERA DSP Builder. Figure 19 shows the basic steps that must take place in order to have an image compressed using the JPEG2000 standard. When dealing with RGB images, the first step is to reduce redundancies by transforming the red, green, and blue components of the image into another color domain called YCbCr (20). The RGB components will provide the values to perform the YCbCr conversion based on the Equation 4 (9)(21).

$$\begin{matrix} Y \\ Cb \\ Cr \end{matrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 65.481 & 128.553 & 24.966 \\ -37.797 & -74.203 & 112 \\ 112 & -93.786 & -18.214 \end{bmatrix} . \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Equation 4-Color Transformation Equation

After the color transformation has been performed, another transformation takes place called wavelet transform. A wavelet transform provides details regarding the image's spatial and frequency characteristics (22). A wavelet transform can further reduce redundancy by eliminating higher frequency components on an image. The quantization step is optional and it should not be used if lossless compression is to be achieved. The symbol encoder step is necessary to represent the most and least occurring symbols more effectively (6). Once all the steps on Figure 19 have been performed, a base-line JPEG2000 lossy compression has been achieved.

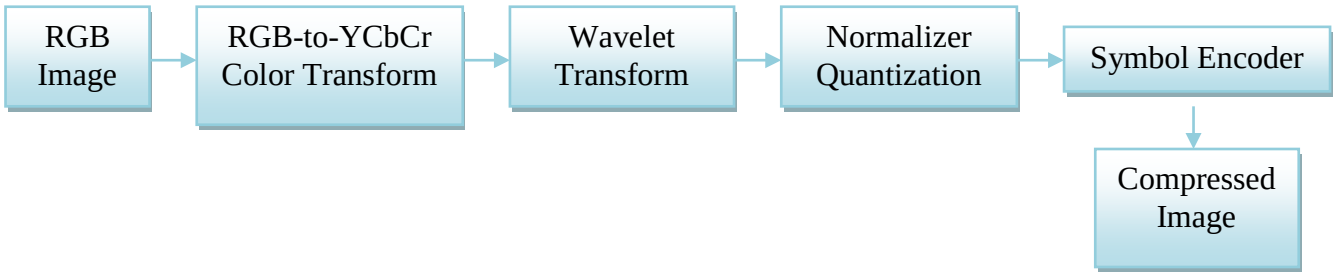

Figure 19 - JPEG2000 Basic Block Diagram

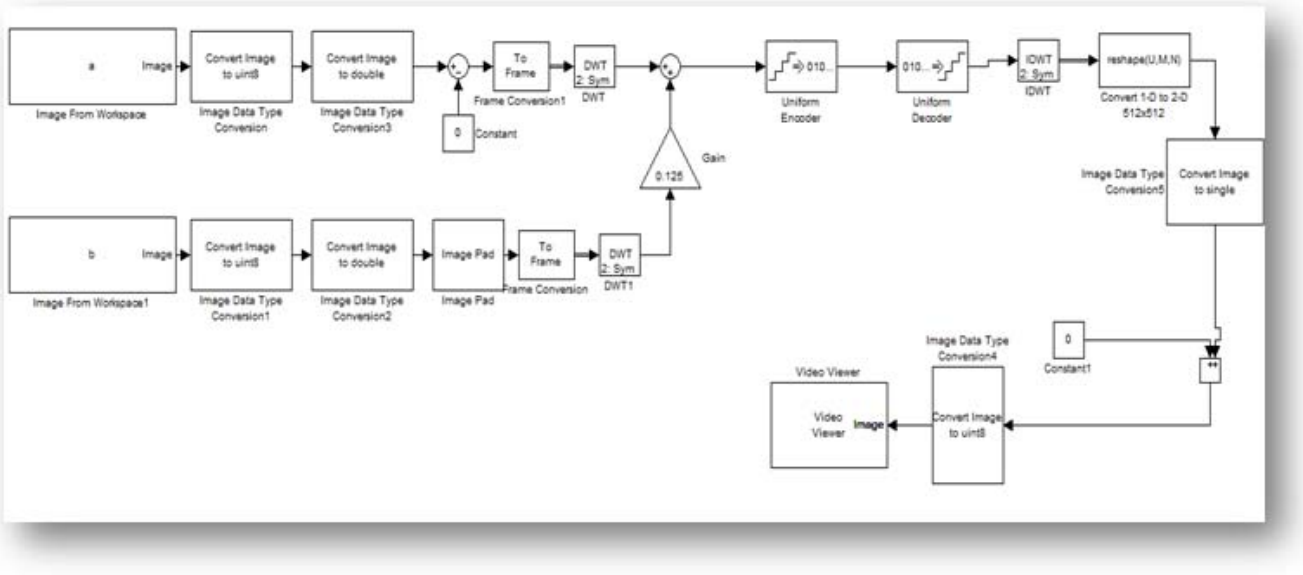

Figure 20 - JPEG2000 SIMULINK Model

The JPEG2000 watermarking architecture was done using SIMULINK. Figure 20 shows the steps involved in compressing and watermarking an M x N x 1 image using the base line JPEG2000 standard. Each component of the color transformed image is processed separately.

The first step is to convert both the watermark image and the carrier image to unsigned integer with a value range of 2^8 (0 – 255). Secondly, the image is then converted to class "double." The term "double" simply means that the values have been transformed to floating-

point numbers. This data class conversion is done because MATLAB expects to do all numerical computations using “double” quantities.

The watermark image has to be equal or smaller than the carrier image. If the watermark image is smaller than the carrier one, then image zero-padding is performed on the watermark. Once both images, watermark and carrier, are the same size, they are both decomposed into vector frames and subsequently transformed into the wavelet domain. The watermarking takes place in the wavelet domain and the strength of the watermark depends on the β value. The watermark strength β value range is from 0 to 1. The visible additive watermarking used for JPEG2000 watermarking architecture is described by the Equation 5 (15) (23).

$$Im_{wmkd} = Im + \beta \cdot Im_{WMK}$$

Equation 5 - Additive Watermarking

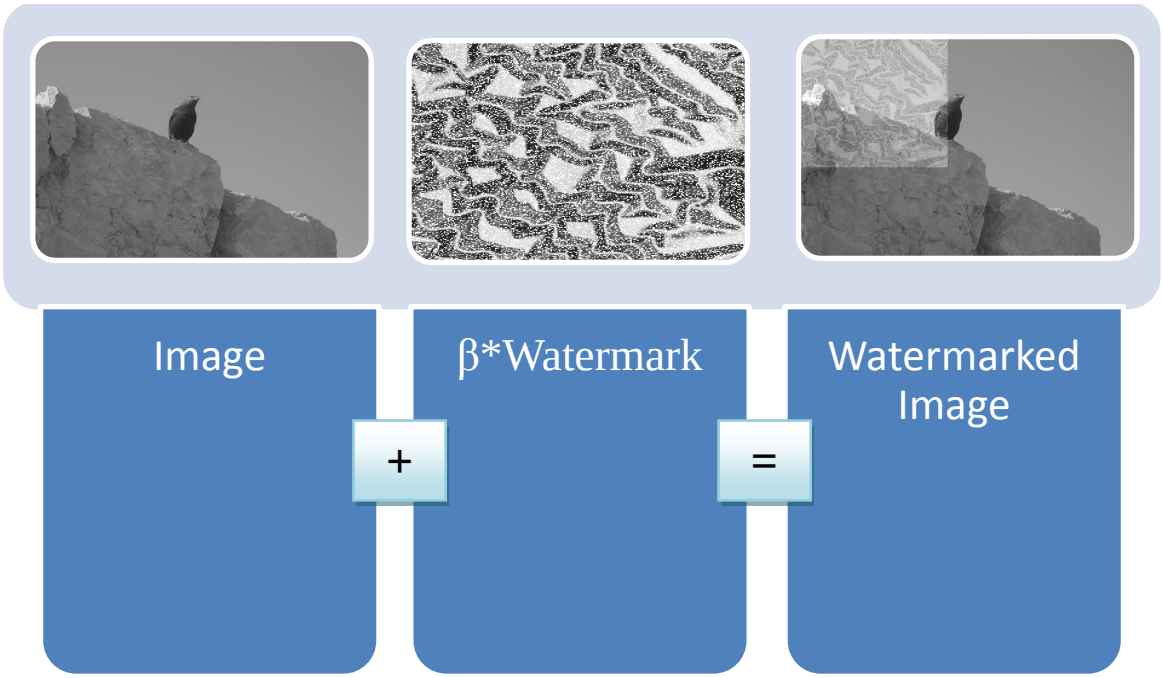

Figure 21- Representation of Equation 5

The watermarking strength β can be modified by modifying the "gain" SIMULINK block on Figure 20. After the watermark has been added to the carrier image in the wavelet domain, the wavelet coefficients are then quantized and encoded by the uniform encoder block on the SIMULINK model (24). The result is an image that has been watermarked and compressed using the base-line JPEG2000 compression standard. In order to decompress the image, the reverse process is performed. The image is decoded, and the inverse wavelet transform is applied. The image is then transformed from frame vectors into a matrix, and then converted into data class single and uint8. The final output image is shown in Figure 24 using the SIMULINK video viewer block. This process is repeated for each component of the YCbCr image (25). In other words, the image component Y undergoes the JPEG2000 watermarking algorithm, followed by the Cb image component and finally the Cr image component. Each image component is of the size M x N x 1.

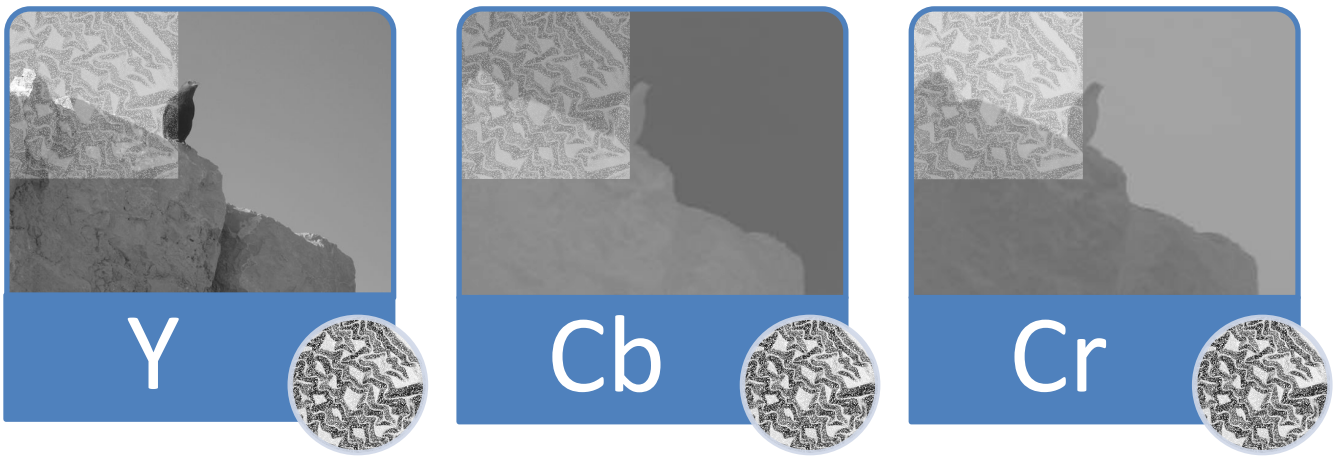

Figure 22 - Watermarked Image Color Components

After the watermark has been added to Y, Cb and Cr, the matrix for each of the three components is concatenated in order to form an M x N x 3 YCbCr image. Once the YCbCr image is being formed, the reverse color YCbCr-to-RGB color transformation takes place and

the “red,” “green” and “blue” components are input to the SIMULINK video viewer block. The steps involved in watermarking an image using the JPEG2000 standard are illustrated in Figure 23 and the resulting image is shown in Figure 24.

The first step is to separate the RGB components of the image and to input each image component (R, G, and B) to the DSP Builder block and apply the color transformation to the three components. The second and final step is to take the color transformed components and input them into the SIMULINK JPEG2000 block and process each component according to the JPEG2000 standard and output a watermarked RGB image.

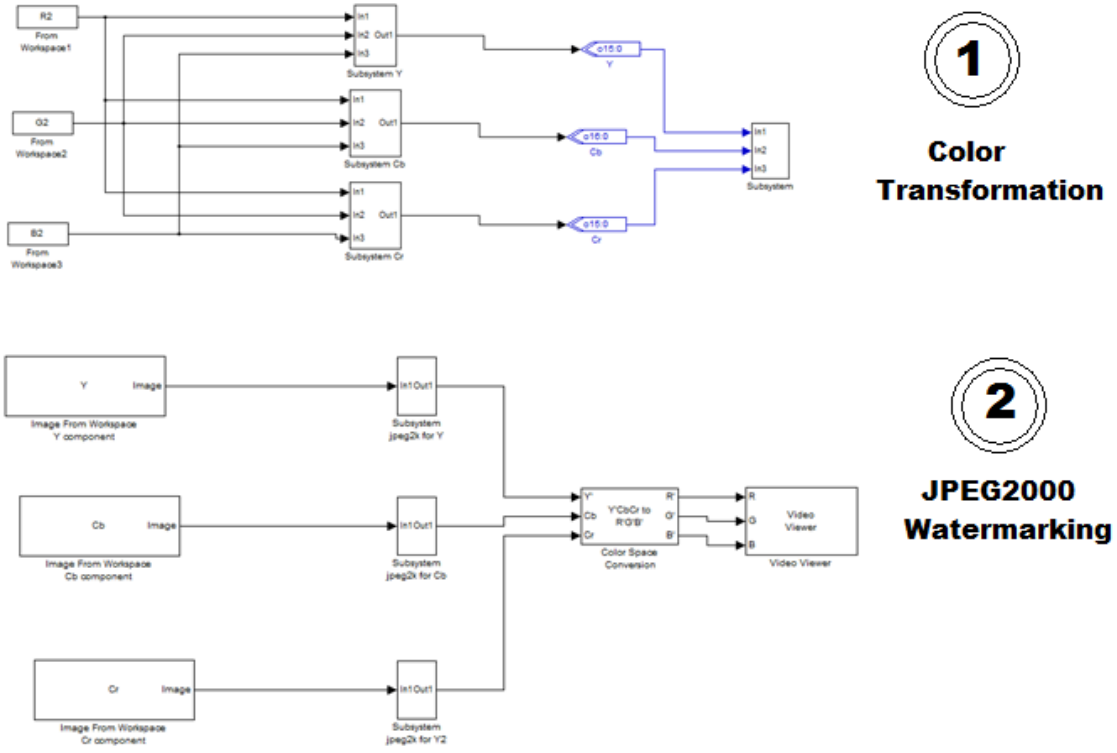

Figure 23 – DSP Builder + JPEG2000 Watermarking Architecture In SIMULINK

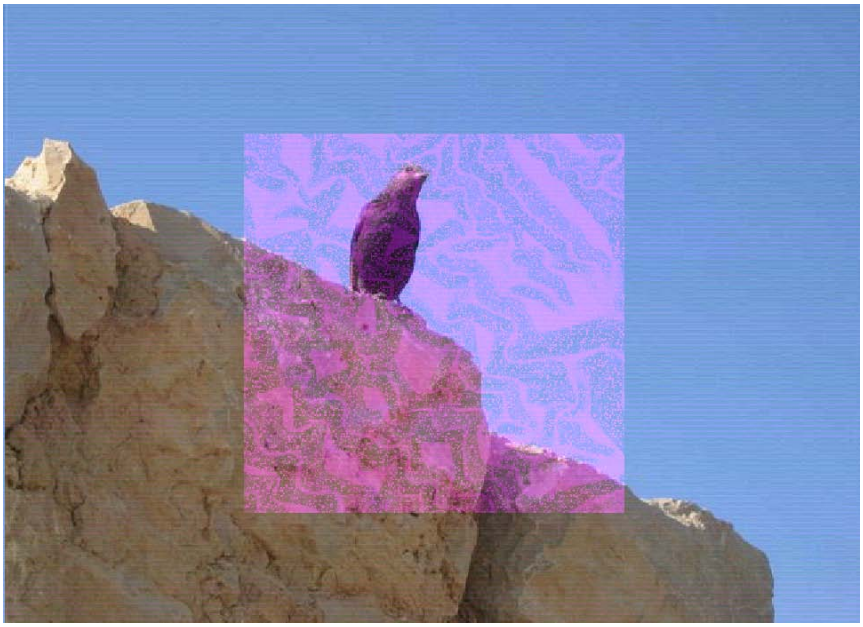

Figure 24 - SIMULINK JPEG2000 Watermarked Output Image

A good understanding of the compression and decompression principles of the JPEG2000 algorithm has been obtained using the image processing toolbox in MATLAB and other functions from the Gonzalez Image Processing book (9). Compression and error analysis techniques were based on the algorithms described in appendix A. The JPEG2000 compression and decompression algorithms were slightly modified to input and output images using the DSP builder and SIMULINK software.

5.2 Hardware Implementation

The hardware implementation for this thesis was done using a field programmable gate array (FPGA) device, SIMULINK and the ALTERA DSP Builder software. FPGAs are semiconductor devices that have built in logic blocks that are programmable. The logic blocks can be configured to operate as basic logic gates such as AND, XOR and NOT in order to build more complex logic functions. These logic blocks and nodes can be programmed by a designer to implement any logic function in order to achieve a desired task. The ALTERA DE2 Board

with the Cyclone II 2C35 microprocessor FPGA in a 672-pin package was used for the hardware prototyping of the RGB-to-YCbCr color transformation algorithm. The ALTERA DSP Builder and SIMULINK software were used for hardware integration purposes. Similar research on SIMULINK-based hybrid codesign in FPGA work has been done in Torino, Italy. For details see (26).

5.2.1 Reasons for Using FPGAs

The reasons for using the ALTERA DE2 FPGA to prototype the watermarking algorithm using the JPEG2000 standard are low cost, re-programmable capabilities and debugging flexibility. In addition to the advantages already mentioned, research shows that the ALTERA DSP Builder and DE2 FPGA provide better results than the Xilinx FPGA counterpart (27). More details on the FPGA brand and model used for the color transformation hardware prototyping can be found in appendix B.

5.3 RGB-to-YCbCr Color Transform FPGA Implementation

5.3.1 Color Transformation

The color transformation block from the JPEG2000 CODEC, see Figure 19, was implemented using the DE2 ALTERA FPGA board. In order to program the FPGA, a hardware description language (HDL) was necessary. The most common HDLs are VHDL and Verilog. These languages generate a “netlist” that can be fitted to the actual FPGA and map the path to create the desired logic to execute.

The VHDL code for the RGB-to-YCbCr transformation was generated using the ALTERA DSP Builder, MATLAB and SIMULINK software.

5.3.2 Hardware-In The-Loop (HIL) Using The ALTERA DSP Builder And SIMULINK

The hardware-in-the–loop configuration is often used to implement architectures in hardware more rapidly by converting SIMULINK files into VHDL files using the ALTERA DSP Builder (16). In some instances, it is less cumbersome to use HIL and let a computer handle memory allocation, image reading and writing setups than to configure an FPGA manually to read and write images in memory and subsequently process an algorithm. Practicality is one of the reasons the HIL configuration was used in this thesis. The other reason for using HIL is the great troubleshooting capabilities and fast VHDL code generation for SIMULINK architectures to be implemented. Figure 25 shows the block diagram of the process involved in transforming an RGB image into a YCbCr image using the HIL configuration.

The results of the color transformation algorithm implemented in hardware were compared with the “rgb2ycbcr” MATLAB command. The root-mean-square error was less than 3, which is a satisfactory result. The steps involved in the HIL algorithm flow are explained the following way:

The MATLAB block reads the RGB image and converts it into unsigned integers of rage 2^8 (uint8 class). After this conversion has taken place, the RGB image has values in each plane that are unsigned integers that range from 0-to-255. Once the image is converted into data class uint8, the planes red, green and blue are separated and converted into structures so that they can be transferred to SIMULINK. This process is performed using the MATLAB code “ed_in2_script.m” described in appendix A.

The SIMULINK block receives the structured RGB planes from the image and processes the structured values from the image based on the timing and solver parameters given to SIMULINK.

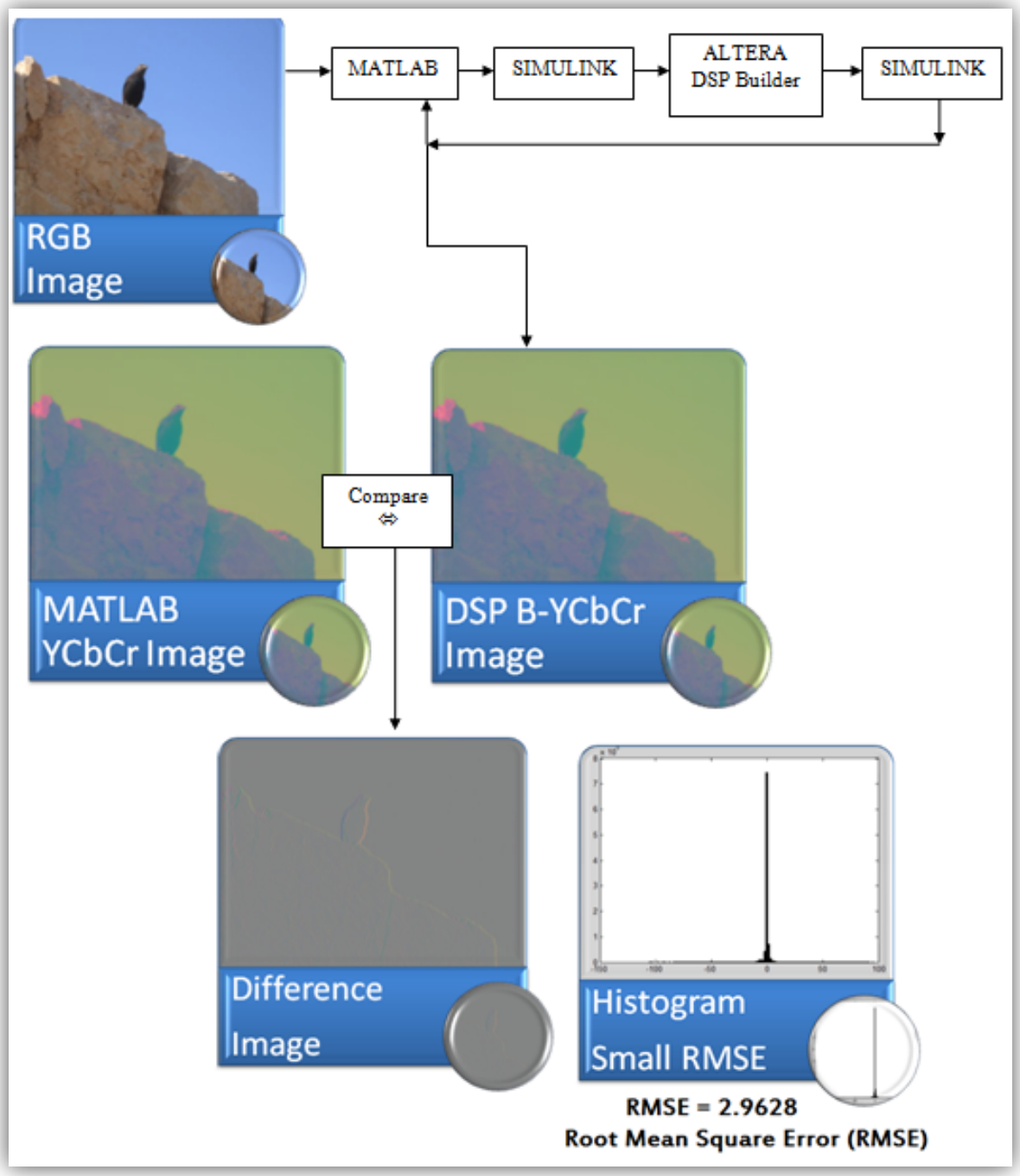

Figure 25 - Color Transformation HIL Algorithm Flow

Once the simulation has finished successfully, the appropriate DSP Builder blocks are included and simulated to verify that both SIMULINK and DSP Builder architectures provide

exact results. The DSP Builder color transformation blocks used for the HIL hardware prototyping can be seen in Figure 26.

The SIMULINK block at the output receives the color-transformed values and stores the values in the variables “simout, simout1, simout2,” and sends them to MATLAB.

The MATLAB block uses the code “ed_out_2script.m” to process the color-transformed values. The Y, Cb, and Cr components of the input image are converted from structures to matrices. Each matrix represents a color component. Once Y, Cb and Cr have been transformed into matrices, MATLAB concatenates them into one single image and compares the DSP Builder results with the results obtained from the rgb2ycbcr MATLAB command. An error image and a histogram are generated to provide a visual representation of the analysis of results.

5.3.3 DSP Builder VHDL Generation For SIMULINK Color Transformation Structure

The SIMULINK blocks R2, G2, and B2 in Figure 26 are sources that output structures generated in MATLAB by the “ed_in2_script.m” code. Each block has the corresponding values for each color component of the image.

The inputs in1, in2, and in3 go to a set of DSP Builder multipliers and adders. These blocks implement the color transformation for Y, Cb, or Cr. The details for each subsystem can be seen in Figure 27, Figure 28 and Figure 29. Notice how each input terminal has a SIMULINK-to-DSP Builder converter block. The blocks in Figure 48 convert SIMULINK values into bits. The number of input/output bits and logic involved in the image color processing can be modified at will.

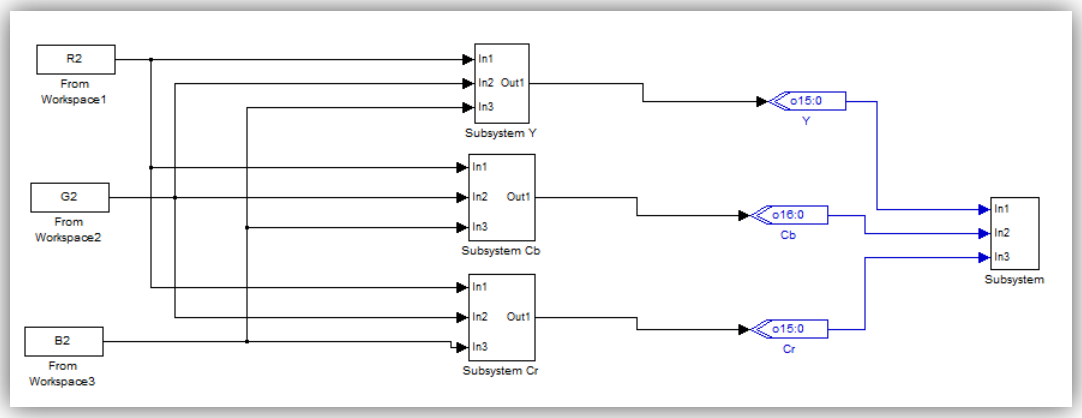

Figure 26 - DSP Builder RGB-to-YCbCR Blocks

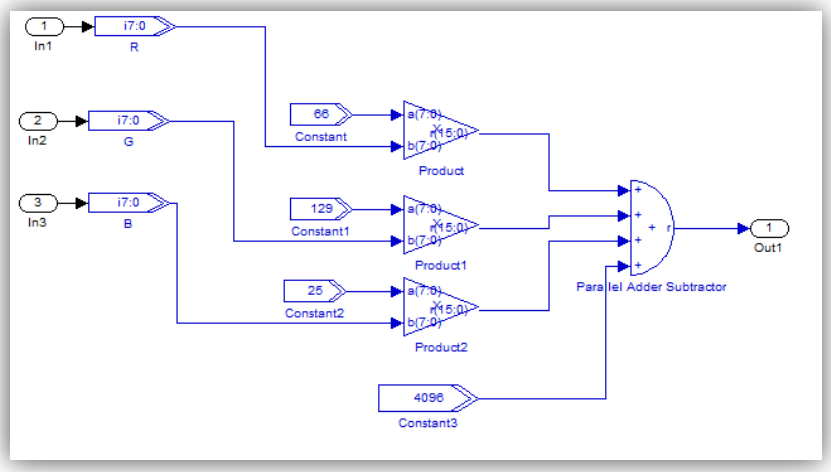

Figure 27 - DSP Builder Blocks For Subsystem Y

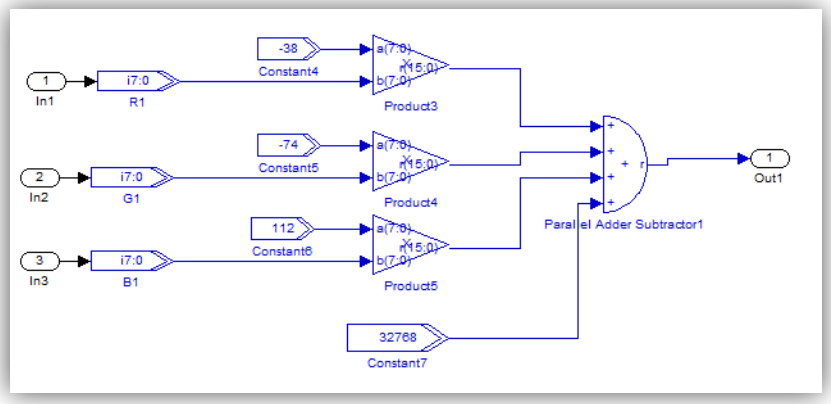

Figure 28 -DSP Builder Blocks For Subsystem Cb

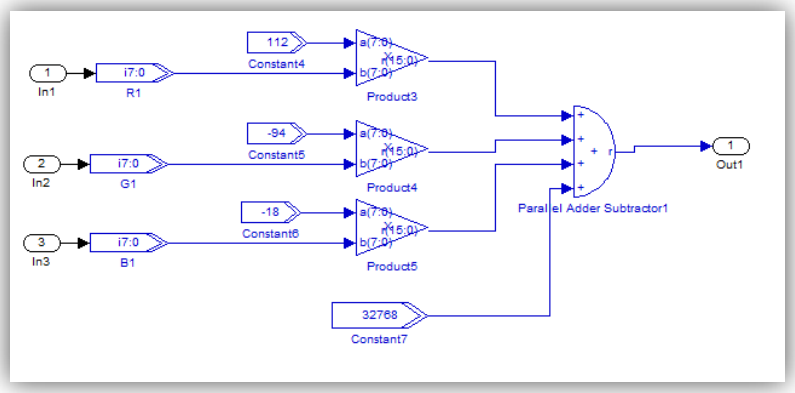

Figure 29- DSP Builder Blocks For Subsystem Cr

5.3.4 Hardware-In the-Loop (HIL) Set Up Process

After the DSP Builder blocks have been simulated successfully, the first step is to insert the DSP "signal compiler" block located in the ALTERA block set library in SIMULINK (28). Once the signal compiler block has been inserted, the FPGA chip family and device is specified by double-clicking on the compiler block. See Figure 30.

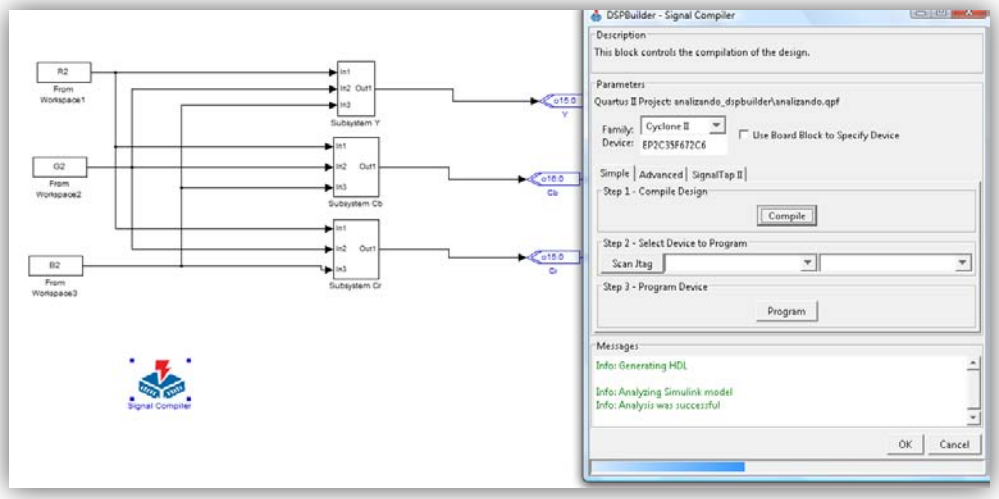

Figure 30 - HIL Setup => Step #1

To finalize the first step, one should click on the compile button to simulate the SIMULINK and generate the HDL for the DSP Builder blocks. One should also click on the "advanced" tab and click on the 'generate QuartusII project button."

The second step on setting up the HIL for the file is to create a QuartusII project and to synthesize the VHDL code generated in step 1. Figure 31 shows the signal compiler settings window and the “synthesis” button. By pressing this button the QuartusII project and VHDL generated in step 1 are synthesized and everything is ready for step 3.

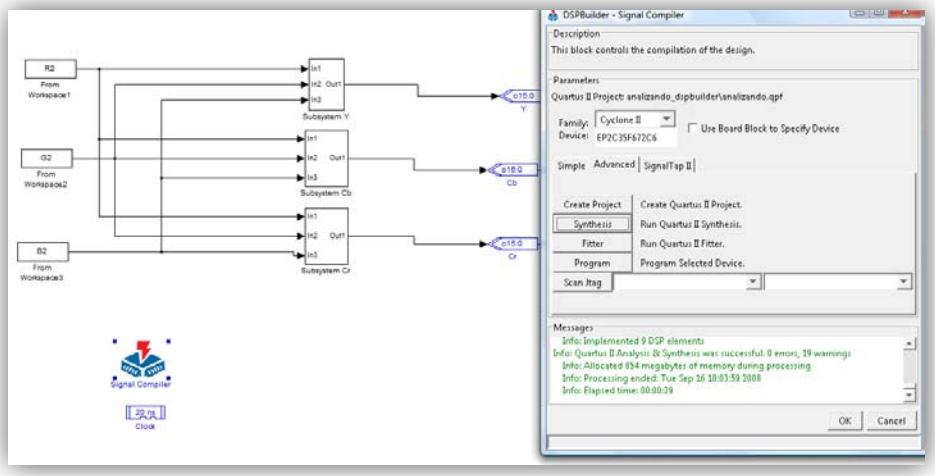

Figure 31- HIL Setup => Step #2 =>Run Synthesis

The next step after running the QuartusII project and VHDL synthesis is to add a “clock” block from the ALTERA block set and use the fitter button to run the QuartusII assembler. Notice that on the lower left “messages” window, the status of the QuartusII project is shown. See Figure 32.

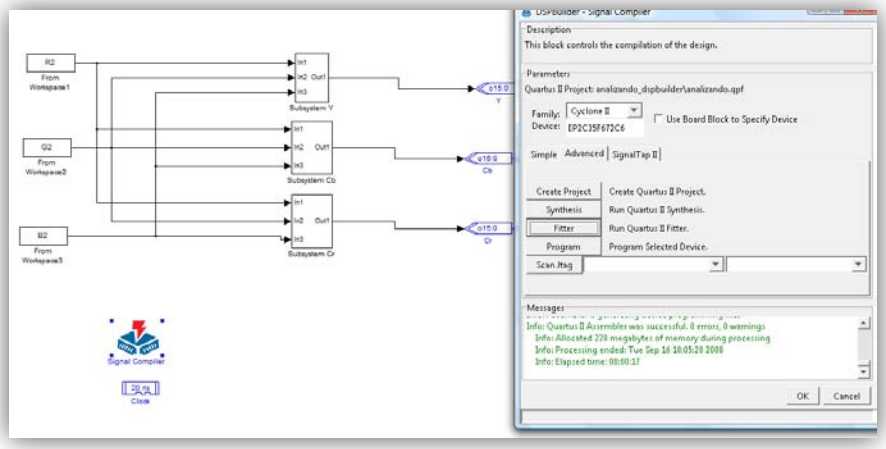

Figure 32-HIL Setup => Step #3 =>Run Quartus Fitter

After the synthesis and assembler have run successfully, the fourth step is to open a new SIMULINK file and insert an HIL DSP Builder Block like the one shown in Figure 33. By double-clicking on the HIL block, one can select the number of inputs and outputs, clock pin, and resolution . Notice that on the left-upper window in Figure 33 there is a little section that says “select the QuartusII Project.” It is in this little window where the QuartusII project generated in step 1 is selected.

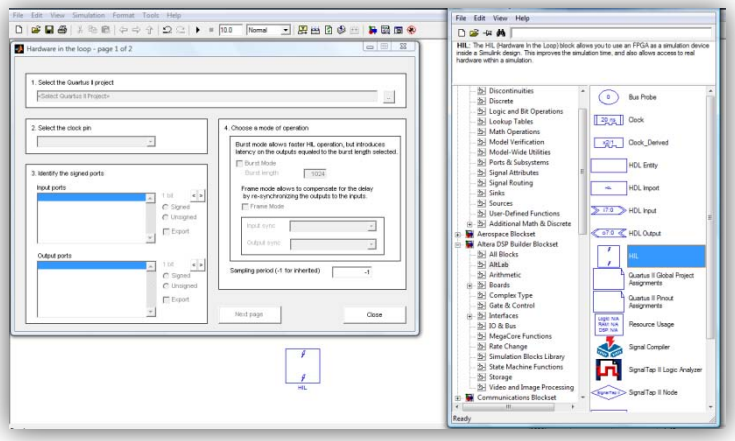

Figure 33-HIL Setup => Step #4 =>Create New SIMULINK Project + Insert HIL Block

Once the QuartusII project has been selected and the inputs and outputs have been assigned, for step 5 one should click on the “next page” button to compile the FPGA device and configure it using QuartusI (28)I. See Figure 34.

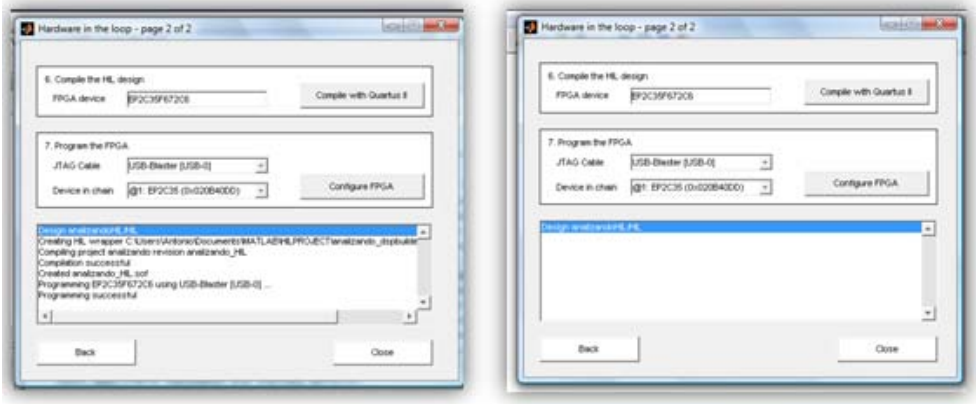

Figure 34-HIL Setup => Step #5 => Compile + Configure FPGA

After completing step five, the final result is a block with the necessary logic, inputs and outputs to operate as the DSP Builder Blocks. See Figure 35. The final step is to connect all the input and output terminals appropriately and input the right signal sources and sinks.

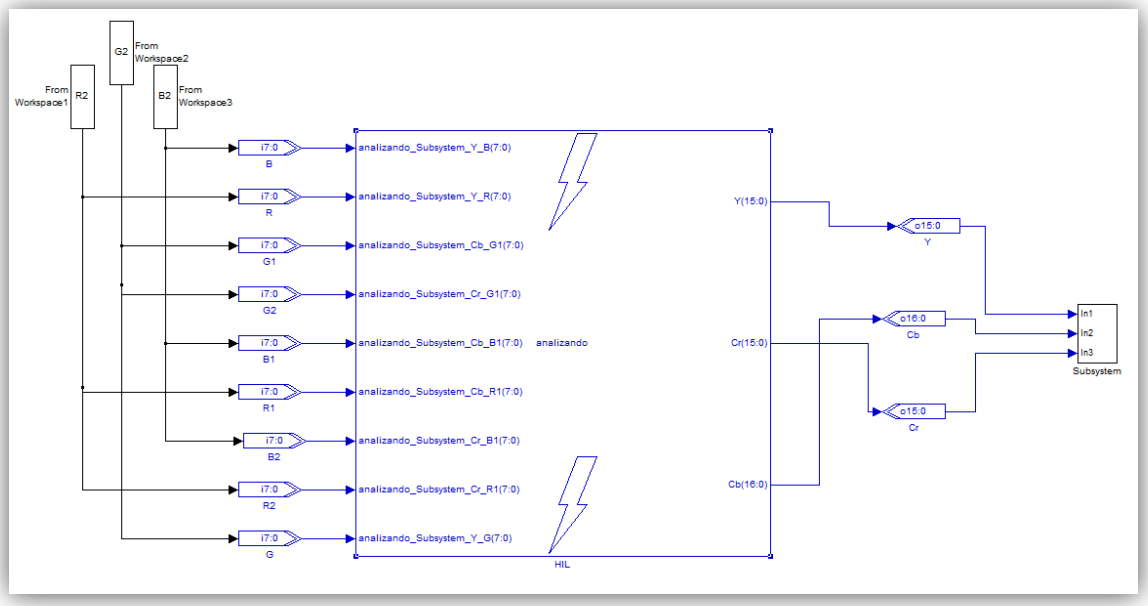

Figure 35-HIL Setup => Final Step

The simulation time for this structure is determined by the number of columns times the number of rows of single plane image plus 1. In other words $T_{simulation}$= #Columns x #Rows + 1 (29). The results from the HIL color transformation can be seen in Figure 25.

A vector waveform file was generated in QuartusII to test part of the VHDL code needed to perform the RGB-to-YCbCr color transformation. Figure 36 and Figure 37 show the simulation results of the vector file and the RTL view of the VHDL code. One of the advantages of using the ALTERA DSP Builder HIL is that MATLAB does all the steps necessary to read and write an image in the FPGA automatically. No extra VHDL code is necessary to program the FPGA to allocate memory to store the input image; as a result, more time is spent on the analysis of the algorithm to implement than on the FPGA set up.

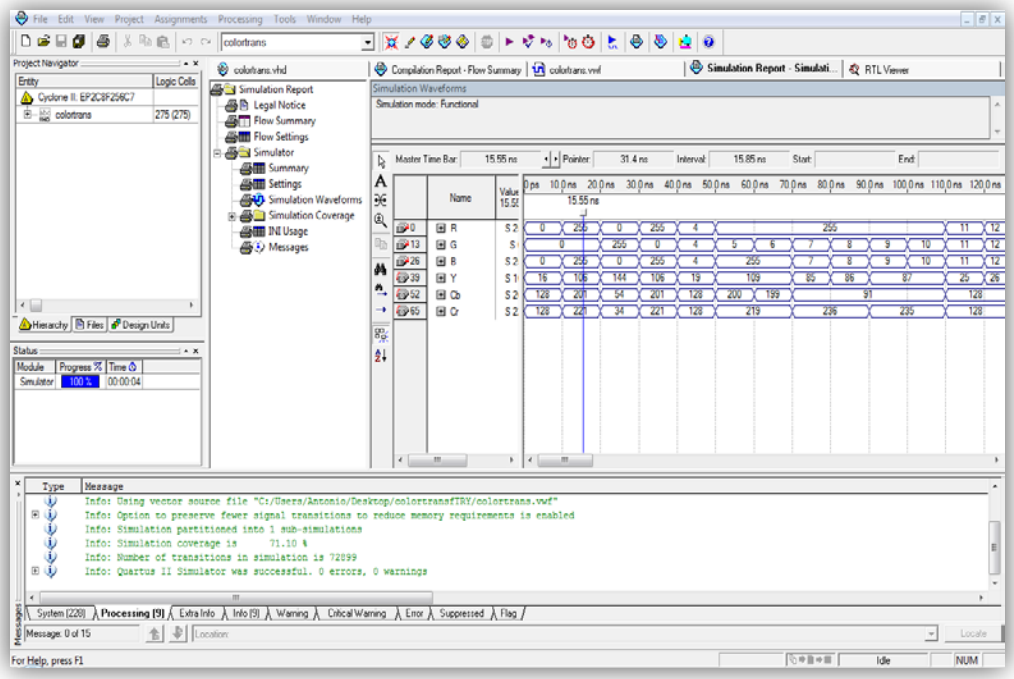

Figure 36 - VHDL Synthesis & Vector File Simulation

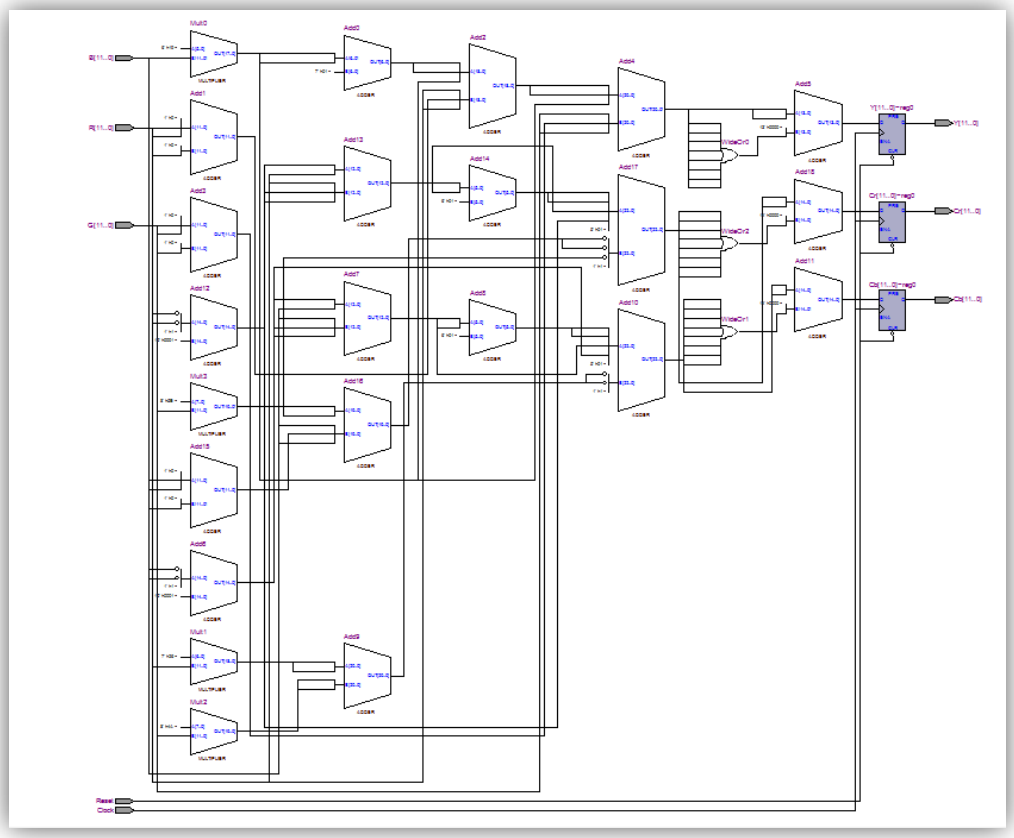

Figure 37 – Color Transform Algorithm RTL View

5.4 MATLAB, SIMULINK And DSP Builder JPEG2000 Watermarking Process Flow

5.4.1 Color Transformation

The color transformation takes place in the DSP Builder HIL block. The red, green and blie color components of the input image are separated and structured in MATLAB and input into SIMULINK to be processed by DSP Builder blocks.

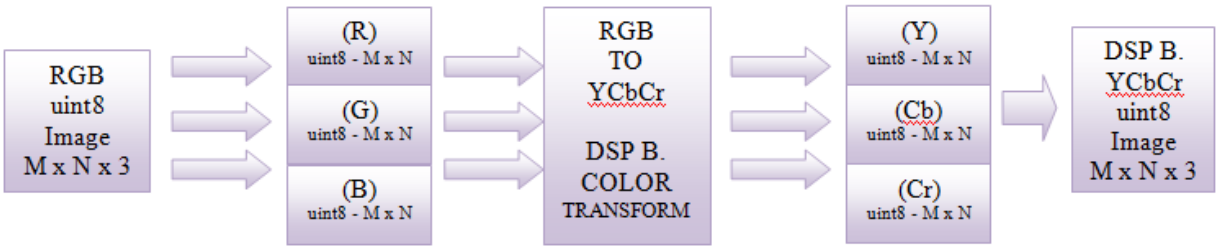

Figure 38-Color Transformation Block Diagram

5.4.2 Wavelet Domain Watermarking

The wavelet transformation, quantization and encoding of the input image takes place in SIMULINK. The DSP Builder color-transformed image is input into the SIMULINK JPEG2000 structure in Figure 20. The steps on watermarking an image in the wavelet domain are shown in Figure 39. The encoding, quantization and encoding steps are illustrated in the block diagram in Figure 40.

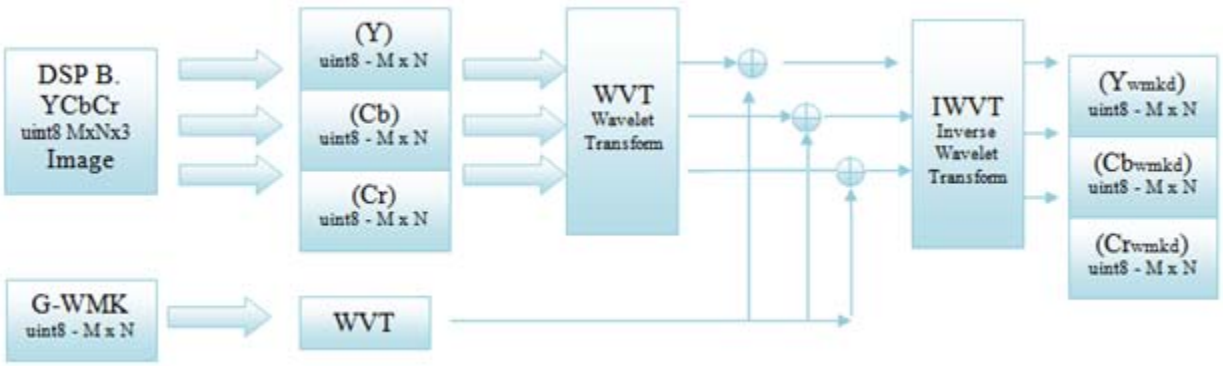

Figure 39 - Wavelet Domain Watermarking

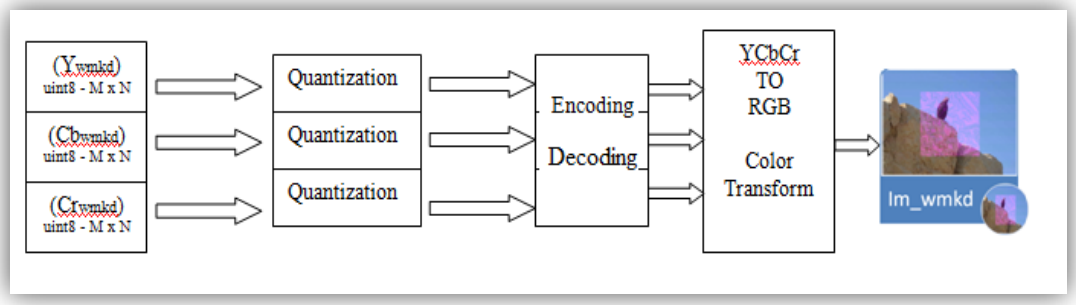

Figure 40 - Remaining Steps After Wavelet Watermarking

The block diagram shown in Figure 41 shows the SIMULINK blocks that watermarks and RGB image using the JPEG2000 baseline standard. Notice how the color components of the output image get transformed from YCbCr to RGB for the final representation of the image.

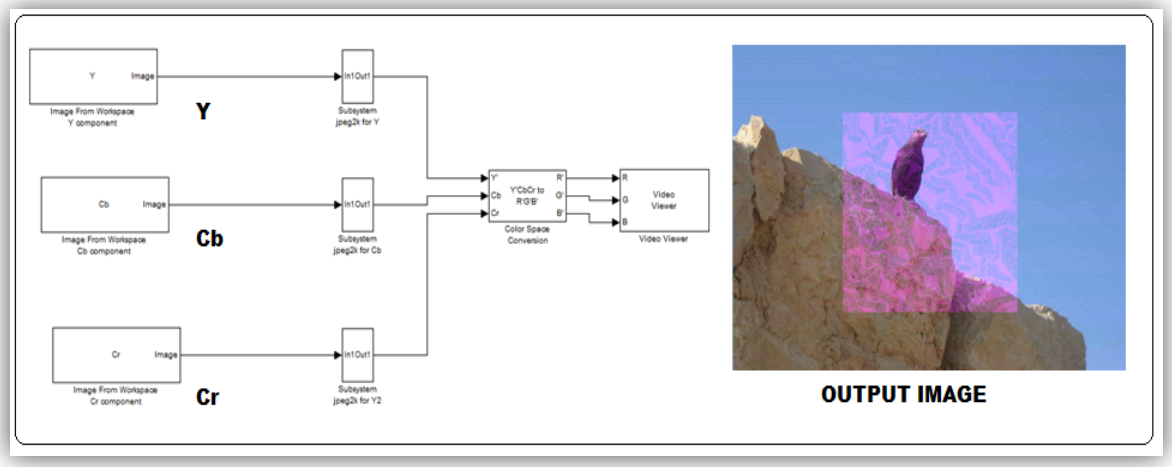

Figure 41 – Complete JPEG2000 Watermarking Process Flow

5.5 The ALTERA DSP Builder and Wavelet Transforms

5.5.1 Wavelets

Wavelets were first used in the area of seismology as tools to analyze and describe the turbulence produced by pulses generated by earthquakes. It was only after 1982, that the

scientists Morlet and Grossman worked on extending the wavelet mathematical theory to describe in terms of translation (movement/position) and scaling (magnitude) any random signal. However, it was Mallat who, in 1989, set forth the theory of signal analysis using the principle of multi-resolution decomposition of signals in time-scale space using wavelets(6)(30) Nowadays a wavelet is considered a mathematical tool used by many professionals in the area of signal processing to analyze the frequency and spatial characteristics of a given signal. Unlike the Fourier transform, there are more than two equations to describe a wavelet transform.(9)

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right)$$

Equation 6 - Wavelet Transform

Wavelets are generated by a single function called mother wavelet (ψ). The mother wavelet is a function of translations, shifts generated by b, and dilation, scaling caused by a, in the time and frequency domain. Figure 42 shows a mother wavelet without any translation or dilation.

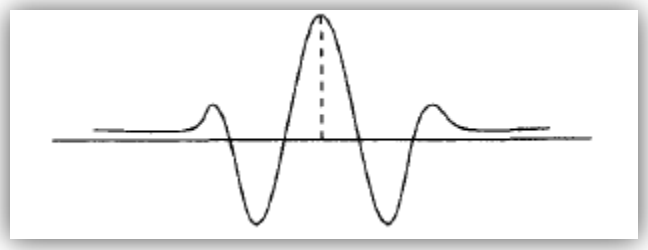

Figure 42- Mother Wavelet

The effects of varying b and a in Equation 6 can be appreciated on Figure 43 and Figure 44.

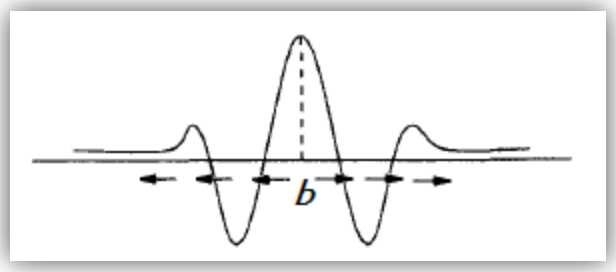

Figure 43- Translation By "b"

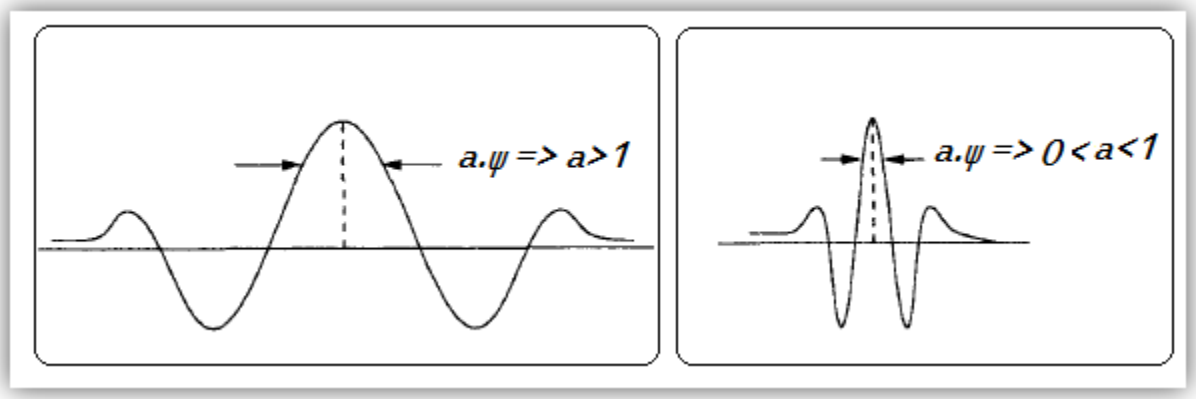

Figure 44- Dilation By "a"

Wavelets allow for frequency and time analysis of a signal because wavelets concentrate their energy in time while still preserving their periodic wavelike characteristics. This special feature of wavelets allow for better analysis of non-stationary signals, which are statistically unpredictable on regions where discontinuities exists. An example of such a region can be found on the edges of most digital images (6)(31). Another advantage of using wavelet transforms is that it provides a multi-resolution analysis of an image when wavelets are used for image compression and coding algorithms (32).

5.5.2 Reasons for Using the Discrete Wavelet Transform

Because most input signals to be analyzed by the JPEG2000 watermarking algorithm are

digital images, it is necessary to go from the continuous domain to the discrete domain. The definition of a discrete wavelet transform into is very similar to the definition of the continuous wavelet transform; the only difference between them lies on the discrete values for “a” and “b.”

Equation 7 - Discrete Wavelet Transform

$$\psi_{m,n}(t) = a_0^{\frac{-m}{2}} \psi(a_0^{-m} t - n b_0)$$

$$\text{Where : } a = a_0^m \quad b = n b_0 a_0^m$$

The discrete wavelet transform can be used to improve compression algorithms thanks to its ability to reduce redundancy. High to medium quality digital images contain large amounts of redundant information such as spectral and spatial data. The redundant information contained in digital images can be further reduced by implementing wavelet transformations to decorrelate the image information. (33)

There are many ways to implement a wavelet transform depending on the application and speed of operation(34). Some researches use the lifting scheme due to its fast operation whereas in some cases filter banks are used to minimize hardware complexity(35)(36). For details on research done in VHDL lifting wavelet scheme implementation using SIMULINK and the DSP Builder software see (37). The type of wavelet also depends on the coefficients for the filters and the mathematical characteristics of the wave, such as Haar, Daubechies, Symlets and discrete Meyer wavelets(9).

The wavelet transform analyzed on this thesis is a Daubechies 4 and it is implemented using two four-channel FIR filter banks based on the filter banks described in (9)(38)(33). The first filter bank, Figure 45, is a “decomposition” filter bank and it separates an input signal in its high and low frequency components. The first set of filters is a low pass and a high pass filter in parallel followed by two more low-pass and high-pass filters in parallel as well. The second filter

bank in Figure 46 is a synthesis filter bank. The synthesis filter bank does the inverse wavelet transformation by reconstructing the original signal from the coefficients outputted by the decomposition filter .

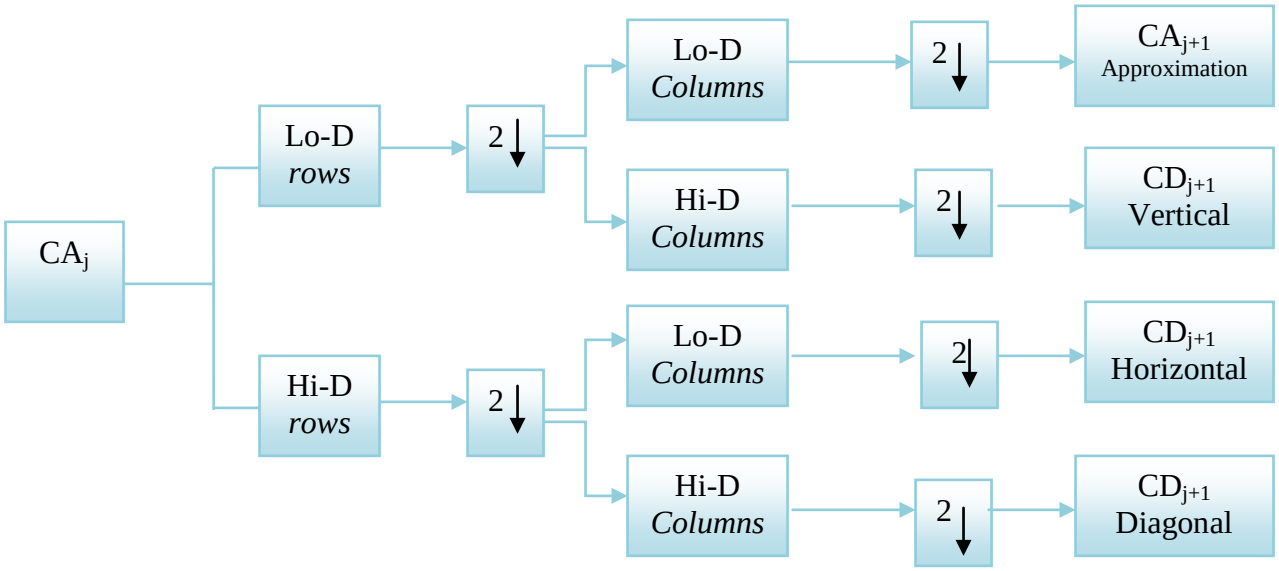

Figure 45 - Decomposition Filter Bank

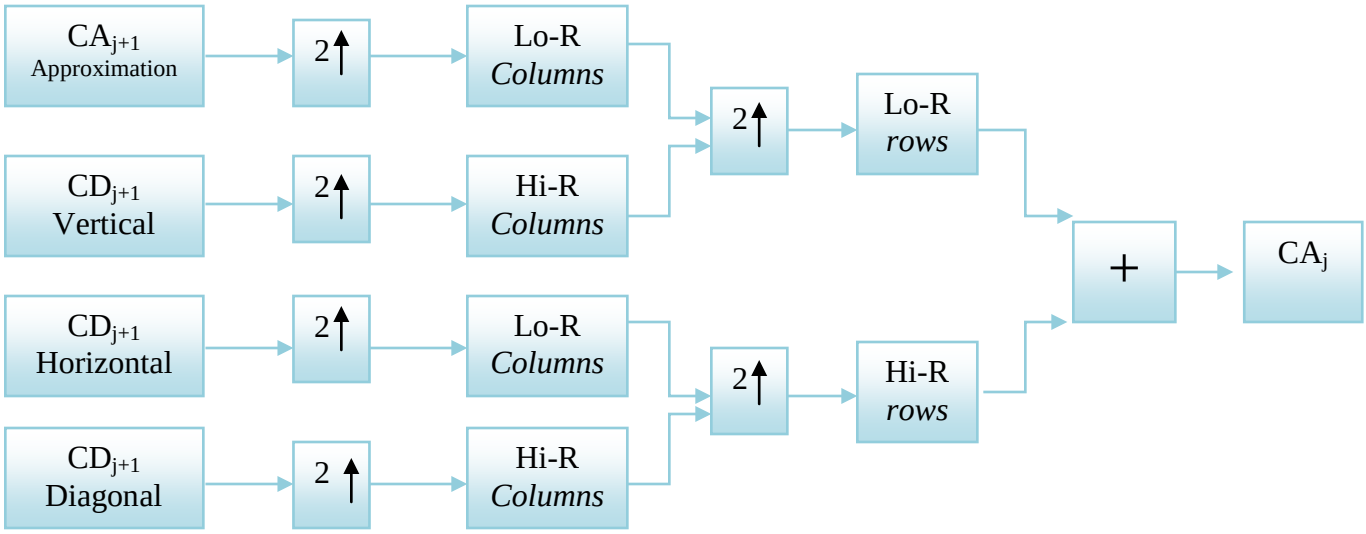

Figure 46 - Synthesis Filter Bank

5.5.3 -Digital Images and Discrete Wavelet Transforms

A digital image is usually represented as a two-dimensional array of coefficients. Each coefficient of that array represents the color and brightness level at a given point of the image. When analyzing the graphical characteristics of an image, it is hard to determine what coefficients are the most or least significant on the quality of the image by just looking at the coefficients values. Nearly all digital images have small color variations on smooth regions. The fine details on images are found on the sharp edges among the borders of the smooth regions. The sharp edges and fine details are the high frequency components of a given image, whereas the smooth color variations are the low frequency components. (33)

An image consists in its majority of smooth regions with small color variations (low frequency components) and not so much of high frequency components, which only provide the fine details that refine the quality of the image. Therefore, major consideration must be placed on the low frequency components than the details of an image when processing a digital image. When performing a wavelet transform, a signal is split into detailed and approximation coefficients. The approximation coefficients represent the low-frequency components of the signal. The high-frequency components are the detailed coefficients.(5)

The number of coefficients obtained after a wavelet transform has been performed on a signal is usually twice as much as the original signal coefficients. As a result, the transformed signal has to undergo downsampling to reduce the number of coefficients to process. The process of generating the wavelet coefficients and downsampling is accomplished by a decomposition filter bank. The process of assembling the wavelet coefficients to generate the original signal is called inverse wavelet transform. This process is accomplished by upsampling the wavelet coefficients and processing them through a synthesis filter bank. See Figure 46.

The use of two-channel filter banks is essential on the implementation of the discrete wavelet transform (DWT) (38). The forward discrete wavelet transform (FDWT) breaks down an input signal into different frequency components using a decomposition filter bank and downsampling. The FDWT can be performed on a signal using high-pass and low-pass filters with the decomposition coefficients generated by db7, db4 or Haar. The inverse discrete wavelet transform (IDWT) is performed by upsampling the forward discrete wavelet transform coefficients and inputing the coefficients through a synthesis filter bank. Note that the IDWT filter bank must have the right set of synthesis coefficients (db7,db4,etc) in order to reconstruct the signal successfully (33).

5.5.4 Discrete Wavelet Transform Hardware Implementation

The ALTERA DSP Builder was used to build the filter bank using high-pass and low-pass FIR filters. The wavelet used for this filter bank was designed using the Daubechies 4 coefficients. The Daubechies decomposition and reconstruction coefficients can be found on Table 4 and Table 5.

Table 3 – DB4- Decomposition Coefficients for FIR Filter Bank

No. Coeff	LP_D Coefficients	HP_D Coefficients
1	-0.0106	-0.2304
2	0.0329	0.7148
3	0.0308	-0.6309
4	-0.1870	-0.0280
5	-0.0280	0.1870
6	0.6309	0.0308
7	0.7148	-0.0329
8	0.2304	-0.0106

Table 4 – DB4- Reconstruction/Synthesis Coefficients for FIR Filter Bank

No. Coeff	LP_R Coefficients	HP_R Coefficients
1	0.2304	-0.0106
2	0.7148	0.0329
3	0.6309	0.0308
4	-0.0280	-0.1870
5	-0.1870	-0.0280
6	0.0308	0.6309
7	0.0329	0.7148
8	-0.0106	0.2304

The input testing signal for the DSP Builder filter bank is a sinusoidal signal that contains high and low frequencies components, see Figure 47. This sinusoidal signal is the result of the addition of two other sinusoidal signals containing low and high frequencies.

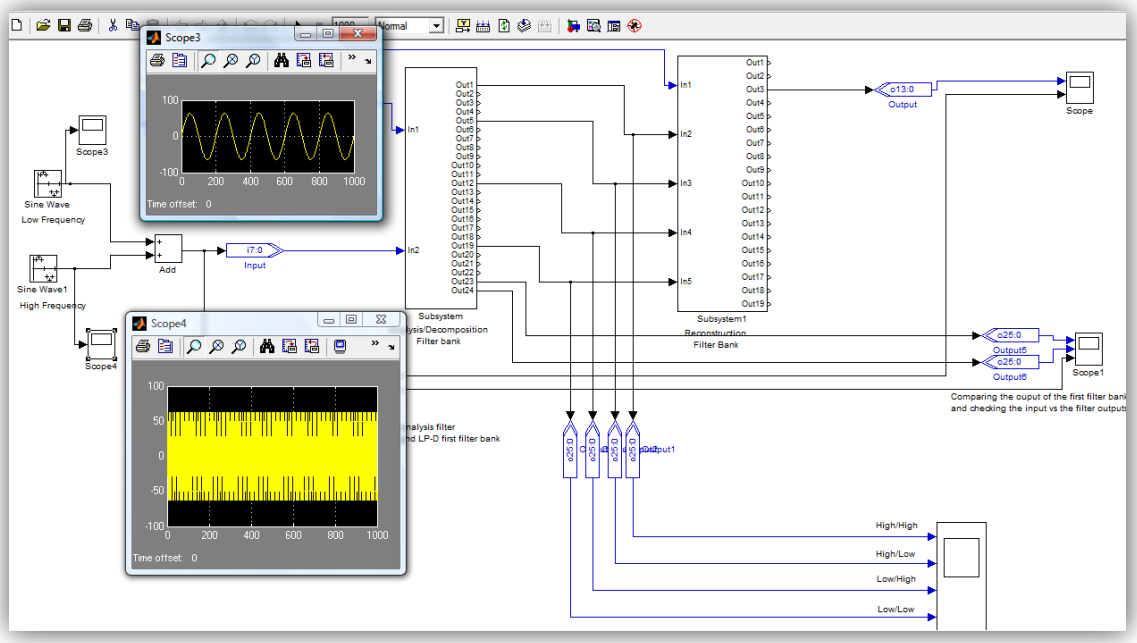

Figure 47 - Input Signal And DSP Builder Filter Bank

Once the input signal is generated, the next step is to convert the signal from a SIMULINK signal into a binary signal that can be understood by the DSP Builder blocks. This is

done by inserting a DSP Builder block like the one shown in Figure 48. Notice that it is an 8-bit input that it is used for the filter bank since our signal desired value range is from 0 – 255, or 2^8.

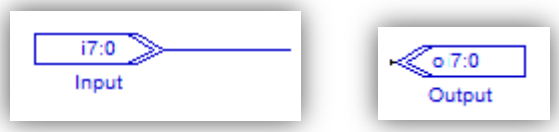

Figure 48 - SIMULINK-To-DSP Builder 8-Bit Input/Output

After converting the SIMULINK signal into bits, the FIR compiler must be opened in order to configure the FIR filters with the proper coefficients and generate the desired wavelet for the signal transformation. See Figure 49.

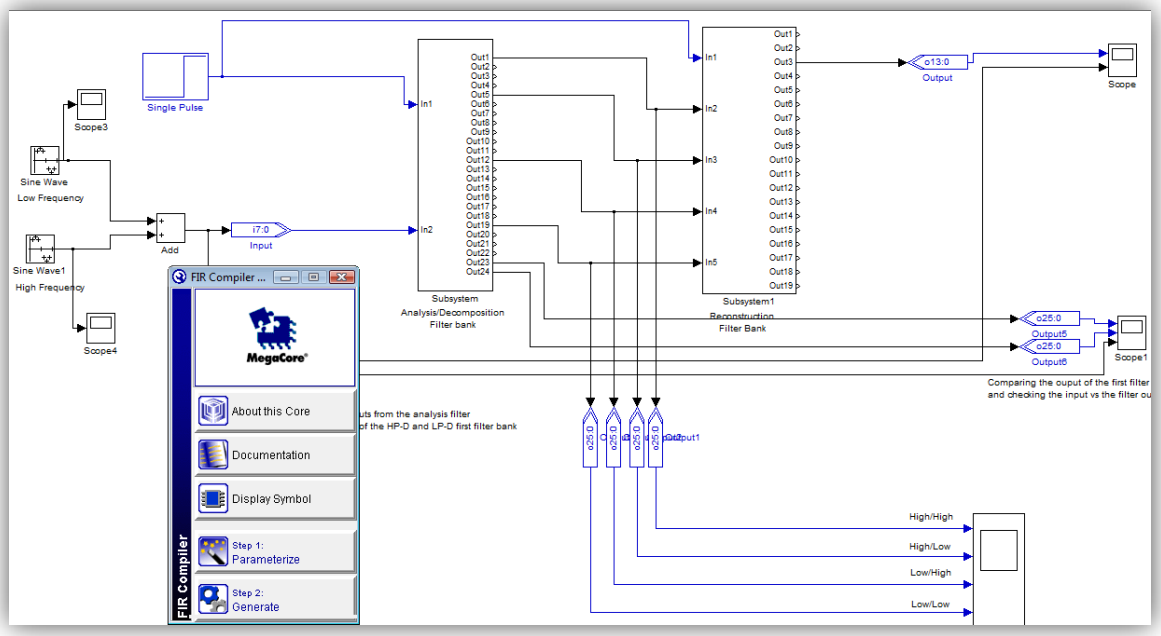

Figure 49- FIR Compiler And Filter Bank

The decomposition and synthesis filter banks were designed with the filter bank designs from (38)(9)(33). The internal FIR filters and components are shown in Figure 50and Figure 51.

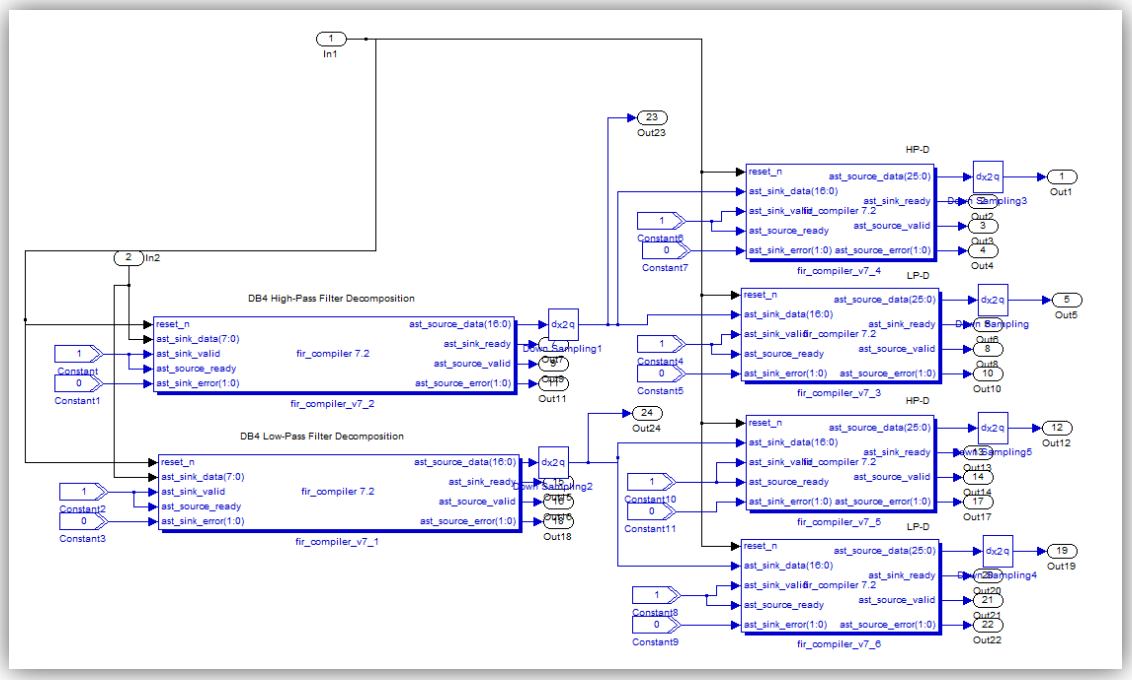

Figure 50- Decomposition FIR Filter Bank

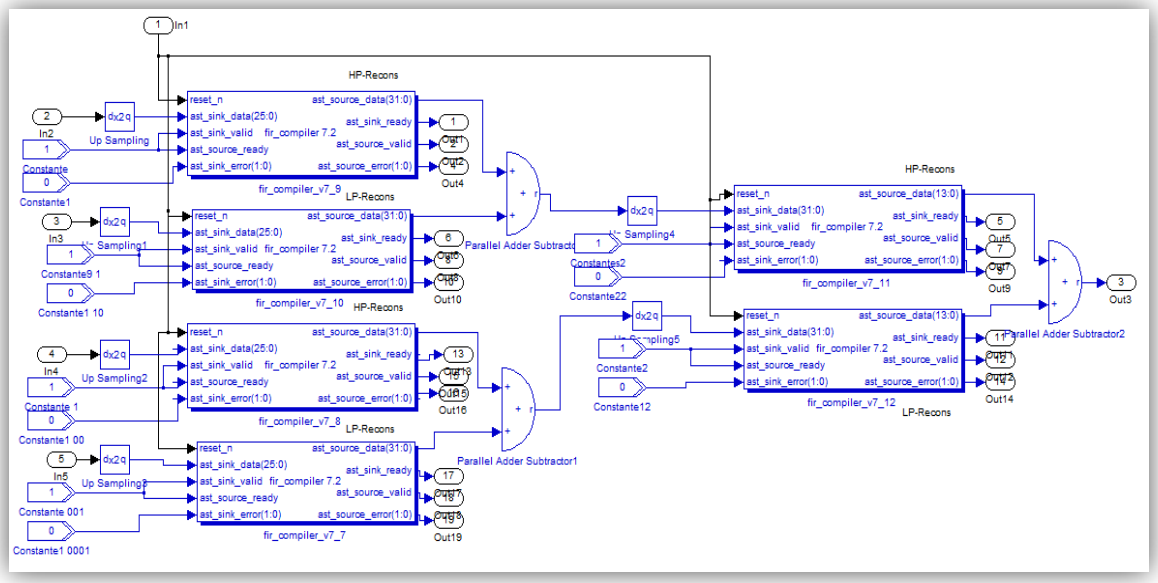

Figure 51 - Synthesis FIR Filter Bank

The DSP Builder FIR Compiler allows the user to input the coefficient set (Wavelet type), device family, scaling, the number of input channels and many more options. The FIR compiler also shows the frequency response of the FIR filter and the number of logic cells needed to build the filter.

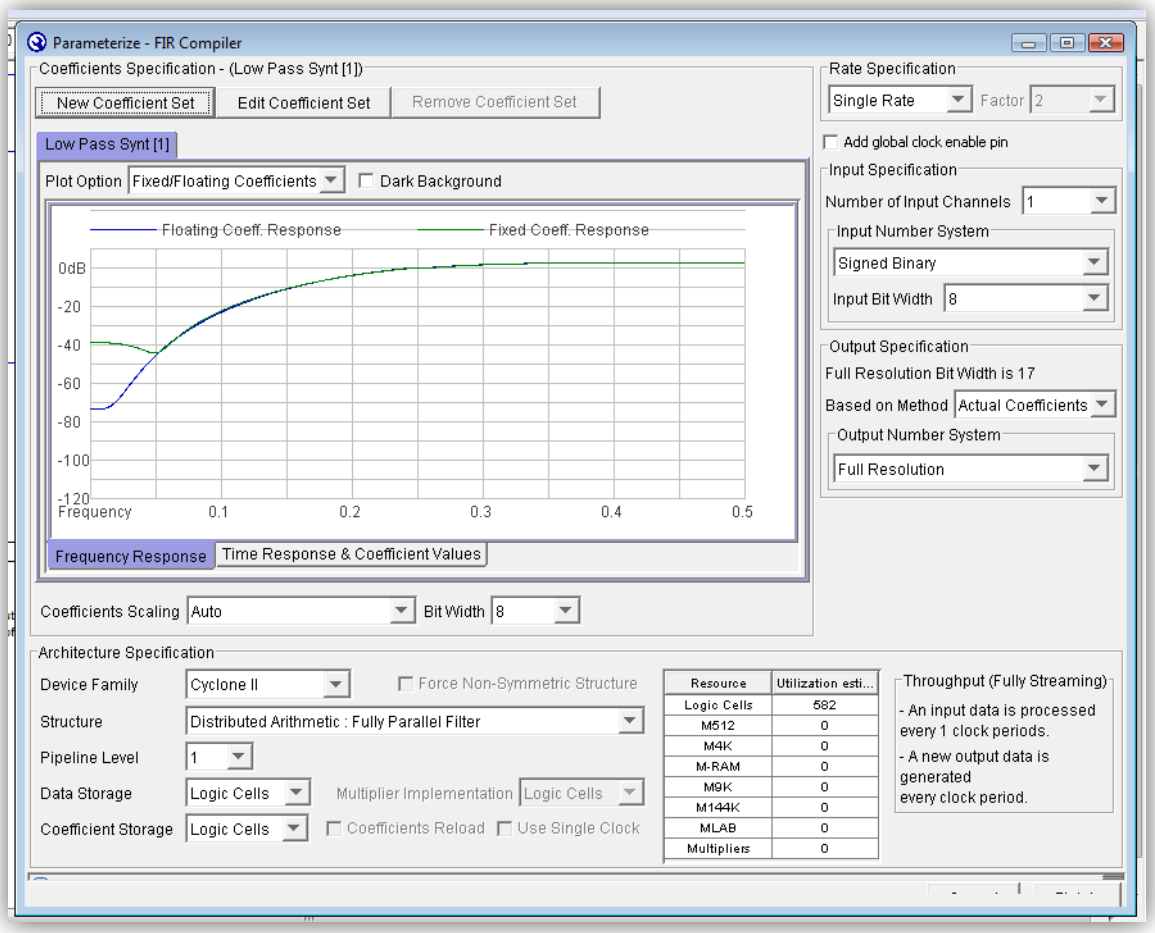

Figure 52 - High-Pass Decomposition Filter Set Up => Step 1

The step 1 is to configure the number of input channels, the resolution, the input bit width and the device family. Notice that the device family used for the filter bank in Figure 49 is the Cyclone II, which is the chip in the DE2 ALTERA FPGA.

The second step is to upload the coefficient set for the desired wavelet type. The coefficients can be uploaded from any text format file. The coefficients for the high-pass FIR

filter shown in Figure 53 were first generated in MATLAB and then copied and pasted in notepad. The notepad file was saved and subsequently uploaded into the FIR compiler. This uploading function is very practical if dealing with a large number of coefficients.

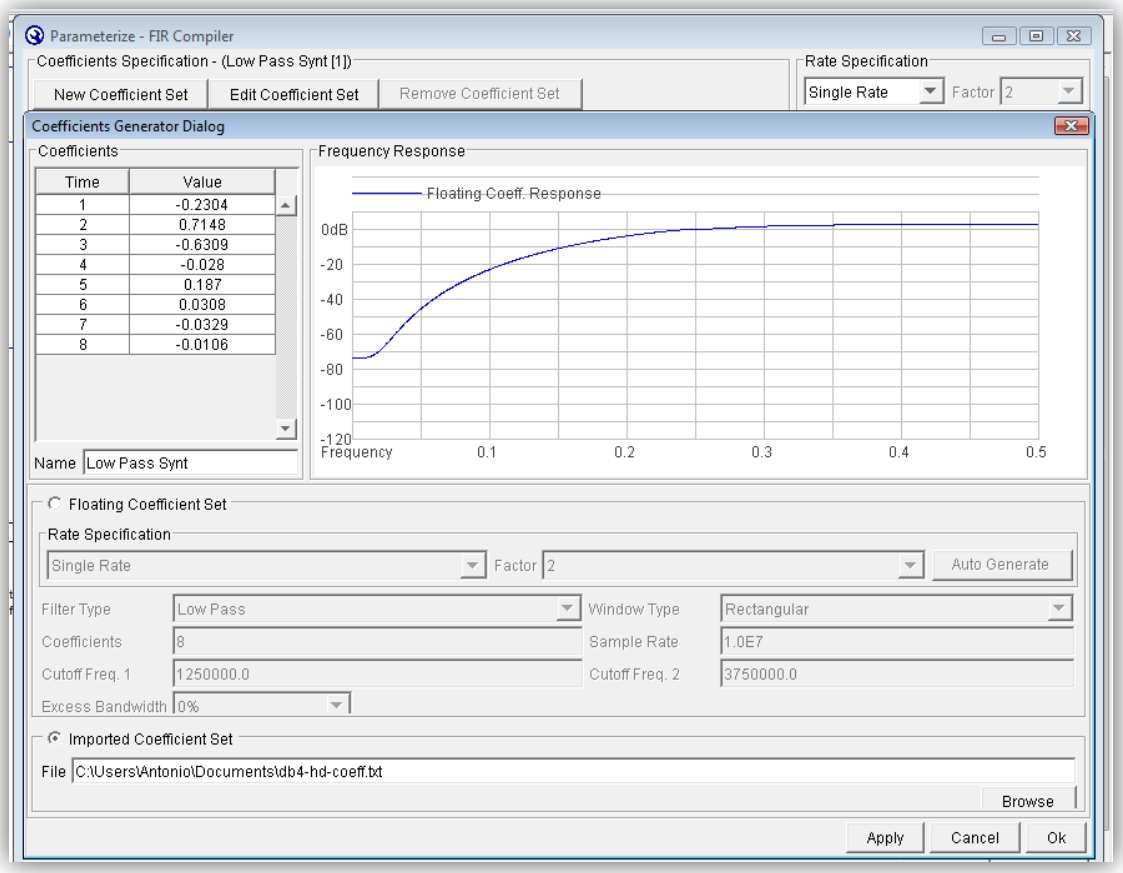

Figure 53 - High-Pass Decomposition Filter Set Up => Step 2

Once all the coefficient sets have been uploaded for the decomposition and synthesis filter banks, the SIMULINK and DSP Builder blocks are simulated and the output displayed using the “scope” blocks. Figure 54 shows the signal’s frequency components and Figure 55 shows the final output with the reconstructed and original images.

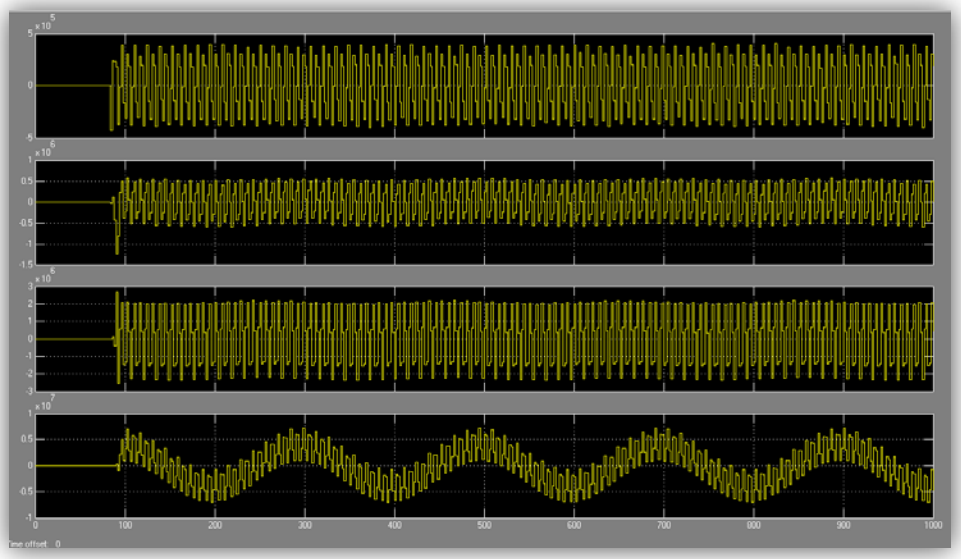

Figure 54 - Signal's High, Medium High, Medium Low And Low Frequency Components

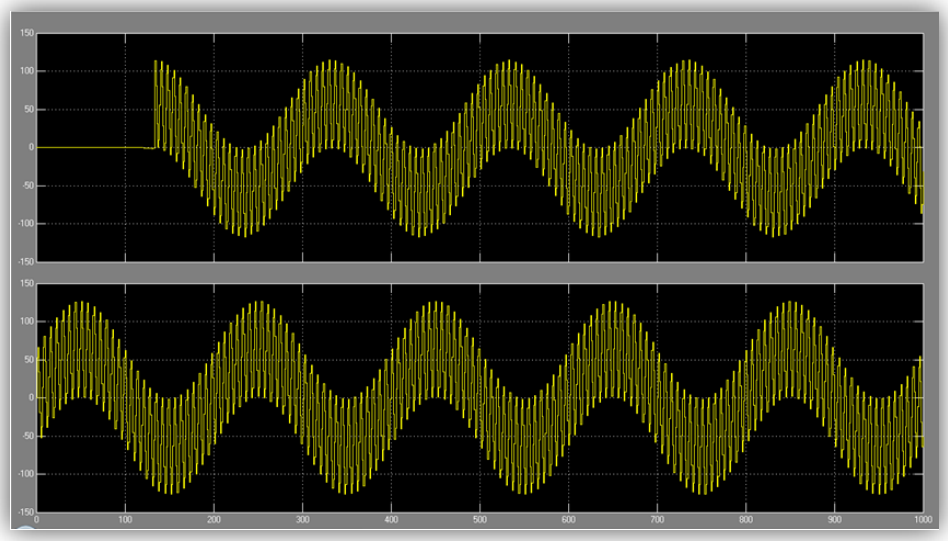

Figure 55 - Filter Bank Ouput. Top = Reconstructed; Bottom = Original

It can be seen from Figure 55 that the reconstructed image and the original signal match in magnitude and shape. The phase shift difference between signals, however, is due to the time delay originated by the filter banks. Unfortunately it does takes some time for the first set of coefficients of the input signal to get processed by the decomposition filters and reconstructed by the synthesis filter bank.

After the simulation has been performed successfully, VHDL files are generated for every FIR filter in the filter bank. These VHDL files can be used to configure the DE2 ALTERA FPGA and to create a QuartusII project to set up the hardware-in-the-loop between SIMULINK and ALTERA. See Figure 56 and Figure 57.

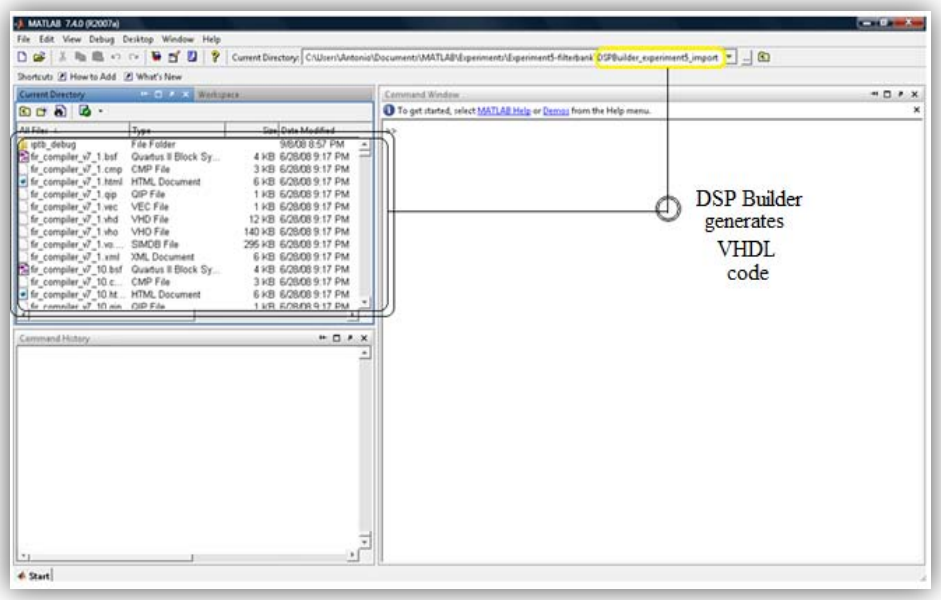

Figure 56 - DSP Builder VHDL Generation

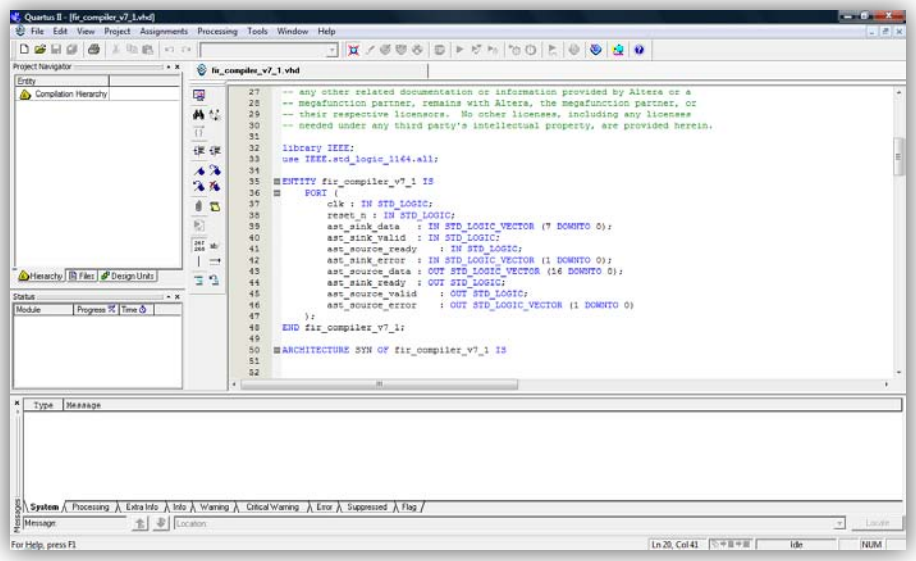

Figure 57 - Quartus Project Generated From DSP Builder VHDL File

5.6 Differences on the JPEG2000 and the MATLAB Simulation Algorithm

The JPEG2000 baseline simulation results obtained from MATLAB approximate the results from a true JPEG2000 baseline standard very closely. The compression rates obtained from the simulation only differ from a true JPEG2000 standard's compression rates by a factor of 2. One of the main differences between the simulation and the actual standard is that the simulation only approximates the JPEG2000 bit-plane-oriented arithmetic encoding algorithm instead of the complicated encoding technique used by a true JPEG2000 encoder. For the study of watermarking images using the JPEG2000 platform, this small discrepancy is inconsequential and the results approximation suffices the requirements for the analysis performed in this thesis.

CHAPTER 6

CONCLUSIONS

6.1 Analog and Digital Technology

Analog technology has been around for a long time. The use of analog technology is necessary since we live in an analog world. However, the transmission and storage of analog technology is more complicated and in many cases less efficient than digital technology. Digital technology, on the other hand, provides fast means to be transmitted and stored. Digital technology continues to grow and it is more widely used than ever before. It has occupied nearly all areas of every day for most people. Kitchens, cars, appliances and even furniture have now digital technology embedded inside. However, with the advent of new technology that can reproduce digital documents or images with unprecedented accuracy, it poses a risk to the intellectual rights of many artists and also on personal security.

6.2 Reasons for Including Watermarks

6.2.1 Intellectual Rights

One way to protect intellectual rights of digital works is by embedding watermarks in them. The watermarks can be visible or invisible depending on the application and the final objective of the intellectual work. In some cases, the watermark is visible to discourage people from copying the work illegally.

6.2.2 National Security

Watermarks could also be used to protect the identity of people and even improve homeland security. By embedding watermarks with the names, social security, and other

pertinent information, immigration officials can track those who enter and leave this nation more efficiently.

6.3 Need for Compression

Digital data has many advantages over analog data. However, even as data becomes digitalized, it does require some compression in order to be stored or transmitted efficiently. Memory space and bandwidth are two of the most common reasons for compressing digital data. Prices for memory space and devices have gone down and the use of limited bandwidth to transmit data continues to increase. As result more efficient compression algorithms are needed. The JPEG2000 standard is the next image compression standard. The JPEG2000 provides more advantages than the former JPEG.

6.4 JPEG2000 Functional Diagram and Basic Principles

JPEG2000 is a wavelet-based image compression standard created by the Joint Photographic Experts Group (JPEG) in the beginnings of the year 2000. Figure 58 shows the basic steps necessary to process an image using the JPEG2000 standard.

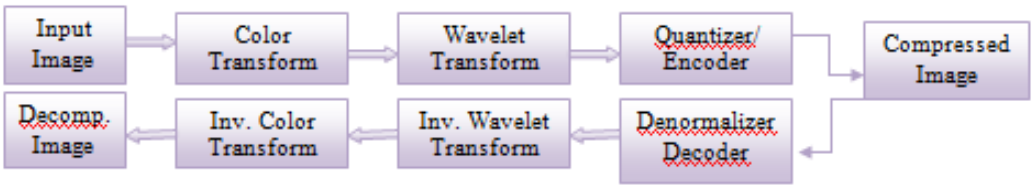

Figure 58- JPEG2000 Compression/Decompression Basic Blocks

The overall goal of creating this JPEG2000 standard is to eventually replace its predecessor the discrete cosine based JPEG standard. JPEG2000 seeks to improve compression

performance over JPEG and add extra features not found in JPEG. Some of these features are random code stream access (ROI) and multiple image resolution representation. JPEG needs to reduce the resolution in a picture before compressing the number of bits in a picture below a certain level. JPEG2000, on the other hand, can deal with any resolution since the image is already decomposed in many resolutions during the compression process.

A better understanding of the watermarking technology being used today has been obtained as well as practical knowledge on how digital image processing works. An analysis on the results from the DSP Builder software and the DE2 FPGA JPEG2000 watermarking hardware prototyping (along with the proper documentation on the work being done in MATLAB and VHDL files) has been provided to the committee of this department for revision and consideration.

6.5 Areas for Further Investigation

The present JPEG2000 watermarking algorithm only implements baseline compression and a basic watermarking scheme. Further optimization is needed on the performance of the JPEG2000 by implementing a better arithmetic encoder. The remaining blocks such as the quantization and encoding blocks need to be implemented in hardware and integrated with the color transform and wavelet blocks.

6.5.1 Wavelet Transforms On Images

The wavelet filter banks were tested using sinusoidal signals. Further research is necessary to transform images using the filter banks described in this thesis. The MATLAB code used to structure an image for the color transformation HIL block may be used as a guide to

stream image values to the filter banks in SIMULINK and perform the desired DSP Builder logic using the appropriate blocks.

6.5.2 Different Watermarking Algorithms.

The watermarking algorithm used for the JPEG2000 standard falls into the “additive” category. This type of watermarking algorithm is not very robust to external attacks and it can be easily removed by hackers or anybody interested in forging, or tempering with an image. Further research should be done in order to implement a more robust watermarking algorithm.

6.5.3 Hardware Integration.

The hardware-in-the loop (HIL) configuration was used to implement the RGB-to-YCbCr color transformation in hardware. A complete VHDL code to implement the JPEG2000 watermarking encoder is desired. Additional VHDL code needs to be written or generated to implement the encoding and quantization remaining sections of the JPEG2000 standard. It is also desired to integrate the existing VHDL code in order to create a single and complete JPEG2000 watermarking encoder algorithm.

APPENDIX A

ALTERA DSP AND JPEG2000 COMPRESSION/DECOMPRESSION MATLAB CODE

The following two MATLAB codes were obtained from the Gonzalez-Woods-Eddins' *Digital Image Processing Using MATLAB* textbook.

-im2jpeg2k.m

The function im2jpeg2k compresses an image using an approximation to the baseline JPEG2000 standard.

```
function y = im2jpeg2k(x, n, q)

% Where:
% X is the image to analyze
% N is the N-scale JPEG2000 wavelet transform = #Levels = 3.N+1
% Q is the quantization parameter
%   Copyright 2002-2004 R. C. Gonzalez, R. E. Woods, & S. L. Eddins
%   Digital Image Processing Using MATLAB, Prentice-Hall, 2004.
```

-jpeg2k2im.m

The function jpeg2k2im performs the decompression of an image that was compressed using the im2jpeg2k function. "Y" is the encoding structure outputted by the im2jpeg2k function

```
function x = jpeg2k2im(y)
%   Copyright 2002-2004 R. C. Gonzalez, R. E. Woods, & S. L. Eddins
```

```
%   Digital Image Processing Using MATLAB, Prentice-Hall, 2004.
```

-Compare.m

The compare function was obtained from the Gonzalez-Woods-Eddins' *Digital Image Processing Using MATLAB* textbook . This algorithm was used to evaluate the actual results with the expected results. The "compare.m" file outputs the root-square-mean-error of two images, a histogram and an error image of the difference between two images.

```
function rmse = compare(f1, f2, scale)

%   Copyright 2002-2004 R. C. Gonzalez, R. E. Woods, & S. L. Eddins
%   Digital Image Processing Using MATLAB, Prentice-Hall, 2004.
```

DSP Builder Input/Output MATLAB Code For SIMULINK

-ed_in2_script.m

This code was used to separate an RGB image into its red, green and blue components. Each component was then converted to “double” floating point for computational purposes. Finally, every value forming each component was configured into a structure that was input into the color transformation blocks.

-ed_out_2script.m

This code was used to integrate the Y, Cb and Cr image components and to compare the hardware and MATLAB results. This code also provides the watermark and Y,Cb and Cr components for the SIMULINK JPEG2000 encoding and decoding. The final result is a JPEG2000 watermarked RGB image.

APPENDIX B

FPGA BRAND AND MODEL USED FOR THE HARDWARE PROTOTYPING OF THE RGB-TO-YCBCR COLOR TRANSFORMATION ALGORITHM

The ALTERA DE2 Board with the Cyclone II 2C35 microprocessor FPGA in a 672-pin package will be used for the hardware prototyping of the RGB-to-YCbCr color transformation algorithm. A few of the reasons for using the ALTERA DE2 board is its versatility, availability and affordability compared to similar FPGA brands such as Xilinx. Figure 59 shows a picture of the ALTERA DE2 board that was used to implement the hardware-in-the-loop configuration for the RGB-to-YCbCr color transformation.

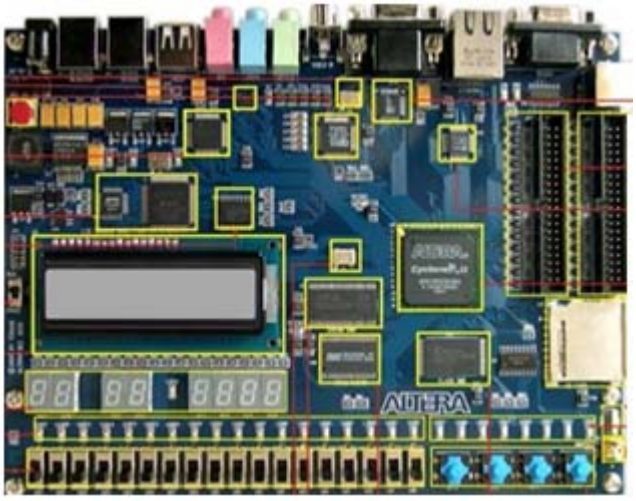

Figure 59- ALTERA DE2 Board

Microprocessor, Devices, Clock Speed And Other Characteristics

FPGA

- Cyclone II EP2C35F672C6 with 16Mb EPCS16 serial configuration device

I/O Devices

- Built-in USB Blaster for FPGA configuration
- 10/100 Ethernet
- RS232

- Video Out (VGA 10-bit DAC)
- Video In (NTSC/PAL/Multi-format)
- USB 2.0 (type A and type B)
- PS/2 mouse or keyboard port
- Line in/out, microphone in (24-bit Audio CODEC)
- Expansion headers (76 signal pins)
- Infrared port

Clock

- 27 and 50 MHz crystals for FPGA clock input
- External SMA clock input

Memory

- 8MB SDRAM, 512K SRAM, 1MB Flash
- SD memory card slot

Displays

- 16 x 2 LCD display
- Eight 7-seg displays

Switches and LEDs

- 18 toggle switches
- 18 red LEDs
- 9 green LEDs

REFERENCES

1. Cai, Wei. *FPGA Prototyping of a Watermarking Algorithm For MPEG4.* [Thesis] Denton, TX : University of North Texas, May 2007. FPGA Prototyping of a Watermarking Algorithm For MPEG4. 174245407.

2. Heiner Hanggi, Theodor H. Winkler. *Challenges of Security Sector Governance.* [Document] NJ : Transaction Publishers, DCAF, 2003. ISBN 3-8258-7158-4.

3. Andy Jones, Gerald L. Kovacich, Perry G. Luzwick. *Global Information Warfare.* FL : Auerbach , 2002. ISBN 0-8493-1114-4.

4. Cole, E. *Steganography, Hiding In Plain Sight.* IN : WILEY, 2003. 10: 0471444499.

5. Lu, Chun-Shien. *Steganography and Digital Watermarking Techniques For Protection of Intellectual Property.* USA : Idea Group Inc, 2005. 1-59140-192-5.

6. Tinku Acharya, Ping-Sing Tsai. *JPEG2000 Standard for Image Compression, concepts, algorithms and VLSI architectures.* NY : WILEY, 2005. 0-471-48422-9.

7. Group, Joint Photographic Experts. JPEG Compression Standard. *JPEG.* [Online] 1992. [Cited: 05 01, 2008.] www.jpeg.org.

8. Russ, C. John. *The Image Processing Handbook, 4th Edition.* USA : CRC Press, 2002. 0-8493-1142.

9. Rafael C. Gonzalez, Richard E. Woods, Steven L. Eddins. *Digital Image Processing Using MATLAB.* CA : Pearson Education, 2007. 9780130085191.

10. *A Method for the Reconstruction of Minimum Redudancy Codes.* Huffman, D. Cambridge, MA : Proceedings of The IRE, 1952. Proc. of the IRE. pp. 1098-1101. IRE-1098.

11. *Digital Watermarking In The Wavelet Transform.* Meerwald, Peter. 5, Salzburg : IEEE, January 2001, Vol. 48, pp. 875-882. 0278-0046.

12. M. Awranjeb, Kankanhalli. *Lossless Watermarking Considering The Human Visual System.* [Lectures Note in Computer Science] Seoul : National University of Singapore, 2003. 0302-9743 .

13. *Novel Architecture for the JPEG2000 block coder.* Darren Freeman, Greg Knowles. 897, Adelaide, Australia : Jorunal of Electronic Imaging, 2004, Vol. XIII, pp. 117-130. 897-906.

14. Antonin Descampe, Francois Devaux, Gael Rauvroy, Benoit Macq, Jean Dider. *An efficient FPGA Implementation of a Flexible JPEG2000 Decoder for Digital Cinema.* [Document] Louvain, Belgium : Universite Catholique de Louvain, Universit´e catholique de Louvain, 2003.

15. *On the Digital Watermarking in JPEG2000.* Suhail, Obaidat. na, NJ : IEEE, 2001, Vol. 2, pp. 871-874 . 0-7803-7057-0.

16. MathWorks. Mathworks. *MATLAB.* [Online] MATLAB, 2008. [Cited: 7 12, 2008.] www.matlab.com.

17. *A Simple and Efficient Watermarking Technique Based on JPEG2000 Codec.* Tong-Shou Chen, Jeanne Chen, Jian-Guo Chen. na, Taiwan : IEEE Fifth International Symposium on Multimedia Software Engineering, 2004, Vol. 1. 0-7695-2031-6/03.

18. *A Comparative Study of Digital Watermarking in JPEG and JPEG2000 Enviroments.* Shuhail Obait, Ipsodoun. na, s.l. : Science Direct, 2003, Information Sciences, Vol. 151, pp. 93-105. 10.1016/S0020-0255(02)00291-8 .

19. *FPGA Based Implementation of An Invisible-Robust Image Watermarking Encoder.* Mohanty, S.P. 1, Heidelberg : Springer Berlin, January 2005, Vol. 3356, pp. 344-353. 0302-9743.

20. Michael Tsvetkov, Vyacheslav Gulyaev. *Color Converter: Overview* . [Document] NY : Open Cores, 2007. na.

21. *Document Processing for Automatic Color Document Form Dropout.* Andreas E. Savakis, Chris R. Brown. NY : Deparment of Computer Engineering, Rochester Institute of Technology, 2005.

22. *VHDL Based Design of an FDWT Processor.* Aziz, Matteo Michel. Australia : IEEE, 2003, Vol. IV, pp. 1609- 1613 . 0-7803-7651-x/03.

23. *Wavelet Domain Adaptive Visible Watermarking.* Yongjian Hu, Sam Kwong. 20, Hong Kong, China : IEEE, 2001, IEEE Electronic Letters, Vol. 37, pp. 1219-1220. 7070534.

24. *An Image Fusion Based Visible Watermarking Algorithm.* Yongjian Hu, Sam Kwomg. na, Guangzhou, China : IEEE, 2003, IEEE Press, Vol. 3, pp. 794-797. 0-7803-7761-3.

25. *A Contrast Sensitive Watermarking Scheme.* Biao-Bing Huang, Shao-Xian Tang. 2, LA, CAL : IEEE MultiMedia, 2006, IEEE Press, Vol. XIII, pp. 60-66. 1070-986X.

26. *A Simulink-Based Hybrid Coding Tool for Rapid Prototyping Of FPGA's In Signal Processing Systems.* Reyneri, L.M. 5-6, Torino, Italy : Science Direct, 2004, Vol. 28, pp. 273-289. 0141-9331.

27. *Discrete Wavelet Transform FPGA Design using MatLab/Simulink.* Uwe Meyer-Baesea, A. Verab, A. Meyer-Baesea, M. Pattichisb, R. Perrya. na, Orlando, FL : SPIE, 2006, Vol. 6247. 10.1117/12.663457 .

28. ALTERA. DSP Builder User Guide. USA : ALTERA, 2008. Vol. 7.2.

29. ALTERA CORP. Video and Image Processing Suite - User Guide. San Jose , CA : ALTERA , 2007. Vol. 7.2.

30. *Reusable Silicon IP Cores for Discrete Wavelet Transform Applications.* Shahid Masud, John V. McCanny. 6, Pakistan : IEEE, 2004, Vol. 51, pp. 1114- 1124. 1057-7122/04.

31. *Wavelet Processing Implementation in Digital Hardware.* P.M. Szecowka, M Kowalski, K. Krysztoforski, A.R Wolczowski. na, Poland : Department of Microelectronics & Computer Science, Technical University of Lodz, 2007, Vol. 14, pp. 651-654. 83-922632-9-4.

32. Xuyun Chen, Ting Zhou, Wei Li, hao Min. *A VLSI Architecture for Discrete Wavelet Transform.* [Document] Shanghai, China : IEEE, IEEEXplore, 1996. 0-7803-3258/96.

33. *VLSI Implementation Of Discrete Wavelet Transform (DWT).* Abdullah Al Muhit, Md. Shabiul Islam and Masuri Othman. Palmerston North, New Zealand : 2nd International Conference on Autonomous Robots and Agents, 2004.

34. Mallat, Stephane. *A Wavelet Tour of Signal Processing, 2nd Edition.* Oxford, UK : Academic Press, 2003. 0-12-466606-X .

35. *Segmentation by Color Sspace Transformation Prior to Lifting And Integer Wavelet Transformation.* Gilberto Zamora, Shuyu Yang, Mark Wilson, and Sunanda Mitra. Lubbock, Texas : IEEE, 2000. pp. 136-140. 0-7695-0595-3.

36. *Design and Implementation of a Wavelet Based System.* Mokhtar Nibouche, Omar Nibouche, Ahmed Bouridane. 1, England : IEEE, 2003, Vol. 2, pp. 463- 466 . 0-7803-8163-7/03.

37. *VHDL Implementation of Wavelet Packet Transforms Using SIMULINK Tools.* Mukul Shirvaikar, Tariq Bushnaq. 1, San Jose, CA : Electronic Imaging, 2008, Vol. 6811, pp. 50-62. 0277-786x/08.

38. Meyer-Baese, Uwe. *Digital Signal Processing With Field Programmable Gate Arrays.* Tallahassee, FL : Springer, 2004. 3-540-21119-5.