Logarithmic Discrete Wavelet Transform For High Quality Medical Image Compression

Mohammed Shaaban Ibraheem

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité : Informatique

École doctorale : EDITE 130 Paris

réalisée

à LIP6

présentée par

Mohammed Shaaban IBRAHEEM

pour obtenir le grade de :

DOCTEUR DE L'UNIVERSITÉ PIERRE ET MARIE CURIE

Sujet de la thèse :

Logarithmic Discrete Wavelet Transform for High Quality Medical Image Compression

soutenue le 29/03/2017

devant le jury composé de :

M^{me}.	YANG-SONG Fan	Rapporteur
M.	RABAH Hassan	Rapporteur
M.	MEHREZ Habib	Examinateur
M.	BENSRHAIR Abdelaziz	Examinateur
M.	LEMIRE Daniel	Examinateur
M.	HACHICHA Khalil	Co-encadrant
M.	HOCHBERG Sylvain	Co-encadrant
M.	GARDA Patrick	Directeur de thèse

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisors Prof. Patrick GARDA and Dr. Khalil HACHICHA for the continuous support of my PhD study and related research, for their patience, motivation, and immense knowledge. Their guidance helped me in all the time of research and writing of this thesis. I could not have imagined having better advisors and mentor for my Ph.D study.

Besides my advisors, I would like to thank the team in CIRA Company, represented by: Mr. Sylvain HOCHBERG, and Mrs. Imen MHEDBI and Ms. Sabrina DJEDI for their insightful comments and encouragement, but also for the hard question which incented me to widen my research from various perspectives.

I thank all the SYEL team members, especially my fellow lab-mates Syed Zahid AHMED, Laurent LAMBERT, Imen DHIF, Chen CHEN, Alexandre BRIÈRE, Orlando CHUQUIMIA, Olivier TSIAKAKA , for the stimulating discussions, and for all the fun we have had in the last four years.

My sincere thanks also goes to Prof. Bertrand GRANADO, SYEL team leader for his time and his plenty discussions and sharing the latest scientific ideas which helped me a lot.

Last but not the least, I would like to thank my family: my mother, my brother, sister and my lovely wife for supporting me spiritually throughout writing this thesis and my life in general.

Abstract

ABSTRACT

Nowadays, medical image compression is an essential process in eHealth systems. Compressing medical images in high quality is a vital demand to avoid misdiagnosing medical exams by radiologists. WAAVES is a promising medical images compression algorithm based on the discrete wavelet transform (DWT) that achieves a high compression performance compared to the state of the art. The main aims of this work are to enhance image quality when compressing using WAAVES and to provide a high-speed DWT architecture for image compression on embedded systems. Regarding the quality improvement, the logarithmic number systems (LNS) was explored to be used as an alternative to the linear arithmetic in DWT computations. A new LNS library was developed and validated to realize the logarithmic DWT. In addition, a new quantization method called (LNS-Q) based on logarithmic arithmetic was proposed. A novel compression scheme (LNS-WAAVES) based on integrating the Hybrid-DWT and the LNS-Q method with WAAVES was developed. Hybrid-DWT combines the advantages of both the logarithmic and the linear domains leading to enhancement of the image quality and the compression ratio. The results showed that LNS-WAAVES is able to achieve an improvement in the quality by a percentage of 8% and up to 34% compared to WAAVES depending on the compression configuration parameters and the image modalities.

For compression on embedded systems, the major challenge was to design a 2D DWT architecture that achieves a throughput of 100 full HD frame/s. A novel unified 2D DWT computation architecture was proposed. This new architecture performs both horizontal and vertical transform simultaneously and eliminates the problem of column-wise image pixel accesses to/from the off-chip DDR RAM. All of these factors have led to a reduction of the required off-chip DDR RAM bandwidth by more than 2X. The proposed concept uses 4-port line buffers leading to pipelined parallel four

operations: the vertical DWT, the horizontal DWT transform, and the read/write operations to the external memory. The proposed architecture has only 1/8 cycles per pixel (CPP) enabling it to process more than 100fps Full HD and it is considered a promising solution for future 4K and 8K video processing. Finally, the developed architecture is highly scalable, outperforms the state of the art existing related work , and currently is deployed in a video EEG medical prototype.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

Contents

1.1 Thesis Motivation

Today E-health systems are dominating the medical society thanks to the rapid growth and evolution of the information technology (IT). Hospitals and medical centers depend widely on all the new technologies starting from the simple medical signals measurement devices, e.g. (blood pressure, sugar level, body temperature, electrography, etc.). On the other hand, large medical devices like medical imaging acquisition became an essential tool for radiologists to diagnose diseases [1]. By the time, the continuous increasing in the image resolution and the number of their modalities e.g., (magnetic resonance imaging (MRI), X-Ray, Computed Tomography (CT), ultrasound, etc.) have created many obstacles. That raised the needs to have a huge storage capacity to meet the daily exams archiving requirements, which costs the hospitals heavily.

Figure 1.1 shows an example of the medical imaging in the eHeath systems [1].

[1] Photo credit: www.bu.edu

Fig. 1.1 Medical imaging in eHeath systems

In addition to that, there are many critical and urgent health problem cases that require radiologists to ask for a diagnostic help from experts who are not in the same hospital at that time. That requires transferring the medical exam via the internet to receive a remote consultation as shown in Figure 1.2.

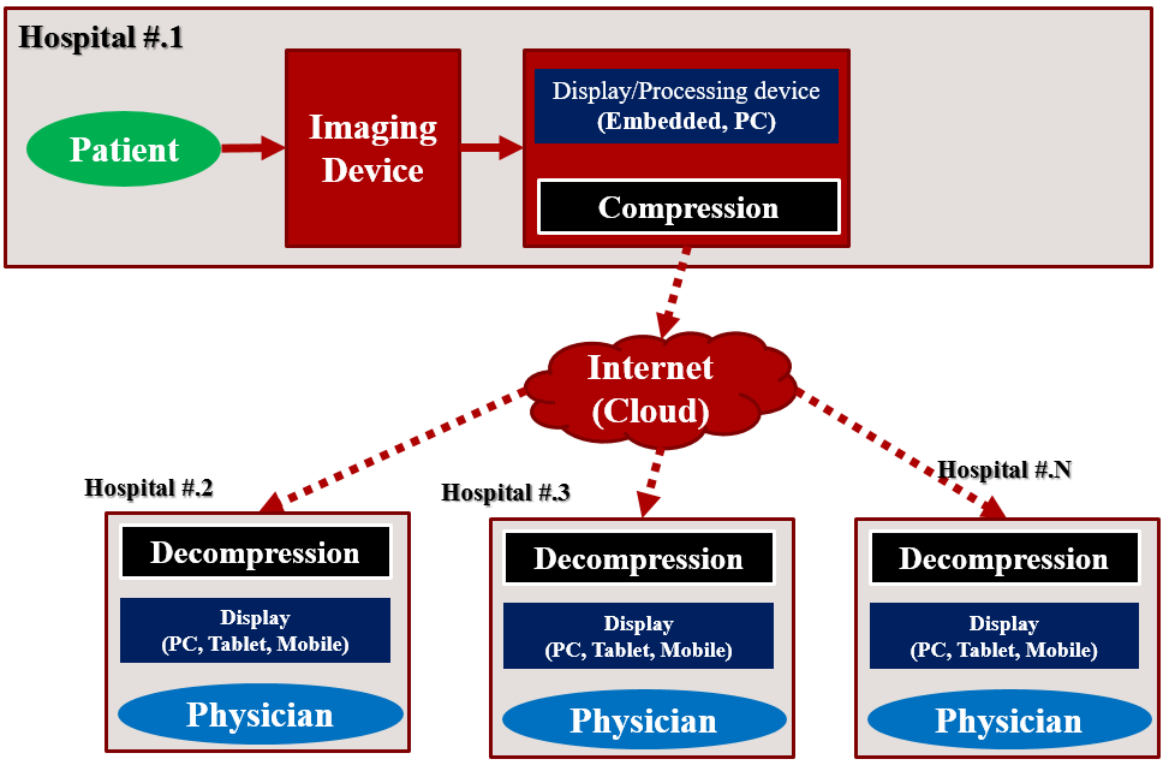


Fig. 1.2 Remote consultation becomes essential in eHealth today

When sending the medical exams online, the key issue appears in the places that have a limited bandwidth. That makes the upload time is longer and causes a slower

download speed on the other side. Since the time is a critical factor to save the patients' lives when they need quick decisions in the emergency cases, it became a big challenge to reduce the transfer and the receive time. Consequently, medical image compression became an essential tool in E-health systems.

In summary, there are several advantages of the image compression in the medical domain that motivated us to explore the challenges and to find the solutions. Image compression helps to reduce the required archiving storage space used in the hospitals, which reduces the storage cost. Also, that makes it feasible to consume less bandwidth when sending and receiving the medical exams, consequently, it reduces the send/receive time via the internet. Finally, it helps in managing the archived medical images faster and more efficient.

1.2 Challenges in the Medical Imaging

Medical image compression became dominant in the medical informatics field. Image compression algorithms that are used in the medical domain provide two types of compressions: lossless and lossy. Although lossless compression provides the perfect reconstruction of the compressed image, it has a limitation in the compression ratio. That is due to the need to preserve more information while performing the compression process [2, 3]. Thus, lossy compression became an interesting topic for the researchers. On the other hand, another challenge is performing the image compression on embedded devices to utilize the device portability feature that enables the availability and the reliability of the remote consultation.

In general, there are the two big challenges in the medical image compression field, which motivated us to work on this research area. They are summarized as the following:

1. The challenge of the compromise between the image quality and the compression efficiency.

2. Compression on embedded systems.

The challenge of the compromise between the image quality and the compression efficiency

Firstly, medical images are vital information, therefore, preserving a sufficient quality on image compression is a critical factor to radiologists to avoid misdiagnoses. Secondly, compression efficiency in terms of the compressed file size is a key element to transfer

the file in a high-speed, in addtion, to reduce the archiving cost by saving the storage space. However, the main problem appears when compressing with a high compression ratio, which leads to a degradation in the image quality. Table 1.1 lists exsmaples of the storage requirements for different medical exams modalities [4, 5].

Table 1.1 Storage requirements for different medical image Modalities

Modality	Organ	Resolution	BPP	No. of images	Exam Size
MRI	brain	512 ×512	16	500	250MB
	abdomen	512×512	16	800	400MB
	heart	256×256	16	2020	252MB
CT	brain	512×512	16	500	250MB
	abdomen	512×512	16	800	400MB
	heart	512×512	16	2020	1G
Angiography	Cardiac	1024×1024	8	2000	2G
X-ray	Chest	2060×2060	16	-	8MB
Mammography	Breast	4000×5000	12	-	28MB
Retinal	Eye	2240×1488	RGB,8	-	10MB

Table 1.2 list a comparison between the different image types and the required bandwidth to transmit them [6]. It indicates how the medical images require a large bandwidth compared to the other types of images which is evident that image compression is essential to save the transmit time.

Table 1.2 The required bandwidth to transmit different types of images

	Size/Duration	Transmission Bandwidth
Grayscale Image	512 × 512	2.1 Mb/image
Colored Image	512 ×512	6.29 Mb/image
Medical Image	2048 × 1680	41.3 Mb/image
Super HD Image	2048 × 2048	100 Mb/image
Full-motion Video	640 × 480, 1 Min (30 frames/Sec)	221 Mb/sec

According to the previous issues, we raised the first research question as:

- **How to achieve an efficient image compression with preserving the diagnostic quality?**

Compression on Embedded Systems

Nowadays, radiologists look for the availability of the images anywhere and anytime to give a fast respond diagnose in the emergency cases. Thus, the need of running the compression algorithm on embedded systems became an essential demand. However, embedded systems have many constraints, for example, it has a limited processing speed, and limited memory resources. Therefore, it raises a challenge to adapt the compression algorithm with the available resources on the embedded systems.

The work presented here is a part of the Smart-EEG project. which is an ongoing telemedicine project aiming to perform remote video EEG exams. Figure 1.3 shows the basic idea of this project. Medical signals of a patient, such as EEG (Electroencephalogram), ECG (Electrocardiogram) are recorded using an EEG cap as shown in Figure 1.4[2]. The EEG signal is combined with the video scene to be compressed, then to be sent via the internet to the desired destination. Adding video allows doctors to detect fast myoclonus jerks on the patient. In current systems, the maximum frame rate that can be recorded and transmitted is limited to 30 frame/s. Since the current frame accusation rate is taken every 33ms, this frame rate does not provide a full detection of the myoclonus jerks that requires 10ms which is estimated by 100 frames per second (fps) [7]. To achieve this goal, the video must be in high quality as well as high frame rate. Below this frame rate, doctors are not able to guarantee a trusted detection of the pathology. That took us to another challenge:

- **How to adapt the compression algorithm to meet the acceptable speed requirement?**

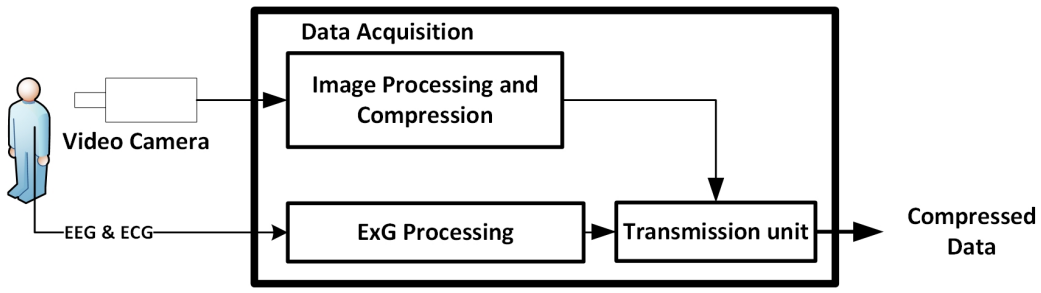


Fig. 1.3 The fundamentals of Smart EEG telemedicine system.

[2]Photo credit: www.biosemi.com

Fig. 1.4 EEG Cap is used to read the EEG signals

1.3 Thesis Organization

The rest of the thesis is organized as the following:

Chapter Two provides an overview of the background of main elements in the image compression domain. It also reviews the most popular the state of the art image compression algorithms used for medical images. Addition to, it discusses the issues regarding the image quality and what are the preferred metrics for image quality assessment. It also, gives a review of the state of the art of the existing DWT hardware architectures.

Chapter Three It describes the proposed logarithmic-based library and discusses the mathematical issues and the challenges when using the logarithmic arithmetic. It also presents the proposed DWT implementation using the proposed LNS library.

Chapter Four provides analysis of the impact of applying the logarithmic DWT in image compression. Then, it presents the proposed logarithmic DWT-based compression scheme. It also discusses the obtained results when applying the new compression scheme on the medical images, also it gives deep analyses of all the factors that have effects on the compression efficiency.

Chapter Five is embedded systems oriented; it proposes a new hardware architecture to give a solution to the speed bottleneck on hardware.

Finally, **Chapter Six** draws the conclusion of the conducted research in this thesis and gives the suggestions for the future work.

Chapter 2

Background and State of The Art

Contents

2.1 Introduction

This chapter covers the concepts and the background of the main elements required to conduct the research. It is organized as the following: Section 2.2 overviews the most popular image compression algorithms that are used in the medical domain. Section 2.3 addresses the issues related to the image quality in the image compression algorithms. Section 2.4 discusses the image compression on embedded systems. Finally, in Section 2.5, we draw the conclusion on the issues related image compression.

2.2 Image Compression Algorithms

In this section, we will illustrate the three popular algorithms that are used in medical image applications: JPEG, JPEG2000, and WAAVES.

2.2.1 JPEG

The Joint Photographic Experts Group (JPEG) is a standard image compression algorithm (ISO/IEC 10918) [8, 9]. Its compression flow consists of four stages: first, the color space of input image is transformed from RGB to YCbCr. After that, each color component matrix is divided into blocks of a size 8×8 to be processed by the discrete cosine transform (DCT) to generate the DCT coefficients. The output of the DCT stage is quantized using the uniform quantization. The quantized values are encoded using one of the two entropy encoders; Huffman [10] or arithmetic encoder [11]. The JPEG compression flow is shown in Figure 2.1.

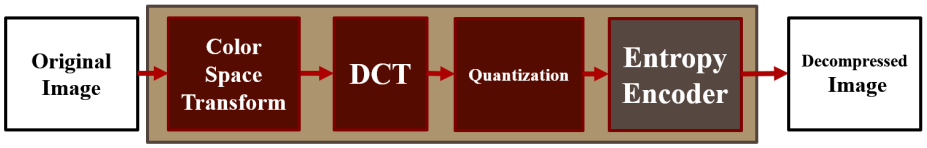


Fig. 2.1 JPEG block diagram

The main limitation of the JPEG is the quality degradation when using high quantization step. Also, the segmentation of the input image into blocks of 8×8 pixels that introduces the blocking artifacts [12].

2.2.2 JPEG2000

Unlike the JPEG which uses the DCT to transform the input image, JPEG2000 uses the DWT instead of it. Although DWT requires more computation time, it eliminates the blocking artifacts. That enhanced the image quality significantly compared to the JPEG. JPEG2000 has two compression modes: lossy and lossless. It uses two different DWT filters: Cohen-Daubechies-Feauveau CDF 9/7 [13] for the lossy compression and the LeGall 5/3 for the lossless compression [14]. JPEG2000 uses an entropy encoder which is based on the arithmetic coding. The compression flow of JPEG2000 is shown in Figure 2.2.

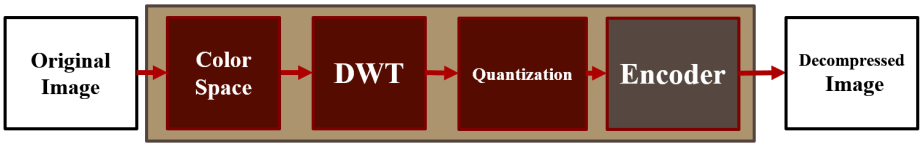


Fig. 2.2 JPEG2000 block diagram

2.2.3 WAAVES

WAAVES [15, 16] is a modern image compression algorithm it was developed by CIRA company [17]. WAAVES accepts RGB color components and transforms them into YCbCr. Also, it supports CMYK (Cyan, Magenta, Yellow, and Black), but in this case, it converts them into RGB, then into YCbCr. Then, the DWT is calculating for each component. WAAVES uses the same DWT filters that are used in JPEG2000. The quantized DWT coefficients are coded using an entropy encoder. WAAVES has an efficient encoder based on Hierarchical Enumerative Coding (HENUC) [18] which gives much better compression than JPEG2000.

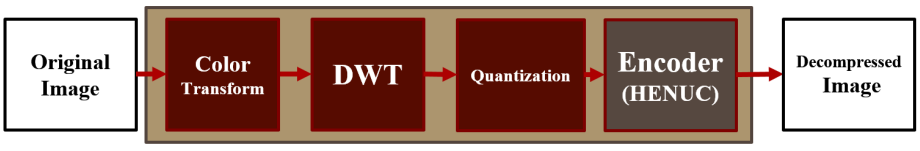


Fig. 2.3 WAAVES Compression flow block diagram

Compression Algorithms Summary

To summarize, the image compression algorithms are evaluated using three factors: the compression efficiency, the image quality, and the computation complexity.

JPEG has a better speed than the other compression algorithms since it is based on the DCT. However, it has a limitation in the image quality due to the blocking artifacts. On the other hand, WAAVES is considered better than JPEG2000 in terms of the image quality and the compression ratio [17]. Also, WAAVES is certified for medical applications.

Image compression algorithms are divided into three main stages: an image forward transform (DWT or DCT), a quantization, and an encoding. Since one of the objectives of this work is to explore how to improve the image quality, we started first by investigating the two main sources that affect the image quality: the precision of the arithmetic in the transform stage, and the quantization error. In the next section, we will discuss those issues in details.

2.3 Image Quality Issues in Image Compression

As we stated before, there are two issues that affect the image quality: the arithmetic precision and quantization. We will cover those two topics in this section. Also, we will list the popular image quality assessment metrics and which one of them is suitable for medical imaging applications.

2.3.1 Arithmetic for Digital Signal Processing

Most of the algorithms in digital signal processing (DSP) applications include real number calculations. Thus, there are many data types that give a trade-off between the precision and the computation speed. As we mentioned in the previous section, image compression algorithms include a transform operation on the input image before the encoding stage, e.g. the DWT or the DCT. That transform uses the real numbers during the calculation process. Therefore, the arithmetic precision has an impact on the image quality.

The data representation plays an important role in the computation precision. Depending on the application, we choose the appropriate numerical representation. There are two main number representations: the floating point (FLP) which is suitable for applications that include real number calculation, and; the second type is the fixed-point (FXP) that represents the real data in integer format.

Floating point

The floating point (FLP) representation is defined in the IEEE 754 [19] with two different word-length formats; the single precision format is 32-bit, and the double precision format is 64-bit. FLP consists of 3 parts: sign bit, exponent, and mantissa. Considering the single precision format [20] as shown in Figure 2.4, there are two possibilities for the dynamic range depending on the exponent value if it is equal zero (denormalized) or not (normalized) as the following:

$$\begin{array}{lllll} \text{denormalized:} & \pm 2^{-149} & \text{to} & (1-2^{-23})\cdot 2^{-126} \\ \text{normalized:} & \pm 2^{-126} & \text{to} & (2-2^{-23})\cdot 2^{127} \end{array}$$

Fig. 2.4 Single precision FLP format (32-bit)

The last decade, many publications addressed the usage of the floating-point in many applications that require higher accuracy than fixed-point (single precision), [21–28]. Also, several publications addressed the double precision [29–32].

Fixed point

The second data type is the fixed point (FXP) format, which is used as an alternate to the floating-point. Because the calculation using FLP is expensive and slow. FXP is considered an attractive candidate for DSP algorithms implementation on hardware. That is due to its simplicity which gives faster spped for the arithmetic operations. However, FXP has a limitation regarding the dynamic range that affects the accuracy. As shown in Figure 2.5 a W-bit FXP consists of three parts: the sign bit, the integer part w_i, and the fractional part w_f, where $w = 1 + w_i + w_f$ and its dynamic range is described in equation (2.1) [33].

Fig. 2.5 FXP format of W-bit word length

$$-2^{w_i} \leq x < 2^{w_i} \tag{2.1}$$

Logarithmic Number System

The logarithmic number system (LNS) has been recently introduced in the literature [34–37] as an alternative solution to the floating-point. It also can provide a high precision and a wider dynamic range than the fixed-point. LNS is an interesting research area because it gives a simple implementation of the multiplication, division, square root and square of numbers. Although the addition and the subtraction operations in LNS are complicated, many publications have presented that LNS can achieve a higher performance than floating-point [35, 38]. Due to the raising of the interest in the LNS research area, an LNS library has been introduced by [39]. The main objective of that library is to provide a tool for the purpose of the comparison with the fixed-point format on hardware in terms of speed and accuracy.

The LNS format is inherited from the fixed-point representation. Although there is no standard to define the format as the IEEE FLP, however, the LNS format in [40] [36] is commonly used. LNS is defined in Figure 2.6 [36]. The exponent value for the base 2 is described as in equation (2.2):

$$X = (-1)^s \times 2^{m.f} \tag{2.2}$$

where s is the sign bit, m is the integer part, f is the fractional part.

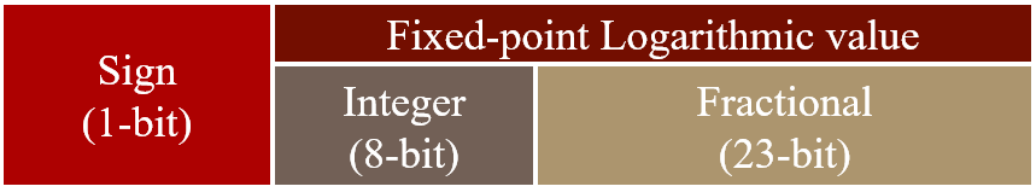


Fig. 2.6 32-bit LNS format

The LNS 2-bit dynamic range is between 2^{-128} to 2^{128}, which is near to the FLP dynamic range.

Logarithmic Number System Arithmetic

Performing computations in LNS requires converting the input data to the logarithmic domain by computing their logarithm. Assuming that there are two numbers x and y in the linear domain (real world values) and their logarithmic representations are a and b respectively, where $x \geq y$ are as follows:

$$a = \log(|x|) \tag{2.3}$$

$$b = \log(|y|) \tag{2.4}$$

The basic LNS arithmetic operations on x and y using the signed logarithmic format [36] [41] are listed as the following:

Multiplication:	$\log(x \times y) = a + b$	(2.5)
Division:	$\log(x \div y) = a - b$	(2.6)
Addition:	$\log(x + y) = b + \log(2^{a-b} + 1)$	(2.7)
Subtraction:	$\log(x - y) = b + \log(2^{a-b} - 1)$	(2.8)

As described in equation (2.5) and equation (2.6), the main advantage of the LNS arithmetic is simplifying the multiplication and division operations by converting them into addition and subtraction operations respectively. On the other hand, as shown in equation (2.7) and equation (2.8), the addition and subtraction operations became a challenge in the LNS domain. Since they require nonlinear functions that involve a logarithm and an exponent operation that make their computation is complex.

Choosing between the LNS or the FXP representation depends on the algorithm. Hence, to take the advantage of the logarithmic arithmetic, the number of the multiplication and the division operations has to be larger than the number of the addition and the subtraction operations. The reason of that is the number of addition and subtraction operations has a negative impact on the overall speed. That is because they contain additional logarithmic calculation.

In summary, the main goal of the published LNS arithmetic in the literature is to provide an alternative representation to the fixed-point to achieve high-performance hardware architectures with a better precision. In contrast, the main aim in this

context is to explore if the LNS arithmetic is used as an alternate to the FLP can improve image compression algorithms due to its data compactness feature with a precision similar to the FLP.

2.3.2 Uniform Quantization

The quantization is the main source of the quality degradation in the image compression. However, it is an essential process because it reduces the dynamic range and converts the DWT coefficients from real numbers to integers. That is because the encoder works only on integer values. The quantization process is achieved by dividing the DWT outputs by a quantization step. Using a large value of the quantization step gives a better compression ratio, but it leads to a low image quality. The quantization process is defined in equation (2.9) [42] [43] as the following:

$$X_q = sign(X)\lfloor \frac{X}{q} \rfloor \tag{2.9}$$

where X is the DWT coefficients, X_q is the quantized coefficients with a quantization step q.

2.3.3 Image Quality Assessment

Image quality assessment is an essential process to evaluate image processing algorithms. In this context, the main objective of using the quality assessment is to evaluate the compression algorithms. In the literature, there are many quality metrics that are used to evaluate the image quality. The most popular metrics are: the peak signal-to-noise ratio (PSNR) and the structural similarity index (SSIM). Both metrics requires the original image as a reference to be compared with the reconstructed image after decompression.

The PSNR between the original and the reconstructed image is defined in (2.10), it is widely used as an image quality metric. Recently, it has been shown by researchers that PSNR is not a sufficient metric to assess the image quality because it depends only on the mean square error (MSE)[44].

$$PSNR(dB) = \frac{\log_{10}(I^2)}{\sqrt{MSE}} \tag{2.10}$$

where I is the maximum pixel value in the image, and MSE is defined in equation (2.11).

$$MSE = \frac{1}{MN}\sum_{i=1}^{N}\sum_{j=1}^{M}(F_{i,j} - G_{i,j}) \tag{2.11}$$

where M, N are the image dimensions, F, G are the reference and the reconstructed image respectively, (i, j) is the pixel location in the image matrix.

On the other hand, the structural similarity index (SSIM) [45, 46] is considered as more appropriate metric for evaluating the visual performance since it measures the image in terms of the human visual system components: structural, luminance and contrast between two images. Due to these reasons, radiologists are moving towards considering the SSIM as an alternative to the PSNR metric [47] for the image quality assessment. The SSIM is defined in equation (2.12).

$$SSIM(f,g) = \frac{(2\mu_f\mu_g + C_1) + 2\sigma_f\sigma_g + C_2}{(\mu_f^2\mu_g^2 + C_1)(\sigma_f^2\sigma_g^2 + C_2)} \tag{2.12}$$

where f and g are the original and reconstructed images, respectively, μ_f , μ_g are the mean intensity values for the two images f and g respectively, σ_f, σ_g are the standard deviation of the two images and $C1$, $C2$ are constants.

2.4 Acceleration on Embedded Systems

As we described in the previous chapter, one of the main objectives of the Smart-EEG project is to build a portable medical device that can compress up to 100 frames/s. WAAVES was chosen for the project to be implemented due to its compression and image quality efficiency. Since the WAAVES HENUC encoder module was implemented by the team in LIP6 [48], the DWT module is the only targeted part in the compression algorithm that will be covered in this work.

In this section, we provide a description of the popular DWT algorithms. After that, we highlight the different embedded systems platforms that are suitable for image compression algorithms. Then, we review the related work of the DWT implementation on embedded systems.

2.4.1 Discrete Wavelet Transform

DWT is an essential stage in the compression chain in many image compression algorithms such as WAAVES and the JPEG 2000. One of the key advantages of the DWT-based compression is providing a multi-resolution transform and giving an analysis in the spatial

frequency and the location. In contrast to the Discrete Cosine Transform (DCT) which is used in JPEG, DWT provides better image quality and compression.

The 2D-DWT flow has two stages: In the first stage, the 1D-DWT is applied along the image rows to perform the horizontal DWT. That splits the image into low (L) and high (H) frequency sub-bands. In the second stage, another 1D-DWT (Vertical transform) is applied to the output of the horizontal DWT to obtain the four sub-bands (LL, LH, HL, HH) coefficients. Figure 2.7 shows the 2D-DWT computation flow. In image compression, the image requires a multi-level 2D-DWT. That is achieved by applying the 2D-DWT to the LL sub-band. That process is repeated for the required number N of 2D-DWT levels depending on the image size. An example of applying the 2D-DWT on a medical image is shown in Figure 2.8.

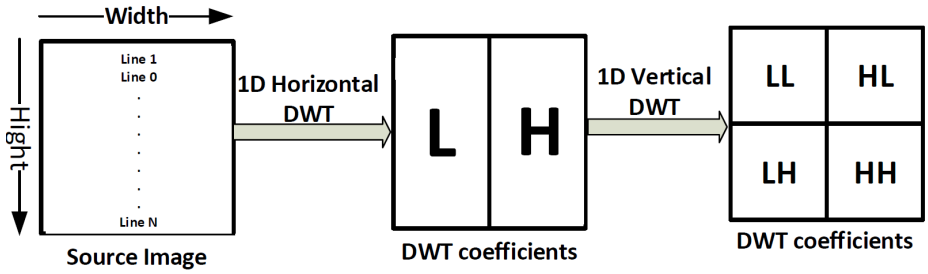
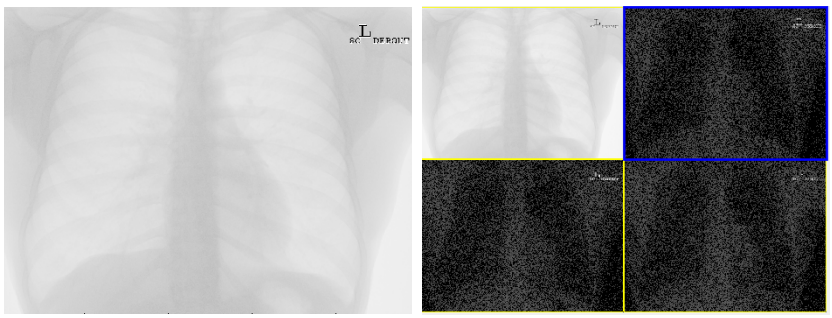


Fig. 2.7 The 2D DWT computation flow



(a) input image before DWT (b) DWT decomposition

Fig. 2.8 Example of an 2D-DWT for a medical image

There are two popular DWT algorithms: Filterbank-based DWT and Lifting scheme, we describe them in the following two sections.

Filter Bank (Convolution)

The convolution based DWT consists of two perfect-reconstruction (PR) finite impulse response (FIR) filter banks: lowpass $H(z)$ and highpass $G(z)$. They are followed by a down

sampling operation that iis applied to their outputs. That yields a low band x_l and a high band x_h for the input $x(n)$. Figure 2.9 shows a block diagram of the convolution based DWT.

The main limitation of this structure requires is the computation time. That is because there is a wasted time due to the filtering operation to all the samples, then applying the down sampling operation. The down sampling means discarding half of the samples that were computed from the previous filtering stage [49].

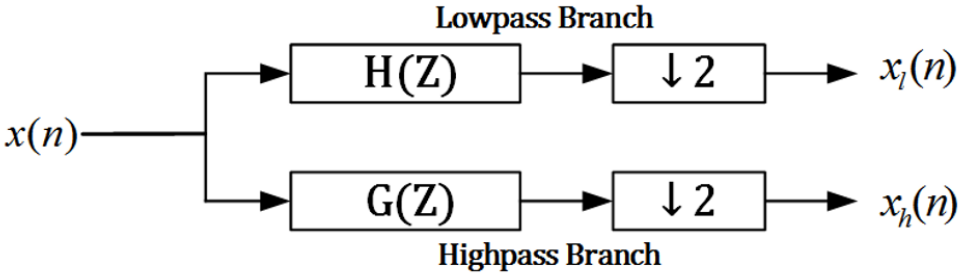


Fig. 2.9 Block diagram of convolution based DWT.

$$x_l(n) = \sum_{i=0}^{l} h(i)x(2n-i) \tag{2.13}$$

$$x_h(n) = \sum_{i=0}^{m} g(i)x(2n-i) \tag{2.14}$$

where, l and m are the lengths of the lowpass and highpass filters respectively.

To calculate the 2D-DWT, let $x(n)$ is a row in the input image. The horizontal transform is calculated by applying the lowpass $H(z)$ and highpass $G(z)$ to each row. For the vertical transform, $x(n)$ is considered the column in the output of the horizontal transform; it is passed to the two filters. To calculate the multi-level 2D-DWT, each row in the output of the lowpass filter of the vertical is considered $x(n)$ and so on.

Lifting Scheme

The second approach to compute the DWT is the lifting scheme. Its main idea is to decompose the filter bank into a sequence of lifting steps [50–52]. Looking to the polyphase filter matrix $p(z)$ in the equation (2.15) which includes the low pass filter $h(z)$ and the highpass filter $g(z)$ in the equation (2.16) and the equation (2.17) respectively. The matrix $p(z)$ is factorized into a series of upper and lower triangular matrices is called the lifting steps as described

in the equation (2.18). The described filter in the equation (2.18) is the bi-orthogonal Cohen-Daubechies-Favreau (CDF) 9/7 filter bank which is used in both lossy compression algorithms (WAAVES and JPEG2000).

$$p(z) = \begin{bmatrix} h_e(z) & h_o(z) \\ g_e(z) & g_o(z) \end{bmatrix} \tag{2.15}$$

$$h(z) = h_e(z^2) + z^{-1} h_o(z^2) \tag{2.16}$$

$$g(z) = g_e(z^2) + z^{-1} g_o(z^2) \tag{2.17}$$

where $h_e(z)$ and $g_e(z)$ are the even parts and $h_o(z)$ and $g_o(z)$ are the odd parts of the lowpass $h(z)$ and highpass $g(z)$ filters respectively.

$$p(z) = \begin{bmatrix} 1 & \alpha(1+z^{-1}) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \beta(1+z^{-1}) & 1 \end{bmatrix} \times \begin{bmatrix} 1 & \gamma(1+z^{-1}) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \delta(1+z^{-1}) & 1 \end{bmatrix} \begin{bmatrix} k & 0 \\ 0 & \frac{1}{k} \end{bmatrix} \tag{2.18}$$

where the CDF 9/7 filter coefficients values are:

$\alpha = -1.586134342 \quad \beta = -0.05298011854$

$\gamma = 0.8829110762 \quad \delta = 0.4435068522$

$K = 1.149604398$

Figure 2.10 shows the block diagram of the DWT lifting scheme. It includes four steps: (1) split, (2) predict, (3) update, and (4) scaling.

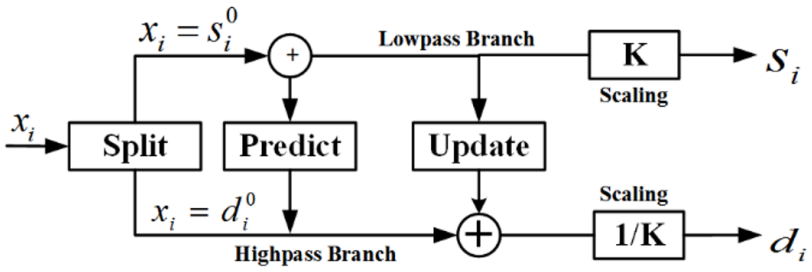


Fig. 2.10 DWT lifting Block Diagram

The detailed mathematical description of the lifting scheme algorithm for the 9/7 CDF filter is listed in the equations (2.19)-(2.26). The signal x is split into odd d_i^n and even s_i^n parts, where n is the lifting step number. Each step consists of predict and update, the last stage is a scaling step:

Spilt stage:

$$\text{Odd:} \quad d_i^0 = x_{2i+1} \tag{2.19}$$

$$\text{Even:} \quad s_i^0 = x_{2i} \tag{2.20}$$

First lifting step:

$$\text{Predict 1:} \quad d_i^1 = d_i^0 + \alpha(s_i^0 + s_{i+1}^0) \tag{2.21}$$

$$\text{Update 1:} \quad s_i^1 = s_i^0 + \beta(d_{i-1}^1 + d_i^1) \tag{2.22}$$

Second lifting step:

$$\text{Predict 2:} \quad d_i^2 = d_i^1 + \gamma(s_i^1 + s_{i+1}^1) \tag{2.23}$$

$$\text{Update 2:} \quad s_i^2 = s_i^1 + \delta(d_{i-1}^2 + d_i^2) \tag{2.24}$$

Scaling step:

$$\text{Odd scaling:} \quad d_i = \frac{d_i^2}{k} \tag{2.25}$$

$$\text{Even scaling:} \quad s_i = s_i^2 \times k \tag{2.26}$$

There are many advantages of the lifting scheme; (1) it requires less number of multiplication and addition operations than the convolution based approach; (2) the lifting scheme consumes less memory space thanks to the in-place calculation feature, that is because input data is updated and is overwritten by the output of the lifting stages; (3) the transform is a simply revisable, to calculate the inverse of the DWT, it requires only to reverse the lifting steps by performing them backwards (scaling, update, predict). [53, 51].

2.4.2 Embedded Systems Platforms

There are a variety of embedded system platforms depending on the targeted application [54]. In general, we can divide the embedded systems according to the processing speed and the development time. In general, there are three popular solutions: DSP, FPGA, and ASIC.

DSP

Digital signal processors (DSPs) are considered much faster and have higher performance than Microcontrollers. The main advantage of DSPs, they include hardwired functions that are used in the signal processing algorithms. They also, has a higher clock speed, optimized instruction set, and memory organization.

FPGAs

Field programmable gate arrays are reconfigurable hardware circuits according to the required application. The register transfer language (RTL) is used to describe the targeted algorithm. There are two commonly used RTL languages: VHDL and Verilog. Since everything is implemented on hardware, FPGAs are considered a perfect solution for high-speed processing requirements, they also require reasonable development time.

ASICs

Application-specific integrated circuit (ASIC) are circuits that are dedicated for a specific application. There is no possibility of the reprogramming or the reconfiguration after the manufacturing. Hence, it requires longer development time. However, in terms of performance, it can reach a higher speed than FPGA according to the design. RTL is used to describe the architectures.

2.4.3 Previous Implementation of the DWT on Embedded Systems

There are several existing DWT architectures in the literature. They exist in different technologies: FPGAs and ASICs. DWT architectures read the image data with one of the three scanning schemes, line-based [55, 56], block-based [57] or stripe-based [58].

In terms of control path complexity and memory requirements. Lifting-based DWT architectures are divided into three categories: folded [59, 60], parallel [61–63] and recursive [51, 64, 65]. Darji et al. [65] proposed three different architectures based on the folded multi-level architecture (FMA), pipelined multi-level architecture (PMA) and recursive multi-level architecture (RMA).

Dual-scan and clock gating techniques were used to increase the throughput and minimize the on-chip frame line buffers. Aziz and Pham [62] presented a parallel 2D-DWT multiplier-free architecture with a high operating frequency by replacing the multiplications by shift operations.

A unified FPGA-based architecture for 2D DWT was proposed by Sameen at al [66], it included a Nios II soft-core processor with attached hardware accelerators giving a high throughput and a real-time capability for gray-scale video processing with a full HD resolution.

A memory-efficient convolution-based 2D DWT architecture was introduced by Mohanty et al. [57]. Their architecture involved more multipliers than the existing ones, but it requires less memory. They synthesized their architecture for 90nm ASIC implementation. Hu and Jong [67] proposed another parallel lifting-based 2D DWT scalable architecture which requires less memory. They synthesized the design for ASIC with a 90-nm technology. Hu and Jong

[58] introduced another parallel 2D DWT architecture based on a stripe scanning method which gives high throughput and requires less temporal memory.

After the analysis of the related work, we concluded that most of the high throughput DWT architectures do not meet the requirements of the Smart EEG-project. There is no existing architecture support 100 frame/s and taking in consideration the latency of the double data rate random access memory (DDR RAM).

2.5 Discussion and Conclusion

In this chapter, we reviewed the famous image compression algorithms that are used in medical image applications. WAAVES is considered better than JPEG and JPEG2000 since it compresses images more efficiently and reconstructs them with a better quality.

We also discussed the two issues that have an influence on the image quality: the arithmetic and the quantization. For the arithmetic, floating point (FLP) and logarithmic number system (LNS) are considered better than fixed point (FXP), since they provide a high precision. LNS has a similar dynamic range to the FLP, and it also has the data compactness feature. That gives it another advantage which attracted us to investigate the possibility of utilizing it in the image compression applications.

Also, we explored the image quality assessment metrics. According to the literature, the PSNR is not sufficient to evaluate the image quality, while the SSIM is used as an alternative because it measures the visual properties in the image.

The second objective of our work is the image compression on embedded systems as a part of the Smart-EEG project. In this context, we investigate the DWT implementation on hardware since the other compression modules have implemented previously. We started by exploring the two DWT algorithms, and we found that the lifting scheme has many advantages over the convolution approach regarding its computation efficiency and memory requirements. Between the different embedded systems platforms, we concluded that the FPGA is considered as the best solution for prototyping the targeted application. That is because the development time is significantly shorter than ASIC. It is also cheaper, since it is reconfigurable, so the modification in the design does not require to fabricate the chip again.

We also provided a review of the existing solutions for accelerating the DWT on embedded systems. We found that most of the existing architectures did not consider the DDR ram latency which is the main bottleneck. Since the previous work did not meet the Smart-EEG requirements, it motivated us to investigate how to fulfill the Smart-EEG prototype requirements.

In summary, the main objective of this work is to find the answers to the following problematic research questions:

1. **Does the logarithmic representation has the ability to improve the compromise between the compression ratio and the image quality?**
2. **How to provide a new DWT hardware architecture that can fulfill the Smart-EGG high-speed requirement?**

Chapter 3

Logarithmic-Based Discrete Wavelet Transform

Contents

3.1 Introduction

As we stated in the previous chapter, preserving the diagnostic quality of the medical images is the main aim of this work. Therefore, we raised the research question; if changing the arithmetic domain can improve the quality of the reconstructed images after the compression.

In this chapter, we investigate switching the calculation from the linear to the logarithmic domain. In addition, evaluating the logarithmic arithmetic demanded that we build a custom logarithmic-based software library that we present in this chapter.

This chapter is organized as the following: In Section 3.2 we present and discuss the proposed logarithmic Library (LNS-LIB) and the mathematical issues related to the logarithmic domain. Section 3.3 discuss the LNS-LIB validation phases. Section 3.4 presents the logarithmic DWT using LNS-LIB. Section 3.5 presents the new logarithmic quantization method and provides a comparison with the linear quantization method. Section 3.6 describes the used dataset and the conducted experiments. Section 3.7 discusses the obtained results. Finally, in Section 3.8, we draw the overall conclusion for the presented work in this chapter.

3.2 LNS Library Implementation

This section present the LNS custom library we developed to ease the implementation of algorithms in LNS arithmetic. This library was built using MATLAB software, and it consist of two parts: the definition of the LNS object format (see section 3.2.1) and the functions that implement the four arithmetic operators (see section 3.2.2).

3.2.1 LNS Data Object Definition

The LNS data object is defined as a container that holds the logarithmic data. Let x be a real number in the linear domain, then, the LNS data object L is defined according to the equation (3.1), where L contains two elements: the logarithmic value field v with base B and the sign flag s.

$$\text{LNS-Object} \quad L = \{v, s\} \tag{3.1}$$

Notation of the LNS-Object Data Structure

For an LNS-object, we use dot (.) operator in the notation $(.v)$ and $(.s)$ to access the value field and the sign flag respectively. For example, the value field in the LNS-object L can be accessed using the notation $L.v$, and in the same manner for sign flag using $L.s$.

Data Handling Issues

There are two issues related to the properties of the logarithmic function $f(x) = log(x)$:

1. The logarithm of a negative number is undefined.
2. The logarithm of zero is undefined ($\log(0) = -\infty$).

These two issues must be taken into consideration when converting the data from the linear to the logarithmic domain. Therefore, we introduce two solutions to handle them: *the sign flag* and *the virtual zero*.

The Sign Flag

Regarding the first issue, since the logarithm of a negative number is undefined. Thus, we must take the logarithm of the absolute value. However, this causes a phenomena known as the sign ambiguity. This happens because the LNS domain contains two sources of a negative sign.

1. The first case: if the value of x lies in the range of $(0 < x < 1)$, then, its logarithm $(log(x))$ yields a negative value.
2. The second case: if x is a negative $(x < 0)$, then, its logarithm $(log(|x|))$ yields a negative value if it lies within the range $(-1 < x < 0)$, or a positive value otherwise.

As a consequence of the previous two cases, the negative sign in the logarithmic domain is not enough to indicate if the linear value was originally negative or not. Hence, it raised the importance of the sign flag which has the purpose to resolve the sign ambiguity. Table 3.1 lists all the cases for the sign flag of an LNS-object. As shown in the example listed in Table 3.1, the x values 0.25 and -0.25 have the same logarithmic value -1.3863. However, thanks to the sign flag, we can distinguish between them.

Table 3.1 LNS object sign cases with examples

Case	**Sign flag**	**Numerical example**	
		Linear domain x	**$v = \log(\lvert x \rvert)$**
$x \geq 1$	0	1.25	0.2231
$0 < x < 1$		0.25	-1.3863
$x = 0$		0	0
$x \leq -1$	1	-1.25	0.2231
$-1 < x < 0$		-0.25	-1.3863

The Virtual Zero

Concerning the second issue related to the fact $log(0) = -\infty$, we handled it by introducing a new feature in the LNS-Library called the **virtual zero**. We define the virtual zero as following:

"The zero value in the logarithmic domain is originally a zero value in the linear domain."

In other words, when converting $x = 0$ from the linear domain to the logarithmic domain, we keep it zero without applying the logarithm operation. Equation (3.2) describes the

virtual zero concept. We set the data field of an LNS-object to zero ($L.v = 0$) if ($x = 0$). The sign flag is described in equation (3.3).

$$L.v = \begin{cases} \log(|x|) & x \neq 0 \\ 0 & x = 0 \end{cases} \tag{3.2}$$

$$L.s = \begin{cases} 0 & x \geq 0 \\ 1 & x < 0 \end{cases} \tag{3.3}$$

The virtual zero feature gives the library two advantages: (1) image processing compatibility; since virtual zero makes the LNS library able to convert an input image containing zeros to the logarithmic domain. (2) optimizing the logarithmic arithmetic operations; since they resembles the linear domain when handling the zeros (see in Section 3.2.2).

Logarithmic to Linear Conversion

To convert back an LNS-object L to its linear representation x, we use the exponent function to calculate the inverse of the logarithm as defined in equation (3.4). The virtual zero and the sign flag $L.s$ are handled in the conversion operation.

$$x = \begin{cases} -1^{L.s} \times B^{L.v} & L.v \neq 0 \\ 0 & L.v = 0 \end{cases} \tag{3.4}$$

The library also handles the true zero, which occurs when $log_2(x) = 0$ with $x = 1$. It simply replaces the $x = 1$ with the nearest value. For example X is replaced with 1.00000000001, therefor, $Log_2(1.00000000001) = 1.4427 \times 10^{-11}$ which is a very small value that nearly equal zero. The reason for using this is to distinguish between the virtual and the true zero. The library gives a priority to the virtual zero since it was oriented for image processing applications.

3.2.2 LNS Arithmetic Operators

The second part of the LNS library is the arithmetic operators. The basic four operations (multiplication, division, addition, and subtraction) were implemented as the following:

LNS Multiplication and Division

The function *LNS_MulDiv* as shown in equation (3.5) performs the multiplication and division for $A = \{v, s\}$ and $B = \{v, s\}$ respectively. It returns the result into an LNS object $C = \{v, s\}$. The function is based on the equation (2.5) and the equation (2.6) respectively presented in Chapter 2.

$$C = \text{LNS\_MulDiv}(A, B, MD) \tag{3.5}$$

The function accepts three input parameters: the two LNS-objects and a 1-bit flag parameter MD. The value of MD defines the type of the operation. If $MD = 0$, then a multiplication operation will be performed. Otherwise, if $MD = 1$, then a division operation is performed. The function output has three possible results as indicated in equation (3.6). The output can be zero if the value of one of the inputs or both of them are equal to zero. Hence, the virtual zero gives us the advantage of treating these operations in a way similar to that in the linear domain.

$$C.v = \begin{cases} 0 & A.v = 0 \quad \text{or} \quad B.v = 0 \\ A.v + B.v & MD = 0 \\ A.v - B.v & MD = 1 \end{cases} \tag{3.6}$$

Regarding the sign flag as shown in equation (3.7), the output sign remains the same if both inputs have the same sign. On the other hand, the output sign flag is set to 1 if the inputs have a different sign, which is similar to the linear multiplication/division.

$$C.s = \begin{cases} 0 & A.v = 0 \quad \text{or} \quad B.v = 0 \\ 0 & A.s = B.s \\ 1 & A.s \neq B.s \end{cases} \tag{3.7}$$

LNS Addition and Subtraction

The function *LNS_Add* performs the addition and the subtraction operations for two LNS-objects A and B respectively. It returns the result into an LNS-object C as shown in equation (3.8).

$$C = \text{LNS\_ADD}(A, B) \tag{3.8}$$

The value of the fields $C.v$ and sign flag $C.s$ of resulted LNS-object C are defined in equation (3.9) and equation (3.10) respectively. These operations are based on equation (2.7) and equation (2.8) that were mentioned in the previous chapter.

$$C.v = \begin{cases} A.v + f\_add & A.s = B.s \\ A.v + f\_sub & A.s \neq B.s \\ A.v & B.v = 0 \\ B.v & A.v = 0 \\ 0 & A.v = B.v \quad \text{and} \quad A.s \neq B.s \\ 0 & A.v = B.v = 0 \end{cases} \tag{3.9}$$

$$C.s = \begin{cases} A.s & A \geq B \\ B.s & A < B \end{cases} \tag{3.10}$$

The nonlinear functions f_add and f_sub are defined in equation (3.11) and equation (3.12) respectively.

$$f\_add = \log(1 + 2^{B.v - A.v}) \qquad \text{where} \qquad A > B \tag{3.11}$$

$$f\_sub = \log(1 - 2^{B.v - A.v}) \qquad \text{where} \qquad A > B \tag{3.12}$$

There are five possible result values for $C.v$ depending on the input values of A and B. The results are calculated using the similar principles of the addition in the linear domain. The addition is performed using the function in equation (3.11) if the sign fields of both inputs are the same. Otherwise, if the sign fields of both inputs are different, the subtraction is performed using the function in equation (3.12).

On the other hand, there are two special cases related to the virtual zero. The first case occurs when one of both inputs is virtual zero. Therefore the result is the non-zero input. The second case occurs when both inputs are virtual zeros; then the result is a virtual zero. In summary, if the inputs contain a virtual zero, the operator becomes an assignment, this limits the overhead of the addition and subtraction operations that resulted from the nonlinear functions f_add and f_sub. That is an interesting aspect of the library because introducing the virtual zero decreases the computation time. In other words, having more zeros leads to less addition/subtraction operations.

Example 3.1 gives a numerical example of how the functions are used to calculate the addition/multiplication.

▼ **Example 3.1.**

Let $x = 3.5$ *and* $y = -2.5$*, Then, their addition is:*
$z = x + y = 1$*, and their multiplication is:*
$w = x \times y = -8.75$.
To perform the same operations in LNS, first we convert x *and* y *to LNS-objects* x_l *and* y_l*, as the following:* $x_l = \{v, s\} = \{1.8073, 0\}$*, and* $y_l = \{v, s\} = \{1.3219, 1\}$ *then, the addition of the two number in LNS is:*
$z_l = LNS\_add(x_l, y_l)$
$z_l = \{v, s\} = \{-4.44089 \times 10^{-16}, 0\}$*, which yields* 1 *when converting back* z_l *to the linear domain and the multiplication of the two number in LNS is:*
$w_l = LNS\_MulDiv(x_l, y_l, 0)$
$w_l = \{v, s\} = \{3.1292, 0\}$ *which yields* -8.75 *when converting back* z_l *to the linear domain.*

3.3 Library Validation

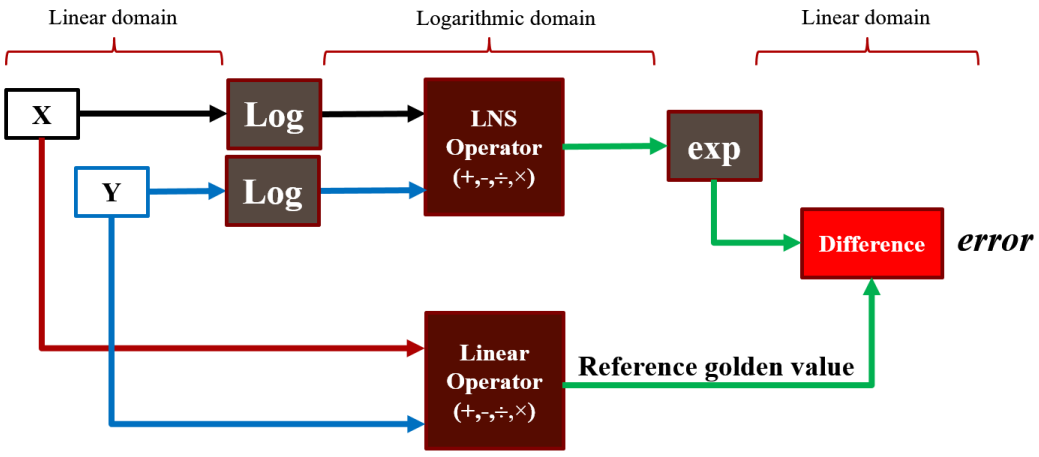


Fig. 3.1 The validation methodology of LNS-Library operators

The LNS library was validated by performing a set of verification test phases. The first test phase objective was to validate the four basic mathematical operations (addition, subtraction,

multiplication, and division). The operations were tested in the logarithmic domain. Then their results were converted back to the linear domain. These converted results are compared to golden reference values. A description of the first phase of the validation is shown in Figure 3.1.

The second validation process was done by performing the multiply and accumulate (MAC) operation. The MAC operation is described in equation (3.13). The MAC was chosen because it combines two operations, which is similar to the calculation behavior in many image processing algorithms. The purpose of this phase was to measure the error ϵ between the linear M_{lin} and the logarithmic domain M_{lns} for different n iterations of MAC as described in equation (3.14).

$$M = \sum_{i=0}^{n} a \times i \tag{3.13}$$

where, M is the total of multiplying i by a constant a.

$$\epsilon = |M_{lin} - M_{lns}| \tag{3.14}$$

where, ϵ is the absolute error, M_{lin} is the MAC output in the linear domain, and M_{lns} is the MAC output in the logarithmic domain.

Figure 3.2 shows the absolute error for different MAC iterations with different constants. The constants of the lifting DWT 9/7 CDF filters were chosen in the MAC experiment since the main aim of the LNS Library is to be used for the DWT implementation. The results show that the error was increased when calculating an enormous number of iterations. However, the error was still slightly small, since it reached around 7×10^{-8} at 1000 iterations. In conclusion, this evaluation indicates that the library is reliable regarding the accumulated error.

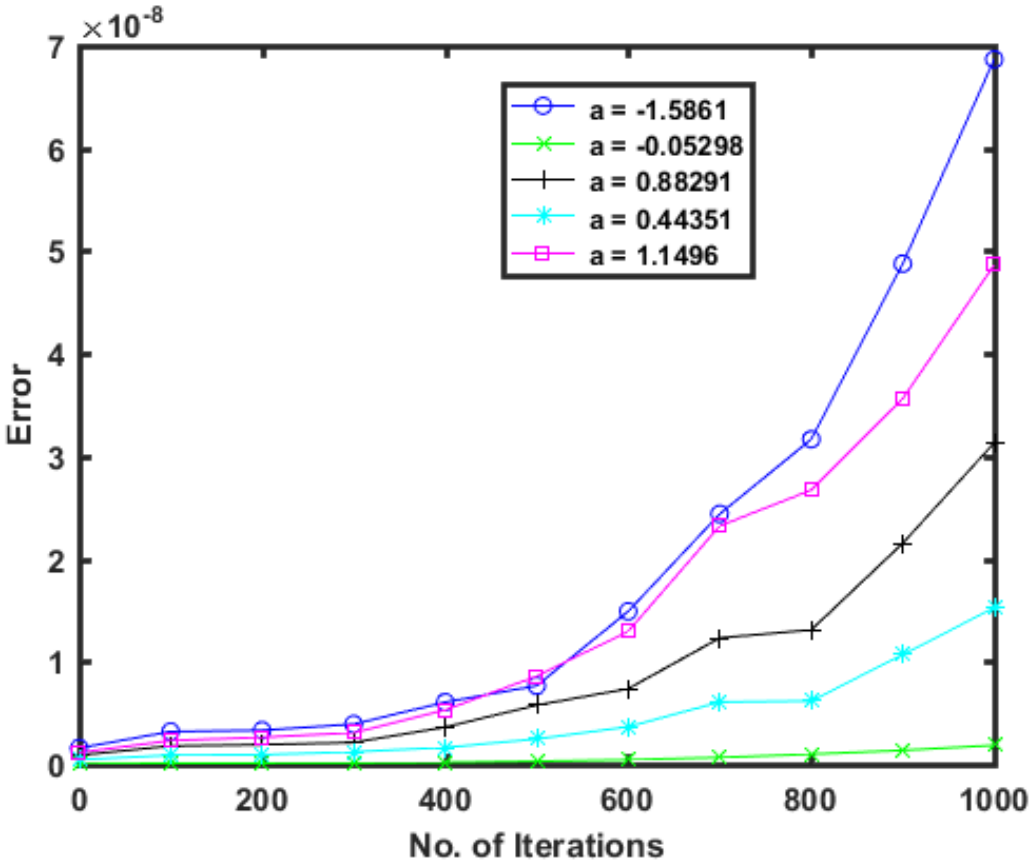


Fig. 3.2 Error between linear and logarithmic for MAC using different constants

3.4 Discrete Wavelet Transform using The LNS Library

The discrete wavelet transform (DWT) is considered as the backbone of modern image compression algorithms. The DWT performs the image processing part as a first phase in the compression algorithm before the coding phase. After the first validation of LNS library, this section investigates the DWT implementation using the logarithmic arithmetic as an alternative to the linear arithmetic.

According to the state of the art, there are two approaches for DWT implementation: convolution based and lifting based approaches. In this work, the DWT lifting scheme has been chosen because of its efficiency in terms of computation and memory requirements [53]. We selected the DWT-based 9/7 CDF filter to be implemented in LNS since it is used in WAAVES and JPE2000 [16].

The 2D-DWT implementation in the logarithmic domain consists of two basic steps as listed in Algorithm 1: The first step is to convert the input image from the linear to the logarithmic domain. After that, we calculate the 1D-LNS-DWT for each line in the

logarithmic image, yielding the horizontal DWT-coefficients matrix. Finally, we calculate the vertical transform for each line in horizontal DWT-coefficients matrix.

Algorithm 1: 2D LNS-DWT

Input : A logarithmic image LI, Width, High
Output : 2D LNS-DWT Coefficients V

```
for L = 0 to Hight-1 do // 1D DWT horizontal transform for each image line
    H(L) = 1D LNS-DWT of LI(L)
end
for C = 0 to Width-1 do          // 1D vertical transform for each column
    V(C) = 1D LNS-DWT of H(C)
end
```

The 1D-DWT transform was implemented using LNS library based on the equations from (2.19) to (2.26) as presented in Chapter 2. Its implementation is divided into six parts which are described in the algorithms from Algorithm 2 to Algorithm 8. The first stage in the 1D-DWT is the Predict-1 which works on the even elements in the input line X. For each odd element $X\_current$, the previous element X_prev and the next element X_next are added then multiplied by the constant α. Then, the multiplication output is added to that $X\_current$ yielding the Predict-1 output $P1$ as described in equation (3.15):

$$P1 = X\_current + alpha \times (X\_prev + X\_next) \tag{3.15}$$

The next stage is the Update-1 which is similar to the Predict-1, it works on the odd elements on the input line X and the output of the Update-1. The next stage of the computation includes the Predict -2 followed by the Updated-2 stage. Both are calculated in the same manner like Update -1 and Predict-2 but with changing the filter constants. The output of both previous stages is scaled, then packed together to form the final output of the 1D-DWT transform.

Algorithm 2: 1D-DWT LNS : Part 1 - Lifting stage Predict 1

```
Input  : X , N                                          // one line of an image
Output : Y                                                 // Y = 1D_DWT_LNS(X)
// Where X is an input vector LNS object type with a length N
// Predict 1:  X_next = X_current + alpha*(X_prev + X_next)
for i = 1 to N-1 step 2 do                                          // odd values
    ADD = Add_LNS(X(i-1) , X(i+1) )
    MUL = Mul_LNS(alpha, ADD)
    P1(i) = Add_LNS(X_current , MUL)
end
```

Algorithm 3: 1D-DWT LNS: Part 2 - Lifting stage Update 1

```
Input  : X , N                                          // one line of an image
Output : Y                                                 // Y = 1D_DWT_LNS(X)
// Update 1:  X_next = X_current + Beta *(X_prev + X_next)
for i = 0 to N-1 step 2 do                                         // even values
    ADD = Add_LNS(P1(i-1), P1(i+1))
    MUL = Mul_LNS(Beta, ADD)
    U1(i) = Add_LNS(P1(i) , MUL_1)
end
```

Algorithm 4: 1D-DWT LNS: Part 3 - Lifting stage Predict 2

```
Input  : X , N                                          // one line of an image
Output : Y                                                 // Y = 1D_DWT_LNS(X)
// Predict 2:  X_next = X_current + Delta *(X_prev + X_next)
for i = 1 to N-1 step 2 do                                          // odd values
    ADD = Add_LNS(U1(i-1), U1(i+1))
    MUL = Mul_LNS(Delta, ADD)
    P2(i) = Add_LNS(P1(i) , MUL)
end
```

Algorithm 5: 1D-DWT LNS: Part 4 - Lifting stage Update 2

```
   Input  : X , N                                          // one line of an image
   Output : Y                                                 // Y = 1D_DWT_LNS(X)
   // Update 2:  X_next = X_current + Gama *(X_prev + X_next)
1  for i = 0 to N-1 step 2 do                                          // even values
2  |  ADD = Add_LNS(P2(i-1) , P2(i+1))
3  |  MUL = Mul_LNS(Gama, ADD)
4  |  U2(i) = Add_LNS(U1_current , MUL_3)
5  end
```

Algorithm 6: 1D-DWT LNS: Part 5-A - Lifting stage Scaling odd elements

```
   Input  : X , N                                          // one line of an image
   Output : Y                                                 // Y = 1D_DWT_LNS(X)
   // Lifting Scale :
1  for i = 1 to N-1 step 2 do                                           // odd values
2  |  P2S(i) = Mul_LNS(P2(i), Z0)
3  end
```

Algorithm 7: 1D-DWT LNS: Part 5-B - Lifting Stage Scaling even elements

```
   Input  : X , N                                          // one line of an image
   Output : Y                                                 // Y = 1D_DWT_LNS(X)
1  for i = 1 to N-1 step 2 do                                          // even values
2  |  U2S(i) = Mul_LNS(U2(i), Z1)
3  end
```

Algorithm 8: 1D-DWT LNS: Part 6 - Packing Odd/Even Elements

```
   Input  : X , N                                  // one line of an image
   Output : Y                                        // Y = 1D_DWT_LNS(X)
   // Reorder values :
1  for i = 0 to N-1 step 1 do
2      if I < N/2 then
3          Y(i) = P2S(i)
4      else
5          Y(i)= U2S(N/2-i)
6      end
7  end
```

LNS-DWT Validation

Finally, the LNS-DWT validation was done by comparing the output of DWT in the logarithmic domain with the linear DWT. The absolute error was measured in the same manner as performed in the second validation phase. Figure 3.3 shows the absolute error between two 1D-DWT outputs, linear and logarithmic. The input was a vector of 1024 values with a 16-bit dynamic range that simulates a single line in an image with width 1024. The error was around 7×10^{-10} which is considered a slight value.

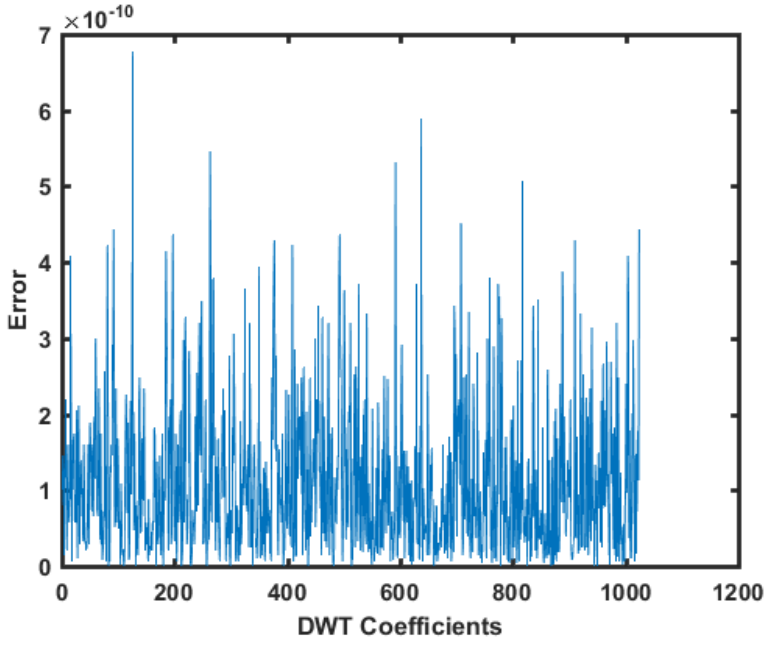


Fig. 3.3 Error between linear and logarithmic 1D-DWT

3.5 LNS Quantization

Developing and validating the LNS library were the first steps in this research. In this section, we present the quantization process which is an essential stage in the compression algorithms. The main objective of the quantization operation is converting the real number data into integer values which are required by the encoders. Also, the quantization helps to decrease the dynamic range of the data to be compressed efficiently. In the linear quantization, the input data are divided by the quantization step, then followed by the rounding operation as described in equation (2.9) in Chapter 2.

On the other hand, the quantization in the logarithmic domain is different to that in the linear domain. Since the division operation in the linear quantization became a subtraction in the logarithmic domain which shifts the logarithmic values, it did not convert them the data into discrete values. Therefore in this work, we propose a novel quantization method called LNS-Q. It is based on scaling the logarithmic values, followed by rounding the scaled logarithmic values to have integer values. Equation (3.16) describes the LNS-Q process of an un-quantized data X scaled by a scale factor SC.

$$X_{LNS-Q} = round(X \times SC) \tag{3.16}$$

The main idea of the scaling operation is introducing the flexibility to control the desired precision. Hence, the key advantage of the LNS-Q method is preserving the precision of the quantized data depending on the chosen scale factor. The scale factor SC is a function of the number of the fractional digits nf to be kept after the decimal point as a result of the quantization process. SC is described in equation (3.17).

$$SC = 10^{nf} \qquad nf \geq 0 \tag{3.17}$$

The scale factor SC value is a multiple of 10 starting from $\{10^0, 10^1, 10^2, etc.\} = \{1, 10, 100, etc.\}$. Keeping the fractional part is important because the number of digits in the fractional part nf has a significant effect on the accuracy when switching back to the linear domain.

Quantization Error

Since the Quantization is the source of the quality degradation in the compression algorithms, it is important to study the quantization error. The quantization error in the linear domain ϵ_{lin} is defined as the difference between the original input signal X and the inverse quantized signal X_{iq} with a quantization step q as shown in equation (3.18).

$$\epsilon_{lin} = X - X_{iq} \tag{3.18}$$

where $X_{iq} = X_q \times q$.

The quantization error in the logarithmic domain ϵ_{log} is defined as the difference between the original input signal X and the inverse of the quantized signal X_{LNS-iQ} as shown in equation (3.19).

$$\epsilon_{log} = X - 2^{X_{LNS-iQ}} \tag{3.19}$$

where X_{LNS-iQ} is the inverse quantized signal of X_{LNS-Q} with a scaling factor SC as shown in equation (3.20).

$$X_{LNS-iQ} = \frac{X_{LNS-Q}}{SC} \tag{3.20}$$

To evaluate the effect of replacing the linear quantization with LNS-Q, we give a comparison example between both of them in Figure 3.4. In this example, we assume a vector of 10 DWT-coefficients pixels, which is considered as the original signal input. This input was quantized with both the linear and the LNS-Q quantization approaches. The quantized values were reconstructed again by applying the inverse of the quantization operation.

Figure 3.4, the pixels that were quantized with the LNS-Q are reconstructed better than when they were quantized with the linear quantization. Figure 3.5 shows the quantization error for both linear and LNS-Q quantization. It shows that LNS-Q quantization error fluctuates between 2 and -2 while using the linear quantization yields a quantization error up to 10. The example shows how the scaling preserves the precision compared to the linear quantization.

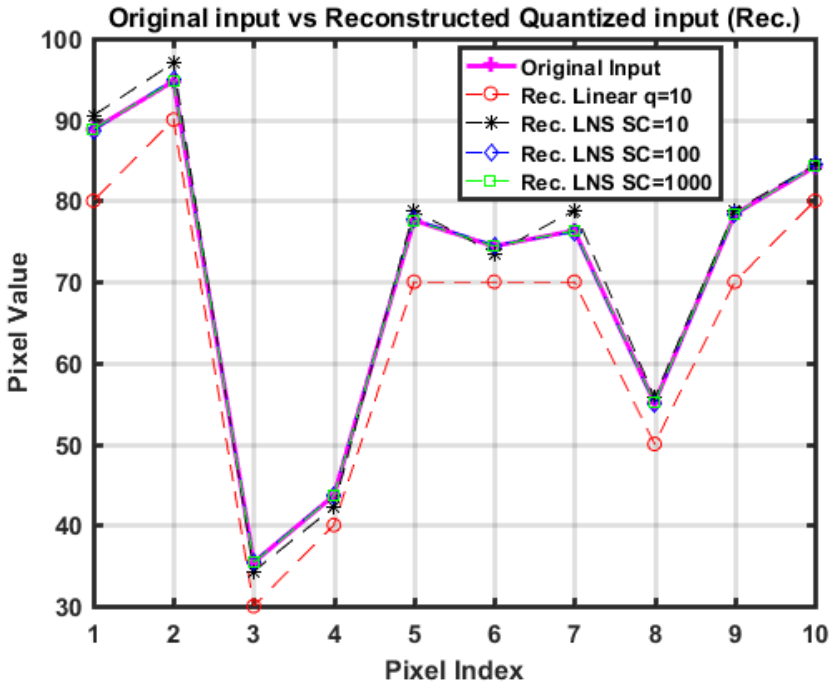


Fig. 3.4 Comparison example between LNS-Q and Linear quantization

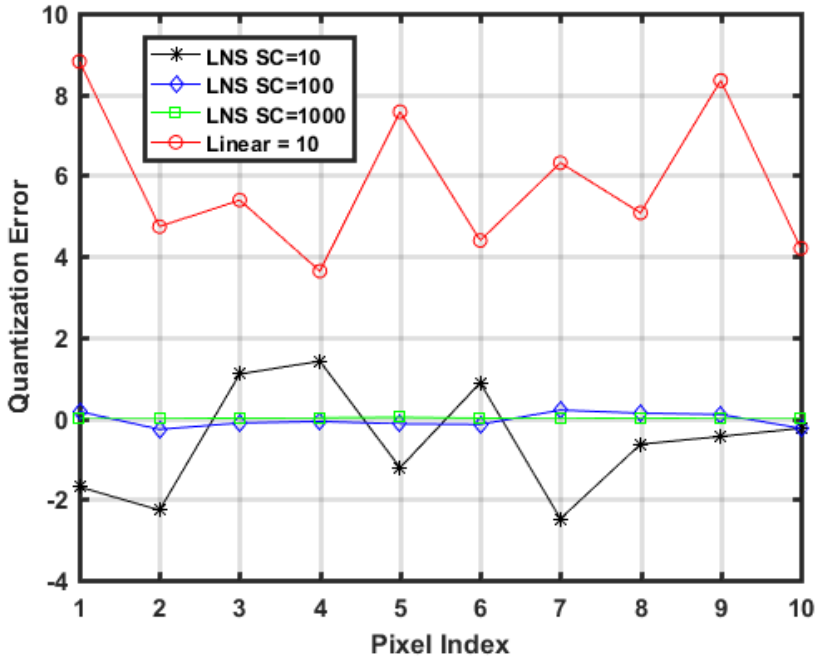


Fig. 3.5 Comparison between the quantization error for LNS-Q and Linear quantization

The previous results were confirmed when we measured the PSNR as shown in Figure 3.6. The quantization with LNS-Q gives higher PNSR than using the linear quantization. The PNSR reaches 76.88 when applying LNS-Q with $SC = 1000$, while it reaches 23.73 when using the linear quantization with q=10.

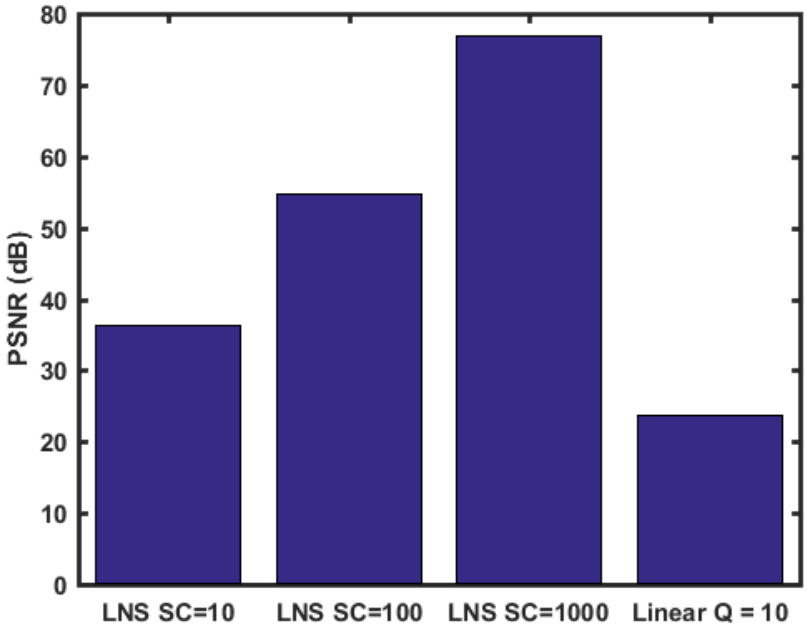


Fig. 3.6 Comparison between the PSNR for LNS-Q and Linear quantization

3.6 Experiments Methodology and Dataset

In the previous section, we presented the LNS-Q method and proved that the quantization error using this method was small compared to the linear quantization method. In this section, we describe the evaluation methodology that was used to assess the image quality using the LNS-DWT with LNS-Q compared to the linear DWT. All the experiments have been conducted using 16-bit medical images in the DICOM standard format that were provided by the hospital *Hôpital E uropéen Georges-Pompidou* (HEGP) [68]. Different image sizes were used in the experiments as listed in Table 3.2. The images are shown in the figures from Figure 3.7 to Figure 3.11. The different tested image modalities are listed as the following:

Tested Medical Image Modalities

1. Mammography (MG)
2. Computer Tomography (CT)
3. Computed radiography (CR)

4. Magnetic resonance imaging (MRI)
5. X-Ray Angiography (XA)
6. Digital Subtraction Angiography (DS)

Table 3.2 List of the tested sample dataset

Modality	**Image size**	**Bits per pixel**	**File size KB**	**File name**
MG	4708×5844	16-bit	53,773	MG-1.dcm
CT	512×512	16-bit	517	CT_IM-0001-0001.dcm
DS	1024×1024	16-bit	2,053	Medic_Anap_01.dcm
XA	1024×1024	16-bit	2,054	XA_IM-0001-0001.dcm
	1024×1024	16-bit	2,053	Medic_Arterio_01.dcm
	1024×1024	16-bit	2,053	Medic_Arterio_02.dcm
	1024×1024	16-bit	2,053	Medic_Arterio_03.dcm
CR	1976×1576	16-bit	6,290	Medic_Hand_01_CR.dcm
	1976×1576	16-bit	6,290	Medic_Dent.dcm
	2494×2048	16-bit	9,982	Medic_Chest_01.dcm
MRI	512×512	16-bit	522	Medic_IRM_01.dcm
	512×512	16-bit	525	MRI_IM-0001-0002.dcm

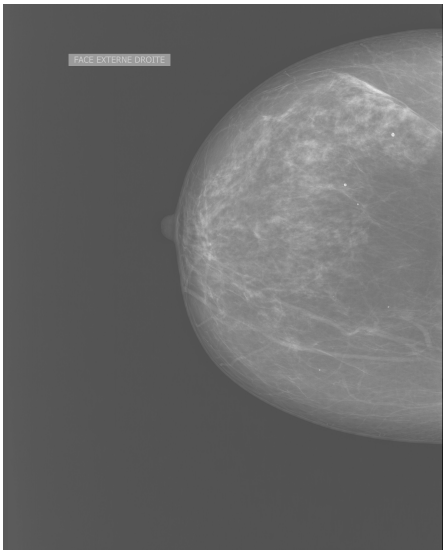
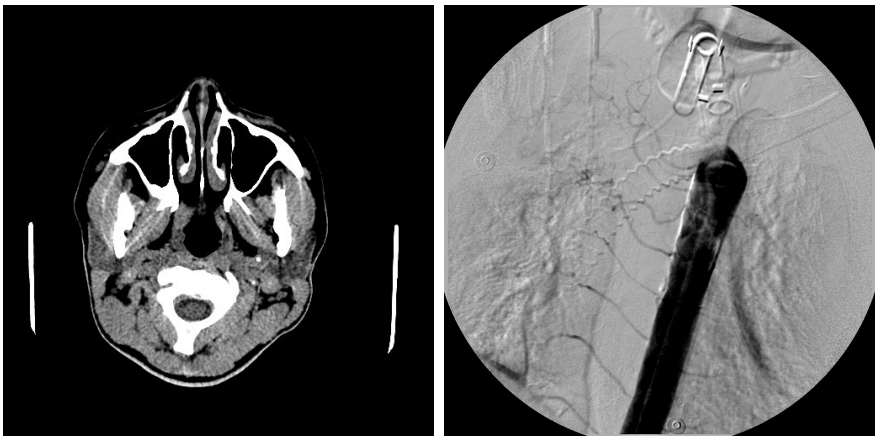


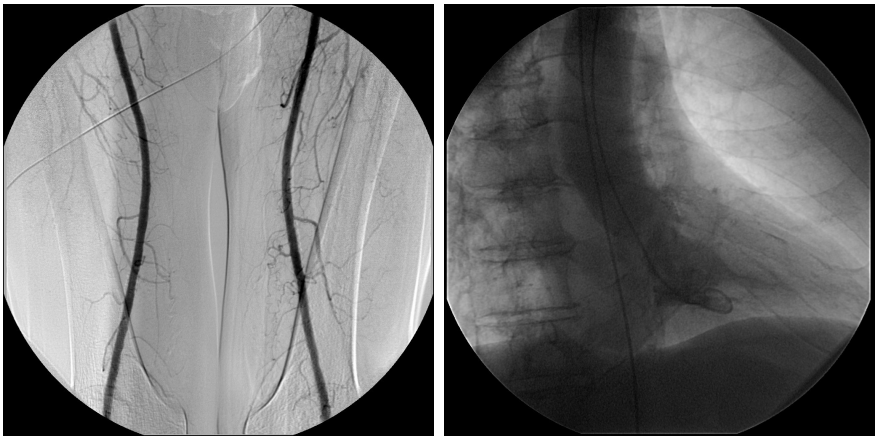
Fig. 3.7 Mammography Modality : MG-1.dcm



(a) CT_IM_0001_0001.dcm (b) DS: Medic_Anap_01.dcm

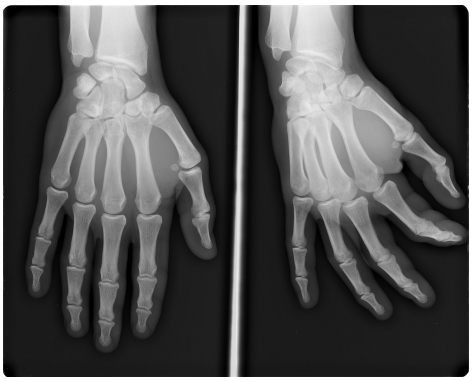
Fig. 3.8 Computer Tomography (CT) and Digital Subtraction Angiography(DS)

(a) IM-0001-0001.dcm (b) Medic_Arterio_01.dcm

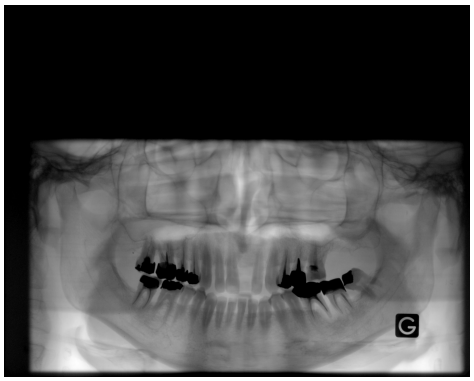


(c) XA: Medic_Arterio_02 (d) XA: Medic_Arterio_03

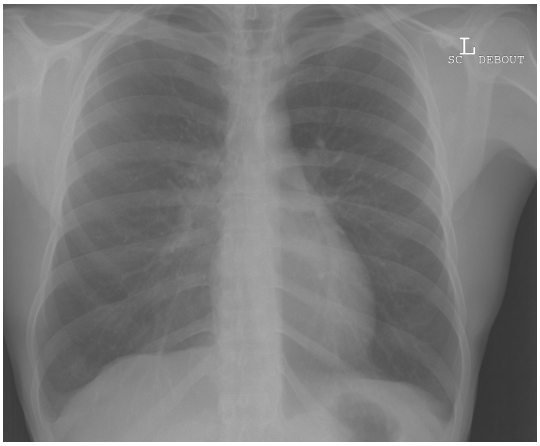
Fig. 3.9 X-Ray Angiography (XA)Modality



(a) Medic_Hand_01_CR.dcm

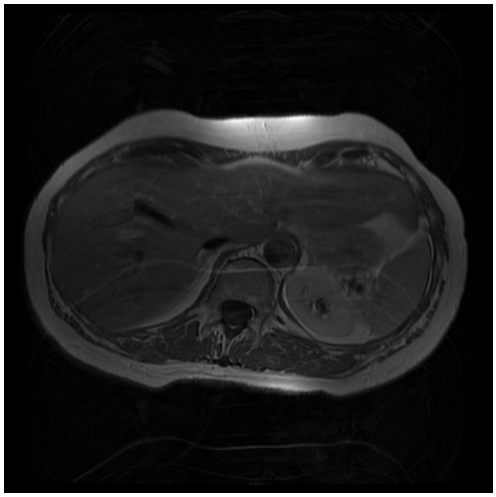


(b) Medic_Dent.dcm

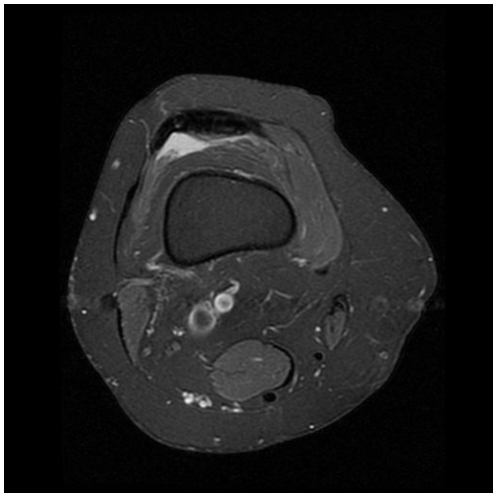


(c) CR: Medic_Chest_01

Fig. 3.10 Computed radiography (CR) Modality



(a) Medic_IRM_01.dcm



(b) MRI_IM-0001-0002.dcm

Fig. 3.11 Magnetic resonance imaging (MRI) Modality

The methodology of the conducted experiments is summarized as the following:

- Reading the DICOM image, and extracting the raw data R from it, then calculating its logarithm with a logarithmic-base b:

$$R_{log} = log_b(R) \tag{3.21}$$

- Calculating the LNS-DWT D for the raw data:

$$D = DWT(R_{log}) \tag{3.22}$$

- Quantizing the LNS-DWT Coefficients using LNS-Q with a scale factor SC:

$$D_{LNS-Q} = LNS\_Q(D, SC) \tag{3.23}$$

- Calculating the inverse quantization of D_{LNS-Q} using the scale factor SC:

$$D_{LNS-iQ} = LNS\_iQ(D_{LNS-Q}, SC) \tag{3.24}$$

- Reconstructing back the raw image by Calculating the inverse of the LNS-DWT for the inverse-quantized R_{rec}:

$$R_{rec} = IDWT(D_{LNS-iQ}) \tag{3.25}$$

- Calculating the exponent of R_{rec} to obtain the linear reconstructed image $\bar{R}$:

$$\bar{R} = b^{R_{rec}} \tag{3.26}$$

- Measuring the quality index (SSIM) between the original image and the reconstructed image: $SSIM(R, \bar{R})$ and $PSNR(R, \bar{R})$

Figure 3.12 shows the block diagram of LNS-DWT and LNS-Q experiments-setup methodology.

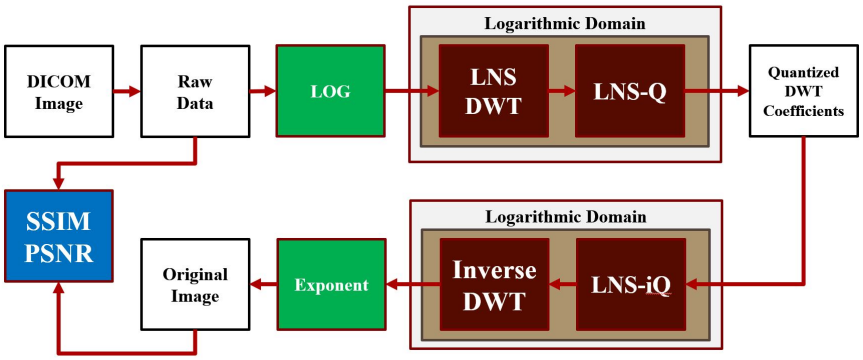
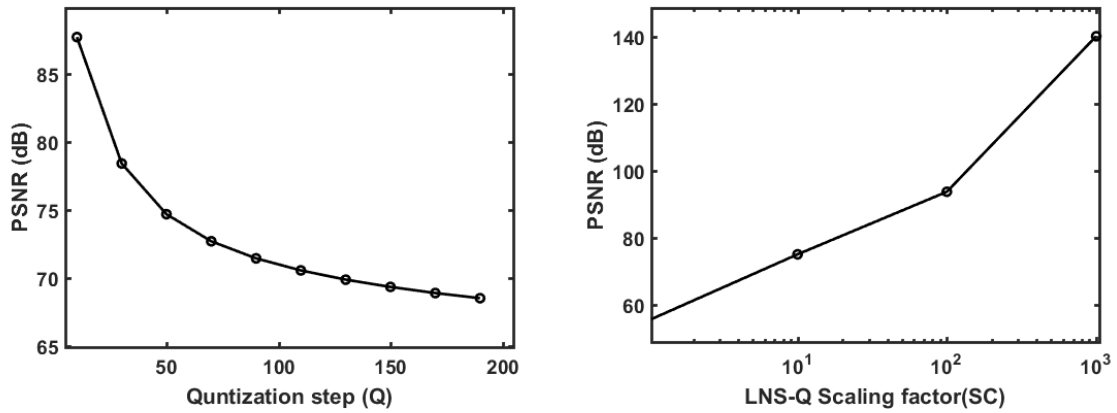


Fig. 3.12 LNS-DWT experiments methodology and setup

3.7 Results and Discussion

The experiments were conducted based on the methodology that we described in the previous section. The primary objective of those experiments is to study the effects of applying the LNS-DWT combined with the new LNS-Quantization method. Thus, we measured the SSIM and the PSNR after reconstructing the input image for both quantization approaches.

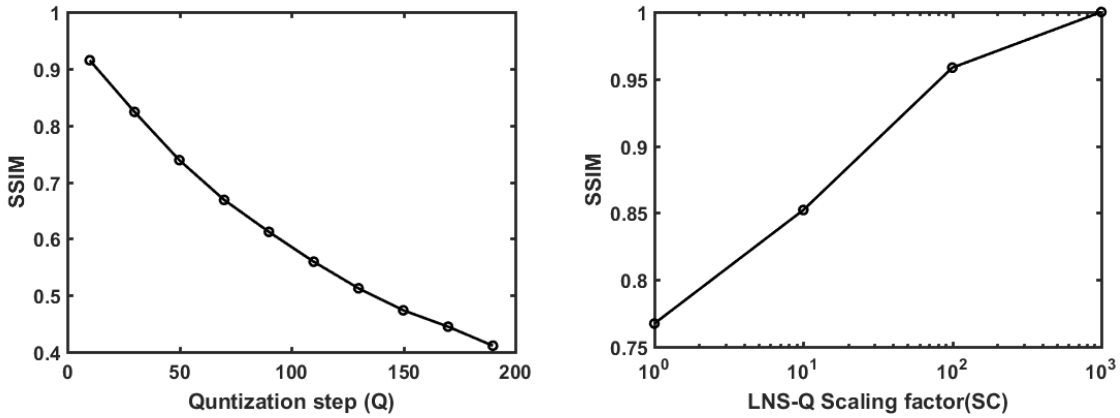
Figure 3.13 shows the PSNR for both Linear quantization and LNS-Q respectively. The results in Figure 3.13a indicate that the PSNR decreases with higher quantization steps. That is evident, because of the quantization error increases with increasing the quantization step. On the other hand, Figure 3.13b shows the PSNR reaches up to 140.3 dB with a higher scale factor $SC = 1000$, while the maximum PSNR using the linear approach reaches up to 87.7 dB with $Q = 10$.



(a) Linear Quantization (b) LNS-Q

Fig. 3.13 Comparison between the PSNR for Linear Quantization and LNS-Q

Figure 3.14 shows the SSIM for the two quantization approaches. The SSIM reaches to 0.99999887 using the LNS-Q with $SC = 1000$. On the other hand, SSIM reaches around 0.91483822 using the linear qunatization with $Q = 10$ as shown in Figure 3.14b.



(a) Linear Quantization (b) LNS-Q

Fig. 3.14 Comparison between the SSIM for Linear Quantization and LNS-Q

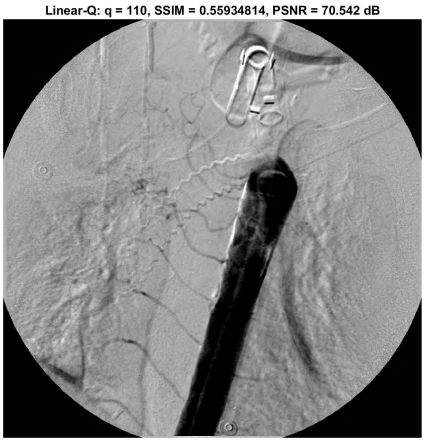
In summary, the LNS-Q approach gives a very high SSIM and PSNR when scaling with scale factor $SC = 100$ or $SC = 1000$. That is because the LNS-Q preserve the precision thanks to the scaling operation. Also, it decreases the quantization error significantly. Consequently, it improves the reconstructed image quality.

Figure 3.16 shows the effects of the linear quantization with different q steps for the input image shown in Figure 3.15. The figure show the reconstructed images after performing the DWT followed by the quantization process. The impact of larger quantization step on the image quality is clear. It introduces many artifacts in the whole image due to the quantization error.

Fig. 3.15 Input image

(a) Linear Quantization with q = 10

(b) Linear Quantization with q = 50



(c) Linear Quantization with q = 110

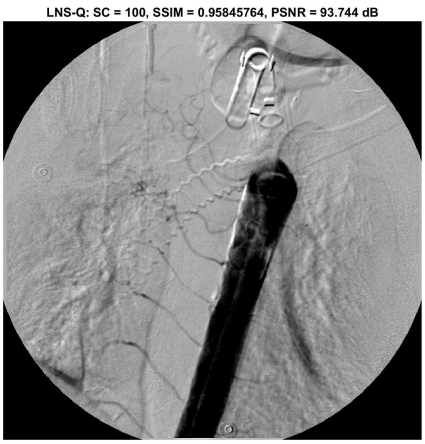
(d) Linear Quantization with q = 190

Fig. 3.16 Comparison between the visual effects using linear quantization with different q steps.

On the other hand, Figure 3.17 shows the effects of the LNS-Q on the reconstructed image using different SC scale factors. The quality is almost similar to the original image with $SC \geq 10$.

(a) LNS-Q with SC = 1

(b) LNS-Q with SC = 10



(c) LNS-Q with SC = 100

(d) LNS-Q with SC = 1000

Fig. 3.17 Comparison between the visual effects with LNS-Q with different SC.

3.8 Conclusion

The aim of this chapter was to find the answer to the research question related to the impact of changing the arithmetic domain. Therefore, the need of a tool that fulfills our study requirement was essential.

We developed an LNS-Library as the first stage in our methodology. The proposed LNS-Library introduced a new feature called the virtual zero. That gave the library the advantages of the both domains. The library also addressed the sign ambiguity problem in the logarithmic domain, and solved it by introducing the sign flag. The library was validated on many phases, starting from verifying the arithmetic operators, then the algorithmic level.

Following the first validation of the LNS library, we presented the LNS-DWT implementation. We validated LNS-DWT by comparing it with the linear DWT. The absolute error between the two domains was slight, roughly 7×10^{-10}.

After that, we presented a novel quantization method in the logarithmic domain (LNS-Q).

We compare it with the linear quantization. The LNS-Q is based on scaling the data, unlike the linear quantization which uses the division. A comparison was made between the linear and the logarithmic quantization. The comparison showed that the LNS-Q yields a quantization error nearly equal to zero with keeping only two digits in the fractional part (SC = 100).

The LNS-DWT and the LNS-Q were applied on medical images to evaluate their impact on the image quality. PSNR and SSIM were used to measure the quality. The experimental results showed that LNS-Q achieves better quality for the reconstructed image.

Using LNS-Q with SC = 1000 yielded a PSNR up to 140.3 dB and an SSIM = 0.99999887. In contrast, the linear quantization with q = 10 yielded a maximum PSNR equal to 87.7 dB and an SSIM = 0.91483822.

In summary, the results showed that the LNS-Q using scaling has a significant impact on the image quality due to the small quantization error compared to the linear solution.

In the next chapter, we investigate the integration of the LNS-DWT and LNS-Q with a full compression scheme to study their impact on the compression efficiency.

Chapter 4

Logarithmic Based Image Compression

Contents

4.1 Introduction

In this chapter, we investigate the integration of the LNS-DWT and the LNS-Q with the full compression chain. It is organized as the following: In Section 4.2 we present LNS-WAAVES as a new compression scheme that is based on LNS-DWT and LNS-Q. In Section 4.3, we provide the results of evaluating the new compression scheme. Also, we provide a comparison between the proposed compression scheme and the existing compression algorithms. Finally, in Section 4.4, we draw a conclusion on the impact of applying the LNS arithmetic on the image compression algorithms.

4.2 LNS-WAAVES

As we mentioned in Chapter 2, WAAVES provides a better compression ratio and a better-compressed image quality than JPEG2000. In this section, we explore how to provide further by introducing the logarithmic domain.

4.2.1 Integration of LNS-DWT to WAAVES

Figure 4.1 shows the full compression chain of the modified WAAVES. The main idea of the LNS-WAAVES is to replace the linear DWT and the linear quantization method with the LNS-DWT followed by LNS-Q quantization method. The new compression steps are: the first step is to convert the input image to the logarithmic domain; then, the LNS-DWT is applied to generate the DWT coefficients, which are quantized using the LNS-Q method; finally, the encoder receives the quantized coefficients and encodes them to generate the compressed bit-stream file which has the (.COD) extension that represents the (WAAVES format).

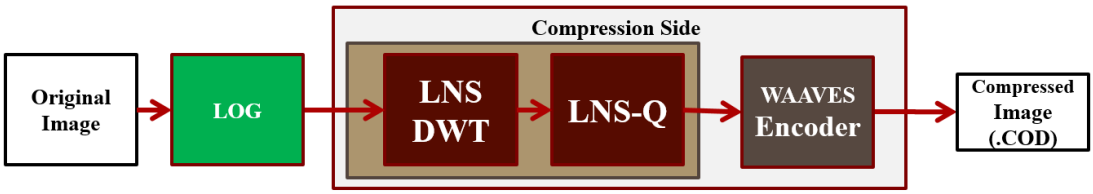


Fig. 4.1 The integration of LNS-DWT/LNS-Q with WAAVES

For the analysis and display simplicity purposes, after the calculation of the DWT coefficients, we converted them from the matrix representation into the vector form. That was done by rearranging the image lines and putting each line besides its previous line

consecutively to form a single vector. Figure 4.2 shows an example of how to convert the image from 2D to 1D. That made the comparison simpler between both domains.

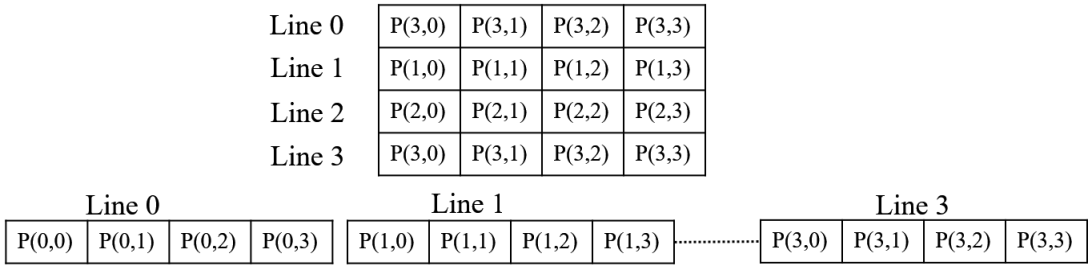


Fig. 4.2 An example of converting an image matrix into a vector

To evaluate the compression performance of the modified WAAVES, we compressed the medical image sample shown in Figure 3.10b using different LNS-Q scale factors. Figure 4.3 and Figure 4.4 show the 1D form of the un-quantized DWT coefficients in both domains, the linear and the logarithmic respectively. Interestingly, the dynamic range is between (-49.59 and 15.49) when using the LNS-DWT, while it is between (-16523.87 and 46103.35) when using the linear DWT.

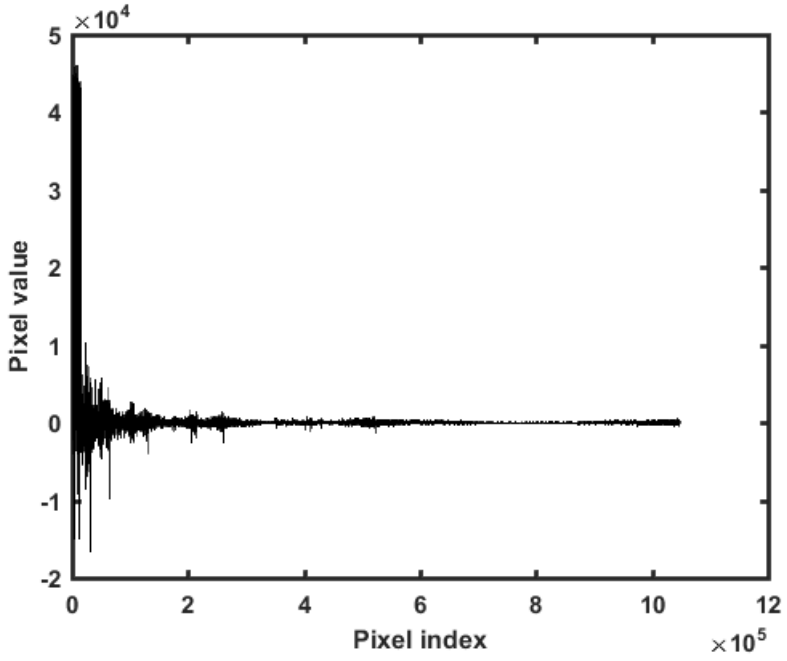


Fig. 4.3 Example of a linear DWT dynamic range

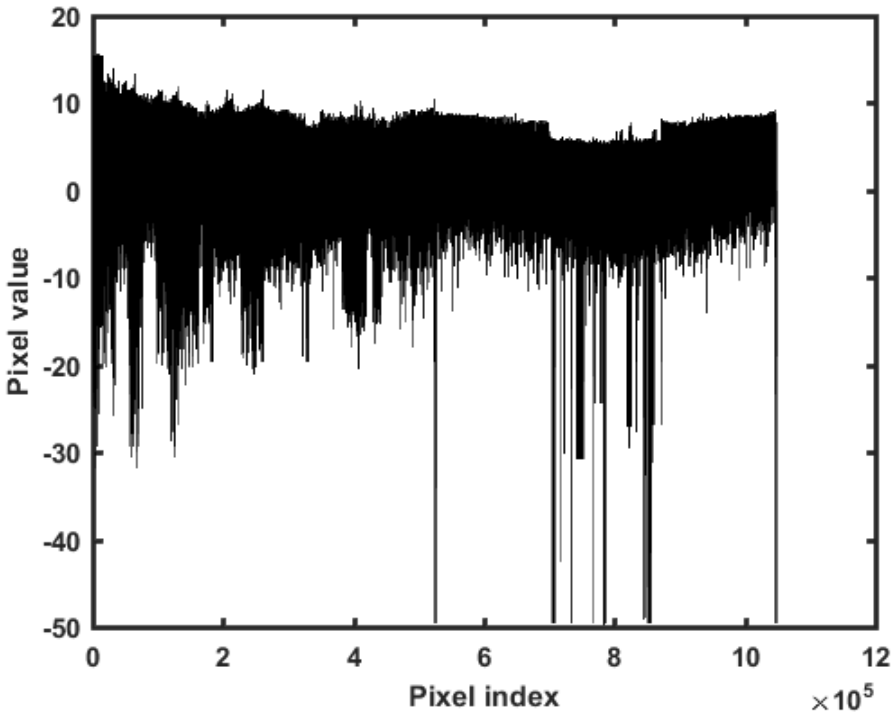


Fig. 4.4 Example of a logarithmic DWT dynamic range

Table 4.1 lists a detailed comparison between the linear and the logarithmic WAAVES. Looking to LNS-WAAVES, the dynamic range after the logarithmic quantization LNS-Q with (SC = 1 or SC =10) was smaller than that after the linear quantization. However, LNS-WAAVES yielded a maximum compression ratio (CR = 5.11) with (SC = 1), while the linear WAAVES yielded (CR = 6.05) with (q=10). Also, the dynamic range of the LNS-DWT was increased when applying LNS-Q with high scale factor (SC = 1000). On the other hand, the number of DWT zero coefficients in the linear domain was larger compared to that in the logarithmic domain (larger by a factor of 2X in this example).

Table 4.1 A comparison between WAAVES using linear DWT and LNS-DWT

BASE = 2	**min**	**max**	**zeros**	**CR**[1]	**C.Size**[2]	**SSIM**	**PSNR**
Input image	0	1023	196,513	0	2048	N/A	N/A
Linear Q = 10	-1653	4610	353,873	**6.05**	338	0.88095201	87.706
LNS No SC	-49.60	15.49	160,689	n/a	N/A	N/A	N/A
LNS SC = 1	-50	15	160,689	**5.11**	400	0.76739324	055.1846
LNS SC = 10	-496	155	160,689	2.64	775	0.85216806	075.0036
LNS SC = 100	-4960	1549	160,689	1.77	1156	0.95845764	093.7436
LNS SC = 1000	-49598	15493	160,689	1.34	1533	0.99999887	140.3030

[1] Compression Ratio, [2] C.Size = file compressed size in KB,

To have a deep insight into the dynamic range and the data distribution, we inspected the histogram of the quantized DWT coefficients. The histogram displays the number of occurrences (frequency) of each coefficient. Figure 4.5 shows the histogram of the linear quantized DWT coefficients. It shows that the DWT coefficients are concentrated around the zero. Also, the majority of coefficients are zeros, while the rest of the values are distributed smoothly. On the other hand, the majority of quantized LNS-DWT coefficients are not equal zero. Figures from 4.6 to 4.9 show the histogram of the quantized LNS-DWT with different scaling factor values.

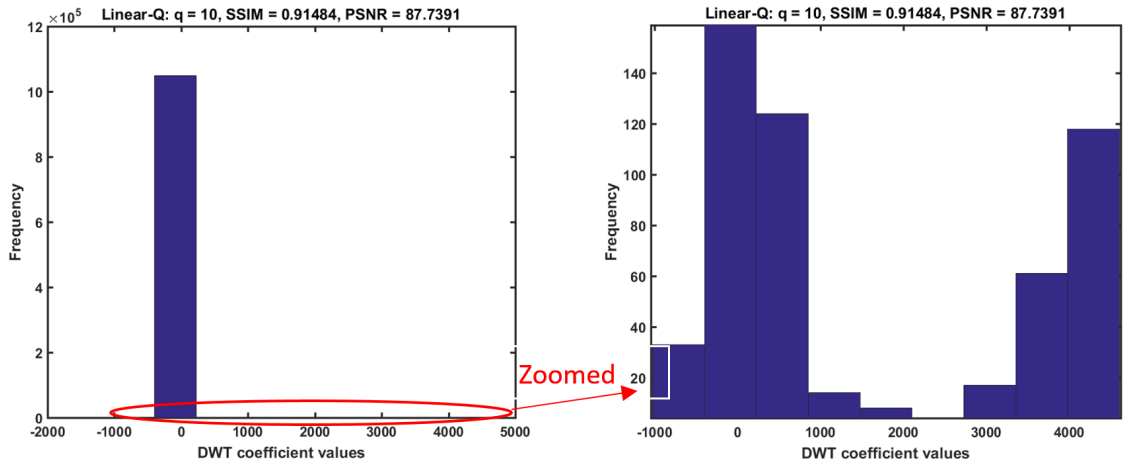


Fig. 4.5 The histogram of the linear DWT coefficients after quantization with q = 10

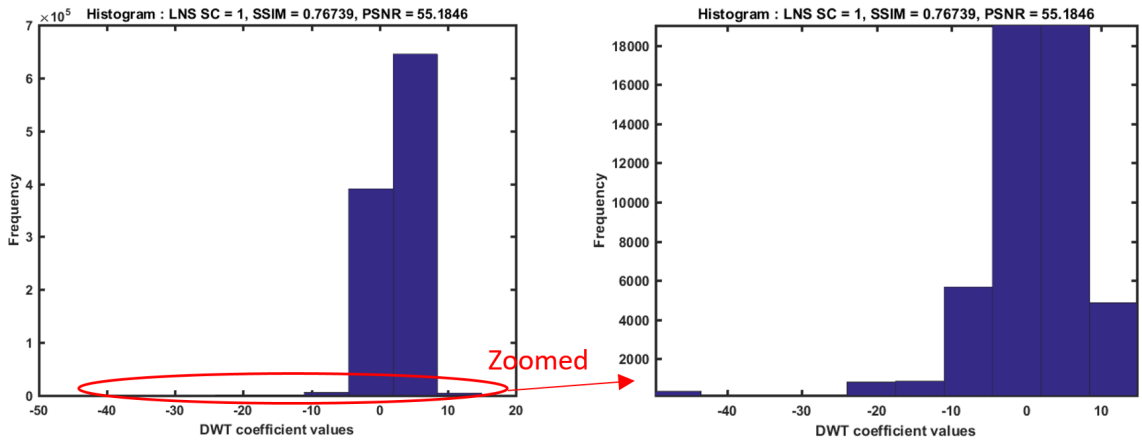


Fig. 4.6 LNS-DWT histogram with SC = 1

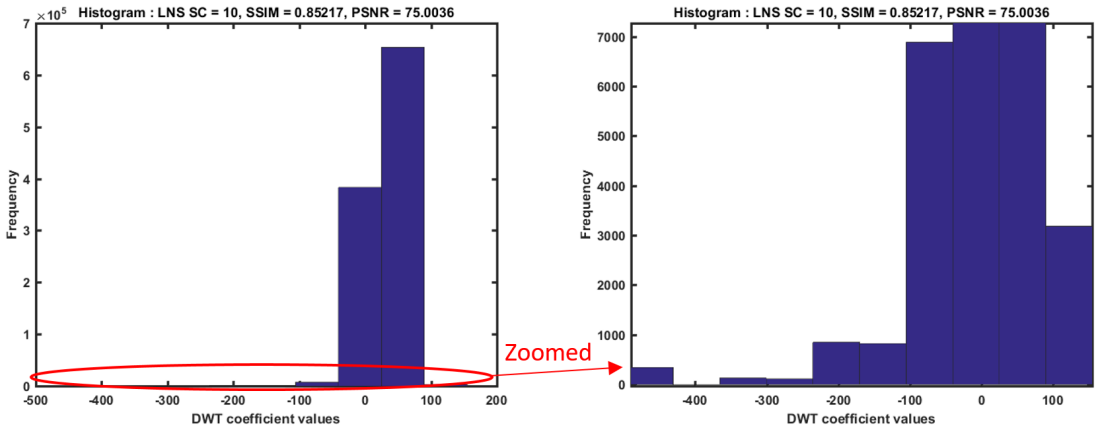


Fig. 4.7 The Quantized LNS-DWT histogram with SC = 10

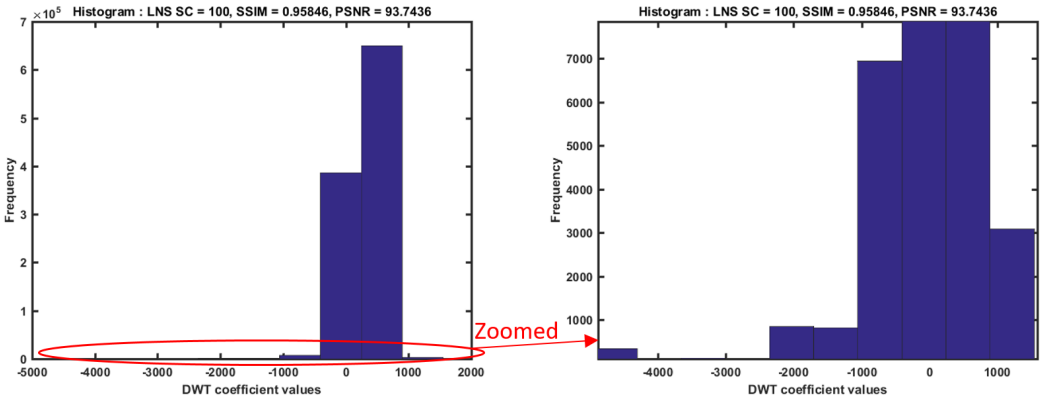


Fig. 4.8 The Quantized LNS-DWT histogram with SC = 100

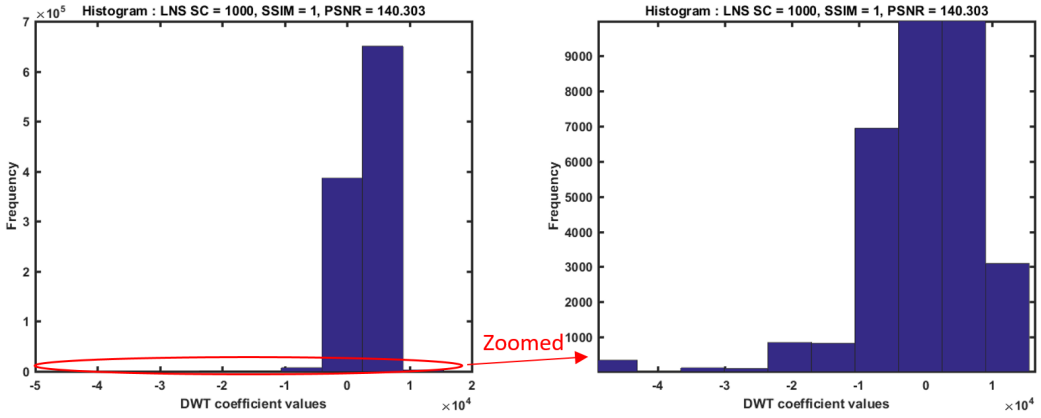


Fig. 4.9 The Quantized LNS-DWT histogram with SC = 1000

Conclusion

The previous results show that the WAAVES encoder (HENUC) did not compress the quantized LNS-DWT efficiently. There are three observed reasons that yielded a smaller compression ratio: (1) the number of quantized DWT zero coefficients is small; (2) the dynamic range of the whole LNS-DWT coefficients was increased when higher scale factors were applied; (3) the LNS-DWT coefficients were not distributed smoothly around the zero. Hence, according to this analysis, we concluded that the basic integration of LNS-DWT/LNS-Q achieved a modest compression ratio with having a high quality compressed image. These results motivated us to propose a new compression scheme that combines the two advantages: better compression ratio with better quality.

4.2.2 Hybrid-DWT Based Compression Scheme

According to the data range analysis of the DWT coefficients, the LL sub-band contains the highest dynamic range values in the whole DWT coefficients, while the other sub-bands contain smaller values. Therefore, since the LL sub-band contains the important values that have a high impact on the image quality, we found that it is necessary to keep them in a high precision. Figure 4.10 shows an example of the DWT coefficients, it is evident that the large values are concentrated in the LL-subband.

Utilizing the characteristics of the dynamic range of the DWT coefficients, we proposed a new compression scheme based on the Hybrid-DWT in the two domains. The main idea of the Hybrid-DWT is to create a compressed bit-stream consisted of two parts: an LNS part which represents the LL sub-band and a linear part which represents the other three sub-bands. This approach has us two advantages: First, we take the benefit from data distribution around the zero in the linear part. It also contains smaller values and more zeros after applying the linear quantization. The second advantage, we take the benefit from the LNS-domain, since it has the data compactness in addition to preserving the precision. Figure 4.11 shows the block diagram of the proposed compression scheme based on the Hybrid-DWT.

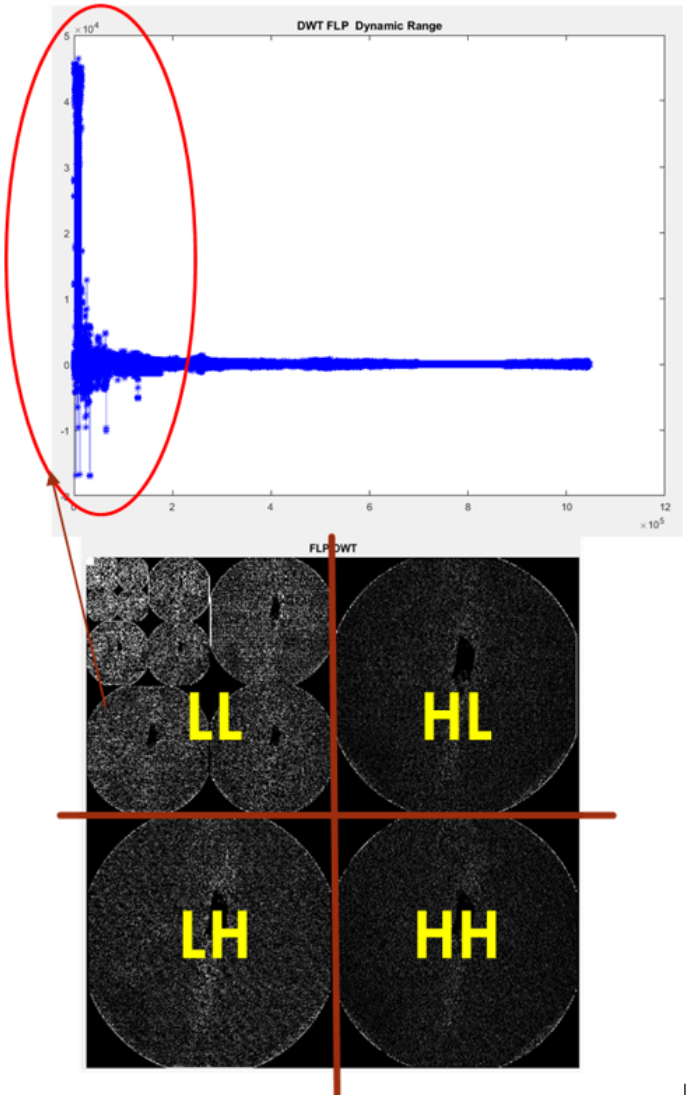


Fig. 4.10 Dynamic range of LL sub band for Linear DWT

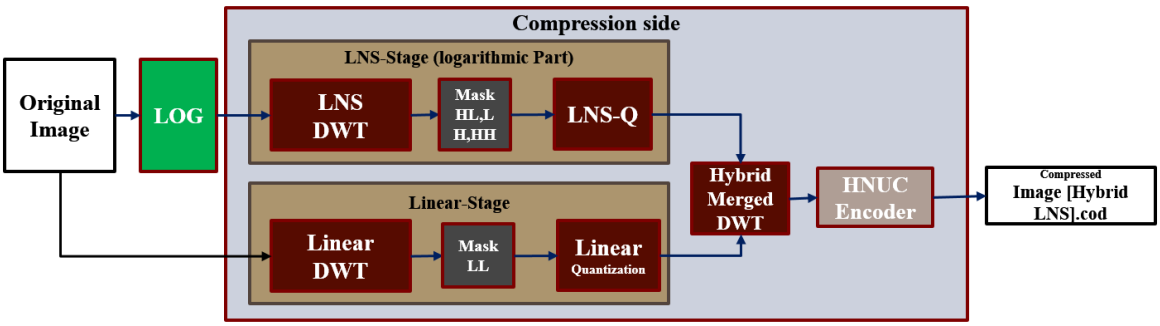


Fig. 4.11 Hybrid-DWT compression scheme

Hybrid-DWT/LNS-Q Description

In the first stage, we calculate the LNS-DWT. Then, we mask the sub-bands (LH, HL, and HH). After that, we apply the LNS-Q quantization method. In the second stage, we calculate the linear DWT for the whole image. Then, we mask the LL sub-band. After that, we quantize the sub-bands (LH, HL, and HH) using the linear quantization. The masking operation is realized by replacing the targeted sub-band coefficients values with zeros. The outputs from the two stages are merged and embedded in a single bitstream to be encoded to generate the final compressed bit-stream. Figure 4.12 shows the construction of the two parts of the Hybrid-DWT. Also, We introduced a new parameter called NL that specifies the number of the 2D DWT levels that will be kept in the linear domain when using Hybrid-DWT.

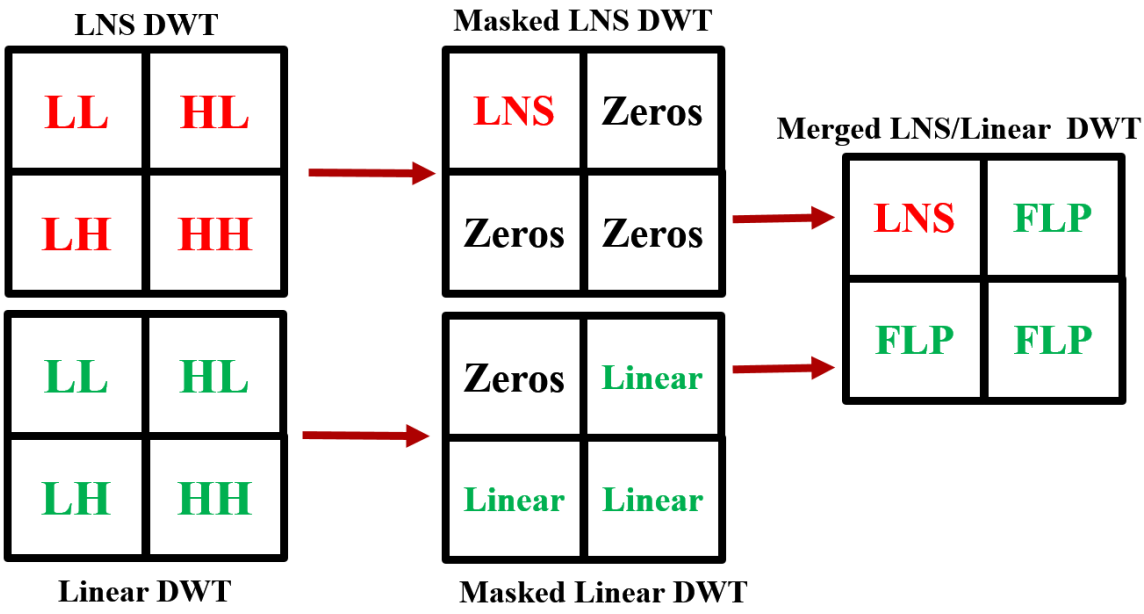


Fig. 4.12 Hybrid-DWT construction

4.2.3 LNS-DWT Dynamic Range Reduction Filter

Due to the fact that, the small values in the linear domain become large negative values in the logarithmic domain which affects the encoding efficiency. Example 4.1 explains the issue of the dynamic range in the logarithmic domain after applying LNS-Q quantization method.

▼ **Example 4.1.**

Let X is a part of a line in the LNS-DWT coefficients matrix with base = 2,

where X = {10.153, -85.157, 14.154, -0.152 }.

and let Y is the linear domain of X, where $Y = 2^X$,

$Y = \{1138.56, 2.31 \times 10^{-26}, 18229.66, 0.90\}$.

After scaling with SC = 100, the value of -85.157 in X becomes -8516, which means it requires higher number of bits to be coded.

Therefore, we propose a solution to the large negative values found in the quantized LNS-DWT. That was handled by introducing a post-processing stage called dynamic range reduction filter (DRR). The main objective of the DDR filter is to decrease the dynamic range by replacing the large negative values that are above the filter threshold (FTH) with zeros. In addition to, the DDR filter helps to introduce more zeros in the DWT coefficients that leads to making them able to be encoded efficiently and, hence, improving the compression ratio.

The filter threshold (FTH) parameter gives the proposed compression scheme more flexibility to enhance the compression performance. Choosing FTH value depends on the dynamic range of the LNS-DWT coefficients and the logarithmic base used, as we will see in the results section (Section 4.3).

DRR Filter Description

The DRR filter divides the pixel values of the LNS-DWT coefficients L into two subsets PE, VE that corresponds to positive values PE and the negative values VE respectively, as defined in equation (4.1):

$$\text{LNS DWT Coefficients (L)} = \begin{cases} PE & L \geq 0 \\ VE & L < 0 \end{cases} \tag{4.1}$$

The filter converts the VE to zeros, only if VE are less than the threshold FTH as shown in equation (4.2):

$$\overline{VE} = VE < FTH \tag{4.2}$$

where $\overline{VE}$ are the targeted filtered values, $\overline{VE} \in VE$.

The order of applying the LNS-Q operation after the DRR filter is necessary. That is because the filter will provide extra zero values which will remain zeros even after the scaling operation.

Figure 4.13 shows an example of two small matrices that represents few DWT coefficients in the logarithmic domain before and after LNS-Q with SC = 100. As shown in Figure 4.13b, the large negative values increase the dynamic range dramatically from {-104.15 , 12.5} to { -10415 , 1250}.

1.23	-19.8	0.1	0
12.5	-17.1	12.1	3.2
-104.15	10.2	-13.8	1.1
-5.23	0	2.7	-16.9

(a) LNS-DWT coefficients before scaling
Dynamic range: (-104.15 to 12.5)

123	-1980	10	100
1250	-1710	1210	320
-10415	1020	-1380	110
-523	100	270	-1690

(b) LNS-DWT after scaling
Dynamic range: (-10415 to 1210)

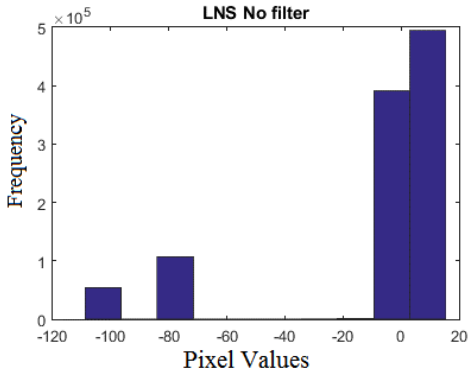
Fig. 4.13 Example of scaling the LNS-DWT coefficients without the DRR post filtering (SC = 100)

On the other hand, Figure 4.14 shows the impact of using the DRR filter. The example shows how the DRR optimizes the LNS-DWT dynamic range to {-523, 1210}, which make them are encoded in fewer bits.

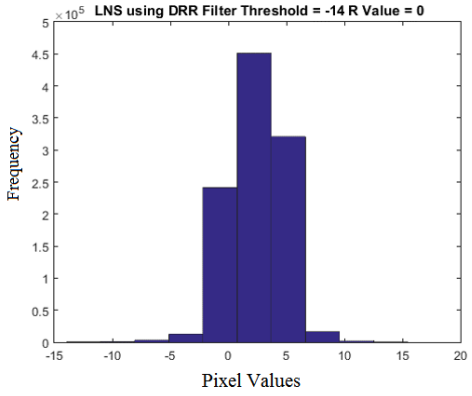
123	0	10	0
1250	0	1210	320
0	10.2	0	110
-523	0	270	0

Fig. 4.14 Scaling the LNS-DWT coefficients (SC = 100) after DRR filter
FTH = -12, dynamic range: between -523 and 1250

Figure 4.15 shows the histogram for full unscaled 2D-LNS DWT before and after applying the DRR with $FTH = -14$. The data are distributed smoothly around the zero. On the contrast to the original values that contained large negative values which is considered as extra bit for the ecnoder.



(a) Histogram of LNS-DWT coefficients before applying the DRR filter



(b) Histogram of LNS-DWT coefficients after applying the DRR filter $FTH = -14$

Fig. 4.15 Example of the LNS-DWT Coefficients before and after using the DRR filter

In summary, the DRR filter acts as a rounding-to-zero operation. Hence, in addition to the SC and q step quantization parameters, the proposed compression scheme has additional parameter FTH that plays an important role to control the dynamic range.

4.2.4 The Log Base

One of the most the interesting questions that was raised during this research; *"What is the impact of changing the logarithm base on the image compression?"*. To answer this question, we used the change-of-base formula for calculating the logarithm of x with a base b as described in equation (4.3). It requires dividing the logarithm of x by the logarithm with the base b using a common auxiliary base k.

$$\log_b(x) = \frac{\log_k(x)}{\log_k(b)} \tag{4.3}$$

Since the logarithmic value is a function of the required base, it is evident that using a higher base of a logarithm yielding a smaller logarithmic value. Therefore, it was interesting to use higher logarithmic base values and observe their influence on the dynamic range of an input image and on the compression ratio as well.

Figure 4.16 shows the relation between the logarithmic base and the maximum value in the logarithm of an input image when using different bases. The data on the curve indicates that the dynamic range was decreased from 11 to 3 when varying the log base from 2 to 16. Also, we found that higher base values (b $>$ 12), yielded stable logarithmic values near to 3.

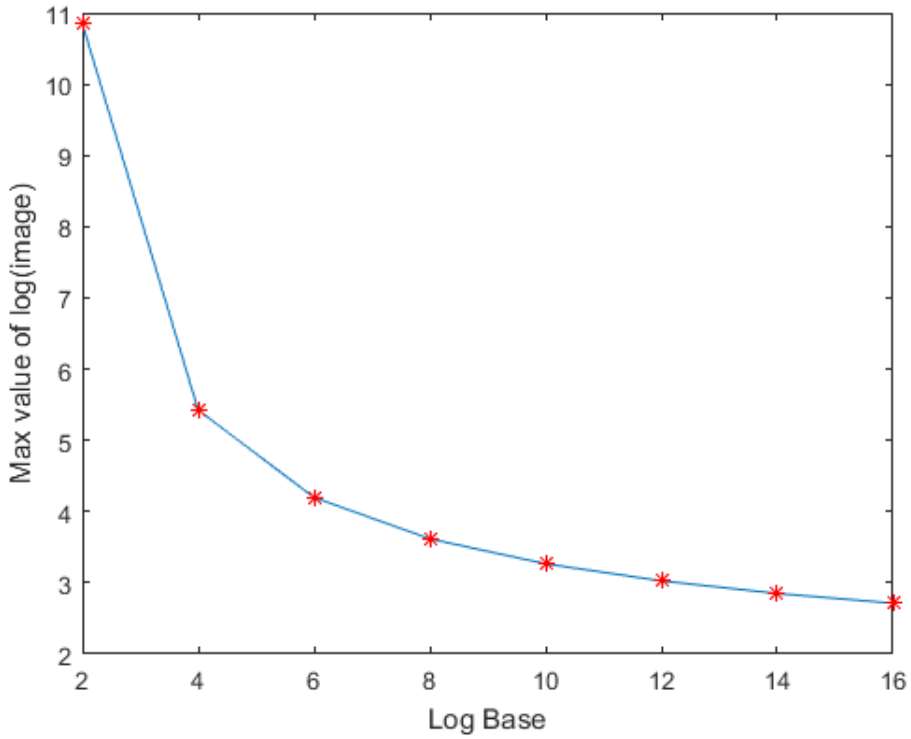


Fig. 4.16 The impact of using higher log base on the dynamic range of an input image

4.2.5 Summary

The full compression scheme based on the hybrid-DWT is shown in Figure 4.17. It requires five parameters which give a tread-off between the compression ratio and the image quality. Those five parameters are listed as the following:

- SC: Scaling factor for the DWT sub-bands that are computed in LNS.
- q: Quantization step for the DWT sub-bands that are computed in the linear domain.
- FTH: DRR filter threshold.
- NL: The number of the DWT levels that are computed in linear domain .
- B: the logarithm base used in the conversion from the linear to the logarithmic domain.

Algorithm 9 lists the basic steps used to implement the proposed Hybrid-DWT based compression scheme.

Algorithm 9: LNS-WAAVES Compression based on the Hybrid-DWT

Input : Image in the linear domain, SC, q, FTH, NL, B
Output : Compressed image in logarithmic domain

1 Calculate the logarithm of the input image.
2 Calculate the 2D LNS-DWT of the logarithmic image.
3 Apply the DRR filter on the LNS-DWT
4 Calculate the 2D DWT of the linear image.
5 Quantize the masked LNS DWT coefficients (LL-sub-band only) using LNS-Q
6 Quantize the masked linear DWT coefficients (LH, HL, and HH sub-band) using linear quantization.
7 Merge the two DWT parts
8 Encode the hybrid-DWT using HENUC encoder
9 Write the encoder outputs into a bit stream WAAVES format file

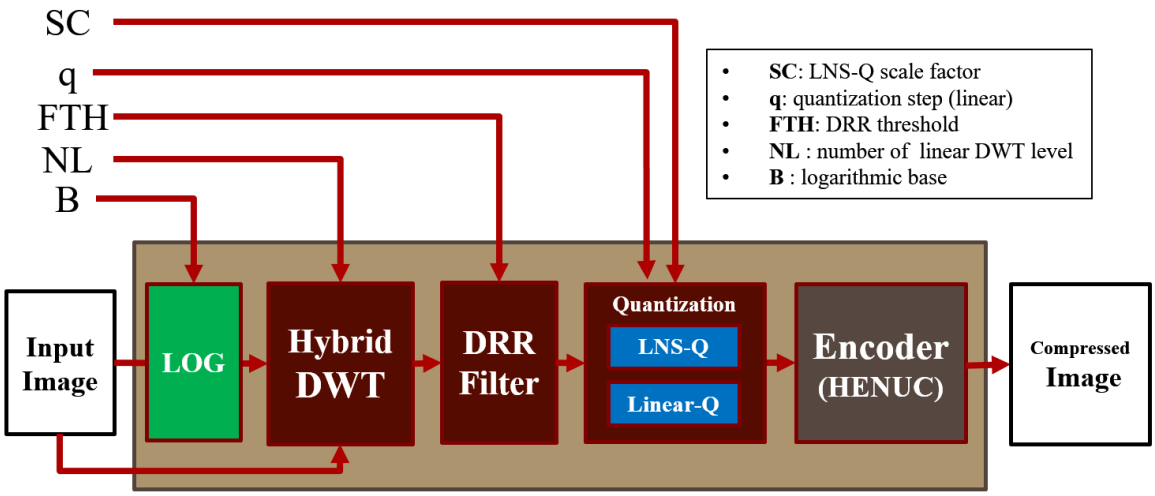


Fig. 4.17 Full-LNS Compression scheme with DRR

4.3 Results and Discussion

In this section, we present the results of integrating the Hybrid-DWT scheme, LNS-Q, and DRR filter with WAAVES to construct the new full compression chain (LNS-WAAVES). First we start with describing the experiments methodology.

4.3.1 Methodology

The block diagram in Figure 4.18 shows the basic steps of the experiments methodology. We started by extracting the raw image data from the DICOM file. Following to that, compressing the input images using LNS-WAAVES. Then, reconstructing the images again by decompressing it. After that, we evaluate the image quality using the SSIM and PSNR metrics. Also, for the purpose of comparison, the same steps are followed to compress and reconstruct the input images using WAAVES and JPEG2000.

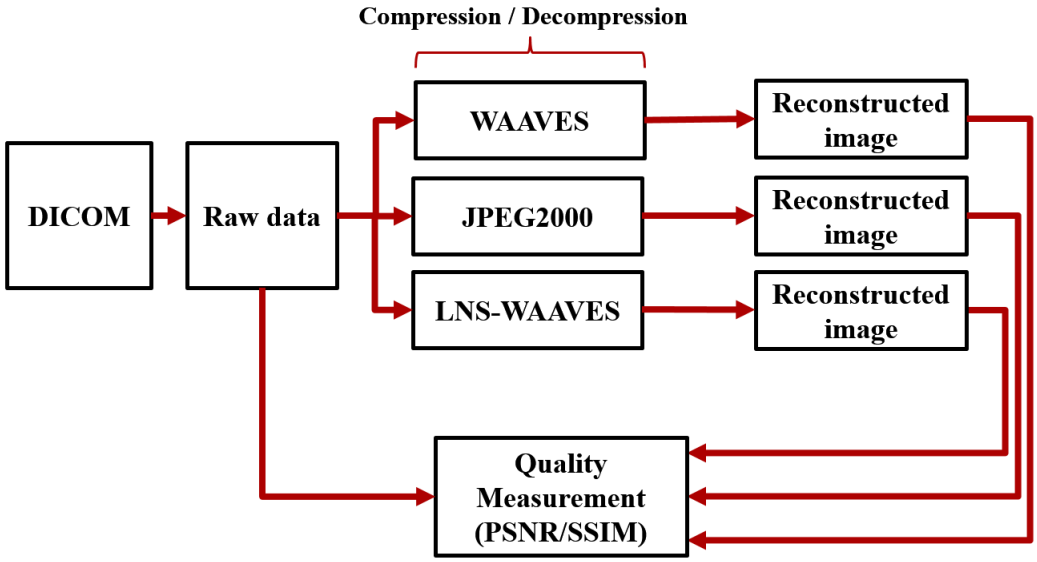


Fig. 4.18 Experiments Methodology for evaluating the LNS-WAAVES.

Both the compression ratio CR and the image quality are functions of the compression parameters as shown in equation (4.4). The objective of the experiments in this chapter is to study the effects of these parameters on the image compression efficiency.

$$CR, image equality = f(q, SC, NL, FTH, B) \tag{4.4}$$

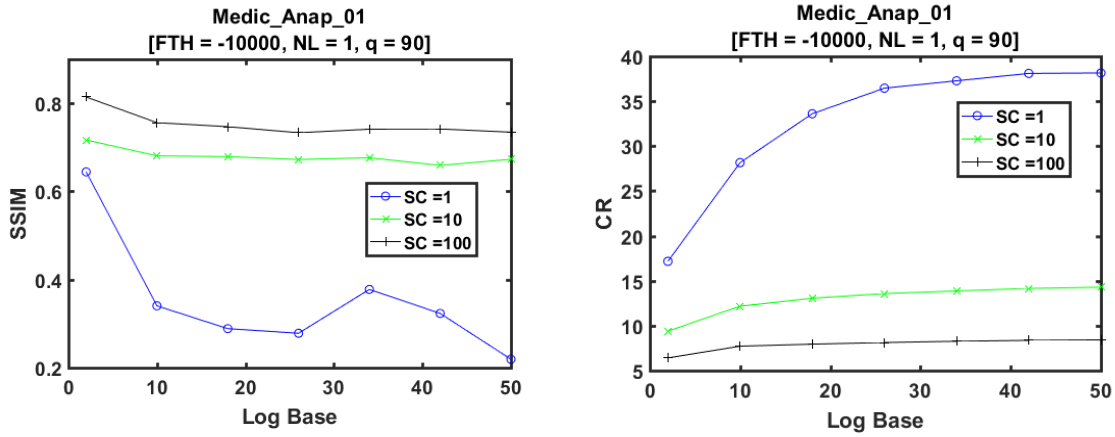
We started the experiments by exploring the log-base effect when compressing LNS-WAAVFES using different log bases. Then we study the hybrid quantization effects by compressing the images using different quantization parameters (SC,q). Also, we investigate the impact of varying the number of the linear DWT levels (NL) on the image quality and the compression efficiency. Then, the effect of the DRR filter on the compression ratio was explored. All the obtained results are compared with the linear WAAVES and the JPEG2000 standard as well.

In the first experiments, we disabled the DRR filter to investigate the effects of the other compression parameters. As we will see in the following graphs, the value of FTH = -10000 means the filter was disabled.

4.3.2 The Impact the Logarithmic Base

In the following examples, a medical image was compressed using LNS-WAAVES with different logarithmic bases by varying them from base 2 to base 50. As shown in Figure 4.19, when we compressed with NL = 1 and SC = 1, the compression ratio jumped from 17.17 to 38.15 by a factor of 2.2X. When we compressed with SC = 10, the CR was increased by a factor of 1.5X and was increased by a factor of 1.3X with SC = 100. Also, we can notice from the curves that, the compression ratio became steady at the higher logarithmic bases. The reason for that is the dynamic range stability, which makes the encoder gives similar results.

On the other hand, the SSIM was decreased significantly with SC =1. The main reason for the quality degradation is the implicit division in the logarithmic calculation when using higher logarithm base according to equation (4.3). Thus, it gave a similar effect of the quantization operation. However, the quality degradation was limited when higher scale factors were used. For example, when using SC = 100, the SSIM was decreased by a value of 0.03.

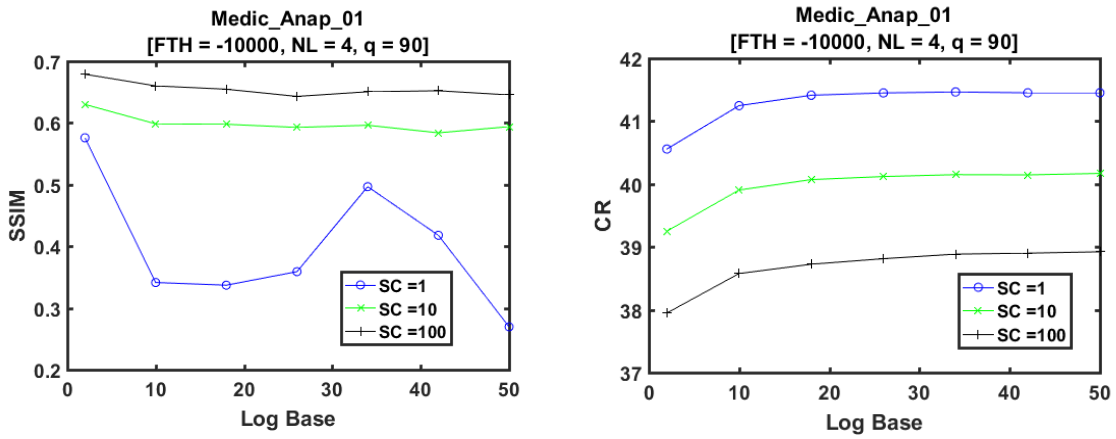


(a) Log base verses SSIM

(b) Log base verses CR

Fig. 4.19 The impact of changing the logarithm base (NL = 1)

The effect of changing the base on the compression ratio was limited when high NL values were used. For example, Figure 4.20 shows that the CR was increased by a value of 0.8 when we compressed with NL = 4.



(a) Log base verses SSIM

(b) Log base verses CR

Fig. 4.20 The impact of changing the logarithm base (NL = 4)

From the previous results, we concluded that choosing the logarithmic base value (B = 2) yielded best results for the image quality. Therefore, we chose B = 2 in the following experiments.

4.3.3 The impact of LNS-Q and NL

Figure 4.21 to Figure 4.25, show the impact of changing the number of the linear DWT levels (NL) on the compression ratio and the image quality. The input image was compressed with different quantization steps for the linear part, and with different scaling factors for the LNS part. The SSIM and the compression ratio were measured for each case.

The results showed that a limited compression ratio was achieved when compressing using LNS-WAAVES with NL = 1. For example, Figure 4.21 shows that the maximum compression ratio that LNS-WAAVES can achieve was CR = 17.48 with SC = 1. In contrast, WAAVES and JPG2000 reached a compression ratio up to CR= 120.

In Figure 4.22, when using LNS-WAAVES with NL = 2, the compression ratio was increased to 59.14. Also, with SC = 100, it yielded an SSIM better than WAAVES and JPEG2000 at the range (6 < CR < 22). For example, at CR = 16.4, LNS-WAAVES yielded better SSIM than WAAVES with a factor of 1.06X and better than JPEG2000 with a factor of 1.08X. Since the SSIM is a very sensitive metric, a slight positive change in its value indicates much better image quality.

In Figure 4.23, when using LNS-WAAVES with NL = 3 the compression ratio jumped significantly to 140. Also, it yielded an SSIM better than JPEG2000 and WAAVES. For example, within the range (10 < CR < 61), LNS-WAAVES with SC = 100 yielded a better SSIM by a factor of 1.17X than WAAVES, and by a factor of 1.16X better than JPEG2000. Also, at the higher compression ratio values within the range (74 < CR < 122), with SC = 1,

it yielded a better SSIM by a factor of 1.06X than WAAVES, and by a factor of 1.16X better than JPEG2000.

Figure 4.24 shows when using LNS-WAAVES with NL = 4 the compression ratio jumped to 143. With SC=100 it yielded a better SSIM by a factor of 1.12X than WAAVES, and by a factor of 1.22X better than JPEG2000.

Figure 4.25 shows when using LNS-WAAVES with NL = 5 and SC=100, it yielded a better SSIM by a factor of 1.05X than WAAVES and better than JPEG2000 by a factor of 1.15X, both within the range (70 < CR < 120).

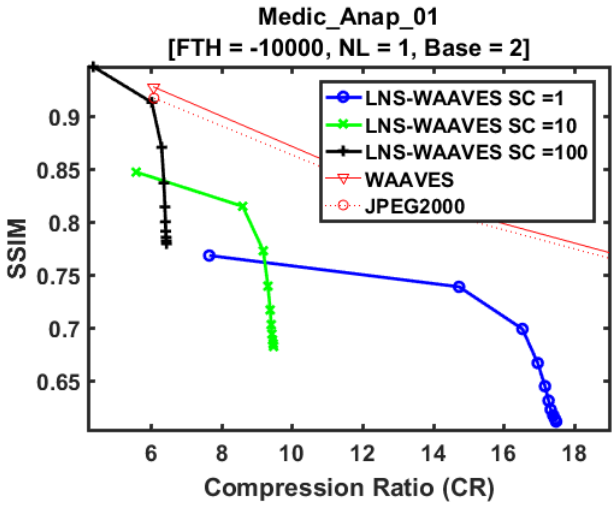


Fig. 4.21 CR versus SSIM, Hybrid-DWT with NL = 1

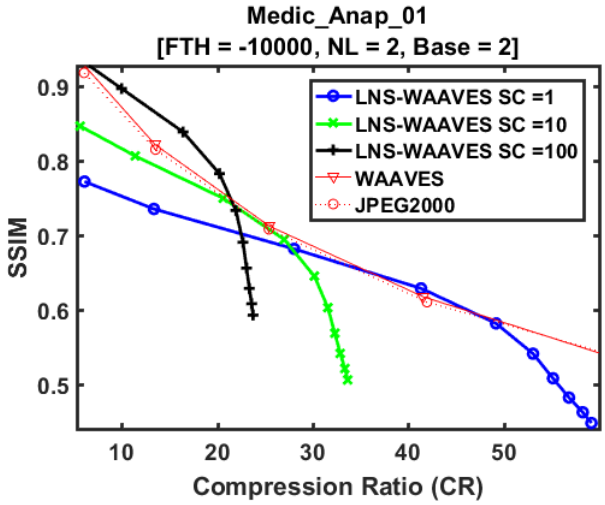


Fig. 4.22 CR versus SSIM, Hybrid-DWT with NL = 2

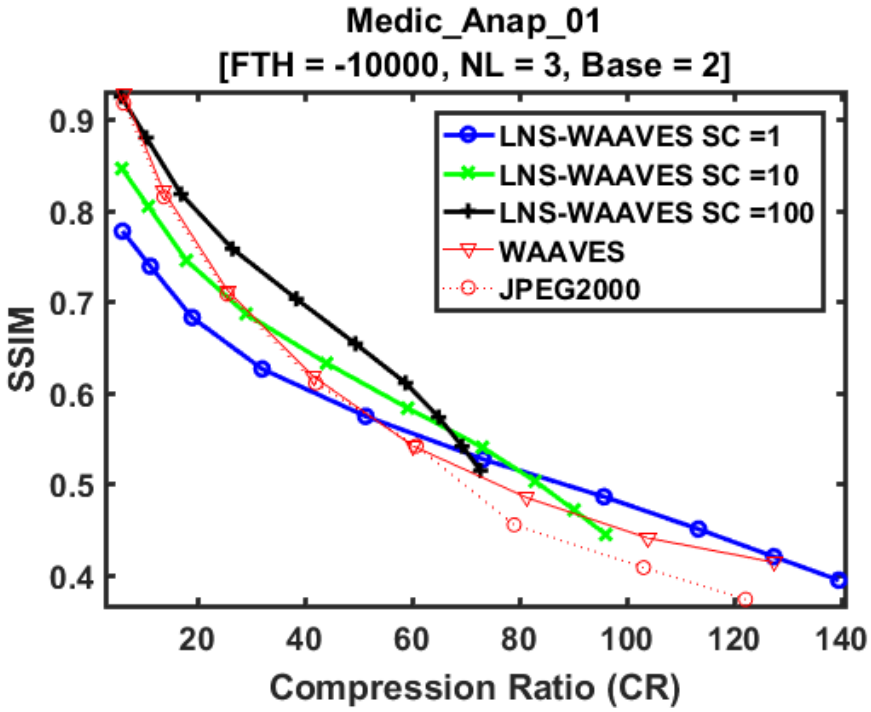


Fig. 4.23 CR versus SSIM, Hybrid-DWT with NL = 3

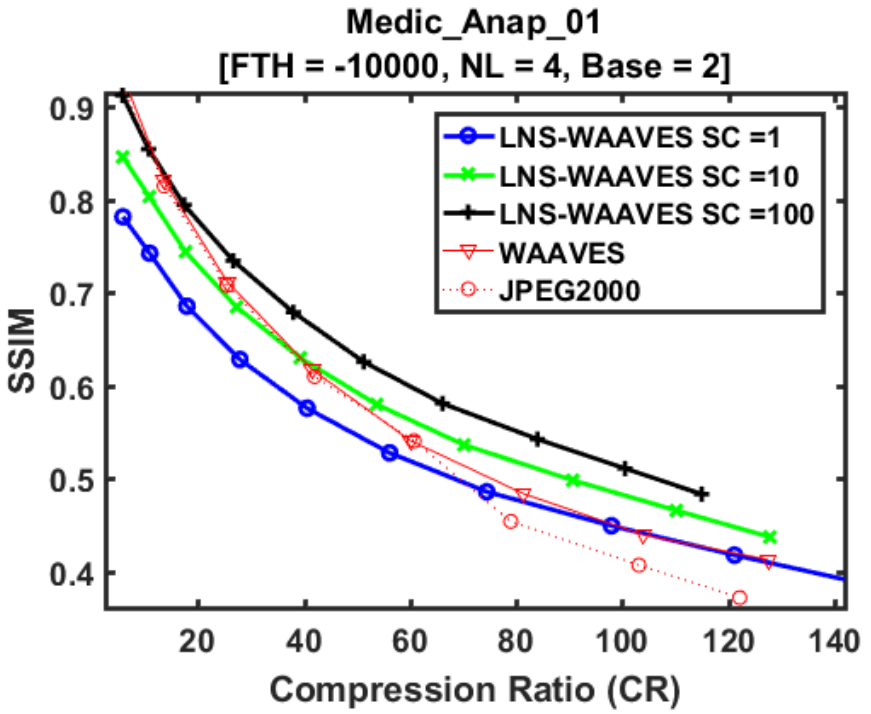


Fig. 4.24 CR versus SSIM, Hybrid-DWT with NL = 4

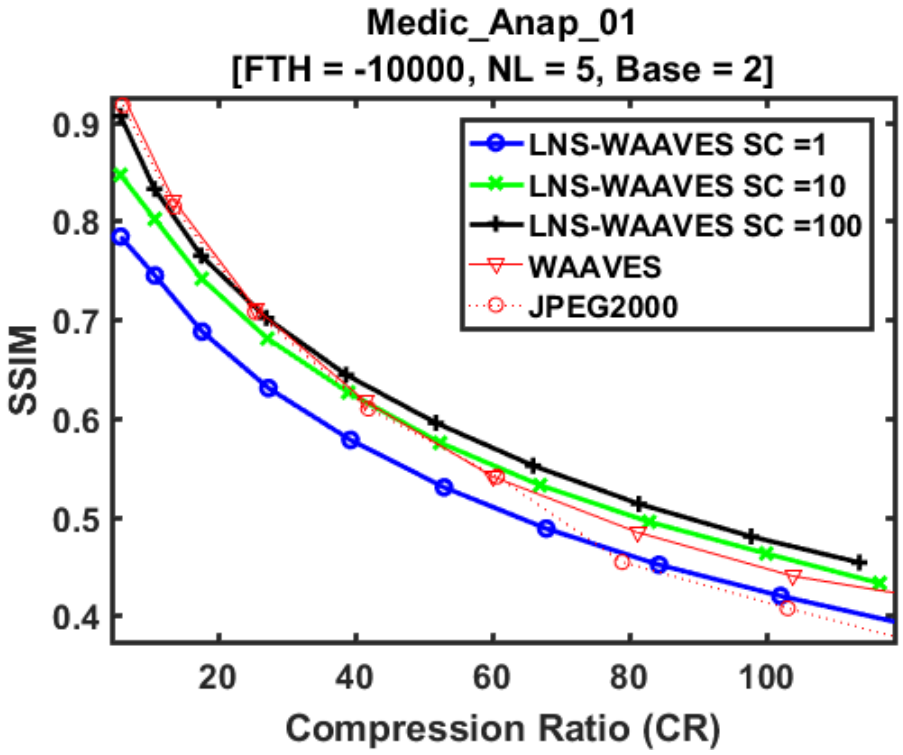


Fig. 4.25 CR versus SSIM, Hybrid-DWT with NL = 5

4.3.4 The Influence of the Quantization

In LNS-WAAVES, there are two components affect the quality: the LNS-Q scaling factor in the logarithmic part and the quantization step in the linear part. Both of them leads to a quantization error. The LNS-Q has less quantization error than the linear quantization. The effect of the quantization error depends on which part has a larger number of coefficients which is determined by NL. In other words, the number of the linear DWT levels (NL) indicates which part has the dominant effect. Small NL value means having a smaller number of linear DWT coefficients than the logarithmic LNS. Therefore, LNS-WAAVES gives better quality because the LNS-Q effect dominates the linear effect. In general, LNS-Q limits the linear quantization effect, especially when using large q-step in the linear part, which gives the Hybrid-DWT an advantage over the classical method.

The percentage of the SSIM improvement was calculated, to observe the influence of the quantization error on the image quality. It compares LNS-WAAVES with WAAVES and JPEG2000 for each quantization step for the different SC values. Equation (4.5) describes the percentage of the SSIM improvement; the positive value indicates an improvement in the quality.

$$\%\text{Percentage of SSIM improvement using LNS} = \frac{SSIM_{lns} - SSIM_x}{SSIM_x} \times 100 \tag{4.5}$$

where $SSIM_{lns}$ is the SSIM value for LNS-WAAVES, and $SSIM_x$ is the SSIM value for JPEG2000 or WAAVES.

Figure 4.26 to Figure 4.28 show the percentage of the SSIM improvement using LNS-WAAVES compared to WAAVES and JPEG2000. For example, in case of (NL = 4, SC = 100, q= 190), at CR = 115, the quality was improved by 49.43% better than JPEG2000, and by 34.96% better than WAAVES. That indicates the quantization error was increased significantly at the larger quantization step values when using WAAVES and JPEG2000, while it was limited when LNS-DWT was used. Therefore, the quality was improved significantly when Hybrid-DWT was used. The negative values in the curves mean that WAAVES and JPEG2000 gave better quality than LNS-WAAVES; that happens in certain cases. The first case occurs when using a small SC value; in this case, the linear quantization with a small q-step achieves better quality. The second case occurs when NL is large; so the linear quantization effect dominates the LNS effect.

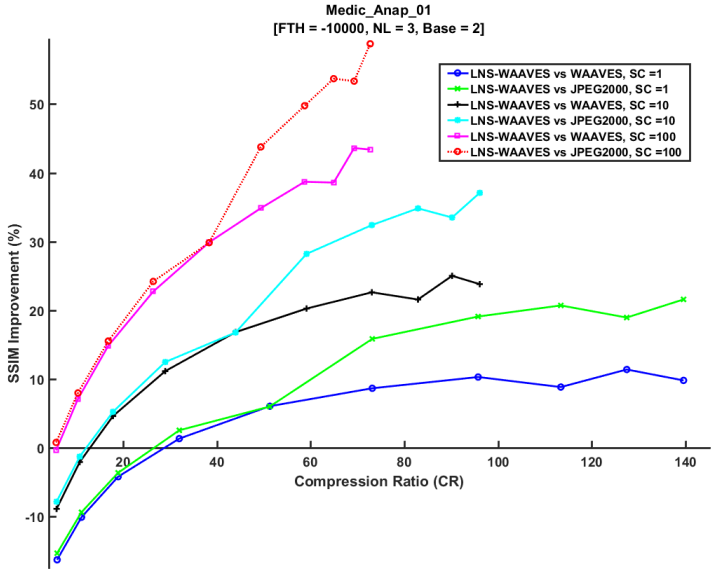


Fig. 4.26 CR versus the SSIM improvement, Hybrid-DWT with NL = 3

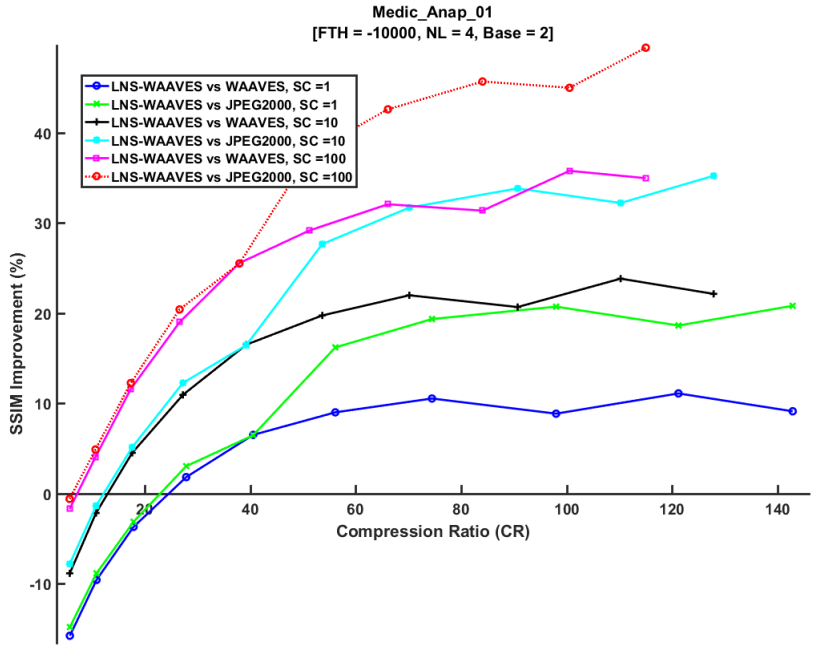


Fig. 4.27 CR versus the SSIM improvement, Hybrid-DWT with NL = 4

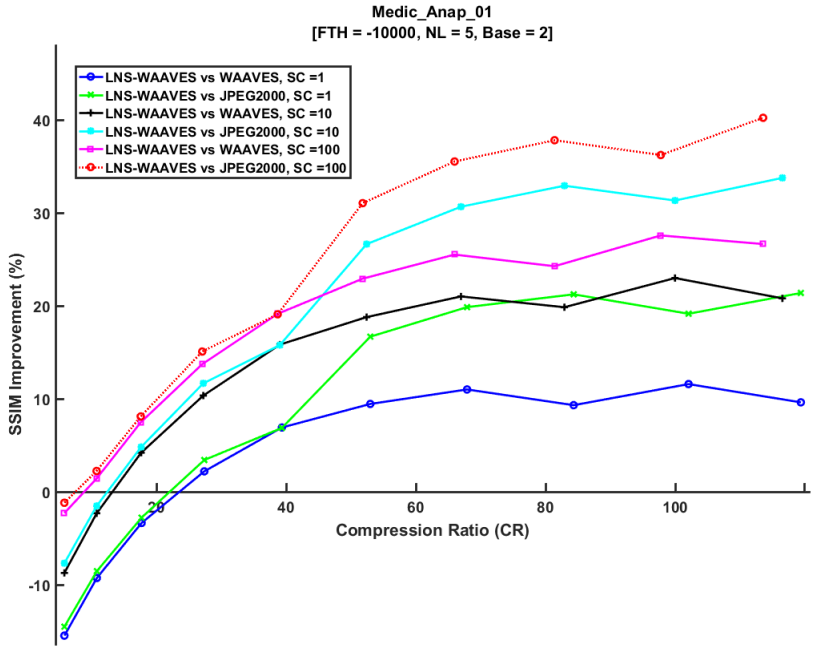


Fig. 4.28 CR versus the SSIM improvement, Hybrid-DWT with NL = 5

4.3.5 The Impact of the DRR

Figure 4.29 shows an example of the impact of applying the DRR filter on the compression ratio. First we disabled the filter (FTH = -10000) and compressed the input image to calculate the compression ratio that was used as a reference to compare with when we use different FTH values. After that, different threshold (FTH) values were applied to the LNS-DWT part in the hybrid DWT. For example, when choosing FTH = -1 with SC = 1, the CR jumped from 17 to 19 and yielded 11% improvement in the compression ratio. Also, the SSIM was decreased slightly; since the SSIM difference between setting FTH=-1 and without DRR is equal 6.6×10^{-6}. The small difference in the quality was because that the DRR replaced only the near-zero DWT coefficients with zeros.

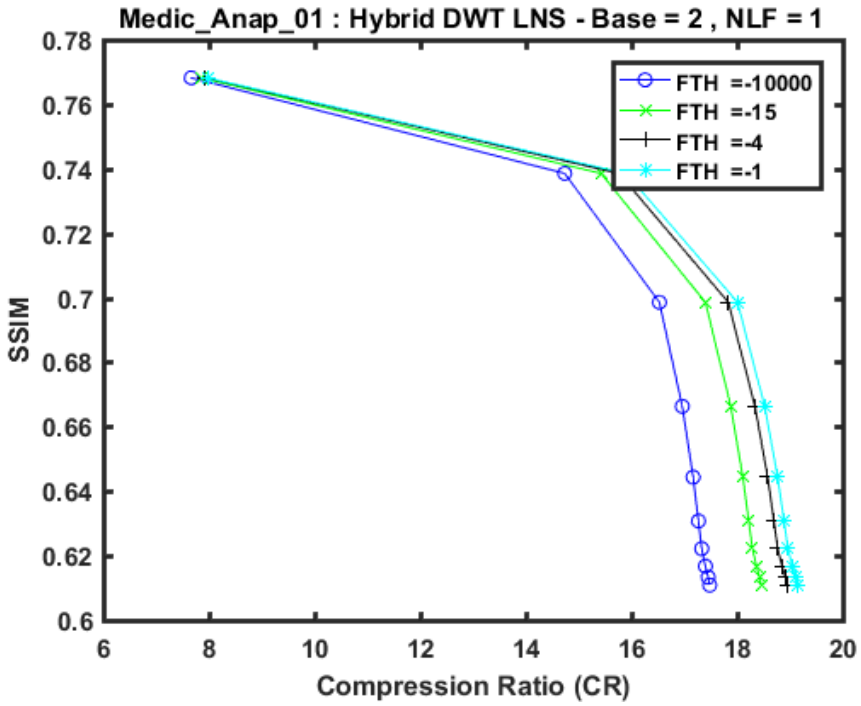


Fig. 4.29 DRR Filter impact on the CR with NL = 1

4.3.6 Results Summary and Discussion

We presented the effects of the LNS-WAAVES compression parameters on the image quality and the compression ratio. LNS-WAAVES is based on Hybrid-DWT scheme.

First, regarding the logarithmic base impact on the compression ratio, we found that using a higher base value, achieve a better compression ratio, but at the coast of the image quality, hence, using base = 2 is considered the best choice for achieving high image quality.

In Hybrid-DWT the coefficients are divided into two parts: the logarithmic and the linear parts. The image quality and the compression ratio are affected, according to the majority of one of the two parts. The number of linear DWT level (NL) controls which part can be the majority. Therefore, in the case of using a small NL value, it yields a small compression ratio, but with a better quality, because the majority of the coefficients is in the logarithmic part. Also, the quality is controlled by two parameters: the LNS-Q scale factor (SC) for the logarithmic part and the quantization step (q-step) for the linear part. Also, using a high SC yields a better image quality. For the DRR filter effect, it is found that it has a positive impact on the compression ratio when using small NL value. That is because replacing the large negative values with zeros reduces the dynamic range, which makes them encoded more efficiently.

In summary, to achieve the best results in both the compression ratio and the image quality, NL = 3 and NL = 4 are considered the best values, since they give a balanced effect from the logarithmic and the linear parts in the Hybrid-DWT. Also, the results showed that, using SC = 100 yielded the best quality results.

4.4 Conclusion

In this chapter, we introduced LNS-WAAVES as a novel compression scheme. It is based on compressing the DWT coefficients in a hybrid fashion by merging the coefficients from the both linear and logarithmic domains. Thus, it combined the advantages of the both domains. Introducing LNS-WAAVES with its compression parameters gave it a high flexibility by having a tradeoff between the compression ratio and image quality by configuring these parameters.

In summary, LNS-WAAVES was evaluated by measuring the quality of the reconstructed image. Then it was compared with the state of art compression algorithms; WAAVES and JPEG2000. The results showed that thanks to the hybrid-DWT scheme,it was able to achieve improvement in the quality by a percentage of 8% up to 34% compared to WAAVES and by a percentage of 10% up to 49% better than JPEG2000.

Chapter 5

2D-DWT Hardware Architecture

Contents

5.1 Introduction

The second research question is related to the compression on embedded systems. There are two motivations to address the speed problem. First for image compression, as we mentioned in the previous chapter, the Hybrid-DWT contains two parts: the logarithmic and linear. Hence, the hardware solution is considered to solve the speed overhead when computing the DWT in both domains. The challenge is to provide a DWT architecture that is able to be configured to support the Hybrid Computation. The second motivation is the need of a high speed DWT for a video compression on embedded systems as a part of the Smart-EEG project. The main goal to achieve an accurate diagnosis for doctors, it requires a real time processing speed up to 100fps.

This chapter presents the proposed hardware architecture of the 2D DWT and its implementation on hardware. As we mention in Chapter 2, the lifting scheme was chosen, because it is more efficient than the convolution scheme, in term of the memory requirements, computational simplicity.

The chapter is organized as the following: Section 5.2 presents in details the global architecture and describes the novelties in our approach. Section 5.3 discuss the FPGA implementation results. Section 5.4 presents the integration of the proposed work with the prototype for the Smart-EEG project as an e-health medical video compression application.

5.2 Proposed Unified 2D DWT Architecture

In this section, we present the lifting-based 2D DWT global architecture and the main controller. Also, we discuss in details how the design can achieve high throughput by using our proposed innovative techniques that make our architecture is efficient on FPGA.

The Proposed 2DWT architecture consists of the following modules as in Figure 5.1:

- Global controller and ports arbiter
- Line buffers (LBs)
- The horizontal 1D DWT (H1D) units
- The vertical 1D DWT (V1D) units
- External memory interface

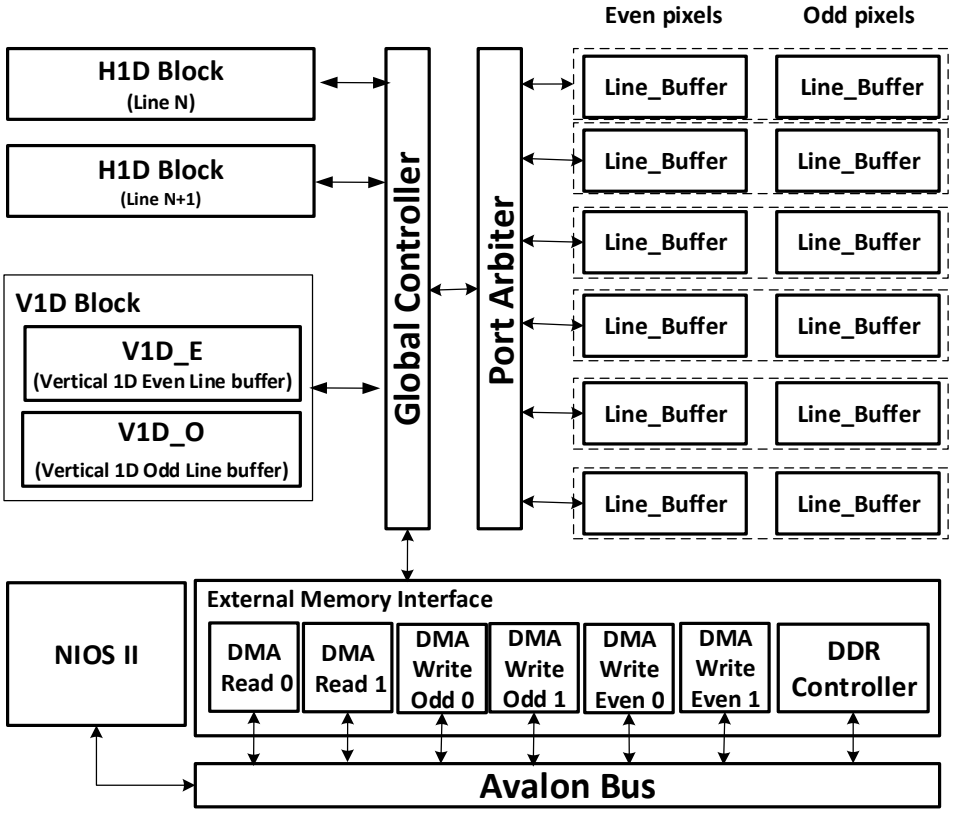


Fig. 5.1 The global architecture of the proposed unified 2D DWT.

5.2.1 Global Controller and Ports Arbiter

The global controller is a large central state machine which orchestrates all the units of the system. It knows the stage of DWT, the portion of the image under calculation and the current DMAs transactions. It also manages the exceptions, e.g., beginning and end of the image that require exceptional calculations. The concentration of intelligence in the central controller allowed to make simpler controllers for subsystems, just performing the computation tasks when the command comes from the central controller. The global controller manages all the data routing between computation units, LBs and DMAs via the port arbiter. The port arbiter is a large cross bar composed of multiplexers. These multiplexers are controlled by the global controller and route the data among different units.

5.2.2 Four Port Line Buffers and Data Concatenation

There are six line-buffers (LBs) which are required in order to run the horizontal transform in parallel with the vertical transform. To double the computation throughput, the LBs are

split into two banks containing even and odd pixels respectively. This split is done on-the-fly when the pixels are written into the LBs from the external memory. The Horizontal 1D DWT (H1D) and Vertical 1D DWT (V1D) units are the core computation units that perform jointly the 2D DWT.

Due to the limited data access, memories often become a key bottleneck while doing highly parallel computations. Although, the even and odd pixels are split into two LBs to double the throughput of computation, it is still not enough to obtain the unified parallel 2D DWT. Thus, we introduced a novel LB scheme and made two improvements exploiting the inherent benefits of the FPGAs that allow creating custom memory architecture.

The first and major modification done for the LBs was to make them 4-port memories without any overhead in memory size. This was achieved by exploiting the fact that since block memories on FPGAs are hard components, they can often be run much faster than the placed and routed logic of the design. As a consequence, we ran the on-chip dual port memories of FPGA at twice the speed of the computing logic that accesses data to/from them. Figure 5.2 shows the conceptual diagram of the architecture. A Phase Lock Loop (PLL) was used to generate phase-synchronous clk_x and clk_{2x}.

All the computation logic runs at clk_x and the dual port memories run at clk_{2x}. The port multiplexer reads and writes data on both rising and falling edges of clk_x. Hence, it virtually provides fully functional 4 memory ports. This 4-port access is the foundation element that makes it possible to do four key 2D DWT operations in parallel without having memory access bottleneck. These four operations are:

- DMA read
- horizontal DWT of an image line
- vertical DWT of an image column (made possible by six LBs)
- DMA processed data write

The second introduced feature was data concatenation. We store four pixels in each memory location instead of just one. This allows to further increase the computation speed by 4X as four pixels can be processed in parallel instead of one. Furthermore, using four pixels per location in line buffer makes more efficient data transfers between the DMA and the LBs. DMA data comes in bursts and DDR RAM controllers allow reading large data width from DDR RAM. In our architecture, we use DMA transaction size of 256 bits which corresponds to eight 32-bit pixels.

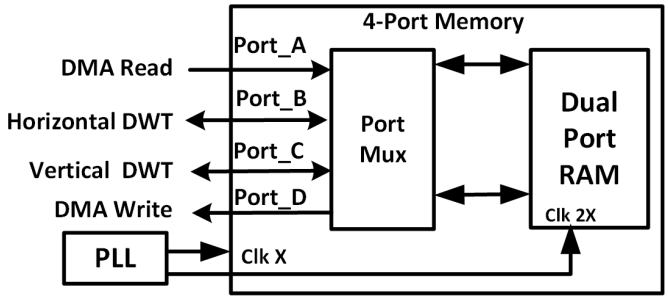


Fig. 5.2 The 4-port line buffer architecture.

5.2.3 Fixed-point Lifting Computation Units

The main goal of this architecture is to be integrated in medical applications, therefore, the precision analysis was taken in consideration. Since the fixed-point design is more popular than floating-point for hardware designers due its simplicity and speed, we chose it to implement the 2D DWT. One of the main goals in the design is the image quality. Consequently, we built a fixed-point DWT model in MATLAB using the fixed-point toolbox to measure the required precision to achieve high SSIM. We found the for 32-bit data word length it is needed to have 10-bit for the fractional part to calculate 6 stages of the 2D-DWT.

On the other hand, there are two types of the core processing units included in the proposed architecture: *(1)* $P1U1$ unit that performs the Predict $P1$ and Update $U1$ operations and *(2)* $P2U2S$ unit that performs $P2$ and $U2$ operations. In addition, it performs the lifting scheme scaling. Table 5.1 shows the DWT 9/7 filter coefficients in fixed-point that were used in the proposed architecture. Figure 5.3 and Figure 5.4 show the fixed-point core processing units of the lifting scheme that are used in the H1D and V1D units.

Table 5.1 Lifting coefficients in Fixed-point

Filter Coefficients	Real Value	Fixed-point*
α	-1.586134342	-1624
β	-0.0529801185	-54
γ	0.8829110762	904
δ	0.4435068522	454
$1/k$	0.8698644523	890
k	1.149604398	1177

* 32-bit word size, scaled with 10-bit (Scale factor = 1024)

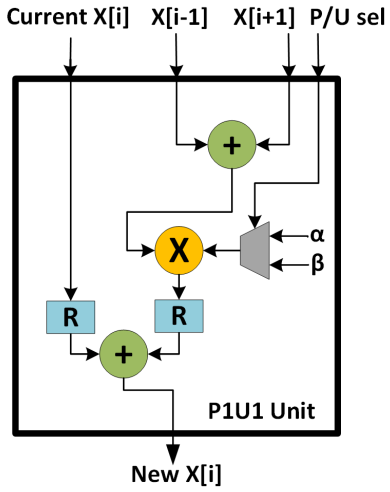


Fig. 5.3 Lifting Predict and Update computation unit (P1U1).

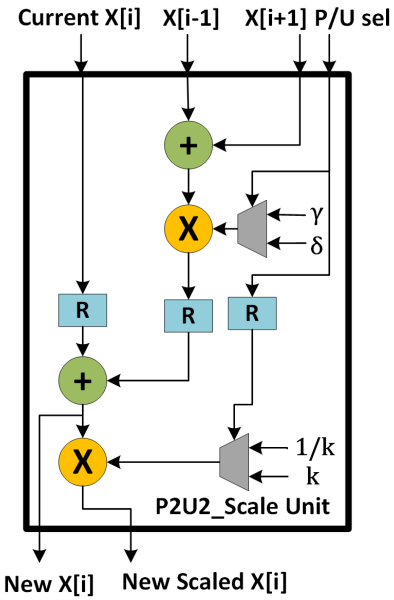


Fig. 5.4 Lifting Predict and Update computation unit with scaling (P2U2).

5.2.4 Horizontal 1D DWT

The horizontal 1D DWT unit (H1D) performs the lifting scheme 1D operations on the image lines. Figure 5.5 shows the LBs filled with even and odd pixels. In the proposed architecture every pixel is 32-bit. The depth of line buffer is equal to the width of the image (W) divided by eight (2×4) as there are two LBs for even and odd pixels and each location of LB has four pixels. The computation of Predict and Update steps requires data accumulation/processing

on a current pixel and its two neighbors. In other words, to compute Predict (odd pixels), we need to read its neighbors that are even pixels. With the split of pixels into even and odd pixels the computation becomes simple and straightforward.

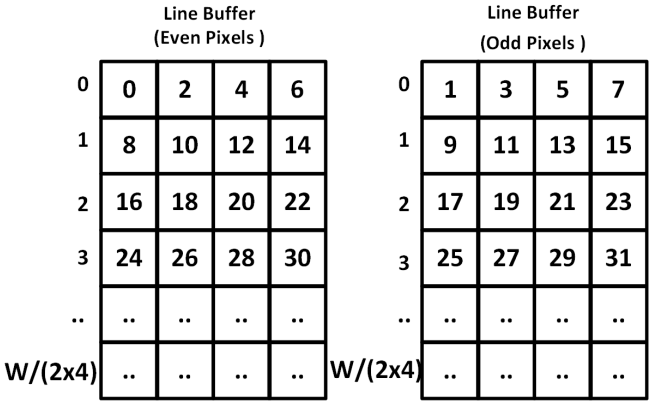


Fig. 5.5 Example of the data representation inside the Line Buffer (LB).

Since the Predict and the Update operations are data dependent on each other, they cannot be executed in parallel. The entire lifting scheme requires four sequential passes on the data: the first Predict (P1), first Update (U1), second Predict (P2), second Update (U2) and finally scaling of P2 and U2 values and splitting them in even and odd. In the proposed architecture, we removed these bottlenecks by exploiting the inherent parallelism of hardware. Data splitting is done on-the-fly while data-reading from the DDR RAM. It eliminates the LS final step that splits the results into even and odd. Having four pixels in each location in LB allows treating 4 pixels in parallel, since individually the Predict or the Update steps of different pixels can be parallelized. However, since it is impossible to run Predict and Update steps of a pixel in parallel, we did an architectural innovation in such a way that we run the pass one (P1 U1) and pass two (P2 U2) in parallel, in a way that doubles the computation throughput. Figure 5.6 explains these steps.

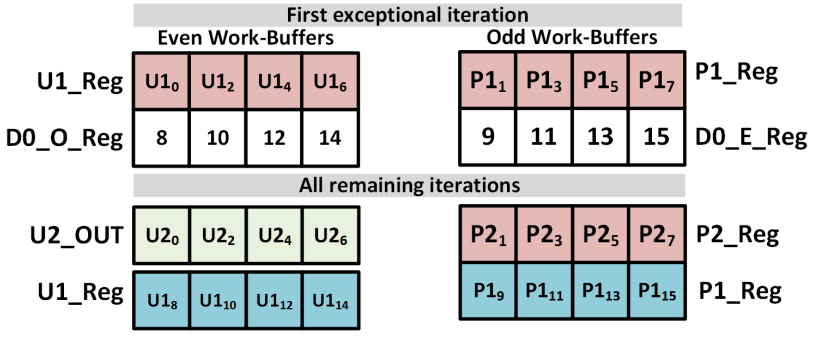


Fig. 5.6 Example of H1D Work Buffers data flow.

The computation of a line starts with an exception state called first exceptional iteration. In this step, only P1 and U1 located at the first position of even and odd LBs are computed. Once this is done, there is always one previous data till the end of the line to process P2U2 on memory index n and P1U1 on memory index n+1. In the exceptional initial iteration at memory index 0, the computation of P1 of pixels 1, 3, 5, 7 and U1 of pixels 0, 2, 4, 6 are computed. Next, the computation sequence remains same till the end of the line and is carried out as follows. We again start from index 0 of even and odd LBs. Since P1 and U1 at this index are already computed, we compute P2 and U2 at index 0 as the source data needed to compute P2 and U2 are available. In parallel of P2 and U2 computation, we compute the P1 (9, 11, 13, 15) and U1 (8, 10, 12, 14) at index 1 (i.e. $n+1$). This process continues till the end of the line, allowing to process 8 pixels (4 for P2, U2 and 4 for P1, U1) in each iteration.

Figure 5.7 shows a simplified block diagram of the horizontal 1D unit that performs all these parallel processing. The unit receives a start command from the global controller and informs the global controller when the processing is done. As we discussed in the 4-port memories section, there is a single port of memory assigned for Horizontal 1D Unit which requires to have an efficient memory reading/writing pattern to avoid any added latency during the operation. It uses internal work buffers and optimizes the memory access pattern exploiting the fact that the P1U1 and P2U2 units take multiple clock cycles to compute the data. Figure 5.8 shows the H1D simplified datapath for one pass of P1U1 and P2U2. The work-buffers have 3 parts: (1) `D0_Reg` and `D1_Reg` to buffer the received input data from the LB, (2) the `P1_Reg` and `U1_Reg` to buffer the P1U1 outputs, and (3) `P2_Reg` for the final P2U2 stage. It is required only P2 register to calculate U2, while the U2 results are written directly to the LB which save one register and one clock cycle.

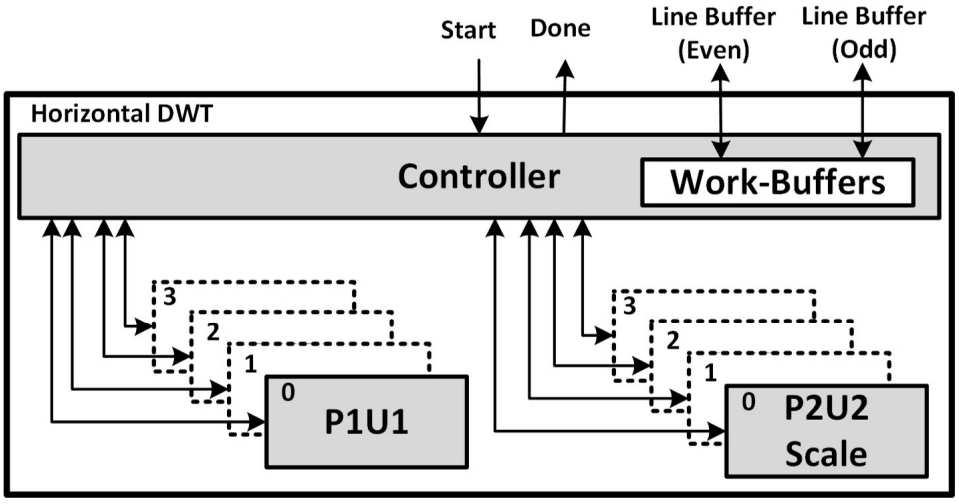


Fig. 5.7 Horizontal 1D Unit (H1D) Top Level.

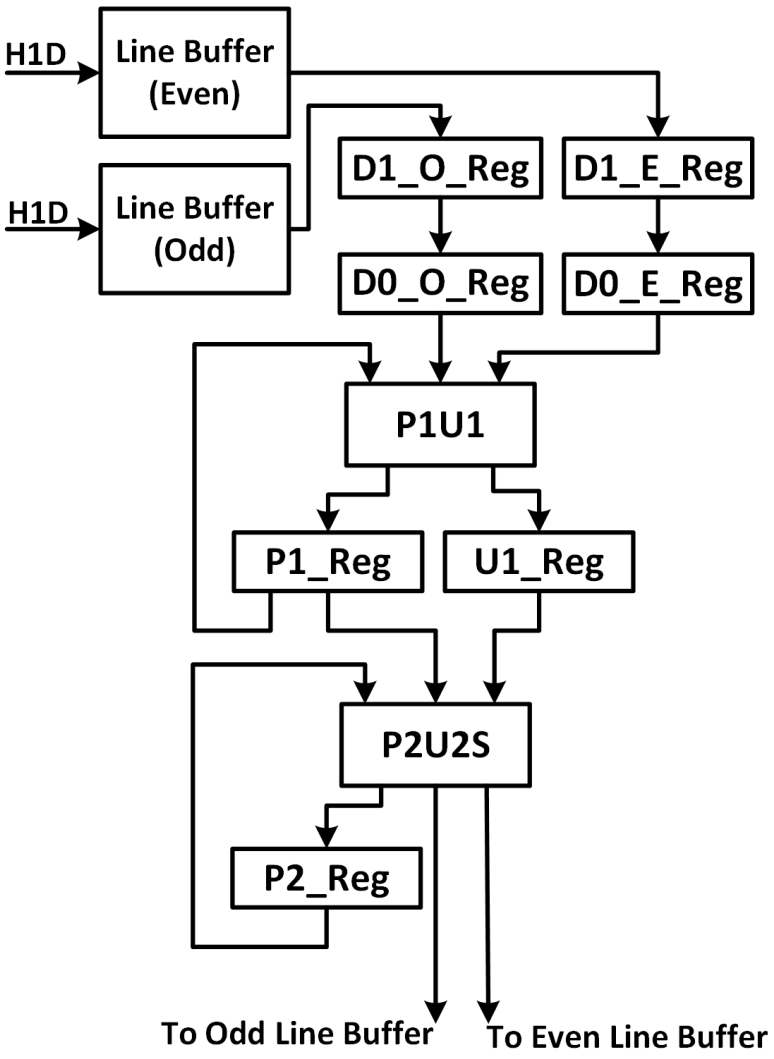


Fig. 5.8 H1D DWT Datapath.

Figure 5.9 shows the H1D operations sequence in term of clock cycles. Every clock cycle 4 pixels are read from the odd and even LBs in an alternate manner, for example: E0-6,O1-7. They are then passed to the Working buffers (WB) data `D_E_Reg` and `D_O_Reg`. The P1U1 units read the data from the WB to perform the Predict, then Update partition on the current pixel set. In parallel the P2U2 calculates P2 when P1U1 units start to calculate U1 achieving 100% hardware utilization of all the units. It can be seen that the operation latency of one iteration of data is nine clock cycles. But due to this heavy pipelining, each clock cycle outputs four pixels of even and odd buffers in an alternate manner. Since there are two H1D units running in parallel as was shown in the global architecture. This yields to a throughput of eight pixels per clock cycle.

CLOCK_CYCLE	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Even LB	R0	R0	R1	R1	R2	R2	R3	R3	R4	W0	R5	W1	R6	W3	R7
Odd LB	R0	R0	R0	R1	R1	R2	R2	R3	R3	R4	W0	R5	W1	R6	W2
DATA_OUT_E		E{0-6}		E{8-14}		E{16-22}		E{24-30}							
DATA_OUT_O			O{1-7}		O{9-15}		O{17-23}		O{25-31}						
D1_E_Reg			E{0-6}		E{8-14}		E{16-22}								
D0_E_Reg				E{0-6}		E{8-14}		E{16-22}							
D1_O_Reg				O{1-7}		O{9-15}		O{17-23}							
D0_O_Reg					O{1-7}		O{9-15}		O{17-23}						
P1U1_OUT						P1{1-7}		P1{9-15}		P1{17-23}					
P2U1_OUT							U1{0-6}		U1{8-14}		U1{16,22}				
P1_Reg							P1{1-7}		P1{9-15}		P1{17-23}				
U1_Reg								U1{0-6}		U1{8-14}		U2{16-22}			
P2U2_OUT									P2{1-7}		P2{9-15}		P2{17-23}		
P2U2_OUT										U2{0-6}		U2{8-14}		U2{16-22}	

- Rx: Read address x
- Wx: write adress x
- O{m-n}: Odd Line buffer outbuts from (m to n)
- E{m-n}: Odd Line buffer outbuts from (m to n)
- P1{m-n}: P1 outbuts of pixels (m to n)
- U1{m-n}: U1 outbuts of pixels (m to n)
- P2{m-n}: P2 outbuts of pixels (m to n)
- U2{m-n}: U2 outbuts of pixels (m to n)

Fig. 5.9 H1D Operations sequence and data flow.

5.2.5 Vertical 1D DWT

The Vertical 1D DWT (V1D) architecture performs the vertical 1D transform of the image columns. Figure 5.10 shows the simplified block diagram of vertical 1D (V1D) unit sub-blocks, V1D even and V1D odd respectively. The V1D unit receives the six inputs from six LBs via the global controller, which arbitrates the LBs in a ring fashion as new lines are read-in. The memory ports of even and odd parts of these LBs are connected to the V1D even and odd sub-blocks. This simplifies the internal controller of V1D as it works just on its six inputs (V0 to V5) in the fashion shown in Figure 5.12 and Table 5.2.

Initially, LBs are filled with six lines in two lines at a time fashion since there are only two read-DMAs. The two H1D units perform in parallel the horizontal DWT of the two lines as they are written in LBs by the read-DMAs. When the Global Controller orders execution of line 4 and 5 to the DMA read channels and H1Ds, it starts the V1D and DMA write. This is due to the fact that four LBs are now filled with processed H1D results and so the V1D processing can start as soon as data start to be available in LBs 4 and 5.

The V1D block is composed of two sub-blocks, the V1D for the even LBs and V1D for odd LBs respectively. The two blocks are reading the H1D units addresses on the even and odd LBs which is fed to them as a control input called (H1D Ref). As soon as data is processed by H1D on even and odd LBs the corresponding V1D sub-blocks use them to calculate the vertical 1D transform. In this manner, both H1D and V1D block run in parallel in a pipelined manner. The V1D architecture is designed to have the same iteration latency of the HID block, hence V1D does not slow down the H1D operations.

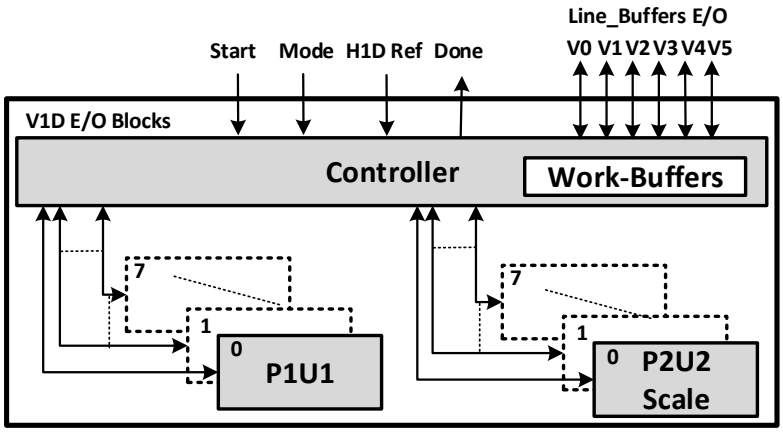


Fig. 5.10 Vertical 1-D (V1D) Sub block V1D_E/O.

Table 5.2 Vertical Transform Operations

Output	Mode-Initial-Exception	Mode-Normal	Mode-Final-Exception
V0	U1(V0,V0,V1) U2(V0,V0,V1)	n/a	n/a
V1	P1(V1,V0,V2) P2(V1,V0,V2)	U2(V1,V0,V1)	U2(V1,V0,V1)
V2	U1(V2,V1,V3)	P2(V2,V1,V3)	P2(V2,V1,V3)
V3	P1(V3,V2,V4)	U1(V4,V3,V5)	U1(V4,V3,V5) U2(V4,V3,V5)
V4	n/a	P1(V4,V3,V5)	P1(V4,V3,V5) P2(V4,V3,V5)
V5	n/a	n/a	n/a

The execution flow goes as follows: the global controller sends the start command to read-DMAs, H1Ds, V1D and write-DMAs at the same time. All these four units have data dependencies among each other. H1D waits until data is available to process in the line buffer. V1D waits if the data are processed by H1D so it can process that data for vertical transform. The write-DMAs wait for V1D to process the data, so they can write them into their internal FIFOs and burst them to external memory when sufficient data are available. This scenario is explained in the timing diagram in Figure 5.11, the latency of six clock cycles is for illustration. We can see that, all the four main operations of 2D DWT are executed in

parallel with latency among each other. The benefits of using 4-port line buffer memories are evident here.

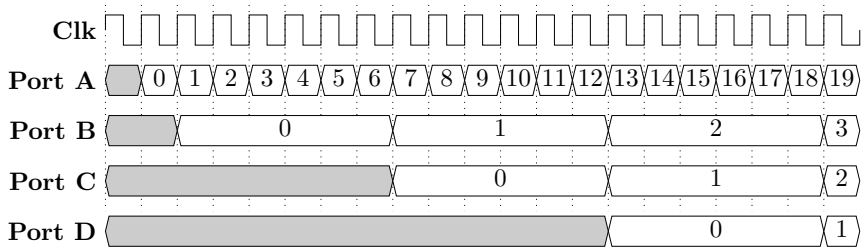


Fig. 5.11 The Parallel operation of DMA Read, Write, Horizontal and Vertical DWT via 4 port line buffers.

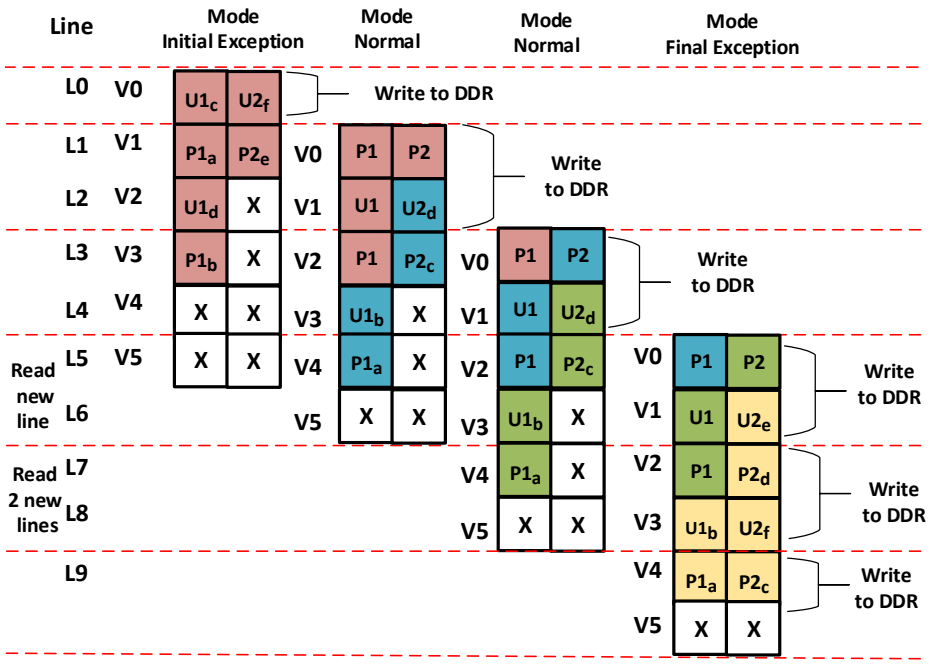


Fig. 5.12 Vertical transform operation sequence and data flow.

Figure 5.12 shows V1D operation principle. While H1D is working on a single dedicated line buffer, V1D gets the data from six LBs. There are three modes of operation: Mode-Initial-Exception, Mode-Normal and Mode-Final-Exception respectively. The Mode-Initial-Exception is only executed once at the beginning of the execution when the first six lines of the image are in process. The subscripts (a, b, c, d, e, f, g, h) with the operation Px/Ux indicate the order of sequential execution of different steps. The details of how these operations are done are shown in Table 5.2. During the execution of this mode, the

values at V0 (which in this case is Line buffer 0) are sent to external DDR RAM in parallel as soon as they are computed. The Normal-Mode is executed for the entire image until the last line of the image is left. The operations that are done and their order is shown in Figure 5.12. Finally, the Mode-Final-Exception processes the end exceptions. Like its counterpart Mode-Initial-Exception, this mode is also executed only once. In Figure 5.12, the Normal-Mode is executed only twice because the illustrative example has ten lines only.

Figure 5.13 shows the simplified datapath of V1D. The input of each P1U1 unit has three sources: Line-Buffer, feedback of P1U1 and VxR work-buffers. The registers VxR are introduced to synchronize and to balance the pipelining. The P2U2 units have the same data sources as the P1U1 units in addition to the output of P2U2. The V1D controller manages the data arbitration during the Predict and the Update operations.

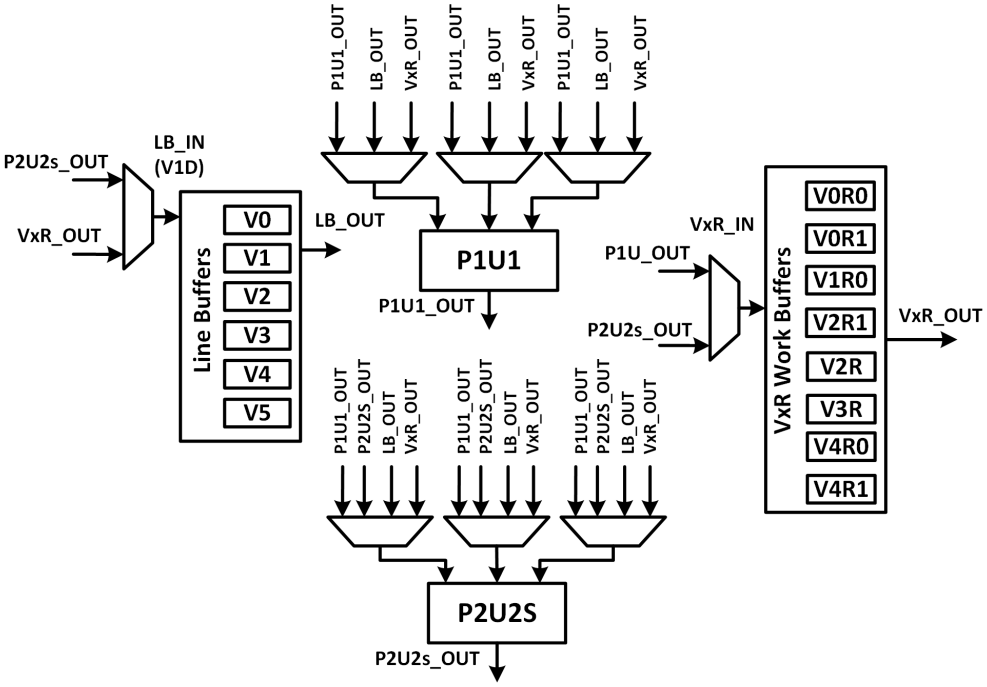


Fig. 5.13 V1D DWT Datapath.

Figure 5.14, Figure 5.15, and Figure 5.16 show the sequence operations of the different modes of V1D. Ri and Wi refer to LBs read and write address where i= 0,1,2,3,4, ... N-1 where N is the number of pixels in the line divided by four (since each LB location has four pixels). P1_i, U1_i, P2_i and U2_i refers to the operations P1, U1, P2 and U2 respectively for the location i of the line buffer. First the V1D receives the results of H1D which are stored in the six LBs. At the Mode-Initial-Exception the inputs are H1D results, so V1D calculates the operation in the following sequence: P1(V1,V0,V2), P1(V3,V2,V4), U1(V0,V0,V1), U1(V2,V1,V3), P2(V1,V0,V2) and finally U2(V0,V0,V1).

These results will be written back to the LBs. U2 value at this first line in these calculations is the final value and is not used by any subsequent calculations. In parallel to its writing back to line buffer the write DMA keeps writing it back to the external memory in the pipelined manner that was shown in Figure 5.11. When the processing of the first line is finished the corresponding line buffer will be ready to receive a new line from H1D unit. Most of computation time is dedicated for the Mode-Normal as the bulk of the lines fall in this mode. Hence the hardware is most optimized for this mode and achieves 100% utilization of the recourse as shown in Figure 5.15. It can be seen that P1U1 and P2U2 are performed continuously in a pipelined manner the operations P1(V4,V3,V5), U1(V4,V3,V5), P2(V2,V1,V3) and U2(V1,V0,V1) respectively. Finally, in the Mode-Final-Exception the V1D works on the last five lines to finish over all the remaining operations.

Clock Cycles	0	1	2	3	4	5	6	7	8	9	10	11
R/W Address	R0				R1		W0	W0	R2		W1	W1
V0		D0			U1_0	D1		U2_0	U1_1	D2		U2_1
V0R0						U1_0			U2_0	U1_1		
V0R1							U1_0			U2_0	U1_1	
V1		D0	P1_0	D0		D1	P1_1/P2_0			D2	P1_2/P2_1	
V1R0				P1_0				P1_1				P1_2
V1R1					P1_0				P1_1			
V2		D0				D1/U1_0				D2/U1_1		
V2R							U1_0				U1_1	
V3		D0		D0/P1_0		D1		P1_1		D2		P1_2
V3R					P1_0				P1_1			
V4		D0				D1				D2		
V4R0												
V4R1												
V5												

Fig. 5.14 V1D Mode-Initial-Exception operation sequence and data flow.

Clock Cycles	0	1	2	3	4	5	6	7	8	9	10	11
R/W Address	R0		R1		R2	W0	R3	W1		W2		W3
V0		P2_0		P2_1		P2_2		P2_3				
V0R0			P2_0		P2_1		P2_2		P2_3			
V0R1				P2_0		P2_1		P2_2		P2_3		
V1		U1_0		U1_1		U1_2/U2_0		U1_3/U2_1		U2_2		U2_3
V1R0			U1_0		U1_1		U1_2		U1_3			
V1R1				U1_0		U1_1		U1_2		U1_3		
V2		P1_0		P1_1	P2_0	P1_2	P2_1	P1_3	P2_2		P2_3	
V2R			P1_0		P1_1	P2_0	P1_2	P2_1	P1_3	P2_2		P2_3
V3		D0		D1/U1_0		D2/U1_1		D3/U1_2		U1_3		
V3R			D0		U1_0		U1_1		U1_2		U1_3	
V4		D0	P1_0	D1	P1_1	D2	P1_2	D3	P1_3			
V4R0				P1_0		P1_1		P1_2		P1_3		
V4R1					P1_0		P1_1		P1_2		P1_3	
V5		D0		D1		D2		D3				

Fig. 5.15 V1D Normal-Mode-operation sequence and data flow.

Clock Cycles	0	1	2	3	4	5	6	7	8	9	10	11	12	13
R/W Address	R0				R1	W0	W0	W0	R2	W1	W1	W1	R3	W3
V0		P2_0				P2_1				P2_2				P2_3
V0R0			P2_0				P2_1				P2_2			
V0R1				P2_0		P2_0		P2_1				P2_2		
V1		U1_0			U1_1	U2_0				U2_1				U2_2
V1R0														
V1R1														
V2		P1_0			P2_0	P1_1			P2_1	P1_2			P2_3	
V2R						P2_0				P2_1				
V3		D0		U1_0		D1		U1_1/U2_0		D2		U1_2/U2_1		
V3R					U1_0				U1_1	U1_1				
V4		D0	P1_0			D1	P1_1/P2_0			D2	P1_2/P2_1			
V4R0														
V4R1														
V5														

Fig. 5.16 V1D Mode-Final-Exception operation sequence.

5.2.6 External Memory Interface

The external memory interface unit is composed of several read/write DMAs operations along with the external DDR RAM controller. This unit allows the access to external DDR memory that holds the source image and all the computed results of DWT stages.

The DMAs of the proposed architecture are managed adaptively by the global controller. The read DMA channels read the lines of the source image and fill the LBs. During processing the unified 2D DWT, an innovative manner is adapted to write back the data in the external DDR RAM. Figure 5.17 explains the proposed idea. After the 2D transform, the image is divided into four sub-bands, which separate the low and high pass coefficients. The splitting of these components is done via the even odd splitting of horizontal and vertical 1D transforms. The horizontal transform leads to the splitting of even and odd coefficients in a line while the vertical transform leads to the splitting of even and odd lines. These four splitting are shown in the middle of Figure 5.17 with the standard terms used in DWT literature LL, HL, LH and HH.

To write these values efficiently back to DDR RAM, we wrote these values separately in a linear fashion. The LL sub-band contains even coefficients of even lines (called EE). The HL sub-band has odd coefficients of even lines (called EO). In a similar manner, LH has even coefficients of odd lines and HH odd coefficients of odd lines (called OE and OO respectively). The global controller manages the addresses of EE, EO, OE and OO. The use of separate even and odd writing DMAs allow writing in parallel the contents of even and odd LBs. The global controller knows which lines in the image are in process by the V1D and assigns the proper EE, EO, OE and OO addresses respectively on-the-fly.

The writing of the four sub-bands in the form shown on the right of Figure 5.17 instead of the classical form shown in the middle, gives two key benefits. Firstly, writing data linearly leads to more efficient bursts of DMAs as there is no address offset jumps to handle in DMAs controllers in the writing/reading of the sub-band. Secondly the writing in the modified form

is beneficial for the processing of the next stage of DWT. In fact, only the EE sub-band will be used for next stage processing leading to more efficient reading. Furthermore splitting all the sub-bands in this form helps to process efficiently the compression of these sub-bands.

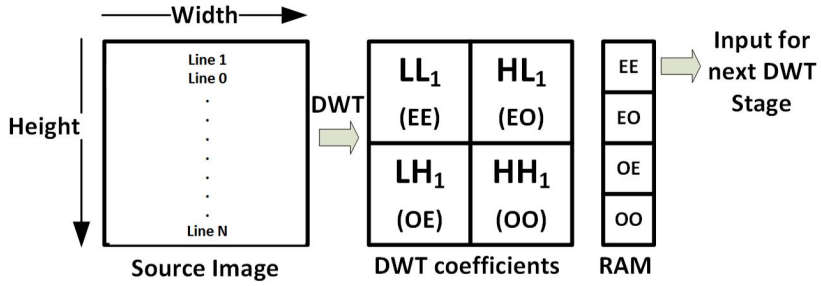


Fig. 5.17 The proposed DWT coefficients addressing.

5.3 Results

The proposed architecture was written in generic VHDL/Verilog, implemented and tested on an Altera StratixIVGX230 based DE4 board from Terasic. The frame rate throughput reaches up to 120 full HD frames per second. In the presented results shown in Table 5.5, the architecture runs at 125 MHz and the DDR2 RAM runs at 250 MHz.

5.3.1 Resources on DE4 FPGA board

Table 5.3 Altera Stratix IV GX230 resources utilization for 1080p.

Unit	ALUTs	Flip-Flops	DSPs	BRAM bits
Line Buffer × 12	265×12	384×12	0	30,720×12
Global Controller& Arbiter	2,291	564	0	0
NiosII & External Memory Interface	10,805	12,065	0	885,920
Horizontal DWT	2,448	3,552	104	0
Vertical DWT	1,947	2,674	104	0
Total	20,671	21,489	208	1,254,560
Available	182,400	182,400	1,288	14,625,792
Percentage	11%	11.8%	16%	8%

Table 5.3 lists the resources used on the Stratix IV GX230 FPGA on DE4 board. The synthesized results are obtained for full-HD 1080p resolution frame size. It shows that the designed architecture consumes only 16% DSPs and 8% of on-chip memories of the targeted FPGA.

5.3.2 The External DDR RAM Bandwidth Gain

The bandwidth is reduced by factor 2X compared to classical implementation since we only read the pixels once and write the results back to the external DDR RAM. In classical implementation this is done twice, once for horizontal and then again for vertical which it added a disadvantage since vertical transform requires reading and writing data in columns from external DDR RAM which is highly inefficient. So in conclusion the bandwidth gain is even more than 2X since we have eliminated the need of column-wise external DDR RAM accesses.

5.3.3 Architecture Scalability

The proposed architecture is highly scalable. The throughput can be easily scaled up by increasing the number of P1U1 and P2U2S units for both horizontal and vertical DWT units. The presented architecture is the base architecture with S (Scalability factor) equal to one. The architecture is also scalable to process an image larger than full HD. For example, the line buffer sizes can be increased (doubled) for untiled processing of 4K video. Furthermore the architecture is coded in generic Verilog so it can be ported easily to FPGAs of other vendors or ASICs. The only things needed to port to other FPGAs or ASICs will be to use PLLs of those devices and change the DMAs front-end in case of using other bus protocols.

The presented results can be scaled up in terms of maximum operating frequency. The core computation blocks H1D and V1D can run at 219 MHz on the used FPGA. This frequency was not used primarily due to two reasons. Firstly the DDR2 controller of Altera on Stratix IV only provides half rate clock for the logic. The DMAs had timing closure problems at 200MHz (half of 400MHz) if we ran the DDR2 at its maximum frequency of 400MHz. The DDR2 controller half rate clock that is synchronous to the physical memory clock is vital to have low latency DMA accesses. The maximum frequency used to run the DMAs was around 130MHz. In view of these technical bottlenecks of the targeted FPGA we adapted the experimental frequency to 125MHz. That permitted us to run all the logic and DMAs with a single synchronous clock. The newer FPGAs with DDR3 and DDR4 controllers provide further flexibility, such as quarter rate clock which will improve this issue. Furthermore, the proposed architecture can easily be upgraded to asynchronous where the logic can run faster or slower (to save power) than the DMAs.

Table 5.4 architecture Scalablity

		Full HD		4K UHD	
S	FPS	F_Exp[b] 125	F_max[c] 219	F_Exp 125	F_max 219
S = 1	Theoretical	361.8	633.9	90.45	158.49
	Actual [a]	120.2	n/a	30.05	n/a
S = 2	Theoretical	723.6	1267.8	180.9	316.95
	Actual [a]	n/a	n/a	n/a	n/a

[a] FPGA prototyping results including DMAs latency, [b] F_Exp: Experimental frequency in MHz, [c] Maximum core logic frequency in MHz.

Table 5.4 illustrates some of these different scaling options. The proposed architecture has a theoretical throughput of 361.8 fps and on actual prototyping we obtained a throughput of around 120 fps. Although this value is much lower than the theoretical fps, it still indicates the high efficiency of the pipelined mechanism of the proposed architecture. The reason for this lower actual value can easily be explained as follows: As was presented in the previous sections, the DMAs read/write eight pixels from/to the external memory in every burst transaction of data. If it is theoretically assumed that there is zero delay between each burst, which means that the DMAs are capable of having a speed of eight pixels per clock cycle for reading and eight pixels per clock cycle for writing. Hence, in total the external memory access for reading and writing a frame under treatment will require two clock cycles for reading and writing the eight pixels. This value is double than the CPP of the core hardware which has a throughput of eight pixels per clock cycle. So even in the ideal case the DMAs can provide a frame rate of around 180fps (361.8/2). Having all this data, it is evident that a real value of 120fps achieved on the FPGA board compared to 180fps theoretical DMAs bandwidths is quite an efficient value. As there are multiple added latencies of switching among the six DMAs running in parallel that are used in the proposed architecture. Furthermore, four out of these six DMAs are for writing the data and write DMAs operation in the Altera Avalon bus are not pipelined. This further solidify the efficient pipelining of the proposed architecture which does DMA read, HID, VID and DMA write in parallel.

If the proposed architecture runs at 219 MHz which is the maximum possible frequency of the computation units, the theoretical throughput will be 633.9 fps full HD images. The table also outlines the relative comparison to 4K Ultra High Definition (UHD) images computation. It can be seen that proposed architecture can easily give 158fps 4K UHD hence is well capable of modern 4K video processing. The table also estimates the throughput increase if the P1U1 and P2U2S blocks are doubled (S=2). It is evident that the architecture even with $S = 1$ (proposed work) is well capable of the road-map of 4K and upcoming 8K video processing.

Furthermore in the current work the external memory bandwidth was limited due to DDR2 on FPGA board. In the latest and upcoming FPGA boards the DDR4 memory runs at much higher frequencies that will easily lift the bandwidth barriers observed in this work between the core processing block and DMAs throughput.

5.3.4 Comparison with Existing Works

Table 5.5 lists the features of the proposed architecture compared to the best known existing LS DWT works in the peer-reviewed literature on FPGAs and ASICs. We compared the results based on key aspects that are used in literature to cross analyze pros and cons of different architectures. One of the main comparison factors to compare architectures is the cycles per pixel (CPP). CPP indicates the architecture efficiency. It can be seen that the proposed architecture with just the base specifications discussed in the previous section outperforms the existing works. Furthermore, it is important to note that most of the articles have not taken into account the importance of external memory access bottlenecks hence their proposed results can easily become compromised if real aspects of external memory access are considered.

Table 5.5 Comparison with similar Lifting Scheme DWT architectures

Features	**Aziz *et al.* [62] 2012**	**Sameeen *et al.* [66] 2012**	**Hu *et al.* [58] 2013**	**Hsia *et al.* [69] 2013**	**Darji *et al.* [63] 2014**	**Darji *et al.* [65]**[e] **2014**	**Proposed**
Technology	FPGA Spartan 3	FPGA Stratix III	ASIC 90nm CMOS	ASIC 180nm CMOS	ASIC 180nm CMOS	FPGA Virtex-4	FPGA Stratix IV
Frame size	512×512	1920×1080	512×512	256×256	256×256	1920×1080	1920×1080
DWT Levels	Up to 5	Up to any	Up to 3	1	1	Up to 5	Up to 6
DWT filter	5/3	9/7 & 5/3	9/7	9/7 & 5/3	9/7	9/7 & 5/3	9/7
Sys Freq.[a]	221.44	133.3	50	100	100	100	125
DDR Freq.[a]	n/a	266	n/a	n/a	n/a	n/a	250
bpp	8-bit	32-bit	8-bit	16-bit	n/a	16-bit	32-bit
CPP [b]	**1**	**1.55**[g]	**0.5/S** [f]	**0.75**	**0.5**	**0.5**	**0.125/S**
Computation Cycle	$N^2 + 93N/16$	n/a	$N^2/2S$ [f]	$(3/4)N^2 + (3/2)N + 7$	$N^2/2$	$N^2/2 + \sum N/2$	$1MN/8S$
Fps [c]	423[h]	24.04	n/a[i]	n/a	n/a	n/a	120/361[j]
CPD [d]	$2T_a$	n/a	$T_m + T_a$	$2T_m + 4T_a$	T_m	$T_m + T_a$	$T_m + T_a$
Line Buffers	n/a	n/a	3N + 341S/8 [f]	4N	4N	4N	6N
Adders	2	n/a	116 at S=8	16	16	16	68
Multipliers	0	n/a	188 at S=8	0	10	10	54
Scalable Through-put	No	No	Yes	No	No	No	Yes

[a] in MHz, [b] CPP: Clock cycles per pixel (lower is faster), [c] FPS: Frame per second, [d] CPD: Critical path delay, [e] PMA 9/7 architecture, [f] S: Strip size, [g] CPP for image size of 1024×1024, [h] 423 fps 512×512, and for 53 fps for HD, [i] 385 fps for HD at 50MHz, S=8, [j] theoretical: 361fps, actual: 120fps on the DE4 board with Stratix-IV FPGA.

Looking at the supported frame size, the proposed architecture is scalable to any resolution, while some other works deal with small size images. In terms of architecture programmability, the proposed work supports only 9/7 DWT while [66], [69] and [65] supports both 9/7 and

5/3 DWT. The architecture [62] achieves frame rate of 423fps for 512x512 image (equal to around 53fps for full HD) but does not take into account the external ram access and is done for 5/3 DWT which has a simpler datapath compared to 9/7 DWT. We found that [66] is the only work that considered in details the issues of external memory access like proposed work. In terms of CPP proposed architecture is 12X faster and in the actual implementation it is 5X faster. The architecture [58] was implemented on 90nm ASIC, it has very high throughput due to novel stripe based data computation of the image. However the stripe based computation of data requires column-wise access of external RAM which is highly inefficient. The authors have not discussed about this issue, and hence, it is hard to achieve the claimed results in a real image processing system where a shared external memory is used by all the components of image/video processing. The proposed architecture outperforms [69], [63],[65] in terms of CPP. In terms of the critical path delay (CPD) most of the architectures have a CPD of $T_m + T_a$, where T_m and T_a are the delays of multiplier and adder respectively. In terms of fps our result of 120fps is a realistic result based on the FPGA prototype taking into account all the latencies of DMA accesses.

5.4 Integration with Smart-EEG Prototype

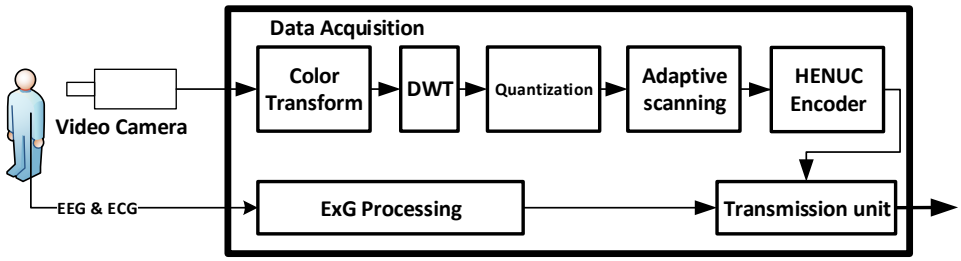


Fig. 5.18 The fundamentals of Smart EEG telemedicine system.

The designed architecture has been integrated with the HENUC encoder architecture developed in [48]. Thus, the complete video processing system based on WAAVES encoder was validated. A snapshot of the acquired coded and decoded video frame is given in Figure 5.19. The video EEG prototype is shown in Figure 5.20. The experimental implementation includes multiple devices. A physiological signal acquisition board developed by ETIS lab to acquire raw physiological samples using medical certified ADS 1299 and ADS 1298 analog to digital converters (ADC) from TI. This board is linked to the Altera DE4 development board using an SPI protocol through a GPIO connection interface. In parallel, a Basler camera (acA2000-340kc) acquiring a full-HD video at 100 fps is connected through cameraLink cable to the DE4 board. This board contains an Altera StratixIVGX230 FPGA in which we have implemented a low level synchronization IP. That allows to trigger data acquisitions as well

as WAAVES compression algorithm. The board sends on-the-fly the acquired data over PCIe port to the computer host machine. The computer uses an Intel SSD (P3600) as a buffer memory to deal with the high data bandwidth and to store the compressed data stream. This stream is formatted and encapsulated before being finally sent over the network [70].

Fig. 5.19 Example coded and decoded frame.

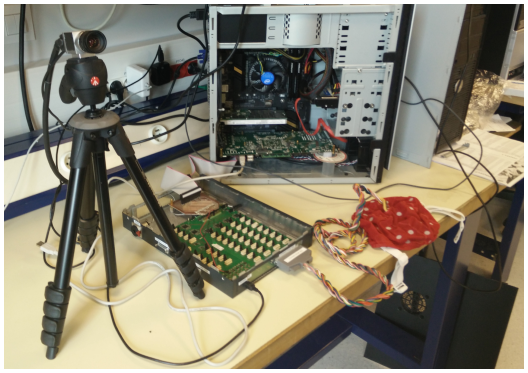


Fig. 5.20 Smart EEG experimental prototype.

5.5 Conclusion

In this chapter we have proposed a novel DWT architecture with a high throughput of 8 pixels per clock cycle, and achieving a processing speed up to 361 Full HD frames per second. The key innovations that lead to this high throughput are: *(1)* a unified 2D DWT computation architecture that performs both horizontal and vertical transform simultaneously in a novel fashion; *(2)* 4-port line buffers allowing to process DMA reading, DMA writing, Horizontal

and Vertical DWT in parallel and in a pipelined fashion. That led to eliminate the inefficient reading or writing columns of an image from/to DDR RAM. This is achieved via unified 2D DWT and adaptive DMAs addressing. Furthermore, a 2X reduction in the required DDR RAM bandwidth has been achieved compared to the classical solution. We highlighted the experimental results on Altera's Stratix IV GX FPGA using DE4 board and showed that by running the logic at just 125 MHz and the DDR2 RAM at 250 MHz, we were able to achieve 120 fps 1080p throughput. We demonstrated that the architecture is scalable and outperforms existing state of the art. Finally, proposed architecture is designed to support the LNS-DWT, since it requires only to attache the logarithmic computation units Predict/Update (P1U1 and P2U2) and a simple controller to be interfaced with the H1D/V1D local controllers.

Chapter 6

Conclusion and Future work

Contents

6.1 Review of the Research Questions

The main aim of this thesis is to provide solutions to the most common problems that faces the medical image compression domain. Specially image quality and the compression speed on the embedded systems. These issues were the motivations of this thesis, so we raised the following research questions:

- **How to improve the image quality?**
- **How to provide a high-speed image compression on embedded systems?**

To answer the first question, we explored the state of the art image compression algorithms to analyze the sources of quality degradation. Since the compression algorithms consist of three basic steps: the forward transform, the quantization, and the encoding, it was evident that the primary source of quality degradation was the quantization, so, we raised another question:

- **How to limit the quantization effect?**

Also, since the forward transform which is the DWT in our case, is computed in the real-numbers domain, we raised another interesting question:

- **Does switching the arithmetic in the DWT from the liner to the logarithmic domain has an effect on the image quality?**

According to state of the art arithmetic techniques, there are three ways to perform computation in the digital signal processing: the floating-point, the fixed-point, and the logarithmic arithmetic.

The logarithmic arithmetic was introduced as an alternate to the floating point to accelerate computations on embedded systems. In This work however we used logarithmic arithmetic compression method and to the best of our knowledge this is the only work that uses this technique and no similar work exists and hence it is considered a novel work.

For the second question related to the compression on embedded systems, the challenge was to provide a hardware implementation of the DWT with a speed up to 100 frame/s, to be integrated with the image compression chain on an FPGA platform, as a part of the Smart-EEG project to fulfill its requirement. After a literature survey, we found no other architecture and work with the same required speed (bearing in mind the DDR memory latency at the same time) leading us to the following question:

- **How to achieve the 100 frame/s with taking in consideration the DDR latency?**

6.2 Thesis Conclusion

To answer the raised research questions, we developed an LNS library to be used as a tool in this work. The library has two advantages: the virtual zero, and the sign flag. Introducing the virtual zero as a compatible feature of image compression algorithms introduced in this work gave us the advantage of efficiently encoding the proceeded images. In addition, the virtual zeros enhanced the library mathematical operations efficiency due to handling the logarithmic operations exactly as the operations of the linear domain. The library also, solved the sign ambiguity problem in the logarithmic domain by introducing the sign flag.

The LNS library was validated by evaluating the multiply-and-accumulate (MAC) operation in the both domains, then calculating the error between the two outputs. The validation results showed that the error was very small, roughly equal to 7×10^{-8} after performing 1000 iterations. The results indicated that the library is reliable when using it as an alternative to the floating point arithmetic in signal processing applications.

The first objective of the LNS library in this work was to implement the DWT in LNS to be used in image compression. The LNS-DWT was validated and compared with the linear DWT. The error between them was round 7×10^{-10}, which is considered a very small value and it confirmed that LNS achieved a precision near to that of the floating point arithmetic.

The second objective of the library was to implement the quantization process in LNS domain. We proposed a novel quantization method based on the logarithmic domain called LNS-Q. This method uses the scaling operation to convert the data into discrete values. Using LNS-Q revealed that it minimized the quantization error compared to the linear quantization. That is because the scaling did not lead to a high loss in the data, in contrast to the division operation in the linear quantization. The LNS-Q and the LNS-DWT were evaluated together by applying them on different medical image modalities. The image quality of the reconstruct image was measured in PSNR and SSIM. The results showed that LNS-Q has an interesting effect on the quality compared to the linear quantization. For example, when quantizing the linear DWT coefficient using the linear quantization with q =10, the PSNR was equal 87.7 dB and the SSIM was equal 0.91483822, while using LNS-DWT with LNS-Q and SC = 1000 for the same image, it yielded PSNR = 140.3 and SSIM = 0.99999887. This confirms that the image quality was improved by limiting the quantization error.

After the validation of the LNS-DWT and the LNS-Q, we integrated them with WAAVES to have a new compression scheme called LNS-WAAVES. The objective of the integration was to measure the compression efficiency using the logarithmic arithmetic. The results of the basic integration showed that, although LNS-WAAVES achieved a high image quality, it has a limitation on the compression ratio. The main reasons of the limitations were: the number of the zeros in the quantized LNS-DWT smaller than those in the quantized linear DWT. Also, the quantized coefficients were smoothly distributed around the zero, while in the LNS-DWT was not the same. Moreover, the dynamic range of the all LNS-DWT coefficients was increased when a large scale factor was applied, while in the linear DWT the quantization process decreased the dynamic range because of the division operation.

Having those issues motivated us to propose a novel compression scheme based on hybrid DWT. The LNS-WAAVES replaced the pure LNS-DWT with the Hybrid-DWT, which divided the DWT coefficients sub-bands into two parts: logarithmic and linear sub-bands that combined the advantages of the both domains. The number of the coefficients in each domain were controlled by introducing a new compression factor called the number of linear DWT levels (NL). This compression scheme improved the image quality significantly compared to the linear domain, in addition to achieving a better compression ratio. The reason for that, the LNS part is responsible for preserving the quality, while the linear part responsible for improving the compression ratio. Also, the result showed that when using larger quantization steps, the quantization error is still smaller compared to the linear scheme.

The compression parameters gave the LNS-WAAVES more flexibility by having a trade-off between the image quality and the compression ratio. The impact of changing the base of

the logarithm was investigated as well. The study showed that using a higher base of the logarithm improves the compression ratio, but the quality was degraded. Since according to the rule of the change of the logarithm base, the calculation of the higher logarithmic base, it requires to be divided by a large number, which yields a similar effect of the quantization error. Thus, the best logarithmic base value was b = 2 that gave the best quality.

For the NL parameter, the results showed that, when using NL = 3 or NL = 4, it yielded the best results for both image quality and the compression ratio. This is mainly because these NL values achieved the balance between the two parts in the DWT coefficients. The results showed that, LNS-DWT based on Hybrid-DWT was able to achieve improvement in the quality by a percentage of 8% up to 34% compared to WAAVES and by a percentage of 10% up to 49% better than JPEG2000 depending on the different compression parameters.

Further improvement was addressed for the logarithmic compression by applying post-DWT processing using the proposed dynamic range reduction filter (DRR). This filter was able to increase the number of zeros by limiting the dynamic range of the LNS-DWT coefficients. Thus, they can be encoded efficiently.

Finally, for the compression on embedded systems, we proposed a novel DWT architecture with a high throughput of 8 pixels per clock cycle, achieving processing speed up to 361 Full HD frames per second. The key innovations that lead to this high throughput are: *(1)* a unified 2D DWT computation architecture that performs both horizontal and vertical transform simultaneously in a novel fashion; *(2)* 4-port line buffers allowing to process DMA reading, DMA writing, Horizontal and Vertical DWT in parallel and in a pipelined fashion. Consequently, this lead to the elimination of the inefficient reading or writing columns of an image from/to DDR RAM. This was achieved via unified 2D DWT. Furthermore, a 2X reduction in the required DDR RAM bandwidth was achieved compared to the classical solution. We demonstrated that the architecture is scalable and outperforms existing state of the art.

From the this conclusion, the thesis contributions can be summarized as the following:

- **A novel study of using a logarithmic number system (LNS) in the image compression.**
- **Development of an LNS library compatible with image compression algorithms.**
- **Novel compression scheme based on the logarithmic discrete wavelet transform.**
- **Finally, a high-speed DWT architecture for embedded systems.**

6.3 Future work

Image compression based logarithmic arithmetic is a promising research area. The work presented in this thesis is considered the first in the image compression domain that employs logarithmic arithmetic. The main objective of the study was to answer the question of the possibilities to improve the quality of the image compression when switching from the linear to the logarithmic arithmetic. Since the initial results of the study were promising, we are motivated to continue the study and expand it by exploring more elements.

The first planned work is to provide an end user application. That is because doctors and radiologists need to use a simple interface to compress the medical exams. The next step is to develop a compression software tool with a user-friendly interface. The objective of this tool is providing a flexible control to the compression parameters without involving the doctors with the complexity of the decision related to the quality and compression ratio which is based on choosing the parameters (FTH, B, SC, NLF, q). That will give the freedom to compress with desired trade-off and compromise, according to what the doctors need.

Another interesting point is to investigate the quantization with different scale factors for each DWT level within the same image. That raises a question: what is the impact of using the different scale factors on the compression ratio and the image quality?

Next, we are planning to explore compressing the medical image sequences using LNS-WAAVES. That is because the sequence images have interesting characteristics; most of the information in the images in the sequence are constants, while the remaining parts are changing from one image to the other in the sequence. Hence, this can be utilized by compressing the difference between the moving parts in the sequence and keeping the first image as a reference. In addition, we plan to explore the WAAVES HENUC encoder and how to switch the encoding algorithm into the logarithmic encoder to be more efficient and compatible with the logarithmic data. Also, WAAVES uses a block called sorting, which is responsible for sorting the data to be compressed efficiently, we are planning to investigate how to adapt it with the logarithmic data and utilizing the small dynamic range achieved by LNS-DWT.

For the hardware part, there are two main goals: the first one is building a logarithmic computation unit and integrating it with proposed architecture. The second goal is to adapt the architecture to perform the LeGall 5/3 in order to to be able to run the DWT lossless compression mode as well. In a longer term, we see that the scalable nature of this architecture is well positioned on the roadway to 4K and upcoming 8K video processing that will be easily applicable due to higher bandwidth DDR4 RAM and larger FPGAs with more DSPs and on-chip memory resources available on future FPGAs.

Publications

International Conferences

1. **Mohammed Shaaban Ibraheem**, Syed Zahid Ahmed, Khalil Hachicha, Sylvain Hochberg, and Patrick Garda: "A low ddr bandwidth 100fps 1080p video 2d discrete wavelet transform implementation on fpga". *In Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '16, pages 274–274, New York, NY, USA, 2016. ACM.

2. **M. S. Ibraheem**, S. Z. Ahmed, K. Hachicha, S. Hochberg, and P. Garda. "Medical images compression with clinical diagnostic quality using logarithmic dwt". *In 2016 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*, pages 402–405, Feb 2016.

3. **Mohammed IBRAHEEM**, Khalil Hachicha, Syed Ahmed, Sylvain Hochberg, and Patrick Garda. "Logarithmic discrete wavelet transform for medical image compression with diagnostic quality". *In Proceedings of the 5th EAI International Conference on Wireless Mobile Communication and Healthcare, MOBIHEALTH'15*, pages 272–275, ICST, Brussels, Belgium, Belgium, 2015. ICST (Institute for Computer Sciences, Social- Informatics and Telecommunications Engineering).

4. I. Dhif, **M. S. Ibraheem**, L. Lambert, K. Hachicha, A. Pinna, S. Hochberg, I. Mhedhbi, and P. Garda."A novel approach using waaves coder for the eeg signal compression". *In 2016 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*, pages 453–456, Feb 2016.

International Journals

1. L. Lambert, J. Despatin, I. Dhif, I. Mhedhbi, **M. Shaaban Ibraheem**, A. Syed-Zahid, B. Granado, K. Hachicha, A. Pinna, P. Garda, F. Kaddouh, M. Terosiet, A. Histace, O.

Romain, C. Bellet, F. Durand, J.P. Commes, S. Hochberg, D. Heudes, P. Lozeron,and N. Kubis. "Telemedecine, electroencephalography and current issues. smart-eeg: An innovative solution". *European Research in Telemedicine*, 4(3):81 – 86, 2015.

International Journals under peer-review

1. **Mohammed Shaaban Ibraheem**, Khalil Hachicha, Syed Zahid Ahmed, Laurent Lambert, and Patrick Garda. "A scalable high throughput 2d dwt architecture for a medical application". *Journal of Real-Time Image Processing*, submitted: Jan 2017.

National Workshops

1. **M. Shaaban Ibraheem**, Khalil Hachicha, Imen Mhedbi, Sylvain Hochberg, Patrick Garda , S. Zahid Ahmed. "Logarithmic-based dwt for medical images compression". *In Colloque de la Fédération d'Électronique, Thème : Internet des objets pour les applications biomédicales*, Issy-les-Moulineaux, France, 2016.

2. **M. Shaaban Ibraheem**, Sylvain Hochberg, Patrick Garda, Syed Zahid Ahmed. "Study of applying logarithmic dwt for medical images compression". *In 11ème Colloque du GDR SoC-SiP*, Nantes, France, 2016.

3. **M. S. Ibraheem**, S. Z. Ahmed B. Granado K. Hachicha A. Pinna, L. Lambert, I. Dhif and P.Garda. "Smart-eeg, a new platform for tele-expertise of electroencephalogram". *In 11ème Colloque du GDR SoC-SiP*, Nantes, France, 2016.

4. I. Dhif, I. Mhedhbi, **M. Shaaban Ibraheem**, A. Syed-Zahid, B.Granado, K. Hachicha, A. Pinna, P. Garda, F. Kaddouh, M. Terosiet, A. Histace, O.Romain, C. Bellet, F. Durand, J.P. Commes, S. Hochberg, D. Heudes, P. Lozeron, N.Kubis, L. Lambert, J. Despatin. "Telemedecine, electroencephalography and current issues smart-eeg: An innovative solution". *In 8ème édition du Congrès SFT ANTEL, Centre Universitaire des Saints Pères*, Paris, 2015.

5. **M. Shaaban Ibraheem**, S. Zahid Ahmed, B. Granado, K. Hachicha, A. Pinna, P. Garda, L. Lambert, I. Dhif, "Smart-eeg : A new platform for tele-expertise of electroencephalogram" *In GDR SOC SIP*, Paris, 2014.

Appendix A

Extended Results

We present here the results of compressing different medical images using LNS-WAAVES based on Hybrid-DWT and LNS-Q. The results include the quality measurement in SSIM for different compression ratios. Also, we present the percentage of the SSIM improvement for the different compression configuration parameters.

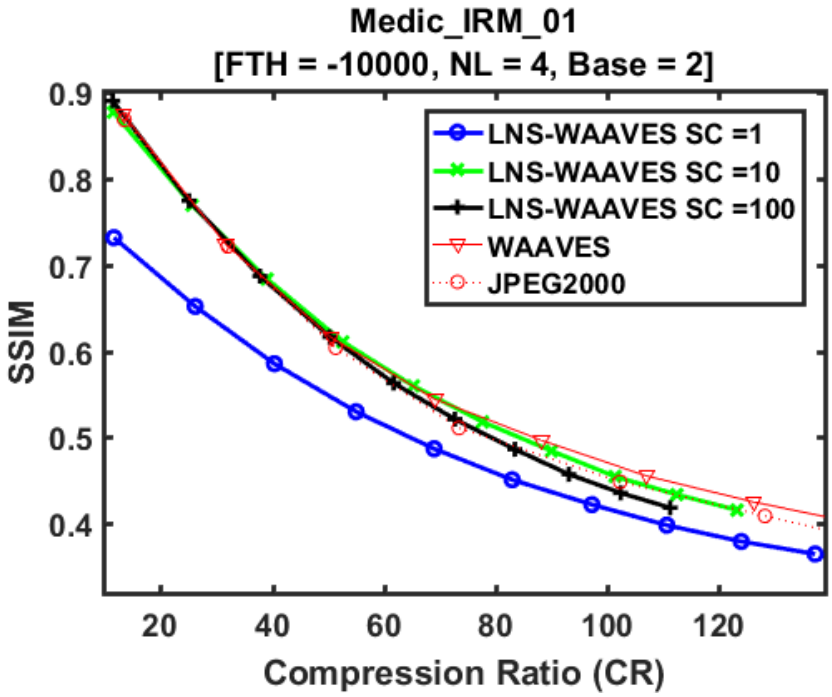


Fig. A.1 CR versus SSIM, LNS-WAAVES, NL = 4

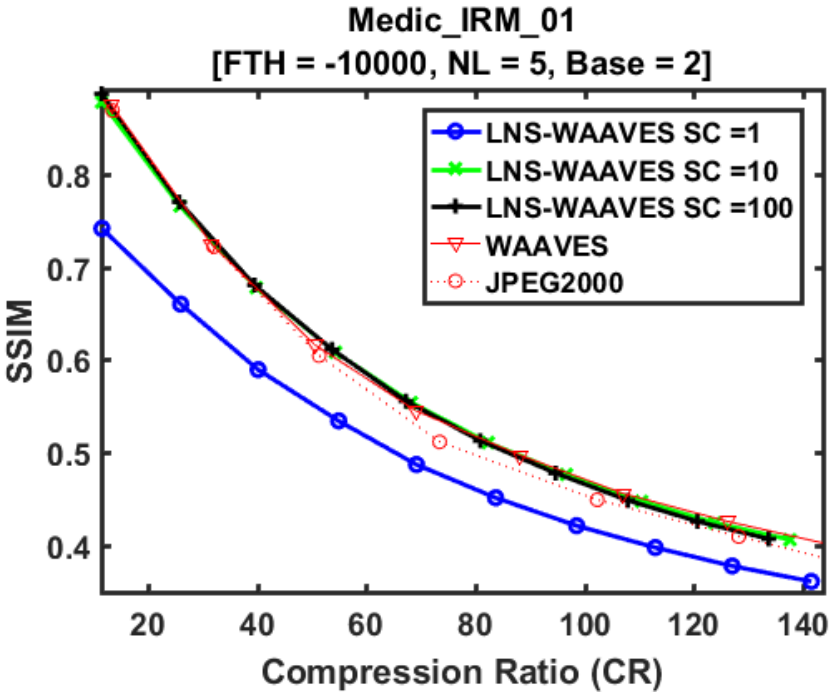


Fig. A.2 CR versus SSIM, LNS-WAAVES, NL = 5

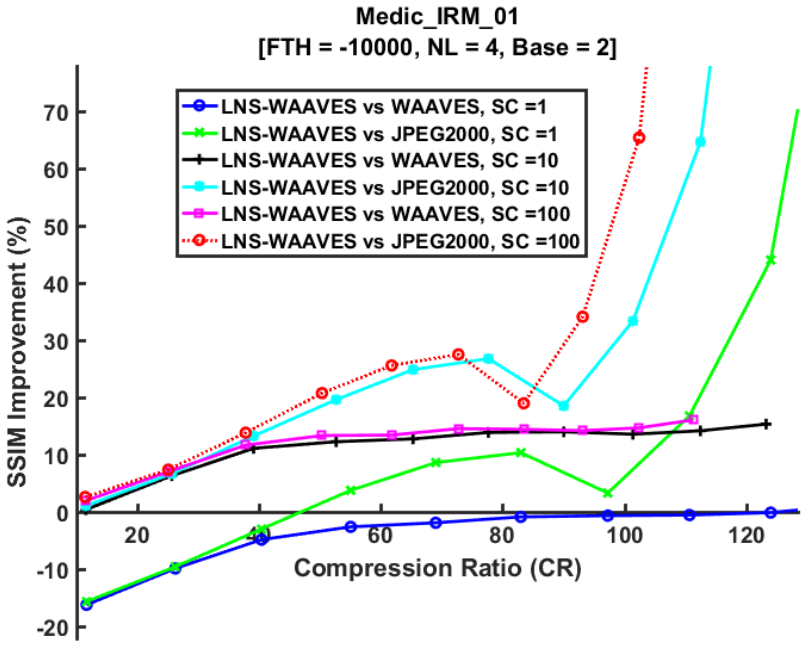


Fig. A.3 CR versus the SSIM improvement, LNS-WAAVES, NL = 4

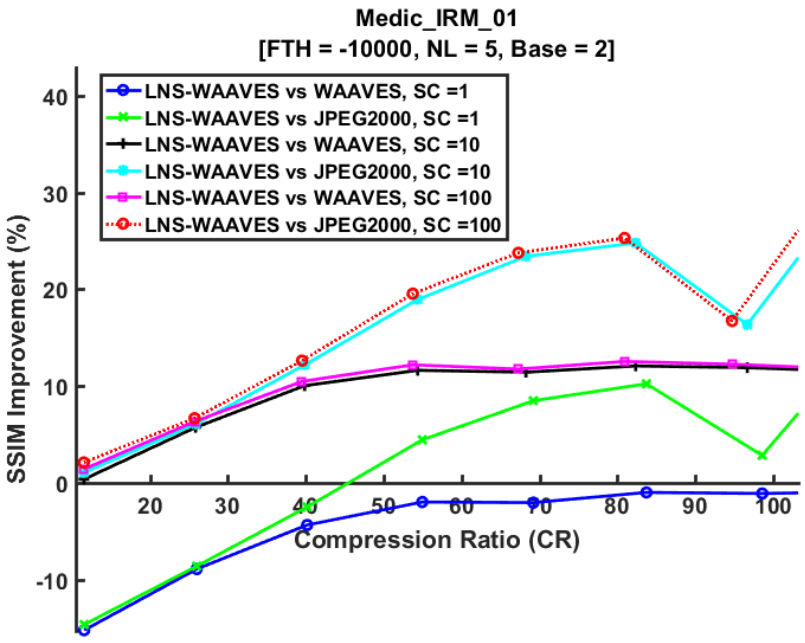


Fig. A.4 CR versus the SSIM improvement, NL = 5

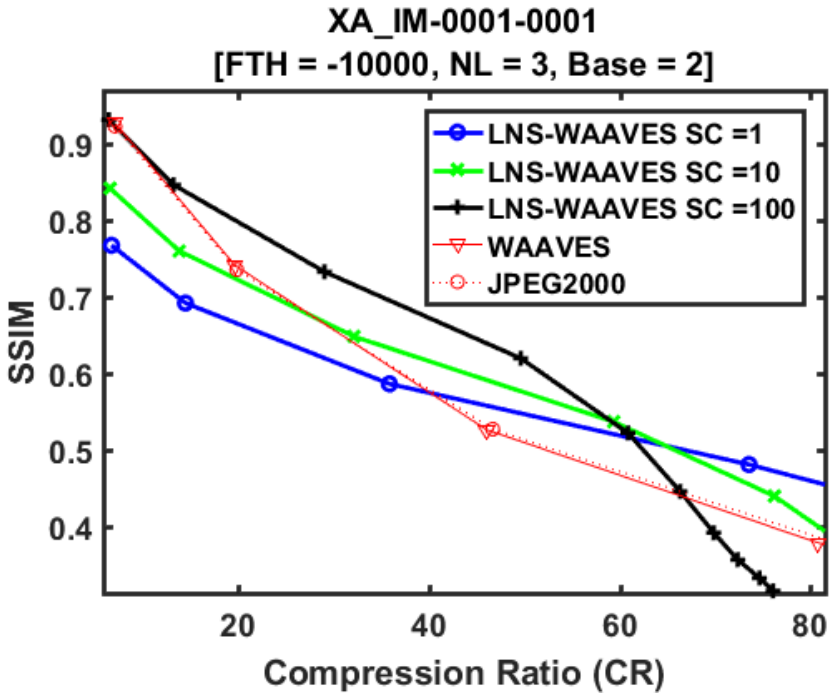


Fig. A.5 CR versus SSIM, LNS-WAAVES, NL = 3

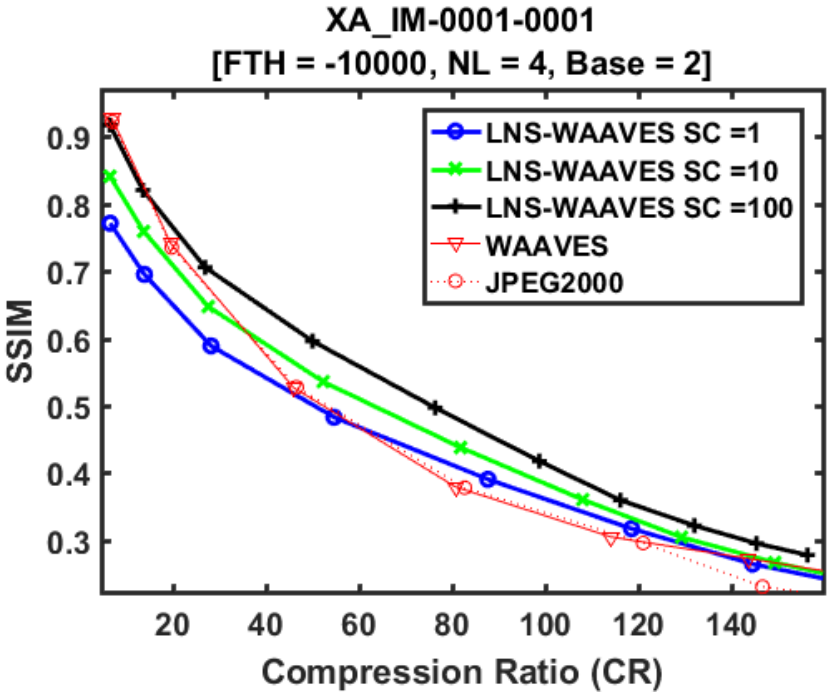


Fig. A.6 CR versus SSIM, LNS-WAAVES, NL = 4

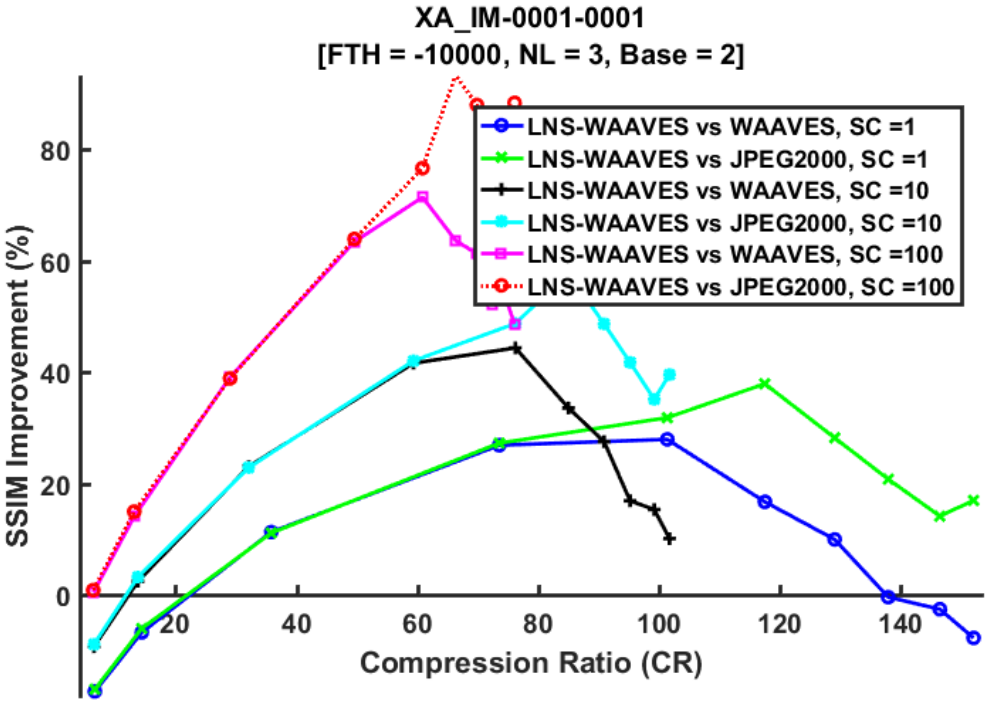


Fig. A.7 CR versus the SSIM improvement, LNS-WAAVES, NL = 3

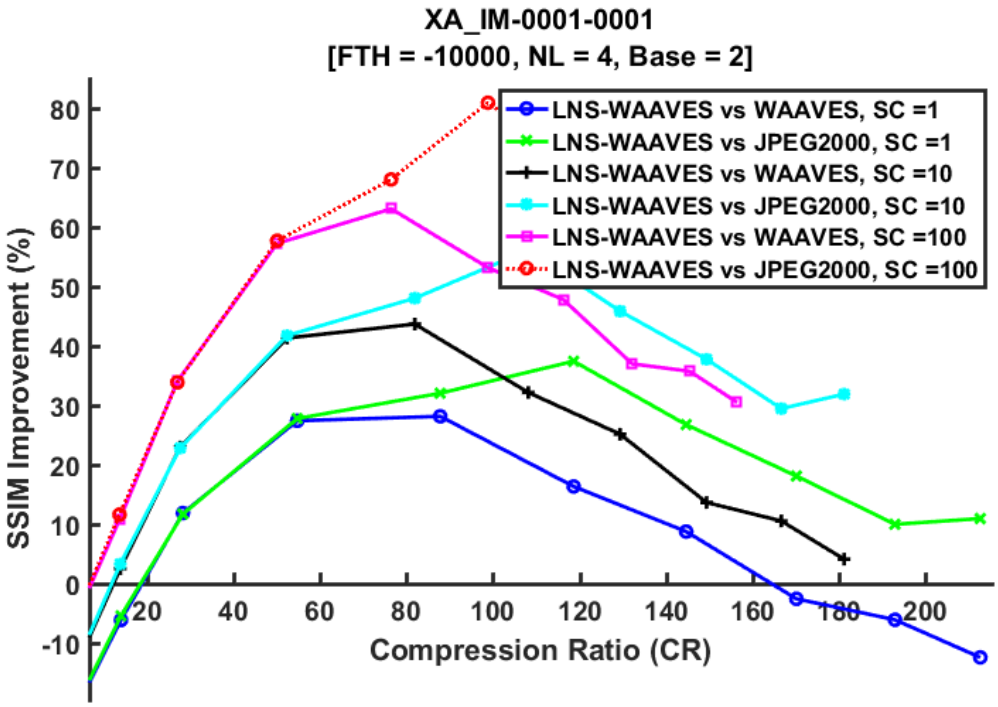


Fig. A.8 CR versus the SSIM improvement, NL = 4

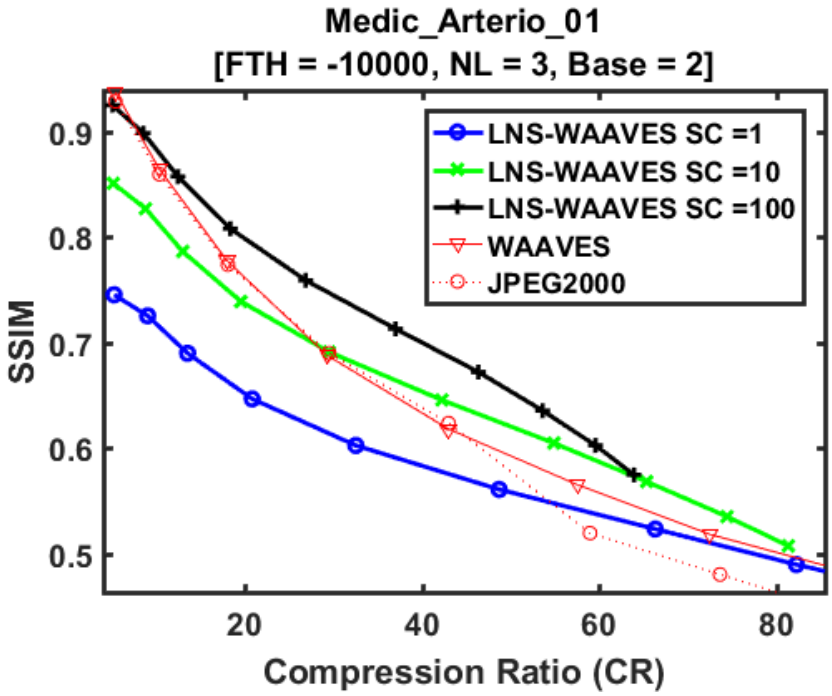


Fig. A.9 CR versus SSIM, LNS-WAAVES, NL = 3

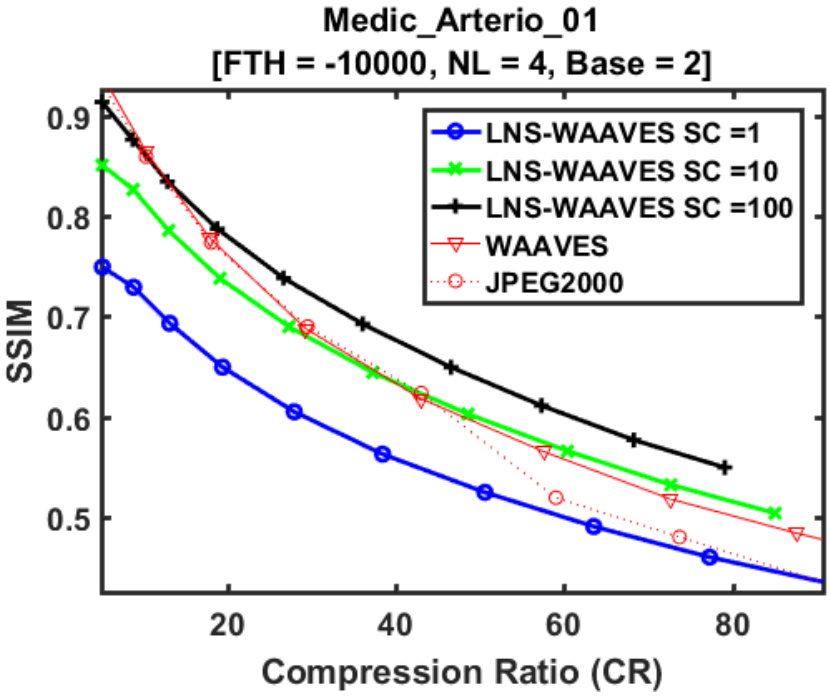


Fig. A.10 CR versus SSIM, LNS-WAAVES, NL = 4

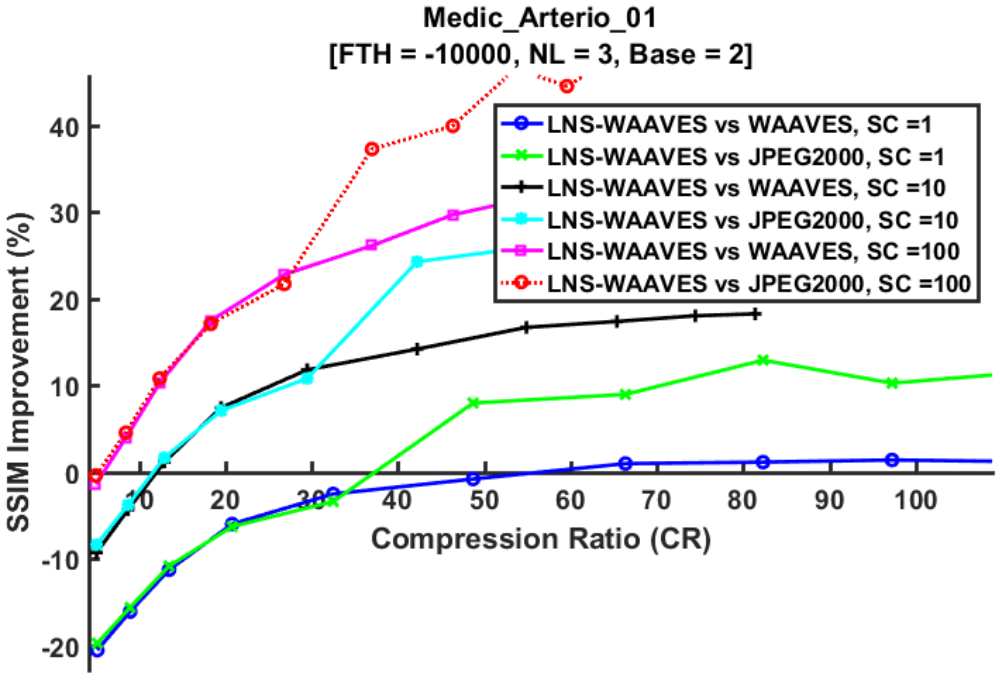


Fig. A.11 CR versus the SSIM improvement, LNS-WAAVES, NL = 3

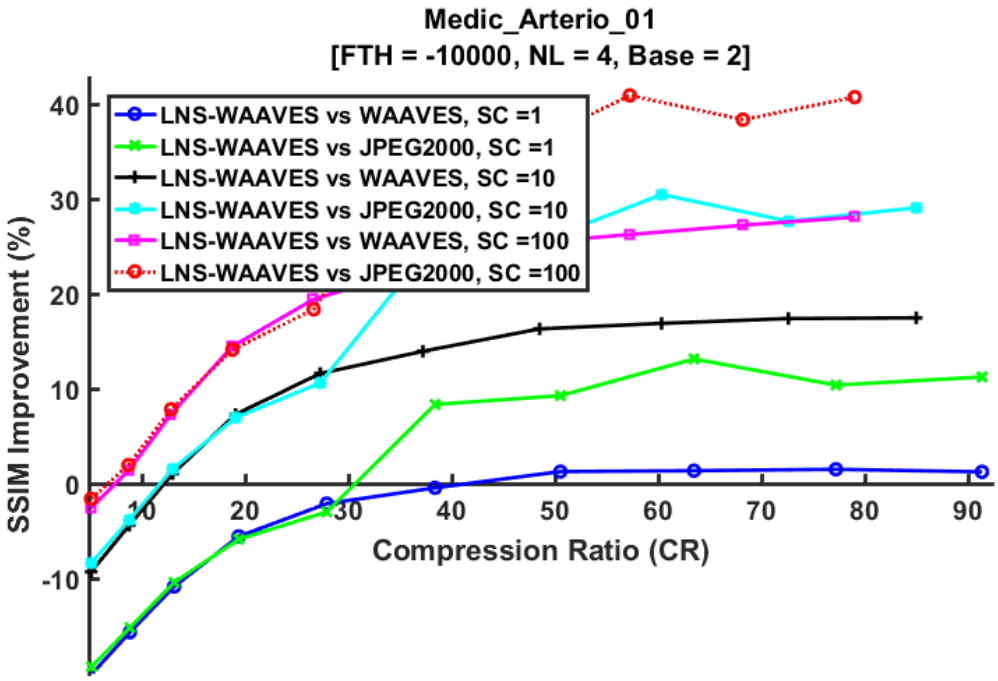


Fig. A.12 CR versus the SSIM improvement, NL = 4

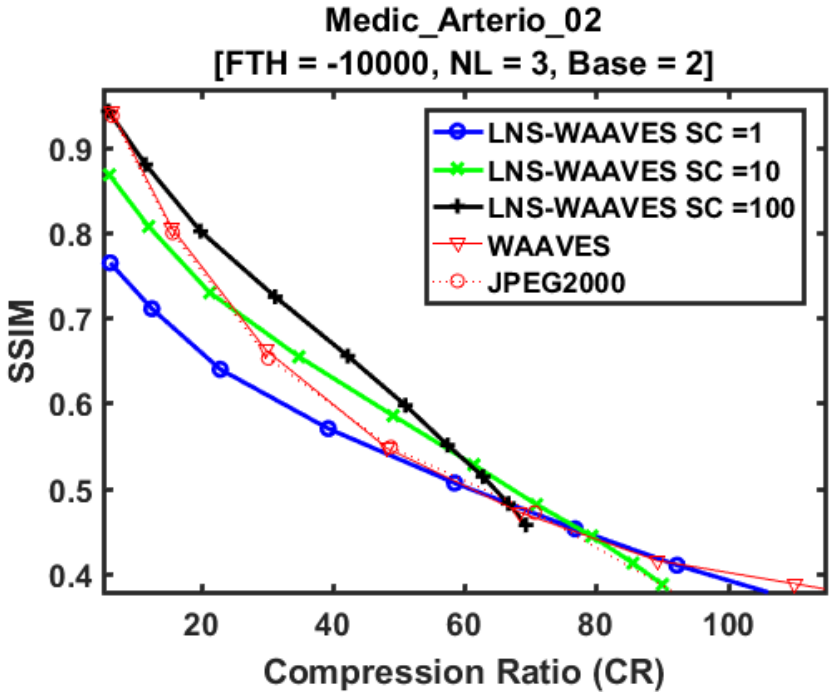


Fig. A.13 CR versus SSIM, LNS-WAAVES, NL = 3

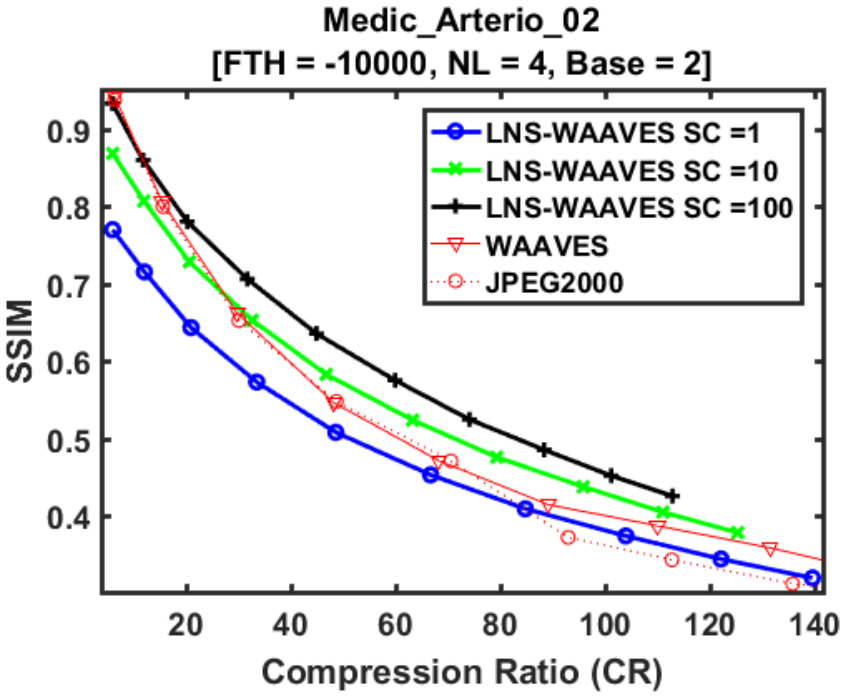


Fig. A.14 CR versus SSIM, LNS-WAAVES, NL = 4

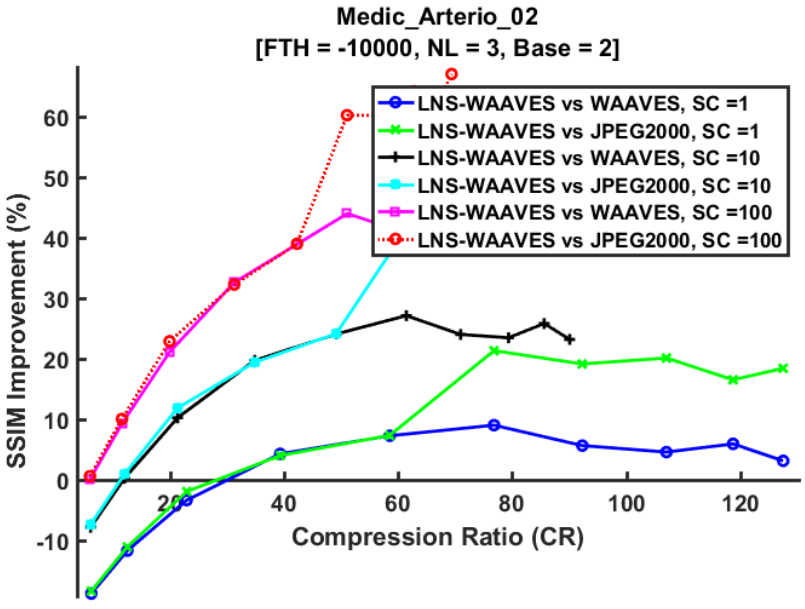


Fig. A.15 CR versus the SSIM improvement, LNS-WAAVES, NL = 3

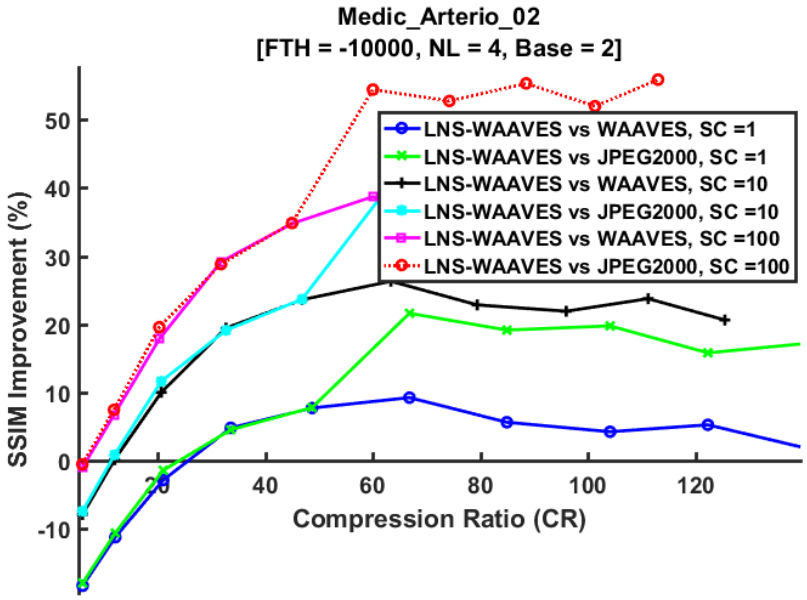


Fig. A.16 CR versus the SSIM improvement, NL = 4

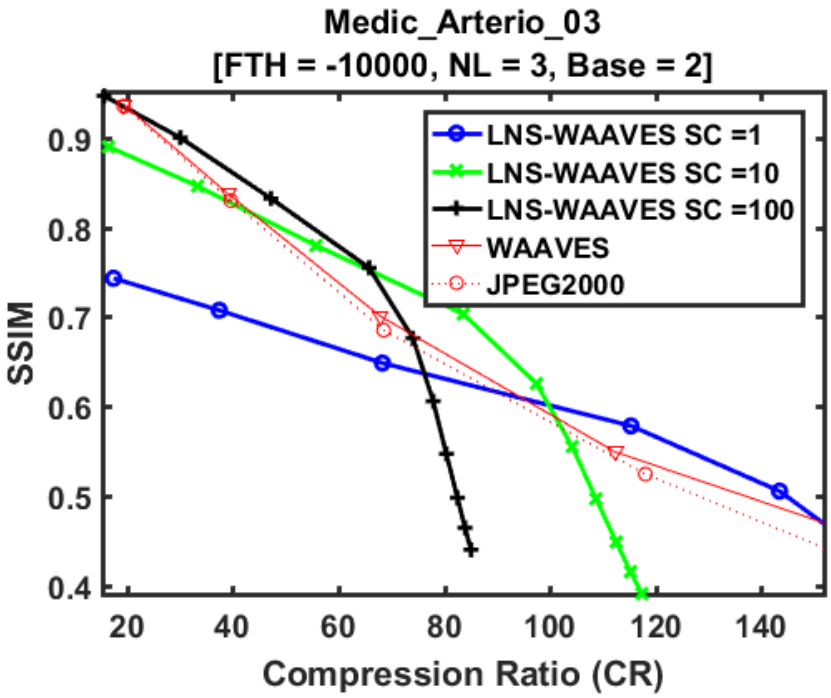


Fig. A.17 CR versus SSIM, LNS-WAAVES, NL = 3

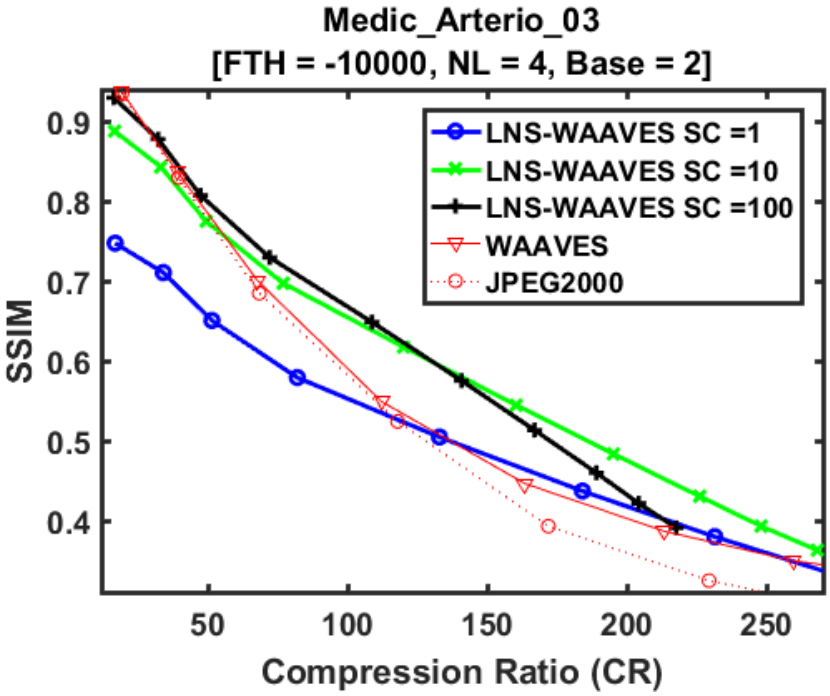


Fig. A.18 CR versus SSIM, LNS-WAAVES, NL = 4

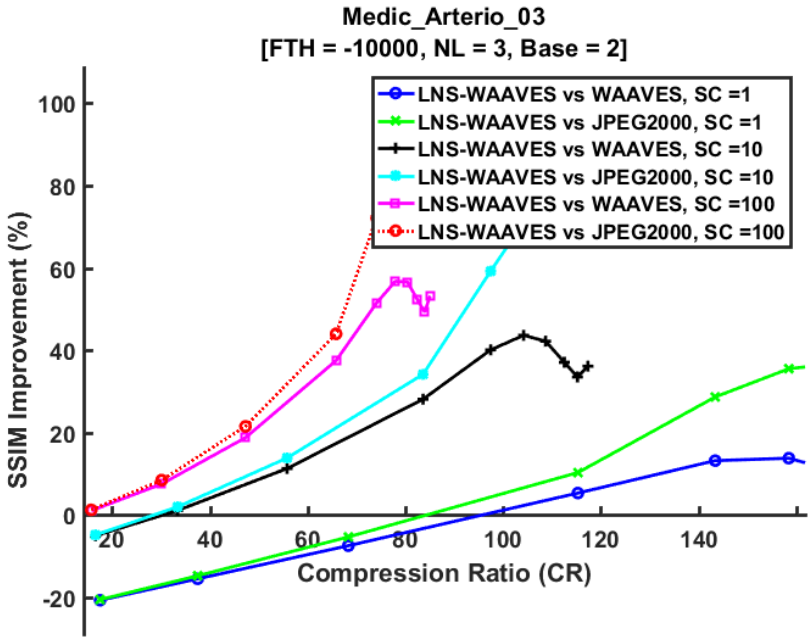


Fig. A.19 CR versus the SSIM improvement, LNS-WAAVES, NL = 3

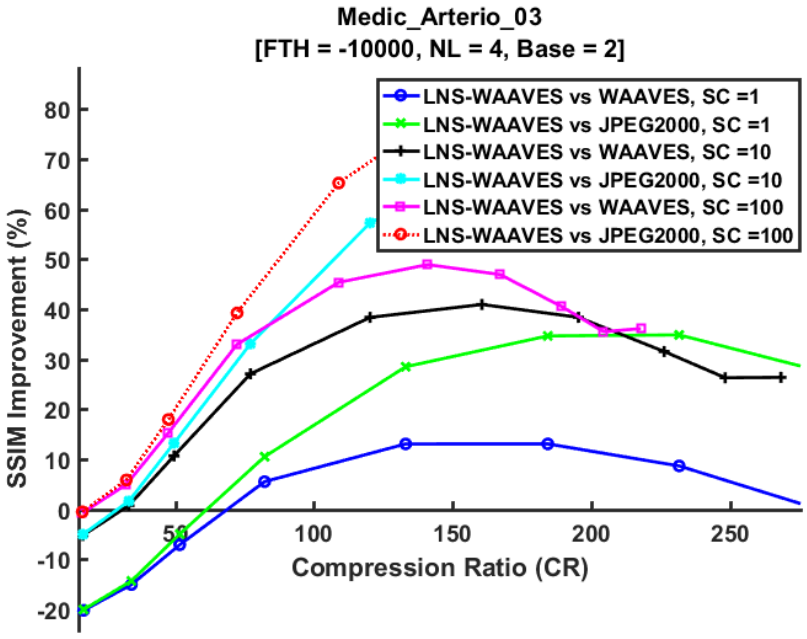


Fig. A.20 CR versus the SSIM improvement, NL = 4

References

[1] David S. Mendelson and Daniel L. Rubin. Imaging informatics: essential tools for the delivery of imaging services. *Academic Radiology*, 20(10):1195–1212, oct 2013.

[2] G. Schaefer, R. Starosolski, and Shao Ying Zhu. An evaluation of lossless compression algorithms for medical infrared images. In *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*, pages 1673–1676, Jan 2005.

[3] David A. Clunie. Lossless compression of grayscale medical images: effectiveness of traditional and state-of-the-art approaches. volume 3980, pages 74–84, 2000.

[4] Christine Cavaro-Menard, Amine Naït-Ali, Olivier Déforges, and Marie Babel. Compression of 2-d biomedical images. In *Compression of Biomedical Images and Signals*, chapter 7, pages 155–186. OISTE-Wiley, 2004.

[5] P.M.A. van Ooijen, P.J.M. ten Bhomer, and M. Oudkerk. {PACS} storage requirements—influence of changes in imaging modalities. *International Congress Series*, 1281:888 – 893, 2005. {CARS} 2005: Computer Assisted Radiology and SurgeryProceedings of the 19th International Congress and Exhibition.

[6] Suma and V Sridhar. Article: A review of the effective techniques of compression in medical image processing. *International Journal of Computer Applications*, 97(6):23–30, July 2014. Full text available.

[7] L. Lambert, J. Despatin, I. Dhif, I. Mhedhbi, M. Shaaban Ibraheem, A. Syed-Zahid, B. Granado, K. Hachicha, A. Pinna, P. Garda, F. Kaddouh, M. Terosiet, A. Histace, O. Romain, C. Bellet, F. Durand, J.P. Commes, S. Hochberg, D. Heudes, P. Lozeron, and N. Kubis. Telemedecine, electroencephalography and current issues. smart-eeg: An innovative solution. *European Research in Telemedicine*, 4(3):81 – 86, 2015.

[8] G. K. Wallace. The jpeg still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, Feb 1992.

[9] M. Shneier and M. Abdel-Mottaleb. Exploiting the jpeg compression scheme for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):849–853, Aug 1996.

[10] D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, Sept 1952.

[11] P. G. Howard and J. S. Vitter. Arithmetic coding for data compression. *Proceedings of the IEEE*, 82(6):857–865, Jun 1994.

[12] S. Grgic, M. Grgic, and B. Zovko-Cihlar. Performance analysis of image compression using wavelets. *IEEE Transactions on Industrial Electronics*, 48(3):682–695, Jun 2001.

[13] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image coding using wavelet transform. *IEEE Transactions on Image Processing*, 1(2):205–220, Apr 1992.

[14] A.R. Calderbank, Ingrid Daubechies, Wim Sweldens, and Boon-Lock Yeo. Wavelet transforms that map integers to integers. *Applied and Computational Harmonic Analysis*, 5(3):332 – 369, 1998.

[15] I. Mhedhbi, F. Kaddouh, K. Hachicha, D. Heudes, S. Hochberg, and P. Garda. Mask motion adaptive medical image coding. In *IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*, pages 408–411, June 2014.

[16] Imen Mhedhbi, Khalil Hachicha, Patrick Garda, Yuhui Bai, Bertrand Granado, Sébastien Topin, and Sylvain Hochberg. *Towards a Mobile Implementation of Waaves for Certified Medical Image Compression in E-Health Applications*, pages 79–87. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[17] CIRA. Waaves compression format, 2016.

[18] L. Oktem and J. Astola. Hierarchical enumerative coding of locally stationary binary data. *Electronics Letters*, 35(17):1428–1429, Aug 1999.

[19] Ieee standard for floating-point arithmetic. *IEEE Std 754-2008*, pages 1–70, Aug 2008.

[20] C. Inacio and D. Ombres. The dsp decision: fixed point or floating? *IEEE Spectrum*, 33(9):72–74, Sep 1996.

[21] J. Detrey and F. de Dinechin. A parameterized floating-point exponential function for fpgas. In *Proceedings. 2005 IEEE International Conference on Field-Programmable Technology, 2005.*, pages 27–34, Dec 2005.

[22] J. Detrey and F. de Dinechin. A parameterizable floating-point logarithm operator for fpgas. In *Conference Record of the Thirty-Ninth Asilomar Conference onSignals, Systems and Computers, 2005.*, pages 1186–1190, October 2005.

[23] Pavle Belanović and Miriam Leeser. *A Library of Parameterized Floating-Point Modules and Their Use*, pages 657–666. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[24] J. Dido, N. Geraudie, L. Loiseau, O. Payeur, Y. Savaria, and D. Poirier. A flexible floating-point format for optimizing data-paths and operators in fpga based dsps. In *Proceedings of the 2002 ACM/SIGDA Tenth International Symposium on Field-programmable Gate Arrays*, FPGA '02, pages 50–55, New York, NY, USA, 2002. ACM.

[25] Altaf Abdul Gaffar, Wayne Luk, Peter Y.K. Cheung, Nabeel Shirazi, and James Hwang. *Automating Customisation of Floating-Point Designs*, pages 523–533. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[26] B. Lee and N. Burgess. Parameterisable floating-point operations on fpga. In *Conference Record of the Thirty-Sixth Asilomar Conference on Signals, Systems and Computers, 2002.*, volume 2, pages 1064–1068 vol.2, Nov 2002.

[27] G. Lienhart, A. Kugel, and R. Manner. Using floating-point arithmetic on fpgas to accelerate scientific n-body simulations. In *Proceedings. 10th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 182–191, 2002.

[28] Eric Roesler and Brent E. Nelson. Novel optimizations for hardware floating-point units in a modern fpga architecture. In *Proceedings of the Reconfigurable Computing Is Going Mainstream, 12th International Conference on Field-Programmable Logic and Applications*, FPL '02, pages 637–646, London, UK, UK, 2002. Springer-Verlag.

[29] Gokul Govindu, L. Zhuo, S. Choi, and V. Prasanna. Analysis of high-performance floating-point arithmetic on fpgas. In *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.*, pages 149–, April 2004.

[30] Michael deLorimier and André DeHon. Floating-point sparse matrix-vector multiply for fpgas. In *Proceedings of the 2005 ACM/SIGDA 13th International Symposium on Field-programmable Gate Arrays*, FPGA '05, pages 75–85, New York, NY, USA, 2005. ACM.

[31] Yong Dou, S. Vassiliadis, G. K. Kuzmanov, and G. N. Gaydadjiev. 64bitt floating-point fpga matrix multiplication. In *Proceedings of the 2005 ACM/SIGDA 13th International Symposium on Field-programmable Gate Arrays*, FPGA '05, pages 86–95, New York, NY, USA, 2005. ACM.

[32] J. Detrey, F. de Dinechin, and X. Pujol. Return of the hardware floating-point elementary function. In *18th IEEE Symposium on Computer Arithmetic (ARITH '07)*, pages 161–168, June 2007.

[33] Ki-Il Kum, Jiyang Kang, and Wonyong Sung. A floating-point to integer c converter with shift reduction for fixed-point digital signal processors. In *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No.99CH36258)*, volume 4, pages 2163–2166 vol.4, Mar 1999.

[34] F. J. Taylor, R. Gill, J. Joseph, and J. Radke. A 20 bit logarithmic number system processor. *IEEE Transactions on Computers*, 37(2):190–200, Feb 1988.

[35] J. N. Coleman, E. I. Chester, C. I. Softley, and J. Kadlec. Arithmetic on the european logarithmic microprocessor. *IEEE Transactions on Computers*, 49(7):702–715, Jul 2000.

[36] J. N. Coleman, C. I. Softley, J. Kadlec, R. Matousek, M. Licko, Z. Pohl, and A. Hermanek. The european logarithmic microprocessor - a qr rls application. In *Conference Record of Thirty-Fifth Asilomar Conference on Signals, Systems and Computers (Cat.No.01CH37256)*, volume 1, pages 155–159 vol.1, Nov 2001.

[37] N. G. Kingsbury and P. J. W. Rayner. Digital filtering using logarithmic arithmetic. *Electronics Letters*, 7(2):56–58, January 1971.

[38] Rudolf Matoušek, Milan Tichý, Zdeněk Pohl, Jiří Kadlec, Chris Softley, and Nick Coleman. *Logarithmic Number System and Floating-Point Arithmetics on FPGA*, pages 627–636. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[39] J. Detrey and F. de Dinechin. A vhdl library of lns operators. In *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*, volume 2, pages 2227–2231 Vol.2, Nov 2003.

[40] H. Fu, O. Mencer, and W. Luk. Optimizing logarithmic arithmetic on fpgas. In *15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM 2007)*, pages 163–172, April 2007.

[41] E. E. J. Swartzlander, D. V. Satish Chandra, H. T. J. Nagle, and S. A. Starks. Sign/logarithm arithmetic for fft implementation. *IEEE Transactions on Computers*, C-32(6):526–534, June 1983.

[42] D. S. Taubman and M. W. Marcellin. Jpeg2000: standard for interactive imaging. *Proceedings of the IEEE*, 90(8):1336–1357, Aug 2002.

[43] Michael W. Marcellin, Margaret A. Lepley, Ali Bilgin, Thomas J. Flohr, Troy T. Chinen, and James H. Kasner. An overview of quantization in {JPEG} 2000. *Signal Processing: Image Communication*, 17(1):73 – 84, 2002. {JPEG} 2000.

[44] Z. Wang and A. C. Bovik. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE Signal Processing Magazine*, 26(1):98–117, Jan 2009.

[45] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, April 2004.

[46] Z. Wang, A. C. Bovik, and L. Lu. Why is image quality assessment so difficult? In *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 4, pages IV–3313–IV–3316, May 2002.

[47] Ilona Kowalik-Urbaniak, Dominique Brunet, Jiheng Wang, David Koff, Nadine Smolarski-Koff, Edward R. Vrscay, Bill Wallace, and Zhou Wang. The quest for 'diagnostically lossless' medical image compression: a comparative study of objective quality metrics for compressed medical images. volume 9037, pages 903717–903717–16, 2014.

[48] Y. Bai, S. Z. Ahmed, and B. Granado. Fpga implementation of hierarchical enumerative coding for locally stationary image source. In *2013 23rd International Conference on Field programmable Logic and Applications*, pages 1–6, Sept 2013.

[49] K. A. Kotteri, S. Barua, A. E. Bell, and J. E. Carletta. A comparison of hardware implementations of the biorthogonal 9/7 dwt: convolution versus lifting. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 52(5):256–260, May 2005.

[50] Ingrid Daubechies and Wim Sweldens. Factoring wavelet transforms into lifting steps. *Journal of Fourier analysis and applications*, 4(3):247–269, 1998.

[51] Wim Sweldens. The lifting scheme: A construction of second generation wavelets. *SIAM Journal on Mathematical Analysis*, 29(2):511–546, 1998.

[52] Wim Sweldens. The lifting scheme: A new philosophy in biorthogonal wavelet constructions. In *in Wavelet Applications in Signal and Image Processing III*, pages 68–79, 1995.

[53] C. Tenllado, J. Setoain, M. Prieto, L. Piñuel, and F. Tirado. Parallel implementation of the 2d discrete wavelet transform on graphics processing units: Filter bank versus lifting. *IEEE Transactions on Parallel and Distributed Systems*, 19(3):299–310, March 2008.

[54] Gustavo D. Sutter Jean-Pierre Deschamps, Gery J.A. Bioul. *Synthesis of Arithmetic Circuits: FPGA, ASIC and Embedded Systems*. Wiley, first edition, 2006.

[55] C. Xiong, J. Tian, and J. Liu. Efficient architectures for two-dimensional discrete wavelet transform using lifting scheme. *IEEE Transactions on Image Processing*, 16(3):607–614, March 2007.

[56] Cheng-Yi Xiong, Jin-Wen Tian, and Jian Liu. Efficient high-speed/low-power line-based architecture for two-dimensional discrete wavelet transform using lifting scheme. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(2):309–316, Feb 2006.

[57] B. K. Mohanty and P. K. Meher. Memory-efficient high-speed convolution-based generic structure for multilevel 2-d dwt. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(2):353–363, Feb 2013.

[58] Y. Hu and C. C. Jong. A memory-efficient high-throughput architecture for lifting-based multi-level 2-d dwt. *IEEE Transactions on Signal Processing*, 61(20):4975–4987, Oct 2013.

[59] K. Andra, C. Chakrabarti, and T. Acharya. A vlsi architecture for lifting-based forward and inverse wavelet transform. *IEEE Transactions on Signal Processing*, 50(4):966–977, Apr 2002.

[60] Bing-Fei Wu and Chung-Fu Lin. A high-performance and memory-efficient pipeline architecture for the 5/3 and 9/7 discrete wavelet transform of jpeg2000 codec. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(12):1615–1628, Dec 2005.

[61] B. K. Mohanty and P. K. Meher. Memory efficient modular vlsi architecture for highthroughput and low-latency implementation of multilevel lifting 2-d dwt. *IEEE Transactions on Signal Processing*, 59(5):2072–2084, May 2011.

[62] Syed Mahfuzul Aziz and Duc Minh Pham. Efficient parallel architecture for multi-level forward discrete wavelet transform processors. *Computers & Electrical Engineering*, 38(5):1325–1335, 2012.

[63] A. Darji, S. Agrawal, A. Oza, V. Sinha, A. Verma, S. N. Merchant, and A. N. Chandorkar. Dual-scan parallel flipping architecture for a lifting-based 2-d discrete wavelet transform. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 61(6):433–437, June 2014.

[64] Hongyu Liao, M. K. Mandal, and B. F. Cockburn. Efficient architectures for 1-d and 2-d lifting-based wavelet transforms. *IEEE Transactions on Signal Processing*, 52(5):1315–1326, May 2004.

[65] Anand D Darji, Shailendra Singh Kushwah, Shabbir N Merchant, and Arun N Chandorkar. High-performance hardware architectures for multi-level lifting-based discrete

wavelet transform. *EURASIP Journal on Image and Video Processing*, 2014(1):1–19, 2014.

[66] Ishmael Sameen, Yoong Choon Chang, Mow Song Ng, Bok-Min Goi, and Chee-Pun Ooi. A unified fpga-based system architecture for 2-d discrete wavelet transform. *Journal of Signal Processing Systems*, 71(2):123–142, 2013.

[67] Y. Hu and C. C. Jong. A memory-efficient scalable architecture for lifting-based discrete wavelet transform. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 60(8):502–506, Aug 2013.

[68] HEGP. Hôpital européen georges-pompidou, 2016.

[69] C. H. Hsia, J. S. Chiang, and J. M. Guo. Memory-efficient hardware architecture of 2-d dual-mode lifting-based discrete wavelet transform. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(4):671–683, April 2013.

[70] L. Lambert, K. Hachicha, S. Z. Ahmed, A. Pinna, and P. Garda. Synchronizing physiological data and video in a telemedicine application: A multimedia approach. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 181–185, Aug 2015.