
# Design of an Image Codec based on Bandelet Transform using NIOS II Processors

***By:***
***Jaime-Andres Arteaga***

***Supervisor:***
***Jaime Velasco-Medina, Ph.D.***

Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of Master in Engineering with Emphasis in Electronics

**School of Electrical and Electronics Engineering**
**Universidad del Valle**
**Cali, Colombia, South-America**
**April 2011**

# Content

# List of Figures

# *Abstract*

This document presents the design of an image codec for grayscale images based on Bandelet Transform. Bandelet bases are obtained with a set of geometric-flow vectors that indicates the directions in which gray levels have regular variations inside an image region. These regions are associated with high 2D Wavelet coefficients with geometric information redundancy, which could be eliminated in order to reduce the number of non-zero coefficients and to increase the compression rate of the image. The codec was designed in C language using a NIOS II processor inside a SoPC with a touch-panel and a SD-Card. Some of the functions were synthesized in hardware with NIOS II C2H Compiler obtaining an acceleration percentage of 8.8%. Numerical tests show that a Bandelet compression has an improvement of up to 2 dB over a Wavelet compression.


***Index Terms:*** Bandelet Transform, hardware acceleration, image compression, NIOS II processor, Wavelet Transform.


# *Resumen*

Este documento presenta el diseño de un codec de imágenes en escalas de gris basado en la Transformada Bandelet. Las funciones base Bandelet se construyen a partir de vectores de flujo geométrico que indican la dirección en la que una región tiene variaciones regulares de los niveles de gris. Estas regiones están asociadas a coeficientes Wavelet 2D de gran magnitud con redundancia de información geométrica, la cual se puede eliminar para reducir el número de coeficientes diferentes de cero y así aumentar la compresión de la imagen. El codec se diseñó en lenguaje C para un procesador embebido NIOS II dentro de un sistema SoPC con pantalla táctil y SD-Card. Algunas de las funciones fueron sintetizadas en hardware con NIOS II C2H Compiler logrando una aceleración del 8.8%. Las pruebas realizadas muestran que la compresión con funciones Bandelet llega a ser hasta 2 dB superior a la compresión realizada con funciones Wavelet 2D.


***Palabras Clave:*** Aceleración de hardware, compresión de imágenes, procesador NIOS II, Transformada Bandelet, Transformada Wavelet.

# *Chapter 1*

# *Introduction*

## 1.1. Motivation

Most of the embedded systems that are now on the market allow taking pictures with different resolutions and sizes. These pictures are stored in an internal memory as a flash memory, or in a memory expansion as a SD-Card. Although these memories can reach relatively large sizes (micro-SD card can be up to 32 GB [1]), photographs are stored in compressed formats to maximize available memory resources and also to decrease the time that a file takes to be transferred between two computers, either in a point-to-point link, on a local network or Internet.

One of the most used compression formats is JPEG (Joint Photographic Experts Group). This standard takes an uncompressed image and after changing its color space to YCbCr (Luminance and Chrominance), subsamples the chrominance components. Then, the color components are transformed into frequency domain using Discrete Cosine Transform (DCT). The resulting frequency components are finally quantized and encoded in order to obtain a smaller file size than the original one. The JPEG standard has been specified for the processing of still images, while for motion pictures the MPEG standard is used. The JPEG standard defines a set of image compression techniques instead of one. The standard has four modes of operation: Sequential lossless, Sequential DCT-based, Progressive DCT-based, and Hierarchical. Only the first mode is a lossless compression technique, while the other three are lossy compression techniques[1]. From these, the most used is the Sequential DCT-based, also known as Baseline JPEG.

[1]In lossy compression techniques, some of the data are discarded in order to obtain a high compression rate. Therefore, the decompressed image is similar to the original one, but it is not exact. In lossless compression techniques, all the original data is preserved so the decompressed image is an exact replica of the original one.

Despite its popularity, the JPEG standard has several flaws. For example, JPEG cannot reconstruct an image at different resolutions unless you are using the Hierarchical mode. This feature is known as scalability and is very attractive for various applications such as web environments, where the user may need to see a low-resolution image copy before downloading the complete image file from the Internet, and as more data is downloaded, the image resolution increases. The JPEG standard also uses two different architectures for lossless compression and lossy compression; the encoder-decoder (codec) of images must be designed from the beginning for one of these two compression types. Random access to image data in JPEG is not possible, so the compressed file has always to be decoded completely even if the user wants to edit only a small area of the image.

To overcome these shortcomings and to provide new compression techniques for mobile multimedia applications, JPEG2000 standard was released in 2001. Unlike JPEG standard, which is based on DCT, JPEG2000 uses the Discrete Wavelet Transform (DWT). This transform processes the image as a whole, while DCT divides the image in blocks (usually of 8x8 pixels) and processes them independently. That way, DWT allows obtaining a better energy compaction of the image with a smaller number of coding errors (e.g. blocking artifacts[2]). DCT, used in JPEG, describes a two-dimensional signal in terms of frequencies and amplitudes, but only for a time instant or scale. DWT in turn, describes a signal in terms of frequencies and amplitudes at different time instants or different scales, achieving a multiresolution representation of the original image. Thus, image processing with DWT can be adapted to local details of the image, unlike DCT, which processes the image with a fixed resolution (Figure 1.a, 1.b). This allows for a more efficient representation of image singularities such as sharp edges and terminations [2].

JPEG2000 standard offers better compression ratios than those obtained with the JPEG, a unified architecture for both lossy and lossless compression, and the option of encoding an area with greater detail than the rest of the image (Region of Interest Coding). One of its main features is the capability to manipulate the image in the compression space, rather than in the temporal space, as JPEG standard does, because JPEG2000 is a scalable compression format.

[2]The blocking artifacts occur in the JPEG standard because of the discontinuities produced from dividing the image in 8x8 pixel blocks.

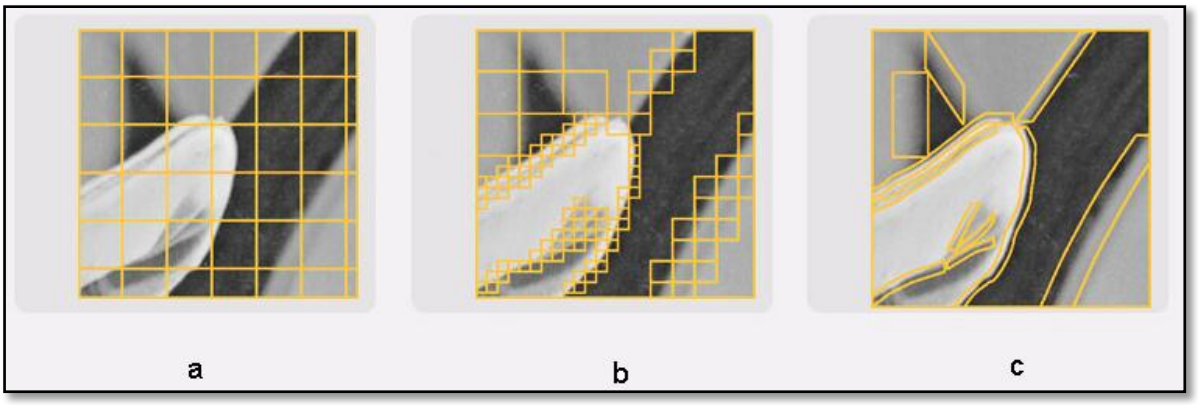



**Figure 1.** Comparison of image-processing techniques: a) Fourier-based techniques such as DCT processes images using blocks of fixed size; b) DWT processing adapts to local image details; c) Bandelet processing adapts to both scale and geometry of the image [3].

This means that if a digital camera uses JPEG to compress the pictures, the resolution and image quality parameters must be established before taking photographs. If the camera uses JPEG2000 instead, these parameters can be modified after taking pictures. In this way, photographs are shot at the highest resolution possible and once the memory is full, the resolution of some of the photographs already stored in memory can be changed to make space for new photos.

Basis functions used in JPEG and JPEG2000 (Cosine and Wavelet respectively) do not take advantage of the geometric regularities of images. Integrating these regularities in the representation of images is a key point in the development of new imaging techniques such as compression and denoising. Having this into account, the Bandelet Transform has been developed as a result of the work in applied mathematics made by professors Stéphane Mallat, Gabriel Peyré, and Erwan Le Pennec [2], [3], [4], [5], [6]. This transform adapts to the image geometry in order to process it with a high accuracy (Figure 1.c). Bandelet basis are constructed from geometric flow vectors that indicate the direction in which a region has regular variations of gray levels (Figure 2). Its main objective is to reduce the geometric redundancy of 2D Wavelet coefficients by applying a reordering, followed by a 1D Wavelet transform [5].

Even though it is a relatively new transform (2003), its potential in compression has not gone unnoticed. Reference [2] shows that a Bandelet compression has an improvement between

0.6 dB and 2.0 dB in the PSNR (Peak Signal-to-Noise Ratio[3]) over a Wavelet compression, using Lena.bmp and Barb.bmp as test images respectively. Bandelet Transform has also been used in image denoising [7], the improvement of JPEG2000 in retaining texture [8], satellite image compression [9], and video compression [10]. For demonstrative purposes, professors Mallat and Peyré have developed a Matlab toolbox for the compression of geometric images using the Bandelet Transform. This package serves a as a teaching tool for understanding the operation of this transform [11]. In the industrial field, Let It Wave Corp. (France) has developed the Super-Resolution Bandelet Technology based on Bandelet algorithms for the conversion of analog television to HD images. This technology is intended to be applied in LCD televisions, DVD players, and video coding [12].

## 1.2. Statement of the Problem

Despite the potentialities shown by the Bandelet Transform for image processing, only one hardware implementation is found in the collected literature, done by Let it Wave in an Altera FPGA. Thus, it is important that DSP designers make a thorough study of the Bandelet Transform using a hardware/software platform, in order to evaluate its capabilities over the transforms currently used for image processing such as DCT and DWT, setting a precedent for further researches and developments.

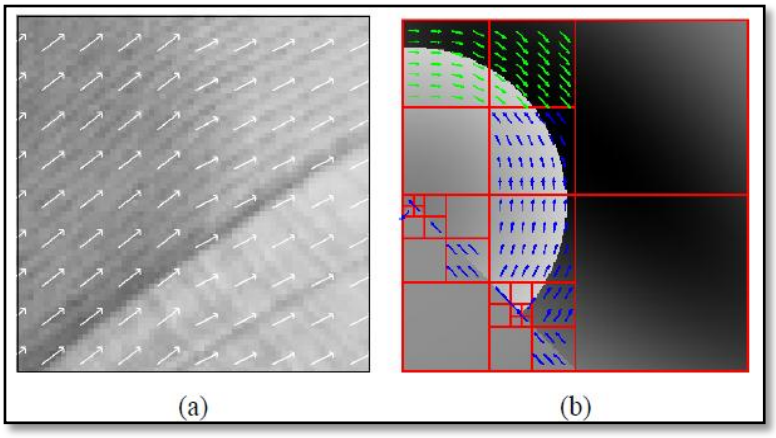



**Figure 2**. Example of the geometric flow of an image: a) Geometric flow in a local area of the image; b) Division of the image in a dyadic fashion for the local calculation of the flow [4].

[3]PNSR indicates the efficiency of a compression system: The bigger the PSNR, the more similar the decompressed image is to the original one.

In this context, this Master research project proposes the design of an image compression codec in which the image frequency coefficients are obtained with the Bandelet Transform, using a hardware/software implementation based on the Bandelet algorithm presented in [5]. Due to the high computational load involved in image processing and looking for taking advantage of the parallel processing capabilities of hardware, the design will be aimed towards a System-on-Programmable-Chip (SoPC) implementation, with an Altera NIOS II embedded configurable processor as its main functional block.

## 1.3. Overview of the Document

The main objective of this project is the design of a codec for image compression based on the Bandelet Transform as a SoPC controlled by a NIOS II processor. To this end, Chapter 2 discusses previous works on image compression systems designed for JPEG and JPEG2000 standards in both software and hardware.

Chapter 3 focuses on explaining briefly the JPEG standard, including the use of DCT, the quantization method proposed by the standard, and the encoding process of the quantized coefficients by variable-length codes (Huffman codes). The JPEG2000 standard is also explained. The main features of DWT are described, as well as its differences with the DCT used in JPEG. The recommendations given by the JPEG2000 standard for the implementation of the DWT in image compression systems are described too. Finally, the quantization and entropy-coding methods used in JPEG2000 are compared with those used in JPEG.

Chapter 4 introduces the Bandelet Transform and the theoretical framework that allows its use in an image compression system. The Bandelet Approximation Algorithm, which is used for the design of the codec presented in this document, is explained, as well as some numerical results of its application in image compression.

Chapter 5 presents two FPGA implementations of the JPEG and JPEG2000 standards. These systems were built for Terasic DE2 and the DE2-70 boards and include a CMOS sensor of 1.3 Mega-pixels and a touch panel of 800x480 pixels. The architecture, the resources used, and the performance in image compression of each implementation are also analyzed.

Chapter 6 presents the design of the image codec based on the Bandelet Transform using the Bandelet Approximation Algorithm. This design was made for the JPEG and JPEG2000 platforms presented in chapter 5.

Chapter 7 focuses on the performance evaluation of the Bandelet codec over a codec based solely on the 2D Wavelet Transform. Decompressed images are evaluated using the SNR, the presence of Blocking Artifacts, and the number of non-zero coefficients, among other metrics.

Finally, Chapter 8 presents the conclusions of this research project and some considerations for future developments that could arise out of the work presented in this document.

# *Chapter 2*

# *Review of Previous Works*

This chapter presents some of the works reviewed for the research project presented in this document. Various compression systems developed for JPEG and JPEG2000 standards are described, including software and hardware implementations. The main paper in which this research is supported is also explained.

## 2.1. FPGA Coprocessing in a JPEG2000 Implementation

Brenan [13], for an undergraduate thesis, developed this project is based on an open source software implementation of JPEG2000 known as JasPer included in Part 5 of the standard. First, the project carries out a timing analysis of the JPEG2000 compression algorithm in an Intel Pentium III computer, concluding that the arithmetic coding is the most time-consuming stage, so it is implemented in an FPGA using VHDL (Very-High-Speed Hardware Description Language). The selected card for the implementation is the Annapolis Micro Systems Wildcard. This card has a Xilinx Virtex 300K, with two SRAM-banks, 256 KB each. The performance of the design was evaluated with simulation tests and the future work of the project aims to the description in VHDL of the complete compression algorithm into an FPGA. Simulation results indicate that the hardware implementation of the arithmetic coding reduces the time consumed by the compression algorithm up to 40% over an implementation designed entirely in software.

## 2.2. Surface Compression with Geometric Bandelets

Peyré and Mallat [5] describe the construction process of the Bandelet basis and its application in geometric images. The proposed algorithm decomposes a surface into a set of Bandelet basis in order to reduce the geometric redundancy of the 2D Wavelet coefficients by

the reordering of these coefficients and the subsequent application of a 1D Wavelet transform. This algorithm is performed in the Wavelet domain and each scale of decomposition is processed independently. The reordering is carried out by selecting a direction *d* as parallel as possible to the actual geometry of the image area that is currently being analyzed. As shown in Figure 3, first a square *S* is selected out of a scale of the 2D Wavelet Transform, and each point of *S* is projected orthogonally to the line *d*, obtaining a new point $\tilde{x}$. To construct the 1D discrete signal, the points are sorted according to their number along the axis *d*. The resulting one-dimensional signal is then processed with a 1D Wavelet Transform to achieve its compression. This transform is also useful to determine if the reordering has been done correctly, using a user-input threshold *T* that controls the compression rate of the algorithm: Only the resulting coefficients from the 1D Wavelet Transform above *T* are used for final compression. These coefficients are stored in a two-dimensional array of the same size of *S* using a zig-zag ordering, so the largest coefficients are at the top left of the array. This arrangement can be exploited later by an entropy encoder. Experimental results of this study show that the compression of geometric images using Bandelet functions can achieve an increase between 1.4 and 2.0 dB in the PSNR of the decompressed image when it is compared to the compression of the same image using Wavelet functions.

## 2.3. NIOS II Processor-Based Hardware/Software Co-Design of the JPEG2000 Standard

Dyer et al. [14] developed a JPEG2000 compression system co-designed in hardware and software based on a software implementation of the standard called Kakadu.

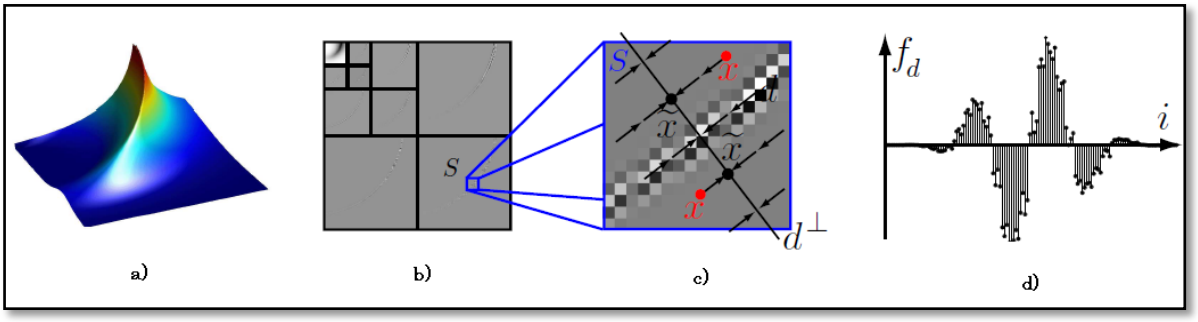



**Figure 3.** Discrete reordering of the 2D Wavelet coefficients: a) Original Image; b) 2D Wavelet Transform; c) Projection of the points of square *S*, d) Resulting 1D Signal [5].

The platform used for the design is the Altera NIOS II processor, selected for its capability of hardware configuration at low levels. For example, designers can insert signal processing instructions into the processor ALU (Arithmetic-Logic Unit) in order to increase the processing speed of the Wavelet Transform and to reduce the code size. The design also uses the real-time operating system RTOS μC/OS-II to control the parallel processing of the peripherals. After performing a timing analysis of execution for Kakadu, the Wavelet Transform and the entropy coding stages are designed in hardware. The filters used in the Wavelet Transform are Daubechies 9/7, which are implemented in a lifting scheme using a state machine and are added to the ALU of the NIOS II processor. For the entropy encoding stages, hardware blocks are designed to implement the bit-plane encoder and the arithmetic encoder. The input data of these encoders come from multiple DMA (Direct-Access Memory) blocks connected to the NIOS II processor (Figure 4). The encoders receive the data from DMA modules at a frequency of 16 bits per clock cycle. The implementation results show that the inclusion in the ALU of instructions to calculate the DWT improves the processing speed of the system by a factor of 1.04 in comparison with the Kakadu software implementation. The hardware implementation of the entropy encoder improves the system speed by a factor of 2.6.

## 2.4. Analysis of the Previous Works

In several of the reviewed projects, it was found that the hardware-software co-design of the systems began with an analysis of the execution times of the software implementation in order to determine the most time-consuming blocks that are the best candidates to be implemented in hardware. These blocks are usually those related to obtaining the frequency coefficients (DCT and DWT) and those related to the encoding process. In the case of JPEG2000 standard, the entropy encoder is the most computationally demanding stage, which involves a bit-plane encoder and an arithmetic encoder.

From the analysis of the paper *Surface Compression with Geometric Bandelets* is concluded that the Bandelet Transform is more complex than the Wavelet Transform because the former is adapted to the geometric characteristics of the image by calculating the direction *d* that best represents the local geometry that is being analyzed. Since this is a parameter determined at runtime, is expected to be one of the most time-consuming processes of the Bandelet codec.

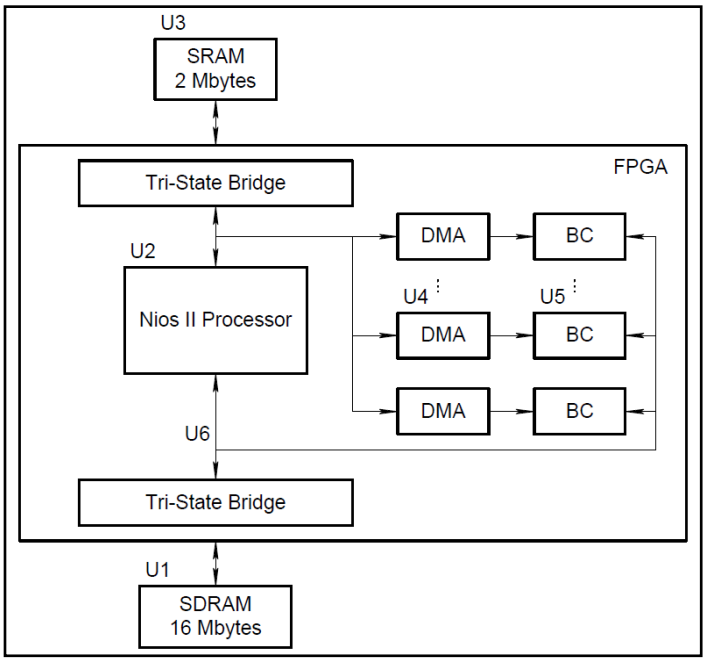



**Figure 4.** Hardware implementation of the entropy encoder of the JPEG2000 standard [14].

# *Chapter 3*

# *JPEG and JPEG2000 Standards*

This chapter briefly explains the JPEG and JPEG2000 standards used for the compression of still images. In chapter 5, these standards are implemented in a SoPC platform to understand the functioning of an image compression method; this platform is used for the design of the Bandelet codec in chapter 6. First, the concept of data compression is introduced, as well as the classification of compression techniques in lossy and lossless techniques. Both are included in the JPEG and JPEG2000 standards. The basic structure of a data compression system is presented, which adjusts perfectly to the two standards, even though each one of them uses different approaches for the data processing. In addition, SNR and PSNR are introduced as the metrics used to measure the quality of an image compression system.

In the second part of the chapter, each one of the processes involved in the JPEG compression algorithm will be analyzed, including the use of DCT, the quantization method proposed by the standard, and the encoding process of the quantized coefficients by variable-length codes known as Huffman codes. The compression algorithm used in the JPEG2000 standard is also explained. The main feature of this standard is the use of the DWT for the decorrelation of the pixels of an image. An explanation of the DWT characteristics is presented, followed by the analysis of its differences with the DCT used in JPEG. Finally, the methods of quantization and entropy coding used in JPEG2000 are explained.

## 3.1. Data Compression

As the quality of multimedia applications increases, so do the memory requirements needed for their implementation. For example, a medium-resolution image of 800x600 pixels in 24 bit RGB requires 1.44 Mbytes for storage. For HD-TV, the amount of memory increases considerably since it uses 60 frames per second with a typical resolution of 1280x760 pixels.

However, much of the information contained in the images is not fully captured by the human eye, so it is redundant. An example of this is the YIQ color representation, where the Y component is known as Luminance and defines the black and white information contained in the image, while I and Q components are known as Chrominance and are responsible for the color information. The advantage of this representation is that the spatial resolution of human eye has a different sensitivity for each component: It is more sensitive to changes in Y and less sensitive to changes in Q; the sensitivity to component I is intermediate. This means that if some of the data of Q component were discarded, the human eye would not notice it. The standard NTSC for color television takes advantage of the YIQ representation to transmit the television signal with a lower bandwidth, using a bandwidth of 4MHz for the transmission of Y, 1.5MHz for I, and 0.6 MHz for Q. The omission of some data in I and Q components eliminates the perceptual redundancy of the television image and implements a basic technique of two-dimensional signal compression.

The objective of a data compression technique is to reduce redundant information in a data stream in order to decrease the storage requirements and costs of information transmission [15]. Data compression is a research area that is in constant growth, taking into account developments in technology and the increase in storage and processing requirements in embedded applications, where a limited amount of memory is present for the implementation of various tasks such as image and video processing, data transmission over mobile internet, and typical tasks of a real-time operating system.

The main advantage of compressing information is the reduction in the amount of memory needed for storage, which translates into lower bandwidth for transmission. But this also implies that the computational load of the system increases since it must perform the tasks of compression and decompression of the information. This computational load varies depending on the information and the application the system is working with. Not all applications can allow omitting some of its data information in order to perform a data compression. Medical diagnostic images for example, always need to have available the largest amount of image details so that patients receive a correct diagnosis and proper treatment. Similarly, satellite images require a high amount of information to make a detailed and deep examination of space. In these cases, data compression is performed maintaining as much information as possible, so that the original image and decompressed image are equal.

## 3.2. Classification of Data Compression Techniques

Compression techniques take an input data stream $D_{in}$ and generate an output data stream $D_{out}$ that represents the input information but using fewer bits than $D_{in}$; $D_{out}$ is called the compressed data stream. The reverse process takes $D_{out}$ and generates a reconstructed data stream $D_{rec}$; this process is called decompression. If the data stream generated from the decompression $D_{rec}$ is identical to the original data stream $D_{in}$, the compression technique is called lossless, otherwise, it is called a lossy. Sometimes the compression system is also known as encoder, whereas the decompression system is called decoder. The complete system that performs compression and decompression of data is denominated codec (Figure 5).

## 3.3. Basic Structure of a Data Compression System

A data compression system consists of three main blocks: Information Redundancy Reduction, Entropy Reduction, and Entropy Coding (Figure 6). The information redundancy is present in a stream when there are more data than needed to represent the same information. In the NTSC television format mentioned before, there is information redundancy in components I and Q. Redundancy is also reduced through the representation of the information in another domain. In this case, Discrete Cosine Transform (DCT) or Discrete Wavelet Transform (DWT) is used to transform the original information in the time domain to the frequency domain. The data output from the transform that represents the information in the frequency domain are called coefficients.

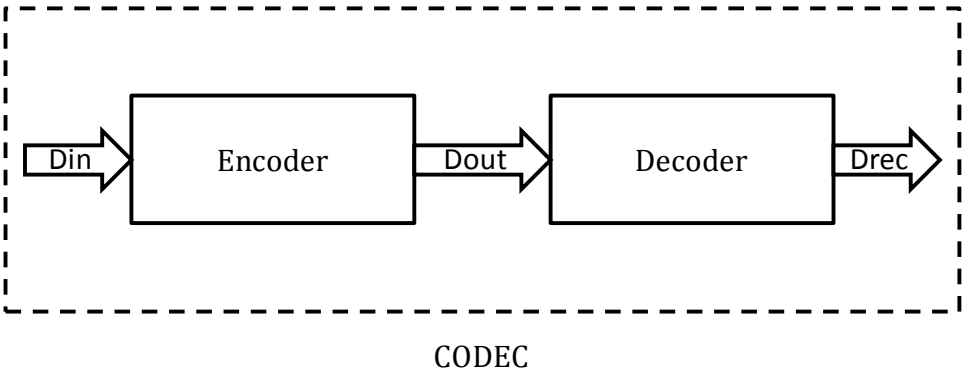



**Figure 5.** Data compression and decompression system [15].

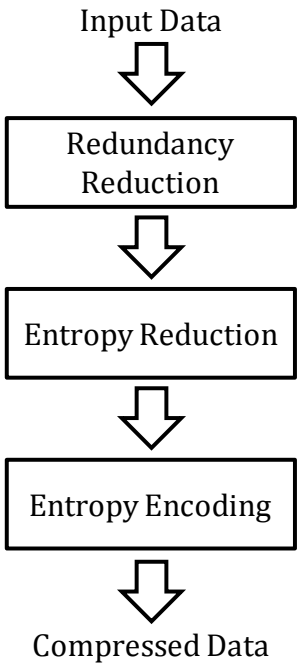



**Figure 6.** Block Diagram of a data compressor system [15].

After the transform, the entropy of the coefficients is reduced. In this context, entropy is defined as the level of surprise or uncertainty present in a data stream: If inside the stream there are only data with high probability of occurrence, then there is no surprise and the entropy is low; if instead inside the stream there are data that was unlikely to occur, there is a surprise, the entropy is higher, and the piece of data that was not expected carries more information than the data with high probability of occurrence.

To reduce the entropy of the coefficients, they are quantized in order to obtain more coefficients with similar values, decreasing that way the surprises: For example, all real-valued coefficients such as 0.5, 0.7, 0.2, are quantized to an integer value of 0.This process is not reversible, since it is not possible to recover the original coefficient values that existed before the quantization process and it is applied only in lossy compression techniques. The coefficients are then encoded using entropy encoding techniques such as Huffman encoding, Run-Length Enconding (RLE), arithmetic encoding, or others. The decompression process is performed in the reverse direction of the information flow in Figure 6, starting with the entropy decoding and ending with the inverse transform of the coefficients. In order to have a

lossless codec, the quantization step has to be omitted and the frequency transform has to be reversible.

## 3.4. Quality Metrics of a Data Compression System

When the data compression system is based on lossy techniques, the decompressed data are not an exact replica of the original data, and the differences between them are called artifacts. A compression system is said to be of high quality when the artifacts of the decompressed data are very low. There is not a standard metric for measuring the quality of a compression system, but two of the most commonly used are the Signal to Noise Ratio SNR, defined by (1), and the Peak Signal to Noise Ratio PSNR, defined by (2), in the case of two-dimensional signals.

$$SNR = 20\,log_{10}\left(\frac{\sqrt{\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}[I(i,j)]^2}}{\sqrt{\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}[I(i,j)-I'(i,j)]^2}}\right) \quad (1)$$

$$PSNR = 20\,log_{10}\left(\frac{255}{\sqrt{\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}[I(i,j)-I'(i,j)]^2}}\right) \quad (2)$$

Where M and N indicate the resolution of the image, *I(i, j)* represents the original image, *I'(i, j)* is the decompressed image, and 255 is the maximum value for a pixel with eight bits. Analyzing the expressions for SNR and PSNR is concluded that when the decompressed and the original image are very similar, the internal difference of the summations in the denominator are very low, and therefore, the internal quotient of the logarithm is high. This means that the larger the PSNR is, the smaller the distortions in the reconstructed image are. In this case a balance between quality and memory requirements has to be considered: When the compression is aggressive, less amount of memory is required to store data, and the PSNR gets low; if the objective is to keep as many original data as possible, the PSNR increases, and also the memory requirements for the compressed data.

## 3.5. JPEG Standard

One of the most widely used standards in multimedia applications is the JPEG (Joint Photographic Experts Group) defined by the ITU-T T.81 Recommendation for still images [16]. The JPEG standard is not really a compression technique, but a set of several techniques that are available for different applications. These techniques can be grouped into four:

- ***Sequential Lossless Compression:*** The image is compressed in a single scan and the decompressed image is identical to the original one.
- ***Sequential DCT-based Compression:*** The image is compressed in a single scan using DCT and the decompressed image is an approximation of the original one.
- ***Progressive DCT-based Compression:*** The image is compressed and decompressed in several scans, producing images with different levels of quality.
- ***Hierarchical Compression:*** The image is compressed with different resolutions to be used in different systems.

From the four techniques, the most used is the Sequential DCT-based Compression, also called Baseline JPEG, defined for images of up to four components. Figure 7 shows the algorithm followed by this technique for the compression of an image component. The first step in the Baseline JPEG algorithm is the color space conversion of the image, usually from RGB to YCbCr, in order to reduce the data of those components for which the human eye is less sensitive. After that, the chrominance components are sub-sampled by two, discarding the data present in the odd positions of each row and/or column. This reduces the amount of information up to one fourth of the original one.

Once the color space conversion is done, each component is divided into blocks of 8x8 pixels. The blocks are processed independently by the entire algorithm, starting with a transformation to the frequency domain using the DCT. Each block is then quantized, producing loss of information and therefore a lossy compression. The quantization is performed by dividing each of the coefficients of the 8x8 block by an element of a quantization matrix, which has integer values between 1 and 255. After quantization, the coefficients are differentially encoded with RLE and Huffman codes.

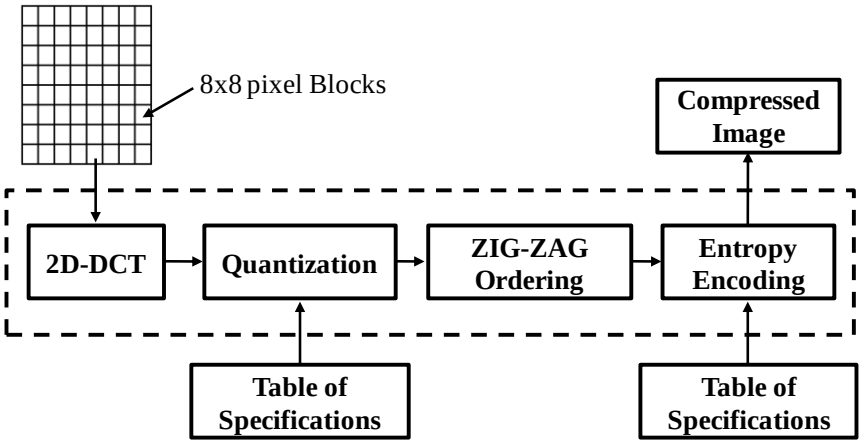



**Figure 7.** Compression of a component of an image according to the Baseline JPEG.

Huffman coding is a technique for variable length encoding that assigns a smaller number of bits to the coefficients that appear the more in the current block. Huffman tables used for the compression of the image are stored in the header of the compressed file in order to be available for the decompression system; the same happens with the quantization matrices. The final data stream comprises the encoded bits of each of the blocks of the original image components ordered from left to right, top to bottom, in a zig-zag ordering.

## 3.6. JPEG2000 Standard

Despite its popularity, the standard JPEG is not optimal and has several disadvantages. One of these is the blocking artifacts produced in the reconstructed image and caused by processing the original image into separate blocks of 8x8 pixels. These distortions give the image a gridded appearance and are more visible when the JPEG standard is set for a high compression ratio. The JPEG2000 standard was developed with the aim of overcoming the shortcomings of JPEG. One of its main characteristics is that the same architecture can be used for lossy and lossless compression. As a result, the same compression system is suitable for applications that require exact replicas of the original image (e.g. medical applications) as well as for applications where the goal is to save memory (digital still cameras) [15]. JPEG2000 standard also allows of random access to the compressed data stream, giving the possibility of editing certain areas of the image without needing to decode the complete data stream. The basic architecture of a JPEG2000 encoder for lossy and lossless image compression is divided into five main stages and is presented in Figure 8.

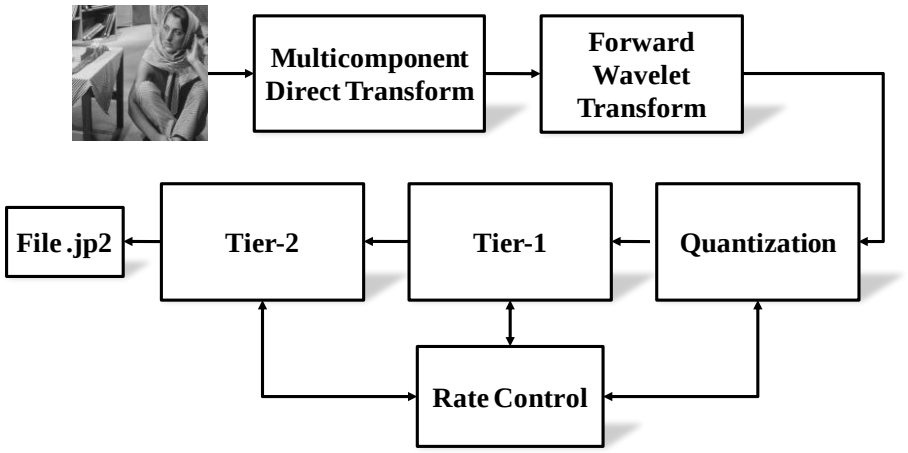



**Figure 8.** Architecture of a JPEG2000 encoder.

### *3.6.1. Multicomponent Direct Transformation*

If the image is very large, each component is divided into blocks of 256x256 or 512x512 pixels. This partition produces blocking artifacts as in the JPEG standard and may be omitted, but it is recommended as it allows0 the efficient use of memory resources available in the processing system. After the partition, each component is converted to a 2's complement representation with a DC-offset level to ensure that the dynamic range of the samples is centered on zero. If desired, a color space conversion can be performed in each component.

### *3.6.2. Forward Wavelet Transform*

To obtain the frequency coefficients of the image, JPEG2000 standard uses the Discrete Wavelet Transform (DWT) instead of the JPEG DCT. The DWT decomposes each component of the image into four bands (LL1, LH1, HL1, and HH1) at different levels of resolution (Figure 9). Sub-bands LH1, HL1, and HH1 contain the high frequency information of the image component in the horizontal, vertical, and diagonal directions respectively. The LL1 sub-band is a sub-sampled version by two of the original component and can be subsequently divided into four new sub-bands LL2, LH2, HL2, and HH2. The process can continue decomposing LL2 and so on. For lossless compression, the filter bank used in the DWT is Le Gall 5/3, while for lossy compression the filter bank used is Daubechies 9/7. The numbers indicate the size of the lowpass and highpass filters: Daubechies filter bank has a low-pass filter of 9 taps and a high-pass filter of 7 taps.

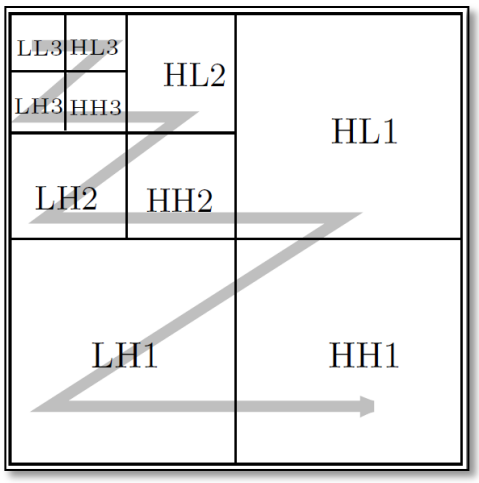



**Figure 9.** DWT performed in one image component [17].

### *3.6.3. Quantization*

In lossy compression, DWT coefficients are quantized using a uniform scalar quantization with a dead zone at the origin according to (3):

$$q_b(i,j) = sign\big(y_b(i,j)\big)\left[\frac{|y_b(i,j)|}{\Delta_b}\right] \tag{3}$$

Where $y_b(i,j)$ is a coefficient in the sub-band $b$ of the DWT and $\Delta_b$ is the step of quantization for that sub-band. The standard supports different quantization steps for each sub-band [15].

### *3.6.4. Tier-1*

Quantized coefficients are partitioned into blocks of 32x32 or 64x64 samples, which are then passed through a bit-plane encoding to reduce the number of bits required to represent the compressed data. Three scans are performed in each block: Significance Pass, Refinement Pass, and Clean-up Pass. Each one of these scans generates a context and a binary decision for each block position. The context and the decision are later used to generate the compression codes for each block.

### 3.6.5. Tier-2

The encoded data generated by Tier-1 are evaluated to find the optimal truncation point that gives the desired compression ratio. Therefore, only a subset of the data generated by Tier-1 is included in the final data stream. If the compression is lossless, all the data generated by Tier-1 are included [18].

### 3.6.6. Rate Control

Adjustments in the quantization steps and the selection of the truncation point in Tier-2 are processes which together are known as Rate Control of the JPEG2000 standard. This control allows achieving the desired compression ratio for the output data stream with the highest possible PSNR.

# *Chapter 4*

# *The Bandelet Transform*

This chapter presents the definition of the Bandelet Transform, also known as Bandlet Transform or Geometric Wavelet Transform. This transform has been successfully used in various multimedia applications such as noise reduction in images [4], geometric image compression [5], and HDTV [12]. Differences with DCT and DWT, mode of operation, and the advantages offered in digital image processing are explained. Finally, the Bandelet Approximation Algorithm for grayscale images is presented, which is the basis for the codec designed in this research project.

## 4.1. Definition of the Bandelet Transform

Image representation in orthonormal functions such as Cosine and Wavelet does not take into account of the geometric characteristics of the image. DCT processes images independently in blocks of 8x8 pixels, giving to the reconstructed image a gridded appearance due to the blocking artifacts. DWT in turn, processes images as a whole in different scales and orientations H-V-D (Horizontal, Vertical, and Diagonal respectively), as shown in Figure 10, where a section of Lena.bmp has been processed with DWT in a single scale, followed by a thresholding. In this figure, it is clear that the high-magnitude coefficients are close to the singularities of the image, a common feature in all the images processed by DWT. A better data compression than that obtained with DWT can be achieved if the compression system focuses on these high-magnitude coefficients in order to get an increased amount of coefficients that are equal to zero, without losing quality in the reconstructed image [4].

The purpose of the Bandelet Transform is to eliminate the redundancy of information in the DWT coefficients present in the singularities of the image, after performing the two-dimensional Wavelet Transform.

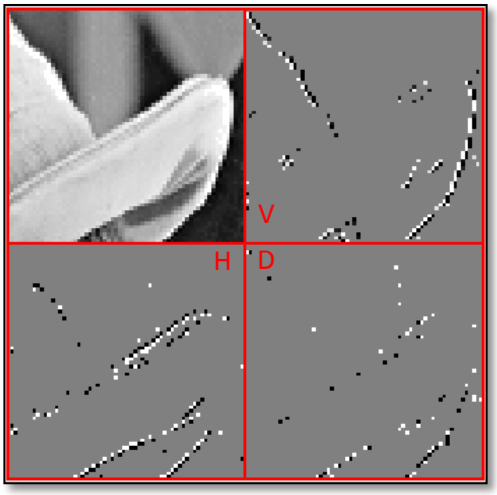



**Figure 10.** Two-dimensional Wavelet Transform of a section of Lena.bmp.

This is done by calculating a 1D DWT in the local directions where 2D DWT coefficients have regular variations of gray levels; this means that Bandelet Transform is defined only for two-dimensional signals. If the 1D DWT is applied in a direction close to the real geometry of the local area that is currently being analyzed, then it is possible to perform a thresholding of DWT coefficients in that area in order to reduce the number of nonzero coefficients, increasing data compression ratio without loss of image quality. The Bandelet Transform is the result of the research work in applied mathematics led by professors Stéphane Mallat, Gabriel Peyré, and Erwan Le Pennec from the *Centre de Mathématiques Appliqués* in France [4], [5].

## 4.2. Bandelet Orthonormal Bases

The set of Bandelet orthonormal bases is defined by segmenting the 2D-array of DWT coefficients in squares of various sizes that are subsequently processed with the 1D DWT. For every image, several sets of bases can be constructed using different segmentations and applying the 1D DWT in different directions. The optimal set of Bandelet orthonormal bases will be the one that best represents the real geometry of the image. To calculate this set of bases, a tree structure is used where each array of coefficients is divided using a dyadic segmentation, as shown in Figure 11. Each square taken from the 2D DWT coefficients is successively divided into four squares of equal size $L$ (called sub-squares) and for each one of these, the geometric direction $d$ that minimizes the Langrangian $\mathscr{L}$ is sought; this direction is

the one that is best adjusted to the local geometry of the image [19]. The Lagrangian associated with a direction $d$ in a square $S$ is calculated by the projection of the square elements in the selected direction as shown in Figure 12 (in each square $S$ there are maximum $2L^2$ possible directions [5]).

For each direction $d$, every point $x$ of the square $S$ is projected orthogonally to the axis given by $d$, generating a new point $x'$. The set of all points $x'$ on the axis $d$ produces a one-dimensional signal $f_d$ (Figure 12.e), which is processed with the 1D Wavelet Transform, also called Directional Wavelet Transform since it is applied on the 2D DWT coefficients in a specific direction. The resulting 1D Wavelet coefficients grouped in the signal $f_w$ (Figure 12.f) are discriminated using a threshold $T$, which is given by the user to control the compression ratio. Coefficients greater than threshold $T$ are used to calculate the corresponding Lagrangian $\mathscr{L}$ The objective is to use the Lagrangian to find the square size and the direction that produces the lowest number of 1D Wavelet coefficients above $T$ without generating distortions in the image.

The specific dyadic segmentation performed on the image will depend on its geometry. The optimal segmentation is found by dividing each scale of the 2D Wavelet Transform in a dyadic fashion until a minimum square size (usually 4x4 pixels) is reached. For each dyadic square of the division, the best direction $d$ is determined by point projection and the calculation of the Lagrangian $\mathscr{L}$Then, a successive bottom-up comparison of the Lagrangians of a square and its sub-squares is performed in order to determine if the latter are combined into the former. In Figure 11, for example, Lagrangians of sub-squares S421, S422, S423, and S424 will be compared with the Lagrangian of their immediate superior square S42 to determine whether to leave this square or to divide it into the four sub-squares mentioned.

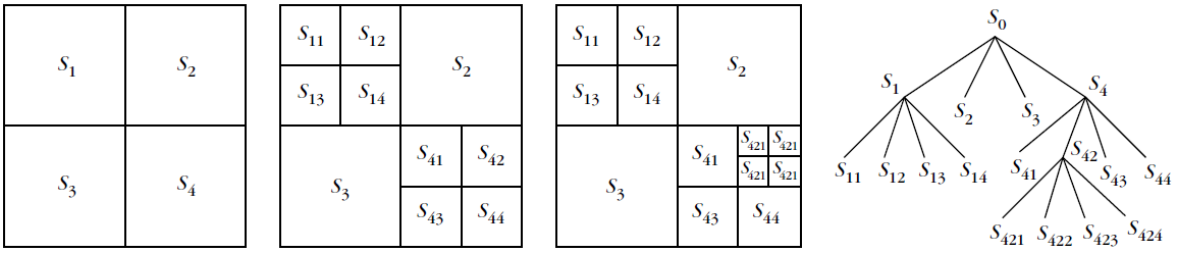
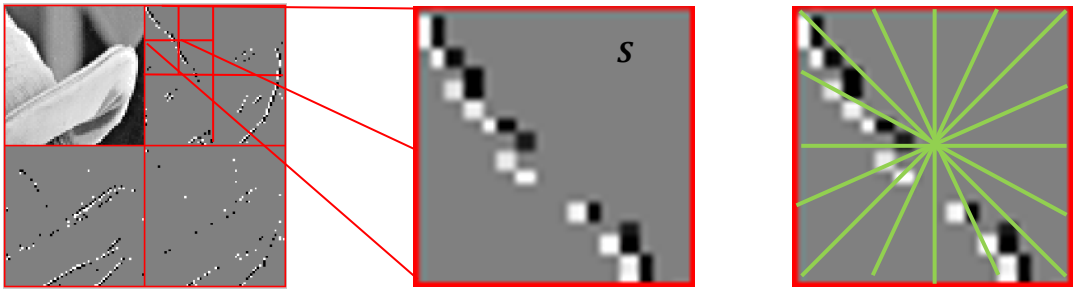


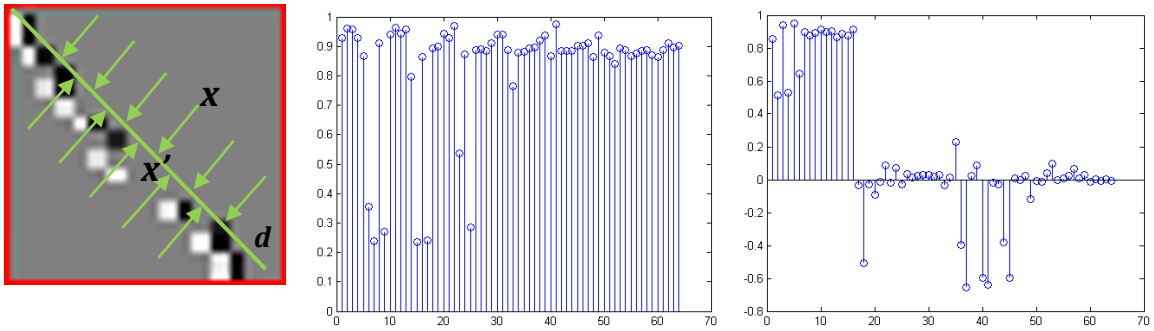

**Figure 11.** Construction of a dyadic segmentation by the successive division of squares. Squares $S_{41}$, $S_{42}$, $S_{43}$, y $S_{44}$ are sub-squares of square $S_4$. In turn, sub-squares $S_{421}$, $S_{422}$, $S_{423}$, y $S_{424}$ are sub-squares of square $S_{42}$ [19].




a) Dyadic Division of 2D Wavelet Transform.

b) Selection of a square *S* of size *L*.

c) Possible Geometric Directions.




d) Projection of square points.

e) 1D Signal from point projection.

f) Wavelet Transform of the 1D Signal.

**Figure 12.** Construction of Bandelet orthonormal bases considering the best geometry [5].

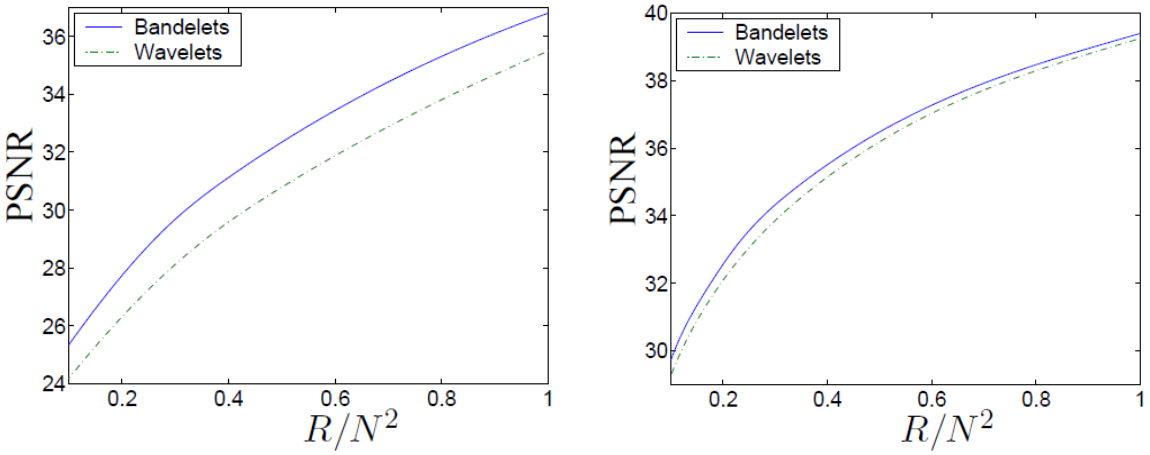
As an example of the construction of the Bandelet orthonormal bases for an image, Figure 13.a shows the geometric directions and the dyadic segmentation for Barb.bmp at a rate of 0.44 bpp [4]. Dyadic segmentation has adapted to the geometry of the image, having small squares in areas with the finest geometric details. Image areas where the variation of gray levels is monotone have no special geometry so they have no associated geometric directions. Figure 13.b shows the reconstructed image where it is noticeable the non-appearance of blocking artifacts. The blocking artifacts appear in an image compressed by the JPEG standard because of processing independent blocks of 8x8 pixels. In the Bandelet Transform such effects could also appear since the image is processed in separate squares defined by the dyadic segmentation. However, any blocking artifact caused by Bandelet processing is softened by the 2D Inverse Wavelet Transform applied always at the end of the Bandelet-based decompression system. Figure 14 compares the PSNR obtained for Barb.bmp of Figure 13 and Lena.bmp for various compression rates when they are compressed by Wavelets and Bandelets [4]. At all times, the Bandelet encoder has better results than the Wavelet encoder:

For Barb, the average difference between the two encoders is approximately 1.5 dB, whereas for Lena the difference is about 0.5 dB.

The selection of the best dyadic image segmentation is the most time-consuming process in the Bandelet Transform since the Lagrangian must be calculated for all possible directions of the complete set of squares of each possible dyadic segmentation. This gives the Bandelet Transform a higher complexity than the present in DCT and DWT.

a) Optimal dyadic segmentation. b) Reconstructed image.

**Figure 13.** Bandelet compression of Barb.bmp at 0.44 bpp, PSNR=31.3 dB [4].




a) Compression of Barb.bmp. b) Compression of Lena.bmp.

**Figure 14.** PSNR of Barb.bmp and Lena.bmp using Wavelets and Bandelets for various compression rates [4].

## 4.3. Bandelet Approximation Algorithm

The following section explains the details of the approximation algorithm for the implementation of the Bandelet Transform [5] adapted for the research work presented in this document.

### *4.3.1. Step 1: Input Algorithm*

For its implementation, the algorithm requires two inputs: The grayscale image to be compressed and a threshold $T$ that controls the level of compression obtained by the algorithm.

### *4.3.2. Step 2: Two-Dimensional DWT Transform*

The two-dimensional DWT is applied to the image (Figure 12.a). The result is an image at different scales $j$ in the orientations H-D-V. The steps 3 to 8 are repeated for each scale and each orientation.

### *4.3.3. Step 3: Image Segmentation*

The currently selected scale is divided into squares to process locally the geometry of the image (Figure 12.b). A dyadic division can be applied by dividing the image recursively into four equal-sized squares. Another option is to split the image into blocks of 8x8 or 16x16 pixels, similar to what is done in JPEG. This implies the appearance of blocking artifacts, which are smoothed at the end of the decompression process when the 2D Inverse DWT is performed.

### *4.3.4. Step 4: Selection of Geometry*

For each one of the squares in which the image was split, the pixels of the square are reordered in a one-dimensional array $f_d$ along a direction $d$ candidate (Figure 12.d and Figure 12.e).

### *4.3.5. Step 5: One-Dimensional Wavelet Transform*

The one-dimensional signal generated in the previous step is processed with the one-dimensional Wavelet Transform. The result is a signal $f_w$ that contains the 1D coefficients for the current direction $d$ (Figure 12.f).

### 4.3.6. Step 6: Calculation of the Lagrangian

Using the threshold $T$ given by the user, the Lagrangian $\mathscr{L}$ is computed for the coefficients from step 5 according to (4):

$$\mathscr{L} = E + AT^2 \tag{4}$$

Where $T$ is the threshold given by the user and $E$ and $A$ are respectively:

$$E = \sum_i \langle f_w^2(i) * [absolute\_value(f_w(i)) < T] \rangle \tag{5}$$

$$A = \sum_i \langle absolute\_value(f_w(i)) \geq T \rangle \tag{6}$$

### 4.3.7. Step 7: Selection of the Best Geometric Flow

In order to select the direction $d$ that best approximates to the local geometry of the image, it is chosen the one that minimizes the Lagrangian given by (4). This means that steps 4, 5, and 6 must be repeated for all possible geometric directions, maximum $2L^2$ for an $L$x$L$ pixel square.

### 4.3.8. Storage of Bandelet Coefficients

The 1D-Wavelet coefficients associate with the best direction $d$ for each square are stored in a particular fashion such as zig-zag ordering. These are the coefficients of the Bandelet Transform.

### 4.3.9. Selection of the Optimal Dyadic Segmentation

Once the best direction $d$ for each dyadic square has been determined, a bottom-up comparison is performed to choose the squares to be left in the segmentation. This requires a recursive comparison of the Lagrangian of each square and its respective sub-squares.

## 4.4. Encoding and Decoding of the Bandelet Coefficients

The Bandelet coefficients associated with the optimal image segmentation are the end product of the Approximation Algorithm. These coefficients can be encoded following any of the encoding techniques present in the JPEG and JPEG2000 standards. To decompress the

data, the algorithm is executed backwards. The group of Bandelet coefficients associated with each square of the segmentation is processed with the 1D Inverse Wavelet Transform and then they are projected onto the two-dimensional plane of the square according to the best direction *d* computed for the square in the compression process. Once this is done for all the squares of the segmentation, for all scales and orientations, the 2D Inverse Wavelet Transform is applied, obtaining the reconstructed image.

Decompression of data is less complex than compression because it is not necessary to determine the optimal image segmentation. This means that it is not required to calculate the Lagrangian for each possible direction in each square of the segmentation. However, in order to arrange the points of the squares in a corresponding fashion to the ordering done in the compression, the decompression system needs to know the segmentation performed on the image and the direction *d* chosen for each square. Therefore, this information must be encoded with the Bandelet coefficients produced by the Approximation Algorithm to obtain a decompressed image close to the original one.

The encoding of these additional data (the optimal segmentation and the best direction of each square) can increase the total number of bytes that are needed for encoding an image using Bandelets compared to a Wavelet encoding. However, this information can be omitted in the encoded data if the Bandelet encoder arranges the coefficients in a standard way that is always identifiable by the decoder to decompress the image without errors.

# *Chapter 5*

# *Implementation of JPEG and JPEG2000 Standards*

In order to learn more about the current methods of image compression before implementing the codec based on the Bandelet Transform, I implemented two image processing platforms for JPEG and JPEG2000 standards. These implementations were designed as a SoPC system, in which the NIOS II processor was configured in an FPGA together with a set of logical blocks described in VHDL and Verilog. The routines running on the NIOS II processor were written in C, having a hardware-software co-design.

## 5.1. JPEG File Viewer using NIOS II processor

The first project was designed as an FPGA-based portable system for displaying images stored in JPEG [23]. As a storage device, a SD-Card was selected that can be easily formatted and written in a personal computer. The selected file system was FAT16, also known as FAT in Windows platforms. As an output device, a 3.6-inch LCD panel with a resolution of 320x240 pixels was chosen, thus allowing the system to be portable. The image is decoded in software. Figure 15 shows the block diagram of the system according to the above specifications. The system was implemented in an Altera DE2 board (Figure 16). This board features a Cyclone II EP2C35F with 33.216 logic elements and 483.840 memory bits. This device allows the implementation of systems with multiple clock domains through its four PLLs (Phase-Locked Loop) and has 70 9-bit embedded multipliers to optimize operations related to digital signal processing. For the configuration of the FPGA during power-up, there is a serial configuration device EPCS16.

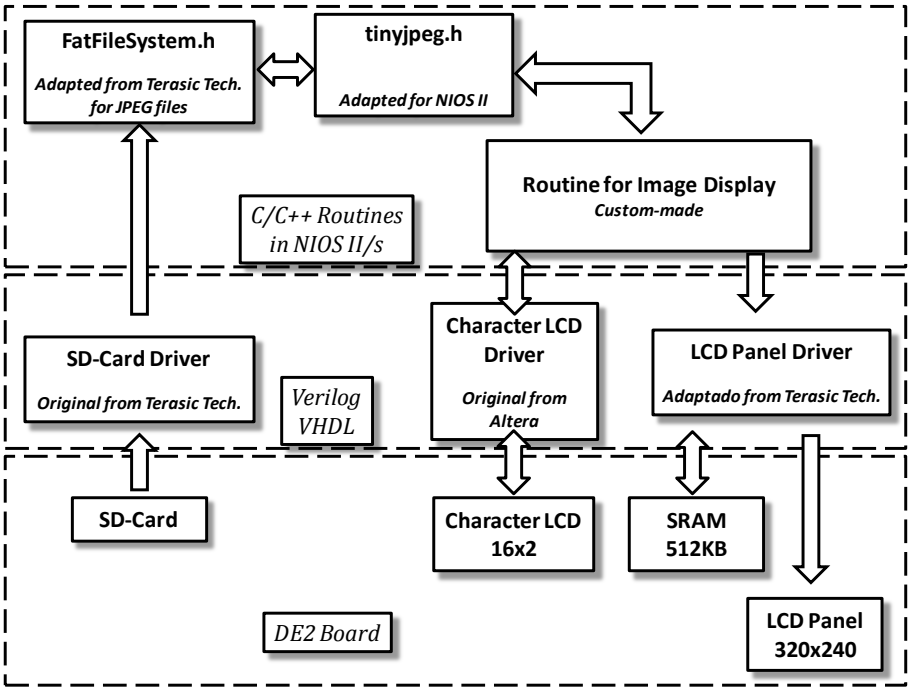



**Figure 15.** Block diagram of the JPEG File Viewer.

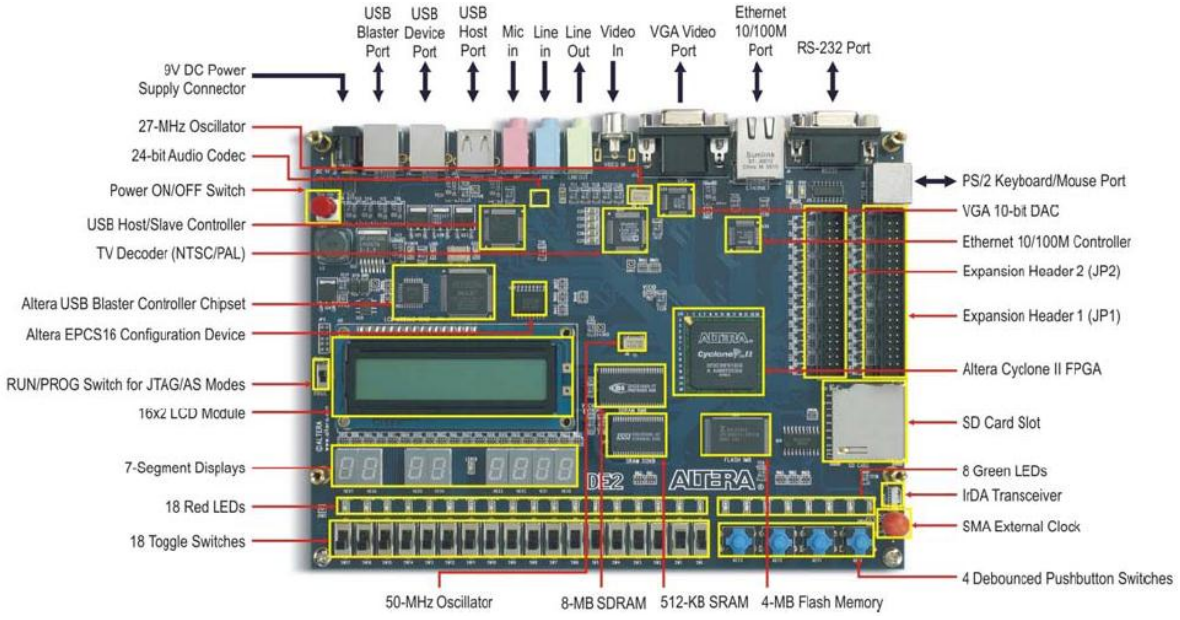



**Figure 16.** Altera DE2 Board [24].

The DE2 board resources used by the application are presented in the bottom of Figure 15. The board has a socket to insert the SD-Card, while the LCD panel is connected with a cable ATA (IDE). The LCD panel used is the TRDB_LCM by Terasic Technologies. The 512-KB SRAM is used as image buffer of the LCD panel. The hardware blocks written in Verilog and VHDL for the system are in the middle of Figure 15. SD-Card and character LCD drivers are original from Altera: the first is part of the design examples of the DE2 board, and the second is an IP core from Altera University Program. The LCD panel controller was designed based on the DE2-board examples [25]. The software routines written in C and running on the NIOS II processor are presented in the top of Figure 15. The FatFileSystem.h routine reads the contents of the SD-Card and builds the file directory [25]. Once the file directory is ready, the decompression of each one of the images is executed with the software decoder tinyjpegdecoder, developed by Luc Saillard and distributed under GNU public license [26]. The resulting decompressed data are passed to the routine responsible for writing the image buffer of the LCD panel – the SRAM memory. Other components of the DE2 board that are used but not shown in Figure 15 are 8-MB SDRAM for data and program memory (its driver is an IP core from Altera), push- buttons to restart the system and move to the next file, and the serial configuration device EPCS16.

During the implementation of the JPEG File Viewer, an entirely hardware JPEG decoder for the DE2 board was also designed. Tests showed that the images decompressed with this decoder were of lower quality than those obtained with the tinyjpegdecoder, mainly due to the truncation of data in the hardware decoder, which reduced the accuracy of the calculations. For this reason, it was decided to leave the software decoder, which also allowed configuring more widely the different characteristics of the JPEG standard.

### *5.1.1. NIOS II Embedded Processor*

The NIOS II is a soft-core processor designed by Altera and available as an IP core. This means that is implemented in an FPGA using any available resources, unlike hard-core processors such as Altera Excalibur and Xilinx PowerPC, which are implemented in the FPGA using specific and dedicated resources to them. It is available in three versions: NIOS II/e or economy, NIOS II/s or standard, and NIOS II/f or fast. NIOS II is a RISC general-purpose processor with a 32-bit Load-Store instruction set supported by 32 general-purpose registers.

It can handle up to 32 external interrupts and it is programmed using C/C++. Its internal architecture is shown in Figure 17. The blocks enclosed by a dotted line are optional: it can be omitted or configured to meet the design specifications. In this way, designers are able to define the cache memory size, the inclusion or exclusion of certain debugging features, and the execution mode of operations such as multiplication and division, whether in hardware or emulated in software. The custom instructions in Figure 17 are logic blocks connected in the processor datapath. These blocks accelerate critical software functions through its implementation as ALU instructions (Figure 18). An example of these custom instructions is the set of floating-point instructions that implements the four basic operations (addition, subtraction, multiplication and division) alongside functions from the ANSI C math library like *sin()*, *cos()*, *log()* and *floor()*. When the floating-point custom instructions are not used, the processor emulates these operations in software, increasing the execution time of the application.

NIOS II processor is integrated into a SoPC system using the SoPC Builder tool of Altera Quartus II. This tool allows the designer to select and configure the processor and peripherals as a memory-mapped system. The SoPC Builder automatically generates the data and instruction buses, along with the elements of bus arbitration, instruction decoding, and interrupt handling. Features such as priority interrupts, memory-base addresses, and clock domains are also configurable inside the SoPC Builder.

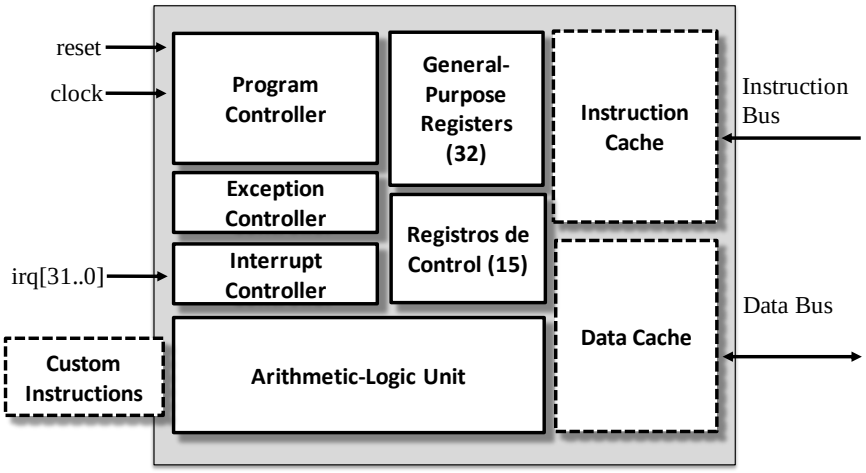



**Figure 17.** Internal architecture of the NIOS II processor [27].

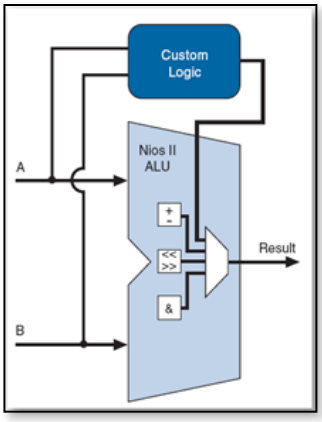



**Figure 18.** Custom instructions connected to processor ALU [28].

Once the system has been captured with this tool, it can be integrated into a hardware design using schematic capture or HDL language. The software routines for the processor are developed in C/C++ using the NIOS II Integrated Development Environment (NIOS II IDE). The processor selected for the JPEG File Viewer was a NIOS II/s with 4KB instruction cache, operating at a frequency of 100 MHz. This processor has no data cache (as opposed to the NIOS II/f that has both data and instruction cache).

## *5.1.2. Hardware Design*

The principal hardware blocks of the JPEG viewer are three: SD-Card driver, LCD Panel driver, and character LCD driver.

### *5.1.2.1. SD-Card Driver*

This driver implements the SD 1-bit protocol using a data line, a command line, and a clock line. It is written in Verilog and is integrated into the system using the SoPC Builder. This driver is used by the routine responsible for constructing the FAT file directory of the card.

### *5.1.2.2. LCD Panel Driver*

The LCD panel driver generates the necessary timing signals for correct operation of the panel and controls the transmission of RGB signals at a pixel data rate of 18.42 MHz. The color data are sent to the panel serially, one component after the other, unlike VGA monitors that receive RGB components in parallel. When the pixels are not being refreshed, the controller reads the data sent by the NIOS II processor and writes the image buffer using back-buffering. With this

technique, the SRAM memory is divided into two buffers, in which the processes of writing and reading images alternate. On startup, the controller displays on the LCD panel the image stored in the first buffer, while in parallel it writes the next image in the second buffer. The writing process is performed when the controller is not refreshing the pixels of the LCD Panel. Once the next image to be displayed is completely written in the second buffer, the driver displays it in the LCD Panel and begins to write a new image in the first buffer and so on. The back-buffering technique allows images to be displayed on the LCD screen without black flickering as a result of SRAM memory access during the writing process. The RGB data are stored in SRAM memory with only 15 bits in order to save resources, discarding the three least-significant bits of each component. These bits are replaced with zeros when they are sent to the panel.

#### *5.1.2.3. Character LCD Driver*

The character LCD has 16 characters, 2 lines with a data bus of 8 bits. This LCD is used for information display like the number of files found in the SD-Card, the name of the file currently being decompressed, and the time it took the decoding process of the image.

### *5.1.3. Software Routines*

The software routines were developed with the NIOS II IDE, which is based on the Eclipse platform. JPEG File Viewer has three main routines: Construction of the FAT file system, JPEG decoder, and Image Display. All routines were written in C.

#### *5.1.3.1. Construction of the FAT File System*

This routine determines if a SD-Card is connected to the system. If so, it builds the list of JPEG files that are in it (up to 128 files). Once the list is available, each file can be opened, closed and read in packets of 512 bytes. This routine is based on an example of DE2 board developed for WAV files [25].

#### *5.1.3.2. Jpeg Decoder*

This routine generates RGB data from the JPEG compressed file. The decoder used is tinyjpegdecoder by Luc Saillard [26]. The great advantage of this decoder is that it consists of only five files, facilitating the code analysis and use.

#### *5.1.3.3. Image Display*

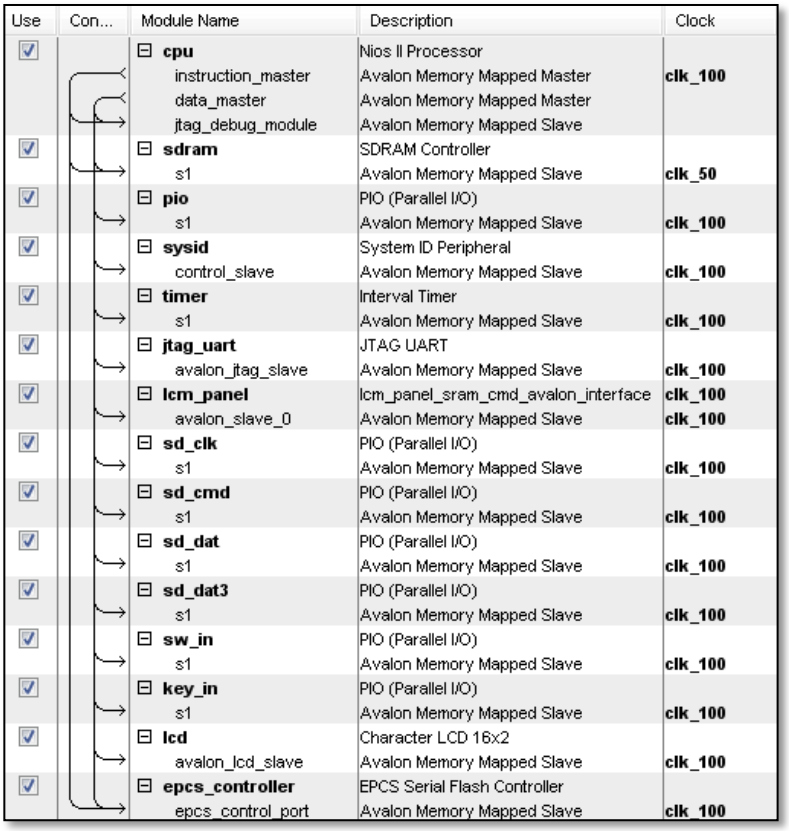
This routine receives the RGB data generated by the tinyjpegdecoder and writes them in the SRAM memory as 15-bit words (5 bits per color). Also, it handles the character LCD and calls the routines used to measure the execution times of the application:

- *alt_timestamp_start ():* Initialize the system timer counter.
- *t1 = alt_timestamp ():* Returns an integer with the current value of the counter.
- *freq = alt_timestamp_freq ():* Returns an integer with the number of clock cycles per second the counter is working with. This value is the clock frequency associated with the timer in the system designed in the SoPC Builder.

### *5.1.4. Block Integration*

Following the application requirements and using the hardware blocks mentioned above, the architecture of the system was captured with the SoPC Builder (Figure 19). The system has two clock domains: a 50MHz domain for the SDRAM memory and a 100MHz domain of the processor and other components. SDRAM memory is in a different clock domain due to clock synchronization issues (clock skew) associated with this device on the DE2 board, which are prevented by generating a clock frequency of 50 MHz with a -3ns phase using a PLL. This phase ensures that the address, data, and control signals of the SDRAM remain stable during the positive edge of clock. The 100MHz clock for the processor is generated using a Cyclone II PLL having as a reference the 50MHz oscillator of DE2 board. To generate the 18.42 MHz clock for the LCD panel, the design uses a second PLL referencing the 27 MHz oscillator present on the board. The system also includes a SYSID component that verifies that the executable software produced by NIOS II IDE has been compiled for the hardware system programmed into the FPGA.

According to the characteristics of the hardware component, the SoPC Builder determines whether it needs to be connected to the instruction bus and/or to the data bus. SDRAM memory for example, is connected to the two buses since it works as program and data memory; the character LCD, in turn, is connected only to the data bus because it is a processor slave.



| Use | Con... | Module Name | Description | Clock |
|---|---|---|---|---|
| ☑ | | **cpu** | Nios II Processor | |
| | | instruction_master | Avalon Memory Mapped Master | **clk_100** |
| | | data_master | Avalon Memory Mapped Master | |
| | | jtag_debug_module | Avalon Memory Mapped Slave | |
| ☑ | | **sdram** | SDRAM Controller | |
| | | s1 | Avalon Memory Mapped Slave | **clk_50** |
| ☑ | | **pio** | PIO (Parallel I/O) | |
| | | s1 | Avalon Memory Mapped Slave | **clk_100** |
| ☑ | | **sysid** | System ID Peripheral | |
| | | control_slave | Avalon Memory Mapped Slave | **clk_100** |
| ☑ | | **timer** | Interval Timer | |
| | | s1 | Avalon Memory Mapped Slave | **clk_100** |
| ☑ | | **jtag_uart** | JTAG UART | |
| | | avalon_jtag_slave | Avalon Memory Mapped Slave | **clk_100** |
| ☑ | | **lcm_panel** | lcm_panel_sram_cmd_avalon_interface | **clk_100** |
| | | avalon_slave_0 | Avalon Memory Mapped Slave | **clk_100** |
| ☑ | | **sd_clk** | PIO (Parallel I/O) | |
| | | s1 | Avalon Memory Mapped Slave | **clk_100** |
| ☑ | | **sd_cmd** | PIO (Parallel I/O) | |
| | | s1 | Avalon Memory Mapped Slave | **clk_100** |
| ☑ | | **sd_dat** | PIO (Parallel I/O) | |
| | | s1 | Avalon Memory Mapped Slave | **clk_100** |
| ☑ | | **sd_dat3** | PIO (Parallel I/O) | |
| | | s1 | Avalon Memory Mapped Slave | **clk_100** |
| ☑ | | **sw_in** | PIO (Parallel I/O) | |
| | | s1 | Avalon Memory Mapped Slave | **clk_100** |
| ☑ | | **key_in** | PIO (Parallel I/O) | |
| | | s1 | Avalon Memory Mapped Slave | **clk_100** |
| ☑ | | **lcd** | Character LCD 16x2 | |
| | | avalon_lcd_slave | Avalon Memory Mapped Slave | **clk_100** |
| ☑ | | **epcs_controller** | EPCS Serial Flash Controller | |
| | | epcs_control_port | Avalon Memory Mapped Slave | **clk_100** |

**Figure 19.** System capture of the JPEG File Viewer using SoPC Builder.

Along with system generation, SoPC Builder produces the HDL files necessary to declare the system as a component in a Verilog or VHDL file, or to use it in a schematic capture (Figure 20). In order to evaluate the advantage of including floating-point custom instructions into the processor, two systems were generated, one where floating-point operations are emulated in software, and another where the custom instructions are included.

### *5.1.5. Resources Used by the System*

The system with floating-point instructions emulated in software used 4.052 of 33.216 logic elements in the Cyclone II, equivalent to 12%. The FPGA internal memory was 50.483 of 483.840 bits, which included 32.768 bits for instruction cache memory, 1.024 bits for the JTAG module responsible for handling communication with the host computer, and 4.096 for the EPCS16 serial configuration device driver.


**Figure 20.** Schematic capture symbol generated by SoPC Builder for JPEG Filer Viewer.

The second system used 5.379 logic elements, i.e. 4% more than the first system. The additional 1.327 logic elements were used by the floating-point unit added to the ALU. This unit also required 147 bits of memory, a 0.03% increase in the total amount of memory used by the first system.

### *5.1.6. Performance Evaluation Results*

Four JPEG files were used to verify correct operation of the application (Figure 21). The resolution of all images was 320x240 pixels. The file sizes were 2.73 KB for the image 1 (top left), 41.1 KB for the image 2 (top right), 8.66 KB for the image 3 (bottom left), and 57 KB for image 4 (bottom right). As shown, the smaller file corresponds to image 1 since it has very few color changes, which is reflected in a higher compression using RLE and Huffman coding in the JPEG standard. Table 1 shows the acceleration factors introduced in the decoding process when the ALU has floating-point instructions; the average acceleration factor is 2.58.

**Figure 21.** Test images used in the JPEG File Viewer.

**TABLE I**

ACCELARATION FACTORS INTRODUCED IN JPEG FILE VIEWER BY FLOATING-POINT CUSTOM INSTRUCTIONS

| Image | Instrucctions Emulated in Software [s] | Instructions Implemented in Hardware [s] | Acceleration Factor |
|---|---|---|---|
| 1 | 15.43 | 6.57 | 2.35 |
| 2 | 31.79 | 11.23 | 2.83 |
| 3 | 33.37 | 11.67 | 2.86 |
| 4 | 42.95 | 18.79 | 2.29 |

As expected, the custom instructions increased system performance, requiring few hardware resources of the FPGA (4% more logic elements and 0.03% bits of internal memory).

## 5.2. Image Editor with JPEG/JPEG2000 Compressor

JPEG File Viewer was complemented with image editing and compression capabilities in JPEG and JPEG2000 standards. In this case, a DE2-70 board was used, which is very similar to the DE2 board shown in Figure 16, but with a Cyclone II EP2C70 with 68.416 logic elements, two 32MB SDRAM, and one 2MB SSRAM. The block diagram of this design is presented in Figure 22. The system uses 8.350 logic elements (12%) and 179.731 memory bits (16%) of the FPGA.

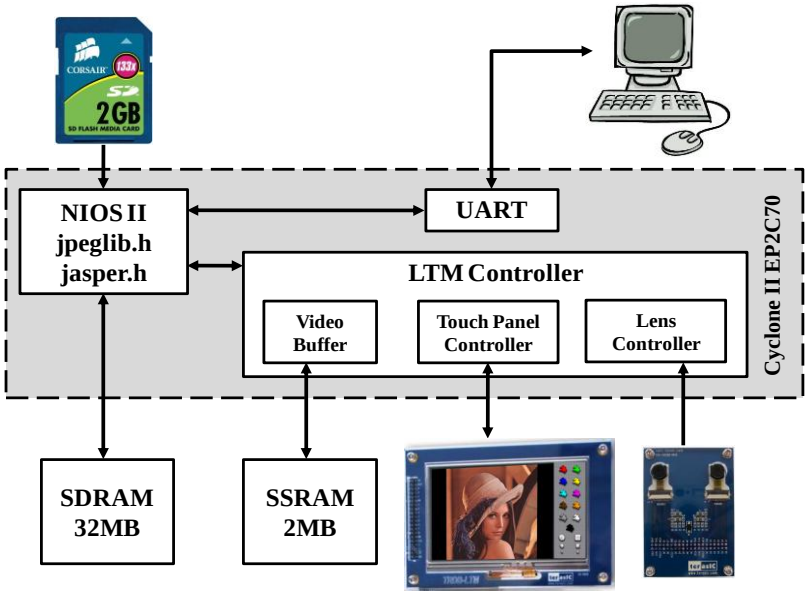



**Figure 22.** Block diagram of the Image Editor with JPEG/JPEG2000 Compressor.

### *5.2.1. System Architecture*

The 320x240 pixel LCD panel was replaced with a Touch Panel (LTM-LCD Touch Panel Module) of 800x480 pixels; SSRAM memory working as image buffer as in the JPEG file viewer. The system also has a 1.3Mega-pixel CMOS lens. Both the panel and the lens are manufactured by Terasic and theirs drivers are written in VHDL and Verilog. The system is controlled by a processor NIOS II/f with data cache and instruction cache, 4 kB each. The processor is configured with floating-point custom instructions and runs at a 100 MHz frequency. The data and program memory are located in the SDRAM, which runs at the same clock frequency of the processor but with a phase of -1.76 ns to prevent synchronization problems. The system keeps SD-Card as storage device for the files that will be displayed on the screen, which for this design can be JPEG or BMP.

Inside NIOS II processor, there are three main routines for reading files and encoding images: The tinyjpegdecoder, the jpeglib library, and the JasPer library. The tinyjpegdecoder, also used in the JPEG File Viewer, decompress JPEG files. The jpeglib library, distributed by the Independent JPEG Group, encodes a set of RGB data into a JPEG file [29]. Finally, the JasPer library contains all the necessary functions for JPEG2000 compression [30], [31]. Additional routines include those for BMP file reading, SD-Card reading, and character LCD handling.

JasPer is a free software implementation of JPEG2000 codec that is included in Part 6 of the standard as reference software for testing and validation of JPEG2000 systems. A second software implementation of this codec that is recommended by the standard is Kakadu, which is not free. JasPer usage is subject to the open source MIT license, which allows using this software in open source or commercial applications. Kakadu has been successfully implemented in a FPGA using the NIOS II processor as a coprocessor (see Section 2.3: *NIOS II Processor-Based Hardware / Software Co-Design of the JPEG2000 Standard*). In the other hand, JasPer has not been integrated into NIOS II systems, but it is part of several image processing applications for desktops and laptops.

### *5.2.2. System Features*

The system allows editing an image and encoding it as a JPG or JP2 file (JP2 is the file extension for JPEG2000 standard). The image the user edits can be a new image created by the user from the Touch Panel, a BMP or JPG file stored on the SD-Card, or a photographic capture from the 1.3Mega-pixel lens. The user chooses one of these three options by selecting it on the Touch Panel at the beginning of the execution of the system (Figure 23.a)[4].

The process of reading a BMP or JPG file from the SD-Card is performed by the NIOS II processor using FAT file routines. New image creation, edition, and photographic capture in turn are made entirely in hardware. The image edition is performed directly on the image buffer of the Touch Panel, located on SSRAM memory. When the user has finished editing the image (new, file, or photograph), NIOS II reads the SSRAM content and writes it in its data memory (SDRAM) as a data array. This array is the input for jpeglib.h and jasper.h library functions to encode the edited image in a JPG and a JP2 file respectively. To verify that the generated files are valid FAT files, the processor sends them to a host computer via the system UART. On the computer, an application designed in Visual Basic receives the files and stores them in a designated folder. Since JPEG2000 standard has limited use in common computer systems, it is necessary to have programs installed in the host computer that supports JP2 files, such as Adobe Photoshop with JP2 plug-in or Apple Quick Time Player.

[4] Videos of the system are available in: http://bionano.univalle.edu.co/english/jpeg_jpeg2000compressor.html


a) Boot image to select the image to be edited.

b) Color and brush palettes to create a new image.

c) Edition of an image file stored in the SD-Card.

d) Photographic capture from the 1.3Mega-pixel lens.

**Figure 23.** Shots of the Image Editor with JPEG/JPEG2000 Compressor running.

Figure 24 shows a screenshot of a host computer where the Visual Basic application is running. Figure 25 shows the JP2 file sent by the system to the host computer. This file corresponds to the edition of the photographic image in Figure 23.d.

### *5.2.3. Touch Panel Module*

The main component of the Image Editor with JPEG/JPEG2000 Compressor is the Touch Panel that performs two main functions: Image display and user input device. The use of this component in the system was considered of great importance given the large number of electronic devices with touch screens that are now available on the market. The Touch Panel has three main components that connect to the card through an ATA connector (Figure 26): A 12-bit ADC converter, an LCD controller, and an LCD timing controller [32].

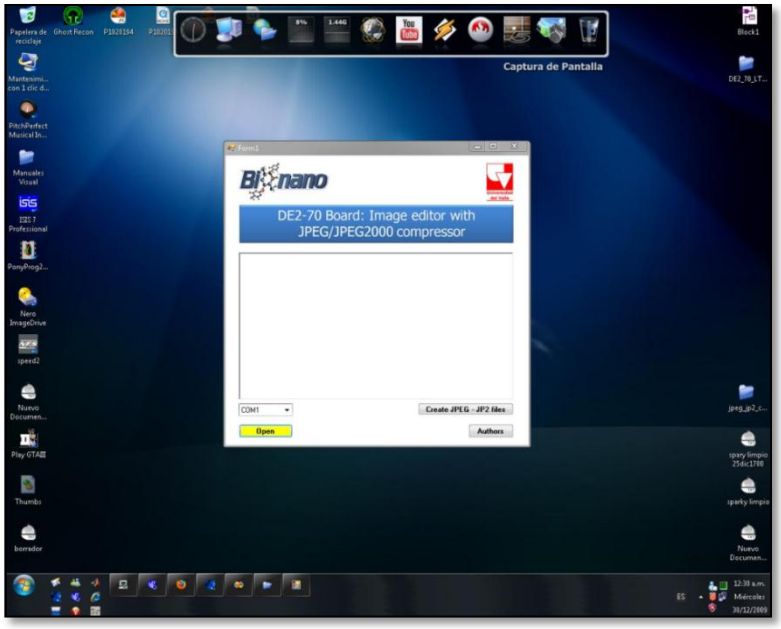



**Figure 24.** Visual Basic Application running on the host computer to communicate with the Image Editor.

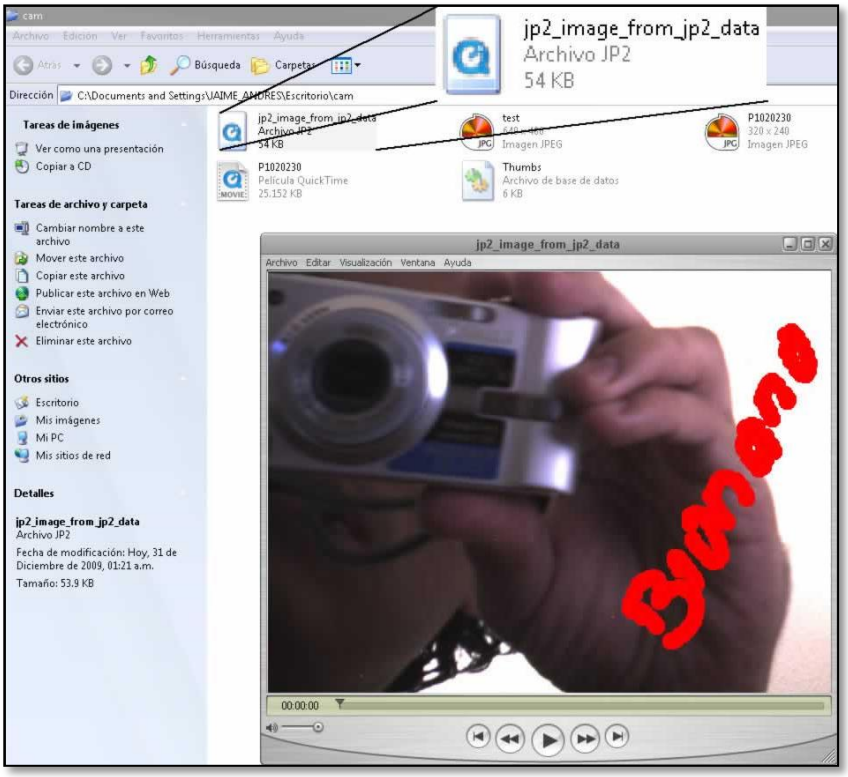



**Figure 25.** JP2 file sent by the Image Editor with JPEG/JPEG2000 compressor stored on the host computer.

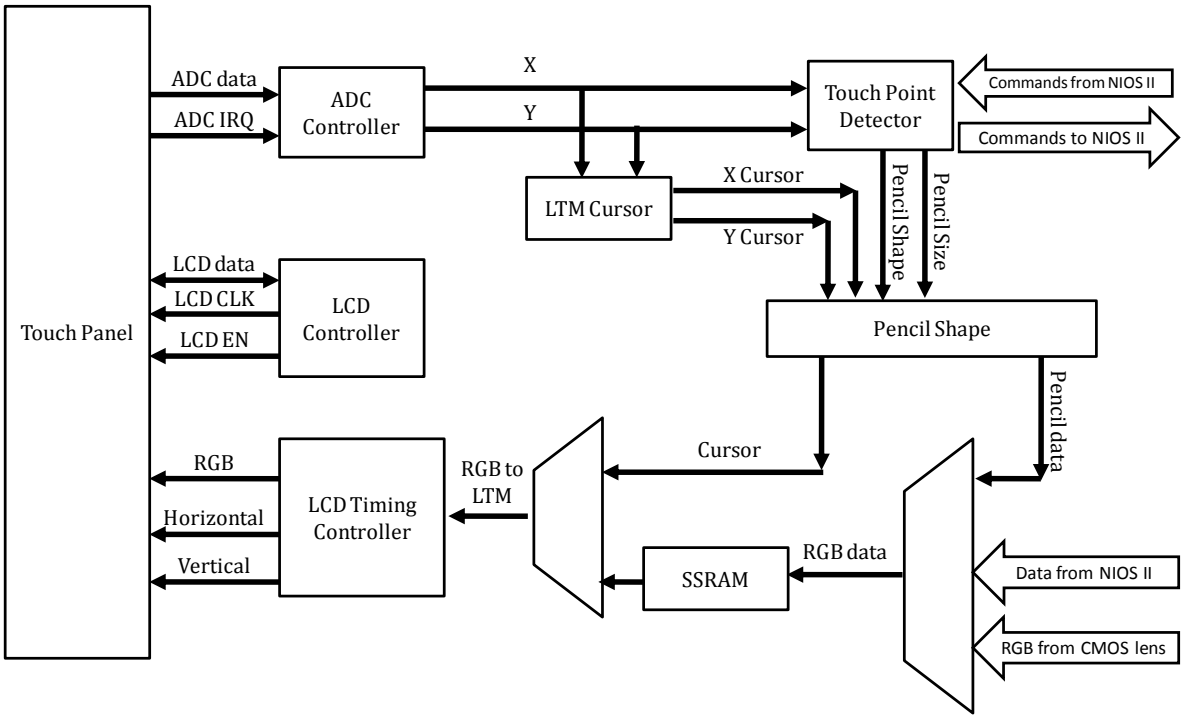



**Figure 26.** Touch Panel Controller in the Image Editor with JPEG/JPEG2000 compressor.

The ADC is responsible for converting the analog coordinates *x,y* of the points where the user touches the screen to its digital representation for the FPGA. The LCD controller uses a register bank to configure timing and power characteristics, synchronization signals, and resolution of the screen. The LCD timing controller sends the RGB data of each pixel to the Touch Panel alongside the vertical and horizontal synchronization signals. RGB data is sent to the Touch Panel in a two-layer configuration, where the bottom layer corresponds to the current content of the image buffer and the top layer to the cursor the user employs to edit an image.

The cursor position given by the user in the edition process is controlled by the LTM Cursor. This component translates the coordinates *x,y* produced by the ADC to the corresponding pixels of the Touch-Panel display area in order to paint the cursor at the point where the user touched the screen. This conversion is necessary because the ADC returns the coordinates *x,y* of the point where the user touched the screen in the range [0, 4095], [0, 4095], while the display area resolution of the panel is 800x480 pixels. The Touch Point Detector interprets the commands sent by the user in the Touch Panel screen using the coordinates *x,y* returned from the ADC and the commands sent by the NIOS II processor. Thus, the system knows the

screen that is currently being displayed (boot image - Figure 23.a, image editing - Figure 23.b and Figure 23.c, or photographic capture - Figure 23.d) and the commands the user has access to. The Touch Point detector informs the NIOS II processor of the command selected by the user and configures the cursor and other hardware components accordingly.

The data stored in the image buffer (SSRAM) can come from three different sources. One source is the data sent by the NIOS II processor which correspond to the BMP and JPG files that are read from the SD-Card. The second source is the RGB data sent from the 1.3Mega-pixel CMOS lens when the user chooses to perform the photographic capture. Finally, data stored in the SSRAM can come from the changes made to the image when the user is editing it with the cursor on the screen. The selection of the SSRAM data source will depend on the commands sent by the user from the Touch Panel that are later decoded by the Touch Point Detector. The content of the SSRAM is also read by the NIOS II processor when the user wants to compress the edited image using JPEG and JPEG2000.

### *5.2.4. Performance Results*

The performance evaluation of the Image Editor with JPEG/JPEG2000 Compressor was focused on the operation of the JPEG2000 encoder. The library JasPer was tested for different compression rates using color Lena.bmp. The encoder was configured for six levels of decomposition, using blocks of 64x64 samples. For each compression ratio, the corresponding PSNR was calculated for the compressed image with JasPer encoder, as well as for the compressed image with the library jpeglib; the mathematical definition of PSNR for an 8-bit image is given by (2). To calculate the PSNR, the compressed image in the NIOS II with the JasPer encoder was firstly converted to a BMP file on the host computer. Thereafter, this BMP file and the JPEG compressed file with jpeglib library were converted to an YCbCr representation in Matlab. From this representation, only the Luminance components were used to calculate the PSNR of each image using Simulink blocks. Figure 27 summarizes the results. According to this graphic, images encoded with the JPEG2000 encoder have better PSNR values in comparison with those encoded with the JPEG encoder for all the compression rates. This improvement is more evident with high compression ratios than with low ones: For a rate of 106:1, the difference is 4.56 dB, compared with a difference of 0.15 dB when the rate is 8:1.

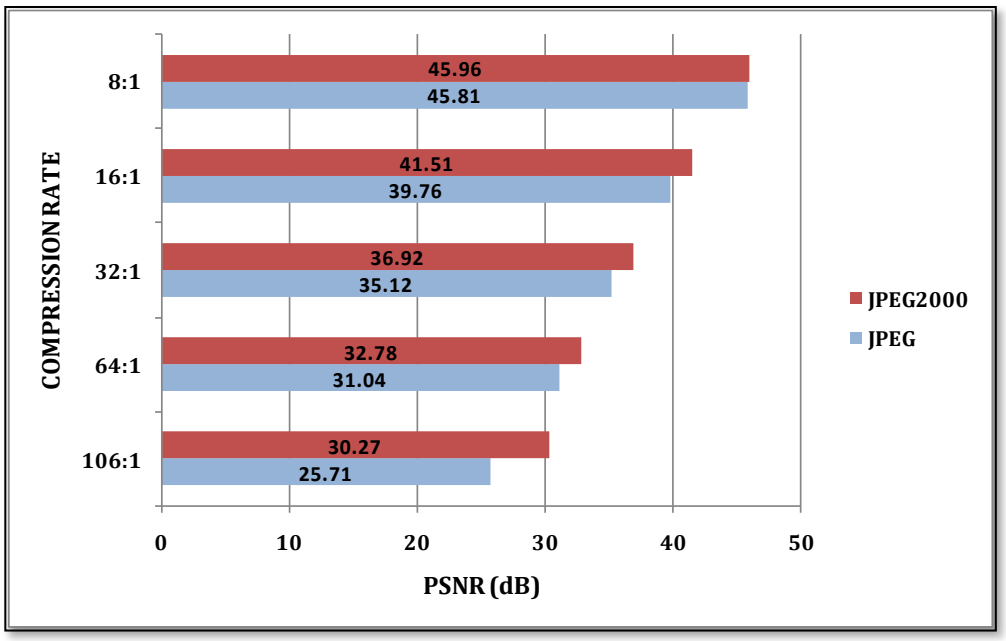



**Figure 27.** PSNR for Lena.bmp compressed with JPEG2000 and JPEG encoders for different rates.

The superiority of the JPEG2000 encoder over its JPEG counterpart is also clear when the reconstructed images are visually evaluated. In Figure 28, the JPEG2000 file is 2154 bytes long, while the JPEG file size is 2163 bytes. For this compression rate (106:1), JPEG2000 image is still similar to the original one. However, JPEG image has lost a great deal of information and Lena figure is almost vanished. Moreover, in the JPEG image is clear the appearance of blocking artifacts, unlike JPEG2000 image, where such distortions are not present.

The difference between the two encoders is not only noticeable at high compression rates. As seen in Figure 29, the eye in the JPEG image has more distortions than in the JPEG2000 when compared with the original image. Although the distortions of Figure 29 are small, they give a general idea of the behavior of JPEG2000 and JPEG encoders in more demanding applications such as medical applications.

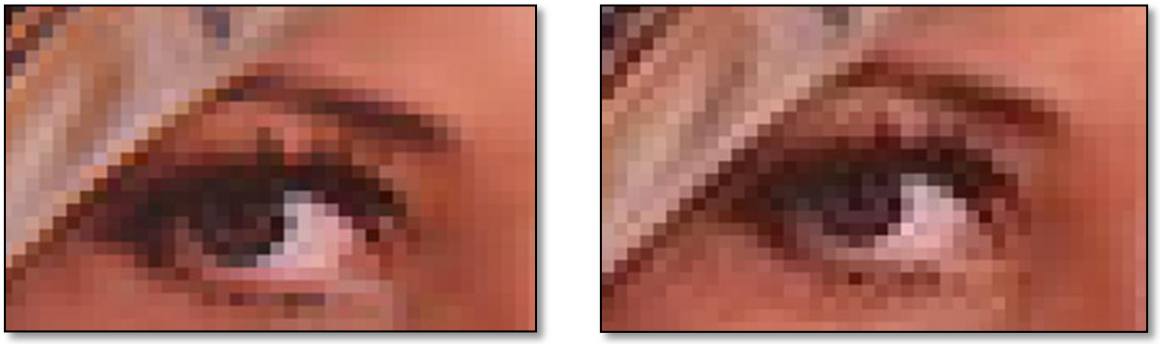
a) Original Lena.bmp.

b) Lena.bmp compressed with JPEG2000 (2154 bytes).

c) Lena.bmp compressed with JPEG (2163 bytes).

**Figure 28.** Lena.bmp compression with Image Editor with JPEG/JPEG2000 Compressor at 106:1.

a) Original section of Lena.bmp.



b) Section of Lena.bmp compressed with JPEG2000: PSNR=41.51 dB.

c) Section of Lena.bmp compressed with JPEG: PSNR=39.76 dB.

**Figure 29.** Region of Lena's eye compressed at 16:1.

# *Chapter 6*

# *Design of the Codec Based on the Bandelet Transform*

Based on the Bandelet Approximation Algorithm [5] and the Matlab toolbox [11] by G. Peyré and S. Mallat, a Bandelet codec for grayscale images was designed, which was implemented on the Image Editor with JPEG/JPEG2000 Compressor presented in Chapter 5. This codec includes both the Bandelet image compressor and decompressor in order to have at the exit of the system the reconstructed image for later comparison with the original image. The performance of the Bandelet codec is compared with a 2D Wavelet codec for several grayscale images in Chapter 7. The codec was designed in language C; some of its software functions were later accelerated with Altera NIOS II C2H Compiler.

The image encoder based on the Bandelet Transform is composed of three main blocks: 2D Wavelet Transform, Image Segmentation and Extraction of Points (Figure 30). The system output are the Bandelet coefficients, which can be encoded with any of the techniques defined by JPEG or JPEG2000 standards (RLE, Huffman, arithmetic encoding, etc.). The design of the Bandelet-coefficient encoding stage and of several processes of the codec as hardware blocks is part of the recommendations made for future work for this research project at the end of this document.

## 6.1. 2D Wavelet Transform

The 2D Wavelet Transform is the entry point of the encoder and generates the 2D Wavelet coefficients in the Horizontal, Diagonal, and Vertical (H-D-V) orientations for each scale of decomposition. The maximum number of scales $J_{max}$ in which an image of size $NxN$ pixels can be decomposed is given by (7):

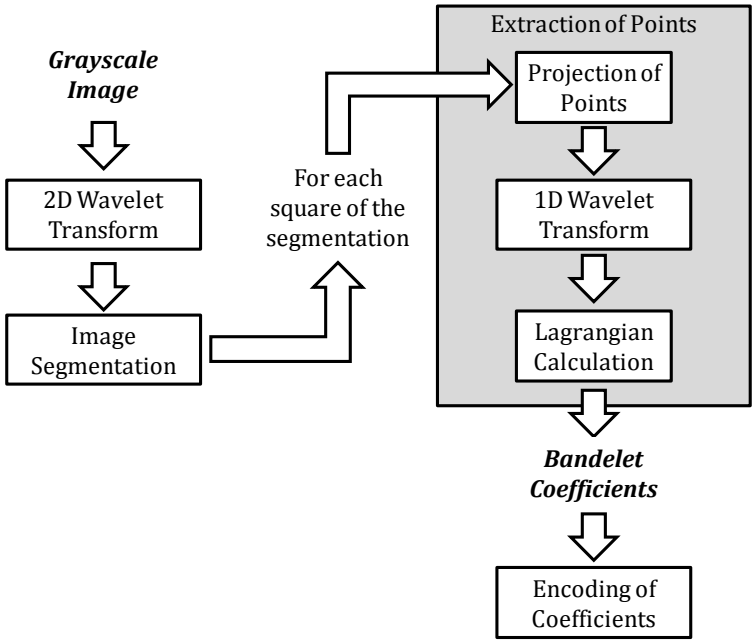



**Figure 30.** Block diagram of the image encoder based on the Bandelet Transform.

$$J_{max} = log_2(N) - 1 \tag{7}$$

The transform is performed firstly on the rows of the image and then on the columns because this is a separable transform. As recommended by the JPEG2000 standard, the filters used in the 2D Wavelet Transform (as well as in the 1D Wavelet Transform of the Extraction of Points block) are the odd-length filters Le Gall 5/3. The numbers indicate the size of the low-pass filter (5 coefficients or taps) and high-pass filter (3 coefficients or taps) used in the decomposition or analysis of the signal [15]:

- ***5-tap low-pass filter:***

  $h_0$=-1/8; $h_1$=1/4; $h_2$=3/4; $h_3$=1/4; $h_4$= -1/8;
- ***3-tap high-pass filter:***

  $g_0$=-1/2; $g_1$=1; $g_2$=-1/2;

The synthesis or reconstruction filters of the bank are:

- ***3-tap low-pass filter:***

  $hi_0$=1/2; $hi_1$=1; $hi_2$=1/2;

- ***5-tap high-pass filter:***

  $gi_0$=-1/8; $gi_1$=-1/4; $gi_2$=3/4; $gi_3$=-1/4; $gi_4$=-1/8;

The filters Le Gall 5/3 are Quadrature Mirror Filters (QMF) that allows achieving an exact reconstruction of the signal processed with the Wavelet Transform. These filters are FIR and symmetrical, which means that their phase is linear and they can be implemented using half of the number of multiplications plus one required originally. For example, in the low-pass decomposition filter $h_0$=$h_4$, $h_1$=$h_2$; hence it requires three multiplications instead of five (Figure 31). The same applies for the high-pass decomposition filter and the synthesis filters.

```
int filter5(int h0, int h1, int h2,
            int h3, int h4, int x4,
            int x3, int x2, int x1,
            int x0)
{
    int s0, s1, s2, s3, s4, s;
    s0=h0   *   xx4;
    s1=h1   *   xx3;
    s2=h2   *   xx2;
    s3=h3   *   xx1;
    s4=h4   *   xx0;
    s=s0+s1+s2+s3+s4;

    return s;
}
```

a) 5-tap filter in canonical implementation.

```
int filter5(int h0, int h1, int h2,
            int x4, int x3, int x2,
            int x1, int x0)
{
    int s0, s1, s2, s;
    s0=h0   *   (x4+x0);
    s1=h1   *   (x3+x1);
    s2=h2   *   (x2);
    s=s0+s1+s2;

    return s;
}
```

b) 5-tap filter in symmetric implementation.

**Figure 31.** Software optimization of a 5-tap filter.

### *6.1.1. Acceleration of Filters with NIOS II C2H Compiler*

The Wavelet Transform filters were implemented in hardware using the Altera NIOS II C2H Compiler [33]. This tool generates custom hardware accelerators directly from their software description in language C using the available resources of FPGA such as logic elements and embedded multipliers. The addition of accelerators to the system reduces the workload of the processor and increases the application performance. When the function is ready to be accelerated, SoPC Builder generates a new system, including the necessary bus connections to communicate the accelerator with the processor and other components of the system. In the Bandelet codec, the software functions of the 5-tap and 3-tap filters were accelerated (Figure 32).

These blocks were selected for acceleration since they are the most used functions in the codec: In the compression process, they are used in the 2D Wavelet Transform and in the 1D Wavelet Transform of the Extraction of Points block; in the decompression process, they are used in the 1D Inverse Wavelet Transform that reconstructs the 2D Wavelet coefficients from the Bandelet coefficients, and in the 2D Inverse Wavelet Transform.

To obtain the maximum algorithm acceleration, the software routines has to be rewritten to meet the software-hardware mapping requirements of the NIOS II C2H Compiler. This includes avoiding data dependencies, data cache coherency problems, and excessive pointer dereferences. In addition, the functions cannot have float variables. To meet this requirement, the variables associated to the functions that implemented the filters Le Gall 5/3 were converted to integer, even though their coefficients are defined as real (float in C). The acceleration results obtained with the NIOS II C2H compiler are discussed in Section 7.2.

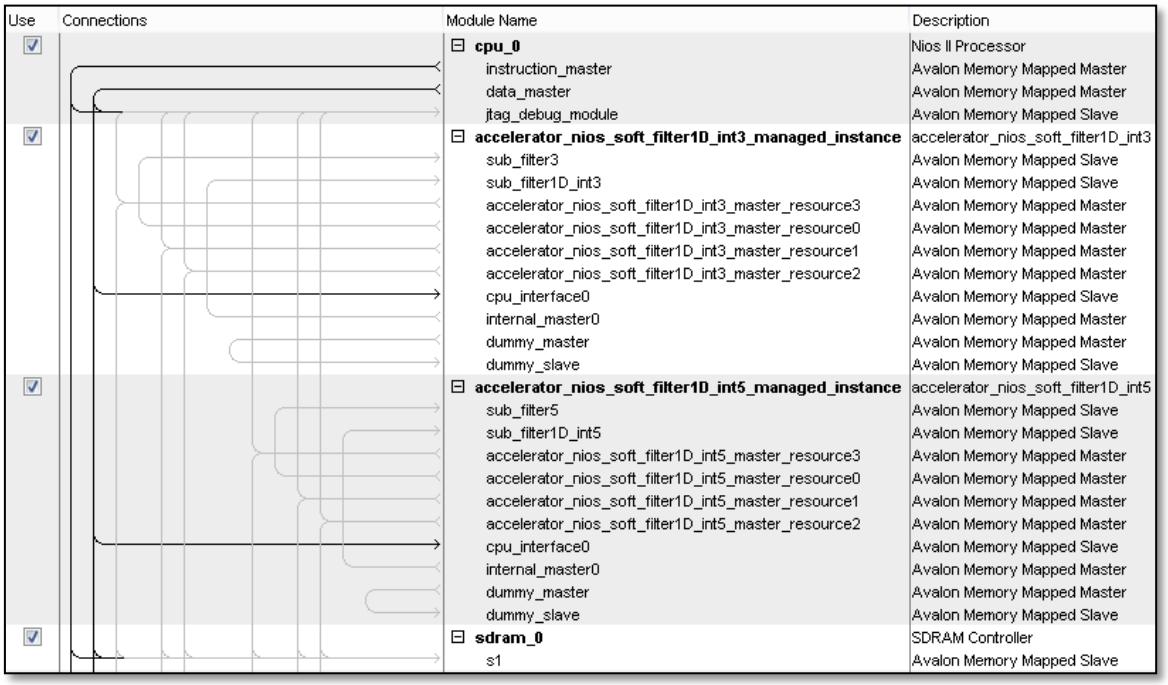



**Figure 32.** System generated by SoPC Builder with accelerators for the 5-tap and 3–tap filters.

### 6.1.2. Processing of Signal Borders in the Wavelet Transform

The filter presented in Figure 31 is implemented with a convolution operation defined by (8), where the results of multiplying the elements of the input sequence *x(n)* by the filter coefficients *h(n)* are successively accumulated:

$$y(n) = \sum h(k)x(n-k) \tag{8}$$

The immediate problem presented in this calculation is that *x(n)* is a real and finite signal and the calculation of the convolution may need values such as *x(-1)* or *x(-2)* that are not defined for this signal. If this signal is not properly treated at the borders, the Wavelet Transform will have problems at the ends, preventing an accurate reconstruction of the original signal. Figure 33 presents the case where a signal is processed with filters Le Gall 5/3 without taking into account the signal borders. The SNR of the reconstructed signal is very low due to the difference in the values at the ends between the original and the reconstructed signal. For the Bandelet codec, the processing of finite-signal borders was studied for the implementation of the 2D and 1D Wavelet Transform to obtain an exact reconstruction of the signal. Figure 34 presents the main methods for the extension of a finite signal [34]: Extension by zeros (zero-padding), extension by periodicity (wraparound), and extension by symmetry (symmetric extension). The selection of the extension method depends on the filter used, which also influences the way the down-sampling of the filter outputs is performed during the implementation of Wavelet Transform.

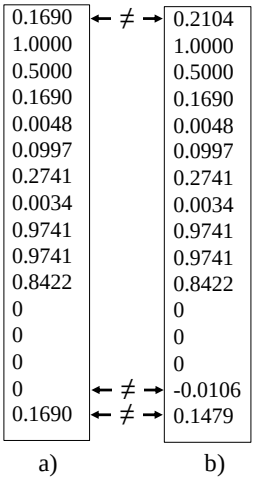
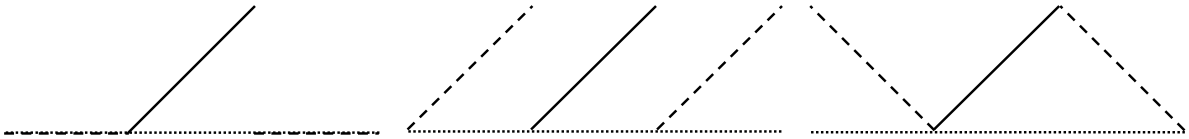



**Figure 33.** Sequence of 16 items: Original (a) and reconstructed (b) after processing with filters Le Gall 5/3 without taking into account the problems at the borders (SNR = 38.48 dB).



a) Extension by zeros. b) Extension by periodicity. c) Extension by symmetry.

**Figure 34.** Extension methods for finite signals: Extension by zeros and extension by periodicity introduced jumps at the borders of the sequence, while the extension by symmetry is continuous.

When the filter length is odd, it was found that it is best to use a symmetric extension to address the problems at the borders. The input signal is extended at both ends, without repeating the first or last samples, for an amount equal to the filter length minus one. For the filter Le Gall 5/3, the low-pass decomposition filter has a length of five coefficients; hence it must have a four-element extension at both ends of the input signal (Figure 35.a). The high-pass decomposition filter has a length of three coefficients, so the extension must be of two elements (Figure 35.b). This way, the filter input signal after extension has a length equal to:

$$L_{filter\ input} = L_{original\ input} + 2(L_{odd\ filter} - 1) \quad (9)$$

The filter output signal has the same length of the input signal, but the first element is located in the position:

$$Pos_{1st\ element} = (L_{odd\ filter} - 1) + \frac{(L_{odd\ filter} - 1)}{2} \quad (10)$$

Where it has been assumed that the first position of the sequences is the position 0. Figure 36 shows the outputs of the filters Le Gall 5/3 for the input signals of Figure 35; the first element of each sequence is highlighted. When the first element has been located, a down-sampling of the signal by two is performed. When the filters are odd, it was found that the down-sampling must be performed in an opposite way for the approximation and detail coefficients. For the approximation coefficients, only the odd-numbered elements are kept (odd down-sampling), while for the detail coefficients, the even-numbered elements are maintained (even down-sampling). Figure 37 presents the down-sampling performed in the signals of Figure 36. This work has denominated even down-sampling the process that keeps the even elements of a sequence, and even up-sampling the process in which zeros are inserted in the even positions of a sequence.

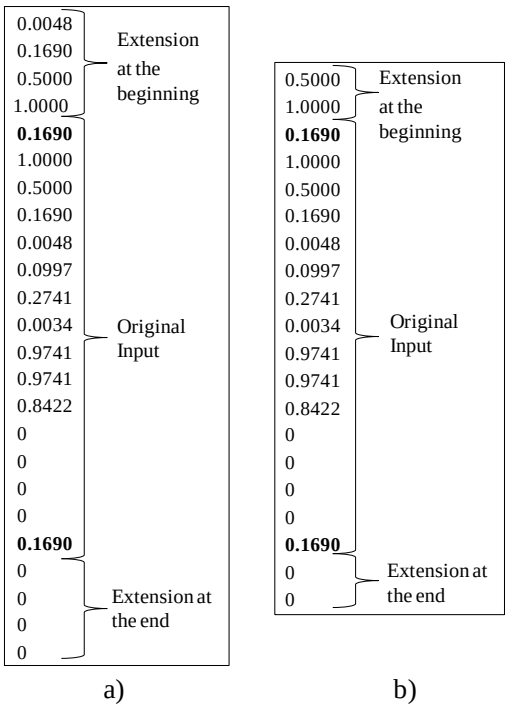



**Figure 35.** Input signal of 16 elements with symmetric extension to be processed with filters Le Gall 5/3: a) low-pass filter of 5 elements, b) high-pass filter of 3 elements.

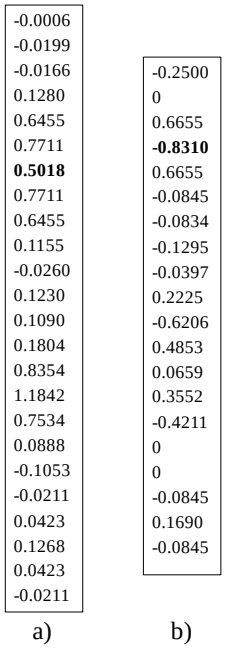



**Figure 36.** Outputs of the filters Le Gall 5/3 to the inputs shown in Figure 35: a) low-pass filter, b) high-pass filter.

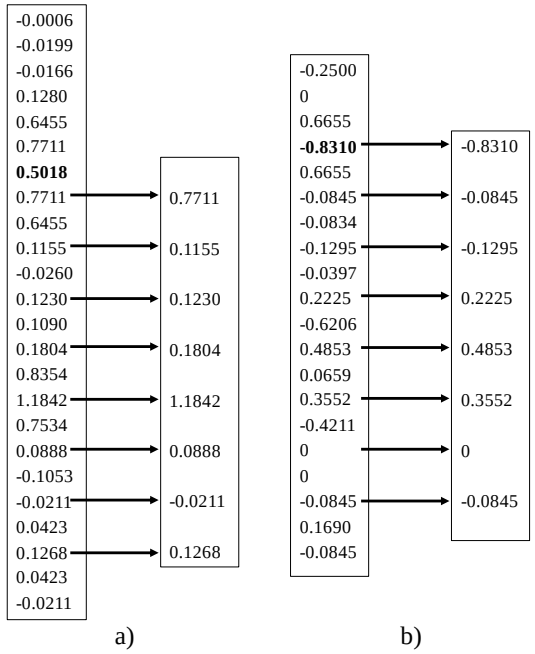



**Figure 37.** Down-sampling of the outputs of the filters Le Gall 5/3: a) low-pass filter, b) high-pass filter.

The number of elements at the output of both low-pass filter (approximation) and high-pass filter (details) remaining after down-sampling is:

$$L_{aprox\ and\ details} = \frac{L_{original\ input}}{2} \tag{11}$$

Therefore, the length of the transform output is equal to the length of the original input:

$$L_{transf\ output} = L_{aprox} + L_{details} = L_{original\ input} \tag{12}$$

Thus obtaining a non-expansive transform, something that does not occur for example in Matlab, where the function *dwt()* produces an expansive transform. This means that after every level of decomposition, the number of memory locations used to store coefficients increases.

For the Inverse Transform, the process is executed in reverse order. First, an up-sampling is performed in the inputs of the synthesis filters. The approximation coefficients are even up-sampled since the odd-numbered elements of the sequence were kept during the analysis; the detail coefficients are odd up-sampled. This up-sampling is presented in Figure 38 for the signals of Figure 37. After up-sampling, the reconstruction filters process the coefficients, following the same rules used during decomposition to extend the input signal and to locate the first element of the output. The length of the outputs of the reconstruction filters is the same as the original input from the position where the first element is located. Then, the outputs of the reconstruction filters are added to produce the reconstructed input signal of the transform for the corresponding decomposition level. By following the rules outlined here in both the analysis and the synthesis process, the reconstructed signal will be exact to the original one, which can be checked by calculating the SNR between the original and the reconstructed signal. If down-sampling or up-sampling of the signal is done conversely, keeping the even-numbered elements of the approximation and the odd-numbered elements of the details, an exact reconstruction of the signal also obtained. If, however, down-sampling and up-sampling of the coefficients is not performed in opposite way, the signal reconstruction is not exact and has a very low SNR. This occurs if, for example, the outputs of both decomposition filters are odd sub-sampled, leaving only the odd elements of the

sequence in the approximation and detail coefficients. Figure 39 summarizes the three cases described indicating the corresponding SNR of the reconstructed signal in relation to the original signal.

| a) | b) |
|---|---|
| 0 | -0.8310 |
| 0.7711 | 0 |
| 0 | -0.0845 |
| 0.1155 | 0 |
| 0 | -0.1295 |
| 0.1230 | 0 |
| 0 | 0.2225 |
| 0.1804 | 0 |
| 0 | 0.4853 |
| 1.1842 | 0 |
| 0 | 0.3552 |
| 0.0888 | 0 |
| 0 | 0 |
| -0.0211 | 0 |
| 0 | -0.0845 |
| 0.1268 | 0 |

**Figure 38.** Up-sampling of the approximation (a) and detail (b) coefficients for the reconstruction filters Le Gall 5/3.

| a) | b) | c) | d) |
|---|---|---|---|
| 0.1690 | 0.1690 | 0.1690 | 0.4384 |
| 1.0000 | 1.0000 | 1.0000 | 1.1975 |
| 0.5000 | 0.5000 | 0.5000 | 0.2978 |
| 0.1690 | 0.1690 | 0.1690 | -0.0253 |
| 0.0048 | 0.0048 | 0.0048 | 0.1500 |
| 0.0997 | 0.0997 | 0.0997 | 0.1812 |
| 0.2741 | 0.2741 | 0.2741 | 0.3168 |
| 0.0034 | 0.0034 | 0.0034 | -0.2883 |
| 0.9741 | 0.9741 | 0.9741 | 0.8210 |
| 0.9741 | 0.9741 | 0.9741 | 1.3639 |
| 0.8422 | 0.8422 | 0.8422 | 0.7253 |
| 0 | 0 | 0 | -0.2353 |
| 0 | 0 | 0 | 0.1391 |
| 0 | 0 | 0 | 0.0104 |
| 0 | 0 | 0 | 0.0106 |
| 0.1690 | 0.1690 | 0.1690 | 0.2535 |
| | **SNR=375.64dB** | **SNR=376.16dB** | **SNR=62.57dB** |

**Figure 39.** Original (a) and reconstructed signal using filters Le Gall 5/3 for various down-sampling and up-sampling: b) Odd in approximation coefficients and even in detail coefficients; c) Even in approximation coefficients and odd in detail coefficients; d) Odd in approximation and detail coefficients.

The problems in the signal borders are also evident when a two-dimensional signal is processed, especially when the image size is not significantly bigger than the filter length. Figure 40 shows a 64x64 pixel section of Lena processed with filters Le Gall 5/3. When an extension by zeros is used for its rows and columns, a darkening of the edges in the reconstructed image is produced, reducing the SNR calculated for the image (SNR = 34.9dB). These problems are noticeable in the Wavelet Transform of the image. When using the extension by zeros, the horizontal detail coefficients in Figure 41.b (lower left quadrant) present a distortion at the top of the quadrant that obviously do not follow the pattern of the image. The same is true when observing the left edge of the vertical detail coefficients (upper right quadrant).


**Figure 40**. Sections of Lena: Original (a) and processed with symmetric extension (b) and extension by zeros (c).

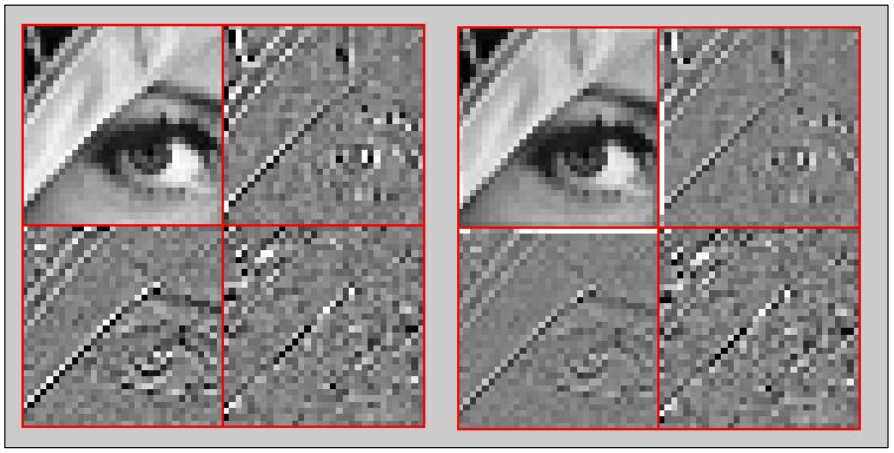



**Figure 41**. 2D Wavelet Transform of a section of Lena with symmetric extension (a) and extension by zeros (b).

## 6.2. Image Segmentation

The two-dimensional array of 2D Wavelet coefficients must be segmented using an optimal dyadic configuration, which is generated from the evaluation of the Lagrangian of the squares and sub-squares of the segmentation. To simplify the design and operation of the encoder, the segmentation is performed with fixed-size squares without the subsequent bottom-up comparison of the Lagrangian. This size is denominated inside the algorithm as *L* or *w* and can be modified by the user at the beginning of the encoder execution.

## 6.3. Extraction of Points

Each square of $w^2$ pixels is processed by the Extraction of Points block, which determines the best geometric direction in each square and returns the Bandelet coefficients related with this direction. This block is composed of three sub-blocks: Projection of Points, 1D Wavelet Transform, and Lagrangian calculation.

The Projection of Points process is responsible of projecting orthogonally the square pixels and ordering them in the one-dimensional array $f_d$ for each possible direction *d*. The maximum number of possible directions for a square of side *w* is $2w^2$, i.e. there are up to $2w^2$ one-dimensional functions that must be analyzed to find the direction *d* that is closest to the local geometry of the image. Depending on the value of *w* entered by the user, the encoder generates a matrix of indices *ind* where each column contains the ordering of the square pixels associated with one of the possible directions *d*. These orderings are obtained by the rotation of the vertical midline of the square and the subsequent orthogonal projection of the center of each pixel to this line, following the method proposed by Peyré and Mallat in [5] and [11]. In the case of a segmentation with $w = 4$, the total number of pixels to project is $w^2=16$. The projections of the points for three rotations of the square midline are shown in Figure 42; in the encoder, the clockwise rotations has been assumed to be positive, while they are negative in the opposite direction. The resulting orderings for the rotations of Figure 42 are:

- *Rotation of Figure 42.a:* $f_d=[x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}]$
- *Rotation of Figure 42.b:* $f_d=[x_1, x_2, x_3, x_5, x_4, x_6, x_7, x_9, x_8, x_{10}, x_{11}, x_{13}, x_{12}, x_{14}, x_{15}, x_{16}]$
- *Rotation of Figure 42.c:* $f_d=[x_1, x_2, x_5, x_3, x_6, x_4, x_9, x_7, x_{10}, x_8, x_{13}, x_{11}, x_{14}, x_{12}, x_{15}, x_{16}]$

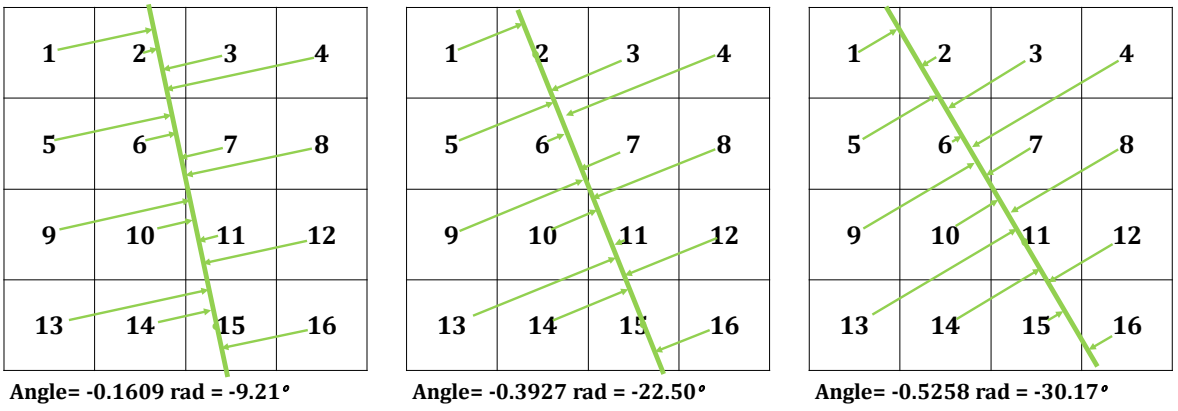



**Figure 42**. Projection of points in a 4x4 square for three rotation angles.

For this square, the total number of orderings is 16. They are stored in the matrix *ind* shown in Figure 43. When *w* is equal to 8, 16 or 32, the total number of orderings is 72, 288, and 1232 respectively.

Each of the signals $f_d$ generated with the orderings given by the array *ind* is processed with the 1D Wavelet Transform, obtaining the output signal $f_w$ with the resulting coefficients. The implementation of the 1D Wavelet Transform uses the same functions of the 2D Wavelet Transform for the filters Le Gall 5/3. The signal borders are also processed with the considerations mentioned earlier. At the exit of 1D Wavelet Transform, the Lagrangian is calculated for each signal $f_w$ following (4), (5), and (6). Then, the Lagrangian of all possible directions for the current square are stored in a vector; the vector position with the lowest value indicates the best direction *d* and the thresholded coefficients by threshold *T* of the associated signal $f_w$ are stored in an array memory that contains all the Bandelet coefficients of the image.

## 6.4. Bandelet Decoder

To perform the image decoding, it is necessary to have the set of coefficients generated by the Bandelet encoder alongside the number of the ordering associated with the best direction *d* found for each square during encoding. These data are stored in two separate memory arrays at the exit of the encoder. The decoder is faster than the encoder as it does not need to calculate the Lagrangian for each of the possible directions of each square.

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 16 |
| 5 | 5 | 5 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 8 | 8 | 8 | 8 | 12 |
| 9 | 2 | 2 | 5 | 5 | 3 | 3 | 2 | 2 | 8 | 8 | 3 | 3 | 12 | 12 | 8 |
| 2 | 9 | 9 | 3 | 3 | 5 | 4 | 1 | 8 | 2 | 2 | 12 | 12 | 3 | 16 | 4 |
| 13 | 6 | 6 | 6 | 6 | 4 | 5 | 8 | 1 | 7 | 7 | 7 | 7 | 16 | 3 | 15 |
| 6 | 13 | 3 | 9 | 4 | 6 | 6 | 7 | 7 | 1 | 12 | 2 | 16 | 7 | 7 | 11 |
| 10 | 3 | 13 | 4 | 9 | 7 | 7 | 6 | 6 | 12 | 1 | 16 | 2 | 11 | 11 | 7 |
| 3 | 10 | 10 | 7 | 7 | 9 | 8 | 5 | 12 | 6 | 6 | 11 | 11 | 2 | 15 | 3 |
| 14 | 7 | 7 | 10 | 10 | 8 | 9 | 12 | 5 | 11 | 11 | 6 | 6 | 15 | 2 | 14 |
| 7 | 14 | 4 | 13 | 8 | 10 | 10 | 11 | 11 | 5 | 16 | 1 | 15 | 6 | 6 | 10 |
| 11 | 4 | 14 | 8 | 13 | 11 | 11 | 10 | 10 | 16 | 5 | 15 | 1 | 10 | 10 | 6 |
| 4 | 11 | 11 | 11 | 11 | 13 | 12 | 9 | 16 | 10 | 10 | 10 | 10 | 1 | 14 | 2 |
| 15 | 8 | 8 | 14 | 14 | 12 | 13 | 16 | 9 | 15 | 15 | 5 | 5 | 14 | 1 | 13 |
| 8 | 15 | 15 | 12 | 12 | 14 | 14 | 15 | 15 | 9 | 9 | 14 | 14 | 5 | 5 | 9 |
| 12 | 12 | 12 | 15 | 15 | 15 | 15 | 14 | 14 | 14 | 14 | 9 | 9 | 9 | 9 | 5 |
| 16 | 16 | 16 | 16 | 16 | 16 | 16 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 1 |
| -1,178 | -1,044 | -0,884 | -0,686 | -0,525 | -0,392 | -0,160 | 0,160 | 0,392 | 0,525 | 0,686 | 0,884 | 1,044 | 1,178 | 1,409 | 1,731 |

**Figure 43**. Orderings for a square of 4x4 pixels; the last row shows the rotation angle (in radians) of the vertical midline of the square on which the orthogonal projections of the points are made.

In the decoding process, the set of Bandelet coefficients of each square passes through the 1D Inverse Wavelet Transform, obtaining the reconstruction of the square pixels arranged in a one-dimensional manner. The pixels are subsequently reordered in the two-dimensional space of the square according to the best direction *d* found in the encoding process. The arrangement of the pixels of all squares of the image produces the reconstructed 2D Wavelet coefficients that are processed with the 2D Inverse Wavelet Transform, generating the decompressed image. The 1D and 2D Inverse Wavelet Transform uses the synthesis filters Le Gall 5/3 and takes also into account the problems of finite signals at the edges.

## 6.5. System Operation

The Bandelet image codec was implemented on the Image Editor with JPEG/JPEG2000 Compressor. Figures 44 and 45 show two pictures of the system running. The system reads the BMP files of grayscale images that are stored in the SD-Card memory and displays them on the Touch Panel so the user can select the image he wants to process. The user can also enter the value for the threshold *T* used in the Approximation Algorithm. The system compresses the selected image with the Bandelet encoder, decompresses it later with the decoder, and returns the SNR calculated for the reconstructed image alongside the number of Bandelet coefficients produced by the encoder. The system also compresses and decompresses the image with a Wavelet codec in order to evaluate the performance of the Bandelet codec. At the end of the whole process, the Touch Panel displays the images associated with both codecs:

- Original Image and 2D Wavelet coefficients.
- *Bandelet codec:* Bandelet coefficients, 2D Wavelet coefficients reconstructed from Bandelet coefficients, reconstructed image with the Bandelet decoder.
- *Wavelet codec:* Thresholded 2D Wavelet coefficients, reconstructed image with Wavelet decoder.

**Figure 44**. Thumbnails of the images available in the SD-Card for Bandelet processing.

Any of these images can be zoomed in on the Touch Panel and sent to the host computer connected to the DE2-70 board through UART component.

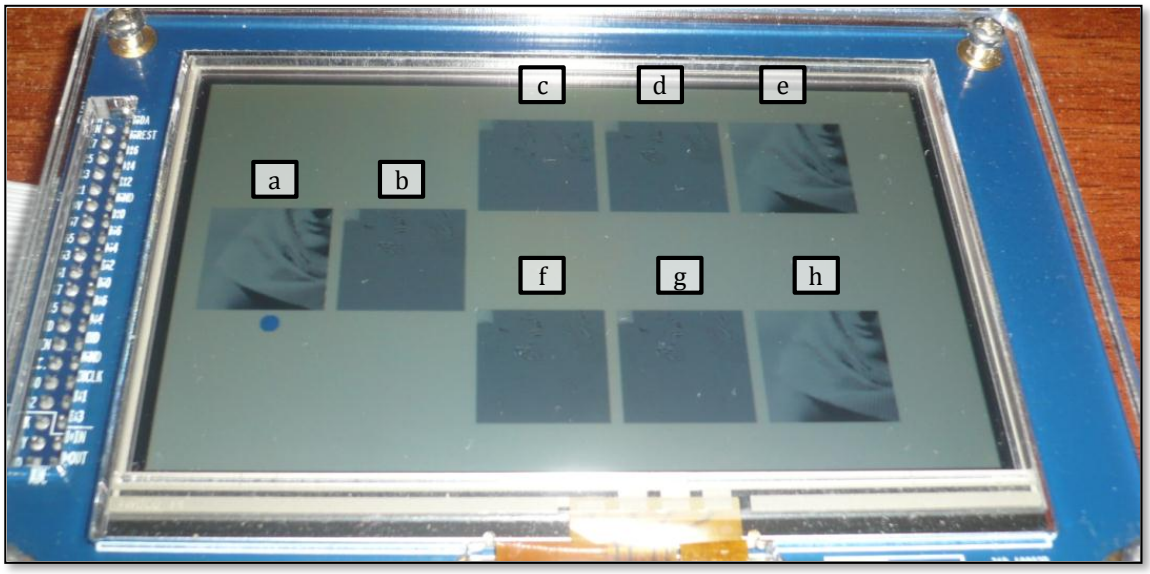



**Figure 45**. Display of images at the end of system execution: a) Original Image; b) 2D Wavelet coefficients; c) Bandelet coefficients; d) 2D Wavelet coefficients reconstructed from Bandelet coefficients; e) Reconstructed image with the Bandelet decoder; f) and g) Thresholded 2D Wavelet coefficients; h) Reconstructed image with Wavelet decoder.

# *Chapter 7*

# *Performance Evaluation of the Codec*

The performance of the image codec based on the Bandelet Transform was evaluated for various compression ratios using a base set of grayscale images. Images were selected with various geometric components so that it was possible to observe the operation of the codec in different types of images. Figure 46 presents the test images used for the evaluation, indicating their original resolutions.

Each image was processed with the Bandelet codec using a threshold $T$ as input parameter. The decoded image with this codec was then compared with the image obtained from a Wavelet codec. In the latter case, the original image was passed through the 2D Wavelet Transform and the resulting coefficients were discriminated according to the user threshold, but without applying the Bandelet processing to them. As quality metric the SNR was used. The number of scales in the Wavelet Transform was three, having a fixed segmentation of squares with $w$ = 8. This size was selected after performing several tests with images as it allows having a compromise between performance and algorithm execution time: With a square of 4x4, it is not possible to clearly identify the local geometry of the image, and with a 16x16 square, the algorithm takes too long to determine the best geometric direction $d$, because the maximum number of possible directions is quadratically proportional to the size of the square.

## 7.1. Analysis of Image Compression

The first image analyzed was Barb.bmp. To begin, a section of 128x128 pixels was used, which included part of Barb's face and part of the garment that wraps her head (Figure 47). It was considered important to include this piece of clothing in the analysis because it contains a pattern of black and white lines with high contrast and a clear geometric orientation.

a) Barb.bmp: 512x512 pixels.

b) Lena.bmp: 512x512 pixels.

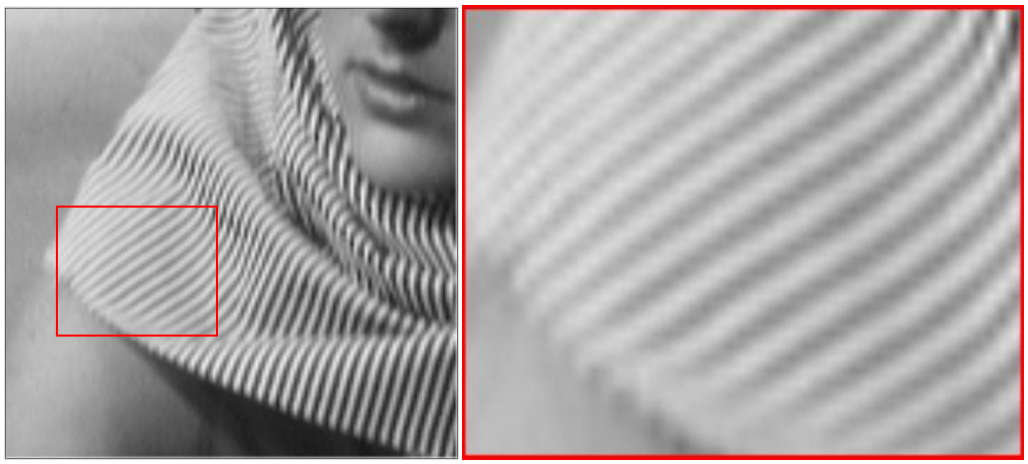
c) fingerprint.bmp: 256x256 pixels.

**Figure 46**. Test images for the performance evaluation of the Bandelet codec.

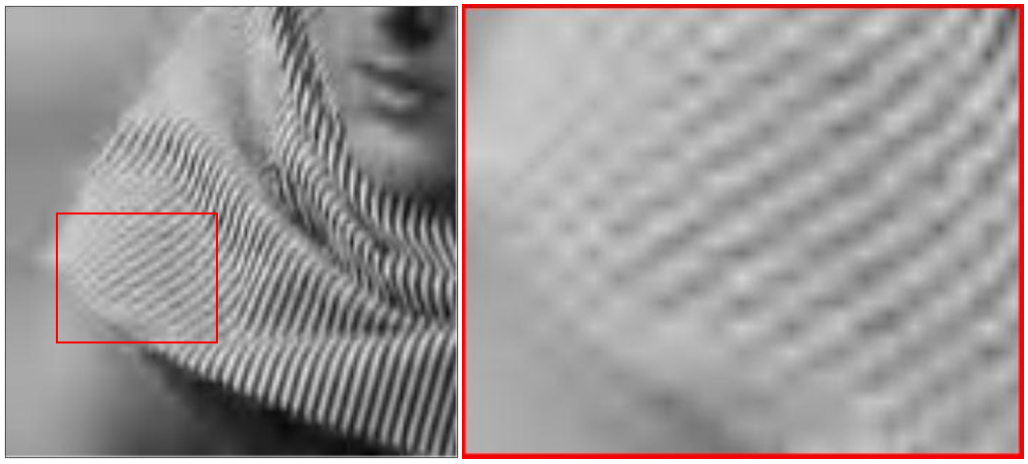
Figure 47.a presents the original image, enclosing the section the analysis will focus on. Figures 47.b and 47.c present the reconstructed images from the Bandelet and Wavelet processing respectively. These images have a compression rate of 0.96 bpp (bits per pixel). When the reconstructed images are compared, it can be seen that the image obtained from the Bandelet codec retains more elements of the original pattern than the image obtained from the Wavelet codec. With Bandelet Transform, the lines of the garment still retain their geometry, and their orientation and contrast are distinguishable. With the Wavelet Transform however, the original pattern of lines has started to fade.

Figure 48 presents the Bandelet and Wavelet coefficients for images of Figure 47. Figure 48.a contains the 2D Wavelet coefficients for the original image at three scales in the Horizontal-Vertical-Diagonal orientations. These coefficients are then processed according to the Bandelet algorithm using the threshold *T* to increase data compression.

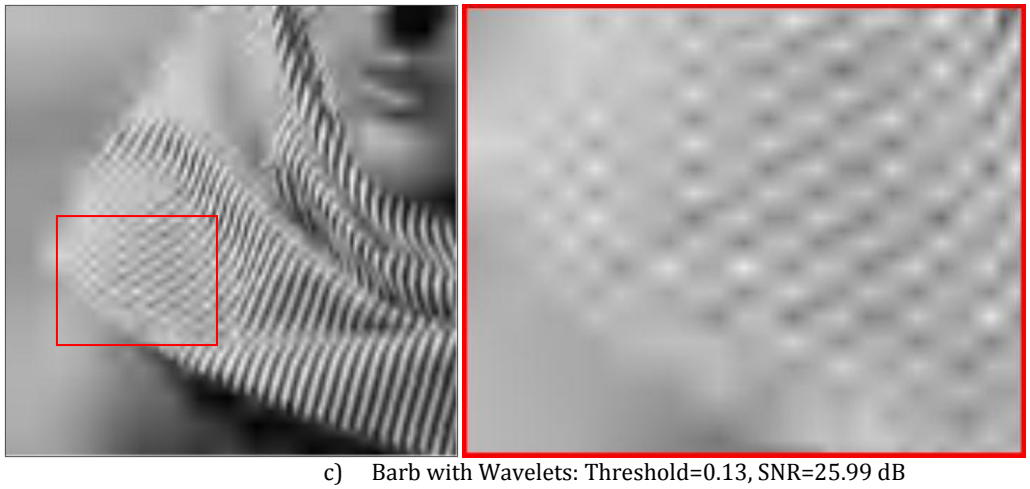
The result is 1974 Bandelet coefficients in the encoder output and presented in Figure 48.b. To decompress the image, the Bandelet inverse algorithm is applied to reconstruct the 2D Wavelet coefficients (Figure 48.d), which are processed with the 2D Inverse Wavelet Transform. Despite Bandelet coefficients lose some patterns and orientations of the original image, the 2D Wavelet coefficients reconstructed from them are very similar to the original ones, even more than the thresholded 2D Wavelet coefficients presented in Figure 48.c. This means that even though the number of Bandelet and thresholded 2D Wavelet coefficients is almost the same (1974 and 1975 respectively), the former are able to store more information than the latter.



a) Original Section of Barb: 128x128 pixels = 16384 bytes.

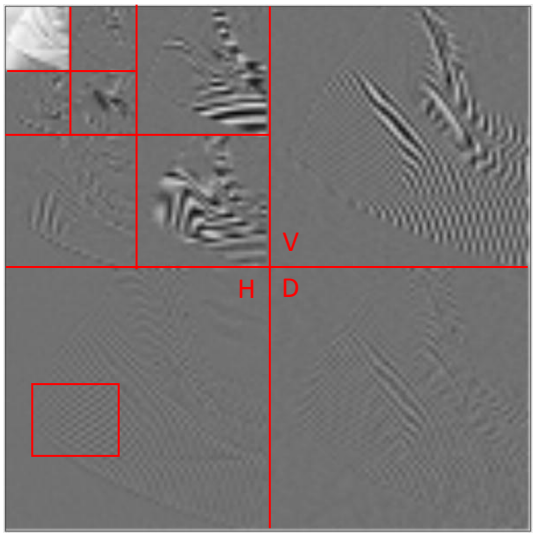


b) Barb with Bandelets: Threshold=0.1, SNR=27.76 dB

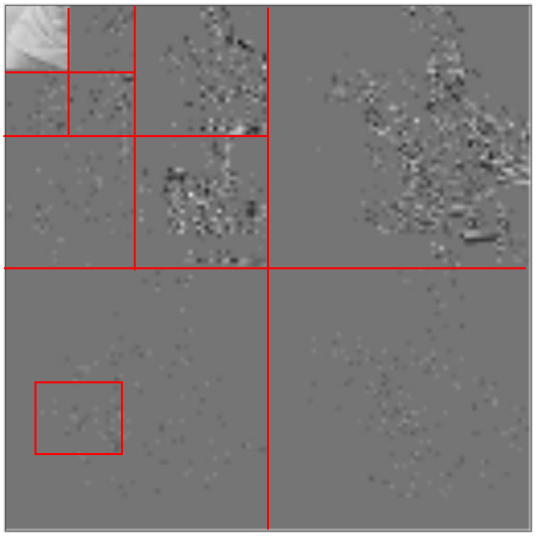


c) Barb with Wavelets: Threshold=0.13, SNR=25.99 dB

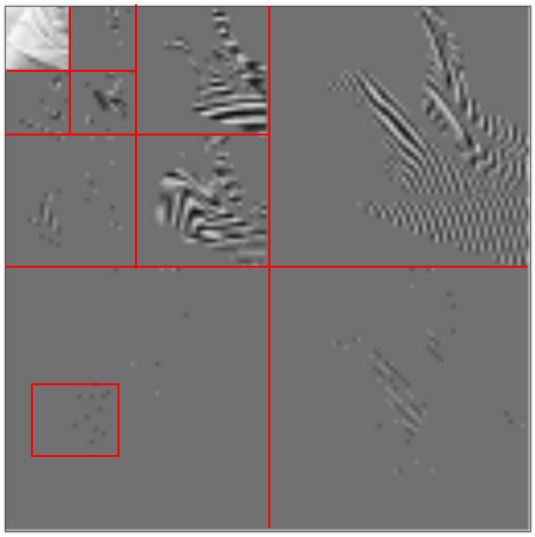
**Figure 47**. Compression of a section of Barb at 0.96 bpp.



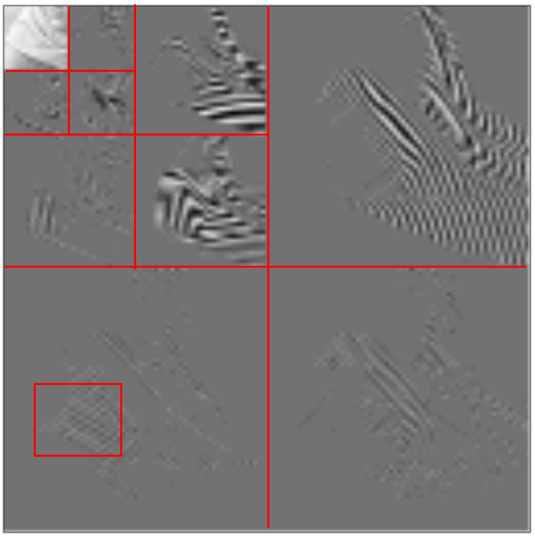

a) Original 2D Wavelet Coefficients.



b) Bandelet Coefficients: 1974.



c) Thresholded 2D-Wavelet Coefficients: 1975.



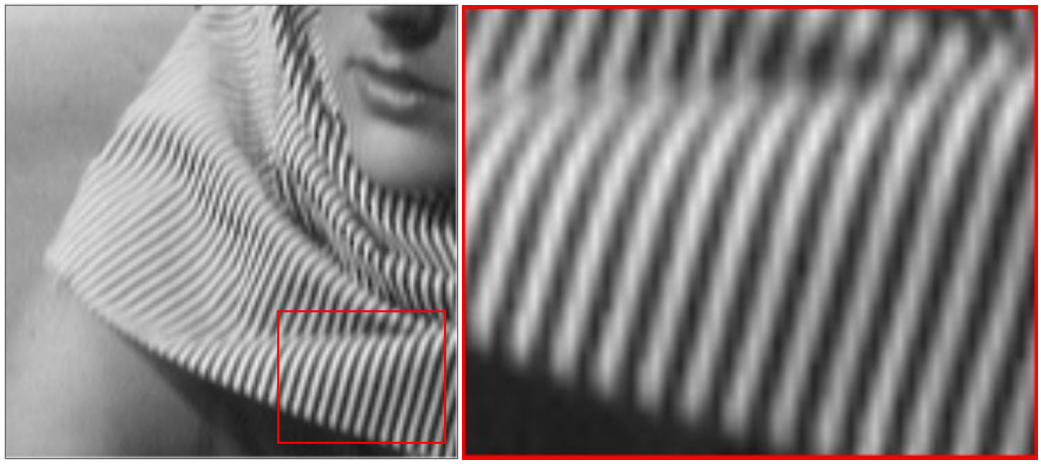
d) 2D Wavelet Coefficients reconstructed from Bandelet coefficients.

**Figure 48**. Bandelet and Wavelet coefficients for a section of Barb at 0.96 bpp.

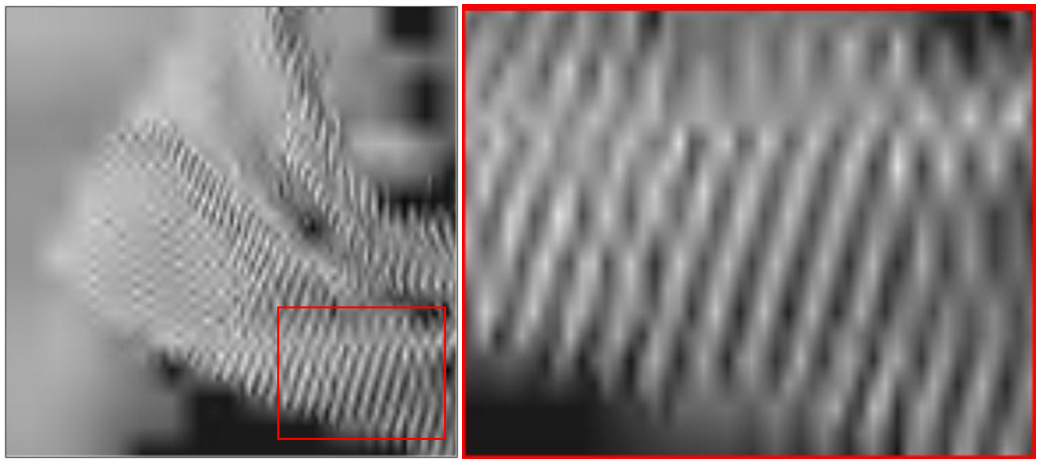
For the same section of Barb, a stronger compression is applied at a rate of 0.26 bpp (Figure 49). In the reconstructed images from both the Bandelet and Wavelet codec, Barb's face is blurred, but in the reconstructed image from the 2D Wavelet coefficients her mouth can be better appreciated than in the Bandelet case. As for the garment of Barb, it is noticeable that the line pattern is still distinguishable in the image from the Bandelet decoder, whereas in the Wavelet case this pattern has almost completely vanished and a pattern of diamonds has become dominant, making no longer possible to determine the original direction of the garment lines.

Figure 50 shows the Bandelet and the 2D Wavelet coefficients for images of Figure 49. It is clear that the reconstructed 2D Wavelets coefficients of Figure 50.d have more details and therefore more information than the thresholded 2D Wavelet coefficients of Figure 50.c. This enables the Bandelet codec to produce a decompressed image more similar to the original one. This difference is translated into a higher SNR for the image decoded by Bandelets than the SNR obtained with Wavelets, as presented in Figure 49.

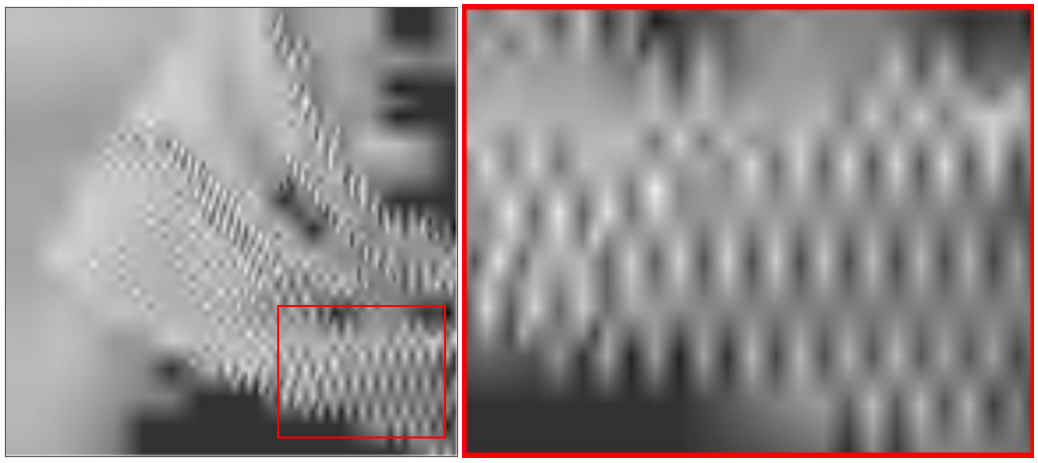
The analysis of the section of Barb in Figures 47 and 49 for various compression rates shows that at all times Bandelet processing produces a higher SNR than that obtained with 2D Wavelet processing, as shown in Figure 51, where the maximum difference is up to 2 dB for the same compression ratio. The superiority of Bandelet processing over 2D Wavelet processing is largely due to the presence of significant geometric components in the analyzed image (such as the garment that covers Barb's head). This allows that Bandelet Transform capabilities are utilized to the maximum since this transform eliminates the geometric redundancy of the 2D Wavelet coefficients, which is presented in these areas specially.



a) Original section of Barb: 128x128 pixels = 16384 bytes.

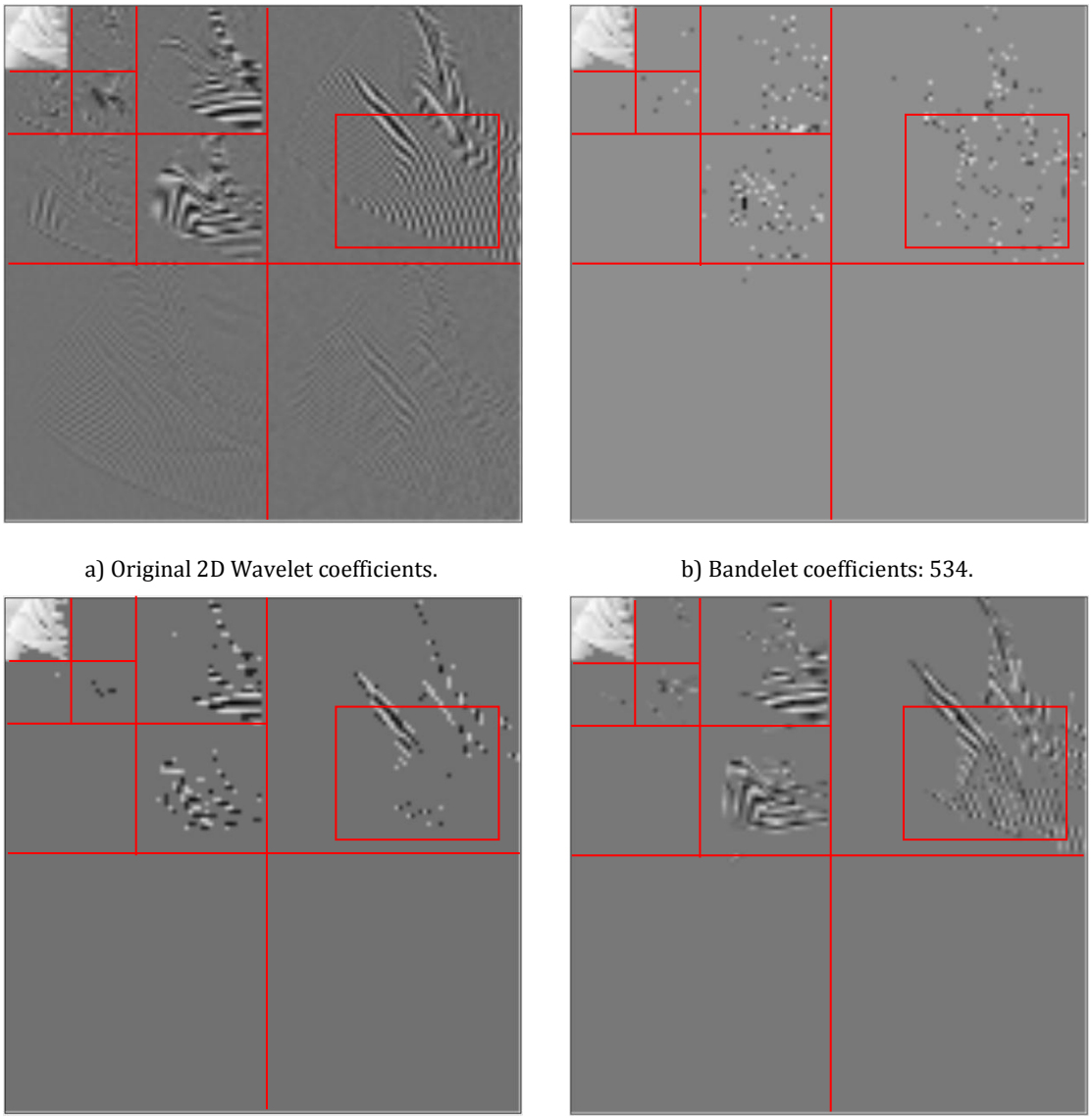


b) Barb with Bandelets: Threshold =0.3, SNR=20.18 dB.



c) Barb with Wavelets: Threshold=0.37, SNR=18.03 dB.

**Figure 49**. Compression of a section of Barb at 0.26 bpp.



a) Original 2D Wavelet coefficients.

b) Bandelet coefficients: 534.

c) Thresholded 2D Wavelet coefficients: 536.

d) 2D Wavelet Coefficients reconstructed from Bandelet coefficients.

**Figure 50**. Bandelet and Wavelet coefficients for a section of Barb at a 0.26 bpp.

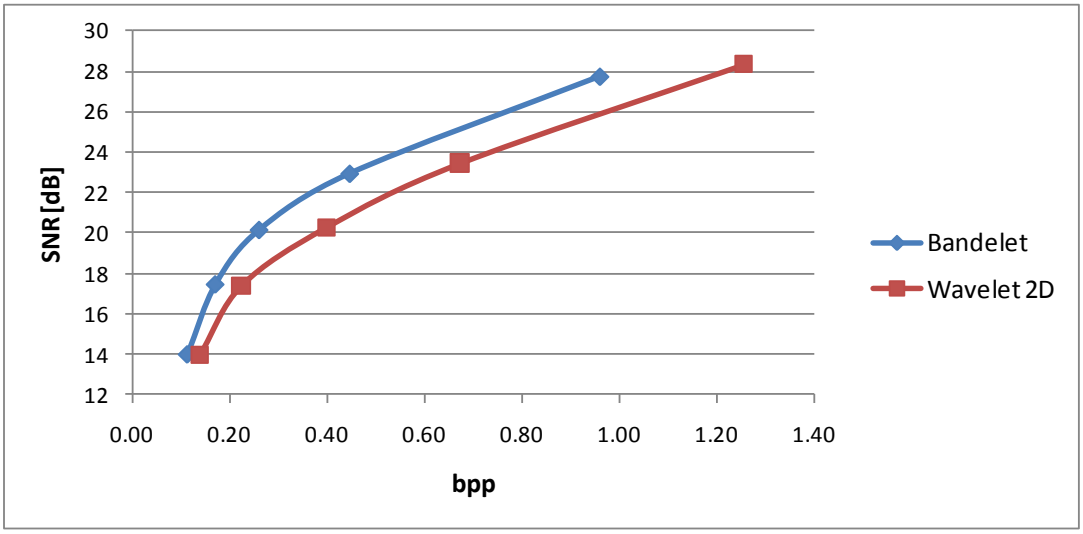
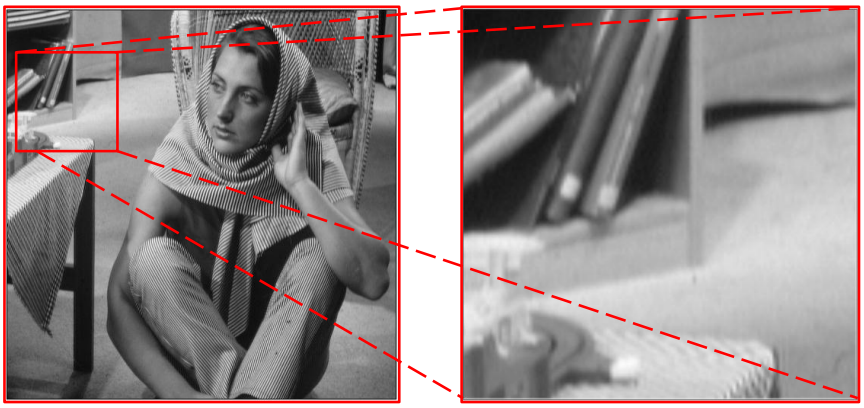



**Figure 51**. SNR vs. bpp for the section of Barb analyzed in Figures 47 and 49.

If the analyzed image has no geometric components with high contrast, Bandelet processing offers no advantages over 2D Wavelet processing. To analyze this, a second section of Barb.bmp was selected (Figure 52.a). Clearly this image has geometric elements like the straight lines from the books and the library, but these elements do not have a high contrast to differentiate them from the objects around so their geometry is not completely distinguishable by the Bandelet Transform (the library for example has almost the same color as the floor).

In front of the library there is a table with a tablecloth that has a pattern of lines similar to the garment in Barb's head. However, this pattern is not well compressed by the Bandelet encoder since those lines do not have good contrast. Because of this, the decoded images for a compression rate of 0.11 bpp with the Bandelet Transform (Figure 52.b) and the 2D Wavelet Transform (Figure 52.c) are very similar. For other compression rates, the same trend remains as shown in Figure 53.



a) Original section of Barb: 128x128 pixels = 16384 bytes.

b) Barb with Bandelets:
Threshold=0.3, SNR=21.18 dB.

c) Barb with Wavelets:
Threshold=0.37, SNR=21.28 dB.

**Figure 52.** Compression of a section of Barb at 0.11 bpp.

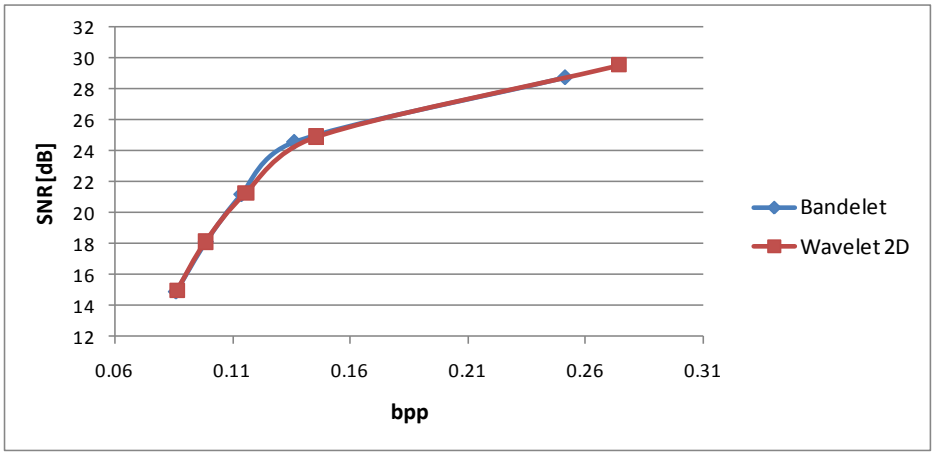
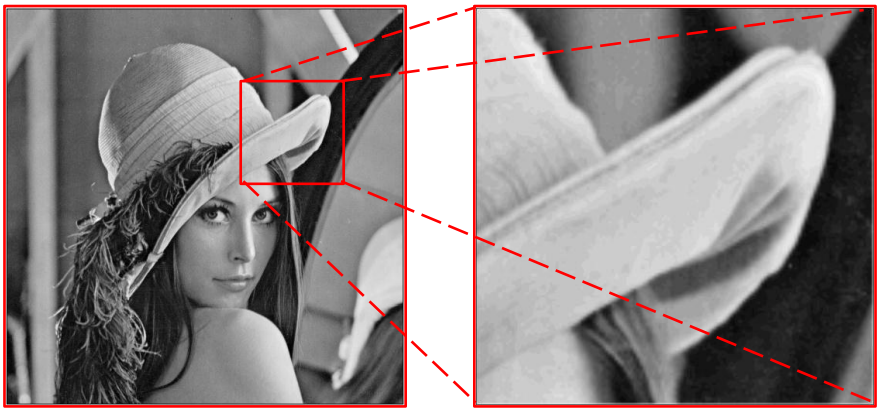



**Figure 53**. SNR vs. bpp for the section of Barb analyzed in Figure 52.

Similar compression results were presented when Lena.bmp was compressed with the Bandelet and the 2D Wavelet Transforms. Figure 54 shows a section of this image compressed at a rate of 0.11 bpp with an approx. SNR of 21 dB for the images decoded with both transforms. Bandelet and Wavelet coefficients for this image are shown in Figure 55. As can be seen, the 2D Wavelet coefficients that were thresholded (Figure 55.c) and those that were reconstructed from the Bandelet coefficients (Figure 55.d) contain much the same information; hence the images from both decoders have similar SNR. These results are repeated for other compression rates according to the graph in Figure 56.

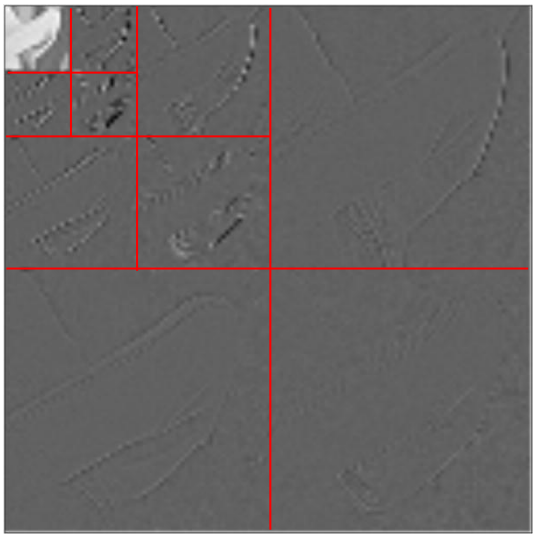


a) Original section of Lena: 128x128 pixels = 16384 bytes.

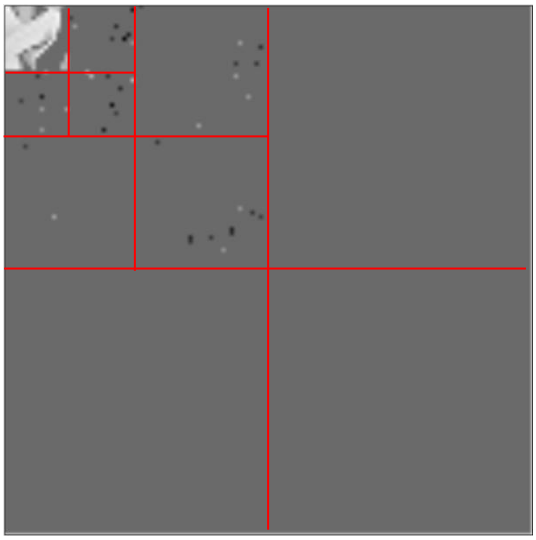
b) Lena with Bandelets: Threshold=0.3, SNR=21.40 dB.

c) Lena with Wavelets: Threshold=0.3, SNR=21.54 dB.

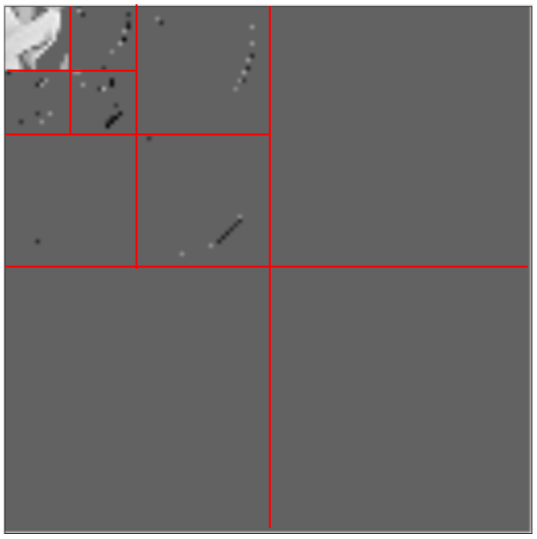
**Figure 54**. Compression of a section of Lena at 0.11 bpp.



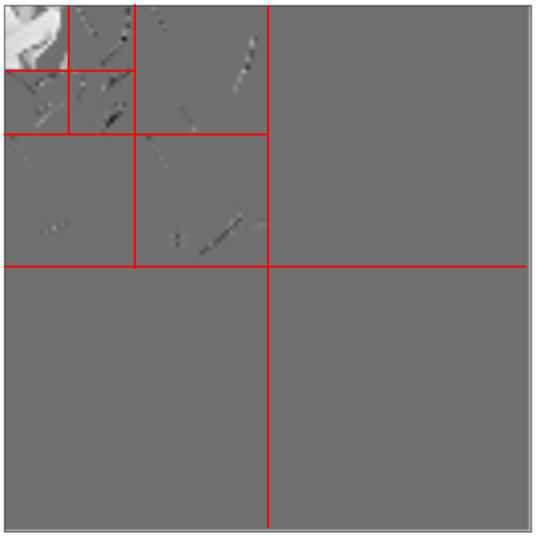
a) Original 2D Wavelet coefficients.



b) Bandelet coefficients: 222.



c) Thresholded 2D Wavelet coefficients: 235.



d) 2D Wavelet Coefficients reconstructed from Bandelet coefficients.

**Figure 55**. Bandelet and Wavelet coefficients for Lena's section of Figure 54 at a 0.11 bpp.

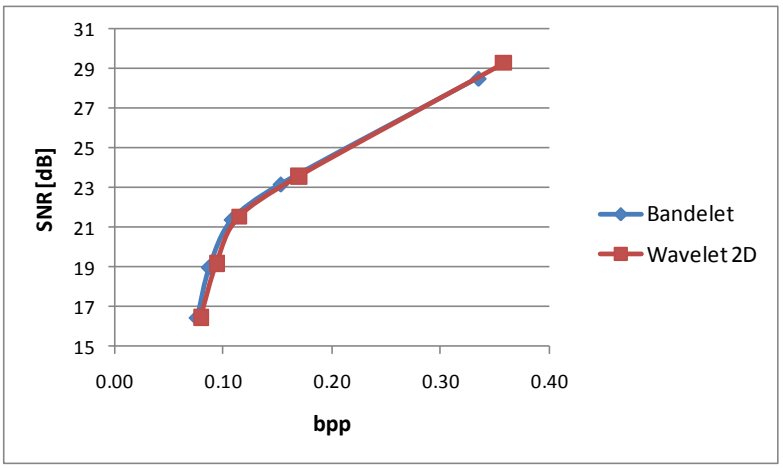
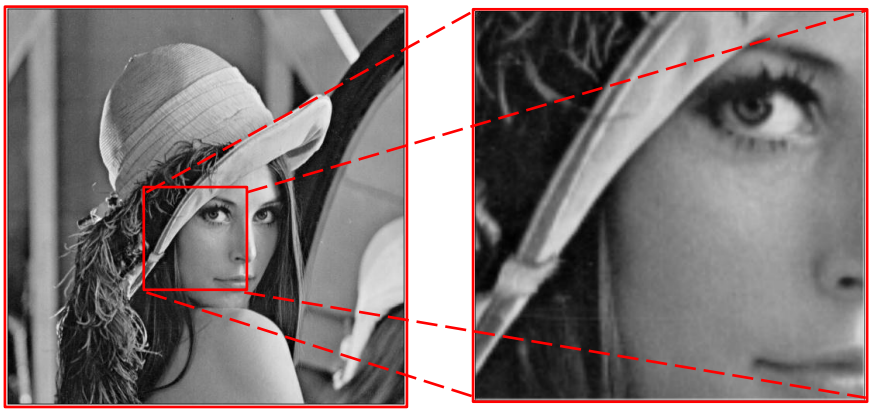



**Figure 56**. SNR vs. bpp for the section of Lena analyzed in Figure 54.

A second section of Lena was also analyzed. This is shown in Figure 57 for a rate of 0.19 bpp, where the Wavelet decoder has a slight advantage over the Bandelet decoder (22.12 dB for the former compared to 21.89 dB for the latter). This advantage is not significant when the graph in Figure 58 corresponding to the SNR calculated for various compression rates for this image is analyzed. In this graph, the curves of SNR vs. bpp for both codecs are almost superposed, as it happens in the graph in Figure 56.

These results indicate that when Lena is processed with the 2D Wavelet Transform, the coefficients resulting from this transform does not contain redundant geometric information, so Bandelet Transform has no coefficients in which to focus to increase the compression ratio while maintaining image quality. The same happens with the section of Barb in Figure 52.



a) Original section of Lena: 128x128 pixels = 16384 bytes.

b) Lena with Bandelets: Threshold=0.2, SNR=21.89 dB.

c) Lena with Wavelets: Threshold=0.2, SNR=22.12 dB.

**Figure 57.** Compression of a section of Lena at 0.19 bpp.

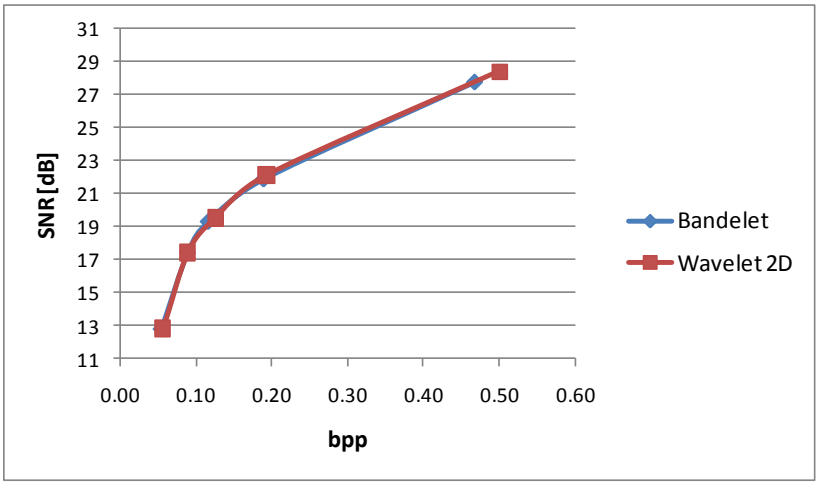
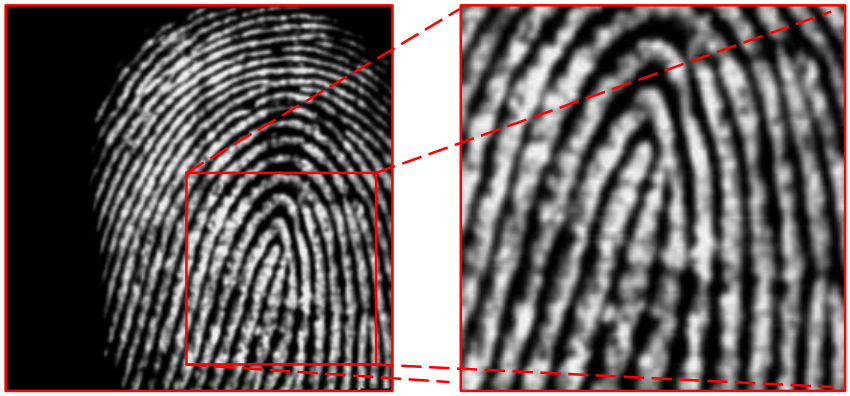
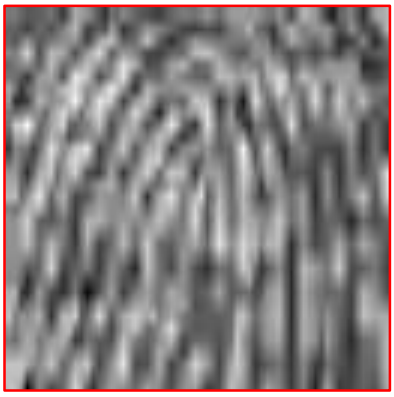



**Figure 58**. SNR vs. bpp for the section of Lena analyzed in Figure 57.

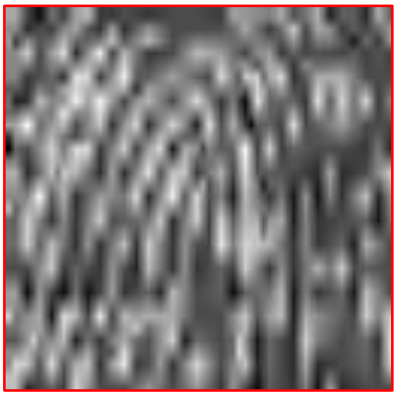
Figure 59 presents another example of an image that is best compressed with Bandelet Transform; in this case, the area of interest is the center of fingerprint.bmp. The image compressed with Bandelets keeps more ridge characteristics of the fingerprint than the image with Wavelets, allowing for a best identification of the image in a security system. Figure 60 presents the calculated SNR for this image for various compression rates, with a clear superiority of the Bandelet Transform over the 2D Wavelet Transform.



a) Original section of fingerprint: 128x128 pixels.



b) Fingerprint with Bandelets:
Threshold: 0.4, SNR=13.19 dB.



c) Fingerprint with Wavelets:
Threshold: 0.44, SNR=12.19 dB.

**Figure 59.** Compression of fingerprint.bmp at 0.14 bpp.

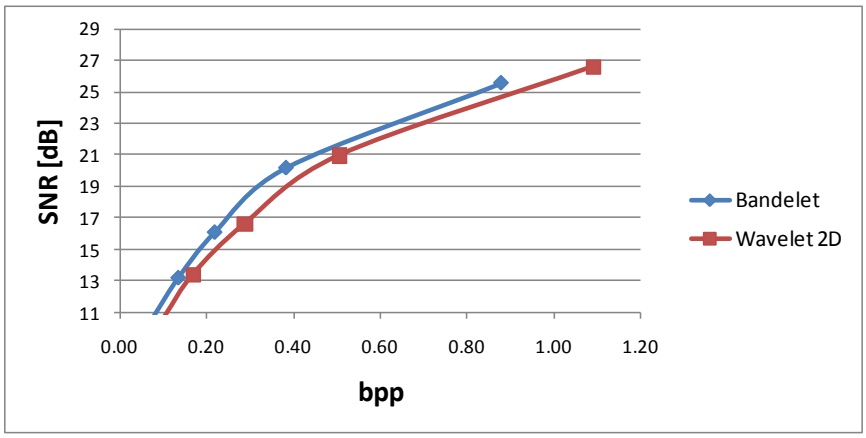



**Figure 60.** SNR vs. bpp for fingerprint.bmp.

## 7.2. Analysis of the Execution Times of the Encoder

The execution time of each of the processes involved in the Bandelet encoder was measured for the image fingerprint.bmp using the Altera timestamp timer. This component is a simple timer included in the system using SoPC Builder and is configured as a timestamp in NIOS II IDE. Its main function is to count the clock cycles since the last time it was reset. The basic functions associated with this timer are explained in Section 5.1.3.3.

For the analysis of the execution times of the Bandelet encoder, two scenarios were considered: The encoder implemented only in language C routines and the encoder with routines of the filters Le Gall 5/3 accelerated in hardware with NIOS II C2H Compiler. The complete timing analysis is presented in Table II. In this table, the time for the Extraction of Points process only corresponds to the diagonal orientation of the upper scale and the Direct Bandelet Transform (DBT) time refers to the complete compression process, including additional processes such as matrix handling and miscellaneous calculations not shown in the table.

**TABLE II**

TIME REDUCTION PERCENTAGE INTRODUCED TO THE BANDELET ENCODER BY THE ACCELERATION OF THE 5-TAP AND 3-TAP FILTERS (ALL UNITS IN SECONDS)

|  fingerprint.bmp Threshold = 0.20 | Data Cache 1kB *Data Vectors in On-Chip Mem Filter Coeff as Constants* Instruction Cache | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 32kB | | 16kB | | 8kB | | 4kB | |
| | **Soft** | **C2H** | **Soft** | **C2H** | **Soft** | **C2H** | **Soft** | **C2H** |
| **2D DWT** | 3.1640 | 2.9323 | 3.3530 | 3.0281 | 3.5134 | 3.1788 | 3.9210 | 3.5842 |
| **1D DWT (one row)** | 0.0073 | 0.0067 | 0.0076 | 0.0069 | 0.0080 | 0.0073 | 0.0089 | 0.0082 |
| **Lagrangian** | 0.0003 | 0.0003 | 0.0003 | 0.0003 | 0.0003 | 0.0003 | 0.0003 | 0.0003 |
| **Extraction of Points (Diagonal)** | 34.9101 | 32.4310 | 36.6031 | 33.4353 | 38.3981 | 35.0622 | 42.6013 | 39.3605 |
| **Direct Bandelet Transform** | **139.48** | **129.43** | **146.27** | **133.47** | **153.42** | **139.99** | **169.28** | **157.15** |
| **Time Reduction Percentage in DBT** | | **7,2%** | | **8,8%** | | **8,7%** | | **7,2%** |

The time reduction percentage in the DBT was calculated for various data and instruction cache sizes in the NIOS II processor. It was found that the maximum time reduction percentage (8.8%) is presented with a 1kB data cache and a 16kB instruction cache, having the input and output data vectors of the DWT stored in the FPGA on-chip memory and the filter coefficients defined as constants in the code. However, the minimum DBT execution time (129.43 s) is obtained when a 32kB instruction cache is used.

The data cache size has a significant impact on the performance of the DBT since every time the system calls the wavelet filter accelerators, the processor flushes the data cache and write it to memory to avoid data cache coherency problems. This is shown in Table III, where the time reduction percentage with several data cache sizes is presented. With a data cache size of 64kB, the time reduction percentage introduced to the system is negative, but this percentage becomes larger as the data cache size decreases, having a maximum value with a 1kB data cache.

The accelerators generated for the 5-tap and 3-tap filters have a Loop Latency of 23, while their CPLI is 1. The Loop Latency is defined as the number of clock cycles that the accelerator state machine needs to fill its internal pipeline and compute the results of the first loop iteration, while the CPLI (Cycles per Loop Iteration) is the number of clock cycles it takes to fill the pipeline with the data required to compute every new iteration. This means that after the initial delay caused by Loop Latency, the state machine completes every loop iteration in CPLI = 1 clock cycles.

**TABLE III**

IMPACT OF THE DATA CACHE SIZE IN THE TIME REDUCTION PERCENTAGE INTRODUCED TO THE BANDELET ENCODER (ALL UNITS IN SECONDS).

| Instruction Cache | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4kB | | | | | | | | | | | | | | | | | | | | | | | |
| Data Cache | | | | | | | | | | | | | | | | | | | | | | | |
| | No Data Cache | | 64kB | | 64kB | | 4kB | | 2kB | | 1kB | | 1kB | | 512B | | | | | | | | |
| | *Data Vectors in SDRAM* | | *Data Vectors in SDRAM* | | *Data Vectors in SDRAM* | | *Data Vectors in SDRAM* | | *Data Vectors in SDRAM* | | *Data Vectors in SDRAM* | | *Data Vectors in On-Chip Mem* | | *Data Vectors in On-Chip Mem* | | | | | | | | |
| | *Filter Coeff in SDRAM* | | *Filter Coeff in SDRAM* | | *Filter Coeff as Constants* | | *Filter Coeff as Constants* | | *Filter Coeff as Constants* | | *Filter Coeff as Constants* | | *Filter Coeff as Constants* | | *Filter Coeff as Constants* | | | | | | | | |
| | Soft | C2H | Soft | C2H | Soft | C2H | Soft | C2H | Soft | C2H | Soft | C2H | Soft | C2H | Soft | C2H | | | | | | | |
| fingerprint.bmp | | | | | | | | | | | | | | | | | | | | | | | |
| Threshold = 0.20 | | | | | | | | | | | | | | | | | | | | | | | |
| **2D DWT** | 11.9284 | 10.9592 | 3.6720 | 3.8995 | 3.6022 | 3.8150 | 3.6683 | 3.5514 | 3.7266 | 3.5733 | 3.9593 | 3.6371 | 3.9210 | 3.5842 | 4.0634 | 3.7034 | | | | | | | |
| **1D DWT (one row)** | 0.0267 | 0.0244 | 0.0084 | 0.0097 | 0.0082 | 0.0094 | 0.0084 | 0.0081 | 0.0084 | 0.0081 | 0.0088 | 0.0083 | 0.0089 | 0.0082 | 0.0092 | 0.0085 | | | | | | | |
| **Lagrangian** | 0.0009 | 0.0009 | 0.0003 | 0.0003 | 0.0003 | 0.0003 | 0.0003 | 0.0003 | 0.0003 | 0.0003 | 0.0003 | 0.0003 | 0.0003 | 0.0003 | 0.0003 | 0.0003 | | | | | | | |
| **Extraction of Points (Diagonal)** | 42.8394 | 42.8395 | 39.8172 | 42.9120 | 39.0097 | 42.9123 | 40.1073 | 38.9253 | 40.4721 | 38.9061 | 42.2869 | 39.7495 | 42.6013 | 39.3605 | 42.9054 | 41.0123 | | | | | | | |
| **Direct Bandelet Transform** | 251.5282 | 241.3405 | 159.0865 | 172.0275 | 155.8581 | 171.1274 | 160.3156 | 155.4077 | 161.6052 | 155.5258 | 168.9907 | 159.0459 | 169.2874 | 157.1568 | 170.8031 | 163.4626 | | | | | | | |
| **Time Reduction Percentage in DBT** | | 4,1% | | -8,1% | | -9,8% | | 3,1% | | 3,8% | | 5,9% | | 7,2% | | 4,3% | | | | | | | |

# *Chapter 8*

# *Conclusions and Future Work*

This research work presents the design of a codec for grayscale images based on the Bandelet Transform and implemented in a SoPC with an Altera NIOS II processor. Using several test images like Barb and Lena, it was found that the Bandelet codec offers an improvement of up to 2dB compared to a codec based uniquely on the 2D Wavelet Transform for the same compression ratio. However, tests showed that the Bandelet codec is effective only when the processed image has areas with highly geometric components, i.e., edges and line patterns. When the image is processed with the 2D Wavelet Transform, the coefficients of these geometric components are those that are further away from zero and therefore have a greater amount of redundant geometric information than the rest of the coefficients. The Bandelet Transform focuses on these high coefficients and re-transforms them to store the same information contained in the 2D Wavelet coefficients with a lower amount of memory. The results presented in this project were obtained with a fixed segmentation of the image into squares of 8x8 pixels for purposes of simplifying the design and operation of the Bandelet codec. However, its performance can be enhanced if an optimal dyadic segmentation is implemented by the Lagrangian comparison of the squares and sub-squares, but this comparison process will increase the complexity of the algorithm and its execution time.

The codec was implemented as software routines in language C; two routines, those related with the filters of the 2D and 1D Wavelet Transforms, were synthesized in hardware using Altera NIOS II C2H Compiler. The acceleration of these functions introduced to the system a total acceleration percentage of 8.8% when the codec timing analysis was performed for a grayscale image of 128x128 pixels with a 1kB data cache and a 16kB instruction cache. The codec system can have a greater acceleration percentage than that obtained here if all or most of the components of the codec are implemented in hardware. For this implementation, designer can exploit the fact that the Bandelet encoder processes independently each of the

squares of the image segmentation to find the best geometric direction in which the 1D Wavelet Transform of the square pixels is performed. Therefore, it is possible to design a parallel set of *n* Extraction of Points blocks for processing concurrently *n* squares of the image in order to accelerate the compression process.

This research project also allowed having a thorough understanding of JPEG and JPEG2000 compression standards based on DCT and DWT respectively. The study of these standards was made with the implementation of two FPGA-based embedded platforms that supported the processes of reading, editing, and compressing images in the two standards. The design of the JPEG Image Viewer and the Image Editor with JPEG/JPEG2000 Compressor provided a better understanding of the techniques, methodologies, and state-of-the-art tools available on the market for implementing embedded multimedia applications for digital image processing using modern components as touch screens and SD-Cards.

Summarizing, the future work for this research project could be aimed to:

- The complete implementation of the Bandelet codec in hardware, starting with the construction of a parallel set of *n* Extraction of Points blocks.
- The design of a process responsible of obtaining the optimal dyadic segmentation of the image based on the Lagrangian comparison of squares and sub-squares.
- The application of the Bandelet codec in color images.

# *References*

[1]. SD Association, ***microSDHC: The Smallest Card That Does Even More***, October 2009. Available: http://www.sdcard.org/developers/tech/microsdhc/

[2]. G. Peyré, S. Mallat, ***Discrete Bandelets with Geometric Orthogonal Filters***, IEEE International Conference on Image Processing, 2005, vol. 1, pp. I– 65–8.

[3]. Let It Wave, ***Let It Wave Webpage***, September 2008. Available; http://www.letitwave.fr/

[4]. E. Le Pennec, S. Mallat, ***Sparse Geometric Image Representations With Bandelets***, IEEE Transaction on Image Processing, 2003, vol. 14, No. 4, pp. 423–438.

[5]. G. Peyré, S. Mallat, ***Surface Compression with Geometric Bandelets***, ACM Transactions on Graphics, 2005, vol. 24, pp.601-608.

[6]. E. Le Pennec, S. Mallat, ***Bandelet Image Approximation and Compression***, Society for Industrial and Applied Mathematics, 2005, vol. 4, No. 3, pp. 992–1039.

[7]. B. Song, L. Xu, W. Sun, ***Image Denoising Using Hybrid Contourlet and Bandelet Transforms***, Fourth International Conference on Image and Graphics, 2007, ICIG 2007, pp. 71-74.

[8]. S. Li, R. Yang, Q. Qin, ***Improvement of JPEG2000 in Retaining Texture via Bandelet***, International Conference on Computer Science and Software Engineering, 2008, vol. 4, pp 952-955.

[9]. X. Delaunay, M. Chabert, V. Charvillat, G. Morin, R. Ruiloba, ***Satellite Image Compression by Directional Decorrelation of Wavelet Coefficients,*** IEEE International Conference on Acoustics, Speech and Signal Processing, 2008, pp. 1193-1196.

[10]. O. Alatas, O. Javed, M. Shah, ***Video Compression Using Structural Flow,*** IEEE International Conference on Image Processing, 2005. ICIP 2005, vol. 3, pp. 245-248.

[11]. G. Peyré, S. Mallat, ***Bandelets toolbox***, Matlab Central, September 2008. Available: http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=7914&objectType=FILE.

[12]. Altera Corp., Let It Wave Corp., ***The Quest for Digital Broadcast Quality: Addressing Quality Hot Spots,*** 2007. Available: http://www.altera.com/literature/wp/wp-01022.pdf

[13]. J. Brennan, ***FPGA Coprocessing in a JPEG2000 Implementation***, Undergraduate Thesis, University of Queensland, Australia, 2001. Available: http://innovexpo.itee.uq.edu.au/2001/projects/s369272/thesis.pdf.

[14]. M. Dyer, A. Kumar Gupta, N. Galin, ***NIOS II Processor-Based Hardware/Software Co-Design of the JPEG2000 Standard***, NIOS II Embedded Processor Design Contest—Outstanding Designs 2005, Altera Corp., 2005, pp. 24-36.

[15]. T. Acharya, P.S. Tsai, ***JPEG2000 Standard for Image Compression: Concepts, Algorithms and VLSI Architectures***, John Wiley & Sons, 2005, pp. 137-225.

[16]. The International Telegraph and Telephone Consultative Committee (CCITT), ***Information Technology – Digital Compression and Coding of Continuous-Tone Still Images – Requirements and Guidelines,*** Rec. T.81, 1992.

[17]. D. Salomon, ***Data Compression: The Complete Reference***, Springer, 3rd Edition, 2004.

[18]. M.D. Adams, F. Kossentini, ***JasPer: A Software-Based JPEG-2000 Codec Implementation***, Proceedings of 2000 International Conference on Image Processing, 2000, vol.2, pp. 53 – 56.

[19]. S. Mallat, ***A Wavelet Tour of Signal Processing: A Sparse Way***, 3rd. Edition, Academic Press, 2009.

[20]. X. Qu, J. Yan, G. Xie, Z. Zhu, B. Chen, ***A novel image fusion algorithm based on bandelet transform***, Chinese Optics Letters, 2007, vol. 5, Issue 10, pp. 569-572.

[21]. S. Yang, F. Liu, M. Wang, L. Jiao, ***Multiscale Bandelet Image Compression***, Proceedings of 2007 International Symposium on Intelligent Signal Processing and Communication Systems, 2007, pp. 412-415.

[22]. G. Peyré, E. Le Penne, C. Dossal, S. Mallat, ***Geometrical Image Estimation with Orthogonal Bandlet Bases***, Proceedings of the SPIE, 2007, vol. 6701, pp. 67010M.

[23]. J. Arteaga, J. Velasco-Medina, ***Visor de Archivos JPEG usando el Procesador NIOS II***, IBERCHIP XV Workshop, Argentina, 2009.

[24]. Altera Corp, ***DE2 Development and Education Board: User Manual***, 2006.

[25]. Terasic Technologies, ***DE2 CD-ROM***, December 2008. Available: http://www.terasic.com/downloads/cd-rom/de2/

[26]. L. Saillard, ***Tiny Jpeg Decoder: Small jpeg decoder to be used by embedded applications***, December 2008. Available: http://www.saillard.org/programs_and_patches/tinyjpegdecoder/

[27]. Altera Corp, ***NIOS II Processor Reference Handbook***, 2008.

[28]. Altera Corp, ***NIOS II Custom Instruction: User Guide***, 2008.

[29]. Independent JPEG Group, ***Free library for JPEG image compression***, July 2009. Available: http://www.ijg.org/

[30]. M.D. Adams, ***The JasPer Project Home Page,*** February 2010. Available: http://www.ece.uvic.ca/~mdadams/jasper/

[31]. M.D. Adams, R. K. Ward, ***JasPer: A Portable Flexible Open-Source Software Tool Kit for Image Coding/Processing***, Proceedings of 2000 International Conference on Acoustic, Speech and Signal, 2004, vol. 5, pp. 241 – 244.

[32]. Altera Corp, ***TRDB_LTM 4.3 Inch Digital Touch Panel Development Kit***, 2008.

[33]. Altera Corp, ***NIOS II C2H Compiler User Guide NIOS II***, 2008.

[34]. G. Strang, T. Nguyen, ***Wavelets and Filter Banks***, Wellesley-Cambridge Press, 1996.

[35]. O. Cantineau, B. Jentz, ***Enabling real-time JPEG2000 with FPGA architectures***, Altera Corp. 2005. Available: www.altera.com/literature/cp/gspx/jpeg2000.pdf

[36]. Altera Corp, ***DE2-70 Development and Education Board: User Manual***, 2008.

[37]. C. Christopoulos, A. Skodras, T. Ebrahimi, ***The JPEG2000 still image coding system: an overview***, IEEE Transactions on Consumer Electronics, 2000, vol. 46, issue: 4, pp. 1103-1127.

[38]. D. Santa-Cruz, T. Ebrahimi, ***A study of JPEG2000 still image coding versus other standards***, Proceeding of the X European Signal Processing Conference, 2000, vol. 2, pp. 673-676.

[39]. S. Mallat, ***A Theory for Multiresolution Signal Decomposition: The Wavelet Representation***, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1989, vol. 11, No. 7, pp. 674-693.

[40]. R. L. de Queiroz, K. R. Rao, ***On Reconstruction Methods for Processing Finite-Length Signals with Paraunitary Filter Banks***, IEEE Transactions on Signal Processing, 1995, vol. 43, No. 10, pp. 2407-2410.

[41]. M.E. Domínguez, N. González Prelcic, ***Smooth Orthogonal Signal Extensions For Paraunitary Tree-Structured Filter Banks***, IEEE International Conference on Acoustics, Speech, and Signal Processing, 1993, pp. II-II.

[42]. G. Peyré, ***A Numerical Tour of Signal Processing***, June 2010. Available: http://www.ceremade.dauphine.fr/~Peyré/numerical-tour/

[43]. P. Nilsson, ***Hardware / Software codesign for JPEG2000***, Master Thesis, Linköping University, Sweden, 2006. Available: http://liu.diva-portal.org/smash/record.jsf?pid=diva2:21516.

[44]. S. Mallat, ***Super-Resolution Bandlet Upconversion for HDTV***, Let It Wave, 2006. Available: http://www.cmap.polytechnique.fr/~mallat/papiers/whitepaper.pdf.

[45]. R. de Queiroz, C.K. Choi, Y. Huh, K.R. Rao, ***Wavelet transforms in a JPEG-like image coder***, IEEE Transactions on Circuits and Systems for Video Technology, 1997, vol. 7, No. 2, pp. 419-424.

[46]. Altera Corp, ***NIOS II Software's Developers Handbook***, 2008.