FPGA IMPLEMENTATION OF A VIDEO WATERMARKING ALGORITHM

by

William A. Irizarry-Cruz

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
in
ELECTRICAL ENGINEERING

UNIVERSITY OF PUERTO RICO
MAYAGÜEZ CAMPUS
2006

Approved by:

_____	_____
Nayda G. Santiago Santiago, PhD Member, Graduate Committee	Date
_____	_____
Rogelio Palomera García, PhD Member, Graduate Committee	Date
_____	_____
Manuel Toledo Quiñones, PhD President, Graduate Committee	Date
_____	_____
Pedro Resto, PhD Representative of Graduate Studies	Date
_____	_____
Isidoro Couvertier-Reyes, PhD Chairperson of the Department	Date

Abstract

This thesis explores the FPGA-implementation of video watermarking. The watermarking of the real-time video stream produced by a battery-operated surveillance camera was selected as a prototype application. This application was chosen because it can take advantage of the low-power characteristics of the FPGA, and provided the project's context and requirements.

A comprehensive study of the relevant literature revealed no previous research on the use of an FPGA to implement video watermarking. After a detailed analysis of the relevant literature, the algorithm developed by Hartung and Girod to watermark MPEG-2 compressed video was chosen, and adapted to the requirements of the present project. A low-power, distributed-arithmetic version of the discrete cosine transform -the primary computational unit required by the algorithm- was used.

The FPGA implementation was compared with one performed on a digital signal processor. In terms of speed of processing, power consumption and device cost, the results suggest that the FPGA is a better option. However, the digital signal processor implementation appears to require a lower development cost.

The following contributions have been accomplished in this project: (1) a methodical analysis of the issues related to watermarking for surveillance systems was performed; (2) design practices for the low-power implementation of this type of algorithm were identified; and (3) a novel architecture for video watermarking on an FPGA was developed. These results can be extended relatively straightforwardly to other watermarking methods that rely on the DCT.

Resumen

Esta tesis explora la implementación de un algoritmo de marca de agua digital para video en un FPGA. El uso de este algoritmo en el video producido por una cámara de vigilancia operada con pilas eléctricas fue seleccionado como una aplicación prototipo que proveyó un contexto para el proyecto y definió los requisitos del mismo. Esta aplicación fue escogida porque puede sacar ventaja del bajo consumo de potencia del FPGA.

Un estudio comprensivo de la literatura relacionada con el tema reveló que aparentemente no existe investigación previa en el uso de un FPGA para implementar el marcaje de agua de video. Luego de un detallado análisis de la literatura, el algoritmo desarrollado por Hartung y Girod para marcar video comprimido con el formato MPEG-2 se seleccionó, y fue adaptado para cumplir con los requisitos del proyecto presente. Una versión de bajo consumo de potencia de la transformada discreta del coseno -la unidad computacional principal requerida por el algoritmo escogido- que utiliza aritmética distribuida fue seleccionada para ser implementada

La implementación en el FPGA fue comparada con una similar efectuada en un procesador de señales digitales. En término de rapidez de procesamiento, consumo de potencia, y costo por unidad, los resultados sugieren que el FPGA es una mejor opción. Sin embargo, aparentemente el costo de desarrollo de la implementación que utiliza el procesador de señales digitales es inferior.

Este trabajo contribuye los siguientes logros: (1) se efectuó un análisis metódico de los problemas relacionados con el marcaje de agua de video; (2) se identificaron prácticas efectivas de diseño para implementar este tipo de algoritmo con un bajo consumo de potencia; y (3) se desarrollo una nueva arquitectura para la implementación del marcaje de agua de video en un FPGA. Estos resultados pueden extenderse con relativa sencillez a otros métodos de marca de agua basados en el cómputo de la transformada discreta del coseno.

To my God and my Creator,

And to my Parents William and Nilda.

Acknowledgements

I am deeply thankful to my advisor Dr. Manuel Toledo Quiñones, who guided me through this entire journey. His words of advice, his trust, and his patience and understanding helped me reach this pinnacle of my life.

I need to thank my family, my father, my mother and my sister, whose support and unconditional help were invaluable in completing this work; My girlfriend, who accompanied me in every stage of this work with her patience, friendship, openness and her caring; And my deepest thanks to the rest of my family - my grandparents, uncles, and my second family here in Cabo Rojo.

I want to express my gratitude to Dr. Rogelio Palomera, whose help since my first semester here in Mayaguez has been priceless. Thank you for not giving up on me on the TI experience. I need to thank also my former supervisor in Texas Instruments, Venkateswar Kowkutla, who allowed me to grow both professionally and academically during my Co-op term, and directed my Co-op towards my Master's studies.

Also I want to show my appreciation to Professor Nayda G. Santiago for being part of my graduate committee, for her help and consideration during the completion of this work, and for lending me the DSP board used for this project. It was a shame that I got to meet her so late in my academic studies.

I would like to acknowledge the Electrical and Computer Engineering department for the financial help during several semesters of my graduate studies, and for the opportunity of a teaching experience as a laboratory instructor.

I am also very grateful to Javier Morales and Ivan Rivera, whose friendships and their unconditional help during the hardest times of my studies were priceless. I sincerely wish the best to them wherever they go.

Finally, I want to thank my Lord, He who watches over me wherever I go. Thank you especially for the difficulties, vicissitudes, obstacles and struggles I have lived through out my Master's studies. With out them I could have not been able to experience your support and sustain in those difficult times.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

Digital watermarking is the process of embedding information, or a watermark, into a digital multimedia object such that the watermark can be detected or extracted later to make an assertion about the object. Watermarking has proven to be a reliable mean to provide copy protection and authenticity proof for digital media, and therefore a lot of research has been performed in these areas.

The concept of watermarking comes from more than 700 years back. It was a technique used by paper manufacturers to identify their products. Today, watermarks in paper can still be seen. Along with the years the concept of watermarking has penetrated into the field of security. Currency, such as dollar bills, checks, postal stamps, and official documents from government can be seen to carry watermarks. Besides these paper-based applications, watermarking can also be used to provide the same degree of security to digital media data, such as audio, text and still images.

Figure 1 Example of a U.S. ten-dollar bill watermark. [59]

In this work we are interested in video watermarking, which can be also be analyzed as an extension of still image watermarking. In still image watermarking there exist two major types of watermarking: visible and invisible types. Visible watermarking marks a logo or text on the image of interest and resembles the traditional paper-based ones.

Figure 2 Example of a visible watermarking of an Image. Left picture: Watermark. Right picture: Watermarked image. Taken from [6]

Invisible watermarking marks the image of interest, but the mark is not visible to the human eye. In order to detect and retrieve this hidden watermark computer processing must be performed. If you subtract the original image to the watermarked image the effect of the watermark can be observed, although this may not be the actual watermark.

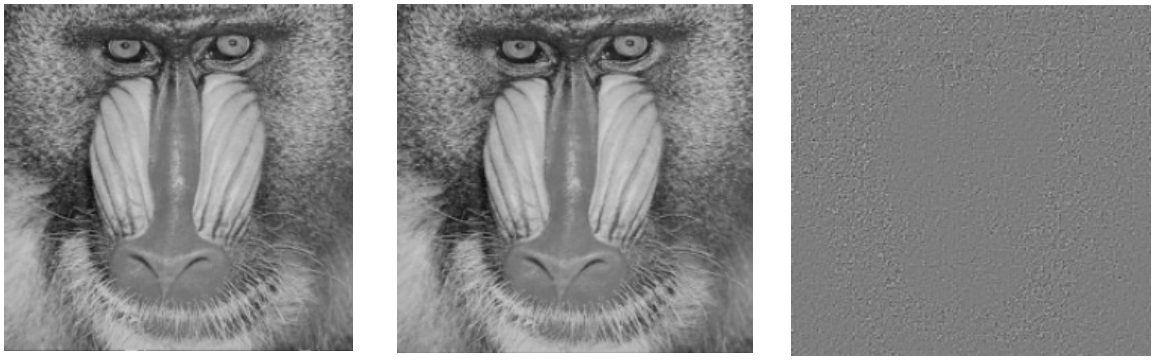


Figure 3 Example of watermarking of an Image. Left picture: original image. Middle picture: watermarked image. Right picture: difference between original and watermarked images. Taken from [1]

Invisible watermarking is usually used for authentication purposes, for fingerprinting, and for copyright protection.

One application where watermarking could be of significant use is in the authentication of video output from surveillance or security cameras. The transition from analog to digital in the surveillance field has left a security hole when trying to provide surveillance video evidence to a court of law. Digital surveillance video cannot be accepted as legal evidence because modifying digital

media is straightforward, and the availability of image processing tools make authenticity questionable. Essentially it is possible to tamper a digital video without leaving a trace of the tampering activity.

Watermarking can provide the authentication and tampering detection needed for this new array of digital surveillance systems. Some systems currently provide watermarking options for stored video. These systems assume that the transmission path from the video camera to the central/storage unit is secure and they concentrate on securing the video that is already at the central unit. This assumption is not true if the system can be attacked in the transmission stage. In such cases authentication must occur inside the video camera to provide a secure system. This can be accomplished using a hardware-based or software-based device to apply a watermark to the video signal. Since the process of embedding a watermark is computationally intensive, a hardware implementation may be needed. Some current security cameras include signal processing logic to process the images captured from the camera sensors, so adding additional watermarking logic may not represent dramatic changes to the system.

Hardware implementations in digital watermarking are scarce and may not be justified for most applications. Several hardware implementations are intended for video applications, were software solutions tend to struggle to satisfy the computational demands. A high percentage of these works are ASIC-based implementations (Application Specific Integrated Circuit). Only two of the works found are FPGA-based implementations (Field Programmable Gate Array).

FPGA devices have improved on speed, capacity, flexibility, and power dissipation over the years. Current applications where FPGA can be utilized include digital signal processing, computer vision, speech recognition, computer hardware emulation, and cryptography. Applications that contain heavy amounts of parallelism can benefit the most from the FPGA architecture [2]. For video watermarking, the FPGA should provide the benefits of parallel processing and specific-architecture design offered by ASICs, but at a fraction of the price.

Therefore, the goals of this thesis are to select a watermarking method appropriate for working on a low-power secure camera, and to develop a hardware implementation of the watermarking algorithm for digital video in an FPGA device, in order to study the issues related to implementing such an algorithm in an FPGA device, and explore the potential of performing such algorithms in hardware platforms. Since the Digital Signal Processor (DSP) is traditionally used to handle similar

tasks, a comparison with the FPGA implementation is provided. The comparison will be in terms of performance or speed, power dissipation, unit cost, and development cost. The results will be discussed and compared to previous work in the area.

1.1. Contributions

The main contributions of this project can be summarized as follows:

- a methodical analysis of the issues related to watermarking for surveillance systems was performed;
- design practices for the low-power implementation of this type of algorithm were identified;
- a novel architecture for video watermarking on an FPGA was developed;
- the requirements of a digital security watermarking system were identified;
- a watermarking method capable of satisfying the requirements of a low-power security camera was methodically identified after a comprehensive study of the relevant literature;
- an appropriate data-path word-size was selected;
- the watermarking method developed by Hartung and Girod [42] was adapted for implementation in an FPGA;
- the previously unexplored area of FPGA-based low-power digital video watermarking was studied.

The results presented in this document were obtained for the specific algorithm and device selected. However, they can serve as an indication of how other similar watermarking methods would perform in an FPGA implementation. The methods described here to other watermarking algorithms that rely on the DCT appear to be relatively straightforward.

1.2. Organization

The following chapters are organized as follows. Chapter 2 introduces the reader to the concept of watermarking and presents the different types. Relevant work in the areas of video watermarking, authentication, tamper detection, and hardware implementations are reviewed. Chapter 3 presents in more detail the case study and introduces the requirements of a watermarking method for this particular application. A brief overview of compression techniques is also presented. Chapter 4 discusses the watermarking method selection. Chapter 5 describes the resulting FPGA and DSP implementations in detail. Synthesis and Place & Route results for the FPGA implementation are presented. The methods employed for the characterization are also discussed. Finally, an analysis of the result is included in chapter 6, were our conclusions are presented.

2 THEORETICAL BACKROUND AND LITERATURE REVIEW

2.1. Background on Watermarking

Watermarking is defined in [3] as the direct embedding of additional information into the original content or host signal, and in [4] as a technique to embed invisible or inaudible data within multimedia content. Usually the watermark is imperceptible to the human. There should be no way to remove or modify the watermark without changing or altering the content (or signal). The watermark can carry or provide information. In general, watermarking must comply with the following three requirements: (1) imperceptibility, (2) robustness and (3) capacity. In [3] the authors found that sometimes these requirements conflict with each other. The tradeoff between these requirements depends on the application the watermark is intended for. For example, if one desires to test for tampering, one would employ an algorithm that guarantees imperceptibility but that is not robust to any modification of the content (this is termed as fragile or semi-fragile watermark). On the other hand, if one desires a copyright protection that must withstand an irreversible or lossy transformation or additional attacks, a very robust watermarking algorithm would then be selected. Some transformations, or attacks, that the signal may be subject to are resampling, rescaling, compression, linear and nonlinear filtering, additive noise, A/D and D/A conversion. Watermarks typically are not retrieved, they are only detected. Detection is usually performed by correlation methods, correlating the watermarked data with the watermark sequence. The value of correlation is compared against a threshold value, which is then used to decide if the watermark was detected or not. The threshold value is determined by the application and trial-and-error runs.

Watermarking can have various purposes which include copyright protection, authentication, tamper detection, and data hiding. It can be applied to different media types such as digital images, video, graphics, audio, text, and multimedia content. The creation of the internet and the conversion of audio-visual and textual content to digital format have allowed replicating and distributing digital content over and over, without any visible penalty on the data. Therefore, there seems to be an increasing desire to protect property rights for digital media according to [3]. The authors in [5] agree

that in the past 10 years there has been a new and great interest in the area of digital watermarking as a way to help to protect authenticity and prevent unauthorized replication of media.

Watermarking is as old as paper manufacturing itself because watermarks were a by-product of the process of making paper. Years ago our ancients, during the process of making paper, poured a mix of slurry of fiber and water on to mesh molds to collect the fiber. Then this slurry was dispersed to add shape and uniformity, and finally great pressure was applied to the mesh molds in order to expel the water and cohere the fiber [6]. Years ago this by-product, coined watermark, was use to establish the authenticity of a product or certify something about the product. In present days this same principle is applied using digital watermarks. And as the authors in [6] state, whether the product of paper press or discrete cosine transformations, watermarks of varying degrees of visibility are added to presentation media as a guarantee of authenticity, quality ownership and source.

The watermarking technique has evolved from steganography [6], but steganography and watermarking have their differences. In watermarking, protecting the content that carries the watermark is essential, whereas in steganography the content is of no value and the message that is covered in the content is the significant one. So the applications of both concepts are very different, but the uses sometimes overlap.

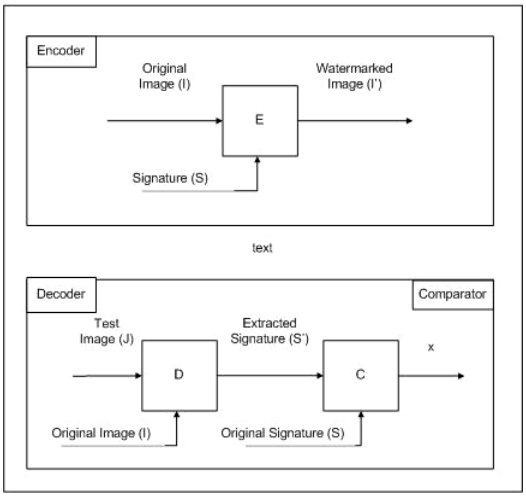


Figure 4 Block diagram of the encoder and decoder stages of a watermarking system [6]

Typical applications for watermarking include digital archives, copyright protection, legal delivery of content, anti-piracy, and automatic broadcast monitoring. One example of a practical use of

watermarking comes from the Academy Awards. When the Academy sends "screeners" to its voters, the movies include a watermark in each of its frames. A distinct watermark is used for each recipient. If the movie gets illegally distributed, the watermark then allows the Academy to know which voter was the source for the pirated version.

2.2. Types of Watermark

In [4] we have a good decomposition of the variety of watermarks currently available, their definitions, their features and possible applications, advantages and disadvantages (see Fig. 5.). One thing to point out of this classification is that video watermarking is an extension of image watermarking, which utilizes characteristics of the Human Visual System (HVS) to embed the watermark. HVS methods take advantage of the way the human eye processes images in order to add watermarks to the images. The video method requires that the watermarking encoding or decoding process be done in real-time and that the watermark is robust for compression, that is, the watermark should be present in the signal after compressing the video. According to the survey done in [3], Podilchuk et al. were the first to state that for the watermark to be robust it had to be embedded into the perceptually significant portions of the data, although Cox et al. [8] were also pioneers in watermarking perceptually significant areas. Another good source of information on the different watermarking methods available is [8].

Watermark processing methods fall into two categories: spatial domain and frequency domain. Spatial domain usually changes the value of the pixels in a minor way so it is not perceived by the human eye and the watermark is scattered through the entire object. Frequency domain watermarking transforms the object into its frequency counterpart and then embeds the watermark in the transform coefficients, distributing the watermark over the entire frequency distribution of the object. The frequency domain watermarking methods are relatively robust to noise, image processing and compression compared with the spatial domain methods [4, 6]. Extensive work has been done in both processing areas, but watermarking techniques based on the transfer domain are more popular than those of the spatial domain. The Discrete Cosine Transform (DCT) methods are the most widely used among these types of methods [4].

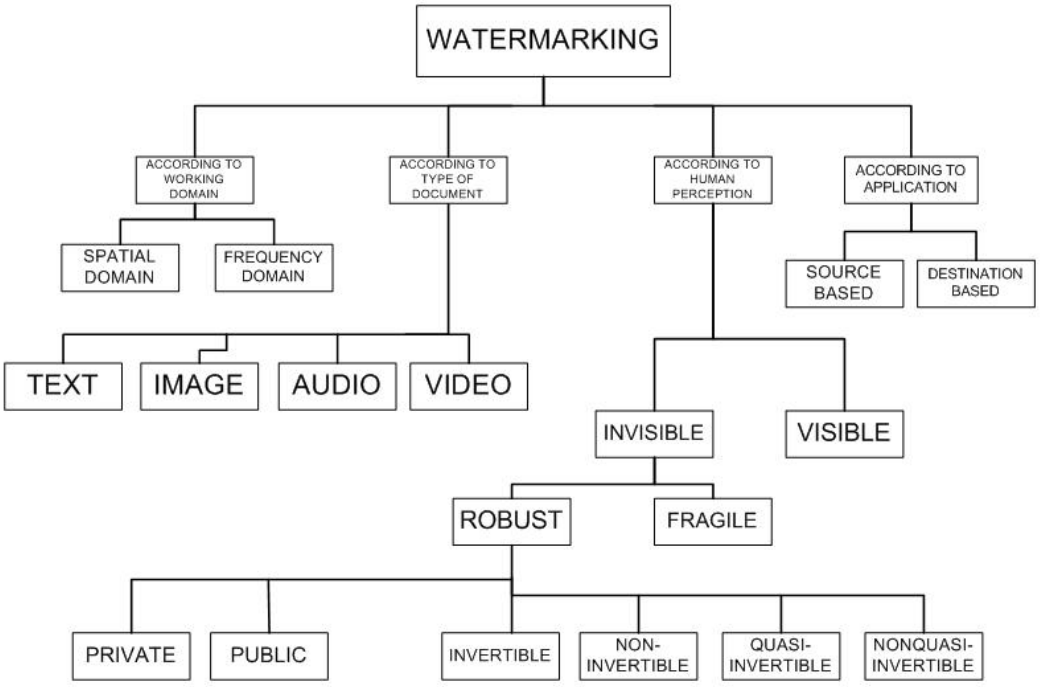


Figure 5 Types of digital watermarking schemes [6]

2.3. Video Watermarking

Video watermarking can be seen as an extension to image watermarking, although it owns particular features that differentiate it from image watermarking:

- The presence of watermark should not cause a visible or audible effect on the playback of the video.
- The watermark should not affect the compressibility of the digital content.
- The watermark should be detected with high degree of reliability. The probability of false detection should be extremely small.
- The watermark should be robust to various intentional and unintentional attacks.
- The detection algorithm, if implemented in circuitry, should be with small extra cost.
- Inserting the watermark should not increase the bit rate of the video signal.

Other features that may apply to video applications are:

- Watermarking could be done directly on compressed video also.
- Watermarking should have low complexity.
- Unauthorized removal of the watermark should be impossible once it is embedded.

Although these features exist, video watermarking can still be performed using image watermarking on a frame per frame basis, as the authors in [6] state. The authors in [7] also agree that all image watermarking techniques are equally applicable to video when the individual frames are treated as images. There are two categories in the video watermarking method: compressed and uncompressed (raw) video watermarking. There has been a lot of work done in this area. Some examples of both areas will be presented.

In [9] a spread spectrum method is used to watermark video frames and later compress them. In a color frame, the green component of the frame is chosen for insertion because it is the most robust after compression. The watermark is composed of a random sequence of real numbers with values from -1 to 1 generated with a key. The process divides the video frame up into 8 x 8 blocks and a forward Hadamard transform is executed for each block. The mid-frequency coefficients of the transform are selected to embed the watermark. The relative strength of the watermark can be controlled by the number of coefficients altered. The frame is then inverse-transformed and compressed for transmission. A correlation based detection scheme, wherein the modified coefficients are extracted and correlated with the *key,* is used to verify authenticity.

In [5] the direct spread spectrum concept of communications is employed to watermark a video signal. In a comparison to communication theory, the narrow band signal is represented by the watermark, and the wide-band channel with interference is represented by the video signal. The concept is applied to uncompressed and compressed video, where the basic idea is embedding the watermark in the transform domain as represented in the entropy coded DCT coefficients. This is done in an MPEG-2 video signal. Although an existing MPEG-2 bit-stream is partly modified, the scheme avoids visible artifacts by adding a drift compensation signal. This signal is needed because P and B frames on the MPEG-2 compression format rely on information found on I frames for encoding and decoding. For retrieval of watermark no original signal is needed. With this scheme the authors have achieved data rates for the watermark of a few bytes/second for ITU-R 601 format video and a robust watermark scheme against friendly or hostile manipulations. The complexity of the scheme is

comparable to MPEG decoding. The authors conclude that working on encoded rather than un-encoded video is important for practical watermarking applications.

Another work on MPEG-2 video compression is presented in [10] where the authors achieve a watermarking scheme that is undeletable, perceptually invisible, statistically undetectable, and robust to lossy compression and survives to video manipulation and processing. The scheme is also based on DCT transformations and coefficient alteration.

The work in [11] is suitable for watermarking and monitoring video streams in TV-broadcasting environments, such as movies or sporting events, as the authors affirm. The algorithm survives MPEG-2 compression of high-quality, real-world video sequences without degrading their quality.

Many other works in compressed video use ancillary fields, or other features of the compression format such as motion vectors to add authentication data. Such schemes are weak as anyone with a computer or a digital editing workstation would be able to convert the information to another format and remove the watermark at the same time [7].

2.4. Watermarking Hardware Implementations

Over the past decade, numerous watermarking algorithms have been invented and their software is available, however recently, hardware implementations are being presented in literature, the authors in [12] affirm. According to [13] only a few hardware schemes have been proposed. As proof of that, [12] provides the following table of most of the watermarking hardware implementations available in current literature.

Table 1 Watermarking Hardware Implementations [12]

Research	Design Type	Multimedia	Chip Features
Hsiao	Custom IC	Image	*NA*
Maes	FPGA board/IC	Video	17/14 kG Logic
Tsai	Custom IC-0.35μ	Image	3.3V 50MHz
Petitjean	FPGA board	Image	50MHz

Garimella	Custom IC-0.13μ	Image	1.2V
Mathai	Custom IC-0.18μ	Video	1.8V
Tsai	Custom IC	Video	*NA*
Mohanty	Custom IC-0.35μ	Image	3.3V 545MHz
Seo	FPGA board	Image	82MHz
Mohanty	Custom IC-0.35μ	Image	3.3V 292MHz

Hardware implementations of watermarking can be implemented in Application Specific Integrated Circuits (ASICs) or in Field Programmable Gate Arrays (FPGAs). As can be seen from the table above, most of the current hardware implementations have been done for ASIC designs. Recent advances in FPGA technology, such as 90nm process devices, higher gate densities, better interconnect architectures, reduction in power consumption, multiple I/O formats and embedded optimized logic, have allowed for applications that were previously intended for ASICs to be implemented in FPGA devices, with the added value of a lower FPGA cost when compared to an ASIC. But for some reason that field has been understudied.

Watermarking implementation in hardware is a recent interest in the area. According to [6], in 1999 there were still no works of video watermarking implementation in hardware. Therefore, all of the works in this area are from the past 5 years, and nevertheless there are still very few works published. Although the small amount of available works is a fact, the work in [14] states that there is a strong motivation for hardware implementations because real-time watermarking of video streams is too expensive for software. Audio and Image watermarking are typically done in software because their low data rates allow them to be processed in software.

As can be seen, watermarking implementations can be done in software or in hardware. The authors in [15] state that although it might be faster to implement an algorithm in software, there are a few compelling reasons for a move toward hardware implementation. They add that in consumer electronic devices, a hardware watermarking solution is often more economical because adding the watermarking component takes up a small dedicated area of silicon. In software, implementation requires the addition of a dedicated processor such as a DSP core that occupies considerably more area, consumes significantly more power, and may still not perform adequately fast [15].

In [14] the authors present a 0.18µm CMOS technology implementation of the Just Another Watermarking System (JAWS) embedder and detector. The author selected this watermarking algorithm because it is a well-known algorithm, because it works on raw video data allowing the author to concentrate on the watermark process and not on the compression issues, and because there was a previous implementation on a Trimedia TM-1000 VLIW DSP done before, useful work to compare their design [15]. The JAWS processes uncompressed real-time video. The authors claim that their work is the first step toward analyzing the relationship between watermarking algorithmic features and implementation cost for practical systems, and the first 0.18 micron implementation published at that time. The implementation features a pipelined architecture, and FFT and IFFT processing cores. The results show watermarking of video streams at a rate of 30 frames/sec and 320 x 320 pixels/frame. The chip is capable of operating at 75 MHz and process a peak pixel rate of over 3 MPixels/sec. The watermark is 4 bits/frame. The power consumption for the embedder is 60 mW and for the detector is 100 mW.

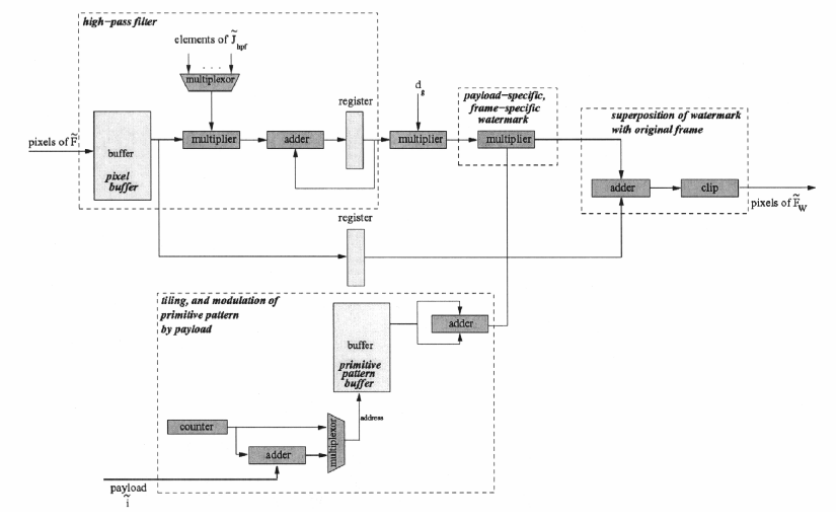


Figure 6 JAWS Embedder Implementation. Taken from [15]

In [13] the authors also agree on the lack of hardware implementations for digital watermarking. They make reference to 5 implementations, which include the two stated before. Their work is a VLSI implementation of two spatial domain visible watermarking schemes, one proposed by Braudaway, and the other by Mohanty for still digital images. The VLSI chip can insert either one or both watermarks depending on the requirement of the user. The proposed watermarking chip can be

integrated within any existing digital still camera. Their results show a chip with an area of 3.34 x 2.89 mm^2, capable of running at 292 MHz, and consuming 6.93 mW of power.

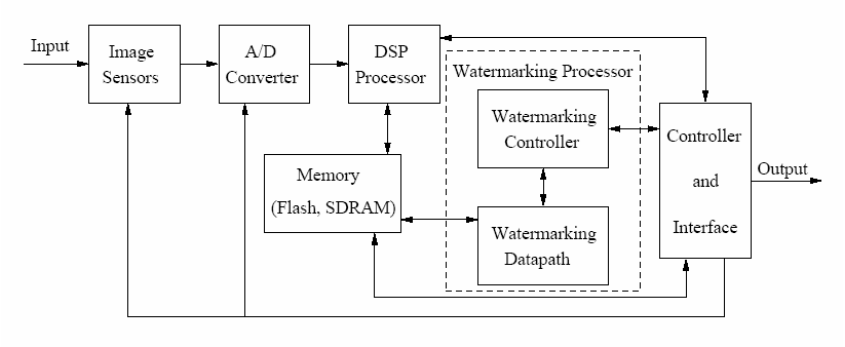


Figure 7 Diagram of the Mohanty VLSI implementation. Taken from [13]

An FPGA based implementation of an invisible, robust, spatial domain, still-image watermarking encoder is presented in [12]. The authors state that a hardware based watermarking system can be designed on a field programmable gate array (FPGA) board, a TriMedia DSP, or a custom integrated circuit. Seeming as the more practical choices, the selection reduces to choosing an FPGA or an integrated circuit implementation. The selection is a trade-off between development cost, power consumption, and performance. The FPGA implementation was explored. The watermarking encoder chip developed by them consists of a watermark generator, watermark insertion module, and a controller. The invisible watermarking algorithm implemented by the authors inserts pseudorandom numbers to host data. Synthesis was performed with SynplifyPro, simulations were run with ModelSim software and the FPGA device used was the Xilinx Virtex2 XCV50, which can be operated at 50MHz frequency. The results show an execution time of 19.842 ns for the overall process. The low power-high performance part of their project is still under progress.

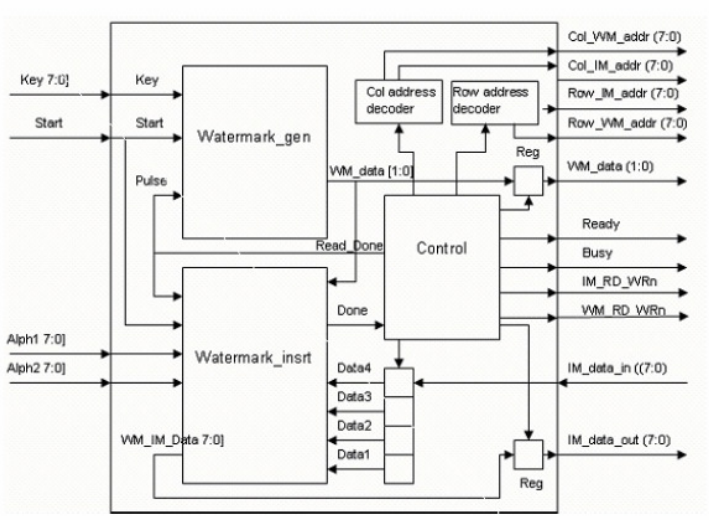


Figure 8 Block Diagram of the Mohanty FPGA Implementation. Taken from [12]

2.5. Watermarking DSP Implementations

A search in the literature for DSP implementations of image or video watermarking returns only one paper. In [60] the authors implement a real-time invisible digital video watermark method on a TriMedia TM-1000 by Philips Semiconductors. The TriMedia is a media processor, similar to a DSP, that possess a Very Long Instruction Word (VLIW) architecture and Single Instruction Multiple Data (SIMD) instructions for parallel and vector processing, and a 100 MHz clock. The method works on the spatial domain for ease of processing, and embeds a pseudo-noise signal to the video pixels. The watermark gain factor is not fixed and is content dependent. The media processor performs embedding and detection of the watermark. It was coded using the C language, due to the simplicity and the reduced time in coding, when compared to assembly programming. They are processing PAL standard video (50 Hz, 625 lines), representing a data rate of 414720 pixels per 40 ms. This translates to 9.6 clock cycles per pixel in order to achieve real-time processing. The achieved results show for the watermark embedding stage a processing speed of 5.64 clock cycles per pixel, representing a rate of 42.75 frames per second, well above the rate for real-time video.

2.6. Video Watermarking for Surveillance Applications

Previous work in digital watermarking of surveillance video is not abundant. A significant fraction of the research on digital video surveillance deals with Digital Signature techniques, such as [16] and [17]. This is due to their ease of implementation and lesser complexity. Most of them only cover the requirements and limitations, without offering a specific solution. Usually, watermarking techniques for authentication and tampering are borrowed and adapted to the video surveillance profile. Examples are presented next as background and as further specifications of what digital watermarking in surveillance video involve.

One method takes an arbitrary number of pixels in an image, and performs a checksum which is later embedded in the LSBs of selected pixels [18]. The method is fast and simple but not resistant to compression. Another method takes hashes of selected blocks in the image and stores them in a separate file [19]. This is a very popular technique in authentication because it provides great security and good localization properties, but actually does not perform a watermarking operation. Saving the authentication data independent from the data itself is also not convenient. The work developed by Fridrich [20] proposes a method to retrieve the authenticated data by self-embedding a highly compressed version of the image into the LSBs of the pixels. While this seems as an excellent method to employ, it is not resistant to compression. Another work from Fridrich [21] describes a technique capable of distinguishing malicious processing from innocent ones. A robust watermark is inserted in medium-size blocks using a spread-spectrum technique. If the watermark is present in all blocks using a certain threshold value, there has not been any tampering. If the watermark detection is below the threshold in all blocks, then some innocent processing was performed. If the detection is above the threshold in some blocks and below the threshold in some other blocks, then one can conclude that probably there was some form of tampering in the image. This technique uses correlation to detect the watermark, and the watermark is a robust one, which is not the norm on authentication and tamper detection.

Besides tamper detection, the ability to localize the place where the changes were performed on the image is of great desire. Usually the ability to localize is performed with changes to the LSBs of the pixels, as seen before. This makes the watermarking technique too sensitive, and not resistant to many innocuous and common processing such as lossy compression, contrast adjustment and filtering that maybe required in the system. Another work from Fridrich [22] addresses this conflicting issue by

superimposing two watermarks. One robust watermark is embedded into the image while a second, fragile, watermark is embedded on top of the first one. According to the author, the fragile watermark is usually very weak and should not influence the robust one in any significant manner [22]. The robust watermark is added to the middle 30% DCT coefficients of a 64x64 block. The fragile watermark is a binary image embedded into the pixels. The authors declare that the proposed technique can be extended to digital video by using a frame dependent key to generate the robust watermark. The only problem found with this work is that after compression the fragile watermark is gone, thus it is only useful if the compression is left to be done at the final stages of the system.

In [23] a similar approach is followed where a robust watermark is embedded to provide ownership proof, and a fragile watermark based on a digital signature is embedded to provide authentication. The robust watermark is embedded into the middle frequency coefficients of an 8x8 block DCT calculation. The 8x8 DCT was chosen because of the popularity of the JPEG compression format. The digital signature is generated according to the edges of the image. This information will not change after lossy compression, thus it will resist compression but will not resist intentional modifications to the image.

In [24] a semi-fragile watermark is also embedded into an 8x8 DCT block for tamper detection in compressed digital video. The authors mention that this scheme is valid for DVR and security systems. In this case the watermark is embedded in two different ways, according to the number of non-zero-quantized AC DCT coefficients (NQAC) in the block. If the number of NQACs in the 8x8 block is above a determined threshold, one embedding method is used, otherwise another method is used. This avoids embedding the watermark in zero-quantized DCT coefficients. Tampering can be localized.

The work in [25] also addresses the problem of authentication of digital still images for courtroom evidence, insurance claims, and journalistic photography. They do so by using watermarking in the discrete wavelet compressed domain. Information about frequency modification allows for selective tamper detection, according to the application. Wavelet compression was selected over DCT compression due to its better localization properties.

In [26] the authors argue that using still-image watermarking techniques for video is not efficient and may be prone to attacks due to the large amount of images with the same watermark. Their work is centered on watermarking compressed data, and they review some works on

watermarking on the DCT coefficients, run-length codes, and motion vectors. They also realize that although watermarking can be done by different methods it is usually intensive for the CPU and some design tradeoffs must be taken. These tradeoffs come as degree of robustness, complexity and other performance metrics. The authors settle for embedding the watermark in 8x8 DCT blocks due to their robustness and computational flexibility, compared to other watermarking methods. The key is embedding the watermark in only some selected 8x8 blocks of the frame, thus helping for real-time processing. One hundred (100) or more blocks must be marked for correct detection. The watermark embedding follows a human visual system approach to avoid artifacts after watermarking the image.

A work by Checcacci, et. al. [27] focuses on watermarking compressed video data that will be transmitted by a wireless channel.

In [28] the problem of digital video's value in front of a court of law is also presented. The need for a mechanism that can prove that the video stream presented is indeed the video stream captured by the camera, without any modifications, moves the authors to present watermarking as a possible solution. They provide a good summary of requirements for a digital video surveillance and watermarking system applied to surveillance systems. According to the authors, digital video authentication in the context of surveillance applications proves that any or all of the following facts are true:

- No frame in the video stream has been modified.
- No new frame was inserted in the video stream.
- No frame was removed from the video stream.
- The video being viewed actually came from the indicated camera.
- Every manipulation is detectable at its location, for legal liability issues.
- Important metadata (source, date, time, serial number, etc.) can be retrieved even after editing (compression, conversion, etc.) which guarantees authenticity of the data

Authenticity and integrity must be provable even if only parts of the video stream are taken. The impossibility to recover the tampered watermarking is taken as evidence that data has been tampered. Although watermarking is a feasible solution, many systems use digital signatures embedded in header fields instead of using watermarks.

In [29] a key-based encryption watermark is proposed, where the watermark is a low-quality version of the data. This allows not only the detection of areas that have been tampered or damaged, but also to recover some of the missing information. The "acquisition-transmission-storage chain" is presented as the process that all video data goes through. Authentication must be done as early as possible in that chain – in the camera. If the bandwidth available can not support uncompressed content, the content could be sent to a PC station to be compressed. If the signature of data must be performed inside the camera, watermark embedding should not be too demanding. A typical watermark-based authentication system for AVS applications is shown below.

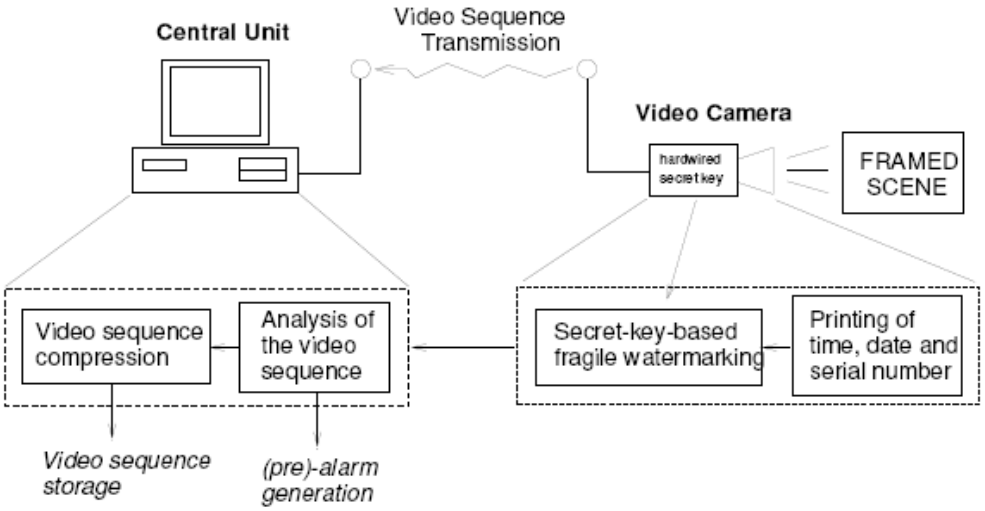


Figure 9 Watermarking Authentication System for AVS data. Taken from [29]

In [56], a Discrete Cosine Transform (DCT) is used to create the watermark, just as in [41]. They encounter a problem with surveillance video because it is typically quite static, and many DCT blocks have many zero coefficients. The authors state that embedding watermark into the zero blocks may significantly increase the bit-rate, but using as few blocks as possible will make visible the watermark. Two methods are proposed to deal with the watermarking problem. The first method divides the image in blocks, and performs a DCT in each of the blocks. The watermark will be embedded only in the blocks that have the highest DC coefficients. This solution minimizes bit-rate. The other method performs a DCT on the whole image, and embeds the watermark on the nonzero DCT coefficients after quantization. This is based on the Human Visual System. This solution minimizes watermark visibility. Conclusion shows that the first method offers a good compromise

between bit-rate and perceptual quality using an appropriate ratio, and method number two should be used to get minimal perceptual distortion if storage is not the major concern. Compression was done by using H.263 version 2 (H.263+) compression format and the "hall monitor" video sequence (300 frames – 352 x 288) was used as the test sequence.

The work in [30] provides more insight into surveillance watermarking. Some of the findings presented in this paper are:

- A variety of compression methods are in use in surveillance systems, including both spatio-temporal (MPEG) and still-image compression (JPEG).
- Where still-image compression is applied, compression in the temporal direction is achieved by retaining, for example, only one image every 5 seconds.
- Guideline for tampering detection is the minimum size at which a human face is recognizable.
- In semi-fragile watermarking security can be improved by generating the authentication bits such that they are dependent upon the image content.
- Compute a Hash of the image and embed it as a watermark.

In [31] the authors propose a watermarking method that processes 640x480 pixel still-images in the luminance component, using a PN sequence as the watermark on a 100,000 gate FPGA. They propose to use this FPGA implementation inside a digital camera. They achieve a run time of 1.28 seconds per image with 45% logic utilization.

3 SECURE VIDEO CAMERAS

The case study application selected for the investigation is a surveillance video camera system. In order to better present the description and the justifications of the selection, a brief overview of surveillance systems will be provided next, followed by a detailed description of the system, and previous works in the area.

3.1 Surveillance Systems

A surveillance system is a system that monitors the behavior of people, objects or processes to see if they comply with an expected behavior. This is basically done for security control. Although video surveillance is the most recognized type of surveillance system, there exist other forms of surveillance. From [2] we know that a surveillance system usually is comprised of a video camera or other technological device mounted in a high, and typically inaccessible, location where the device "observes" the activities below. Many of these systems can be found in places where a high degree of security is needed. For example, you will probably find one of these systems in banks, government offices, postal offices, airports, or places that control the way in. Other uses for surveillance systems can be found in traffic control, automated intelligent systems, telephone systems for security purposes, and so on. Video Surveillance systems are normally associated with reducing criminal incidences. The United Kingdom is an example of a place where many of the public areas are under video surveillance for this purpose.

Over the last four years (2001 to 2005), the digital video surveillance market has shown an unprecedented increase. The authors in [32] state that the reasons for this sudden surge in interest on these systems can be linked to a response to the terrorist attacks on September 11, 2001, the rise of internet infrastructure and broadband connections, and the availability of high performance, low-cost video processors. The benefits provided by digital video, as the ease of processing, enrich the surveillance process, and present a new degree of features not found with analog systems. Ironically, in the field of surveillance, the up-side of digital video also seems to be its down-side. The ease of

processing that digital video offers also limits its authenticity. When digital imaging is considered for use in law enforcement, the question of its admissibility in court is often raised. Authentication has become a serious issue due to the relative ease in which digital images can be manipulated and altered. Unless there is a valid proof or evidence that the video is authentic and has not been intentionally tampered, surveillance video is of no use in a court of law. The proper use of the watermark on multimedia data has been proven to provide such legal proof.

3.2 Types of Surveillance Systems

Actual surveillance systems can be divided into three main groups: Analog Systems, Digital Video Recorder (DVR) Systems, and Smart/Intelligent Systems. Analog systems are the most affordable systems, and are composed of an analog video camera and a monitor. No processing is performed to the data. DVR systems are also composed of an analog video camera, but the output of the camera is then converted to digital data using an Analog to Digital (A/D) capture card. The digital data is then received by a PC system with an Image Processing suite to perform various processing on the data and it is later stored in the PC. These systems have been lately the most popular choice by the market because of the price and flexibility offered. The last group, the Smart/Intelligent systems, provides digital video cameras with embedded microcontrollers or digital signal processing chips that process the data on real time. The output can be provided in compressed format and the embedded chips maybe a Commercial off-the-shelf (COTS) DSP solution or customized ASIC, such as an MPEG codec chip. Internet Protocol (IP)-based cameras are included in this group, which can transmit the video directly to the internet and even wirelessly.

Some features found today in DVR and Smart systems include:

- Video stabilization: Removing the shaking enables digital recorders to store more, higher-quality video, and allows for compression tools to perform better.
- Video motion detection: DVR software tools can save considerable disk space by recording only when motion is detected, in addition to alerting security systems.
- Encryption: output data is encrypted for security reasons.
- Watermarking with timestamp: Watermarking for increased security and legal purposes.

- Compression: The compression methods mostly used are Wavelet, MPEG2, MPEG4, and MJPEG which is becoming the most popular option by offering increased compression rates.
- Control provided through the Internet or a mobile device.
- Alarm notification to e-mail, pager, cell phone or audible/visible alert.
- 25 to 30 frames per second (fps).
- Recording modes: around the clock, scheduled recording, motion detection, post-alarm.

The most common hardware solutions are Pentium-based solutions, for DVR systems, and TI TMS320C6000™ DSP platform solutions for Smart Systems [33]. Some examples of actual surveillance systems are in [34, 35, 36, 37, 38].

3.3 Case study Application

The conceptual case study application selected for the project is an autonomous secure or surveillance camera. The secure camera is expected to work autonomously in a proper environment suitable to the correct operation of the system. The camera will be a digital video camera that captures 640 x 480 (standard definition video – 24 to 30 fps) grayscale images. All of the processing regarding the watermarking process will take place inside the camera, leaving no space for unsecured data. The output of the camera will be a watermarked video which will be transmitted to a central unit, where it will be decoded, or authenticated, and then stored. The power for the camera will be supplied by a battery and the transmission of the video data to the central unit will be by wireless means. Therefore the camera will need to use minimal power for its operation, and will periodically transmit image sequences to the receptive or central unit. Absolutely no power or data cables should be attached to the camera, as it will be battery-power driven. To minimize the amount of data that will be transmitted, it is assumed that the video stream will go through a process of compression. The amount of compression will be determined by the end user and the degree of invisibility of the watermark embedded in the images. A pictorial description of the system is shown below.

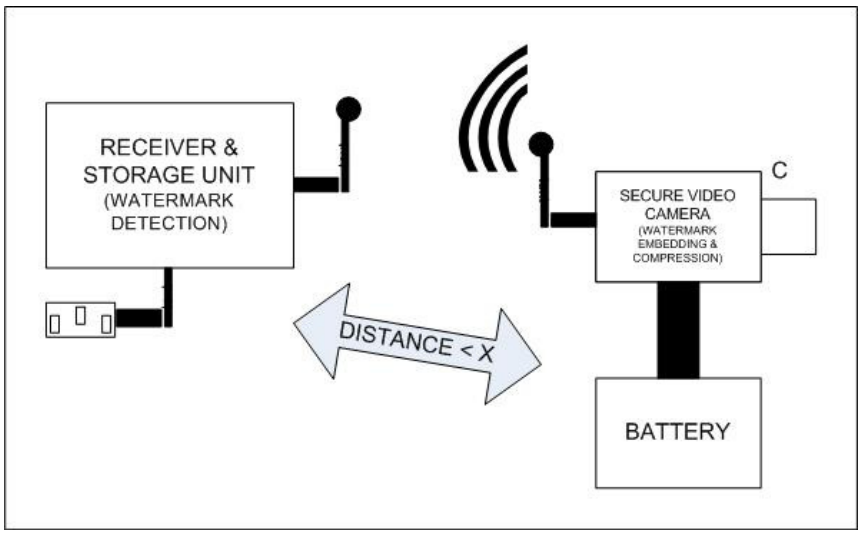


Figure 10 Case study application

Since watermarking is a practical, useful, and beneficial process in surveillance video, it seems as an ideal field to apply our efforts. Therefore the selection of the case study on the surveillance video watermarking field is based on the previously mentioned issues. At this point is worth pointing out that the case study application may currently be found commercially. However, the scope of this investigation encompasses the proper selection of a watermarking scheme and its hardware/software implementations, and how the proposed solution fits the problem defined by the case study, which will be detailed next.

The feature of being autonomous was selected because it provides greater value to a system of this nature, and a greater degree of flexibility. This allows installing such a system in remote places without the need of wiring installations, installing it in not so accessible locations, mobile locations such as vehicles, and several other places that are limited today by the available systems. All of this comes with the security of having an authentic signal, and without the fear of being prone to intentional or unintentional attacks such as the power line of the system being cut.

By being an autonomous system, it has to operate on its own and therefore all power must be supplied by a battery, which may or may not be connected to photovoltaic cell, or similar device. This feature poses stringent requirements on power consumption by the system, and a proper low-power watermarking architecture must be used. The battery will supply the power for the camera, the watermarking engine, the compression engine, and the wireless transmission module. As transmission

will consume considerable power, the transmitted data must be compressed in order to avoid draining the battery with unnecessary operations. Compression of the data will also be helpful for storage reasons, considering the large amount of space needed to store raw video. The transmission intervals should also be selected considering the power requirements.

Finally, after a review of the commercially-available surveillance products at the time of the development of this project, the 640 x 480 grayscale digital video camera was selected due to the large amount of products available with that specification, thus being a practical selection for out tests.

3.4 Requirements for Surveillance & Surveillance Watermarking

The requirements for a surveillance video watermarking scheme are:

- Watermark must identify the camera sending the video stream.
- Watermark must allow detecting if the video stream was altered.
- The watermark must be imperceptible and invisible to the human eye.
- Watermark embedding must occur in the same time of the current frame rate of the camera.
- Watermark will not degrade the quality of the content up to a certain degree or threshold, threshold that is lower in surveillance video than in other types of video. Surveillance video is interested in behavior and activity, and not in quality of image.
- Watermark must depend on the frame and be time-varying.
- Watermark must prove that no frame in the video has been modified, no new frame was inserted to the stream, no frame was removed from the stream and the video comes from the actual camera.
- Watermark detection must not require the original video.
- It must be difficult for non-authorized persons to forge a watermarked image or frame.
- The overhead induced by authentication calculations must be constrained to a minimum.
- Surveillance video may or may not be a real-time video stream. The frames per second sent can vary.

Recommended features for this type of watermarking scheme are:

- Watermark detection could also optionally detect where the tampering, if any, occurred. This helps to decide if the manipulation was intentional or non-intentional.
- Watermarking for surveillance video should be resistant only to compression. Since it is surveillance video, any other modification to the video should not be allowed.
- The watermark should include some sort of time stamp.
- The watermark should include the number of the frame sequence.
- Watermark should be of the noise type instead of the image or logo type, due to the large amount images found with the same watermark. This prevents a non-authorized party from easily forging the watermark by collusion attacks.
- If the signature of data must be performed inside the camera, watermark embedding should not be too demanding.
- The camera must be a secure environment to prevent attacks from the camera to the server, so embedding must be inside the camera.
- It should be possible to detect changes due to compression or random bit errors and make application-dependent decisions concerning whether or not the signal still has credibility.

Some of the requirements were gathered from the works in [25, 23, 29, 17, 39] among others.

3.5 Brief description of Compression schemes

Compression techniques can be divided into two main groups: lossless techniques and lossy techniques. The former compresses data without removing information from the source, while the latter relies on removing redundant information from the source to perform the compression. Decompressing lossless-compressed data will result in the exact same data that was available prior to compression, while decompressing lossy-compressed data will result in a similar, but not exact, copy. Lossless compression is popular in text-based data compression where modifying the data is not an option. Formats such as .zip and .gzip use lossless compression. Lossless compression can also be used for audio, images and video, but is not common. Usually they are used in combination with a lossy compression method.

Lossy compression is popular in media data, due to its high compression ratios based on the human visual and aural systems. Still Image compression uses spatial redundancy found in adjacent pixel values to remove information. Adjacent pixels are averaged out by using low-pass filtering, and high-frequency components are removed from the image. Video compression takes advantage of spatial and temporal redundancy found in video data in order to do compression. Spatial compression, or intra-frame compression, is basically the same type of compression performed on still images. Temporal compression, or inter-frame compression, is essentially removing the similarities between adjacent sets of video frames. Most video compression formats in use today apply a discrete cosine transform (DCT) for spatial compression [2]. Popular video and image compression formats employ a combination of lossy and lossless techniques to compress the data, as seen on the figure below.

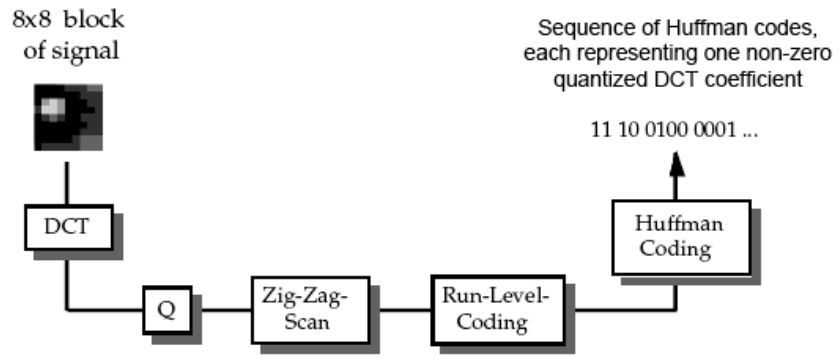


Figure 11 Simplified block diagram of JPEG compression. Taken from [40]

A popular format for lossy image compression is JPEG, which is based on DCT compression, and JPEG2000 that is based on Wavelet compression. Popular formats for lossy video compression are the MPEG-x (ISO standard) and H.26x formats (ITU-T standard), both based on DCT compression.

The following table resumes the features of some commonly used compression formats.

Table 2 Popular Compression Formats

COMPRESSION FORMAT	FEATURES
MPEG-4 part 10 (H.264)	• 33% improvement over MPEG2. • 4 times frame size of MPEG4 part 2 at a given

	data rate. • Targeted for all media applications: mobile, internet, standard video, high definition, and full high definition.
H.263	• Aimed at video coding for low bit rates (20 to 30 kbps or below). • Typically used for web video-conferencing (dial-up connections).
MPEG-2	• Outperforms MPEG1 at 3 Mbps • Below 1 Mbps, MPEG2 is similar to MPEG1. • Typically used for DVDs.
MPEG-1	• Aimed for 1.5M bit/sec data-rates and 352 x 240 resolutions. • Typically used for VCDs.
H.261	• Aimed at bit rates from 40 kbit/s to 2 Mbit/s. • Typically used in ISDN video-conferencing.
JPEG2000	• Based on the Wavelet transform • 100% improvement over JPEG • MJPEG2: video compression format based on JPEG2000 compression.
JPEG (DCT)	• Popular still image compression format • MJPEG: video compression format based on JPEG compression.

4 SELECTION OF VIDEO WATERMARKING ALGORITHM

The selection of the algorithm to be used for this work was a critical step. As part of this effort, after going through a number of papers and literature, more than thirty papers were selected as a first group to further analyze. This group was formed by possible candidate works, that were to be used individually or in conjunction with another work in the project. The methods used in these papers were compared with the requirements of the application and were classified according to the following aspects: type of media (video or image), compressed or uncompressed format, watermark done in spatial domain or frequency domain, transform method used (DCT, Wavelet, or Fourier), spread-spectrum techniques usage, type of watermark embedded (normally distributed, chaos, logo or image), secret key protection usage, robust to compression or fragile, and watermark localization characteristics. A table showing all the selected papers and their classification was developed in order to track the features of the watermarking methods and their corresponding papers.

For this implementation a video or a still-image watermarking method could be employed. Still-image watermarking could be done on a frame per frame basis of the video. Compression is a feature that must be present in the system; hence the algorithm must be robust to compression, or work on compressed data. Whereas spatial domain watermarking is much simpler, frequency domain watermarking is more robust, especially to compression attacks. The DCT transform does a better work of grouping energy and identify perceptually-significant areas in the image than other frequency transforms. Spread-spectrum techniques are popular because of the high degree of protection added to the watermark data. Therefore, a spread-spectrum watermarking technique would be desirable. The type of watermark itself usually can add more protection to the watermarking process by making it less detectable by unauthorized parties. Watermarks such as logos are more easily detectable than pseudo-random ones. The watermark should be fragile to other attacks besides compression so that it can detect if some tampering was performed on the data. Watermark localization properties add value to the process by providing the additional information of where in the image the tampering was performed. This information can be analyzed to obtain further information about the type of tamper attack performed.

A subset of 20 papers was selected then. The selection focused on papers from semi-fragile watermarking methods, detection without the original data, methods working with the DCT, and compressed video watermarking. From those 20 papers, 10 were further selected according to the following criteria: feasible hardware implementation, simplicity (in order to achieve low-power consumption), and robust to compression. Finally, five papers were selected as possible candidate papers, from which the selected watermarking method was selected. The selection criteria used for the selection of the final 20 and 10 papers will be explained and expanded in the following sections.

4.1 Simple, Low-Power, Hardware Implementation

The watermarking process can be divided into two main stages: encoding or embedding of the watermark, and decoding or detection of the watermark. As the embedding of the watermark and probably the compression would have to be done inside the camera, the algorithm for the process has to be simple so that power consumption is minimized. This implies using an algorithm that requires a small amount of processing, that does not employ advanced techniques in processing such as Human Visual System analysis and/or techniques based on analysis of the characteristics of the image, not using methods based on a lot of decision making, and that provides flexibility to build in top of, and to couple with compression. The algorithm should be capable of being run in real-time and that in turn helps to be implemented in hardware. The second stage of the watermarking process has less limitations because it will be executed on a full-fledge computer system.

4.2 Compression and Watermarking

As explained previously, most compression algorithms are based on the DCT transformation, which represents data on the frequency domain. Most robust watermarking techniques also rely on spatial-to-frequency transformation to achieve their robustness, especially by using the DCT and FFT transforms [22, 40, 41]. This robustness is usually with respect to compression, among others. Since the output data had to be compressed because of power and size issues, and there is a great similarity in the process of compression and embedding a robust watermark, it makes sense to consider both at the same time. By selecting a watermarking technique based on the DCT, resource sharing with, and

robustness to compression can be simultaneously achieved. A lot of research has already been done in trying to develop efficient and low-power-consuming implementations, because of the importance of the DCT to compression. Tying the watermarking scheme to compression raises another issue, since many of the watermarking methods that work on compressed video are strongly dependent on the format. This is very true for methods that work on MPEG-4 video. Since this work is not focused on a particular compression format, but on the process of watermarking, it is desirable to select a method that is independent of compression formats or standards.

4.2.1 Compression and the DCT

From the different compression methods available, the DCT compression mechanism was selected because of its wide acceptance in the digital video and image domain. Selecting the DCT for the application gives flexibility in terms of the final compression format to be used. Wavelet compression seems to provide better compression and localization information for watermarking, but lacks the flexibility of and the resources currently available for DCT compression. The various MPEG formats, the various H.26x formats, JPEG, Motion JPEG, and even proprietary formats based on DCT compression plus some sort of entropy encoding can be used in the future. As was seen earlier, many of today's system perform MPEG4, MPEG2 or H.263 compression on the data. Utilizing the DCT engine therefore provides a practical and efficient solution for our application.

4.3 Detection of Watermark without Original

Being a surveillance application, the original un-marked version of the video will not be available once it is watermarked. Therefore, a method of watermarking must be selected that does not require the original video at the moment of detection. Detection of the watermark must be done using some other means such as correlation.

Semi-fragile and fragile watermarks seemed like the best option. Semi-fragile watermarking is used for authentication and tamper detection. Fragile watermarking is widely used for tamper detection, because of its inherent fragility to attacks, thus providing proof of modifications in the data. But fragile watermarking can not determine if the attack is of malicious nature or not [22]. Thus, it

will not be resistant to compression. With careful design, semi-fragile watermarks can make that distinction and resist compression attacks, but operate on the pixel domain and require high-precision floating-point operations. Another issue encountered was the high complexity of the semi-fragile algorithms which made them unappealing to a hardware implementation for a low-power application.

4.4 Final Selection

The selected work was from Hartung and Girod [40, 42, 43, 44, 45, 46], one of the most influential works in video watermarking. The work borrows from spread spectrum communications, and consists of an adding an encrypted, pseudo-noise signal to the video data that is invisible, statistically unobtrusive, and robust against manipulations. The method can be applied to uncompressed and compressed video in MPEG or similar format (MPEG-1, MPEG-2, MPEG-4, H.261, H.263). The watermark can be retrieved from the decoded video and without knowledge of the original, un-marked video. It also contemplates the problem found in modifying the DCT coefficients for MPEG video, by adding a compensation drift signal. The method is robust and of much lower complexity than a complete decoding process followed by watermarking in the pixel domain and re-encoding.

The watermarking method for uncompressed video will be described. In order to describe the method we will fist define a_j as a sequence of watermark bits that will be embedded into the video stream. a_j is further defined as:

$$a_j \in \{-1,1\} \quad \textbf{(1)}$$

where j is any integer number. Note that the watermark is defined as a binary valued sequence. The selection of the binary values -1 and 1 comes from the watermark detection method. The watermark signal, a_j, is then spread by a large factor called the chip rate, cr:

$$b_i = a_j, j \cdot cr \le i < (j+1) \cdot cr, i \in \mathrm{N} \quad \textbf{(2)}$$

where b_i is the resulting spread sequence having the value of the corresponding a_j value. This basically means that every watermark bit is replicated cr times, in order to add redundancy and therefore robustness for the detection. For example, if the watermark a is composed of only 3 bits: 1, -1 and 1, then $a_0 = 1$, $a_1 = -1$ and $a_2 = 1$. If cr is equal to 3 then every bit of the watermark is replicated 3 times for a total sequence length of 9. The resulting spread sequence b would be: $b_0 = 1$, $b_1 = 1$, $b_2 = 1$, $b_3 = -1$, $b_4 = -1$, $b_5 = -1$, $b_6 = 1$, $b_7 = 1$, and $b_8 = 1$.

The spread sequence b_i is then amplified by a variable amplitude factor $\alpha_i \geq 0$. In our past example if $\alpha = 5$ (fixed amplitude factor) then the resulting amplified and spread sequence would be: $a_i \cdot b_i = [5, 5, 5, -5, -5, -5, 5, 5, 5]$. Next, the amplified spread sequence is modulated by a binary pseudo-noise sequence, p_i defined as:

$$p_i \in \{-1,1\}, i \in \mathrm{N} \tag{3}$$

If having a pseudo-noise sequence defined as: $p_i = [1, -1, 1, -1, 1, -1, 1, -1, 1]$ then the modulated sequence would be: $a_i \cdot b_i \cdot p_i = [5, -5, 5, 5, -5, 5, 5, -5, 5]$. Finally, the modulated signal $a_i \cdot b_i \cdot p_i$ is added to the video signal v_i yielding the watermarked video signal:

$$\tilde{v}_i = v_i + \alpha_i \cdot b_i \cdot p_i, i \in \mathrm{N} \tag{4}$$

$$w_i = \alpha_i \cdot b_i \cdot p_i \tag{5}$$

Due to the noisy nature of the pseudo-noise signal p_i, the spread-spectrum watermark signal w_i is also a noise-like signal and thus difficult to detect, locate, and manipulate [42]. A visual description of the algorithm is found below.

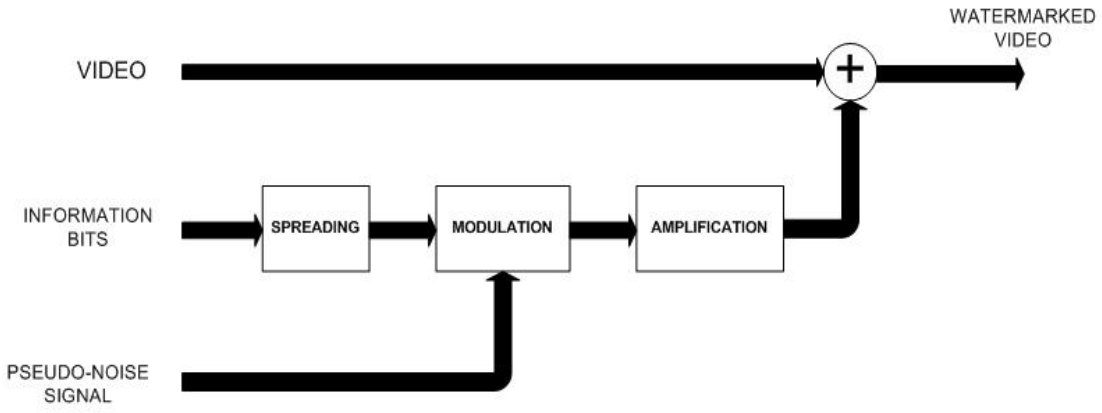


Figure 12 Watermark generation and addition to the video signal [42]

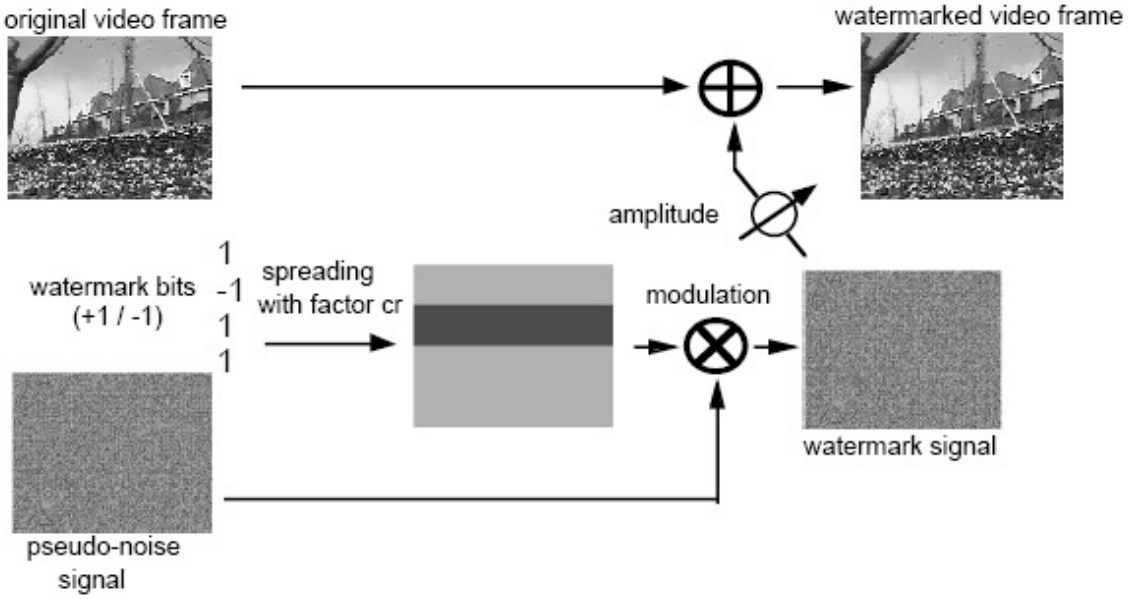


Figure 13 Watermark generation and addition to the video signal. Taken from [42]

The pseudo-noise sequence is defined as binary valued for simplicity, argue the authors, but a Gaussian, non-binary sequence can be used instead. The authors state that different pseudo-noise sequences are in general orthogonal to each other and do not significantly interfere, thus several watermarks can be superimposed [42]. The amplitude factor α_i can be varied according to the characteristics of the video and the application, making it possible for a Human Visual System method

implementation [1]. HVS methods, as previously mentioned, take advantage of the way the human eye processes images in order to add watermarks to the images. The authors state that with the high data rates found in video data, HVS is not so necessary and further advice to not use them in real-time systems due to their complexity.

The watermark process for compressed video is very similar to the uncompressed one. First of all, the authors state that the method is intended for MPEG-2 data because all international standards for video compression use the same basic scheme, motion prediction and block-based transform using the DCT. Thus, this allows applying their idea to similar compression formats. In order to better understand the method a brief overview of the MPEG-2 process is given.

An MPEG-2 video bit-stream is made up of three frame-encoded pictures. These are the I-frame, the P-frame and the B-frame pictures. Several I, P and B-frames form a Group of Pictures (GOP). I-frame pictures are intra-coding pictures, P-frame pictures are forward prediction pictures, and B-frames are bidirectional prediction pictures. The video image is separated into macroblocks, which are the basic unit of coding within a picture. Each macroblock is divided into four 8×8 blocks. A depiction of this classification is shown below.

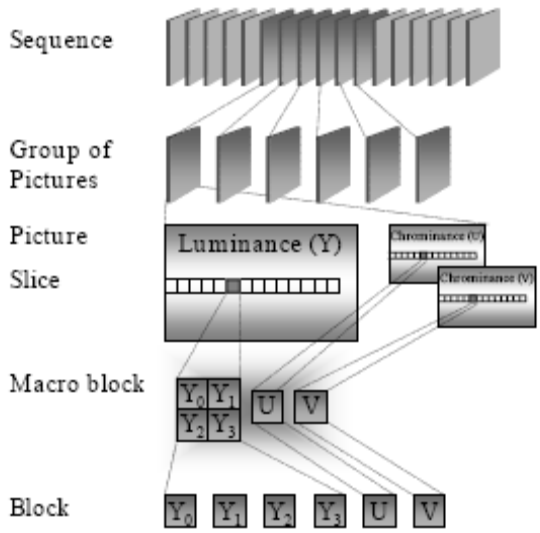


Figure 14 MPEG-2 syntax description. Taken from [47]

All 8x8 blocks from the three frames are transformed using the Discrete Cosine Transform. The resulting 64 DCT coefficients are quantized using a Quantization matrix, and re-ordered using a zigzag

run across the 8x8 block. The resulting vector is run-length encoded into (run, level) pairs and then entropy encoded using Huffman tables. Only the non-zero coefficients will be saved (see Figure 11). P-frames and B-frames result from a process of motion estimation where the encoder searches for similarities with the previous, and in the case of B-frames also the next, image in time order. The difference between the P-frame macroblock or B-frame macroblock and the reference frame macroblock is encoded as a motion vector. I-frames achieve their efficiency by exploiting spatial redundancy in images, because usually a pixel and its neighboring pixels are very similar in value. P-frames and B-frames achieve efficiency by taking advantage of temporal redundancy since there is usually small difference in adjacent frames. A typical sequence of I, P and B-frames is shown in Figure 15. P-frames may be 10% of the size of I-frames, and B-frames 2% of their size. The format also provides a header space to add information along the video file.

Figure 15 MPEG-2 Group of Pictures. Taken from [47]

Now, the way the algorithm for MPEG-2 compressed video works is:

1. Generate a spread-spectrum watermark as in the uncompressed video algorithm with the same size as the frame of the video.
2. For each 8x8 block of the watermark frame compute the DCT.
3. The compressed bit-stream is decoded and 8x8 DCT-quantized coefficient blocks are extracted. This block is then inversely quantized in order to obtain the DCT coefficient.
4. Each 8x8 block of the video is added to its corresponding 8x8 block of the watermark frame in the DCT domain with the following constraint: Only the DC coefficient and those coefficients which are non-zero will be added with the watermark.
5. The resulting 8x8 block is re-encoded and written back to the MPEG-2 bit-stream.

The following figure describes the method.

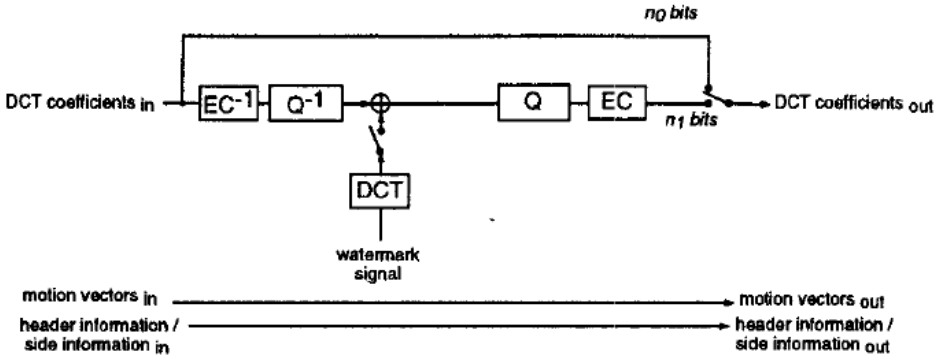


Figure 16 Watermarking in MPEG-2 domain. Taken from [42]

EC^{-1} represents the decoding and Q^{-1} the inverse quantization. As can be seen from the above figure, the motion vectors and header field are left untouched and the only modification is done to the DCT coefficients. One additional feature of the algorithm is that if the bit-ratio of the watermarked video stream must not be increased, the Huffman code-words can be compared before transmitting them. If the number of bits used for the watermarked codeword is bigger than the number of bits used for the un-marked version, the un-marked DCT coefficient is then transmitted. This is possible because of the redundancy introduced when replicating the watermark bits. This does not apply to the DC coefficient which is always watermarked. The authors affirm that around 10-20% of the DCT coefficients are altered, and that their method also complies with the human vision since only non-zero DCT coefficients are watermarked. If some area in the image has only low-frequency components, then only low-frequency components of the watermark will be embedded.

Detection of the watermark is straightforward and it is done in the spatial or pixel domain, for both uncompressed and compressed data. Once the video is in the pixel domain each frame is demodulated by multiplying the pseudo-noise watermark signal p_i by the video frame. Therefore, the pseudo-noise watermark signal has to be re-generated by the watermark detector. The authors state that the video could be high-pass filtered in order to eliminate the crosstalk between the video signal and the watermark, and improve the detection. After demodulation, all the pixels in the spreading window, given by *cr*, are added together and the resulting sign of the summation indicates the embedded information bit. Therefore, if the correlation between the watermarked video signal and the pseudo-noise signal is positive the transmitted or watermark bit was a +1. If the correlation is negative,

then the transmitted bit was -1. The complete derivation of the equation and the assumptions taken can be found in [42]. If the wrong pseudo-noise signal or a shifted version of the signal is used the scheme will not work. This makes the pseudo-noise signal the key for retrieving the watermark.

A good choice of parameters for this method is stated as cr = 2400, mean(α_i) = 3, variance of the pseudo-noise signal = 1 and using a 3x3 high-pass filter, resulting in a 528 byte/s watermark for NTSC video. The degree of robustness of the method is adjusted using the chip rate and amplification variables. Higher chip rates and amplification will provide more resistance to attacks such as filtering, addition of an offset, addition of white, colored or impulse noise, cropping, quantization in spatial or frequency domain, compression, and others. Higher chip rates translate into a smaller watermark word. Higher amplification translates into increased watermark visibility.

Another issue in the watermarking of compressed data is presented in the work. Since motion prediction relies on previous or subsequent data in order to generate the P and B frames, any degradation in the video sequence "will propagate in time and spread in space" [42]. This causes a significant problem, since adding a watermark can be seen as a degradation of the video. Data modified by the watermark will be used as reference, thus producing a defective prediction which is not real, and in top of that prediction the corresponding watermark which in turn accumulates the prediction-errors for the following frames. Since this topic is specific for MPEG implementations and this work is intended to be implementation independent, further discussion of this problem and the solutions presented by the authors are left to be read in [42].

4.5 Watermarking Tests

Different tests were performed on several images in order to observe the performance of the watermark algorithm. The test conditions were the following:

1. High pass filter, 97% quality compression, amplitude = 8, chip rate = 5,000
2. High pass filter, 97% quality compression, amplitude = 8, chip rate = 10,000
3. No filter, 97% quality compression, amplitude = 8, chip rate = 10,000
4. No filter, 95% quality compression, amplitude = 8, chip rate = 10,000
5. No filter, 90% quality compression, amplitude = 8, chip rate = 10,000

6. No filter, 90% quality compression, amplitude = 5, chip rate = 10,000
7. No filter, 95% quality compression, amplitude = 5, chip rate = 1,000

The compression quality table values are based on the freeware image viewer IrfanView's quantization matrices [48]. The tables below show the result for each test for every image tested. They are coded according to the number of the test. The first column identifies which image was watermarked. The second and third columns indicate how many watermark bits were embedded and how many of them were detected. The last column indicates the percentage of watermark bit detection. If the detection percentage is above a certain threshold we can say the watermark is detected, otherwise is not. The threshold is defined by the designer of the system who is utilizing the watermarking method, and by the application in which it is being used.

Table 3 Watermark Detection Tests

Watermark Detection Statistics									
	1	1	1	2	2	2	3	3	3
	# wm bits	# Detec-ted	% Detec-ted	# wm Bits	# Detec-ted	% Detec-ted	# wm bits	# Detec-ted	% Detec-ted
baboon.jpg	52	49	94	26	26	100	26	26	100
car.jpg	52	51	98	26	25	96	26	26	100
diving.jpg	78	69	88	39	35	90	39	39	100
fruit.jpg	78	61	78	39	37	95	39	38	97
girls.jpg	52	51	98	26	26	100	26	26	100
glass1.jpg	78	58	74	39	32	82	39	22	56
lenna.jpg	52	49	94	26	24	92	26	26	100
money.jpg	78	73	94	39	39	100	39	37	95
med-kitchen.jpg	61	45	74	30	25	83	30	29	97
P5306522.jpg	61	60	98	30	30	100	30	30	100

Watermark Detection Statistics									
	4	4	4	5	5	5	6	6	6
	# wm bits	# Detec-ted	% Detec-ted	# wm Bits	# Detec-ted	% Detec-ted	# wm bits	# Detec-ted	% Detec-ted
baboon.jpg	26	26	100	26	26	100	26	26	100
car.jpg	26	26	100	26	26	100	26	26	100
diving.jpg	39	39	100	39	39	100	39	39	100
fruit.jpg	39	38	97	39	39	100	39	39	100

girls.jpg	26	26	100	26	26	100	26	26	100
glass1.jpg	39	26	67	39	28	72	39	24	62
lenna.jpg	26	26	100	26	26	100	26	26	100
money.jpg	39	38	97	39	38	97	39	37	95
med-kitchen.jpg	30	27	90	30	30	100	30	27	90
P5306522.jpg	30	30	100	30	30	100	30	30	100

Watermark Detection Statistics			
	7	7	7
	# wm Bits	# Detec-ted	% Detec-ted
baboon.jpg	262	161	61
car.jpg	262	138	53
Diving.jpg	393	182	46
Fruit.jpg	393	123	31
girls.jpg	262	134	51
Glass1.jpg	393	90	23
Lenna.jpg	262	110	42
money.jpg	393	220	56
med-kitchen.jpg	307	108	35
P5306522.jpg	307	136	44

With 90% quality compression, amplitude of 8, a chip rate of 10,000, and a 60% threshold value, the watermark will be detected, according to the above test results.

4.6 Brief description of Compression schemes

The selected watermarking method complies with the requirements of being simple, being able to perform in real-time, uses the DCT for the process of watermarking, is robust to compression, can be done in compressed data, is of the noise-type, it is difficult to retrieve the watermark if a person does not know the pseudo-noise seed and sequence, can identify the camera using the camera id number as the watermark, is flexible in terms of the strength of the watermark, and detection of the watermark does not require the original video. So we can conclude that this method can be used for

the purposes of this work. In order to better implement it in hardware and to adjust the method to our case study application, some modifications to the watermarking process are done:

- Since we have first raw video that is to be compressed, the decoding and inverse quantization step of the algorithm can be discarded. The watermark can be added right after computing the 8x8 DCT and quantizing the block using the quantization matrix. The embedding of the watermark can not be done previous to quantization because we need to know the zero-valued coefficients.

- The compressed video data will not have to be reordered in any fashion because the corresponding video and watermark 8x8 blocks will be in the same respective position. Thus adding the watermark becomes just adding the two matrices.

- The verifying step for increased size of the video stream after watermarking will not be implemented. We do not have limitations of storage space or bandwidth.

- As explained before, the drift compensation part of the method is MPEG dependent. Therefore it will not be implemented as part of this work.

- The seed of the pseudo noise signal generator can be defined as the serial number of the frame, and the id number of the camera. A combination of the serial number of the frame plus the id number plus the time and date could also be possible. This would generate a different watermark every frame (time-varying), and creates a mechanism to know if a frame is missing in the sequence. The watermark decoder unit must know the order of the frames in advance.

5 FPGA AND DSP DESIGN

The hardware and software implementations will be described and explained next.

5.1 Hardware Implementation

For the hardware implementation several design practices were taken into account:

- Word length: The word size used has an impact on the power consumption of the hardware. Therefore, with a smaller word size, less power dissipation we will have. The word size should be limited to the smallest possible size in order to minimize power consumption.

- Frequency: CMOS-based circuits dissipate power only when a transition from one logic state to another logic state occurs (although recently this is not completely true due to higher sub-threshold leakage currents). This power consumption is governed by the following equations:

$$P_S \approx 0 \quad \textbf{(6)}$$

$$P_D = \alpha \cdot C \cdot V^2 \cdot f \quad \textbf{(7)}$$

$$P_{TOTAL} = P_S + P_D \quad \textbf{(8)}$$

Total power is determined by the sum of static power dissipation (6) and dynamic power dissipation (7). Static power is almost zero (for this analysis) and dynamic power is defined in (7), where α is the switching activity, or the percentage of time the circuit is switching, C is the total load capacitance, V^2 is the supply voltage squared, and f is the operational frequency. In sequential circuits f is regulated by the clock of the system, and the power consumption of

the system can be partially controlled by controlling *f*. A fast circuit dissipates more power than a slow circuit [49]. In FPGA-based implementations this is especially true due to the ability of the device to allow the designer to determine the clock frequency, and to design clock networks with various frequencies. For low-power design a lower *f* is better. In areas of the system where operating with a lower clock frequency is viable, the option should be considered.

- Switching: From the previous discussion about clock frequency it can be inferred that if no switching occurs in the circuit, no power is dissipated. Therefore switching should not be allowed in a section of the system when that section is not in use. This basically resumes to preventing the outputs of the circuits from changing, by using chip enable controls.

- Clock Gating: clock gating is a very popular low-power design technique that is based on the previous discussion of the switching principle. You can prevent circuit outputs from changing, but still the clock network will be switching and dissipating power although the circuits are not in use. Disconnecting the clock altogether will prevent not only the circuits from switching and dissipating power, but also clock network from dissipating power. This is usually applied to whole sections of the system that are turned off when not in use.

- Area: Usually less area means less power, because fewer circuits are needed to perform the work. More circuits mean more switching, and therefore more power dissipation. Lowering area typically means lowering performance in digital circuit design, since fast-performing circuits generally require more logic. Arithmetic functions that utilize less logic are preferred in this case. Area optimization techniques in the algorithmic description should also be explored.

- Fixed-point representation: Floating-point number representations allow a larger dynamic range and precision, but fixed-point representation is easier to implement, require less area, increases speed, and decreases power consumption. Therefore a fixed-point number representation should be used for a low-power implementation.

- Arithmetic circuits: Arithmetic circuits should be employed that reduce the number of operations performed in order to generate a result, and that their switching activity is lower

when compared to similar circuits. Methods that prevent doing operations that will result in an expected result or a value that will not be used should also be employed.

- Implementation in FPGA: using the manufacturer's provided logic functions for FPGA implementations usually guarantees efficiency in terms of logic utilization (area) and performance (speed) for that particular function. Typically the functions can be tailored to the specific needs of the system.

5.1.1 Field Programmable Gate Array

Field Programmable Gate Arrays, or FPGA's, are electronic devices that contain a number of programmable logic, programmable interconnects, and programmable I/O. This programmable logic can be used to implement any logic based on logic gates such as ANDs, ORs, and NOTs. Simple functions and complex functions, such as arithmetic elements can be performed in an FPGA. They also provide memory elements in the form of flip flops, or dedicated memory units. The high density of programmable logic blocks found in FPGA and their relative fast speeds have made them serious alternatives to Application Specific Integrated Circuits (ASIC's), microprocessors, and Digital Signal Processors (DSP's). Also, their ability to perform any digital circuit, their low time-to-market times, lower development costs, faster debugging, and the re-programmability are additional benefits over ASICs. An example of a simplified programmable logic block found in an FPGA can be seen below.

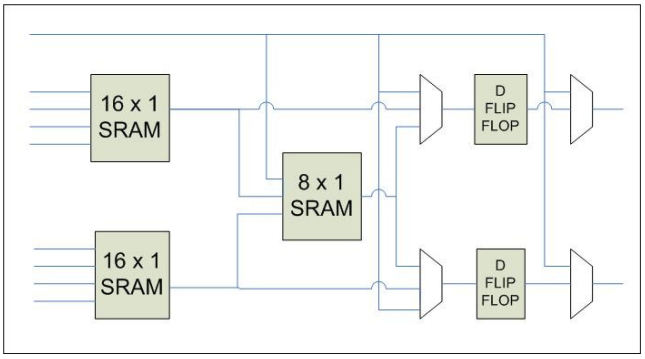


Figure 17 Simplified FPGA programmable logic block

As can be seen from the picture above, the basic building block for performing logic functions in FPGA are SRAMs, which are programmed as an n-input Look-Up Table (LUT).

The FPGA device used for this work is the Xilinx Spartan-3 XC3S200ft256-4. It has 200,000 logic gates which translate to 4,320 programmable logic blocks, or cells. Also has 216k bits of dedicated RAM, 12 18-bit-input dedicated multipliers, 4 Digital Clock Managers with clock skew control, and frequency synthesis and shifting and 173 I/O pins with 18 single-ended programmable standards, and 8 differential-pair programmable standards. Xilinx products allow the LUT SRAM's to be also used as distributed RAM, or shift-registers. The device is powered by a 1.32V internal voltage, a 3.00V auxiliary voltage and a 3.75V output driver supply voltage. The Xilinx Spartan-3 is marketed as being the lowest-cost device per logic and I/O, and as a device that can be used to develop high volume applications for a variety of markets including broadband access, home networking, display/projection and digital television equipment [50].

5.1.2 System Architecture

The top view of the designed logic unit is shown in Figure 18 with its inputs and outputs. It has 8 inputs, of 8 bits each, a reset input, a start input, and a clock input. A single pulse in the Start pin makes the unit start working. It has 8 outputs, of 12 bits each, a Ready output to signal external logic when the unit is about to read inputs, and a Valid Output pin to signal external logic that it may read the outputs.

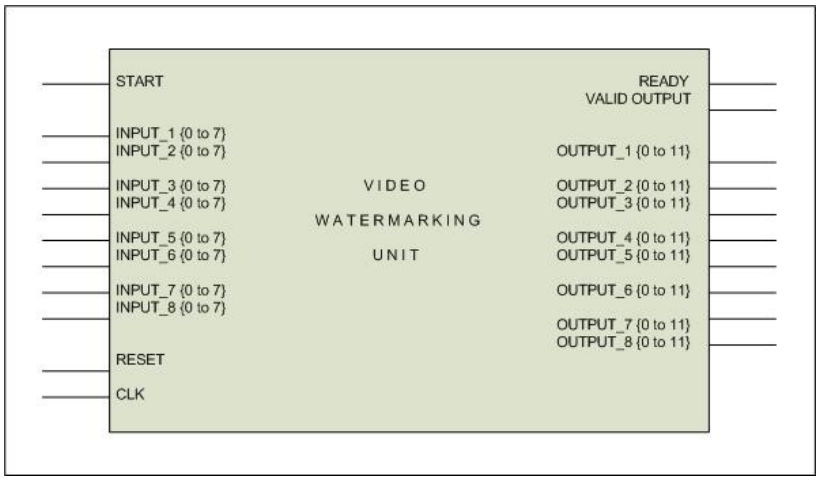


Figure 18 Top view of the proposed watermarking implementation

The system architecture found inside of the unit and developed for this work can be seen below.

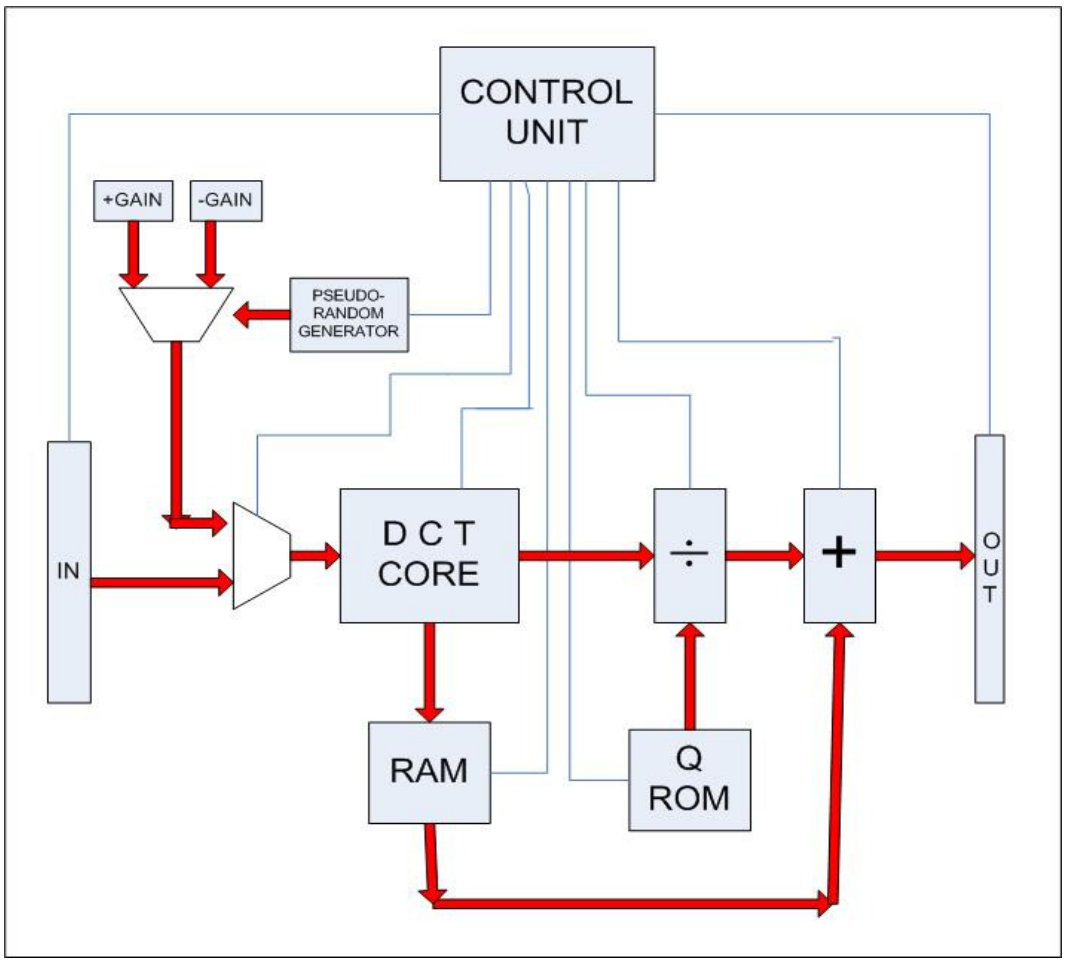


Figure 19 Architecture of the proposed watermarking system

The hardware implementation was developed using the VHDL language. The dataflow is described next. All the processing is done on an 8x8 pixel block basis. The input to the system is an 8x8 block and the output is an 8x8 block. The system has a pipelined design that processes 8 numbers in parallel. First, an 8x8 block of random numbers is generated. These random numbers are the pseudo-noise signal used for modulating the watermark. The length of the sequence of pseudo-random numbers is equal to the length of the spreading factor for every watermark bit. The spreading factor

was calculated to be a factor of 64 in order to fit in the 8x8 blocks completely. Once the 64 pseudo-random numbers are generated they are transformed using the DCT core and saved in a RAM. A block of pixels is now processed by the DCT core. The results of the pixel transformation are quantized using the quantization matrix values (Q ROM), and then the stored contents of the RAM are added to the quantized values, only if the values are non-zero values. The DC value is always watermarked. Since this is a pipelined design, the different stages of processing may overlap in time. For example, when the DCT coefficients of the pseudo-random numbers are being stored in the RAM, the DCT Core is already calculating the DCT of the pixels. A depiction of the dataflow just described is shown in Figure 20. Each step represents a block in the block diagram above.

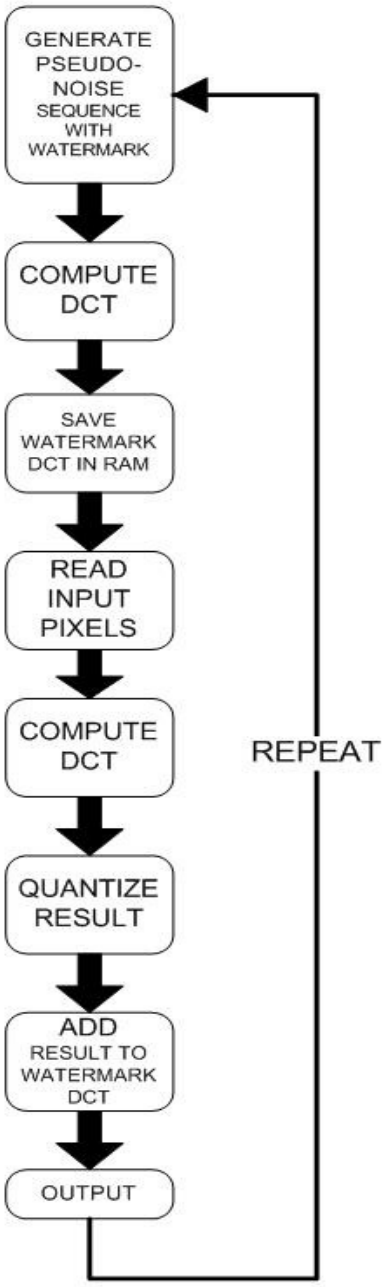


Figure 20 Dataflow description for the developed architecture

There are 4 major stages in the system. These are: the Pseudo-Random Generator stage, the DCT stage, the Quantization stage, and the Watermark Addition stage. The Control Unit controls the data flow across all four stages. In the image above the data flow is represented by the arrows.

5.1.2.1 Pseudo-Random Generator

The pseudo-random generator is implemented by using a Linear Feedback Shift Register (LFSR). The LFSR provides the random characteristics needed for the application [5, 12, 50]. The LFSR is 13-bit wide register, thus producing a 2^13-1 or 8,191-bit long sequence (see Figure 21). This modulates a total of ((2^13-1)/64), or 128, 8x8 blocks (one block not completely watermarked). If we are using a 480x640 pixel video frame, the available watermark bits are ((640x480)/64)/ 128, or 36, bits. Therefore, according to these specifications, a 36 bit watermark word is available for every frame. These specifications can be varied easily making the watermark word smaller for increased watermark robustness. The LFSR used was a Fibonacci LFSR with parallel load and chip enable generated using the Xilinx Core Generator. The modulation of blocks is controlled by counters. Eight binary values are generated consecutively, and these 8 values are then used to select the 8-bit amplified value from a multiplexer. Since the pseudo-random only generates 1's and 0's, we can think of the 0's as being -1's. The amplification factor multiplied by 1 or -1 will result in the amplification factor, or the negative of the amplification factor; therefore we can take advantage of that and avoid a multiplication step. The amplification factor, and its negative, is stored in a register and the proper value is selected by the output of the pseudo-random generator. These eight 8-bit amplified pseudo-random numbers are then passed on to the DCT stage, to compute their Discrete Cosine Transform. This process is repeated 8 times in order to generate a 8x8 block. The 13-bit initialization value of the LFSR represents the key that only the watermark decoder unit knows in order to retrieve the watermark from the video. If each video camera in the surveillance system contains a different key then we can identify the video stream from each camera when detecting the watermark. The Pseudo-Random Generator outputs one number every clock cycle. The 8 numbers are generated in 8 clock cycles; the 64 numbers of the 8x8 block are generated in 64 clock cycles, but take 64 + 16 cycles to be read by the DCT core due to its timing limitations.

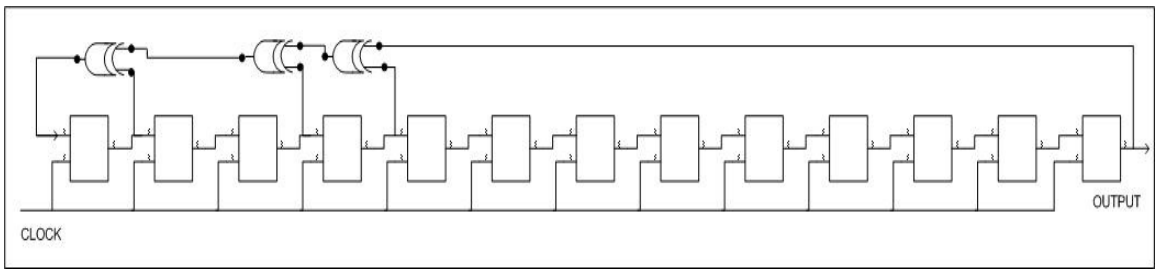


Figure 21 13-bit Linear Feedback Shift Register with Maximum Sequence Length

5.1.2.2 DCT Core

The DCT Core is the stage where the Discrete Cosine transform of a 8x8 block of numbers is computed. The DCT core is the main processing stage of the system and the most significant in terms of power consumption/reduction. The DCT core is composed of 3 stages: the first one-dimensional DCT block, a transpose matrix, and the second one-dimensional DCT block. The first and second one-dimensional DCT blocks are basically the same. They only differ by the word size they work with. The transpose matrix is an 8 rows by 8 columns RAM that basically transposes the values stored in the matrix. The implementation of the DCT core was based on a low-power implementation developed in [51], and the transpose matrix on [52]. The DCT core takes a set of 8 8-bit numbers at a time and outputs 8 12-bit numbers at a time. The latency of this unit is 140 clock cycles and its throughput is one output of 8 numbers every 12 clock cycles. Hence to compute a 8x8 block DCT it would take a total of 140 + (7 x 12), or 224 clock cycles. The DCT core will be discussed in more details later on. An image showing a simplified version of the DCT core is presented below.

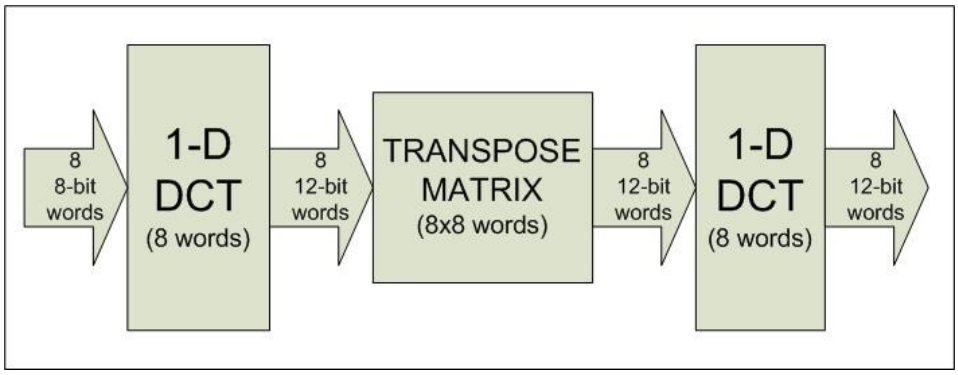


Figure 22 Simplified block diagram of DCT core

5.1.2.3 Quantization

Quantization is performed by dividing the 8x8 result block from the DCT core by the quantization matrix. The quantization matrix is a 8x8 matrix that is used to remove high-frequency components from the DCT coefficients and provide a compressed version of the original 8x8 block. The values inside the quantization matrix are defined by the application and can vary accordingly,

although JPEG provides standard quantization matrices [48]. The implementation of the quantization was performed by a signed number Pipelined Divider generated using the Xilinx Core Generator. The quantization matrix was stored in a 7-bit-word Block RAM configured as a Read Only Memory (ROM). This allows storing a divisor value of 0 to 127. The size of this ROM is 448 bits. Only the quotient of the division is used, and the remainder is discarded. For simplification, the fractional part of the fixed-point numbers is included in the word after the division. The divider has a latency of 8 clock cycles and a throughput of 1 result per clock cycle.

5.1.2.4 Watermark Addition

Watermark addition was implemented as a 12-bit adder that adds the contents of the RAM storing the DCT coefficients of the 8x8 pseudo-random blocks and the quantized DCT coefficient of the pixels. If the quantized DCT coefficient is 0.5 or greater, or -0.5 or lesser, the two values are added together. Otherwise no addition is performed and the output of that stage is 0. This complies with the quantization step by rounding values of 0.5 or greater to 1 and rounding the lesser values to 0. This decision was implemented by looking if all the integer bits of the word were 0's for a positive number (ORing), or all 1's for a negative number (ANDing). The adder adds one pair of number every clock cycle.

5.1.2.5 Control Unit

The control unit controls and synchronizes the overall process and the dataflow. The Control Unit is a state machine controlled by a counter that counts every clock cycle. The functions of the control unit are:

- Sends ready signals to other devices whenever the system is about to read input values.
- Starts and stops the Pseudo-Random Generator.
- Multiplexes data entering the DCT Core between random data and pixels.
- Starts and stops the DCT Core. Can start and stop each internal stage individually.
- Controls the read and write operations of the RAM for the DCT coefficients for pseudo-random numbers.
- Controls the read operations of the Quantization matrix ROM (Q ROM).
- Starts and stops the Quantization and Watermark Addition stages.

- Takes the serial output of the Watermark Addition stage and accommodates it into 8-word sets for parallel output.
- Sends "Valid Output" signals for other devices to read the output of the system.

Clock gating is provided for the Pseudo-Random Generator, the first 1-D DCT stage plus the transpose matrix, and the second 1-D DCT stage. Those units are turned off completely by the Control Unit, thus saving power. The Quantization stage (including the Quantization ROM) and the Watermark Addition stage are controlled by a chip enable, and they will not begin working until the DCT of the pixel block is ready. The outputs will not change for these stages and power will again be saved. The system will be idle after a reset until the start signal transitions from 0 to 1, and from 1 to 0. At that moment the Control Unit starts working and assumes that the video frame is in memory. Before receiving the start signal all units are either turned off or disabled.

The Xilinx Core Generator offers different types of dividers according to their throughput rate. The maximum throughput divider outputs 1 result per clock cycle. The minimum throughput divider outputs 1 result every 8 clock cycles. In terms of area, the highest throughput divider occupies more area than a lesser throughput one. Both dividers were used in these configurations: 1 1-clock-cycle divider, and 8 8-clock-cycles divider. The former occupied 34% slices, 28% slice flip flops, and 18% 4-input LUTs less than the latter. The former configuration was finally selected.

5.1.3 DCT Core Details

As mentioned before the details of the DCT core are found in [51], and [52]. A pictorial description is provided of the DCT Core and the Rom-and-Accumulate (RAC) structure for better understanding of the details that will be discussed next.

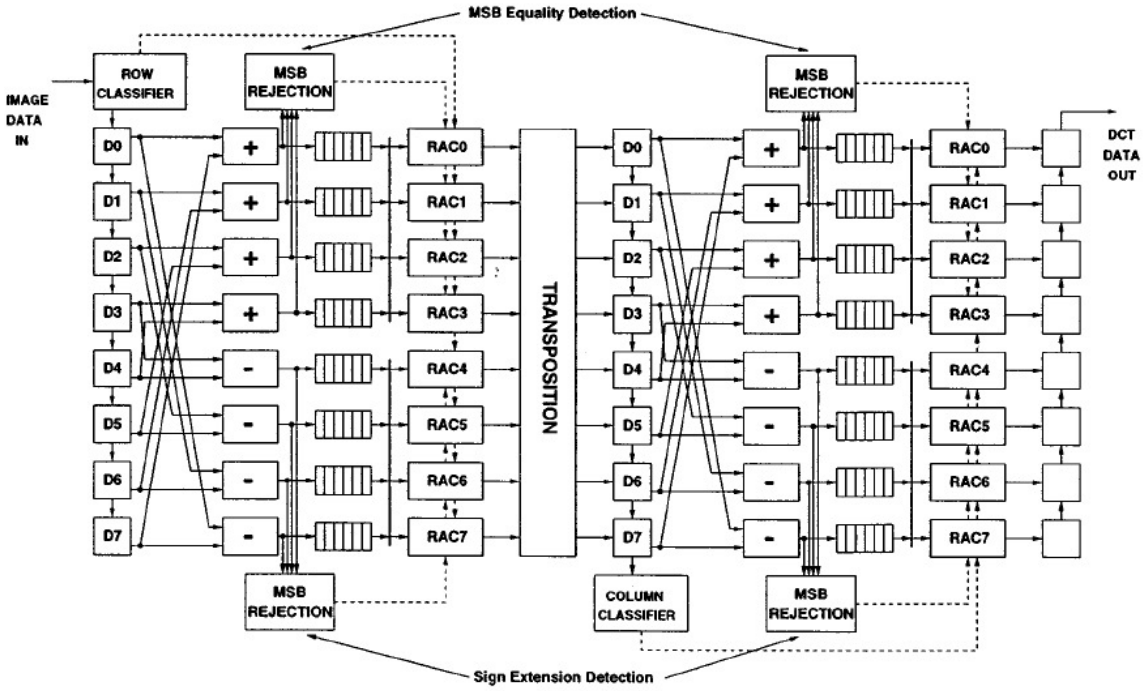


Figure 23 Block diagram of DCT Core, in detail. Taken from [51]

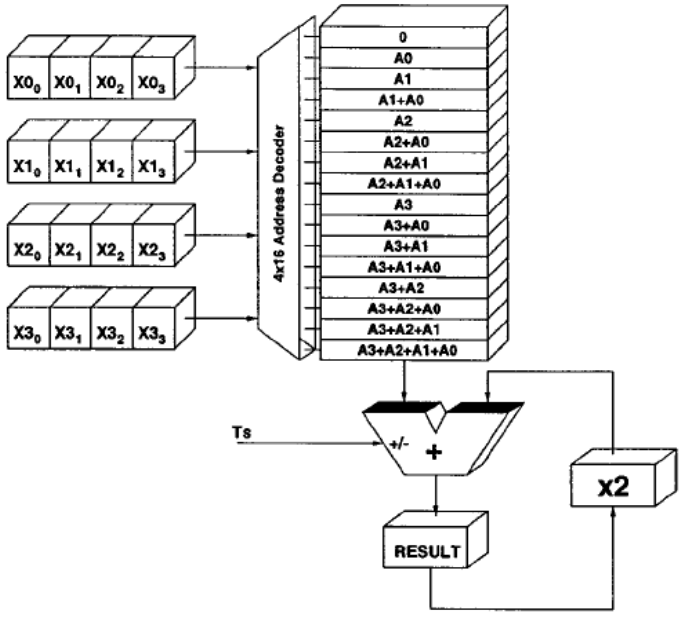


Figure 24 Block diagram of the RAC structure. Taken from [51]

The MSB Rejection and Row Classifier logic was not used in the implementation for simplification purposes.

Word sizes selected for the DCT core are shown in the following table.

Table 4 Selected Word Sizes for DCT Core

Word sizes for FIRST 1-D DCT stage				
	Word size bits	**Sign bits**	**Integer bits**	**Fractional bits**
Cosine table values	12	1	1	10
Input	8	0	8	0
Add/ Sub	10	1	9	0
RAC output	22	1	11	10
Output	12	1	11	0
Word sizes SECOND 1-D DCT stage				
	Word size bits	**Sign bits**	**Integer bits**	**Fractional bits**
Cosine table values	12	1	1	10
Input	12	1	11	0
Add/ Sub	13	1	12	0
RAC output	25	1	14	10
Output (DC coeff.)	12	1	11	0
Output (AC coeff.)	12	1	9	2

The cosine table value words were selected after reviewing the literature and doing experimental tests using MATLAB. These tests were used to see the resulting error of using a determined word size. The input to the DCT core is the word size of a grayscale pixel value. The word size of the output from the first 1-D DCT stage is taken from literature, as well as the output from the second 1-D DCT stage.

Although the first and second DCT stages seem similar, they are not. Besides the word size difference, the second stage has an additional control module that the 1st stage DCT module does not have. This control module basically buffers the data into the second stage due to the difference in word sizes between them. The first stage outputs data at a rate of 10 clock cycles, while the second stage inputs data at a rate of 12 clock cycles. The remaining of the logic is the same.

There is a drawback from this implementation. Outputs must be transposed and columns rearranged in order to provide the correct result. For this system we assume that the decoder unit can

manage that change in order, and therefore the outputs are left as they are. The quantization matrix was re-ordered in order to match the order of the DCT Core outputs.

5.1.4 Synthesis and Place & Route Results

After simulating and verifying the RTL design, synthesis and place & route for the Spartan-3 XC3S200 were performed. The optimization goal was set for Area, and the optimization effort was set to high. Register balancing and automatic extraction of logic such as multiplexers and RAMs/ROMs were other options selected. The Maximum Fan Out value was left at 500. No pin locking constraints were given. Effort level for the mapper and place & route tools were also set to high. Besides the clock constraint, no other constraint was fed to the implementation tool. Synthesis results are exposed next.

Table 5 Synthesis Report for the Watermarking System Implementation

CATEGORY	UTILIZATION	% OF UTILIZATION
Number of Slices	1892 out of 1920	98%
Number of Slice Flip Flops	1924 out of 3840	50%
Number of 4 input LUTs	2979 out of 3840	77%
Number of bonded IOBs	166 out of 173	95%
Number of BRAMs	11 out of 12	91%
Number of GCLKs	4 out of 8	50%
Minimum period	17.111ns (Maximum Frequency: 58.443MHz)	

A more detailed synthesis report, which fine points the logic used by every stage of the design, is shown in the following table.

Table 6 Detailed Synthesis Report for the Watermarking System Implementation

Stage	D-type flip-flops	Adder/ Subtractor	Multiplexer	Counter	ROM	Compara-tors

1st 1-D DCT	347	24	2	1	2	N/A
Transpose matrix	216	2	112	N/A	N/A	N/A
2nd 1-D DCT	628	26	11	1	2	2
Pseudo random gen	74	N/A	1	4	N/A	2
Control unit + watermark adder	253	5	45	3	N/A	10

Note that Block RAMs and Xilinx' Core Generator logic is not included here (divider and LFSR). This may be because that logic is already synthesized. Observing the distribution of logic, we can notice that the DCT Core utilizes most of the resources. This unit was synthesized alone in order to see the percentage of resource utilization. These were the results:

Table 7 DCT Core Synthesis

CATEGORY	UTILIZATION	%
Number of Slices	1334 out of 1920	69%
Number of Slice Flip Flops	1169 out of 3840	30%
Number of 4 input LUTs	2329 out of 3840	60%
Number of bonded IOBs	168 out of 173	97%
Number of BRAMs	8 out of 12	66%
Number of GCLKs	3 out of 8	37%

69% of the slices are occupied by DCT Core logic, and 66% of the Block RAMs. This makes clear that the DCT Core is the biggest unit of the device. The post-place and route results are described in the following table.

Table 8 Place & Route Report for the Watermarking System Implementation

CATEGORY	UTILIZATION	% OF UTILIZATION
Total Number Slice Registers	1,774 out of 3,840	46%
Number used as Flip Flops	1,773	N/A
Number used as Latches	1	N/A
Number of 4 input LUTs	2,804 out of 3,840	73%

Number of occupied Slices	1,823 out of 1,920	94%
Total Number 4 input LUTs	2,831 out of 3,840	73%
Number used as logic	2,804	N/A
Number used as a route-thru	25	N/A
Number used as Shift registers	2	N/A
Number of bonded IOBs	166 out of 173	95%
Number of Block RAMs	11 out of 12	91%
Number of GCLKs	4 out of 8	50%
Minimum period	16.301ns (Maximum frequency: 61.346MHz)	

5.2 Software Implementation

For the software implementation several design practices were taken into account:

- Avoid the use of functions. Calling functions introduces the overhead of saving registers and saving the current state of the processor, and restoring them when returning from the function.
- Avoid using loops if possible. Loops introduce overhead code.
- Avoid using confusing code. The simpler the code, the more control over what is being compiled you will have.
- Multiplications and Divisions can be performed by shift registers, if the denominator is a power of 2.
- Control instructions should be performed using logical operators because they are faster and simpler.
- Integer arithmetic is faster and simpler than floating-point arithmetic.
- Concentrate memory accesses in one point of the code. Memory accesses are slow.
- Try having frequently used operands in registers.
- Try to concentrate control code in one area, and processing code in another area to avoid cache misses.

These practices help the code to be simpler, and by being simpler the code becomes more power efficient.

5.2.1 Digital Signal Processor

The software implementation was done for the Texas Instruments TMS320C6713 Digital Signal Processor Developers Starting Kit (DSK). This DSP belongs to the C6000 family of high-performance DSPs. The C6713 is a Harvard Architecture processor with separate Data and Instruction memory spaces, and uses the VELOCITI architecture, an architecture derived from the Very Long Instruction Word (VLIW) architecture. The instruction fetch, instruction dispatch, and instruction decode units can deliver up to eight 32-bit instructions to the functional units every CPU clock cycle, one for each of its 8 execution units. The eight functional units are divided into two data paths, A and B. Each path has a unit for multiply operations (.M), for logical and arithmetic operations (.L), for branch, bit manipulation and arithmetic operations (.S), and for load/store and arithmetic operations (.D). Arithmetic operations such as add and subtract can be performed by all the units except the .M unit. The eight functional units consist of four floating/fixed-point ALUs (two .L and two .S), two fixed-point ALUs (.D), and two floating/fixed-point multipliers (.M). Two cross paths allow functional units from one path to access a 32-bit operand from the register file on the opposite side. Max to cross-paths source reads per cycle. The DSP includes two sets of register files, A and B, each set with 16 32-bit general-purpose registers.

Ideally, with the included 225 MHz clock of the 6713 DSK, two multiplies and accumulates per cycle can be computed for a total of 450 million multiplies and accumulates (MMACs) per second. With six of the eight functional units (not the .D) capable of handling floating point operations, it is possible to perform 1350 million floating-point operations per second (MFLOPS). This translates into 1800 million instructions per second (MIPS) [53].

The C6713 has a 32-bit, byte-addressable memory space. Internal (on-chip) memory is organized in separate data and program spaces. The C6713 internal program memory can also be used as a program cache. Direct memory access (DMA) controller, two 32- bit timers, power-down logic, serial ports, host ports and external memory interface (EMIF) peripherals come with the CPU. A block diagram of the C6713 DSP is shown below.

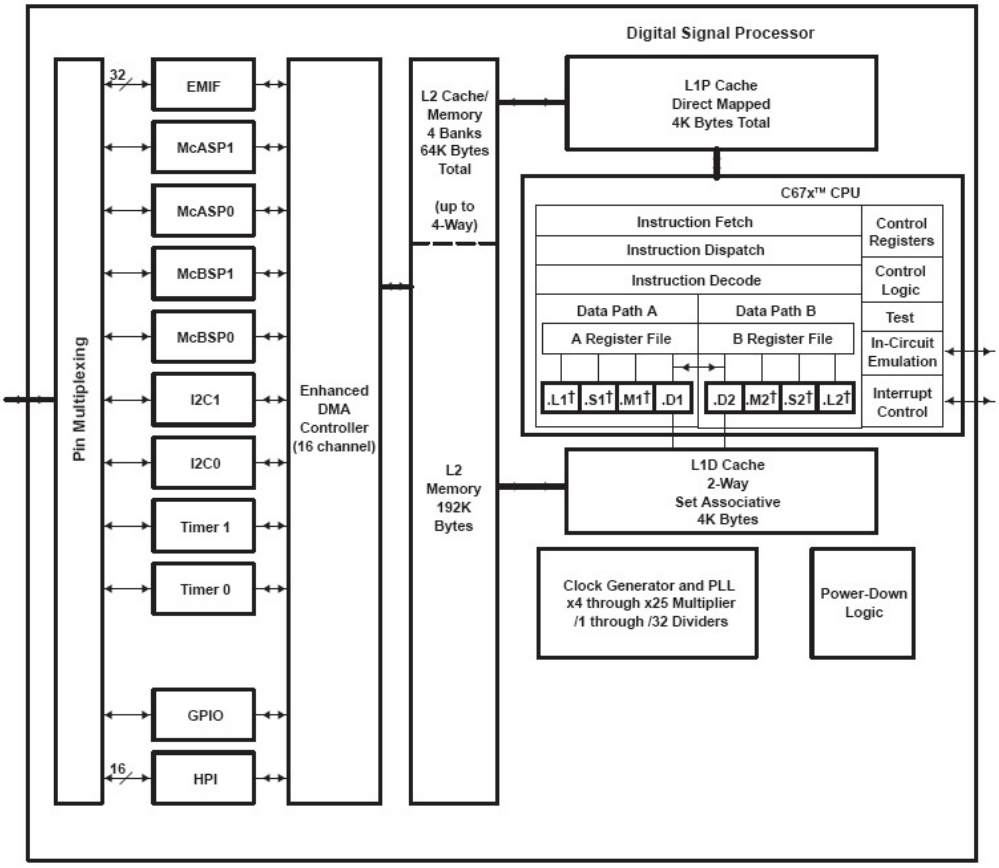


Figure 25 Block diagram of C6713 DSP. Taken from [33]

5.2.2 Flow Chart and Code description

A flow chart describing the code developed for the DSP is presented next.

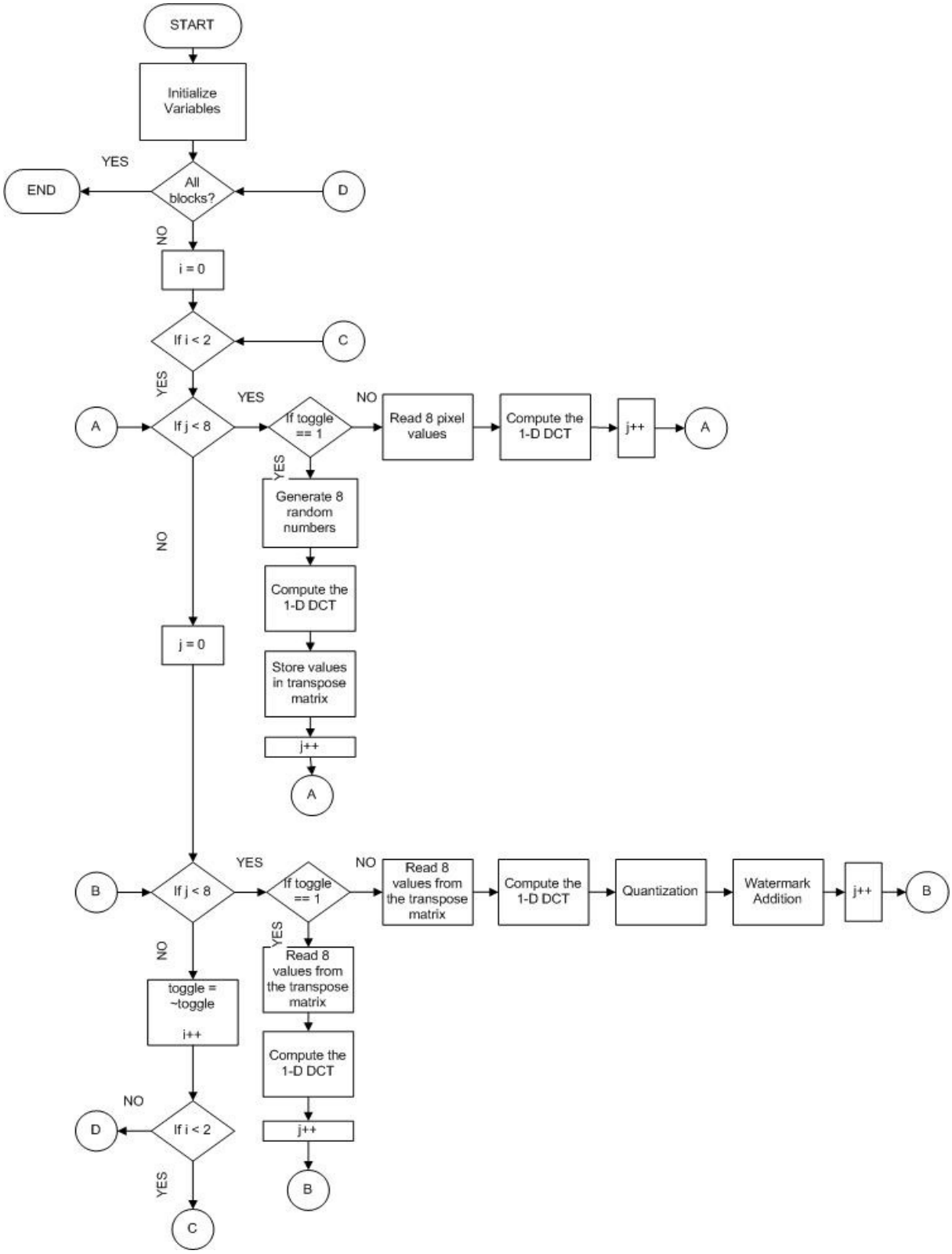


Figure 26 Flowchart of developed DSP code

The algorithm was implemented as a function in C. Although C code is less efficient (speed performance) than assembly code, it typically involves less coding effort than assembly code. But in recent years, the C compiler optimizer has become more and more efficient [53].

In order to have more control of the output of the optimizer, the code style was kept simple only using basic C constructs such as if, else, and for loops. Data types were defined as 8-bit chars, 16-bit short int, 32-bit int, unsigned, and const to help the optimizer save storage space and produce a more efficient output. The reasons to utilize a fixed-point number representation also apply to the software implementation, therefore only integer data types were employed. Using integer data types forces the DSP to only use fixed-point hardware, which is desired. The code represents the same algorithm implemented in hardware, but in this case the multiplications were performed by the DSP multiplier unit and the word size of the data was kept fixed at 32 bits. Since the word size is fixed, there is no reason to truncate the outputs of the 1-D DCT stages, although it can be done. The same values for the cosine table values were used. It is assumed that the function starts running when the image frame is already in memory and some sort of interrupt signal indicates the DSP to invoke the function (interrupt service routine).

The highest level of optimization (-o3) was used. For program level optimization the op3 (“No external variable referenced”) was selected. “Speed most critical” was the optimization goal selected. The memory model was “near calls and data” since all the data and instructions fit under 32K of memory space.

6 RESULTS AND CONCLUSIONS

In this chapter the results obtained from the implementations along with the methodology used to verify these results will be discussed. The conclusions of this work will be presented in the last part of the chapter.

6.1 Methodology for Verification

Ten pictures were selected to verify the two implementations. For each picture a full-frame DCT was performed to observe the frequency distribution of the image. Based on their frequency distribution, three images were selected for the test data: the image with the highest number of high-frequency components, the image with the lowest number of high-frequency components, and an image in between. The images are "baboon", "cars", and "lenna", respectively. From each picture three 8 x 8 blocks were extracted at random. These nine blocks, plus a randomly generated 8 x 8 block make up the ten data matrices used for verification of the implementations. The reader is referred to the Appendix section for a detailed description of the process, along with the images and the test blocks selected. The DCT calculation was performed using the *dct2( )* function in MATLAB. The random selection of blocks was performed using the *random( )* function in MATLAB. The count of frequency components is normalized to 1 by MATLAB.

The verification was performed for the fixed-point FPGA and DSP implementations using MATLAB results from a floating-point version of the same watermarking and DCT algorithms. This MATLAB version was first verified against the results of the dct2( ) MATLAB function from the image processing toolbox. The two sets of values (MATLAB results-FPGA results, MATLAB results-DSP results) were then compared. The amount of error found from the comparisons was quantified by using the Peak Signal to Noise Ratio (PSNR) term. This term is used in image processing to determine the fidelity of an image, and the amount of corruption found in it. Therefore, it is adequate to obtain a sense of how the results of the implementations compare and their degree of similarity. The formula for the PSNR is:

$$PSNR = 10\log_{10}(\frac{MAX_I}{MSE}) \quad \textbf{(9)}$$

where MAX is the maximum value that the elements in the image can take. For example, in a grayscale image, pixel values can have values from 0 to 255. Thus, in that case, MAX would be 255. The MSE term stands for Mean Squared Error, and is defined as:

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\left(I(i,j) - K(i,j)\right)^2 \quad \textbf{(10)}$$

where I and K are the two images of size m x n to be compared. As can be seen, the PSNR gives the error value proportionally to the size of the image. Usual PSNR values for compression are between 30 and 40 dB. Higher PSNR values represent better quality of the image.

6.2 DSP Verification Results

The output values from the DSP were gathered using the Code Composer Studio debugger and using Probe Points. The Probe Point was linked to a text file where the results from the DSP were automatically saved. The following table shows the PSNR values obtained for the 1st 8x8 block in each major step of the implemented algorithm.

Table 9 Verification Results for each stage in DSP implementation

DESCRIPTION	PSNR (MAX = 255) dB
1st 1-D stage DCT output of pixel block	75.23
1st 1-D stage DCT output of watermark block	85.01
2nd 1-D stage DCT output of pixel block	70.33
2nd 1-D stage DCT output of watermark block	81.85
Output after division by quantization matrix	65.86
Output after embedding watermark	72.70
Output after embedding watermark – compared with dct2( ) MATLAB function	72.70

As can be seen, the error found is very small. The following table presents the PSNR values for the output after embedding the watermark, for each of the test blocks.

Table 10 Verification Results for DSP output

BLOCK #	PSNR (dB) (MAX = 2^12)
1	84.76
2	82.87
3	86.75
4	85.42
5	56.68
6	83.90
7	84.03
8	83.32
9	87.68
10	81.25

6.3 FPGA Verification Results

The FPGA results were gathered after functional simulation using ModelSim, a functional and timing simulation tool for FPGA-based HDL code. The following table presents the PSNR values found for the RTL implementation. DCT + Q represent the output after the test block has gone through quantization. DCT + Q + wm represent the output after the test block has gone through watermark addition.

Table 11 Verification Results for ModelSim output

DESCRIPTION	PSNR (dB) (MAX = 2^12)
1st block, DCT	57.65
1st block, DCT + Q	58.78
1st block, DCT + Q + wm	53.47
2nd block, DCT	57.57
2nd block, DCT + Q	58.59
2nd block, DCT + Q + wm	52.87
3rd block, DCT	57.86
3rd block, DCT + Q	59.41

3rd block, DCT + Q + wm	53.99
4th block, DCT	58.95
4th block, DCT + Q	60.74
4th block, DCT + Q + wm	54.01
5th block, DCT	58.53
5th block, DCT + Q	60.28
5th block, DCT + Q + wm	53.35
6th block, DCT	57.73
6th block, DCT + Q	58.97
6th block, DCT + Q + wm	45.21
7th block, DCT	57.67
7th block, DCT + Q	59.38
7th block, DCT + Q + wm	47.06
8th block, DCT	58.66
8th block, DCT + Q	60.65
8th block, DCT + Q + wm	54.68
9th block, DCT	58.76
9th block, DCT + Q	60.22
9th block, DCT + Q + wm	47.15
10th block, DCT	57.43
10th block, DCT + Q	58.59
10th block, DCT + Q + wm	49.47

Note four details from the above table:

- PSNR values for DCT + Q are always higher than the other two. This tendency is contrary to the PSNR DSP results.
- Overall PSNR values for FPGA are in general 3 decades below those for the DSP.
- PSNR values for DCT + Q + wm blocks 6, 9 and 10 are considerably lower than the rest. Notice also that the DCT and DCT + Q values for 6, 9 and 10 are normal if compared with the rest of the blocks.
- The DCT + Q + wm (the final value) has always a lower PSNR value.

Errors in blocks 6, 7, 9, and 10 are due to quantization error. The watermark is embedded only if the quantized value is greater that or equal to 0.5, or lesser or equal to -0.5. Due to word quantization in the implementation, values that should be 0.5 or greater become less than 0.5, and vice versa. Thus the watermark gets embedding when not supposed to, and does not get embedded when supposed to. When analyzing the data on these blocks, one can find that the quantization error occurs only at one or

maybe two of the elements of the 8x8 matrix. Therefore, this error does not affect the watermarking process results because it is not introducing bogus data.

The DCT + Q PSNR values are higher than the corresponding DCT and DCT + Q + wm PSNR values probably because of the rounding step after the division (quantization stage). Least significant digits in the numbers are modified and rounded. This rounding probably eliminates some error from the results. The DCT + Q + wm values maybe lower because of the superposition of the previous quantization errors found in the other terms.

The 3 decades difference between the FPGA and DSP results may come as a result of the truncation of the word between the two 1-D DCT stages in the DCT core. That truncation of the word size is not performed in the DSP because the word size is fixed, and truncating it will not produce the power-saving benefits that this action provides on the FPGA version. Another truncation is performed at the end of the DCT that is not performed in the DSP, which may add to the error. Since the truncation occurs at the fractional bits the impact of this truncation is less significant.

6.4 Characterization of the devices

The two device implementations were characterized for performance (frames per second), power dissipation (average power), unit cost of the devices ($ per unit), and cost of development (man-hour). The details of the characterization methodologies used and the corresponding results can be found in the next sections.

6.4.1 DSP Performance

The performance of the DSP was measured using the Total Cycles information from the Code Composer Studio profiler tool. The device selected for the test was the C6713 Device Cycle Accurate Simulator. The Device Cycle Accurate Simulator models most of the peripherals of a device in a cycle-accurate manner, and can accurately count instruction cycles, including cycles for wait states and pipeline conflicts. Using this simulator makes certain that no aspects of the configuration and

features are missing. The program was set to run for 4800 blocks, which is the number of 8x8 blocks found in a 640x480 image. To make a fair comparison with the FPGA device, the same 8x8 block was used in order to avoid fetching information from memory, a step that was not included in the FPGA performance calculation. As a result, the performance value obtained would correspond to the best-case scenario. The summary for the profiler results are shown in the following table.

Table 12 Summary of Profiling activities for the C6713 DSP

CATEGORY	RESULTS
Total Clock Cycles	61,214,230
NOP Cycles	32.97%
STALL Cycles	9.23%
Instruction Cache hits	92.02%
Instruction Cache misses	7.98%
Data cache hits	99.98%
Data Cache misses	0.02%
Data Cache reads	60.87%
Data Cache writes	39.13%

If we take the total number of cycles and divided it by the clock frequency of the DSP, which is 225 MHz, we obtain the time it takes to watermark 1 frame. This number is 0.272 seconds. If we want to know how many frames can be watermarked in 1 second we find the inverse of the previous result, which is 3.68 frames per second.

6.4.2 FPGA Performance

Latency for the pipelined-FPGA design was determined to be 372 clock cycles. The first output value will appear after 372 clock cycles. After that initial output it takes 212 clock cycles to produce additional outputs. Therefore, the throughput of our design can be defined as 1 output value per 212 clock cycles. If we are using a video with N x M pixel resolution, the time it would take to watermark one frame of the video is defined by the following equation:

$$Tot.\_Cycles\_for\_1\_frame = 1xLatency + ((NxM)/(8x8) - 1)xThroughput \quad \textbf{(11)}$$

For our case in which we are using a 640 x 480 video, the number of cycles it would take to watermark one frame would be 1,017,760 clock cycles. The time it takes to watermark a frame can be obtained by multiplying the number of clock cycles by the period of the clock. The maximum clock frequency is 60 MHz according to the Xilinx ISE 7.1i tool; therefore the minimum clock period is 16.66 ns. This gives 0.01696 seconds per frame. Thus, the FPGA can watermark 58.96, or almost 59 frames per second at maximum clock frequency. This means that if we were to run the watermarking at real-time speed (30 fps), the FPGA would be roughly half-a-second working and half-a-second idle.

6.4.3 Power Dissipation of the DSP Implementation

The power dissipation of the DSP was measured using the method in [54]. The method basically sets the piece of code whose power consumption we want to measure to run in an infinite loop. The power consumed by the processor while this loop is running is the average power dissipation of the algorithm running in that processor. Since we did not have a direct access to the *Vcc* pins of the DSP an alternate method was employed. The method is shown in the image below.

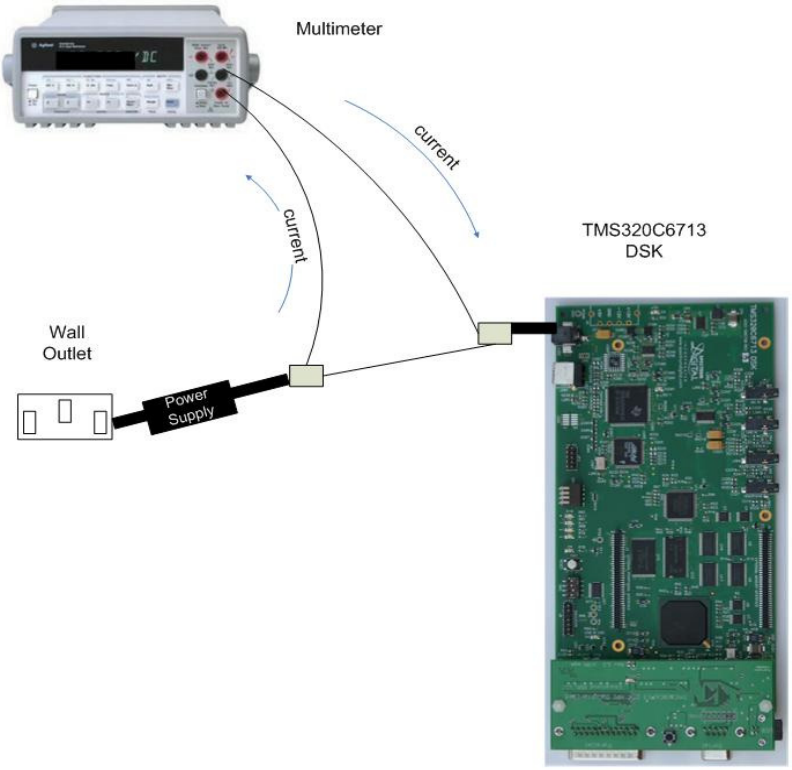


Figure 27 Configuration for measuring the power dissipation of the DSP device

This measures the average current of the whole evaluation board. Since we are only interested in measuring the power dissipation in the DSP, we set the DSP into a power-down mode. The C6713 has 6 power-down modes. We selected the mode with highest power-down activity, PWR_PD3. This mode basically performs clock gating on the core of the processor, the I/Os, the peripherals and turns off the Phase-Lock Loop (PLL) that generates the chip clock. The only way to awake the processor from this power-down mode is through a system reset, and then waiting for the PLL to lock again.

We measure the average current in both scenarios (with the DSP operating and the DSP on PWR_PD3 mode), using the Agilent 34401A digital multimeter, and we subtract the values. We then multiply this current difference by the supply voltage to obtain the power. We used the 10 8x8 test blocks to run in the infinite loop. The results are shown in the following table.

Table 13 Average Power Dissipation for the DSP

CATEGORY	POWER
Chip + board	403 mA
Chip in power-down + board	265 mA
Difference	138 mA
Supply voltage for the DSP	1.26V
Average Power Dissipation	173.88 mW

6.4.4 Power Dissipation of the FPGA Implementation

The power dissipation of the FPGA device was estimated using the XPower tool from Xilinx [50]. The input to the tool was actual data from simulation generated using Modelsim, and the circuit description file generated after the place and route process. We used a clock frequency of 30MHz, and an activity rate of 75% since the processing is intensive. A 100% activity rate means that the logic changes state on every clock tick. The results are distributed among the three voltage supplies the FPGA contains. The core voltage (Vccint) reported 25.81 mW of Dynamic dissipation and 24.00 mW of quiescent current power dissipation. The inputs/outputs voltage (Vcco) reported 2.69 mW of Dynamic dissipation and 0.00 mW of quiescent current power dissipation. Finally the auxiliary volatage, used for configuration of the device, Vccaux, reported 0.00 mW of Dynamic dissipation and 37.50 mW of quiescent current power dissipation. This adds up to a total power dissipation of 90 mW. The results were reported with a "Reasonable" confidence level. If we were to use a 1000 ma-Hour battery to supply the power to the FPGA, it would last for 17.37 hours according to the report.

6.4.5 Cost of Devices

The cost of 1000 units of the TMS320C6713 DSP is $26.40 per unit. The cost of 1000 units of the Spartan 3 XC3S200-FT256C FPGA is $21.29 per unit. The total cost difference between the two devices for the 1000 units is $5,110. The costs were acquired from the respective manufacturers' online stores.

6.4.6 Development Cost

The development cost in terms of man/hours is clearly much less for the DSP, in the case of developing an application such as this one. This assertion could probably be extended to any general application based on the products available for development. The development for the FPGA took around 3 to 3.5 months, compared to the 1 to 1.5 months spent developing the application for the DSP. Assuming 30 day months, and 8-hour work days, the FPGA implementation required around 720 to 840 man-hours. The DSP implementation only took around 240 to 360 man-hours. The development tools provided for the DSP are more helpful to the designer and include more features that FPGA development tools do not possess at this moment. Optimization tools for the DSP are more mature, can perform with less intervention from the designer, and provide feedback in order to help the designer focus on the areas that need the most attention. FPGA development requires careful attention to what is being coded in order to guide the synthesis tool to produce the expected circuit. Simulation and verification is an area where improvements are needed for the FPGA. For the FPGA, all the steps of development (architecture definition, implementation, verification) take more time, more effort, and skill.

6.5 Discussion of Results

When comparing the FPGA-based implementation with the DSP implementation, there is no doubt that the DSP is not an option for real-time video processing. We can see that 30% of the instructions executed in the DSP are NOP instructions. If we can reduce that number, the throughput rate should then increase. The expected results after the NOP rate reduction would not surpass the 4 or 5 frames per second throughput. To achieve real-time processing it would anyway require a DSP with almost 10 times the performance of the C6713. The highest frequency DSP available is the fixed-point TI TMSC64x DSP operating at 1 GHz and capable of performing 8000 million instructions per second (MIPS). We would still not make it since we need around 2.25 GHz or close to 18000 MIPS. Usually video hardware accelerators or ASICs are needed in order to handle the rigorous computational demands of real-time video processing.

In terms of the FPGA performance, we could decrease the clock speed in order to achieve power savings. If we are running at 30 frames per second we can decrease the clock frequency to around 30MHz. For lower frame rates, the clock frequency can be decreased accordingly. However, the FPGA-based implementation has power controls that can turn off the logic if the processing of the data has been completed. Therefore if the FPGA is idle it should not consume the same amount of power. If all the logic is turned off (theoretically), the maximum that it can consume would be due to the maximum leakage current. For the XC3S200-FT256C this power is defined as 37.5 mW.

In terms of power dissipation the FPGA dissipates approximately 90 mW versus the 173.88 mW dissipated by the DSP device. This is almost half the power dissipated by the DSP. The FPGA seems as the better option.

Considering device cost, the FPGA also seems as a better option offering lower prices per unit. Since the C6713 DSP can not perform in a real-time video scenario another device must be selected, unless the application runs at 3 frames per second. This implies higher costs, since we need higher performance. For example the C64x prices vary from $250 to $350 depending on the device selected. This is at least 10 times more the price of the C6713. On the other hand, if we are settling on 3 fps the cost difference would still be $5,000, and probably bigger, since a much smaller and cheaper FPGA device could then be selected.

In terms of development cost, the DSP comes out as a clear winner. According to the results of this work, development costs are increased 2 or 3 times when developing for FPGA devices. This is due to the lack of tools that provide functionality compared to DSP development tools. Therefore a tradeoff must be made between development cost and unit price + performance. If performance goals are relaxed and the product must come out in a short period of time, the DSP will be a better option. If engineering budget is low and a small quantity of products will be released, the DSP is an option to consider. Otherwise, if performance is vital the DSP is no longer an option.

Some tools such as System Generator, by Xilinx and MathWorks, and LabVIEW FPGA from National Instruments provide greater development abstraction at the cost of synthesis control. Usually they are also limited to certain functionalities. This gap should decrease eventually. Tools such as Core Generator from Xilinx are including more optimized processing blocks that can just be inserted in a design and used. Verification tools such as ChipScope are also helping the designer by automatically setting up logic for verification through the programming link, a task that had to be done

manually before. As long as this gap exists, more time, effort and skills will be needed from the FPGA developer than the DSP developer.

In terms of the quality of the results, the DSP implementation offers better quality than the FPGA implementation. This is not a problem of the devices, but of the implementations. To increase the quality of the FPGA results a bigger word length should be used for the watermark DCT. In our implementation there are two DCT calculations: the DCT of the pixel-block and the DCT of the watermark block. These two types of information are processed in the same manner, but they have different meanings. The quality of the resulting pixel-block DCT is acceptable since for the human eye the difference after compression is not visually noticeable. A PSNR value around 20 dB or 30 dB is required in order to begin seeing distortions in the image. Our results are all above that range. However, the results from the watermark DCT may require better precision. The difference relies on the end purpose of each of the information. The end purpose of the pixels is for visual purposes, while the end purpose of the watermark is for numerical and computer analysis purposes. Small changes are not negligible in computer analysis. High-precision is required especially for watermarking purposes. Modifying the DCT to accomplish this translates into designing two different second 1-D DCT stages, or implementing precision-control logic. This eventually means using a bigger FPGA device which has higher unit costs.

One major drawback of the implementation was its size. It basically occupies the entire 200,000 gate device using 95% of the slices available and 11 of the 12 block RAMs. In other words, no more space is available for interface logic or extra features, which probably will be needed or desired. One solution is to reduce parallelism in the implementation at the cost of throughput rate. This basically means that instead of processing 8 values at a time, we can process just 4, at the double of the time. This is feasible since the maximum frequency of the implementation was 60 MHz. If the clock frequency attained after performing those changes is below the required operating frequency, a Digital Clock Manager (DCM) can be included in the design. This circuit basically adds a digital PLL and clock buffers to increase the clock frequency. Another solution to this disadvantage would be to remove all memory elements from the FPGA logic fabric and utilize external memory, which is inexpensive. That would limit the performance of the design by adding memory access times that are not currently present. Since digital video processing anyways requires large amounts of memory, and in order to build a complete system external memory is required, this solution would not represent a major drawback. Another solution would be to hand over the Control Unit (pipeline control) to a small and low-power microprocessor or microcontroller, such as the MSP430 from Texas Instruments, and

leave the FPGA to pure processing only. This would free up that logic space, leading to a possible smaller FPGA selection and consequently reduce on costs. Ultimately, choosing a bigger FPGA can solve this issue.

Most of the hardware implementations of watermarking systems are ASIC-based implementations. Therefore, there are no substantial results available against which we can properly compare our results. One of the FPGA-based implementation that is similar to our work is [31]. Although the watermarking method employed by the authors is different, and is targeted at watermarking still images it can serve as comparison. The authors developed a method which achieved a run time of 1.28 seconds per image for a digital camera with 45% utilization on a 100,000 gate FPGA. In terms of performance our implementation executes better, but they have achieved a smaller implementation size. The small size is probably due to the lack of real-time video constraints found in their application. In [12] a spatial-domain, still-image watermarking method implemented into a Virtex-II FPGA shows an execution time of 19.842 ns (50.398 MHz) for a pixel by pixel processing. For 640 x 480 pixels it would take 6 ms, representing a processing speed of 166 frames per second. In this case the processing capability is greater (about 3 times greater), but the implementation uses a simpler spatial-domain method and a different FPGA chip. Comparing with an ASIC implementation found in [13], the ASIC operated at frequency of 292 MHz and consumed 6.93 mW of power. The method is a spatial-domain technique for still-images that processes 8x8 blocks of pixels. The ASIC solution proves to be superior in both performance and power consumption.

6.6 Conclusions

Given that no research on a video watermarking low-power FPGA implementation for a surveillance application has been published in the literature, a novel architecture for performing video-watermarking on an FPGA, with focus on low-power concerns, is presented. At the same time the results contribute by providing additional insight on how implementations such as this one can perform under the requirements of video watermarking, since a limited number of works on surveillance video watermarking and watermarking hardware implementations are available at the present time. The results presented on these pages can be extended to other watermarking methods that are based on 8x8 block DCT computations such as [1, 22, 55, 56, 57, 58] since the DCT is the main stage of the implementation. The issues regarding the development of such an implementation were

discussed, along with the analysis and the justification for this type of hardware architecture. Other future video watermarking hardware implementations for surveillance systems can benefit from this work, since the requirements for this type of system were identified and presented.

The present work successfully accomplished the selection of a video watermarking algorithm appropriate for a low-power security camera, and the development of the corresponding FPGA implementation. The MPEG-2 based algorithm was successfully adapted to be compression format independent but leaving room for future compression format incorporation. By adapting the algorithm the architecture was then developed. The architecture exploits the DCT processing resources, thus saving on space. The implementation satisfies real-time video requirements, being capable of processing more than 30 frames per second. The resulting architecture is simple and scalable. The inherent parallelism found in the algorithm allows doing the processing in parallel, thus taking advantage of a hardware implementation. Therefore, the implementation size can easily be made smaller by reducing parallelism at the expense of reduce performance. The pipelined design allows for a high throughput rate. Clock gating was used to turn off sections that were not being used, and chip enable controls to prevent outputs from changing and dissipating power. The implementation allows us to reduce the clock frequency and consequently save power from slower switching, since more than 30 frames can be processed per second. A lot of effort was placed on coding the VHDL description in order to achieve a good representation of the architecture using the minimum required logic.

In terms of the contributions on the implementation of a low-power video watermarking system, the achievements were:

- The requirements for a low-power surveillance, or secure, camera watermarking application were presented.
- Low-power hardware design practices and low-power programming practices were also presented.
- The complete architecture of the developed system was exposed in detail, and the resulting FPGA circuit was presented.
- The different stages of the system were compared in terms of logic utilization.

The verification tests show that the outputs from the FPGA implementation are acceptable when compared to the MATLAB results, and can be used for watermark detection.

The results suggest that an FPGA-based implementation is superior in terms of device cost, power consumption, and performance. The results also suggest that the DSP is superior in terms of Development cost. In terms of the specific implementations performed for this work, the DSP seems to offer better quality results due to its fixed word size. Possible solutions to this issue are presented. One drawback of the implementation was its relative big size. Several options to manage this problem are discussed with their respective impacts. The importance of large amounts of memory for video processing systems was also observed. Another minor drawback is the order in which the outputs are presented but a work-around to this situation was offered. In terms of the watermarking algorithm, with a 90% quality compression performed, watermark amplitude of 8, a chip rate of 10,000, and 60% detection threshold, the watermark was detected. Results suggest that increasing the chip rate and the amplitude increases the level of detection, but increasing amplitude can make the watermark visible.

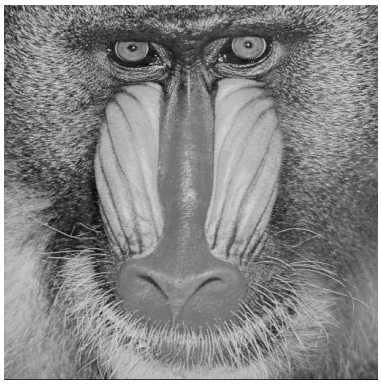
Since the results suggest that the FPGA outperforms the DSP when dealing with video watermarking applications, the DSP no longer seems as an option for this type of application. The selection must be between an ASIC and an FPGA device. FPGAs become more practical and efficient choices due to their reduced cost and development time, when the system is not produced in high quantities.

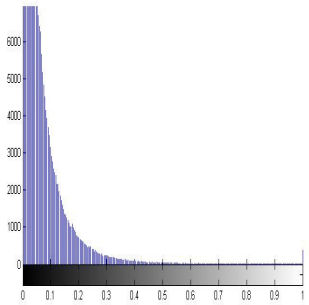
Recommendations for building a secure camera system include using an FPGA device to handle the video processing, with fast external RAM, and using a low-power microcontroller to handle the control of the system, or using a bigger FPGA device that can encompass all the functionality. The compression method selected should be H.263, H.261 or a similar format, since they are aimed at low bit-rate applications.

Future work includes completing the hardware implementation in the FPGA by downloading the design into the FPGA, and developing the I/O interface to acquire, process and output a video stream signal in real-time. For a higher-valued comparison the DSP code should be better optimized for the application and device. Different watermark sizes should be examined and their corresponding power consumptions compared. The complete system, including a digital video camera, battery identification and incorporation, wireless transmission and detection of the watermark in the storage station could be included in a future analysis in order to have a more complete perspective on the performance of the system.

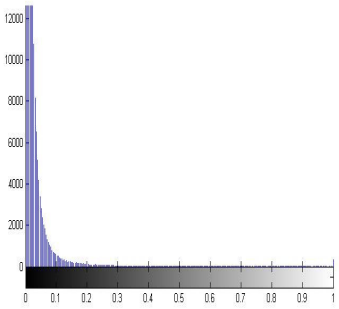
APPENDIX

The ten images used for verification can be seen below, along with their frequency distributions. A table is also included that details the frequency distribution according to specified ranges.

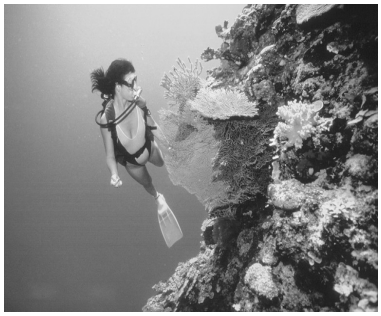


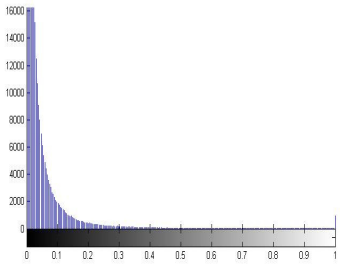


RANGE OF VALUES	% of coefficients
x < 0.05	76.9077
0.05 ≤ x < 0.10	12.1117
0.10 ≤ x < 0. 15	5.1792
0.15 ≤ x < 0.20	2.4883
x ≥ 0.20	3.3131

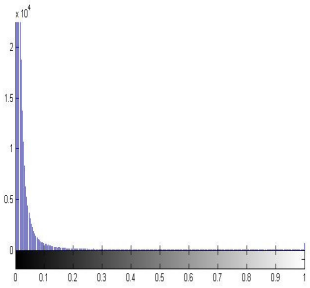


RANGE OF VALUES	% of coefficients
x < 0.05	94.9135
0.05 ≤ x < 0.10	3.2410
0.10 ≤ x < 0. 15	0.8141
0.15 ≤ x < 0.20	0.3510
x ≥ 0.20	0.6805

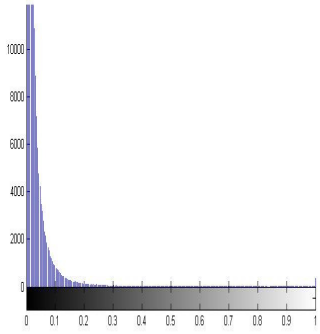




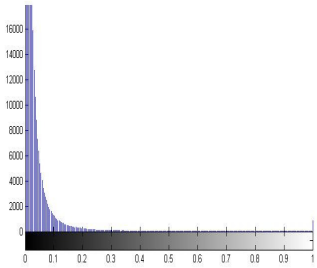
RANGE OF VALUES	% of coefficients
x < 0.05	89.6899
0.05 ≤ x < 0.10	5.6493
0.10 ≤ x < 0. 15	2.0226
0.15 ≤ x < 0.20	0.9201
x ≥ 0.20	1.7181



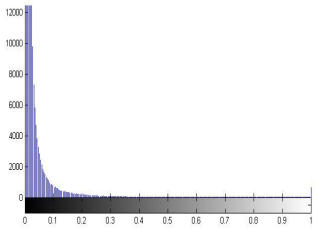
RANGE OF VALUES	% of coefficients
x < 0.05	96.0607
0.05 ≤ x < 0.10	2.3659
0.10 ≤ x < 0. 15	0.6271
0.15 ≤ x < 0.20	0.2754
x ≥ 0.20	0.6709



RANGE OF VALUES	% of coefficients
x < 0.05	93.2209
0.05 ≤ x < 0.10	4.4411
0.10 ≤ x < 0. 15	1.1826
0.15 ≤ x < 0.20	0.4471
x ≥ 0.20	0.7084

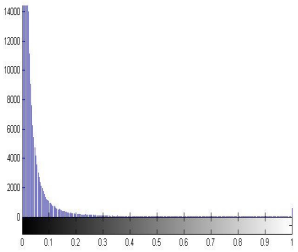


RANGE OF VALUES	% of coefficients
x < 0.05	92.9222
0.05 ≤ x < 0.10	4.3821
0.10 ≤ x < 0. 15	1.2009
0.15 ≤ x < 0.20	0.5198
x ≥ 0.20	0.9750

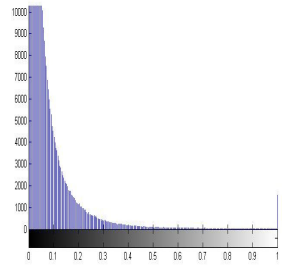


RANGE OF VALUES	% of coefficients
x < 0.05	93.6062
0.05 ≤ x < 0.10	3.5255
0.10 ≤ x < 0. 15	1.1742
0.15 ≤ x < 0.20	0.5615
x ≥ 0.20	1.1326

IMAGE
NOT
PUBLISHABLE
Description: office lounge



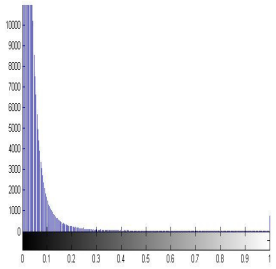
RANGE OF VALUES	% of coefficients
x < 0.05	92.6699
0.05 ≤ x < 0.10	4.3757
0.10 ≤ x < 0. 15	1.3216
0.15 ≤ x < 0.20	0.5742
x ≥ 0.20	1.0586



RANGE OF VALUES	% of coefficients
x < 0.05	76.7456
0.05 ≤ x < 0.10	11.5044
0.10 ≤ x < 0. 15	4.9911
0.15 ≤ x < 0.20	2.5637
x ≥ 0.20	4.1951

IMAGE NOT PUBLISHABLE

Description: Meeting room



RANGE OF VALUES	% of coefficients
x < 0.05	88.3571
0.05 ≤ x < 0.10	7.7910
0.10 ≤ x < 0. 15	1.9010
0.15 ≤ x < 0.20	0.7044
x ≥ 0.20	1.2464

The extracted blocks from the three selected images are shown below, along with the randomly generated 8x8 block:

Test block #1

Location: Rows = 136-143, Columns = 88-95, out of 512 x 512

155	156	185	195	166	157	160	143
124	97	89	98	108	122	114	83
160	175	176	139	85	61	75	96
185	146	98	81	90	94	96	107
133	131	136	155	160	135	117	125
182	172	159	138	107	83	84	96
50	72	95	85	73	111	161	174
126	131	124	82	66	128	175	151

Test Block #2

Location: Rows = 448-455, Columns = 128-135, out of 512 x 512

185	129	149	109	88	140	196	178
188	162	122	78	124	175	183	198
175	176	149	151	196	204	170	199
164	182	198	202	166	187	190	186
138	177	204	181	118	176	203	147
157	170	162	163	160	185	189	153

193	173	142	159	191	177	181	176
142	163	179	181	191	172	172	115

Test Block #3

Location: Rows = 336-343, Columns = 496-503, out of 512 x 512

114	113	108	99	92	93	102	111
120	117	112	108	107	108	110	112
113	110	107	108	113	114	110	106
97	97	98	103	109	111	109	106
88	91	96	101	106	110	112	114
82	87	91	97	103	107	112	114
81	82	86	94	103	108	107	105
87	85	87	97	109	113	107	99

Test Block #4

Location: Rows = 344-351, Columns = 448-455, out of 512x512

113	115	119	111	94	95	94	93
124	124	113	118	123	116	111	107
130	130	124	123	123	123	119	123
130	131	130	131	131	130	128	125
126	130	131	133	123	116	119	119
124	124	127	128	126	126	123	119
124	125	128	131	124	128	124	122
131	131	130	130	128	128	126	121

Test Block #5

Location: Rows = 8-15, Columns = 72-79, out of 512x512

116	111	119	119	111	113	119	118
124	130	131	128	124	131	136	130
130	127	131	137	137	136	137	129
128	130	128	132	135	130	135	131

132	130	131	131	141	133	124	119
128	123	126	139	135	133	128	121
131	126	121	128	123	130	128	130
139	141	141	139	139	139	128	126

Test Block #6

Location: Rows = 424-431, Columns = 224-231, out of 512x512

156	154	143	150	139	145	145	145
154	154	142	141	154	150	145	139
157	161	162	161	154	148	141	126
154	161	161	159	161	154	139	133
154	161	161	153	152	152	143	145
159	161	159	159	147	150	143	139
156	160	154	156	147	143	135	132
152	154	156	147	143	139	131	130

Test Block #7

Location: Rows = 456-463, Columns = 384-391, out of 512x512

139	150	148	147	147	146	157	146
142	150	149	147	150	146	148	142
144	150	154	148	149	144	143	143
134	155	147	151	148	146	146	149
140	147	150	146	150	144	142	144
130	141	149	147	145	146	146	138
124	133	137	138	142	143	144	142
115	122	131	130	134	133	134	135

Test Block #8

Location: Rows = 352-359, Columns = 184-191, out of 512x512

106	119	142	189	155	56	39	48

116	137	181	180	69	44	48	46
127	169	197	117	41	40	40	45
153	198	144	46	31	37	37	39
188	181	65	40	33	37	32	42
198	124	45	40	46	47	43	44
174	60	35	38	47	49	50	50
96	38	37	39	47	44	48	49

Test Block #9

Location: Rows =88-96, Columns = 80-87, out of 512x512

82	83	88	92	94	93	95	95
90	87	92	91	94	88	93	100
87	88	91	93	94	95	97	93
86	89	88	95	96	96	96	96
87	90	90	90	99	95	95	95
84	87	88	92	93	94	99	95
91	87	91	92	88	93	92	91
92	88	89	91	95	92	97	94

Test Block #10

Random Block

48	105	140	192	230	99	108	178
107	229	102	202	145	159	151	250
218	1	50	234	161	178	144	205
125	75	159	215	59	101	182	179
208	12	187	93	139	105	130	123
117	176	95	158	237	167	197	29
116	165	2	186	85	213	124	169
114	250	107	49	167	94	47	93

The script used for verification in MATLAB is shown next.

```
% DCT computation using row-column decomposition, and matrix operations
```

```
% done without MATLAB matrix support (C style), and transpose
% ONLY FOR N x N blocks
%
% Author: William A. Irizarry
%
% Equation from:
% Image Compression and the discrete cosine transform.pdf
%
% Z =            N x N cosine coefficient matrix
% image_block =  N x N block from the original image
%
% I,J =          row and column coordinates in image_block


N = 8;
M = 8;
Z = zeros(N,N);
temp = zeros(N,N);
C = zeros(N,N);
D = zeros(N,N);
Z_trans = zeros(N,N);

%data1 to data_10 are declared


%--------- VARIABLES FOR PSEUDORANDOM GENERATOR ------------------
X = ones(8);
AMPLITUDE = 11;
%flag = 0;
KEY = 5461;    % initial seed = 1010101010101
lfsr = KEY;
BIT0 = 1;         % bit 0    = 0000000000001
BIT2 = 4;         % bit 2    = 0000000000100
BIT3 = 8;         % bit 3    = 0000000001000
BIT12 = 4096;     % bit 12   = 1000000000000
%-------------------------------------------------------------------

for counter = 1:10,

    if counter == 1
        image_block = data_1;
    elseif counter == 2
            image_block = data_2;
    elseif counter == 3
            image_block = data_3;
    elseif counter == 4
            image_block = data_4;
    elseif counter == 5
```

```
            image_block = data_5;
    elseif counter == 6
            image_block = data_6;
    elseif counter == 7
            image_block = data_7;
    elseif counter == 8
            image_block = data_8;
    elseif counter == 9
            image_block = data_9;
    elseif counter == 10
            image_block = data_10;
        end


%-------------------------------------------------------------------------
%  Cosine Coefficients calculations - ONLY 1 TIME CALCULATION FOR A N SIZE
%-------------------------------------------------------------------------
for I = 0:N-1,
    for J = 0:N-1,
        if I == 0
            Z(I+1,J+1) = 1 / sqrt(N);
        else
            Z(I+1,J+1) = (sqrt(2 / N)) * cos (((pi * I) * (2*J + 1))/(2*N));
        end
    end
end

Z_trans = Z.';
%=========================================================================


% Matrix multiplication
% Equivalent to --> temp = Z * (image_block) ;
for U = 0:N-1,
  for I = 0:M-1,
    product = 0;
      for J = 0:N-1,
             temp_1 = Z(U+1,J+1)* image_block(J+1,I+1);
             product = product + temp_1;
      end
    temp(U+1,I+1) = product;
  end
end


after_1st_stage_wm = temp;
```

```
% Equivalent to --> C = temp * Z_trans;
for U = 0:N-1,
  for I = 0:M-1,
    product = 0;
      for J = 0:N-1,
             temp_1 = Z_trans(J+1,I+1) * temp(U+1,J+1) ;
             product = product + temp_1;
      end
    C(U+1,I+1) = product;
  end
end



pixels_after_2nd_stage = C; %* (2^5);

%------------------------------- COMPARING RESULTS ----------------------
% Calculated DCT
%(C)

% DCT from MATLAB function
%DCT_from_MATLAB = (dct2(image_block))
%------------------------------------------------------------------------

% -------------- Watermark Pseudorandom 8x8 block generation --------------

counter_chip_rate = 0;

for j=1:8,
     for i=8:-1:1,

           if (bitand(lfsr,BIT12)) == 4096
              X(i,j) = AMPLITUDE;
           else
              X(i,j) = -1 * AMPLITUDE;
           end

           if (bitand(BIT0,lfsr)) == 1      y1 = 1; else y1 = 0; end
           if (bitand(BIT2,lfsr)) == 4      y2 = 1; else y2 = 0; end
           if (bitand(BIT3,lfsr)) == 8      y3 = 1; else y3 = 0; end
           if (bitand(BIT12,lfsr)) == 4096  y4 = 1; else y4 = 0; end

           %shifting the register
           lfsr = bitshift(double(lfsr),1);

           %limits the shift to a 13 bit length
           lfsr = bitand(lfsr,8191);
```

```
           %XOR feedback - putting new bit
            lfsr = bitset(lfsr,1,(xor(xor(y1,y2),xor(y3,y4))));

     end
end

% Matrix multiplication
% Equivalent to --> temp = Z * (image_block) ;
for U = 0:N-1,
  for I = 0:M-1,
    product = 0;
      for J = 0:N-1,
             temp_1 = Z(U+1,J+1)* X(J+1,I+1);
             product = product + temp_1;
      end
    temp(U+1,I+1) = product;
  end
end

after_1st_stage_pixel = temp;


% Equivalent to --> C = temp * Z_trans;
for U = 0:N-1,
  for I = 0:M-1,
    product = 0;
      for J = 0:N-1,
             temp_1 = Z_trans(J+1,I+1) * temp(U+1,J+1) ;
             product = product + temp_1;
      end
    D(U+1,I+1) = product;
  end
end
%-------------------------------------------------------------------------

wm_after_2nd_stage = D;


% 97% quality
 Q_table = [
 1     1       1       1       1       1       3       3;
 1      1       1       1       2       2       4       4;
 1      1       1       2       2       4       5       3;
 1      1       1       2       3       4       3       6;
 1      1       2       3       4       5       6       4;
 1      3       3       5       7       6       5       7;
 1      3       6       5       6       7       6       6;
 4      4       5       6       7       7       6       6];
```

```
%Q_table = Q_table.';


% Quantizing
C = C ./ (Q_table);
C(find(C < 0.5 & C > -0.5 )) = 0;
after_quant = C;


% Adding the watermark
result = C + D.';
% Not adding where the coefficient is 0
temp = C;
temp(find(temp ~= 0)) = 1;
result = result .* temp;
after_adding_wm = result;

% Correct result, according to MATLAB
DCT_from_MATLAB_image = (dct2(image_block));
DCT_from_MATLAB_random = (dct2(X));
MATLAB_quant = DCT_from_MATLAB_image ./ (Q_table);
MATLAB_quant(find(MATLAB_quant < 0.5 & MATLAB_quant > -0.5 )) = 0;
MATLAB_result = MATLAB_quant + DCT_from_MATLAB_random.';
temp = MATLAB_quant;
temp(find(temp ~= 0)) = 1;
MATLAB_result = MATLAB_result .* temp;
 result  - MATLAB_result;


end
```

BIBLIOGRAPHY

[1] R. B. Wolfgang, C. I. Podilchuk, and E. J. Delp, "Perceptual Watermarks for Digital Images and Video," Proceedings of the IEEE, Vol. 87, No. 7, July 1999, pp. 1108-1126.

[2] Wikipedia, http://www.wikipedia.org.

[3] Christine I. Podilchuk, and Edward J. Delp, "Digital Watermarking: Algorithms and Applications". IEEE Signal Processing Magazine. Vol. 1, July 2001, pp. 33-46.

[4] Sin-Joo Lee, and Sung-Hwan Jung, "A survey of watermarking techniques applied to multimedia". IEEE International Symposium on Industrial Electronics, Korea, June 2001. Vol. 1, pp. 272 – 277.

[5] Frank Hartung, and Bernd Girod. "Watermarking of Uncompressed and Compressed Video". IEEE Transactions on Signal Processing. Vol. 66, No. 3, May 1998, pp. 283 - 302.

[6] Saraju P. Mohanty, "Digital Watermarking: A Tutorial Review". 1999, http://www.cs.unt.edu/~smohanty/research/Reports/MohantyWatermarkingSurvey1999.pdf.

[7] Watermarking World, http://www.watermarkingworld.org.

[8] I. Cox, J. Kilian, T. Leighton, and T. Shamoon. "Secure spread spectrum watermarking for multimedia". IEEE Transactions on Image Processing. Vol. 6, Issue 12, Dec. 1997, pp. 1673 – 1687.

[9] Ambalanath Shan, and Ezzatollah Salari, "Real-Time Digital Video Watermarking". 2002 Digest of Technical Papers: International Conference on Consumer Electronics, June 2002, pp.12 – 13.

[10] Chiou-Ting Hsu, and Ja-Ling Wu, "Digital Watermarking for Video". 13th International Conference on Digital Signal Processing Proceedings, DSP 97. Vol. 1, July 1997 pp. 217 – 220.

[11] Christoph Busch, Wolfgang Funk, and Stephen Wolthusen, "Digital Watermarking: From Concepts to Real-Time Video Applications". IEEE Computer Graphics and Applications. Vol. 19, Issue 1, Jan.-Feb. 1999. pp. 25 – 35.

[12] Saraju P. Mohanty, Renuka Kumara C., and Sridhara Nayak. "FPGA Based Implementation of an Invisible-Robust Image Watermarking Encoder". 7th International Conference on Information Technology, CIT 2004. India, December 2004 pp. 344–353.

[13] Saraju P. Mohanty, N. Ranganathan, and Ravi K. Namballa, "VLSI Implementation of Visible Watermarking for a Secure Digital Still Camera Design". Proceedings 17th International Conference on VLSI Design, 2004. pp. 1063 – 1068.

[14] Nebu John Mathai, Ali Sheikholesami, and Deepa Kundur, “VLSI Implementation of a Real-Time Video Watermark Embedder and Detector”. Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS '03. Vol. 2, May 2003 pp. II772 - II775.

[15] Nebu John Mathai, Ali Sheikholesami, and Deepa Kundur, “Hardware Implementation Perspectives of Digital Video Watermarking Algorithms”. IEEE Transactions on Signal Processing. Vol. 51, Issue 4, April 2003. pp. 925 - 938.

[16] G. L. Friedman, “The Trustworthy Digital Camera: Restoring Credibility to the Photographic Image”. IEEE Transactions on Consumer Electronics, Nov. 1993, pp. 905-910.

[17] Michael Pramatejakis. Tobias Oelbaum and Klaus Diepold. “Authentication of MPEG-4 Based Surveillance Video”. International Conference on Image Processing (ICIP). 2004.

[18] S. Walton, “Information Authentication for a Slippery New Age”. Dr. Dobbs Journal, Vol. 20, No. 4, Apr 1995, pp. 18-26.

[19] R. B. Wolfgang, and E. J. Delp, “Fragile Watermarking Using the VW2D Watermark”. In Proceedings SPIE/IS&T International Conference on Security and Watermarking of multimedia Contents, 1999, pp. 40-51.

[20] J. Fridrich, and M. Goljan, “Protection of Digital Images Using Self-Embedding”. Symposium on Content Security and Data Hiding in Digital Media, New Jersey Institute of Technology, May 14, 1999.

[21] J. Fridrich, “Image Watermarking for Tamper Detection”. Proceedings ICIP ’98, Chicago, Oct 1998, pp.404-408.

[22] J. Fridrich, “A Hybrid Watermark for Tamper Detection in Digital Images”. Fifth International Symposium on Signal Processing and its Applications, ISSPA ’99, Brisbane, Australia, 22-25 August 1999, pp. 301-304.

[23] C. M. Kung, T. K. Truong, and J. H. Jeng, “A Robust Watermarking and Image Authentication Technique”. IEEE 37th Annual 2003 International Carnahan Conference on Security Technology, 14-16 Oct. 2003, pp. 400- 404.

[24] Tsong-Yi Chen, Chien-Hua Huang, Thou-Ho Chen, and ‘Cheng-Chieh Liu, “Authentication of Lossy Compressed Video Data by Semi-Fragile Watermarking”, International Conference on Image Processing (ICIP), 2004, pp. 2637-2640.

[25] Deepa Kundur, and Dimitrios Hatzinakos, “Digital Watermarking for Telltale Tamper Proofing and Authentication”. Proceedings of the IEEE, Vol. 87, No. 7, July 1999, pp.1167-1180.

[26] Ju Wang, Adrian R. Steele, and Jonathan C.L.Liu, “Efficient Integration of Watermarking With MPEG Compression”. IEEE International Conference on Multimedia and Expo (ICME), 2004.

[27] Nicola Checcacci, Mauro Barni, Franco Bartolini, and Stefano Basagni, “Robust Video Watermarking for Wireless Multimedia Communications”. Wireless Communications and

Networking Conference, Chicago, 2000, Vol. 3, pp. 1530-1535.

[28] Focus: Authentication and Tamper detection of Digital Video content for a security surveillance application.

[29] Mauro Barni, Franco Bartolini, J. Fridrich, and A. Piva, “Digital watermarking for the authentication of AVS data”. European signal processing conference Number 10, Tampere, Finland, April 9 2000, pp. 1037-1040.

[30] D. K. Roberts, “Security Camera Video Authentication”. Proceedings of 2002 IEEE 10th Digital Signal Processing Workshop and the 2nd Signal Processing Education Workshop. 13-16 Oct. 2002, pp. 125- 130.

[31] Hyun Lim, Soon-Young Park, Seong-Jun Kang, and Wan-Hyun Cho, “FPGA Implementation of Image Watermarking Algorithm for a Digital Camera”. IEEE Pacific Rim Conference on Communications, Computers and signal Processing, 2003. 28-30 Aug. 2003. Vol. 2, pp. 1000-1003.

[32] Joel Rotem, “Building Digital Video Surveillance Systems: Beyond MPEG-4”, Embedded Edge magazine, Spring 2004, pp 18-21.

[33] http://www.ti.com.

[34] http://www.pixeltools.com/mpg_escort.html.

[35] http://www.kintronics.com.

[36] http://www.samsung.com/Products/SmartDomeSystems/IPSmartDomeCamera/index.htm.

[37] http://www.ezcctv.com/geovision-surveillance.htm.

[38] http://www.tridfx.com.

[39] Franco Bartolini, Anastasios Tefas, Mauro Barni, and Ioannis Pitas, “Image Authentication Techniques for Surveillance Applications”. Proceedings of the IEEE, Vol. 89, No. 10, October 2001, pp. 1403-1418.

[40] F. Hartung, and B. Girod, "Digital Watermarking of MPEG-2 Coded Video in the Bitstream Domain". Proceedings ICASSP 97, Vol. 4, pp. 2621-2624, Munich, Germany, April 21-24 1997, pp.2621-2624.

[41] I.J. Cox, J.Kilian, T.Leighton, and T.Shamoon, “Secure Spread Spectrum Watermarking for Images, Audio and Video”. Proceedings of the 1996 International Conference on Image Processing, Vol. 3, Lausanne, Switzerland, September 16-19 1996, pp. 243-246.

[42] F. Hartung, and B. Girod, “Digital Watermarking of Raw and Compressed Video”. Proceedings European EOS/SPIE Symposium on Advanced Imaging and Network Technologies, Berlin, Germany, October 1996, pp 205-213.

[43] F. Hartung, and B. Girod, "Watermarking of MPEG-2 Encoded Video Without

Decoding and Re-encoding". Proceedings Multimedia Computing and Networking 1997 (MMCN 97), San Jose, CA, February 1997.

[44] F. Hartung, and B. Girod, "Copyright Protection in Video Delivery Networks by Watermarking of Pre-Compressed Video". ECMAST '97, Springer Lecture Notes in Computer Science, Springer, Heidelberg, Vol. 1242, 1997, pp. 423-436.

[45] F. Hartung, and B. Girod, "Watermarking of Uncompressed and Compressed Video". Signal Processing, Special Issue on Copyright Protection and Control, Vol. 66, No. 3, May 1998, pp. 283-301.

[46] F. Hartung, J.K. Su, and B. Girod, "Spread Spectrum Watermarking: Malicious Attacks and Counterattacks". Proceedings of SPIE Electronic Imaging '99, Security and Watermarking of Multimedia Contents, San Jose, USA, January 1999

[47] Gerrit Cornelis Langelaar, “Real-time Watermarking Techniques for Compressed Video Data”. Ph.D. Thesis, Delft University of Technology, 2000, 146 pp.

[48] JPEG Compression Quality from Quantization Tables, http://www.impulseadventure.com/photo/jpeg-quantization.html.

[49] John P. Uyemura, “Introduction to VLSI Circuits and Systems”. John Wiley and Sons, Inc. 2002, p. 259.

[50] http://www.xilinx.com.

[51] T. Xanthopoulos, and A. Chandrakasan, "Low-power DCT core using adaptive bitwidth and arithmetic activity exploiting signal correlations and quantization". IEEE Journal of Solid-State Circuits, Vol. 35, No. 5, May 2000, pp.740-750.

[52] A. Madisetti, and A.N. Willson,“A 100MHz 2-D 8x8 DCT/IDCT processor for HDTV applications”. IEEE transactions on circuits and systems for video technology 1995, vol. 5, pp. 158-165.

[53] Rulph Chassaing, “Digital Signal Processing and Applications with the C6713 and C6416 DSK”. John Wiley and Sons, 2005. New Jersey, Chapter 2.

[54] Vivek Tiwari, Sharad Malik, and Andrew Wolfe, “Power analysis of embedded software: A first step toward software power minimization”. IEEE Transactions on VLSI Systems, 2(4) December 1994, pp. 437-445.

[55] J. Fridrich, and M. Goljan, "Images with self-correcting capabilities". IEEE International Conference on Image Processing Proceedings, Kobe, Japan, October 25-28 1999, pp. 792-796.

[56] Wei Zhang, Sen-ching S. Cheung, and Minghua Chen, “Hiding Privacy Information in Video Surveillance System”, IEEE International Conference on Image Processing. 11-14 Sept. 2005 Vol. 3, pp. II- 868-71.

[57] C. M. Kung, T. K. Truong, and J.H. Jeng, “A Robust Watermarking and Image Authentication Technique”, IEEE 37th Annual 2003 International Carnahan Conference on Security Technology, 14-16 Oct. 2003, pp. 400- 404.

[58] M. Wu, and B. Liu, “Watermarking for Image Authentication”. Proceedings of IEEE International Conference on Image Processing, October 4-7, 1998, Chicago, Illinois, Vol. 2, pp 437 – 441.

[59] The New Currency: About the Notes. http://www.moneyfactory.gov/newmoney/main.cfm/currency/aboutNotes.

[60] L.De Strycker, P.Termont, J.Vandewege, J.Haitsma, A.Kalker, M.Maes and G.Depovere, “Implementation of a real-time digital watermarking process for broadcast monitoring on a TriMedia VLIW processor”. IEE Proceedings-Vis. Image Signal Process., August 2000, Vol. 147, No. 4, pp. 371-376.