Real-Time Perceptual Watermarking Architectures for Video Broadcasting

Saraju P. Mohanty[a,*], Elias Kougianos[a]

[a]*NanoSystem Design Laboratory (NSDL)[1], University of North Texas, Denton, TX 76207, USA*

Abstract

Existing secure embedded systems are primarily cryptography based. However, for effective Digital Rights Management (DRM) of multimedia in the framework of embedded systems, both watermarking and cryptography are necessary. In this paper, a watermarking algorithm and corresponding VLSI architectures are presented that will insert a broadcasters logo into video streams in real-time to facilitate copyrighted video broadcasting and internet protocol television (IP-TV). The VLSI architecture is prototyped using a hardware description language (HDL) and when realized in silicon can be deployed in any multimedia producing consumer electronics equipment to enable real-time DRM right at the source. The watermark is inserted into the video stream before MPEG-4 compression, resulting in simplified hardware requirements and superior video quality. The watermarking processing is performed in the frequency (DCT) domain. The system is initially simulated and validated in MATLAB/Simulink® and subsequently prototyped on an Altera® Cyclone-II FPGA using VHDL. Its maximum throughput is 43 frames/sec at a clock speed of 100 MHz which makes it suitable for emerging real-time digital video broadcasting applications such as IP-TV. The watermarked video is of high quality, with an average Peak-Signal-to-Noise Ratio (PSNR) of 21.8 dB and an average Root-Mean-Square Error (RMSE) of 20.6.

Keywords: VLSI Architecture, Video Broadcasting, IP-TV, Watermarking, Content Protection, Copyright Protection, Real-Time Security, Digital Rights Management, Multimedia Security

*Corresponding author

Email addresses: saraju.mohanty@unt.edu (Saraju P. Mohanty), elias.kougianos@unt.edu (Elias Kougianos)

[1]http://nsdl.cse.unt.edu

1. Introduction

As broadband Internet is widely available, multimedia resources are openly accessed and distributed quickly and widely. From this trend, it is predicted that as more and more music, movies, and images are exchanged on the Internet, download multimedia sales will eventually surpass traditional sales. This development will benefit the multimedia product owners as sales will increase. However, it will also pose challenges to their ownership as most multimedia products are distributed in unsecured formats. This situation is further aggravated by the fact that duplicating digital multimedia products is almost cost-free and fast due to the availability of free or low cost tools. To legal authorities, arbitrating the ownership of multimedia products is not easy, unless a mechanism guarantees the genuine integrity of copyright. Digital Rights Management (DRM) using encryption and watermarking is investigated as a solution for copyright and intellectual property protection of multimedia.

Along these lines, there is a pressing need for real-time copyright logo insertion in emerging applications, such as internet protocol television (IP-TV). This is demonstrated in figure 1. The visible-transparent watermarking unit accepts broadcast video and one or more broadcaster's logos. The output is real-time compressed (MPEG-4) video with the logo embedded in the stream. This situation arises in IP-TV and digital TV broadcasting when video residing in a server is broadcast by different stations and under different broadcasting rights. Embedded systems that are involved in broadcasting need to have integrated copyright protection mechanisms. The majority of the existing research [1, 2] concentrates on invisible watermarking, which cannot be used for logo insertion. Existing research on visible watermarking [3, 4, 5, 6, 7, 8] is primarily for images, not video. The research works presented in [9, 10] are for video, but they can not be used for real-time DRM. The objective of this paper is to fill the gap of real-time watermarking in consumer electronic equipment at the source end.

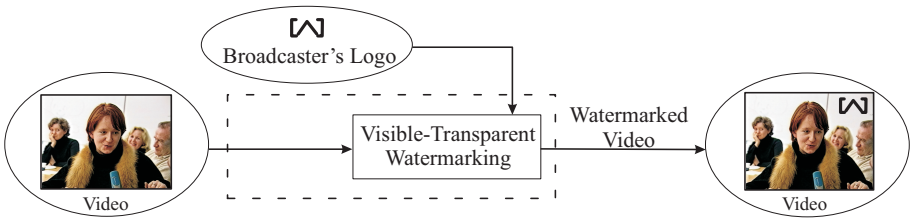


Figure 1: Real-time logo insertion through watermarking.

MPEG-4 has rapidly become one of the mainstream exchangeable video formats in the Internet today because it has high and flexible compression rate, low bit rate, and higher efficiency while providing superior visual quality. Microsoft, Real Networks, and Apple support the MPEG-4 standard and have already embedded MPEG-4 decoders into many of their products. Other companies or organizations also provide MPEG-4 encoders/decoders (or CODECs), and there are even free products available, such as the Xvid codec [11]. This motivated us to consider MPEG-4 as the target video compression method in our research. Thus, the objective of this paper is real-time secure MPEG-4; i.e., watermarking integrated into MPEG-4, operating in real-time.

The novel contributions of this paper are as follows:

1. A perceptual-based adaptive visible watermarking algorithm suitable for real-time applications such as video broadcasting.
2. VLSI architectures for real-time watermarking in the context of compressed video (i.e., the MPEG-4 compression standard).
3. MATLAB/Simulink® prototyping of the watermarking architectures to demonstrate their functionality.
4. VHDL based FPGA prototyping of the VLSI architectures which can be integrated in multimedia producing appliances (e.g., digital video camera, network processor, etc.).

The remainder of this paper is organized as follows: In Section 2, existing literature is discussed. In Section 3, an overview of the MPEG-4 algorithm and its hardware cost perspective is discussed. The proposed watermarking algorithm that produces watermarked compressed video is discussed in Section 4. In Section 5 the proposed hardware architectures are presented. The high-level architectural MATLAB/Simulink® simulations of the overall system and FPGA-based hardware prototyping are discussed in Section 6. Section 7 discusses the experimental results. The paper concludes in Section 8, which summarizes our results and offers suggestions for further research.

2. Significance of Our Work with Respect to Related Prior Research

The existing literature is rich with watermarking algorithms introduced for different types of multimedia objects, such as images, video, audio, and text, and their software implementations. These watermarking algorithms primarily work off-line, i.e., the multimedia objects are first acquired, and then the watermarks are inserted before the watermarked multimedia objects are made available to the

user. In this approach there is a time gap between the multimedia capture and its transmission. The objective of this paper is to present research which will lead to a hardware-based watermarking system to bridge that gap [12, 1, 2, 13, 7, 14]. The watermarking chip will be integrated into the electronic appliance which is an embedded system designed using system-on-chip (SoC) technology.

In prior research [15, 16, 17], a real-time watermarking embedder-detector for a broadcast monitoring system is presented in the context of a VLIW DSP processor. The insertion mechanism involves addition of pseudo-random numbers to the incoming video stream based on the luminance value of each frame. The watermark detection process involves the calculation of correlation values. In [18], the Millennium watermarking system is presented for copyright protection of DVD video in which specific issues, such as watermark detector location and copy generation control, are addressed. In [13], a VLSI architecture for a spread-spectrum based real-time watermarking system is presented. In [19], the graphics processing unit (GPU) is utilized for hardware assisted real-time watermarking. In [20], a watermark detection system is presented for a hardware based video watermark embedder using a Stratix FPGA from Altera®, a PCI controller, and Microsoft Visual C#. In [21, 22], a custom IC for video watermarking is presented that performs all operations using floating-point datapath architectures for both the watermarking embedder and detector. The embedder consumes 60 mW, and the detector consumes 100 mW while operating at 75 MHz and 1.8 V. In [23], a video watermarking system called "traceable watermarking" is proposed for digital cinema. In [24], FPGA prototyping is presented for HAAR-wavelet based real time video watermarking. In [25], a real-time video watermarking system using DSP and VLIW processors is presented that embeds the watermark using fractal approximation.

The research presented in the current paper will significantly advance the state-of-the art in digital rights management. The algorithm and architecture proposed in this paper will be immensely useful for real-time copyright logo insertion in emerging applications, such as IP-TV. An embedded system that will allow such operations needs to have embedded copyright protection facilities such as the one presented in this paper.

3. Watermarking in the MPEG-4 Framework: A Hardware Cost Perspective

The most important phases for video compression are color space conversion and sampling, the Discrete Cosine Transform (DCT) and its inverse (IDCT), Quantization, Zigzag Scanning, Motion Estimation, and Entropy Coding. The

watermarking process is implemented with a single embedding step in the video compression framework. In this section the watermarking algorithm in the framework of MPEG-4 is discussed highlighting the design decisions that were made towards real-time performance through a hardware-based solution.

3.1. Color Space Conversion

The conversion from RGB-color space to YC_bC_r-color space is performed using the following expression [26, 27]:

$$\left.\begin{array}{rcl} Y &=& 0.299R + 0.587G + 0.114B, \\ C_b &=& 0.564(B - Y) + 128, \\ C_r &=& 0.731(R - Y) + 128. \end{array}\right\} \quad (1)$$

A total of 6 adders and 5 multipliers are needed to perform the color conversion, and they can be concurrent. The delay introduced does not contribute significantly to the critical path delay as the conversion takes place at the input stage, where the additive terms in the C_b and C_r components guarantee that their values will be positive. The sampling rate is chosen to be 4:2:0 so that in a 4 pixel group, there are four Y pixels, a single C_b pixel and a single C_r pixel to meet digital TV broadcasting standards.

3.2. Discrete Cosine Transformation (DCT)

DCT is one of the computationally intensive phases of video compression. For ease of hardware implementation, a fast DCT algorithm and its inverse [28, 27] is selected. The fast DCT algorithm reduces the number of adders and multipliers so that the evaluation of the DCT coefficients is accelerated. The two-dimensional DCT and IDCT algorithms can be implemented by executing the one-dimensional algorithms sequentially, once horizontally (row-wise) and once vertically (column-wise).

3.3. Quantization

After the DCT, the correlation of pixels of an image or video frame in the spatial domain has been de-correlated into discrete frequencies in the frequency domain. Since human visual system (HVS) perception is more acute to the DC coefficient and low frequencies, a carefully designed scalar quantization approach reduces data redundancy while maintaining good image quality. In the MPEG-4 video compression standard, a uniform scalar quantization is adopted. The feature of the scalar quantization scheme is an adaptive quantized step size according

to the DCT coefficients of each macroblock. For computational efficiency and hardware simplification, the scalar quantization step size can be chosen from predefined tables [29].

3.4. Zigzag Scanning

Zigzag scanning sorts the matrix of DCT coefficients of video frames in ascending order. For progressive frames and interlaced fields, the zigzag scanning routes are provided by predefined tables as explained in [29, 27].

3.5. Motion Estimation

Prior to performing motion estimation, an image (video frame) is split into smaller pixel groups, called macroblocks, as the basic elements of the image rather than a single pixel. This is driven by a compromise between efficiency and performance to analyze a video's temporal model. A macroblock commonly has a size of 16×16 pixels. With the macroblock in the base frame and its two dimensional motion vector, the current frame can be predicted from the previous frame. In the MPEG-4 standard, the region in which the macroblock is sought for match could be a square, diamond, or of arbitrary shape. For most applications, a square region is considered. For example, if the macroblock has pixel size, the searching region will be a pixel block. The similarity metric for two blocks is the minimized distance between them. For simplicity, the sum of the absolute difference (SAD) is applied as the criterion for matching, as represented in [29, 27]:

$$SAD(x,y) = \begin{cases} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |c(i,j) - p(i,j)| \text{ for } (x,y) = (0,0), \\ \\ \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |c(i,j) - p(i+x, j+y)|, \end{cases} \tag{2}$$

where $c(i, j)$ are the pixels of the current block, $i, j = 0, 1, , N - 1$; $p(m, n)$ are the pixels of the previous block in the searching region, and $m, n = -R, -R + 1, ..., 0, 1, , R + N - 1$, where the size of the macroblock is R pixels. Motion estimation is in the critical path of video compression coding and most time delay will occur at this step. The SAD algorithm will search the square target region exhaustively to find a matching macroblock. The output of this procedure is the prediction error for motion compensation and the motion vector. The hardware implementation of the motion estimation block is sequential and contributes the largest delay to the critical path.

3.6. Entropy Coding

After DCT and quantization compression, additional compression can be achieved via entropy coding, which includes Huffman coding, Arithmetic coding, etc. Unlike lossy compression, as in the color space, DCT and quantization procedures, the entropy coding compression is lossless. The entropy coding efficiency depends on the precision of calculating the probability of occurrence of each coefficient. However, calculating probabilities of all the coefficients is impossible in real-time MPEG-4 coding and watermarking. The approach we followed is to utilize precalculated Huffman code tables for generic images [29].

4. The Proposed Video Watermarking Algorithm

This section presents a watermarking algorithm that performs the broadcaster's logo insertion as a visible watermark in the DCT domain. The robustness of DCT watermarking arises from the fact that if an attack tries to remove watermarking at mid-frequencies, it will risk degrading the fidelity of the image because some perceptive details are at mid-frequencies [29, 27]. The other important issue of visible watermarking, transparency, comes from making the watermark adaptive to the host frame [5]. The proposed watermarking algorithm is presented as a flow chart in figure. 2. For gray scale, the watermark is applied to Y frames. For a color image, the C_b and C_r color spaces are watermarked using the same techniques for Y frames. To protect against frame interpolating attacks, all I, B, and P frames must embed the watermark.

The watermark embedding model used in this paper was originally proposed by us in [5] for images:

$$C_W(i, j) = \alpha_n C(i, j) + \beta_n W(i, j), \tag{3}$$

where $C_W(i, j)$ is a DCT coefficient of watermarked images, α_n is the scaling factor, and β_n is the watermark strength factor, $C(i, j)$ is the original DCT coefficient, and $W(i, j)$ is the watermark DCT coefficient. The relative values of α_n and β_n determine the strength of the watermark. The coefficients are adaptive to image blocks and do not introduce any artifacts; therefore, this scheme is widely adopted in the literature. Their values are computed based on the characteristics of the host video frame.

Given that human perception is sensitive to image edge distortion, for edge blocks, the value of α_n should be close to its maximum value α_{max}, while the value of β_n should be close to its minimum value β_{min}. Since the watermark DCT

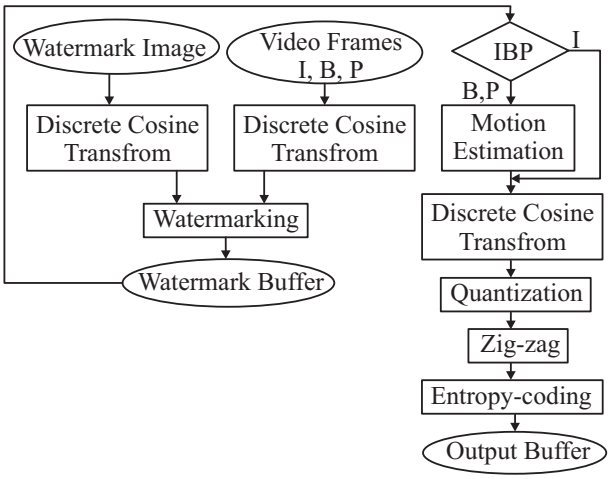


Figure 2: The flow of the proposed watermarking algorithm.

coefficients will be added to the video frame AC DCT coefficients, it will be advantageous to adjust the strength of the watermark such that the distortion of these coefficients is minimal. Given that AC coefficients of strongly textured blocks have small variance σ_n, it is desirable to make α_n proportional to σ_n , and β_n inversely proportional to σ_n. Therefore, for non-edge blocks the following models are used:

$$\begin{aligned} \alpha_n &= {\sigma^*}_n \times \exp\left(-\left({\mu^*}_n - \mu^*\right)^2\right), \\ \beta_n &= \left(\frac{1}{{\sigma^*}_n}\right) \times \left(1 - \exp\left(-\left({\mu^*}_n - \mu^*\right)^2\right)\right). \end{aligned} \tag{4}$$

The subscript n indicates the 8×8 block for which the parameters are being calculated. α_n and β_n in Equation 4 are calculated on a block-by-block basis, for a frame. ${\sigma^*}_n$ is the normalized natural logarithm of the variance of the block's AC DCT coefficients σ_n, given by:

$${\sigma^*}_n = \left(\frac{\ln(\sigma_n)}{\ln(\sigma_{max})}\right) \qquad \text{with} \tag{5}$$

$$\sigma_n = \left(\frac{1}{64}\right) \times \sum_{i=0}^{7} \sum_{j=0}^{7} \left(c_{ij} - \mu_n^{AC}\right). \tag{6}$$

In Equation 6, σ_{max} is the maximum value of all of the σ_n's in a frame, c_{ij} are the DCT coefficients, and μ_n^{AC} is the mean value of the AC DCT coefficients in block n. In Equation 4, ${\mu^*}_n$ is the normalized mean value of the DC DCT coefficient in

block n. μ^* is the normalized mean value of all $c_{00}(n)$ in a frame consisting of N 8×8 blocks. These are calculated as:

$$\mu^*_n = \left(\frac{c_{00}(n)}{c_{max}}\right) \quad \text{and} \tag{7}$$

$$\mu^* = \left(\frac{1}{N}\right) \times \sum_{n=1}^{N} c_{00}(n). \tag{8}$$

Once the intra-frame parameter issue is solved as above, the next challenge is their determination for inter-frames. There are several approaches, including the following: First, calculate the parameters for each frame on the fly. However, it is a fact that continuous, real-time calculation of the values of α_n and β_n for each block within each frame being watermarked is very expensive in terms of resource requirements and processing time. Second, predetermine the parameters for benchmark frames, store them in a buffer, and use them on the fly. The second alternative is followed in this paper. MATLAB® experiments are performed using video frames and values of α_n and β_n applied to all blocks in a frame. Based on these experiments, the constant values $\alpha_n = 0.9$, and $\beta_n = 0.1$ are selected for our hardware implementation. A variety of watermarked videos using these values do not display perceptible degradation in quality. However, the development of a video content adaptive technique is part of our ongoing research. The above models are thoroughly tested and proven to be working for a large variety of video data.

The complete watermark embedding process is shown in Algorithm 1.

5. The Proposed VLSI Architectures

The datapath architecture proposed in this work that can perform watermarking within the MPEG-4 video compression framework uses the components which are discussed in this section. The VLSI architecture for copyright protected MPEG-4 compression is shown in Figure 3. In the system architecture, the "watermark embedding" module performs the watermarking process. After that procedure, watermarked video frames are obtained. The rest of the units (e.g. entropy coding, zig-zag, quantization, DCT, and motion-estimation) of the architecture essentially perform MPEG-4 compression of the video. The system has a controller which generates addressing and control signals to synchronize all components of the system.

Algorithm 1 The proposed MPEG-4 watermarking algorithm.

1: Convert RGB color frames to YC_bC_r frames for a given host video.
2: Resample YC_bC_r frames according to 4:2:0 sampling rate.
3: Split Y frame and watermark image into 8×8 blocks.
4: Run 2-D DCT for each 8×8 block to generate 8×8 DCT coefficient matrix.
5: Watermark each 8×8 Y DCT matrix with an 8×8 watermark DCT matrix.
6: Perform 2-D IDCT for each 8×8 watermarked matrix to obtain the pixels.
7: Buffer watermarked Y frame, non watermarked C_b and C_r frames for a Group of Pictures (GOP, e.g., 15 continuous adjacent frames).
8: Split the Y frame into 16×16 blocks, and C_b and C_r into 8×8 blocks.
9: Perform motion estimation for Y frames. Each 16×16 Y block is rescaled to 8×8 blocks.
10: **if** (Even first frame (I) of GOP) **then**
11: **return** to Step 28.
12: **else if** (P) Frame **then**
13: **return** to Step 17.
14: **else if** (B) Frame **then**
15: **return** to Step 22.
16: **end if**
17: Perform Y frame forward or backward motion estimation P frames with reference frames (I or P frames). Obtain the motion vectors (MV) and prediction errors of residual frame for motion compensation (MC).
18: **if** (Y) Frame **then**
19: **return** to Step 28.
20: **end if**
21: Obtain C_b, C_r motion vectors and prediction errors. Go to Step 28.
22: Using bilinear algorithm motion, estimate B frames with two P frames or I and P frames for Y component. Obtain the motion vectors (MV) and prediction errors of residual frame for motion compensation (MC).
23: **if** (Y) Frame **then**
24: **return** to Step 28.
25: **else if** (C_b and C_r)frames **then**
26: **return** to Step 21.
27: **end if**
28: Perform 2-D DCT on blocks of frames and quantize the 2-D DCT matrix.
29: Zigzag scan quantized 2-D DCT coefficient Matrix.
30: Perform entropy coding of the 2-D DCT coefficients and motion vector.
31: Build structured MPEG-4 stream from buffer.

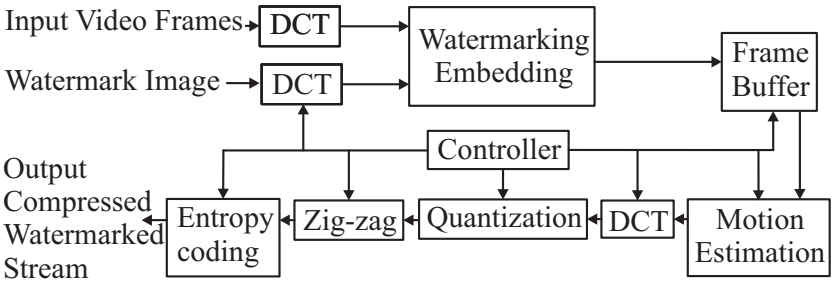


Figure 3: System-level architecture of MPEG-4 watermarking.

5.1. Watermark Insertion Unit

The watermark embedding or insertion unit is composed of several sub-modules, such as DCT, perceptual analyzer, edge detection, scaling factor, insertion, row and column address decoder, registers, and a local controller. The DCT module calculates the DCT coefficients of the host and watermark video frames before they are stored in the buffer memory. The controller schedules the operations of all other modules and the data flow in the watermarking unit. Address decoders are used to decode the memory address where the video frames and watermark frame are stored. This unit embeds a watermark image into a video frame. The input and output video frames are buffered to the frame buffer as shown in Figure 4(a).

5.2. Discrete Cosine Transformation (DCT) Unit

The DCT module calculates the DCT coefficients of the video frames and consists of two 1D DCT sub-modules. The algorithm of Loeffler *et al.* [28] is used in the implementation. The 1D row DCT of each 8×8 block is first computed. The column DCT of each block is then performed. A buffer is used to assist in finding the transpose of the 1D row DCT. The final controller for the watermarking unit controls the DCT module. The buffer stores the 1D row DCT coefficient before the column DCT is computed. The block diagram of the DCT module is shown in Figure 4(b). The 2-D DCT has a 12-bit data bus and a 6-bit address bus for the 64-byte internal buffer. The input data is an 8-bit unsigned integer and the output is a 12-bit integer. For higher precision, the bit length is increased.

5.3. Frame Buffer

The frame-buffer is responsible for buffering the frames for every block procedure module, e.g., the watermark, DCT, and motion-estimation units. Its size

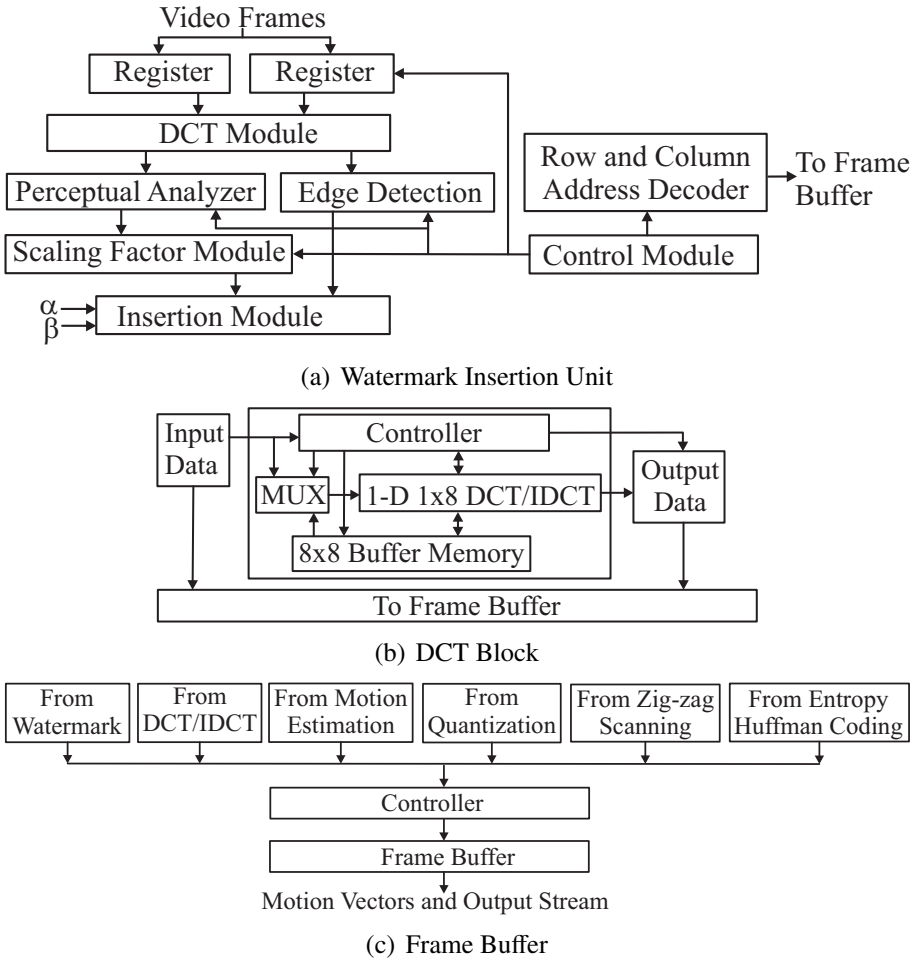


Figure 4: The Proposed Architectures of various Datapath Components.

capacity is enough for one input group of pictures (GOP), motion vectors, and the output stream. The frame-buffer shown in Figure 4(c) is an external buffer which is different from the block-memory RAM used by the motion-estimation module.

5.4. Motion Estimation Module

The motion estimation module is composed of motion detection and half-pel modules. The macro block motion detection core in Figure 5(a) performs a search for the best match for each macro block in the current frame based on a 3×3 macro block area in the previous frame. It is intended to be used as part of a larger video compression system. Pixel data is input 8 bits at a time in 4:2:2 YUV order

at 27 MHz. Data is separated by component type, stored in off-chip RAM, and macro blocks are processed one at a time. Each macro block results in a half-pel motion vector and a set of differences that could be sent on to additional stages for encoding. The core is intended to be used in real time encoding of full-size NTSC. The motion detection core uses 16 block RAM modules, and half-pel uses 9 block RAM modules to do the exhaustive search.

5.5. Entropy Coding Unit

The architecture of the entropy coding unit is shown in Figure 5(b). It is implemented using a Huffman coding look-up table. It has many different submodules including variable length coding (VLC) and pattern matching.

5.6. Quantization Unit

The quantization module architecture is shown in Figure 5(c). This module quantizes the DCT coefficients according to predefined quantization tables. The input and output are buffered in the frame buffer.

5.7. Zig-Zag Scanning Unit

The architecture of the zigzag scanning unit is shown in Figure 5(d). This unit performs re-ordering of the DCT coefficients of video frames.

5.8. Overall Datapath

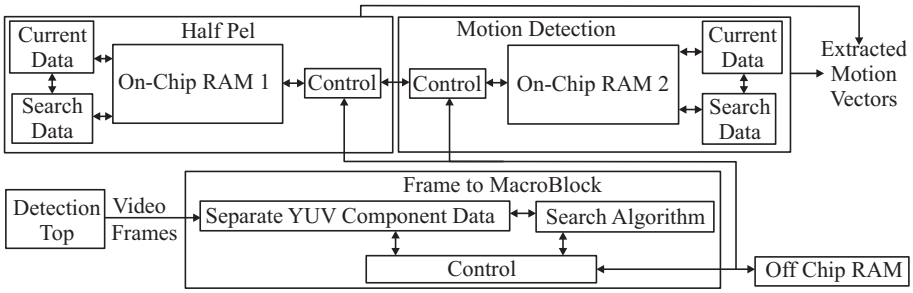
The overall datapath architecture that can perform watermarking in the MPEG-4 video compression framework is shown in Figure 6. The datapath is constructed by stitching various individual units together.

5.9. Overall Controller

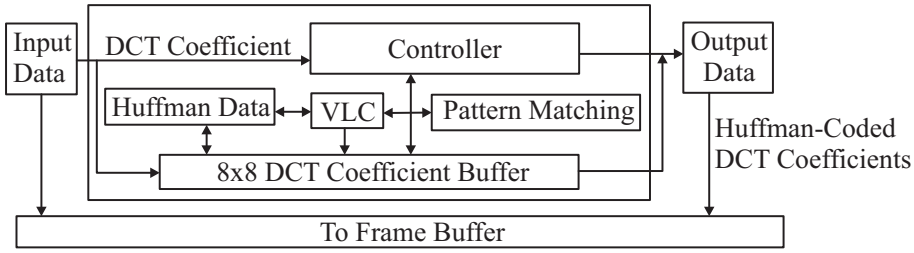
The overall controller of the system to synchronize system functions is shown in Figure 7. The controller generates address and control signals to synchronize the different components of the datapath for their coordinated processing.

6. Architecture Modeling and Prototyping

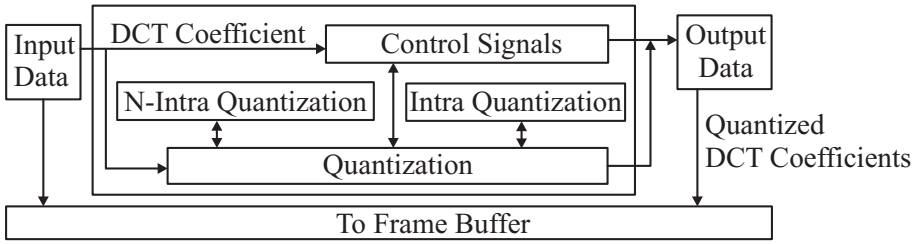
In this section the two different ways of modeling the proposed architecture are presented. First, Simulink® based modeling for functional verification of the architectures is discussed. Then, VHDL-based modeling is presented and is synthesized for an FPGA platform.



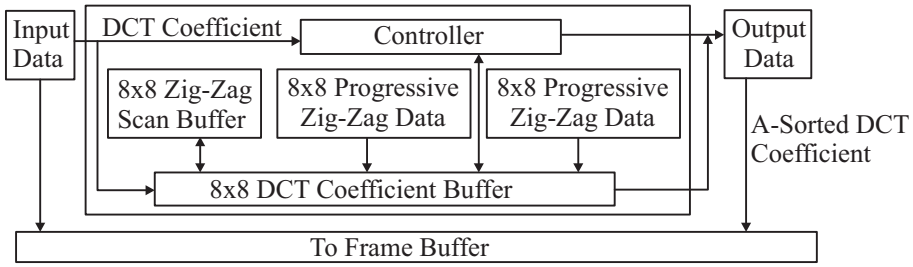
(a) Motion Estimation Unit



(b) Entropy Encoding Unit



(c) Quantization Unit



(d) Zig-Zag Unit

Figure 5: The Proposed Architectures of various Datapath Components (cont.).

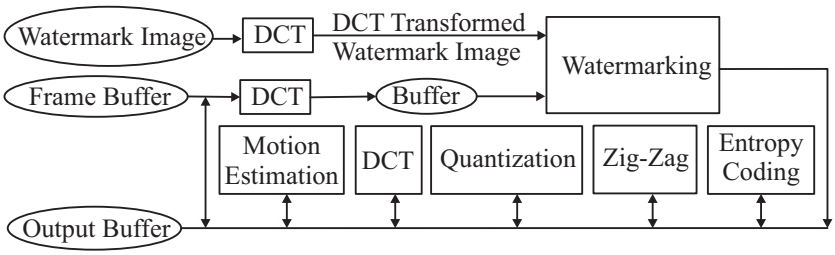


Figure 6: Datapath for MPEG-4 watermarking (bus width 12 bits).

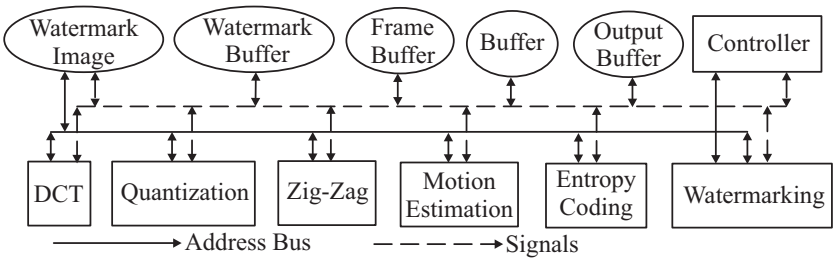


Figure 7: System address bus and signals.

6.1. Simulink® Based Modeling

To verify the functionality of the algorithms and architecture presented in the previous sections, a fast prototype is built in MATLAB/Simulink®. The methodology for this high level system modeling is bottom-up: building function units first, then integrating these units into subsystems, assembling the subsystems into a complete system, and, finally, verifying overall system functionality.

MATLAB/Simulink® offers video and image processing functions and modules that facilitate fast prototyping. Available function units include DCT/IDCT, SAD for motion estimation, block processing (split), and delay (buffer). In addition, quantization, zigzag scanning, and entropy coding modules were built. The system-level modeling is accomplished using different modules as follows: (1) Module 1: Color Conversion and sampling rate compression, (2) Module 2: DCT domain compression in each frame, (3) Module 3: Quantization and zigzag scanning, (4) Module 4: Entropy coding using Huffman codes, (5) Module 5: Motion estimation and compensation only on I and P frames, (6) Module 6: Interpolating B frames, and (7) Module 7: Uncompressed domain watermarking. The MATLAB/Simulink®-based representation of the integrated watermarking MPEG-4 system is shown in Figure 8, and the details of the watermarking insertion unit are presented in Figure 9.

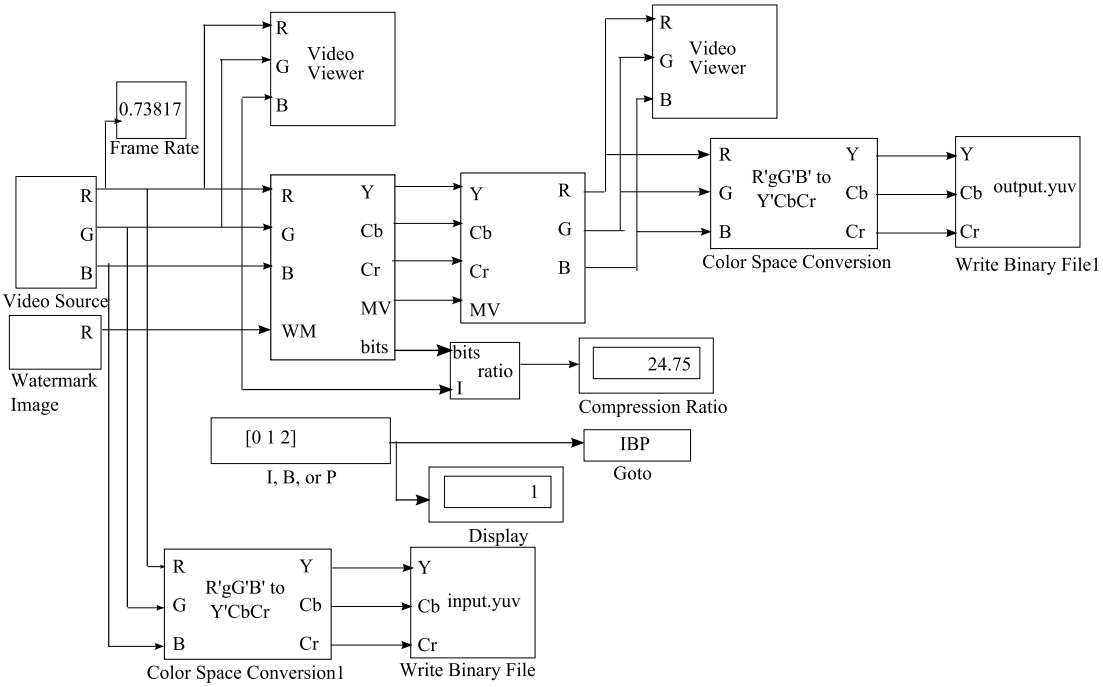


Figure 8: Simulink block diagram of the proposed system.

As seen from Figure 9, the video frames are watermarked in the DCT domain before being compressed. For three (Y, C_r, C_b) color frames, only the Y color frame is watermarked for the following reasons:

1. The watermark image, which is black-white monochrome or gray scale, should only modify the brightness of the picture being watermarked. If the watermark is in color, the C_b and C_r frames must be watermarked as well.
2. The Y color space is more sensitive to human perception such that any unauthorized modification will be easily detected. This makes the Y component of color frames ideal for watermark-based copyright protection.
3. To avoid redundancy, the watermark will not be embedded into the C_b or C_r frames. However, they can also be watermarked for high perceptual quality watermarking with only slight modifications to the architecture.

To protect against frame interpolating attacks, all I, B, and P frames must embed the watermark. Exhaustive simulations are performed to verify the proposed algorithms and architectures with a large variety of watermark images and video clips. Sample watermarked video clips are presented in the experimental results

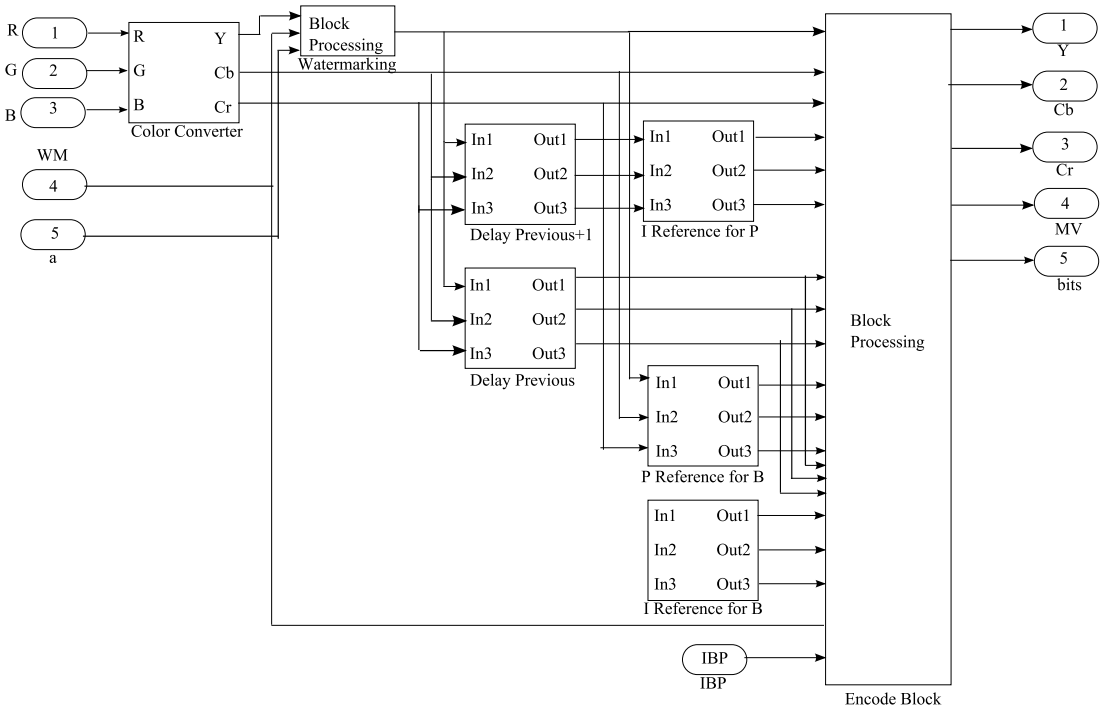


Figure 9: Simulink simulation of the Encoder.

section.

6.2. FPGA Based Prototyping

The VLSI architectures presented here were designed in the form of soft-cores, i.e., in the form of hardware description language modules, such as Verilog and VHDL, to make the digital rights management (DRM) technology available for diverse electronic appliances that can generate the video. During the FPGA-based prototyping development, the working video compression and watermarking modules are implemented in VHDL. A top-down modular design flow for performing the architectural design and simulation and FPGA prototyping is presented in Figure 10. In this approach, the architecture unit is logically and structurally divided into several modules first. Each of the modules is individually tested and verified through simulation and synthesized from VHDL into register transfer level (RTL) form. Once the individual modules are tested and verified to be functionally correct, they are stitched together. Next, the controller is designed that executes the datapath and ensures that the unit performs its assigned operations. The VHDL code was compiled using Altera® Quartus with a Cyclone II FPGA chip as the

target for synthesis. The individual modules are: frame buffer, watermarking, DCT, Quantization, and Zig-Zag. The controller is realized as a finite state machine (FSM), as shown in Figure 11. In this FSM, several sub-states have been merged for simplicity of design.

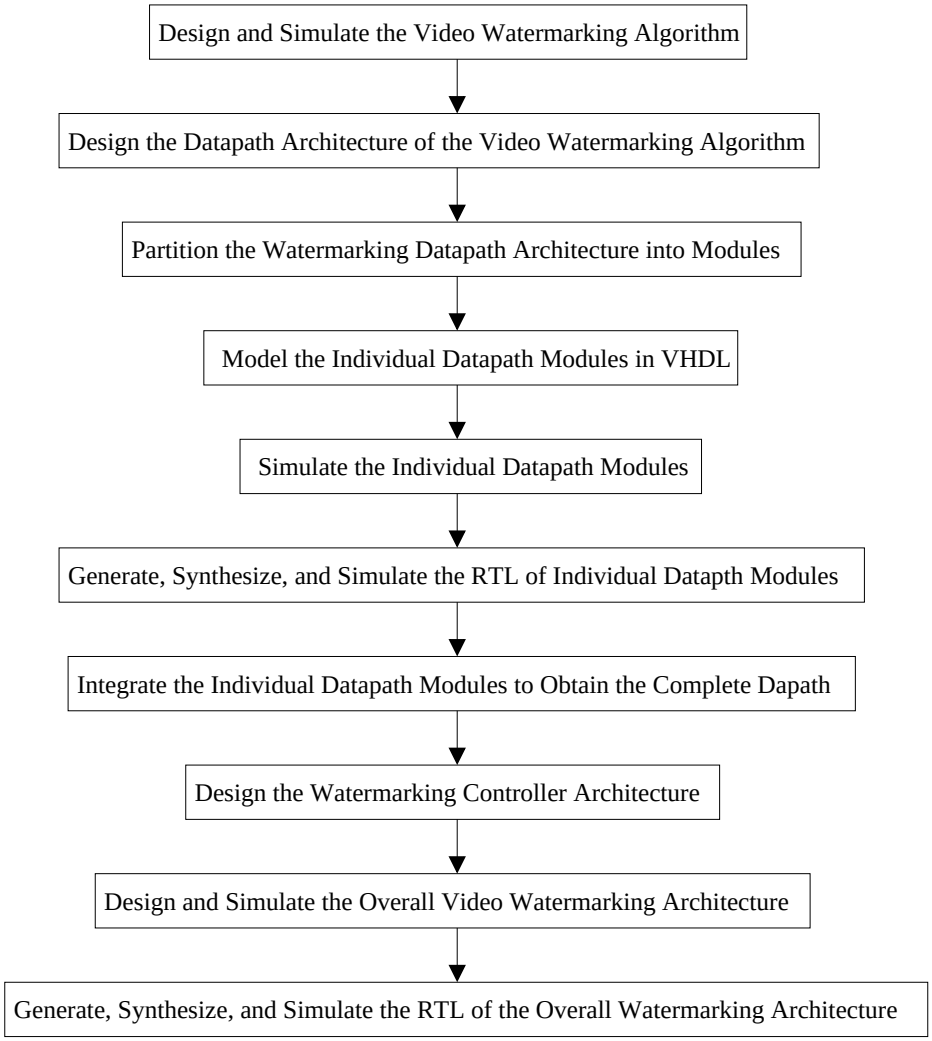
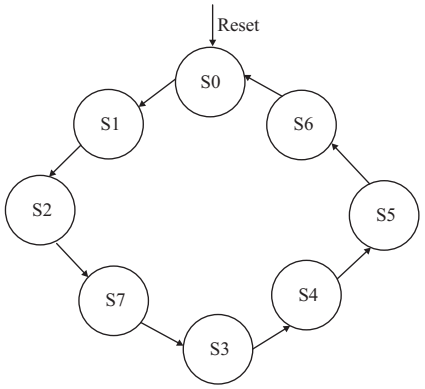


Figure 10: The Design Flow used for the FPGA-Based Prototyping of the Video Watermarking Architectures.

Results for the processing of a (Y, C_r, C_b) frame using Quartus Cyclone-II synthesis tools are shown in Table 1. The motion estimation block currently uses 54% of logic resources but there is sufficient room to add all the other compo-



(a) States of the controller

S0:	1. Video Frame to DCT 2. Watermarking to DCT 3. Watermarking to DCT to Watermark Buffer
S1:	1. Watermark Buffer to IDCT 2. IDCT to Video Frame
S2:	1. Motion Estimation P Frame to Output Buffer
S7:	1. Motion Estimation B Frame to Output Buffer 2. IDCT to Video Frame
S3:	1. Output Buffer to DCT/Quantization 2. DCT/Quantization to Output Buffer
S4:	1. Output Buffer to Zig-Zag Scanning 2. Zig-Zag Scanning to Output Buffer
S5:	1. Output Buffer to Huffman Coding 2. Huffman Coding to Output Buffer
S6:	1. Set End Signal. Back to S0

(b) Finite State Machine of the Controller

Figure 11: State diagram of the controller.

nents (DCT, Entropy, Quantization, Zig-Zag, etc.) and fit an entire system on this FPGA.

We also performed Hardware-In the-Loop (HIL) simulation of the watermarking process using Simulink®, the FPGA board and Altera's® DSP Builder block-set and libraries. In this process, a VHDL description is automatically generated for the DSP Builder blocks and downloaded to the FPGA. To set up the HIL, a QuartusII project is created to synthesize the VHDL code generated. The next step after running the QuartusII project and VHDL synthesis is to add a clock block from the DSP Builder block-set (which is used as the simulation master clock, independent of the FPGA's clock) and use the synthesis fitter to run the QuartusII assembler. After the synthesis and assembler are performed successfully, a new Simulink® file is created and an HIL DSP builder block is inserted. At this point we have a block with the necessary logic, inputs and outputs to operate as the DSP Builder Blocks. Finally, all the input and output terminals are connected appropriately and input the right signal sources and sinks (derived from the Matlab® workspace). The HIL simulations were performed on a state-of-the-art workstation with Intel® Core i7-950 3.06 GHz CPU and 12 GB of RAM (the simulations were performed in 32-bit mode since the DSP Builder tool is unavailable in a 64-bit version at the time this work is being performed). Typical runtimes were approximately 10 minutes for 1 sec. of video. The excessive simulation time is due to the constant communication between Simulink® and the FPGA

Table 1: Resource Utilization Report for a 128×128 Color Frame

Components	Logic Elements	Registers	MUXes
2D DCT	1477	157	0
Quantization	2363	0	1
Zigzag	1030	786	0
Watermark	24	0	0
Frame Buffers	7713	6156	0
Motion Vector Buffer	667	520	0
Watermark Buffer	4043	3048	0
RGB to YC_bC_r	1416	0	8
Motion Estimation	8987	4900	
Controller	575	157	0
Overall Architecture	28322	16532	9

board via USB and the JTAG interface which utilizes a very high overhead protocol. Though clearly unsuitable for real-time implementation, the HIL approach is very effective in verifying the overall system functionality before the entire design is downloaded into the hardware. It also provides a very powerful interface for debugging during the design stage as entire system sub-block input and output vectors can be examined and manipulated in Matlab® during the actual operation of the hardware. This HIL approach is also substantially more powerful than using traditional logic analyzers (most of the sub-block signals are not accessible) or IP-based on-chip scopes (such as Altera's® SignalTap II) for which there are no hardware resources left following the implementation of complex algorithms such as MPEG-4.

The HIL resource usage for the watermark insertion phase is shown in Table 2. The other phases of the overall MPEG-4 watermarking are similarly performed, however not tabulated here for brevity.

7. Experimental Results

This section discusses the experiments performed to test the performance of the watermarking algorithm and architecture proposed in the previous sections.

7.1. Experimental Setup

The proposed watermarking algorithm is first tested on a PC. The high-level system modeling was performed with MATLAB/Simulink® following a bottom-

Table 2: Resource Usage in Hardware-In the-Loop (HIL) Simulation of the Watermark Insertion

Total Logic Elements	21.819/33.216
Total Combinational Functions	15.269/33.216
Dedicated Logic Registers	19.651/33.216
Total Number of Registers	8035
Total Memory Bits	347.218/483.840
Embedded Multiplier 9-bit Elements	45/70
Total PLLs	0/4

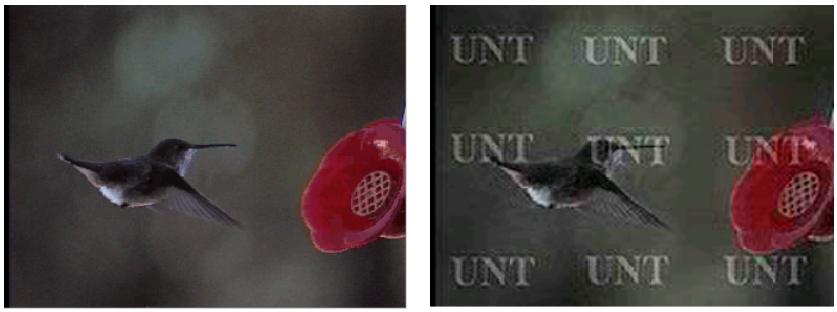
up approach. In this case, all individual components are first designed and tested and then integrated to build a complete system model. The overall system is finally verified for its functionality. This algorithm operates in the uncompressed domain only. As MATLAB/Simulink® has built-in video and image processing modules, it was straightforward to build the various component prototypes.

7.2. Testing of Watermarking Quality

Exhaustive simulations are performed to verify the proposed algorithms and architectures with a large variety of watermark images and video clips. For brevity, selected examples of watermarked video are presented in Figures 12, 13, 14, and 15. A different sequence of AVI [11] video clips and different watermark images, all having the same dimensions of 320×240, were used for the experiments. The video frames are watermarked in the DCT domain before being compressed. Out of three (Y, C_r, C_b) color frames, only the Y color frame is watermarked, as the watermark image, which is monochrome or grayscale, only modifies the brightness of the video frame. If the watermark is in color then C_b and C_r must be watermarked as well. To avoid redundancy, the watermark was not embedded into C_b or C_r. As the Y color space is more sensitive to human perception, any unauthorized modification will be easily detected. This makes the Y color frame ideal to watermark for copyright protection.

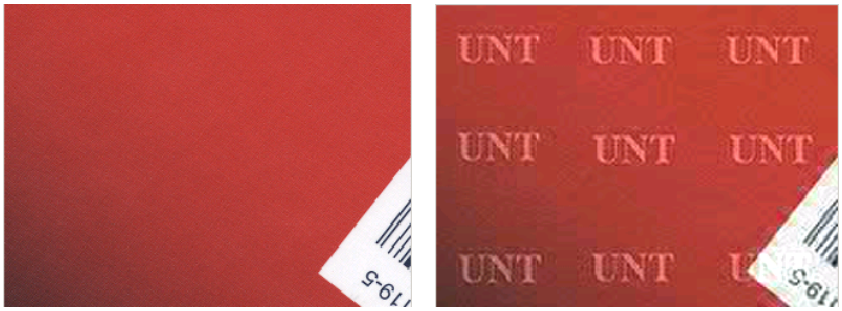
The total processing time for 3 frames (Y, C_b and C_r) is 1.07 msec or 932 frames/sec. If the system is utilized for high-resolution applications, such as the NTSC television video broadcasting system, the peak processing speed is 43 frames/sec, which exceeds the required 29.97 frames/sec.

Standard video quality metrics such as the Mean Square Error (MSE) and Peak-Signal-to-Noise-Ratio (PSNR) [8, 29, 5] are applied to quantify the system's performance. The MSE and PSNR [30, 26, 29, 28] are expressed by the following



(a) Original bird video (b) Watermarked bird video

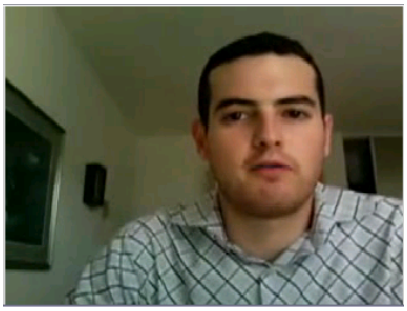
(c) Original dinner video (d) Watermarked dinner video



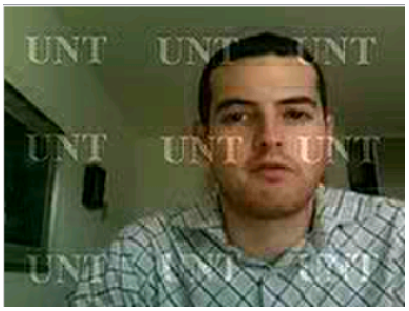
(e) Original barcode video (f) Watermarked barcode video

(g) Original talent video (h) Watermarked talent video

Figure 12: Sample watermarked video - 1 using watermark - 1.

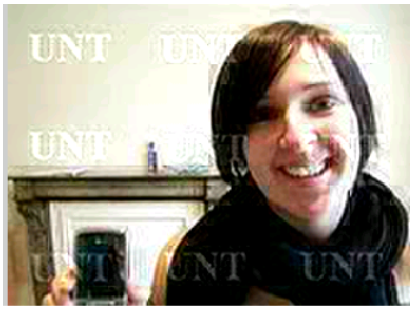


(a) Original Iphone Review video



(b) Watermarked Iphone Review video

(c) Original LGphone video



(d) Watermarked LGphone video

(e) Original Train video

(f) Watermarked Train video

Figure 13: Sample watermarked video - 2 using watermark - 1.

(a) Original bird video

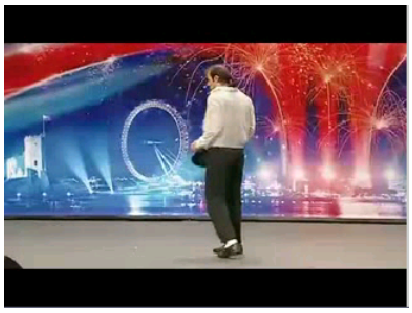
(b) Watermarked bird video

(c) Original dinner video

(d) Watermarked dinner video

(e) Original barcode video

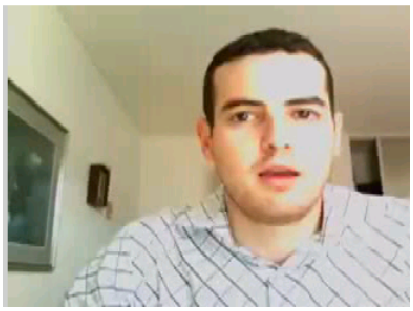
(f) Watermarked barcode video



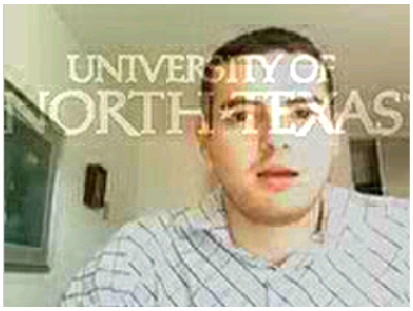
(g) Original talent video

(h) Watermarked talent video

Figure 14: Sample watermarked video - 1 using watermark - 2.



(a) Original Iphone video



(b) Watermarked Iphone video

(c) Original LGphone video

(d) Watermarked LGphone video

(e) Original Train video

(f) Watermarked Train video

Figure 15: Sample watermarked video - 2 using watermark - 2.

Table 3: Video quality metrics of video compression and watermarking using watermark - 1

Clips	PSNR(dB)	RMSE	Compression Ratio		
			Average	Range	Estimate
Bird	21.15	22.34	21.46	16~39	16
Dinner	22.22	19.74			
Barcode	22.77	18.53			
Talent	21.82	23.17			
Iphone	22.20	20.10			
LGphone	22.82	23.17			
Train	21.75	22.11			

equations:

$$MSE = \left(\frac{\sum_{m=1}^{M} \sum_{n=1}^{N} \sum_{k=1}^{3} |p(m,n,k) - q(m,n,k)|^2}{3M \times N} \right), \tag{9}$$

$$PSNR = 10 \log_{10} \left(\frac{(2^i - 1)^2}{\text{MSE}} \right). \tag{10}$$

where m is the image pixel row from 1 to M, n is the image pixel column from 1 to N, and k is the index (1 to 3 for RGB color space) corresponding to the color plane. $p(m,n,k)$ and $q(m,n,k)$ are the images' pixels after and before processing, respectively, and i is the bit length of the image pixel, which is 8 in RGB systems.

The quality metrics of video compression and watermarking in the working model are displayed in Table 3 and Table 4 for two different watermarks added to the video frames, respectively. The criteria of video quality are as follows: for a PSNR between 40 dB to 50 dB, the noise will be beyond human perception; however, for PSNR between 10 dB to 20 dB, the noise is detected by the human visual system [31]. The integrated watermark video system generates video with an average PSNR of 30 dB, which implies that the implementation of MPEG-4 video compression is perceptually of high quality. The low PSNR did not degrade the perceptual quality of the video; rather, the low PSNR value is due to the fact that the watermark logo inserted is visible and consequently becomes noise for the host video, affecting the PSNR. The results are consistent with other visible watermarking algorithms and architectures available in the current literature [8, 5].

The video compression rate consists of two components: the constant component, from the 4:2:0 color space sampling rate, whose compression rate is always 2:1, and the content adaptive compression component, whose rate is variable and

Table 4: Video quality metrics of video compression and watermarking using watermark - 2

Clips	PSNR(dB)	RMSE	Compression Ratio		
			Average	Range	Estimate
Bird	21.70	20.97	21.22	16~39	16
Dinner	21.94	20.39			
Barcode	21.40	21.70			
Talent	21.14	20.45			
Iphone	21.90	21.00			
LGphone	21.39	22.24			
Train	22.33	23.88			

depends on the content data such as motion estimation, DCT coefficients, quantization, and Huffman coding. To estimate the variable compression rate we assume that half of the DCT coefficients are truncated so that the compression rate is 2:1. The redundancy of two frames by the motion estimation results in a compression rate of 4:1. In the working module, one GOP is comprised of one I frame, one B frame, and one P frame or IBP structure. The motion estimation compression is estimated as $(1 + 1 + 1)/(1 + 1/4 + 1/16) \approx 2 : 1$. The DCT coefficient quantization and Huffman coding have compression rates approximately 2:1. Hence the estimated average compression rate of the video compression working module is 16:1. The compression rate obtained from our experimental results is 27:1. To achieve a higher compression rate, one approach is to interpolate more B frames and more P frames in one GOP. After tuning, the average compression rate could be greater than 100:1.

7.3. Comparison With Existing Research

In order to obtain a broad perspective on the quality of the watermarking algorithm and FPGA prototype given in this paper, performance statistics with reference to existing hardware based watermarking for video are presented here. A comparative view is provided in Table 5. The research works are arranged according to their working domain, e.g., spatial, DCT, wavelet, etc. It is noted that of all the research presented, the current system is the only one capable of achieving real-time video watermarking and compression at rates exceeding existing broadcast standards.

Table 5: Video watermarking hardware proposed in the existing literature

Research Works	Design Type	Different Types	Working Domain	Chip Statistics
Strycker, et. al. [16, 15]	DSP board	Invisible Robust	Spatial Fourier	100 MHz
Maes, et. al. [18]	FPGA board Custom IC	Invisible Robust	Spatial Fourier	17 kG Logic 14 kG Logic
Tsai and Wu [13]	Architecture	Invisible Robust	Spatial Spatial	NA NA
Brunton and Zhao [19]	GPU	Invisible Fragile	Spatial	NA
Jeong, et al. [20]	FPGA	Invisible Robust	Spatial	ALTERA STRATIX
Mathai, et. al. [21, 22]	Custom IC	Invisible Robust	Wavelet	0.18μm, 3.53 mm^2, 75 MHz, 160 mW
Vural, et. al. [23]	Architecture	Invisible Robust	Wavelet	NA
Jeong, et al. [24]	FPGA	Invisible Robust	Wavelet	XILINX VERTEX2
Petitjean, et. al. [25]	FPGA board DSP board	Invisible Robust	Fractal	50 MHz takes 6 μs 250 MHz takes 118 μs
This Paper	FPGA	Visible	DCT	100 MHz, 43 fps

8. Conclusions

This paper presented a visible watermarking algorithm and prototyped it using FPGA technology for MPEG-4 video compression. The algorithm and its implementation are suitable for real-time applications such as video broadcasting, IP-TV, and digital cinema. The watermark is embedded before video compression, thus resulting in balanced quality and performance. Our implementation using standard FPGAs demonstrates its suitability for standard NTSC television. The algorithm achieved peak performance of 43 frames/sec and a PSNR of 30 dB. Further development is under way to extend the real-time performance of the system to HDTV and higher resolutions and to improve the PSNR towards the $40-50$ dB range. To this end the following extensions to this research are planned: (1)

Realization of the watermark embedding in the compressed domain. Even though the hardware requirements will increase, it is anticipated that the quality of the watermarked video will improve, particularly at high resolutions. (2) Utilization of advanced MPEG-4 features, such as N-bit resolution, advanced scalable textures, and video objects. It is anticipated that, with modest hardware complexity increase, performance will be significantly improved with the inclusion of these additional features. (3) RTL-level subsystem optimization to improve resource utilization and minimize execution time. (4) Alternative hardware architectures using on-board memory and pipelining will also be considered.

Acknowledgment

The authors would like to thank Wei Cai and Manish Ratnani, graduates of the University of North Texas. This archival journal paper is based on our previous conference publication [32].

References

[1] L. D. Strycker, P. Termont, J. Vandewege, J. Haitsma, A. Kalker, M. Maes, G. Depovere, Implementation of a Real-Time Digital Watermarking Process for Broadcast Monitoring on Trimedia VLIW Processor, IEE Proceedings on Vision, Image and Signal Processing 147 (4) (2000) 371–376.

[2] N. J. Mathai, D. Kundur, A. Sheikholeslami, Hardware Implementation Perspectives of Digital Video Watermarking Algortithms, IEEE Transanctions on Signal Processing 51 (4) (2003) 925–938.

[3] O. B. Adamo, S. P. Mohanty, E. Kougianos, M. Varanasi, VLSI Architecture for Encryption and Watermarking Units Towards the Making of a Secure Digital Camera, in: Proceedings of the IEEE International SOC Conference, 2006, pp. 216–217.

[4] S. P. Mohanty, O. B. Adamo, E. Kougianos, VLSI Architecture of an Invisible Watermarking Unit for a Biometric-Based Security System in a Digital Camera, in: Proceedings of the 25th IEEE International Conference on Consumer Electronics, 2007, pp. 485–486.

[5] S. P. Mohanty, K. R. Ramakrishnan, M. S. Kankanhalli, A DCT Domain Visible Watermarking Technique for Images, in: Proceedings of the IEEE International Conference on Multimedia and Expo, 2000, pp. 1029–1032.

[6] S. P. Mohanty, R. Sheth, A. Pinto, M. Chandy, CryptMark: A Novel Secure Invisible Watermarking Technique for Color Images, in: Proceedings of the 11th IEEE International Symposium on Consumer Electronics, 2007, pp. 1–6.

[7] A. Garimella, M. V. V. Satyanarayan, R. S. Kumar, P. S. Murugesh, U. C. Niranjan, VLSI Impementation of Online Digital Watermarking Techniques with Difference Encoding for the 8-bit Gray Scale Images, in: Proceedings of the International Conference on VLSI Design, 2003, pp. 283–288.

[8] S. P. Mohanty, N. Ranganathan, R. K. Namballa, A VLSI Architecture for Visible Watermarking in a Secure Still Digital Camera (S^2DC) Design, IEEE Transactions on Very Large Scale Integration Systems 13 (8) (2005) 1002–1012.

[9] M. Barni, F. Bartolini, N. Checcacci, Watermarking of MPEG-4 video objects, IEEE Transactions on Multimedia 7 (1) (2005) 23–31.

[10] S. Biswas, S. R. Das, E. M. Petriu, An adaptive compressed MPEG-2 video watermarking scheme, IEEE Transactions on Instrumentation and Measurement 54 (5) (2005) 1853–61.

[11] Xvid Codec, `http://www.xvid.org`.

[12] E. Kougianos, S. P. Mohanty, R. N. Mahapatra, Hardware Assisted Watermarking for Multimedia, Special Issue on Circuits and Systems for Real-Time Security and Copyright Protection of Multimedia, Elsevier International Journal on Computers and Electrical Engineering (IJCEE) 35 (2) (2009) 339–358.

[13] T. H. Tsai, C. Y. Wu, An Implementation of Configurable Digital Watermarking Systems in MPEG Video Encoder, in: Proceedings of the IEEE International Conference on Consumer Electronics, 2003, pp. 216–217.

[14] Y. C. Fan, L. D. Van, C. M. Huang, H. W. Tsao, Hardware-Efficient Architecture Design of Wavelet-based Adaptive Visible Watermarking, in: Proceedings of 9th IEEE International Symposium on Consumer Electronics, 2005, pp. 399–403.

[15] L. D. Strycker, P. Termont, J. Vandewege, J. Haitsma, A. Kalker, M. Maes, G. Depovere, An Implementation of a Real-time Digital Watermarking Process for Broadcast Monitoring on a Trimedia VLIW Processor, in: Proceedings of the IEE International Conference on Image Precessing and Its Applications (Vol. 2), 1999, pp. 775–779.

[16] L. D. Strycker, P. Termont, J. Vandewege, J. Haitsma, A. Kalker, M. Maes, G. Depovere, Implementation of a Real-Time Digital Watermarking Process for Broadcast Monitoring on Trimedia VLIW Processor, IEE Proceedings on Vision, Image and Signal Processing 147 (4) (2000) 371–376.

[17] P. Termont, L. D. Strycker, J. Vandewege, J. Haitsma, T. Kalker, M. Maes, G. Depovere, A. Langell, C. Alm, P. Norman, Performance Measurements of a Real-time Digital Watermarking System for Broadcast Monitoring, in: Proceedings of the IEEE International Conference on Multimedia Computing and Systems (Vol. 2), 1999, pp. 220–224.

[18] M. Maes, T. Kalker, J. P. M. G. Linnartz, J. Talstra, G. F. G. Depovere, J. Haitsma, Digital Watamarking for DVD Video Copyright Protection, IEEE Signal Processing Magazine 17 (5) (2000) 47–57.

[19] A. Brunton, J. Zhao, Real-time Video Watermarking on Programmable Graphics Hardware, in: Proceedings of Canadian Conference on Electrical and Computer Engineering, 2005, pp. 1312–1315.

[20] Y.-J. Jeong, W.-H. Kim, K.-S. Moon, J.-N. Kim, Implementation of Watermark Detection System for Hardware Based Video Watermark Embedder, in: Proceedings of the Third International Conference on Convergence and Hybrid Information Technology (ICCIT), 2008, pp. 450 – 453.

[21] N. J. Mathai, A. Sheikholeslami, D. Kundur, VLSI Implementation of a Real-Time Video Watermark Embedder and Detector, in: Proceedings of the IEEE International Symposisum on Circuits and Systems (Vol. 2), 2003, pp. 772–775.

[22] N. J. Mathai, D. Kundur, A. Sheikholeslami, Hardware Implementation Perspectives of Digital Video Watermarking Algortithms, IEEE Transanctions on Signal Processing 51 (4) (2003) 925–938.

[23] S. Vural, H. Tomii, H. Yamauchi, Video Watermarking For Digital Cinema Contents, in: Proceedings of the 13th European Signal Processing Conference, 2005, pp. 303–304.

[24] Y.-J. Jeong, K.-S. Moon, J.-N. Kim, Implementation of Real Time Video Watermark Embedder Based on Haar Wavelet Transform Using FPGA, in: Proceedings of the Second International Conference on Future Generation Communication and Networking Symposia (FGCNS), 2008, pp. 63 – 66.

[25] G. Petitjean, J. L. Dugelay, S. Gabriele, C. Rey, J. Nicolai, Towards Real-time Video Watermarking for Systems-On-Chip, in: Proceedings of the IEEE International Conference on Multimedia and Expo (Vol. 1), 2002, pp. 597–600.

[26] J. Chen, et al., Design of Digital Video Coding Systems - A Complete Compressed Domain Approach, Marcel Dekker, New York, 2002.

[27] W. Cai, FPGA Prototyping of a Watermarking Algorithm for MPEG-4, Master's thesis, Department of Engineering Technology, University of North Texas (Spring 2007).

[28] C. Loeffler, et al., Practical fast 1-D DCT algorithms with 11 multiplications, in: Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, 1989, pp. 988–991.

[29] M. Barni, et al., A DCT-domain system for robust image watermarking, Signal Processing 66 (3) (1998) 357–372.

[30] L. Qiao, K. Nahrstedt, Watermarking Methods for MPEG Encoded Video: Towards Resolving Rightful Ownership, in: Proceedings of the IEEE International Conference on Multimedia Computing and Systems, 1998, p. 276.

[31] I. E. G. Richardson., H.264 and MPEG-4 Video Compression, John Wiley and Sons, Ltd, England, 2003.

[32] S. P. Mohanty, E. Kougianos, W. Cai, M. Ratnani, VLSI Architectures of Perceptual Based Video Watermarking for Real-Time Copyright Protection, in: Proceedings of the 10th IEEE International Symposium on Quality Electronic Design (ISQED), 2009, pp. 527–534.