Real-time perceptual watermarking architectures for video broadcasting

Saraju P. Mohanty*, Elias Kougianos

NanoSystem Design Laboratory (NSDL), University of North Texas, Denton, TX 76207, USA

**ARTICLE INFO**

Article history:
Received 21 June 2010
Received in revised form 30 October 2010
Accepted 9 December 2010
Available online 24 December 2010

Keywords:
VLSI architecture
Video broadcasting
IP-TV
Watermarking
Content protection
Copyright protection
Real-time security
Digital rights management
Multimedia security

**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 broadcaster’s 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/s 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.

© 2010 Elsevier Inc. All rights reserved.

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 Fig. 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 (Strycker et al., 2000; Mathai et al., 2003a) concentrates on invisible watermarking, which cannot be used for logo insertion. Existing research on visible watermarking (Adamo et al., 2006; Mohanty et al., 2000, 2005, 2007a, 2007b; Carimella et al., 2003) is primarily for images, not video. The research works presented in Barni et al. (2005) and Biswas et al. (2005) are for video, but they cannot 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.

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 (Xvid Codec, 2010). 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 (Strycker et al., 2000; Mathai et al., 2003a; Garimella et al., 2003; Kougianos et al., 2009; Tsai and Wu, 2003; Fan et al., 2005). 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 (Strycker et al., 1999, 2000; Tertm et al., 1999), 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 pseudorandom 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 Maes et al. (2000), the Millenium 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 Tsai and Wu (2003), a VLSI architecture for a spread-spectrum based real-time watermarking system is presented. In Brunton and Zhao (2005), the graphics processing unit (GPU) is utilized for hardware assisted real-time watermarking. In Jeong et al. (2008a), 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 Mathai et al. (2003b, 2003c), a custom IC for video watermarking is presented that performs all operations using floating-point datapath architectures for both the watermarking embedder and the detector. The embedder consumes 60 mW, and the detector consumes 100 mW while operating at 75 MHz and 1.8 V. In Vural et al. (2005), a video watermarking system called “traceable watermarking” is proposed for digital cinema. In Jeong et al. (2008b), FPGA prototyping is presented for HAAR-wavelet based real time video watermarking. In Petitjean et al. (2002), 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 Transformation (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 YCbCr-color space is performed using the following expression (Chen et al., 2002; Cai, 2007):

$$
\begin{align*}
Y &= 0.299R + 0.587G + 0.114B, \\
C_b &= 0.564(B - Y) + 128, \\
C_r &= 0.714(R - Y) + 128.
\end{align*}
$$

A total of six adders and five 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 (Cai, 2007; Loeffler et al., 1989) 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 (Barni et al., 1998).

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 Cai (2007) and Barni et al. (1998).

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, diagonal, 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 Cai (2007) and Barni et al. (1998):

$$\text{SAD}(x, y) = \left\{ \begin{array}{ll} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |c(i, j) - p(i, j)| & \text{for } (x, y) = (0, 0), \\ \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |c(i, j) - p(i+x, j+y)| & \text{for } (x, y) \neq (0, 0). \end{array} \right.$$ (2)

where \(c(i, j)\) are the pixels of the current block, \(i, j = 0, 1, \ldots, N-1\); \(p(m, n)\) are the pixels of the previous block in the searching region, and \(m, n = -R, -R+1, \ldots, 0, 1, \ldots, R+N-1\), 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 pre-calculated Huffman code tables for generic images (Barni et al., 1998).

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 perceptible details are at mid-frequencies (Cai, 2007; Barni et al., 1998). The other important issue of visible watermarking, transparency, comes from making the watermark adaptive to the host frame (Mohanty et al., 2000). The proposed watermarking algorithm is presented as a flow chart in Fig. 2. For gray scale, the watermark is applied to Y frames. For a color image, the \(C_r\) and \(C_b\) color spaces are watermarking 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 Mohanty et al. (2000) for images:

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

where \(C_W(i, j)\) is a DCT coefficient of watermarked images, \(\alpha_n\) is the scaling factor, and \(\beta_n\) 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 \(\alpha_n\) and \(\beta_n\) 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 \(\alpha_n\) should be close to its maximum value \(\alpha_{\max}\), while the value of \(\beta_n\) should be close to its minimum value \(\beta_{\min}\). Since the watermark DCT 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 \( \sigma_n \), it is desirable to make \( \alpha_n \) proportional to \( \sigma_n \) and \( \beta_n \) inversely proportional to \( \sigma_n \). Therefore, for non-edge blocks the following models are used:

\[
\alpha_n = \sigma_n^2 \times \exp\left(\left(-\frac{\mu_n^2}{\sigma_n^2}\right)\right), \\
\beta_n = \left(\frac{1}{\alpha_n}\right) \times \left(1 - \exp\left(-\frac{\mu_n^2}{\sigma_n^2}\right)\right).
\]

The subscript \( n \) indicates the 8 × 8 block for which the parameters are being calculated. \( \alpha_n \) and \( \beta_n \) in Eq. (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 \( \sigma_n \), given by:

\[
\sigma_n = \frac{\ln(\sigma_n)}{\ln(\sigma_{\text{max}})} \quad \text{with} \quad \sigma_{\text{max}} = 8
\]

In Eq. (6), \( \sigma_{\text{max}} \) is the maximum value of all of the \( \sigma_n \)’s in a frame, \( c_{ij} \) are the DCT coefficients, and \( \mu_n \) is the mean value of the AC DCT coefficients in block \( n \). In Eq. (4), \( \mu_n \) is the normalized mean value of the DC DCT coefficient in block \( n \). \( \mu^* \) is the normalized mean value of all \( c_{00}(n) \) in a frame consisting of \( N 8 \times 8 \) blocks. These are calculated as:

\[
\mu_n = \frac{\sum_{i=0}^{7} \sum_{j=0}^{7} c_{ij} - \mu_{AC}^2}{\sigma_n^2} \quad \text{and} \quad \mu^* = \frac{\sum_{n=1}^{N} c_{00}(n)}{N}
\]

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 \( \alpha_n \) and \( \beta_n \) for each block within each frame being watermarked is very expensive in terms of resource requirements and processing time. Second, pre-determine 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 \( \alpha_n \) and \( \beta_n \) 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 Fig. 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 \( YCbCr \) frames for a given host video
2: Resample \( YCbCr \) 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: \( \text{return} \) to Step 28.
12: else if (P) Frame then
13: \( \text{return} \) to Step 17.
14: else if (B) Frame then
15: \( \text{return} \) 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: \( \text{return} \) 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 8 frames with two P frames or 1 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: \( \text{return} \) to Step 28.
25: else if (I and C) frames then
26: \( \text{return} \) 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.

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 Fig. 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. (1989) is used in the implementation. The 1D row DCT of each 8 \times 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 Fig. 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 capacity is enough for one input group of pictures (GOP), motion vectors, and the output stream. The frame-buffer shown in Fig. 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 Fig. 5(a) performs a search for the best match for each macro block in the current frame based on a 3 \times 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 Fig. 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 Fig. 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 Fig. 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 Fig. 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 Fig. 7. The controller generates address and controls 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.

---

**Fig. 5.** The proposed architectures of various datapath components (cont.).

**Fig. 6.** Datapath for MPEG-4 watermarking (bus width 12 bits).
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 zig-zag 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 Fig. 8, and the details of the watermarking insertion unit are presented in Fig. 9.

As seen from Fig. 9, the video frames are watermarked in the DCT domain before being compressed. For three (Y, C_b, C_r) 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 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 Fig. 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 Fig. 11. In this FSM, several sub-states have been merged for simplicity of design.

Results for the processing of a (Y, C_b, C_r) 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 components (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® DSP Builder blockset 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

---

**Fig. 9.** Simulink® simulation of the encoder.

---

**Fig. 10.** The design flow used for the FPGA-based prototyping of the video watermarking architectures.
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 min for 1 s. of video. The excessive simulation time is due to the constant communication between Simulink® and the FPGA 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-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 Figs. 12–15. A different sequence of AVI (Xvid Codec, 2010) 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, Cb, Cr) 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 Cb and Cr must be

![Diagram of the controller states](image-url)
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/s. If the system is utilized for high-resolution applications, such as the NTSC television video broadcasting system, the peak processing speed is 43 frames/s, which exceeds the required 29.97 frames/s.

Standard video quality metrics such as the Mean Square Error (MSE) and Peak-Signal-to-Noise-Ratio (PSNR) (Mohanty et al., 2000, 2005; Barni et al., 1998) are applied to quantify the system’s performance. The MSE and PSNR (Chen et al., 2002; Loeffler et al., 1989; Barni et al., 1998; Qiao and Nahrstedt, 1998) are expressed by the following equations:

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

(9)
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 (1–3 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 Tables 3 and 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 and 50 dB, the noise will be beyond human perception; however, for PSNR between 10 dB and 20 dB, the noise is detected by the human visual system (Richardson, 2003). 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

Table 3

<table>
<thead>
<tr>
<th>Clips</th>
<th>PSNR (dB)</th>
<th>RMSE</th>
<th>Compression ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bird</td>
<td>21.15</td>
<td>22.34</td>
<td></td>
</tr>
<tr>
<td>Dinner</td>
<td>22.22</td>
<td>19.74</td>
<td></td>
</tr>
<tr>
<td>Barcode</td>
<td>22.77</td>
<td>18.53</td>
<td></td>
</tr>
<tr>
<td>Talent</td>
<td>21.82</td>
<td>23.17</td>
<td>21.46 16–39 16</td>
</tr>
<tr>
<td>Iphone</td>
<td>22.20</td>
<td>20.10</td>
<td></td>
</tr>
<tr>
<td>LGphone</td>
<td>22.82</td>
<td>23.17</td>
<td></td>
</tr>
<tr>
<td>Train</td>
<td>21.75</td>
<td>22.11</td>
<td></td>
</tr>
</tbody>
</table>

Table 4

<table>
<thead>
<tr>
<th>Clips</th>
<th>PSNR (dB)</th>
<th>RMSE</th>
<th>Compression ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bird</td>
<td>21.70</td>
<td>20.97</td>
<td></td>
</tr>
<tr>
<td>Dinner</td>
<td>21.94</td>
<td>20.39</td>
<td></td>
</tr>
<tr>
<td>Barcode</td>
<td>21.40</td>
<td>21.70</td>
<td></td>
</tr>
<tr>
<td>Iphone</td>
<td>21.90</td>
<td>21.00</td>
<td></td>
</tr>
<tr>
<td>LGphone</td>
<td>21.39</td>
<td>22.24</td>
<td></td>
</tr>
<tr>
<td>Train</td>
<td>22.33</td>
<td>23.88</td>
<td></td>
</tr>
</tbody>
</table>
noise for the host video, affecting the PSNR. The results are consistent with other visible watermarking algorithms and architectures available in the current literature (Mohanty et al., 2000, 2005).

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 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 com-

Fig. 14. Sample watermarked video-1 using watermark-2.
Table 5
Video watermarking hardware proposed in the existing literature.

<table>
<thead>
<tr>
<th>Research works</th>
<th>Design type</th>
<th>Different types</th>
<th>Working domain</th>
<th>Chip statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strycker et al. (1999, 2000)</td>
<td>DSP board</td>
<td>Invisible</td>
<td>Spatial</td>
<td>100 MHz</td>
</tr>
<tr>
<td>Maes et al. (2000)</td>
<td>FPGA board</td>
<td>Invisible</td>
<td>Robust</td>
<td>17 k Logic</td>
</tr>
<tr>
<td>Tsai and Wu (2003)</td>
<td>Custom IC</td>
<td>Robust</td>
<td>Spatial</td>
<td>14 k Logic</td>
</tr>
<tr>
<td>Brunton and Zhao (2005)</td>
<td>GPU</td>
<td>Invisible</td>
<td>Spatial</td>
<td>NA</td>
</tr>
<tr>
<td>Jeong et al. (2008a)</td>
<td>FPGA</td>
<td>Invisible</td>
<td>Robust</td>
<td>ALTERA STRATIX</td>
</tr>
<tr>
<td>Mathai et al. (2003b, 2003c)</td>
<td>Custom IC</td>
<td>Invisible</td>
<td>Wavelet</td>
<td>0.18 μm, 3.53 mm², 75 MHz, 160 mW</td>
</tr>
<tr>
<td>Vural et al. (2005)</td>
<td>Architecture</td>
<td>Invisible</td>
<td>Robust</td>
<td>Wavelet</td>
</tr>
<tr>
<td>Jeong et al. (2008b)</td>
<td>FPGA</td>
<td>Invisible</td>
<td>Wavelet</td>
<td>XILINX VERTEX2</td>
</tr>
<tr>
<td>Petitjean et al. (2002)</td>
<td>FPGA board</td>
<td>Invisible</td>
<td>Fractal</td>
<td>50 MHz takes 6 μs</td>
</tr>
<tr>
<td>This paper</td>
<td>DSP board</td>
<td>Robust</td>
<td>Fractal</td>
<td>250 MHz takes 118 μs</td>
</tr>
<tr>
<td></td>
<td>FPGA</td>
<td>Visible</td>
<td>DCT</td>
<td>100 MHz, 43 fps</td>
</tr>
</tbody>
</table>

Fig. 15. Sample watermarked video-2 using watermark-2.
pression is estimated as (1 + 1 + 1)/(1 + 1/4 + 1/16) ≈ 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 inter- 
 
late 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.

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/s and a PSNR of 30 dB. Further development is underway 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.

Acknowledgments

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 (Mohanty et al., 2009).

References

Saraju P. Mohanty is currently an Associate Professor at the Department of Computer Science and Engineering, University of North Texas and the Director of the NanoSystem Design Laboratory (NSDL, http://nsdl.cse.unt.edu). He obtained his Ph.D. from the University of South Florida in 2003. He obtained his masters in System Science and Automation from the Indian Institute of Science, Bangalore, in 1999. He obtained B. Tech. (Honors) degree in Electrical Engineering from Orissa University of Agriculture and Engineering in 1995. Dr. Mohanty’s research is in Design and CAD for low-power, high-performance nanoscale VLSI. His research has resulted in 120+ publications in peer-reviewed high-quality journals or conferences, 2 patent disclosures, and 1 book. The 120+ publications are well-received by the world-wide peers and have got a total of 720+ citations with H-index of 15 (as provided by Harzing’s Publish and Perish software). His research is funded by the National Science Foundation and Semiconductor Research Corporation. Dr. Mohanty serves on the program/organizing committee of several international conferences. He steering committee chair a new conference called International Symposium on Electronic System Design (ISED). He is a senior member of IEEE.

Elias Kougianos received a B.S. degree in Electrical Engineering in 1985 from the University of Patras, Greece, an M.S. in Physics, an M.S. and a Ph.D. in Electrical Engineering in 1987, 1988 and 1997, respectively, all from Louisiana State University in Baton Rouge, LA. He is currently an Associate Professor in the Department of Electrical Engineering and Electrical Engineering Technology, at the University of North Texas (UNT), Denton, TX. From 1988 through 1997, he was with Texas Instruments, Inc., in Houston and Dallas, TX. In 1997, he joined Avant! Corp. (now Synopsys) in Phoenix, AZ as a Senior Applications engineer and in 2001, he joined Cadence Design Systems, Inc., in Dallas, TX as a Senior Architect in Analog/Mixed-Signal Custom IC design. He has been at UNT since 2004. His research interests are in the area of Analog/Mixed-Signal/RF IC design and simulation and in the development of VLSI architectures for multimedia applications. He has published 50+ refereed journal and conference papers and a book. He is a member of the steering committee of the International Symposium on Electronic System Design. He is a senior member of IEEE.