VLSI Architectures of Perceptual Based Video Watermarking for Real-Time Copyright Protection

Saraju P. Mohanty[1], Elias Kougianos, Wei Cai, Manish Ratnani
VLSI Design and CAD Laboratory (http://vdcl.cse.unt.edu)
University of North Texas, Denton, TX 76203, USA.
[1]Email-ID: saraju.mohanty@unt.edu

Abstract

For effective digital rights management (DRM) of multimedia in the framework of embedded systems, both watermarking and cryptography are necessary. In this paper, we present a watermarking algorithm and VLSI architecture that can insert a broadcaster's logo in video streams in real-time to facilitate copyrighted video broadcasting and internet protocol television (IP-TV). The VLSI architecture, when realized in silicon can be deployed in any multimedia producing appliance to enable DRM. The watermark is inserted in the video stream before MPEG-4 compression, resulting in simplified hardware requirements and superior video quality. The watermarking process is performed in the frequency domain. The system is initially prototyped and validated in MATLAB/Simulink and subsequently implemented on an Altera Cyclone-II FPGA. Its maximum throughput is $43 frames/sec$ at a clock speed of $100MHz$ which makes it suitable for real-time digital video broadcasting emerging applications such as IP-TV.

Keywords

VLSI Architecture, Watermarking, Copyright Protection, Video, Multimedia Content Protection, IP-TV

1 Introduction, Motivation, and Contributions

There is a 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 uncompressed video and the broadcaster's logo. The output is real-time compressed video with the logo embedded. This situation arises in IP-TV and digital TV broadcasting when video residing in a server has to be broadcast by different stations and under different broadcasting rights. Embedded systems that are involved in broadcasting need to have embedded copyright protection. Existing works [15, 10] are targeted towards invisible watermarking, not useful for logo insertion. Other existing works [7, 12] are for images not for video.

MPEG-4 is the mainstream exchangeable video format in the Internet today because it has higher and flexible compression rate, lower bit rate, and higher efficiency while providing superior visual quality [2, 4, 14]. Microsoft, Real and Apple support the MPEG4 standard and already have embedded MPEG-4 decoders into some of their products, and there are even free software implementations available, such as the Xvid codec [1]. This motivated us to consider MPEG-4 as the target video compression framework in our research.

Novel contributions of this paper are as follows:

1. A perceptual-based adaptive visible watermarking algorithm suitable for video broadcasting.
2. VLSI architectures for real-time watermarking in the context of compressed video (MPEG-4).
3. Simulink and FPGA prototyping of the VLSI architectures which can be integrated in multimedia producing appliances (e.g. digital camera, network processor).

The remainder of this paper is organized as follows: In Section 2, we review the current hardware based watermarking literature. In Section 3, we present an overview of the MPEG-4 and hardware cost perspective. We introduce the watermarking algorithm in Section 4. In Section 5, we introduce proposed hardware architectures. High-level architectural Simulink simulations and FPGA based hardware prototyping are presented in Section 6. Experimental results are presented in Section 7. The paper concludes in Section 8.

2 Significance of the Proposed Research with Respect to Related Prior Research

The existing literature is rich in watermarking algorithms developed for various types of media. These algorithms, which are realized primarily as software, work off-line: the data are first acquired and then the watermarks are inserted before the watermarked data 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 develop hardware based watermarking systems that bridge this gap*. The watermarking chip will be fitted in electronic appliances which are embedded systems designed using SoC technology.

Watermarking hardware systems are designed and implemented on an FPGA board, DSP board, or custom ICs. The

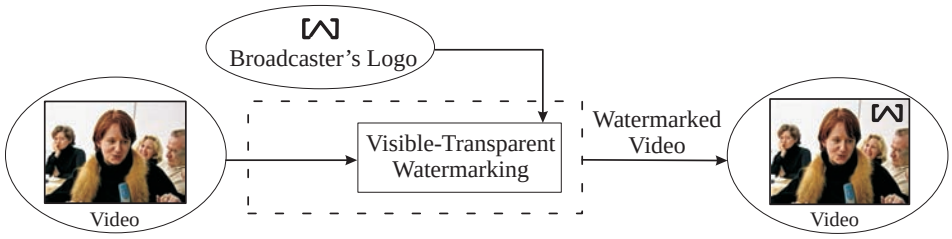


Figure 1. Real-time logo insertion through watermarking.

authors in [15, 10] have proposed a real-time watermarking scheme in the spatial domain for television broadcast monitoring. The authors in [10] describe a VLSI chip for the same video watermarking algorithm. A DCT (discrete cosine transform) domain invisible watermarking chip is presented in [16]. In [7], a VLSI architecture is proposed for invisible-fragile watermarking in the spatial domain. In [12], the authors propose another watermarking hardware architecture that can insert two visible watermarks in images in the spatial domain. In [6] the authors propose an adaptive DWT (discrete wavelet transform) based visible watermarking design. To advance the state-of-the art we propose a DCT domain watermarking algorithm and architecture in the context of MPEG-4 for real-time applications.

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

The main steps for MPEG-4 are color space conversion and sampling, DCT and its inverse (IDCT), quantization, zigzag scanning, motion estimation, and entropy coding. A simpler approach is a single watermarking step in the compression framework because the computation requirements of watermarking are comparable to these steps. In this section we discuss the watermarking algorithm in the framework of MPEG-4 and highlight the VLSI architecture and design decisions that were taken to achieve real-time performance.

3.1 Color Space Conversion

The color space conversion equations are as follows [5] :

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

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.

3.2 DCT or IDCT

For ease of hardware implementation, we selected the fast DCT algorithm and its inverse [8]. The fast DCT algorithm reduces the number of adders and multipliers so that the evaluation of the DCT or IDCT coefficients is accelerated.

3.3 Quantization

In the MPEG-4, a uniform scalar quantization is adopted. The feature of the scalar quantization scheme is an adaptive quantized step size according to the DCT coefficients. For computational efficiency and hardware simplification, scalar quantization step size is chosen from pre-defined tables [3].

3.4 Zigzag Scanning

Zigzag scanning sorts the matrix of DCT coefficients in ascending order. For progressive frames and interlaced fields, zigzag scanning routes are provided by predefined Tables [3].

3.5 Motion Estimation

The criterion of match for two macroblocks is the minimized difference between them. For computational simplification, the sum of absolute difference (SAD) criterion is [3]:

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

Where, $c(i,j)$ and $p(m,n)$ are the pixels of blocks from the current and previous frames, respectively. The hardware implementation of the motion estimation block is sequential and contributes the largest delay in the critical path.

3.6 Entropy Coding

The entropy coding efficiency depends on the precision of calculating the probability of occurrence of each coefficient. The approach we followed is to utilize pre-calculated Huffman code tables for generic images [3].

4 The Proposed Watermarking Algorithm

We now present a watermarking algorithm that performs the broadcaster's logo insertion as 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 [3]. The other important issue of visible watermarking, transparency, comes from making the watermark adaptive to the host frame. 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 watermark image, the C_b and C_r color space are watermarked using the same techniques for Y frames. To protect against frame interpolating attacks on watermarking, all I, B, P frames must embed the watermark.

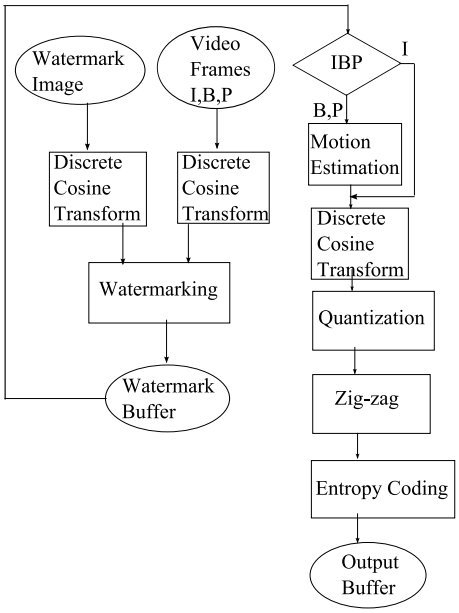


Figure 2. The flow of the proposed video watermarking algorithm.

The watermark embedding approach used in this paper was originally proposed by us for images [11]:

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

where $C_W(i,j)$ is a DCT coefficient after watermark embedding, 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. Their values are computed based on 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}. The user inputs that serve as quality control parameters are α_{max}, α_{min}, β_{min}, and β_{max}. Since the watermark DCT coefficients will be added to the video frame 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 of a video frame:

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

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

$${\sigma^*}_n = \tfrac{ln(\sigma_n)}{ln(\sigma_{max})} \text{ with } \sigma_n = \tfrac{1}{64}\sum_{i=0}^{7}\sum_{j=0}^{7}\left(c_{ij} - {\mu^{AC}}_n\right) \tag{5}$$

In Eqn. 5, σ_{max} is the maximum value of all the σ_n in a frame, c_{ij} are the DCT coefficients, and ${\mu^{AC}}_n$ is the mean value of the AC DCT coefficients in block n. In Eqn. 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:

$${\mu^*}_n = \tfrac{c_{00}(n)}{c_{max}} \text{ and } \mu^* = \tfrac{1}{N}\sum_{n=1}^{N} c_{00}(n). \tag{6}$$

Algorithm 1 The proposed MPEG-4 watermarking algorithm

1: Convert RGB color frames to YC_bC_r frames for the input 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: Perform DCT for each 8×8 block to generate DCT coefficients.
5: Perform perceptual analysis of the host video frame.
6: Compute scaling and embedding factor for different blocks.
7: Each block of Y DCT matrix is watermarked with an 8×8 watermark DCT matrix at same location as at DCT domain.
8: Perform 2-D IDCT for each 8×8 watermarked matrix to transform it back to Y color pixels.
9: Buffer watermarked Y component, non watermark C_b and C_r frames which holds a GOP.
10: Split Y component into 16×16 blocks and C_b and C_r into 8×8.
11: Perform motion estimation for Y component.
12: Obtain the motion vectors (MV) and prediction errors of residual frame for motion compensation (MC) for Y component.
13: Obtain motion vector and prediction error for C_b, C_r components and residual frame for motion compensation.
14: Perform 2-D DCT on blocks of different frames.
15: Quantize 2-D DCT coefficient matrix.
16: Zigzag scan quantized 2-D DCT coefficient matrix.
17: Entropy coding re-ordered 2-D DCT coefficient matrix and motion vector.
18: Build structured compreseed stream from the buffer.

Once the intra-frame parameter issue is solved as above, the next challenge is their determination for inter-frames. There are several ways, 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. We followed the second alternative in this paper.

5 The Proposed VLSI Architectures

The proposed datapath architecture that can perform watermarking in the MPEG-4 video compression framework is

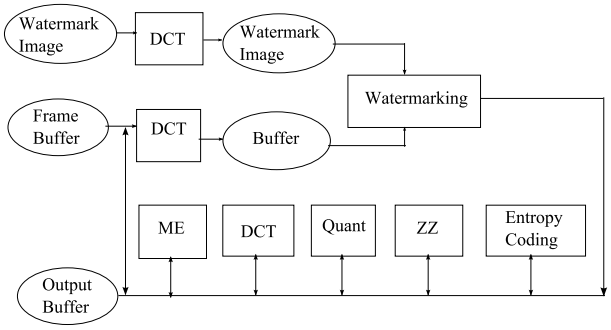
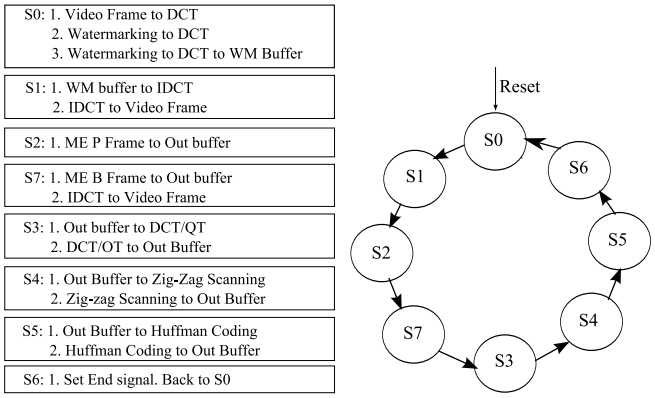


Figure 3. Datapath for MPEG-4 watermarking architecture (bus width 12 bits).



(a) Controller states (b) State diagram

Figure 4. State diagram of the controller for MPEG-4 watermarking architecture.

shown in Fig. 3. This is constructed by stitching the individual modules discussed in the rest of the section using multiplexers. The controller generates address and control signals to synchronize other components. It is realized as a finite state machine (FSM) as shown in Fig. 4. In this FSM several substates have been merged to 8 states for simplicity of design.

In the rest of the section we discuss individual components of the system that can perform MPEG-4 compression and watermarking.

5.1 Watermark Embedding

The watermarking unit is composed of several modules, such as DCT, perceptual analyzer, edge detection, scaling factor, insertion, row and column address decoder, registers and controller. The DCT module calculates the DCT coefficients of host and watermark video frames before they are stored in the buffer memory. The controller governs the operations of all the 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. The input and output video frames are buffered to the frame buffer as shown in Fig. 5(a) and Fig. 5(c)

5.2 Discrete Cosine Transformation (DCT)

The DCT module calculates the DCT coefficients of the video frames and it consists of 2-1D DCT sub-module. The algorithm from [8] is used for our implementation. The 1D row DCT of each 8×8 block was first computed. The column DCT of each block is then carried out. 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. 5(b) for a 2-D DCT with 12-bit data bus and 6-bits address bus for 64 bytes internal buffer. The input data is an 8-bit unsigned integer and the output is of 12 bits.

5.3 Frame Buffer

The frame buffer buffers the frames during the intermediate computation by other modules as shown in Fig. 5(d). Its size capacity is sufficient for one input group of pictures, motion vectors, and the output stream. This is an external buffer which is different than the block memory used by 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, Fig. 6(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. Pixel data are 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 motion detection core uses 16 block RAM modules and half pel uses 9 block RAM modules to do the exhaustive search.

5.5 Entropy Block

It is implemented as Huffman coding look up as shown in Fig. 6(b). It has different submodules for variable length coding and pattern matching, etc.

5.6 Quantization Block

This module shown in Fig. 6(c) quantizes the DCT coefficients according to predefined quantization tables. The input and output are buffered to the frame buffer.

5.7 Zig-Zag Block

The architecture of the zigzag scanning unit that performs re-ordering of the DCT is shown in Fig. 6(d).

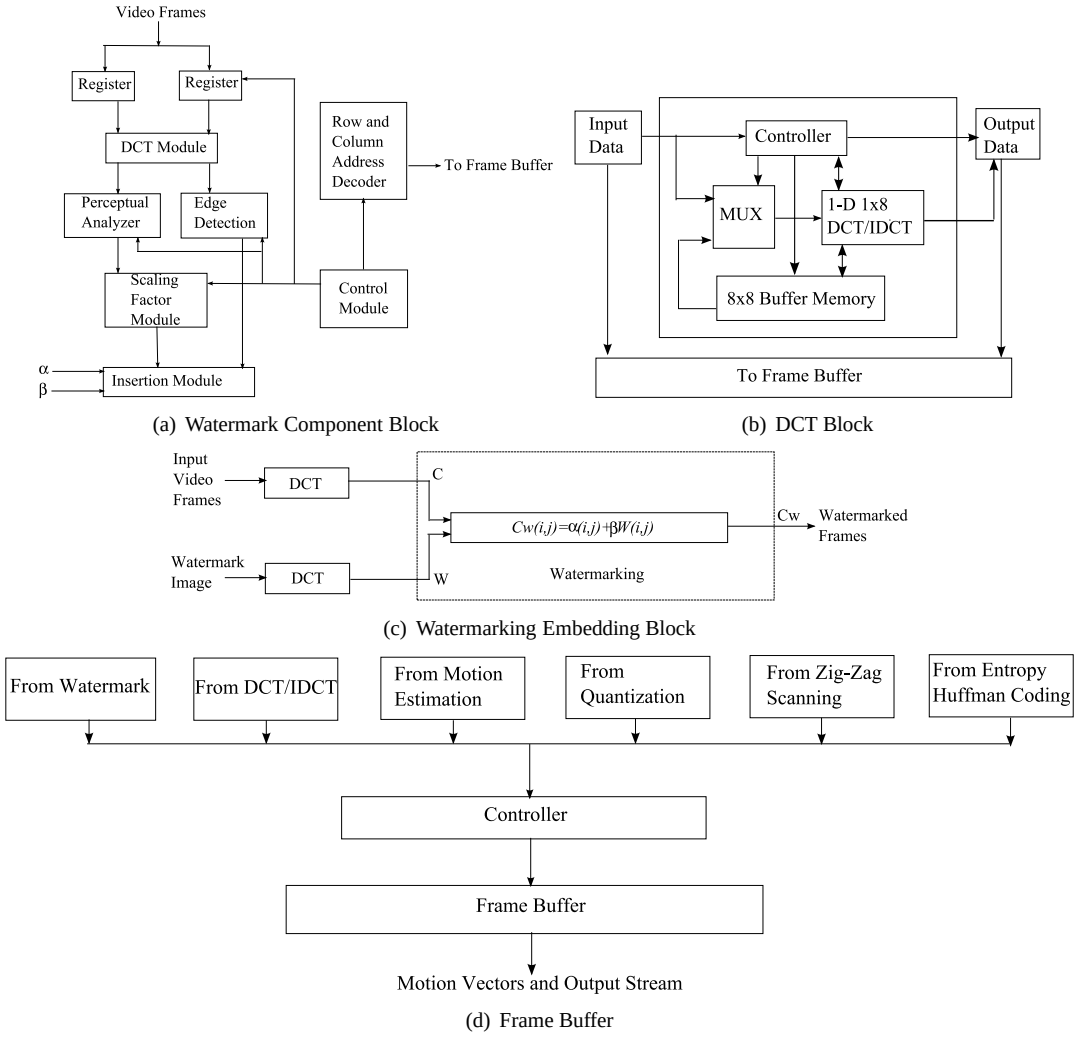


Figure 5. Various datapath components of the proposed architecture.

6 System Level Modeling and Prototyping

6.1 Simulink Based Modeling

To verify the functionality of the algorithms and architecture, a fast prototyping module was built in MATLAB/Simulink. The methodology at 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. We performed exhaustive simulations 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 as a proof of testing of the functionality of the watermarking system.

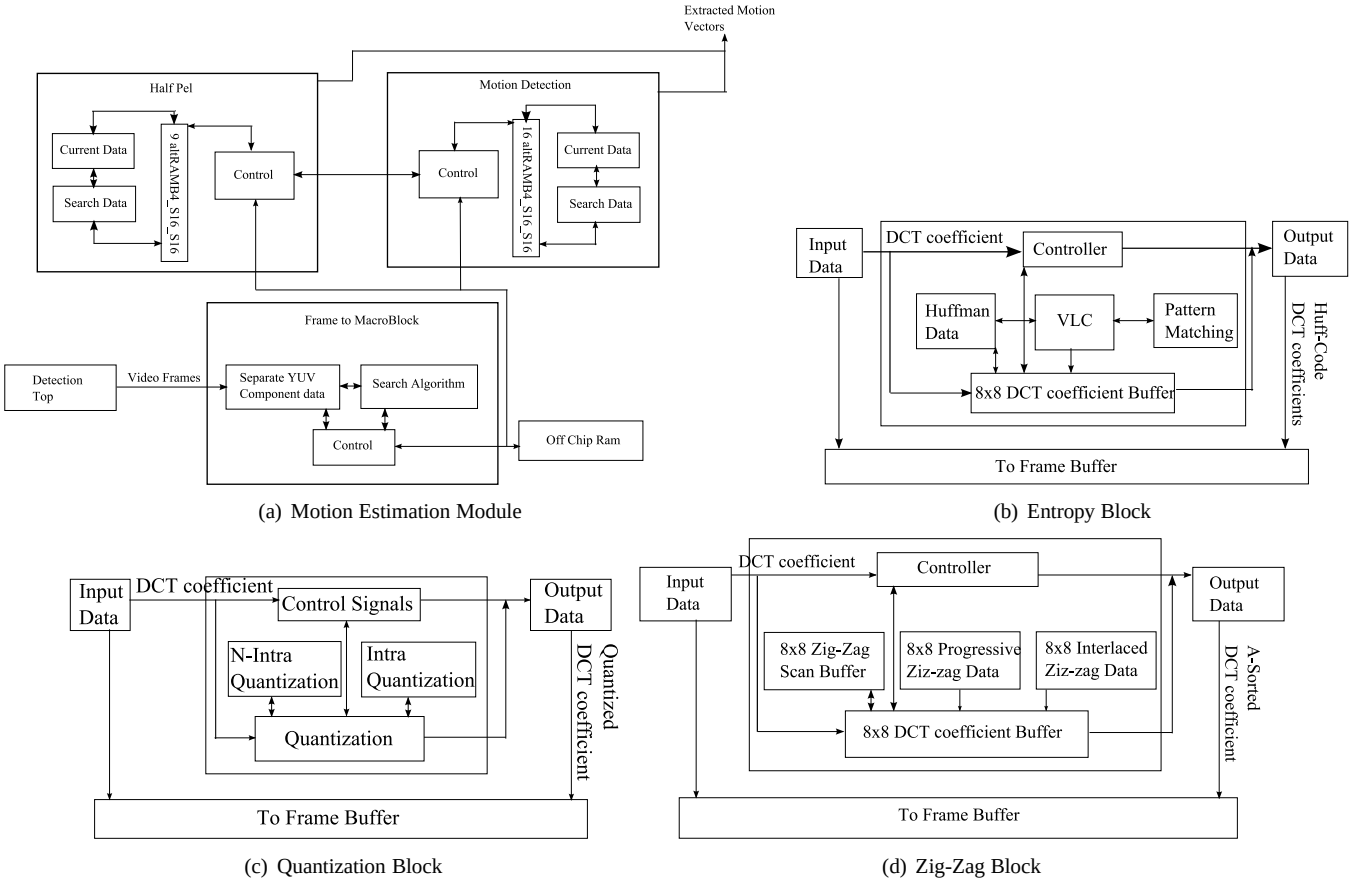
6.2 FPGA Based Prototyping

During the FPGA-based prototyping development the system is modeled in VHDL. The VHDL model was compiled using Altera Quartus 7 for a Cyclone II target. Results for the processing of a (Y, C_r, C_b) frame using Quartus synthesis tools are shown in Table 1. From this table, it is seen that the total processing time for 3 frames (Y, C_b and C_r) of size 320×240 is $1.07 msec$ or $932 frames/sec$. If the system is utilized in applications such as the NTSC television video broadcasting system, the peak processing speed is $43 frames/sec$, which exceeds the required $29.97 frames/sec$. The entire system fits within the logic capacity of this FPGA family.

7 Experimental Results

7.1 Experimental Set up

The high level system modeling was done with MATLAB/Simulink. We introduced a different random sequence of AVI [1] video clips and similarly different watermark images to be embedded with the video clips, all having dimensions of 320×240.



(a) Motion Estimation Module

(b) Entropy Block

(c) Quantization Block

(d) Zig-Zag Block

Figure 6. The proposed architectures of various datapath components.

Table 1. FPGA synthesis report of the VLSI architectures

Components	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 YCbCr	1416	0	8
Motion Estimation	8987	4900	0
Controller	575	157	0
Overall Architecture	28322	16532	9

7.2 Testing of Watermarking Quality

We performed exhaustive simulations to make assessment of watermarking quality with a large variety of watermark images and video clips. For brevity we present selected examples of watermarked video in Figs. 7 and 8.

Standard video quality metrics mean square error (MSE) and peak-signal-to-noise-ratio ($PSNR$) are applied to quantify the system's performance as follows [13, 5, 3, 8]:

$$MSE = \frac{\sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{k=1}^{3}|p(m,n,k)-q(m,n,k)|^2}{3\times M\times N}, \quad (7)$$

$$PSNR = 10\times\log_{10}\left(\frac{(2^i-1)^2}{\text{MSE}}\right). \quad (8)$$

Where $p(m,n,k)$ and $q(m,n,k)$ are the pixels after and before processing, respectively. From the above MSE and $PSNR$ equations, the quality metrics of video compression and watermarking in the working model are displayed in Table 2 and Table 3 for two different watermarks added to the video frames respectively. The average $PSNR$ of watermarked compressed video was $21dB$. It may be noted that the low $PSNR$ did not degrade the perceptual quality of the video, as the low value is due to the fact that the watermark logo inserted is visible and consequently becomes noise for the host video and affects the $PSNR$. The results are at par with visible watermarking schemes available in the current literature.

To estimate the variable compression rate we assume that half of the DCT coefficients are truncated so the compression rate is 2:1. The redundancy of two frames is removed in 75% by the motion estimate for a compression rate of 4:1, and in the working module, one GOP is constituted with one I frame, one B frame and 1 P frame or IBP structure. The motion estimation compression is estimated as $(1+1+1)/(1+1/4+1/16)\approx 2:1$. Huffman coding has compression rates approximately

Table 2. Video quality metrics for watermark - 1

Clips	$PSNR(dB)$	$RMSE$	Compression Ratio		
			Average	Range	Estimate
Bird	21.1	22.3	21.4	16 to 39	16
Dinner	22.2	19.7			
Iphone	22.2	20.1			
LGphone	22.8	23.1			

Table 3. Video quality metrics for watermark - 2

Clips	$PSNR(dB)$	$RMSE$	Compression Ratio		
			Average	Range	Estimate
Bird	21.7	20.9	21.2	16 to 39	16
Dinner	21.9	20.4			
Iphone	21.9	21.0			
LGphone	21.4	22.2			

(a) Original bird video

(b) Watermarked bird video

(c) Original dinner video

(d) Watermarked dinner video

(e) Original Iphone Review video

(f) Watermarked Iphone Review video

(g) Original LGphone video

(h) Watermarked LGphone video

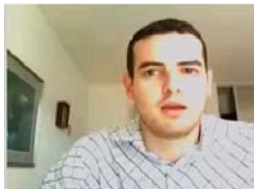
Figure 7. Sample watermarked videos using watermark - 1.

(a) Original bird video

(b) Watermarked bird video

(c) Original dinner video

(d) Watermarked dinner video



(e) Original Iphone video

(f) Watermarked Iphone video

(g) Original LGphone video

(h) Watermarked LGphone video

Figure 8. Sample watermarked videos using watermark - 2.

Table 4. Comparative perspective with other video watermarking hardware

Works	Design Type	Watermarking	Domain	Features
Maes, [9]	FPGA/IC	Invisible-Robust	Spatial	$17k$ Gates
Mathai [10]	Custom IC	Invisible-Robust	Wavelet	$1.8V$
Tsai [16]	Custom IC	Invisible-Robust	Spatial	NA
This Work	FPGA	Visible	DCT	$43 frames/sec$

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.

7.3 Comparison With Existing Research

We present performance statistics with reference to existing hardware based watermarking for video in Table 4. We note that our implementation is the only one capable of achieving real-time video watermarking and compression at rates exceeding existing broadcast standards.

8 Summary, Conclusions, and Future Works

We presented a visible watermarking algorithm using FPGA technology for MPEG4 video. 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. Further research is under way to extend the real-time performance of the system to HDTV and higher resolutions and to improve the $PSNR$. Towards this end we are investigating watermark embedding in the compressed domain. In future, advanced MPEG-4 features, such as N-bit resolution, advanced scalable textures, and video objects will be utilized. It is anticipated that, with modest hardware complexity increase, performance will be significantly improved with the inclusion of these additional features. RTL level subsystem optimization will also be used to improve resource utilization and minimize execution time.

References

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

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

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

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

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

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

[7] A. Garimella, M. V. V. Satyanarayan, R. S. Kumar, P. S. Murugesh, and 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*, pages 283–288, 2003.

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

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

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

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

[12] S. P. Mohanty, N. Ranganathan, and 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):1002–1012, August 2005.

[13] L. Qiao and K. Nahrstedt. Watermarking Methods for MPEG Encoded Video: Towards Resolving Rightful Ownership. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, pages 276–285, 1998.

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

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

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