VLSI IMPLEMENTATION OF INVISIBLE DIGITAL WATERMARKING ALGORITHMS TOWARDS THE DEVELOPMENT OF A SECURE JPEG ENCODER

Saraju P. Mohanty, N. Ranganathan and Ravi K. Namballa

Department of Computer Science and Engineering
Nanomaterial and Nanomanufacturing Research Center
University of South Florida, Tampa, FL 33620
{smohanty,ranganat,rnamball}@csee.usf.edu

ABSTRACT

Digital watermarking is the process that embeds data called a watermark into a multimedia object such that the watermark can be detected or extracted later to make an assertion about the object. Several software implementations of the proposed algorithms are available, but very few attempts have been made for hardware implementation. The goal of hardware implementation is to achieve low power, high performance, and reliability. In this paper, we develop hardware system that can insert both robust and fragile invisible watermarks in images. The hardware module can be easily incorporated into a JPEG encoder to develop a secure JPEG encoder. A prototype chip is implemented using 0.35μ CMOS technology. To our knowledge, this is the first watermarking chip implementing both invisible-robust and invisible-fragile watermarking capabilities.

1. INTRODUCTION

Watermarking is the process that embeds data called a watermark, tag or label into a multimedia object such that watermark can be detected or extracted later to make an assertion about the object [1]. Watermarks of varying degree of visibility are added to presentation media as a guarantee of authenticity, quality, ownership and source. In general, any watermarking scheme consists of three parts, such as the watermark, the encoder (insertion algorithm) and the decoder and comparator (verification or extraction or detection algorithm). The insertion algorithm incorporates the watermark into the object, whereas the verification algorithm authenticates the object, determining both the owner and the integrity of the object. The watermarks can be applied either in spatial or in frequency domain (FFT, DCT or wavelet). Even though spatial domain watermarking is less robust, the spatial domain schemes have less computational overhead compared to frequency domain schemes. According to the human perception, the digital watermarks can be divide into four different types, such as visible, invisible-robust, invisible-fragile and dual [2].

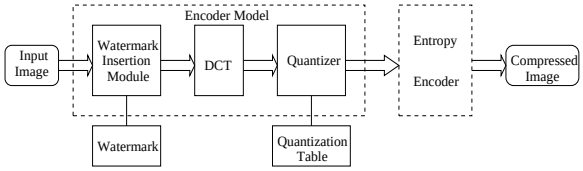


Fig. 1. Secure JPEG encoder : block level view

Each of the above watermarking schemes is equally important due to its unique applications. In this work, we focus on VLSI implementation of an invisible-robust and an invisible-fragile spatial domain watermarking algorithm. The VLSI chip can insert any one or both the watermarks depending on the requirements of the user. The proposed watermarking chip can be easily incorporated as a module in any existing JPEG encoder and a secured JPEG encoder can be developed. We provide an outline of such a secure JPEG encoder in Fig. 1 [3]. It may be noted that the corresponding watermark extraction module has to be inbuilt in a secure JPEG decoder. The secure JPEG codec can be a part of a scanner or a digital camera so that the digitized images are watermarked right at the origin.

2. RELATED WORK

Strycker and et. al. [4] propose a real-time watermarking scheme for television broadcast monitoring. They address the implementation of spatial domain watermark embedder and detector on a Trimedia TM-1000 VLIW processor. In the insertion procedure, pseudo-random numbers are added to the incoming video stream. The watermark detection is based on the calculation of correlation values. Mathai and et. al. [5] present hardware implementation of the same video watermarking algorithm using 0.18μ technology.

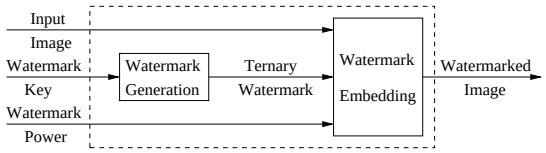


Fig. 2. Invisible robust watermark insertion

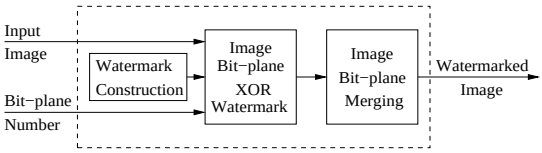


Fig. 3. Invisible fragile watermark insertion

A DCT domain invisible watermarking chip is presented by Tsai and Lu [6]. The watermark system embeds a pseudo-random sequence of real numbers in a selected set of DCT coefficients. They also proposed a JPEG architecture incorporating the watermarking module. The watermark is extracted without resorting to the original image. The watermark chip is implemented using TSMC $0.35\mu m$ technology and occupies a die size of $3.064 \times 3.064mm^2$.

Garimella and et. al. [7] propose a VLSI architecture for invisible-fragile spatial domain watermarking. In this scheme, the differential error is encrypted and interleaved along the first sample. The watermark can be extracted by accumulating the consecutive LSBs of pixels and then decrypting. The extracted watermark is compared with the original watermark for image authentication. The ASIC is implemented using 0.13μ technology.

In this paper, we propose a VLSI architecture that can insert both invisible-robust and invisible-fragile watermarks in spatial domain. The watermarking insertion algorithms implemented are the following : (i) the invisible-robust algorithm proposed by Tefas and Pitas [8, 9] and (ii) the invisible-fragile algorithm proposed by Mohanty, Ramakrishnan and Kanakanhalli [2].

3. WATERMARKING ALGORITHMS

In this section, we discuss the algorithms (invisible-robust and invisible-fragile) implemented as VLSI architectures in this work. We outline the insertion and detection methods in brief with the modifications necessary to make the hardware implementation easy. The following notations are used.

I	: Original image (gray image)
W	: Watermark image (binary or ternary image)
(i, j)	: A pixel location
I_W	: Watermarked image
$N_I \times N_I$	: Image dimension
$N_W \times N_W$	: Watermark dimension
E, E_1, E_2	: Watermark embedding functions
D	: Watermark detection function
r	: Neighborhood radius
I_N	: Neighborhood image (gray image)
K	: Digital (watermark) key
α_1, α_2	: Scaling constants (watermark strength)

3.1. Invisible Robust Watermarking

The block diagram for watermark insertion is provided in Fig. 2 [8, 9]. The watermark W is a ternary image having pixel values $\{0,1 \text{ or } 2\}$. These values are generated using the digital key K. The watermark insertion is performed by altering the pixels of original image as follows.

$$I_W(i,j) = \begin{cases} I(i,j) & \text{if } W(i,j) = 0 \\ E_1\big(I(i,j), I_N(i,j)\big) & \text{if } W(i,j) = 1 \\ E_2\big(I(i,j), I_N(i,j)\big) & \text{if } W(i,j) = 2 \end{cases} \tag{1}$$

The encoding functions E_1 and E_2 are defined as follows, where $\alpha_1 > 0$ and $\alpha_2 > 0$.

$$\begin{aligned} E_1(I, I_N) &= (1-\alpha_1)I_N(i,j) + \alpha_1 I(i,j) \\ E_2(I, I_N) &= (1-\alpha_1)I_N(i,j) - \alpha_2 I(i,j) \end{aligned} \tag{2}$$

It may be noted that the above functions are bit different from the original algorithm, where α_2 was negative and the second encoding function involved addition, instead of subtraction. However, these changes do not affect the overall encoding-decoding scheme, since we make changes in decoding functions accordingly.

The neighborhood image pixel gray value is calculated as the average gray value of neighboring pixels of the original image for a particular neighborhood radius r. For example, for neighborhood radius $r = 1$, it is calculated as :

$$I_N(i,j) = \frac{\frac{I(i+1,j)+I(i+1,j+1)}{2} + I(i,j+1)}{2} \tag{3}$$

The scaling $(1-\alpha_1)$ is used to scale I_N to ensure that watermarked image gray value I_W never exceeds the maximum gray value for 8-bit image representation corresponding to pure white pixel. It may be noted that a simple average could have been $\frac{I(i+1,j)+I(i+1,j+1)+I(i,j+1)}{3}$, but we used the above method of averaging to simplify our hardware implementations, since the division by two can be implemented using a right shift by 1-bit operation.

The first step detection process is generation of watermark W using the watermark key K. Next, the watermark is extracted from the test (watermarked) image using the detection function. By comparing the original ternary watermark image W and extracted binary watermark image W^*, the ownership can be established when the detection ratio is larger than a predefined threshold as explained in [8].

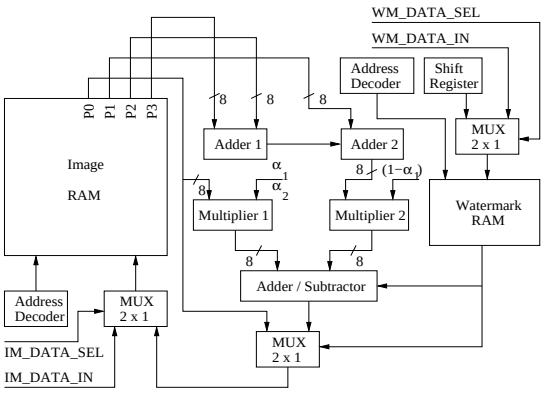


Fig. 4. Datapath for robust watermarking

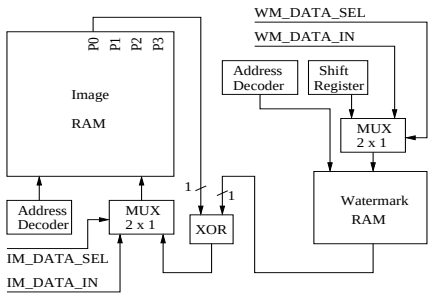


Fig. 5. Datapath for fragile watermarking

3.2. Invisible Fragile Watermarking

The invisible fragile watermark insertion is carried out as follows (Fig. 3 [2]). Pseudo-random binary-sequence {0,1} of period N is generated using linear shift register. The period N is equal to the number of pixels ($N_W \times N_W$) of the image. The watermark is generated by arranging the binary sequence into blocks of size 4×4 or 8×8. The size of the watermark is same as the size of the image. The bit planes of the input image are found out and watermark is inserted in the appropriate bit plane such that the SNR $>$ threshold. Assuming the watermark insertion is to be performed in k^{th} bit plane, the watermark insertion process is given by the following expression.

$$\begin{aligned} I_W[0 \to k-1](i,j) &= I[0 \to k-1](i,j) \\ I_W[k](i,j) &= I[k](i,j) \text{XOR } W(i,j) \\ I_W[k+1 \to 7](i,j) &= I[k+1 \to 7](i,j) \end{aligned} \quad (4)$$

Finding of the candidate bit plane for watermark insertion is an iterative process. We have chosen the $2^{nd}(k = 2)$ bit plane as the candidate for watermark insertion (for LSB $k = 0$). After merging all the bit planes, the watermarked image I_W is obtained.

For image authentication purpose, the testing paradigm provided in [2, 10] is used. To construct the testing paradigm, the cross-correlation of original image and watermark image, and the cross-correlation of watermarked image and possibly forged test image are calculated. Then, based on the cross-correlations, the test statistics is determined. The test statistics is the basis of the test paradigm.

4. VLSI ARCHITECTURE

The datapath for invisible robust watermarking in shown in Fig. 4. The image RAM is used to store the original image, which is to be watermarked. The image data can be written to the image RAM by activating proper control signals. The watermark RAM serves as a storage space for watermark data. The watermark data can either be generated using the shift register or given as external input by the user. In this hardware design, it is assumed that at any point of time, a 256×256 image can be stored in the image RAM and a 128×128 watermark can be stored in the watermark RAM. It is possible to watermark only a 128×128 region of the original image at a time, whereas the full image can be watermarked if the process is repeated for other regions (total in four times for the assumed size).

The invisible robust watermark insertion scheme involves adding (or subtracting) a constant times the pixel (gray value) to be watermarked to (from) a constant times the neighborhood function. The constants are α_1 and α_2, the values of which determine the strength of the watermark. The four output lines from the image RAM provide the pixels $I(i,j)$, $I(i,j+1)$, $I(i+1,j)$ and $I(i+1,j+1)$ for the row-column address pair (i,j). The neighborhood function specified by Eqn. 3 is computed as follows. First, the $I(i,j+1)$ and $I(i+1,j+1)$ are given to the adder1 as input. Then, the resulting sum and carry out from adder 1 are fed to the adder 2 alongwith $I(i+1,j)$. The resulting sum of the adder 2 is the neighborhood function value. The division by two is performed by shifting the results bit right by one bit, consequently discarding the rightmost bit (LSB). The scaling of the neighborhood function is achieved by multiplying it with $(1-\alpha_1)$ using the multiplier 2. At the same time, the scaling of the image pixel gray values is performed in multiplier 1 by multiplying $I(i,j)$ with α_2 or α_1. The eight higher order bits of the the multipliers are fed to the adder/subtractor unit to perform watermark insertion as per the Eqn. 2. Since, we are concerned only with the integer values of the pixels, the lower eight bits of the multiplier results are discarded, which represent the values after the decimal point. The output of the adder / subtractor unit (watermarked image pixels) and the original image pixel values are multiplexed based on the watermark values and are written into the image RAM if the watermark value is "1" or "2", as per Eqn. 1.

Fig. 5 shows the datapath for fragile watermark insertion. The original image is stored in the image RAM and the watermark is created in the same way as in the case of

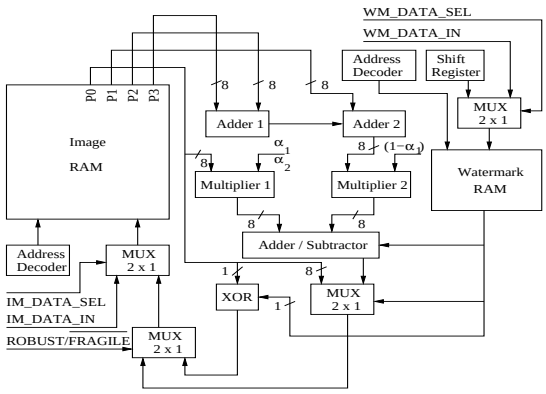


Fig. 6. Datapath for combined robust/fragile watermarking

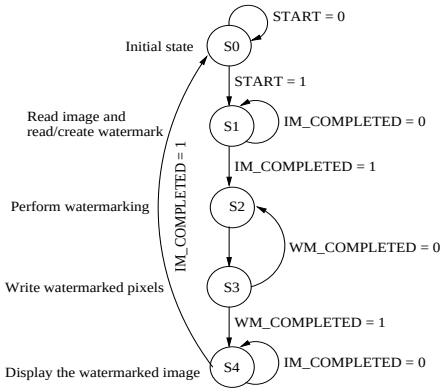


Fig. 7. Controller for combined robust/fragile watermarking

robust watermarking described above and is stored in watermark RAM. For watermark insertion, the 2^{nd} bit-line of the image pixels is fed as input to an XOR gate alongwith that of the watermark value. The output of the XOR gate is returned to the image RAM and the 2^{nd} bit-line is overwritten by selecting appropriate control signals.

The combined datapath for both robust and fragile watermarking is shown in Fig. 6. This datapath is obtained by stitching the above two datapaths (Fig. 4 and 5) using multiplexers, which in turn give rise to additional control signals. The controller that drives the datapath is shown in Fig. 7. The controller has five states, such as S0, S1, S2, S3 and S4. The state S0 is the initial sate. In state S1, image and watermark data are written to the respective RAMs. The image and watermark pixels are read from RAMs in state S2 and watermarking insertion is performed. In state S3, watermarked pixels are written back to the image RAM. In state S4, the watermarked image is ready in the RAM. The control signals and their functional descriptions are given below.

IM_ADDR_COUNT	: Signal for counters used to generate address for image
WM_ADDR_COUNT	: Signal for counters used to generate address for W
IM_$\overline{\text{READ}}$/WRITE	: Image RAM read/write
WM_$\overline{\text{READ}}$/WRITE	: Watermark RAM read/write
IM_DATA_SELECT	: Original/watermarked image
WM_DATA_SELECT	: Input/generate watermark
IM_ADDR_SELECT	: Location of image
WM_ADDR_SELECT	: Select address of watermark
START	: Watermarking begins when 1
IM_COMPLETED	: Set to 1 when all the pixels of the image are covered
WM_COMPLETED	: Set to 1 when all the pixels in watermark are covered
BUSY	: High as long as the watermarking continues
DATA_READY	: High when watermarked image is ready to be read
ROBUST/$\overline{\text{FRAGILE}}$	: Choose robust or fragile

5. CHIP IMPLEMENTATION

The implementation of the watermarking datapath and controller was carried out in the physical domain using the Cadence Virtuoso layout tool using bottom-to-top hierarchical design approach. The design involved the construction of three main modules, such as memory, watermarking module (datapath) and controller unit. Each of the three modules were designed individually through modularization and later interfaced with each other. The layouts of the gates at the lowest level of hierarachy is drawn using the CMOS standard cell design approach. We designed a standard cell library containing basic gates, such as AND, OR, NOT and 1-bit RAM cell.

The memory module involves two read/write memory structure, one for 256×256 size original/watermarked image and other for 128×128 size watermark. The bit size for the image RAM is $8-$bits and for watermark RAM it is $2-$bits. The basic building block for a memory module is a $6-$transistor static RAM cell available in the cell library. The memories are built as $n \times n$ arrays of these SRAM cell and are addressed using two address decoders each, one for the row decoding and other for the column decoding. Each decoder is implemented as a $m-$bit counter with additional AND-logic to address 2^m cells.

The watermarking module involves the implementation of two watermarking algorithms as described in section 3. The main components of this module are two 8-bit adders, two 8-bit multipliers and a 8-bit adder/ subtractor. Each adder is constructed using 1-bit adders in a ripple-carry manner. The adder/subtractor unit is obtained from the adder using XOR gates. The carry inputs to the adder/ sub-

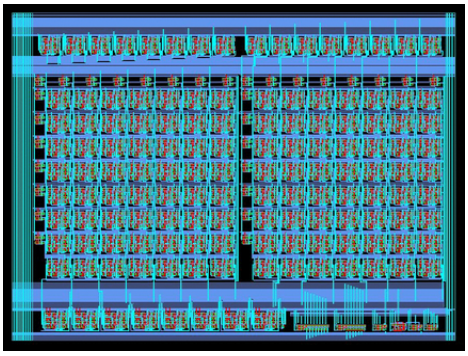


Fig. 8. Layout of the watermarking module (datapath)

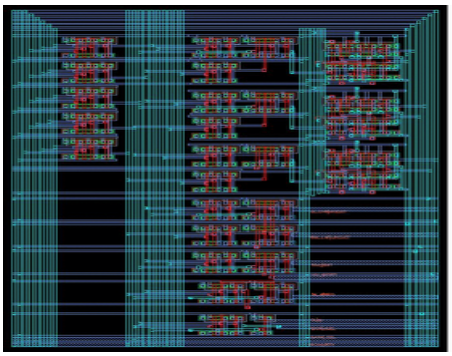


Fig. 9. Layout of the watermarking controller

tractor and one of the inputs to the XOR gate are set to high whenever the watermark pixel value is "2" so that a subtraction is carried out as required for the robust watermarking encoding function (Eqn. 2). An 8-bit parallel array multiplier is obtained from full-adders and AND gates to implement multiplication operations with reduced delay. Multiplexers are used at appropriate places in the design to select one of the incoming lines. Each of such multiplexer is implemented using a combination of transmission gates. Three asynchronously resettable registers are designed to encode the five states of the controller depicted in Fig. 7. At anytime, the three registers could be reset by the user to return the controller to its intial state and from there, the watermarking function could be started afresh.

Each of the above mentioned modules were implemented and tested separately and then connected together to obtain the final chip. The number of gates, power and areas of each module is shown below for operating voltage of $3.3V$.

Modules	Gate Count	Power (mW)	Delay (ns)
Datapath	4547	1.1931	0.9158
Controller	233	0.0045	0.3901
RAM	1183,744	21.8012	2.3891

The statistics are obtained using HSPICE for 0.35μ MOSIS SCN3M SCMOS technology. It is evident from the above statistics that the RAM is the largest power consuming module is this implementation. If we assume that the proposed chip is to be used as a module in any existing JPEG enoder, which in turn will be part of a bigger system, then in that case we do not need the memory in the watermarking chip explicitly. In this scenario, the power consumption will be significantly less, so also the area and delay. Layout of the datapath is shown in Fig. 8. and layout of the controller is shown in Fig. 9. The complete layout of the watermarking chip is given in Fig. 10. The floor plan of the chip is written on the top for clear understanding. The the pin diagram of the chip showing the input and outputs is given in Fig. 11. The overall design statistics of the chip are as follows.

area (with RAM)	$15.01 \times 14.22mm^2$
number of gates (with RAM)	$1188K$
number of gates (without RAM)	4820
clock frequency (with RAM)	$151MHz$
clock frequency (without RAM)	$545MHz$
number of I/O pins	25
power (with RAM)	$24mW$
power (without RAM)	$2.0547mW$

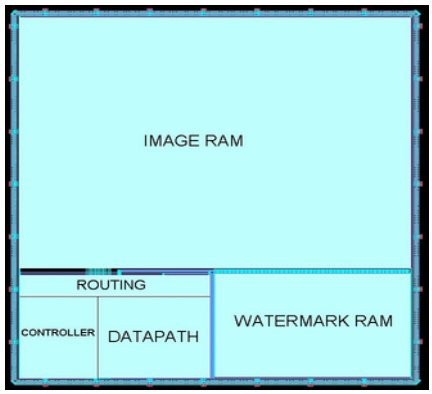


Fig. 10. Floorplan of the prototype watermarking chip

6. RESULTS AND CONCLUSIONS

After the chip is implemented, the functional verification is done by performing watermarking on various test images. We have shown two of such images in Fig. 12. The Fig. 12(b) shows the invisible robust watermarked shuttle image and Fig. 12(d) shows the invisible fragile watermarked bird image. As a quantitative measure of the perceptibility of the watermark, we used signal-to-noise ratio $\left(SNR = 10\ log\left(\frac{\text{Var}_I}{\text{Var}_{I_W}}\right)\right)$, as suggested by [5, 2, 10]. The Var_I is the variance of the original input image and the Var_{I_W} is the variance of the error image (difference between original input image watermarked image). For the watermarked image shown in Fig. 12(b) the SNR is $23dB$ and for the image shown in Fig. 12(d) it is $24dB$.

In this paper, we have presented a watermarking encoder

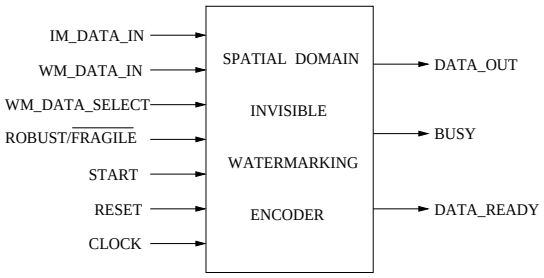
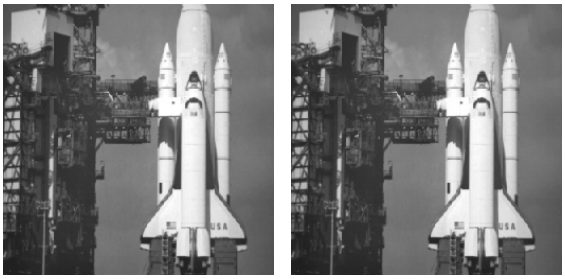


Fig. 11. Pin diagram for the proposed watermarking chip

that can perform invisible robust, invisible fragile watermarking and combination of both in spatial domain. To our knowledge, this is the first watermarking chip having both the functionalities. The chip can be easily integrated in any existing JPEG encoder to watermark the images right at the source end. The disadvantage of the watermarking algorithms implemented is that the processing needs to be done pixel by pixel. In future, we are aiming to implement algorithms that deal with block by block processing. The implementation of watermarking decoder which will be a part of JPEG decoder is currently under implementation. Similarly, a low power and/or high performance implementations is also considered.



(a) Original shuttle (b) Watermarked shuttle

(c) Original bird (d) Watermarked bird

Fig. 12. Watermarking of test images

7. REFERENCES

[1] S. Katzenbeisser and F. A. P. Petitcolas, *Information Hiding techniques for steganography and digital watermarking*, Artech House, Inc., MA, USA, 2000.

[2] S. P. Mohanty, K. R. Ramakrishnan, and M. S. Kankanhalli, "A Dual Watermarking Technique for Images," in *Proceedings of the 7th ACM International Multimedia Conference (Vol. 2)*, 1999, pp. 49–51.

[3] M. Kovac and N. Ranganathan, "JAGUAR: A Fully Pipelined VLSI Architecture for JPEG Image Compression Standard," *Proceedings of the IEEE*, vol. 83, no. 2, pp. 247–258, Feb 1995.

[4] 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*, vol. 147, no. 4, pp. 371–376, Aug 2000.

[5] N. J. Mathai, D. Kundur, and A. Sheikholeslami, "Hardware Implementation Perspectives of Digital Video Watermarking Algortithms," *IEEE Transanctions on Signal Processing*, 2003.

[6] T. H. Tsai and C. Y Lu, "A System Level Design for Embedded Watermark Technique using DSC System," in *Proceedings of the IEEE International Workshop on Intelligent Signal Processing and Communication System*, 2001.

[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*, 2003, pp. 792–796.

[8] A. Tefas and I. Pitas, "Robust Spatial Image Watermarking Using Progressive Detection," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (Vol. 3)*, 2001, pp. 1973–1976.

[9] F. Bartolini, M. Barni A. Tefas, and I. Pitas, "Image authentication techniques for surveillance applications," *Proceedings of the IEEE*, vol. 89, no. 10, pp. 1403–1418, Oct 2001.

[10] S. P. Mohanty, "Watermarking of Digital Images," M.S. thesis, Indian Institute of Science, Bangalore, India, 1999.