A VLSI Architecture for Visible Watermarking in a Secure Still Digital Camera (S^2DC) Design

Saraju P. Mohanty, Member of IEEE, Nagarajan Ranganathan, Fellow of IEEE, and Ravi K. Namballa

***Abstract*— Watermarking is the process that embeds data called a watermark, tag or label into a multimedia object, such as images, video or text for their copyright protection. According to human perception, the digital watermarks can either be visible or invisible. A visible watermark is a secondary translucent image overlaid into the primary image and appears visible to a viewer on a careful inspection. The invisible watermark is embedded in such a way that the modifications made to the pixel value is perceptually not noticed and it can be recovered only with an appropriate decoding mechanism. In this paper, we present a new VLSI architecture for implementing two visible digital image watermarking schemes. The proposed architecture is designed aiming at easy integration into any existing digital camera framework. To our knowledge, this is the first VLSI architecture for implementing visible watermarking schemes. A prototype chip consisting of 28469 gates is implemented using 0.35μ technology, which consumes $6.9mW$ power while operating at $292MHz$.**

***Index Terms*— Digital Watermarking, Spatial domain watermarking, visible and invisible watermarking, still digital camera, JPEG encoder.**

I. 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. The object may be an image, audio, video, or text [1]. Whether the host data is in spatial domain, discrete cosine transformed, or wavelet transformed, watermarks of varying degree of visibility are added to presenting media as a guarantee of authenticity, ownership, source, and copyright protection. In general, any watermarking scheme (algorithm) consists of three parts, such as, (i) the watermark, (ii) the encoder (insertion algorithm) and (iii) the decoder and comparator (verification or extraction or detection algorithm) [2], [3]. Whether each owner has a unique watermark or an owner wants to use different watermarks in different objects, the marking algorithm incorporates the watermark into the object. The verification algorithm authenticates the object determining both the owner and the integrity of the object.

Watermarks and watermarking techniques can be divided into various categories. The watermarks can be applied either in spatial domain or in frequency domain. It has been pointed out that the frequency domain methods are more robust than the spatial domain techniques [4]. On the other hand, the spatial domain watermarking schemes have less computational overhead compared to frequency domain schemes. According to human perception, the digital watermarks can be divided into four categories : (i) visible watermark, (ii) invisible-robust, (iii) invisible-fragile and (iv) dual [2], [3]. A visible watermark is a secondary translucent image overlaid into the primary image and appears visible to a casual viewer on careful inspection. The invisible-robust watermark is embedded in such a way that modifications made to the pixel value is perceptually not noticed and it can be recovered only with appropriate decoding mechanism. The invisible-fragile watermark is embedded in such a way that any manipulation or modification of the image would alter or destroy the watermark. A dual watermark is a combination of a visible and an invisible watermark [5]. In this type of watermark, an invisible watermark is used as a back up for the visible watermark.

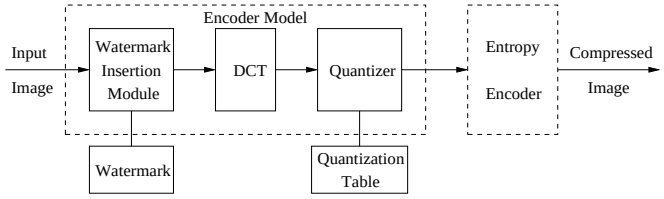


Fig. 1. Block Level View of a Secure JPEG encoder [6]

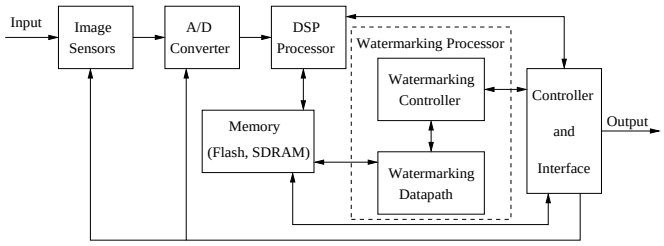


Fig. 2. System Architecture of a Secure Digital Still Camera

There are numerous software based watermarking schemes available in literature. A vast research community involving experts from computer science, cryptography, signal processing, and communications, etc. are working together to develop watermarks that can withstand different possible forms of attacks. Each one of which has its own its applications and thus, are equally important. There is a gap between the image capture and image transmission in the way watermarking is used presently. Once the images are acquired watermarks are inserted in them off-line and then images are made available. The objective of this research work to implement hardware

S. P. Mohanty is with Department of Computer Science and Engineering University of North Texas, Denton, TX 76203. N. Ranganathan is with Department of Computer Science and Engineering, University of South Florida, Tampa, FL 33620. E-mail:smohanty@cs.unt.edu and ranganat@csee.usf.edu

based watermarking schemes so as to bridge that gap. The watermark chip will be fitted in the devices that acquires the image and watermark the images in real time while capturing. In this work, we focus on the VLSI implementation of two visible watermarking schemes, one proposed by Braudaway, Magerlein, and Mintzer [7] and the other visible watermarking scheme proposed by Mohanty, Ramakrishnan, and Kankanhalli [5]. Both these algorithms operate in spatial domain of image data. The VLSI chip can insert either one of the watermarks at a time 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. An outline of such a secure JPEG encoder is provided in Fig. 1 [6]. 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. The proposed watermarking chip can also be directly integrated with any existing digital still camera. We provide the schematic view of a still camera that includes a watermarking module in Fig. 2, and call such a camera as a "secure digital still camera" (S^2DC). The S^2DC is conceptually similar to the "trustworthy digital camera" proposed by Friedman [8], in which cryptography is used for image authentication.

The rest of the paper is organized as follows. A brief overview of the existing watermarking chips is presented in Section II. The visible watermarking algorithms being implemented in this work and the modifications made to them are described in Section III. Section IV discusses the detailed architecture of the watermarking chip. The design of a prototype VLSI chip is discussed in Section V followed by experimental results and conclusions.

II. Related Work

Several watermarking algorithms have been presented in the literature for image, video, audio and text data. The watermarking schemes work in spatial, DCT, and wavelet domain. Moreover, the watermarking algorithms are invisible-robust, invisible fragile, etc. Although many software algorithms exists, very few hardware schemes have been proposed. In this section, we briefly visit the hardware based systems for watermarking. A comparative view of all the proposed watermarking chips are provided in Table I.

Strycker, et. al. [9] proposed a real-time watermarking scheme for television broadcast monitoring. They address the implementation of a real-time spatial domain watermark embedder and detector on a Trimedia TM-1000 VLIW processor developed by Philips semiconductors. In the insertion procedure, pseudo-random numbers are added to the incoming video stream. The depth of watermark insertion depends on the luminance value of each frame. The watermark detection is based on the calculation of correlation values. Mathai, Kundur and Sheikholeslami [10] present a chip implementation of the same video watermarking algorithm.

A DCT domain invisible watermarking chip is presented by Tsai and Lu [11]. The watermark system embeds a pseudo-random sequence of real numbers in a selected set of DCT coefficients. They also proposed a JPEG architecture which included the watermarking module. The watermark is extracted without resorting to the original image. The authors claim that the watermark is resistant to the JPEG attacks of as much as 10% compression ratio. The watermark chip is implemented using TSMC $0.35\mu m$ technology and occupies a die size of $3.064 \times 3.064mm^2$ for 46374 gates. The chip consumes $62.78mW$ power when operated at $50MHz$ with $3.3V$ supply voltage.

Garimella, et. al. [12] propose a VLSI architecture for invisible-fragile watermarking in spatial domain. 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 then compared with the original watermark for image authentication. The ASIC is implemented using 0.13μ technology. The area of the chip is $3453 \times 3453\mu m^2$ and consumes $37.6\mu W$ power when operated at $1.2V$. The critical path delay of the circuit is $5.89ns$.

Mohanty, Ranganathan and Namballa [6] describe a watermarking chip that has both invisible robust and invisible fragile watermarking functionalities in spatial domain. The invisible-robust algorithm proposed by Tefas and Pitas [13], [14] and the invisible-fragile algorithm proposed by Mohanty, Ramakrishnan and Kankanhalli [5] are implemented. In invisible-robust watermarking, a ternary watermark is embedded in the original image using encoding function that involves addition of a scaled grey value of neighboring pixels. A binary watermark generated from pseudorandom numbers are XORed with an original image bit-plane in the invisible-fragile watermarking scheme. The chip implemented using 0.35μ technology occupies an area of $15.01 \times 14.22mm^2$ and consumes $24mW$ when operated at $3.3V$ and $151MHz$ frequency.

In this paper, we propose a VLSI architecture that can insert visible watermarks in images. To our knowledge, this is the first watermarking chip that has such functionalities. Depending on the user's requirements, it can insert either of the watermarks. The spatial domain visible watermarking algorithms proposed by (i) Braudaway, Magerlein, and Mintzer [7] and (ii) Mohanty, Ramakrishnan, and Kankanhalli [5] have been implemented in this work. We first describe briefly the algorithms followed by the proposed VLSI architecture and the chip implementation.

III. Watermarking Algorithms

In this section, we discuss the image watermarking algorithms in brief and then discuss the modifications necessary to facilitate hardware implementation. The modifications are aimed at reducing silicon area through module sharing. The notations used in the description of the algorithms are given in Table II.

A. Visible Watermarking Algorithm 1 :

In general, visible watermarking has three goals: (i) the visible watermark should identify the ownership, (ii) the visual quality of the host image should be preserved, and (iii) the watermark should be difficult to remove from the host image. To satisfy these three conflicting criteria, schemes have been proposed for adding watermark with the original image. The

TABLE I
WATERMARKING CHIPS PROPOSED IN CURRENT LITERATURE

Proposed Work	Type of Watermark	Target Object	Working Domain	Technology	Chip Area	Chip Power Consumption
Mathai, Kundur and Sheikholeslami [10]	Invisible-Robust	Video	Wavelet	0.18μ	NA	NA
Tsai and Lu [11]	Invisible-Robust	Image	DCT	0.35μ	3.064×3.064 mm^2	$62.78mW$ $3.3V, 50MHz$
Garimella, and et. al. [12]	Invisible-Fragile	Image	Spatial	0.13μ	3453×3453 μm^2	$37.6\mu W$ $1.2V$
Mohanty, Ranganathan, and Namballa [6]	Invisible-Robust Invisible-Fragile	Image	Spatial	0.35μ	15.01×14.22 mm^2	$24mW$ $3.3V, 151MHz$

TABLE II
LIST OF VARIABLES USED IN ALGORITHM EXPLANATION

I	: Original (or host) image (a gray scale image)
W	: Watermark image (a gray scale image)
(m, n)	: A pixel location
I_W	: Watermarked image
$N_I \times N_I$	: Original image dimension
$N_W \times N_W$	: Watermark image dimension
i_k	: The k^{th} block of the original image I
w_k	: The k^{th} block of the watermark image W
$i_{W\,n}$	: The k^{th} block of the watermarked image I_W
α_k	: Scaling factor for k^{th} block (used for host image scaling)
β_k	: Embedding factor for k^{th} block (used for watermark image scaling)
μ_I	: Mean gray value of the original image I
$\mu_{k\,I}$	: Mean gray value of the original image block i_k
$\sigma_{k\,I}$	: Variance of the original image block i_k
α_{max}	: The maximum value of α_k
α_{min}	: The minimum value of α_k
β_{max}	: The maximum value of β_k
β_{min}	: The minimum value of β_k
I_{white}	: Gray value corresponding to pure white pixel
α_I	: A global scaling factor
C_1, C_2, C_3, C_4	: Linear regression co-efficients

visible watermarking algorithm proposed in [7] is discussed here. The watermarked image is obtained by adding a scaled gray value of the watermark image to the host image. The amount of scaling is done in such a way that the alternation of each original image pixel occurs to a perceptual equal degree. The original formulae have been simplified as shown below [15], where the scaling factor α_I determines the strength of watermark.

$$I_W(m,n) = \begin{cases} I(m,n) + W(m,n)\left(\frac{I_{white}}{38.667}\right)\left(\frac{I(m,n)}{I_{white}}\right)^{\frac{2}{3}}\alpha_I \\ \qquad \text{for } \frac{I(m,n)}{I_{white}} > 0.008856 \\ I(m,n) + W(m,n)\left(\frac{I(m,n)}{903.3}\right)\alpha_I \\ \qquad \text{for } \frac{I(m,n)}{I_{white}} \leq 0.008856 \end{cases} \quad (1)$$

The above equation can be simplified to make it amenable for hardware implementation. At the same time, it is ensured that the computation in hardware yields results that are as accurate as the software implementation. We assume $I_{white} = 255$ and simplify the above equations to the following.

$$I_W(m,n) = \begin{cases} I(m,n) + \left(\frac{\alpha_I}{6.0976}\right) W(m,n)(I(m,n))^{\frac{2}{3}} \\ \qquad \text{for } I(m,n) > 2.2583 \\ I(m,n) + \left(\frac{\alpha_I}{903.3}\right) W(m,n)\, I(m,n) \\ \qquad \text{for } I(m,n) \leq 2.2583 \end{cases} \quad (2)$$

The above expression involves cubic root calculation, which is complex to implement in hardware. So, we further simplify the above expressions and remove the cubic root function with a piecewise linear model. We divide the gray values range $[0,\, I_{white}]$ to four ranges, such as $\left[0,\, \frac{I_{white}}{4}\right]$, $\left[\frac{I_{white}}{4},\, \frac{I_{white}}{2}\right]$, $\left[\frac{I_{white}}{2},\, \frac{3I_{white}}{4}\right]$, and $\left[\frac{3I_{white}}{4},\, I_{white}\right]$. We fit four linear regression co-efficients that best approximates the cubic root in each of these ranges. Moreover, we roundup the fraction involved in the comparison operation and the final expression that is implemented using hardware is as follows.

$$I_W(m,n) = \begin{cases} I(m,n) + \left(\frac{\alpha_I}{903.3}\right) W(m,n)\, I(m,n) \\ \qquad \text{for } I(m,n) \leq 2 \\ I(m,n) + \left(\frac{\alpha_I C_1}{6.0976}\right) W(m,n)\, I(m,n) \\ \qquad \text{for } 2 < I(m,n) \leq 64 \\ I(m,n) + \left(\frac{\alpha_I C_2}{6.0976}\right) W(m,n)\, I(m,n) \\ \qquad \text{for } 64 < I(m,n) \leq 128 \\ I(m,n) + \left(\frac{\alpha_I C_3}{6.0976}\right) W(m,n)\, I(m,n) \\ \qquad \text{for } 128 < I(m,n) \leq 192 \\ I(m,n) + \left(\frac{\alpha_I C_4}{6.0976}\right) W(m,n)\, I(m,n) \\ \qquad \text{for } 192 < I(m,n) < 256 \end{cases} \quad (3)$$

We performed extensive software simulations for various test images and found that pixel values of the watermarked images obtained using the above set of equations match with that obtained using original equation.

B. Visible Watermarking Algorithm 2 :

In this subsection, we discuss the visible watermarking algorithm proposed in [5]. The pixel gray values are modified based on local and global statistics. The watermarking insertion process consists of the following steps.

- Both host image (one to be watermarked) I and the watermark (image) W are divided into blocks of equal sizes (the two images may be of unequal size).
- Let i_k denote the k^{th} block of the original image I and w_k denote the k^{th} block of the watermark W. For each block (i_k), the local statistics; mean $\mu_{k\,I}$ and variance

σ_{kI} are computed. The image mean gray value μ_I is also found out.

- The watermarked image block is obtained by modifying i_k as follows.

$$i_{W\,k} = \alpha_k\, i_k + \beta_k\, w_k \qquad k = 1, 2... \tag{4}$$

Where, α_k and β_k are scaling and embedding factors respectively, depending on μ_{kI} and σ_{kI} of each host image block.

The choice of α_k and β_k are governed by certain characteristics of human visual system (HVS) and mathematical models are proposed so that the perceptual quality of the image are not degraded due to watermark addition. The α_k and β_k are obtained as follows.

- The α_k and β_k for edge blocks are taken to be α_{max} and β_{min} respectively.
- The α_k and β_k are found out using the following equations.

$$\begin{aligned} \alpha_k &= \tfrac{1}{\hat{\sigma_{kI}}}\, exp\left(-(\hat{\mu_{kI}} - \hat{\mu}_I)^2\right) \\ \beta_k &= \hat{\sigma_{kI}}\, \left(1 - exp\left(-(\hat{\mu_{kI}} - \hat{\mu}_I)^2\right)\right) \end{aligned} \tag{5}$$

Where, $\hat{\mu_{kI}}$ and $\hat{\mu}_I$ are normalized values of μ_{kI} and μ_I, and $\hat{\sigma_{kI}}$ are normalised logarithm values of σ_{kI}.

- The α_k and β_k are scaled to the ranges (α_{min},α_{max}) and (β_{min}, β_{max}) respectively, where α_{min} and α_{max} are minimum and maximum values of scaling factor, and β_{min} and β_{max} are minimum and maximum values of embedding factor. These parameters determine the extent of watermark insertion. A linear transformation is used to scale current α_k and β_k values to the ranges (α_{min}, α_{max}) and (β_{min}, β_{max}), respectively. Let current values of α_k be written as α_k^c, and α_{min}^c and α_{max}^c, respectively denote the current minimum and maximum values. Similarly, let current values of β_k be written as β_k^c, and β_{min}^c and β_{max}^c, respectively denote the current minimum and maximum values. The α_k and β_k values are scaled as follows.

$$\begin{aligned} \alpha_k &= \left(\tfrac{\alpha_{max}-\alpha_{min}}{\alpha_{max}^c-\alpha_{min}^c}\right)\alpha_k^c \\ &\quad + \left(\alpha_{max} - \left(\tfrac{\alpha_{max}-\alpha_{min}}{\alpha_{max}^c-\alpha_{min}^c}\right)\alpha_{max}^c\right) \\ \beta_k &= \left(\tfrac{\beta_{max}-\beta_{min}}{\beta_{max}^c-\beta_{min}^c}\right)\beta_k^c \\ &\quad + \left(\beta_{max} - \left(\tfrac{\beta_{max}-\beta_{min}}{\beta_{max}^c-\beta_{min}^c}\right)\beta_{max}^c\right) \end{aligned} \tag{6}$$

We used first-order derivatives for edge detection. For horizontal edge detection, we compute the horizontal gradient as :

$$G_h(m,n) = I(m,n) - I(m+1,n) \tag{7}$$

The vertical gradient is computed as follows for vertical edge detection.

$$G_v(m,n) = I(m,n) - I(m,n+1) \tag{8}$$

The amplitude of an edge is calculated as,

$$G(m,n) = |G_h(m,n)| + |G_v(m,n)| \tag{9}$$

The mean amplitude for a block is computed as,

$$G_\mu = \tfrac{1}{N_B \times N_B} \sum_m \sum_n G(m,n) \tag{10}$$

When the mean amplitude for a block exceeds a predefined threshold, we declare it as an edge block. The values of m and n correspond to the pixel locations of individual blocks with reference to the original image pixel location.

The mean gray value of a block is calculated as the average of gray values of all pixels in the image block. The mean gray values are normalized with pure white pixel gray value. Thus, we have normalized mean gray values of a block as,

$$\hat{\mu_{kI}} = \tfrac{1}{N_B \times N_B}\left(\tfrac{1}{I_{white}}\right) \sum_m \sum_n I(m,n) \tag{11}$$

Where, m and n are the pixel locations of the k^{th} image block; same as their locations in the original image. The normalized standard deviation of gray values for the k^{th} block is calculated as follows.

$$\hat{\sigma_{kI}} = \tfrac{1}{N_B \times N_B}\left(\tfrac{2}{I_{white}}\right) \sum_m \sum_n \left|I(m,n) - \tfrac{I_{white}}{2}\right| \tag{12}$$

The exponential term in the Eqn. 5 is approximated as a power series. For $0 \le x \le 1$, we have the following Taylor series approximation which was used upto the square term in our implementation.

$$e^x = \textstyle\sum_i \frac{x^i}{i!} = 1 + x + \frac{1}{2}x^2 + ... \tag{13}$$

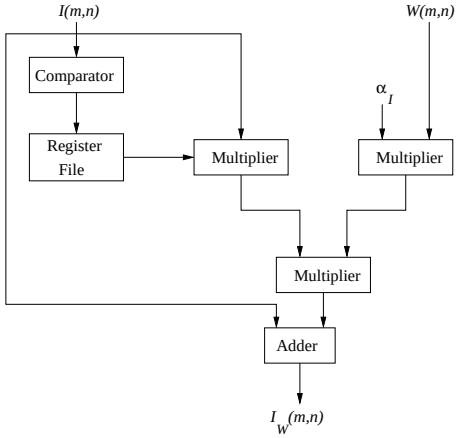
In the step three of the insertion algorithm, scaling needs to be done using a linear transformation. The transformation needs to find the current minimum and maximum values for both α_k and β_k over all the blocks. Due to this the hardware performance is going to be severely degraded since it has to wait till all the pixels of the images are covered to find local statistics of all the blocks. So, we modify the Eqn. 5 to ensure that the performance of the hardware is improved with no compromise on the quality. We find α_k and β_k using the following equations.

$$\begin{aligned} \alpha_k &= \alpha_{min} \\ &\quad + (\alpha_{max} - \alpha_{min})\, \tfrac{1}{\hat{\sigma_{kI}}}\, exp\left(-(\hat{\mu_{kI}} - \hat{\mu}_I)^2\right) \\ \beta_k &= \beta_{min} \\ &\quad + (\beta_{max} - \beta_{min})\, \hat{\sigma_{kI}}\, \left(1 - exp\left(-(\hat{\mu_{kI}} - \hat{\mu}_I)^2\right)\right) \end{aligned} \tag{14}$$

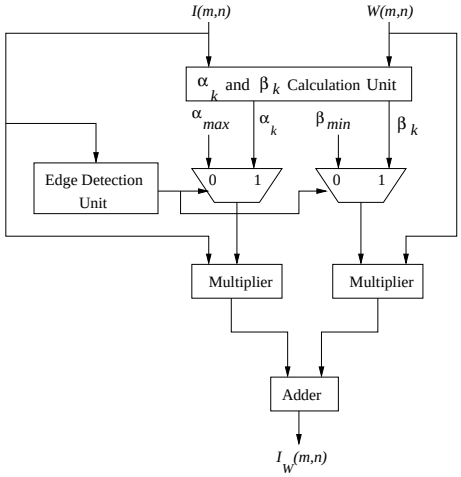
Extensive simulations for various images show that the α_k and β_k obtained using Eqn. 6 and Eqn. 14 are comparable (maximum difference is 5% [2]). Thus, we use Eqn. 14 for the α_k and β_k calculations.

IV. VLSI Architecture

In this section, we discuss the VLSI architectures for the Two algorithms discussed in Section III. The two architectures are combined to develop a single datapath with modules that can be shared by both algorithms. A finite state machine (FSM) based design of a controller that drives the datapath is described. We assume that both the original image and the watermark image are stored in the memory within the digital camera framework and are available for processing. The images may be in either a compressed format or as raw ascii data. We need to have a corresponding decoder to decode the image and get the uncompressed data in case it is in compressed format which was not part of this work.



(a) For Algorithm 1



(b) For Algorithm 2

Fig. 3. Datapath Architectures for the Visible Watermarking Algorithms

A. Architecture for Algorithm 1 :

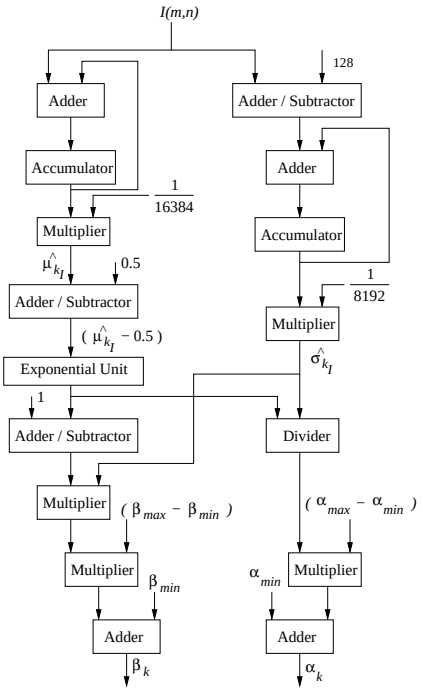
The insertion operation for the first watermarking algorithm is described in Eqn. 1. This insertion function is simplified to Eqn. 3 using a piecewise linear model such that we have a compact and efficient hardware design, as described in the previous section. Fig. 3(a) shows the architecture proposed for the first algorithm. The watermarking in this scheme is performed pixel-by-pixel as evident from the insertion function. A register file is used to store the constants needed to scale the image-watermark product in Eqn. 3. We store the constants $\frac{1}{903.3}$, $\frac{C_1}{6.0976}$, $\frac{C_2}{6.0976}$, $\frac{C_3}{6.0976}$, and $\frac{C_4}{6.0976}$. The other constant α_I is assumed as a parameter, which can be changed by user to vary the watermark strength. The comparator is used to determine the range in which a particular pixel gray value lies, such that an appropriate constant can be picked up from the register file. The left side multiplier calculates appropriate constant times the host image pixel gray values and the right side multiplier is used to find α_I times the watermark image pixel gray value. The results of the above two multipliers are fed to the third multiplier which effectively calculates the product of constants, α_I, host image pixel gray value, and watermark image pixel gray value, respectively. The above product is added to the host image pixel gray values using the adder to obtain watermarked image pixel gray values. The above described process has to be carried out for all the pixels in order to obtain the watermarked image.

B. Architecture for Algorithm 2 :

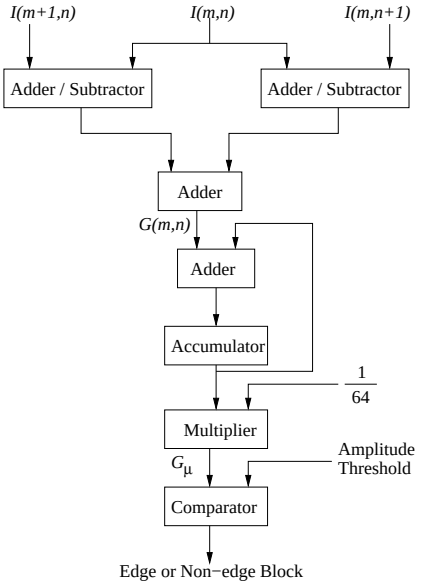
The architecture for the second watermarking algorithm is shown in Fig. 3(b) in which the watermarking insertion is performed block-by-block as described in Eqn. 4. For each block the watermarking insertion is performed on a pixel-by-pixel. The "α_k and β_k calculation unit" computes the α_k and β_k values for the k^{th} non-edge block using expression in Eqn. 14. The "edge detection unit" determines if a block is an edge block or non-edge block; if the G_μ exceeds a user defined threshold, then it is an edge-block. Larger the threshold more are the blocks declared as edge-blocks. The multiplexors help in selecting the scaling and embedding factors between the edge and non-edge blocks. The left side multiplier calculates the scaling factor times the host image pixel gray value. The right side multiplier multiplies the embedding factor with the watermark image pixel gray value. The products from these two multipliers are added using an adder to find the watermarked image pixel gray value. This process is repeated for all pixels in a block, and subsequently for all the blocks in the image.

1) α_k and β_k calculation unit: : The architectural details of "α_k and β_k calculation unit" is shown in Fig. 4(a). This hardware implements Eqn. 14 for α_k and β_k calculation for a block at a time. The left side adder-accumulator combination finds the sum of all the image pixel gray values for a block. After the sum is multiplied with $\left(\frac{1}{N_B \times N_B} * \frac{1}{I_{white}}\right)$, we get the normalized mean gray value of k^{th} block denoted by $\hat{\mu_{k}}_I$. Since we have assumed block size of 8×8, and I_{white} as 256, this evaluates to $\frac{1}{16384}$. It may be noted that I_{white} is 255, but using 256 makes hardware implementation easier, the latter being representable as a power of two. In the original algorithm $(\hat{\mu_{k}}_I - \hat{\mu}_I)$ is the deviation of a mean gray value of a block from the image mean gray value. We are evaluating the deviation of mean block gray value from mid-intensity of $\frac{I_{white}}{2}$ for simplicity, . Thus, $(\hat{\mu_{k}}_I - \hat{\mu}_I)$ is computed as $(\hat{\mu_{k}}_I - 0.5)$, when normalized with I_{white}. This assumption accelerates the hardware performance to a great extent since the block-by-block watermarking can be performed without waiting for the global image statistics computed over the whole image before the watermark insertion can be performed. The expression $exp\left(-(\hat{\mu_{k}}_I - \hat{\mu}_I)^2\right)$ is computed using the "exponential unit".

The adder/subtractor unit finds the image pixel gray value absolute deviation from $\frac{I_{white}}{2}$. The following unit, adder-accumulator accumulates the $\sum_m \sum_n \left|I(m,n) - \frac{I_{white}}{2}\right|$ for a block. When this sum is multiplied with $\left(\frac{1}{N_B \times N_B}\right) * \left(\frac{2}{I_{white}}\right)$, which is 8192 for our case, we get the normalized standard deviation $\hat{\sigma_{k}}_I$. The right side divider divides exponential value computed before by $\hat{\sigma_{k}}_I$. The quotient is then multiplied with $\alpha_{max} - \alpha_{min}$. The above product is added to α_{min} to evaluate α_k expressed in Eqn. 14. The exponential unit result is fed to a adder/subtractor on left side which finds



(a) Architecture of α_k and β_k Calculation Unit



(b) Architecture of Edge Detection Unit

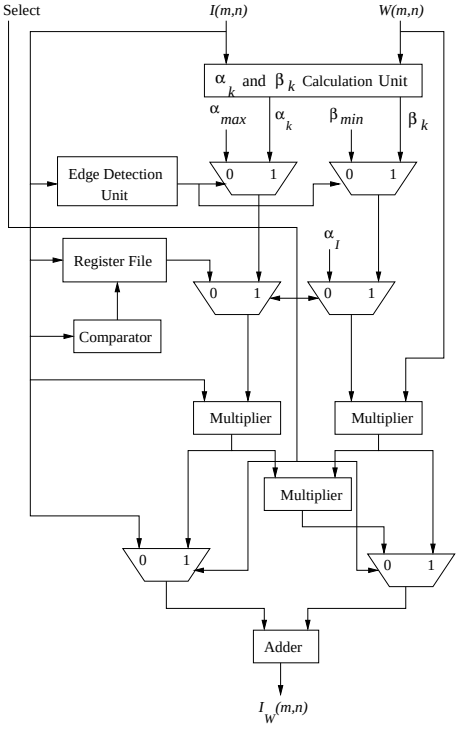
Fig. 4. Individual Datapath Units for Algorithm 2

its difference from 1. The result is then multiplied with $\hat{\sigma_{k}}_I$ obtained from the computations performed before. The product obtained is then multiplied with $\beta_{max} - \beta_{min}$. This product is then added to β_{min} which in turn gives the required β_k as per Eqn. 14.

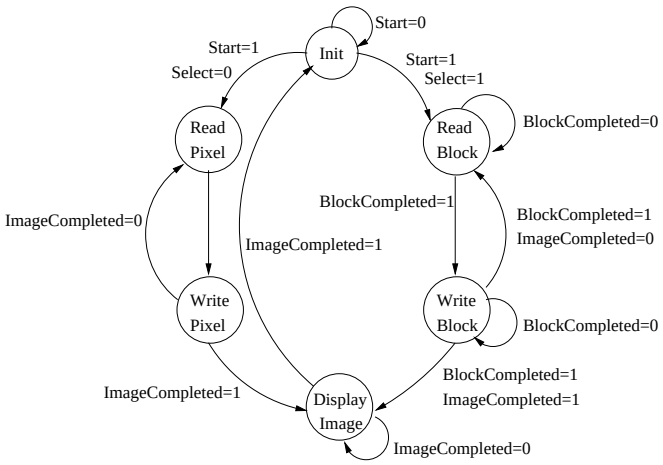
2) Edge detection unit: : The circuit to determine if a block is an edge or non-edge block is shown in Fig. 4(b). The left side and right side calculate the absolute value of horizontal gradient $|G_h(m,n)|$and absolute value of vertical gradient $|G_v(m,n)|$, respectively. The amplitude of an edge $G(m,n)$ is calculated using the first adder. Then, the adder-accumulator combination finds the sum of $G(m,n)$ for all pixels of a block. The above sum when multiplied with $\left(\frac{1}{N_B \times N_B}\right)$, we get the mean amplitude G_μ for a block. The comparator compares the G_μ values with an user defined threshold and declares the block as a edge or non-edge block.

C. Architecture for the Watermarking Processor :

The datapaths for both the algorithms shown in Fig. 3(a) and Fig. 3(b) are stitched together using multiplexors and the combined datapath is shown in Fig. 5(a). Both the algorithms share the same multipliers, as it is evident from Fig. 5(a), the multiplexors help in selecting input for the multipliers. The "Select" signal helps in choosing one of the watermarking scheme. When Select is "0" first algorithm is used and when select is "1", second algorithm is performed.



(a) Merged Datapath for Algorithms 1 and 2



(b) Controller for the Merged Datapath

Fig. 5. Architecture for the Proposed Watermarking Processor

The controller that drives the datapath is shown in Fig. 5(b). The controller has six states, such as Init, ReadBlock, WriteBlock, ReadPixel, WritePixel, and DisplayImage. When the Start signal is "1" the watermarking process is initiated. Depending on the Select signal one of the watermarking schemes is chosen and the corresponding datapath needs to be driven to carry out the watermarking process. When Select is "0", first watermarking scheme is chosen. At the ReadPixel state a pixel is read and the watermarked pixel is written at the WritePixel state after watermarking is performed. The process continues as long as ImageCompleted is "0" so that watermarking can be performed over all the pixels of the image.

The second algorithm is chosen when the Select is "1". In the ReadBlock state the pixel gray values are read for a block. The watermarked image block is written in the WriteBlock state once the watermarking is completed for the block. The system loops between the two states as long as all the blocks of the host image are not watermarked. Once, the watermarking is performed over whole image, the ImageCompleted signal is set to "1"; thus, completing the watermarking process. State DisplayImage is the state at which the watermark image is ready in the digital camera storage.

V. 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 main units, such as the exponential unit, the edge detection unit, the α_k and β_k calculation unit, register file, and the accumulator. All of the above units have multipliers, adders, adder/subtractor, divider, comparator, and so on. These small functional units are laid out individually through modularization and later interfaced with each other to get the above mentioned units. The datapath and the controller are constructed using the main units and the functional units. The layouts of the gates at the lowest level of hierarchy is drawn using the CMOS standard cell design approach. We designed our own standard cell library containing basic gates, such as AND, OR, NOT.

The datapath construction involves the implementation of the proposed architecture in the previous section. The fundamental functional units are 8-bit adders, 8-bit multipliers and 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 [16]. The carry inputs to the adder/ subtractor and one of the inputs to the XOR gate are set to high whenever the select signal for this unit is "2" so that a subtraction is carried out. The output of the adder-subtractor module gives the absolute value of the difference of two numbers when the difference is positive. When the difference is less than 0 (which is indicated by the carry bit taking a value 0), the absolute value is obtained by taking the 2's complement of the output of the adder/subtractor module.

An 8-bit parallel array multiplier is obtained from full-adders and AND gates to implement multiplication operations with reduced delay [17]. The divider is implemented using the shift and subtract logic for the division [16]. The number to be divided is initially stored in two registers, A and Q, and with each subtraction, the values in A and Q are shifted left, with the most-significant bit in Q replacing the least-significant bit in A, and a 1 placed in the least-significant bit of Q. If the value in A is less than that of the divisor, the same shift procedure is repeated, except that a 0 is placed in the least-significant bit of Q. Finally, the quotient is available in the register Q, and the remainder in A.

The comparator was designed to compare the values of two 8-bit numbers for greater-than, equal to, or less-than relations. First, a single-bit comparator was designed to compare the values of two single-bit numbers, and later, instances of this module were cascaded to compare two 8-bit numbers, starting from the most-significant bit position and proceeding towards the least-significant bit position.

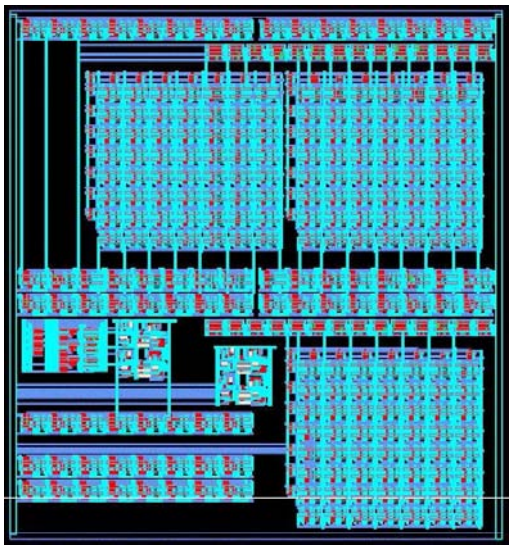
The accumulator is implemented as a 14-bit register to accommodate a maximum value of 64×256. The maximum value occurs when each pixel in a 8×8 block assumes the value of pure white pixel gray value. The register file is an addressable array of 8-bit registers (words) [17]. Based on the address specified and a Read/Write select line, at any time, a value can be either written to or read from the register file. Here, we used a 5-word register file to store the five different constants, such as $\frac{1}{903.3}$, $\frac{C_1}{6.0976}$, $\frac{C_2}{6.0976}$, $\frac{C_3}{6.0976}$, and $\frac{C_4}{6.0976}$, in Eqn. 3. Multiplexors are used at appropriate places in the design to select one of the incoming lines. Each of such multiplexor 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. 5(b). The three registers could be reset by the user to return the controller to its initial state at any time and from there, the watermarking function could be started afresh.

Each of the above mentioned modules are 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 in Table III for operating voltage of $3.3V$. The statistics are obtained using HSPICE for 0.35μ MOSIS SCN3M SCMOS technology. It is assumed that the proposed chip is to be used as a module in any existing JPEG encoder or a digital camera, and use their memory.

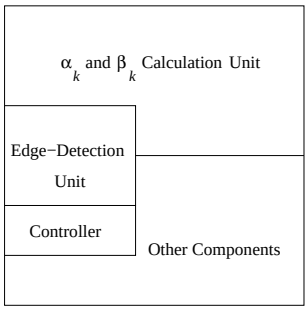
TABLE III
Power and Area of different Units

Modules	Gate Count	Power (mW)	Delay (ns)
Exponential unit	2370	1.2314	0.8981
Edge detection unit	3599	1.4137	1.0967
α_k and β_k calculation unit	16279	3.444	2.0241
Controller	163	0.0034	0.3201

The complete layout of the watermarking chip is given in Fig. 6(a) and the floor plan of the chip is provided in Fig. 6(b). The clock frequency is driven by the critical delay of the watermarking module. Table IV shows the overall design details of the chip and the corresponding pin diagram is shown in Fig. 7.



(a) Chip Layout



(b) Chip Floor Plan

Fig. 6. Layout and Floor Plan of the Proposed Watermarking Chip

TABLE IV
OVERALL STATISTICS OF THE WATERMARKING CHIP

Area	$3.34 \times 2.89mm^2$
Number of gates	28469
Clock frequency	$292.27MHz$
Number of I/O pins	72
Power	$6.9286mW$

VI. EXPERIMENTAL RESULTS

Each of the functional units is simulated individually before being integrated together to develop the whole chip. The functional verification of the whole chip is done by performing watermarking on various test images. Fig. 8 shows various test images and the watermark image used, which are borrowed from [5], [18], [19], [2]. The test images as well as the watermark images are of 256×256 dimension. The watermarked images obtained using the first algorithm is shown in Fig. 9. For this algorithm, the values of α_{min}, α_{max}, β_{min}, and β_{max} are assumed as $0.95, 0.98, 0.02$, and 0.07, respectively. Similarly, Fig. 10 shows the watermarked images obtained using the second algorithm, assuming α_I as 0.03. The regression co-efficients, such as C_1, C_2, C_3, and C_4, are respectively found to be $0.339644, 0.21988, 0.185746$, and 0.172925 using simulations.

A visual inspection of the watermarked images shows that

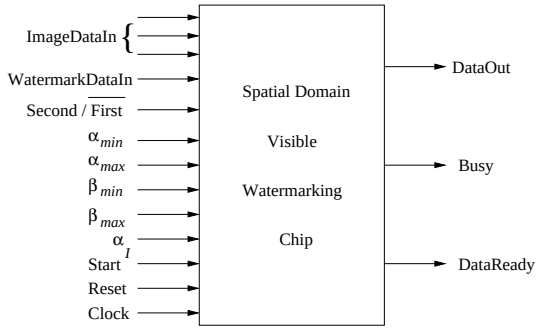


Fig. 7. Pin diagram for the Proposed Watermarking Chip

the watermarking process is able to preserve the quality of the image while explicitly proving the ownership. Of the various quantitative measures available to quantify the quality of the watermarked images, we used signal-to-noise ratio (SNR) given in Eqn. 15 as suggested by [10], [5], [2].

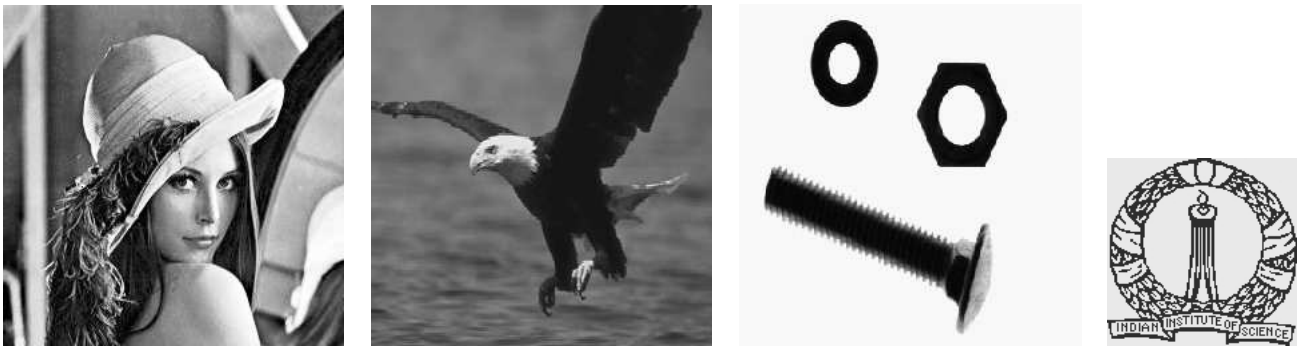
$$SNR = 10\ log\left(\frac{\text{Var}_I}{\text{Var}_{I_E}}\right) \quad (15)$$

The Var_I is the variance of the original input image and the Var_{I_E} is the variance of the error image (difference between original input image and watermarked image). We calculated the SNR using the original and the watermarked image with the help of a software simulator. Simulation results show that the SNR for various watermarked images is in the range of $20dB$ to $25dB$.

To verify whether the proposed chip produces results as effective as the software implementations we have conducted several tests. The algorithms we have chosen for our implementation are well accepted algorithms and are proven to be satisfying the vis-a-vis goals of the watermarking scheme. Thus, as long as the pixel values of a watermarked image from the hardware implementation matches with the pixel values of the same watermarked image obtained using software implements, we prove that hardware implementation do match with software implementations in satisfying the goals. First of all, the visual inspection of the watermarked images shown above match with that of the software schemes. We calculated the signal-to-noise ratio (SNR) of the watermarked images obtained using the proposed chip and also of the watermarked images obtained using software schemes. The SNR in both hardware and software schemes found to be approximately same; thus, proving effectiveness of the proposed chip. Further, we compared the values of the scaling factors and embedding factors (α_ks and β_ks) for both hardware and software schemes for the second algorithm. It is observed that values of the scaling and embedding factors obtained from the chip and that of the softwares are approximately the same.

VII. CONCLUSIONS

In this paper, we presented a watermarking chip that can be integrated within a digital camera framework for watermarking images. The watermarking chip can also be integrated in any existing JPEG encoder. The chip has two different types of watermarking capabilities, in spatial domain. To our knowledge, this is the first watermarking chip having visible watermarking



(a) Lena (b) Bird (c) Nuts and Bolts (d) Watermark

Fig. 8. Original Host Images (a, b, and c) and Watermark Image (d)

(a) Lena (b) Bird (c) Nuts and Bolts

Fig. 9. Watermarked Images for the First Algorithm

(a) Lena (b) Bird (c) Nuts and Bolts

Fig. 10. Watermarked Images for the Second Algorithm

functionalities. Out of the two watermarking schemes implemented, the first one does pixel-by-pixel processing and the second one is a block-by-block processing algorithm. Both algorithms are comparable in terms of SNR values. The design can be improved by a datapath organization in which the blocks can be pipelined to get better throughput.

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, "Watermarking of Digital Images," M.S. thesis, Indian Institute of Science, Bangalore, India, 1999.

[3] N. Memon and P. W. Wong, "Protecting Digital Media Content," *Communications of the ACM*, vol. 41, no. 7, pp. 34–43, Jul 1998.

[4] I. J. Cox, J. Kilian, F. T. Leighton, and T. Shamoon, "Secure Spread Spectrum Watermarking for Multimedia," *IEEE Transactions on Image Processing*, vol. 6, no. 12, pp. 1673–1687, Dec 1997.

[5] 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.

[6] S. P. Mohanty, N. Ranganathan, and R. K. Namballa, "VLSI Implementation of Invisible Digital Watermarking Algorithms Towards the Developement of a Secure JPEG Encoder," in *Proceedings of the IEEE Workshop on Signal Processing Systems*, 2003, pp. 183–188.

[7] G. W. Braudaway, K. A. Magerlein, and F. Mintzer, "Protecting Publicly Available Images with a Visible Image Watermark," in *Proceedings of the SPIE Conference on Optical Security and Counterfiet Deterrence Technique (Vol. SPIE-2659)*, 1996, pp. 126–132.

[8] G. L. Friedman, "The Trustworthy Digital Camera : Restoring Credibility to the Photographic Image," *IEEE Transanctions on Image Processing*, vol. 39, no. 4, pp. 905–910, Nov 1993.

[9] 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.

[10] N. J. Mathai, D. Kundur, and A. Sheikholeslami, "Hardware Implementation Perspectives of Digital Video Watermarking Algortithms," *IEEE Transanctions on Signal Processing*, vol. 51, no. 4, pp. 925–938, April 2003.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] J. Meng and S. F. Chang, "Embedding Visible Video Watermarks in the Compressed Domain," in *Proceedings of the International Conference on Image Processing (Vol. 1)*, 1998, pp. 474–477.

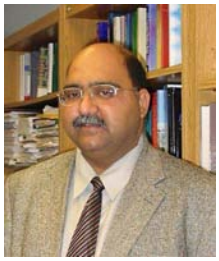
[16] V. P. Nelson, H. T. Nagle, J. D. Irwin, and B. D. Caroll, *Digial Logic Analysis and Design*, Prentice Hall, Upper Saddle River, New Jersey, USA, 1995.

[17] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design : A Systems Perspective*, Addison Wesley, Boston, MA, USA, 1999.

[18] 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*, 2000, pp. 1029–1032.

[19] S. P. Mohanty, K. R. Ramakrishnan, and M. S. Kankanhalli, "An Adaptive DCT Domain Visible Watermarking Technique for Protection of Publicly Available Images," in *Proceedings of the International Conference on Multimedia Processing and Systems*, 2000, pp. 195–198.

Saraju P. Mohanty (S'00-M'04) obtained the B. Tech. (First Class Honors) degree in Electrical Engineering from College of Engineering and Technology, Orissa University of Agriculture and Technology, Bhubansewar, India in 1995. He received the Master of Engineering degree in Systems Science and Automation from the Indian Institute of Science, Bangalore, India in 1999. He obtained Ph.D. in Computer Science and Engineering from the University of South Florida in 2003. Dr. Mohanty is currently an assistant professor at the department of Computer Science and Engineering, University of North Texas. He has published several research papers in areas of VLSI design automation, VLSI design and Digital watermarking. His research interests include CAD for nanometer VLSI circuits, low power synthesis, power aware system design, high level synthesis, and VLSI signal processing. He is a member of IEEE-CS and ACM-SIGDA.



Nagarajan Ranganathan (S'81-M'88-SM'92, F'02) received the B.E. (Honors) degree in Electrical and Electronics Engineering from University of Madras, India in 1983, and the Ph.D. degree in computer science from the University of Central Florida, Orlando in 1988. His research interests include VLSI system design, design automation, power estimation and optimization, computer architecture and bioinformation processing. He has developed many special purpose VLSI chips for computer vision, image processing, pattern recognition, data compression and signal processing applications. He has co-authored about 200 papers in reputed journals and conferences and is a co-owner of five U.S. patents and one pending. He was elected as Fellow of IEEE for his contributions to algorithms and architectures for VLSI systems design.

Dr. Ranganathan is a member of IEEE, IEEE Computer Society, IEEE Circuits and Systems Society and the VLSI Society of India. He served as the Chair of the IEEE Computer Society Technical Committee on VLSI during 1997-2000. He has served on the program committees of international conferences such as ISLPED, ICCD, MSE, CAMP, ICPP, ISVLSI, IPPS, SPDP, VLSI Design and ICHPC. He has served on the editorial boards of various journals such as Pattern Recognition, Intl. Journal of VLSI Design, IEEE Transactions on VLSI Systems, IEEE Transactions on Circuits and Systems TCAS-II, and IEEE Transactions on CAS for Video Technology. He served as the Steering Committee Chair for the IEEE Transactions on VLSI Systems during 2000-02 and is its Editor-In-Chief for two consecutive terms 2003-06. He received the USF Division of Sponsored Research Outstanding Research Achievement Award in 2002, the USF President's Faculty Excellence Award in 2003, the Theodore-Venette Askounes Ashford Distinguished Scholar Award in 2003, and the SIGMA XI Scientific Honor Society Tampa Bay Chapter Outstanding Faculty Researcher Award in 2004. He was a co-recipient of two Best Paper Awards at the Intl. Conf. on VLSI Design in 1995 and 2004.

Ravi K. Namballa obtained the B. Tech. (First Class Honors) degree in Computer Science and Engineering from College of Engineering, Andhra University, Visakhapatnam, India in 2001. He received the Master of Science degree in Computer Engineering from the University of South Florida in Summer 2003. Mr. Namballa is currently working as a Systems Engineer at Aeolus Systems LLC, Clearwater, Florida. His main area of expertise lies in high-level synthesis, developing high-speed architectures for digital watermarking algorithms, and creating interfaces for high-speed digital cameras.