VLSI Architecture and Chip for Combined Invisible Robust and Fragile Watermarking

Saraju P. Mohanty
Computer Science and Engineering
University of North Texas, Denton, TX 76203.
E-mail: smohanty@cse.unt.edu
Elias Kougianos
Electrical Engineering Technology
University of North Texas, Denton, TX 76203.
E-mail: eliask@unt.edu
N. Ranganathan
Computer Science and Engineering
University of South Florida, Tampa, FL 33620.
E-mail: ranganat@cse.usf.edu

This journal paper is based on our following shorter peer-reviewed conference versions:
S. P. Mohanty, R. Kumara C., and S. Nayak, "FPGA Based Implementation of an Invisible-Robust Image Watermarking Encoder," *Lecture Notes in Computer Science (LNCS), CIT 2004*, Springer-Verlag, Vol. 3356, pp. 344-353, 2004.
S. P. Mohanty, N. Ranganathan, and R. K. Namballa, "VLSI Implementation of Invisible Digital Watermarking Algorithms Towards the Development of a Secure JPEG Encoder," in *Proceedings of the IEEE Workshop on Signal Processing Systems (SIPS)*, pp. 183-188, 2003.

Abstract

Research in digital watermarking is mature. Several software implementations of watermarking algorithms are described in the literature, but few attempts have been made to describe hardware implementations. The ultimate objective of the research presented in this paper was to develop low-power, high- performance, real-time, reliable and secure watermarking systems, which can be achieved through hardware implementations. In this paper, we discuss the development of a very-large-scale integration architecture for a high-performance watermarking chip that can perform both invisible robust and invisible fragile image watermarking in the spatial domain. We prototyped the watermarking chip in two ways: (i) by using a Xilinx field-programmable gate array and (ii) by building a custom integrated circuit. To the best of our knowledge, this prototype is the first watermarking chip with both invisible robust and invisible fragile watermarking capabilities.

I. Introduction and Motivation

Owing to the usage of Internet, concerns about protecting and enforcing intellectual property (IP) rights of the digital content are mounting. Unauthorized replication and manipulation of digital content is relatively easy and can be achieved with inexpensive tools. Digital rights management (DRM) systems [1], [2] address issues related to ownership rights of digital content. Various aspects of content management – namely, content identification, storage, representation, and distribution – and IP rights management are highlighted in DRM. Although unauthorized access of digital content is being prevented by implementing encryption technologies, these approaches do not prevent an authorized user from illegally replicating the decrypted content. Digital watermarking is one of the key technologies that can be used in DRM systems for establishing ownership rights, tracking usage, ensuring authorized access, preventing illegal replication, and facilitating content authentication. Therefore, a two-layer protection mechanism utilizing both watermarking and encryption is needed to build effective DRM systems that can address IP rights and copyright issues [3]. We recently designed a high-performance, high-throughput, and area-efficient very-large-scale (VLSI) architecture for the Rijndeal Advanced Encryption Standard (AES) algorithm [4]. This architecture and corresponding chips will eventually lead to a complete hardware-based DRM system, the ultimate objective of our ongoing research.

In this paper, we address the invisible watermarking aspect of DRM. Digital watermarking is the process of embedding data, called a watermark, into a multimedia object such that the watermark can be detected whenever needed for DRM. The object may be an image, audio, video, text, or graphics [5], [6]. However, in this paper, "image" is the primary multimedia object, but similar work can be undertaken for other multimedia objects. In general, any watermarking algorithm consists of three parts: the watermark, the encoder (insertion algorithm), and the decoder and comparator (verification or extraction or detection algorithm) [6], [7]. An entity, called the watermark key, which is unique and exhibits a one-to-one correspondence with every watermark, is also used during the process of embedding and detecting the watermark. The key is private and known only to authorized parties, eliminating the possibility of illegal usage of digital content.

Watermarks and watermarking techniques can be divided into different categories in various ways [6], [8], [9]. Watermarks can be embedded in various domains, including the spatial and the frequency domains. The

various transformations that have been used extensively as alternatives to the spatial domain are the discrete cosine transform (DCT), the Fourier transform (FT), and the wavelet transform (WT). Frequency-based methods have several advantages over spatial domain methods [10], [11], [12]. For example, DCT domain techniques are more robust to attacks, and the perceptible quality of DCT domain watermarked images is better. On the other hand, spatial domain watermarking algorithms have less computational overhead than frequency domain algorithms. Spatial domain watermarking algorithms can also be faster in terms of computational time and hence are more suitable for real-time applications. Thus, we have focused on spatial domain watermarking because our ultimate goal is to develop VLSI architectures and chips such that real-time watermarking in the framework of electronic components would be possible.

Digital watermarks can be divided into visible and invisible types, based on human perception [7], [13], [14], [15]. A visible watermark is a secondary translucent image overlaid onto the primary image [16], [17]. An invisible watermark, on the other hand, is completely imperceptible. An invisible robust watermark is embedded in such a way that alterations made to the pixel value are not noticeable and can be recovered only with the appropriate decoding mechanism [12]. An invisible fragile watermark is embedded in such a way that any manipulation or modification of the image would alter the watermark [18]. In this paper, we deal with both invisible robust and invisible fragile watermarking algorithms and discuss the development of a watermarking chip with these two capabilities.

Each of these watermarking algorithms has its own applications; thus, all are equally important. Over the past decade, numerous watermarking algorithms have been invented, and their software implementations have been demonstrated [19], [20]. However, only a few hardware implementations are presented in the literature. A hardware-based watermarking system can be designed on a field programmable gate array (FPGA) board, Trimedia processor board [21], or custom integrated circuit (IC) [22]. The choice between an FPGA and a cell-based IC is a trade-off among cost, power consumption, and performance [22], [23]. In this paper, we present both FPGA-based prototyping and a custom IC design to facilitate high-performance real-time watermarking at the source end when the image is captured by an electronic component like a digital camera.

The rest of the paper is organized as follows. Section II highlights the contributions of this paper. Section III discusses hardware-based watermarking systems described in the current literature. Section IV discusses the watermarking algorithms in detail. The architecture proposed for implementing the algorithms is described in Section V. Section VI discusses the FPGA prototype of the proposed watermarking chip. Section VII discusses the design and implementation of the custom VLSI chip based on the proposed architecture. The subsequent sections, Section VIII and Section IX, discuss the experimental results and conclusions, respectively.

II. Contributions of This Paper

The contributions of this paper are multifold. First, we introduce an invisible robust image watermarking algorithm and an invisible fragile image watermarking algorithm, both of which are spatial domain watermarking algorithms. The algorithms were first validated by using MATLAB simulation to evaluate their performance. The algorithms were developed such that their hardware implementation is simple yet provides high performance, enabling the

hardware to perform watermarking in real time. Two hardware prototypes were developed as follows: (1) FPGA-based implementation and (2) custom IC implementation. We followed a top-down approach in our FPGA-based prototyping. For a compact custom IC design with a minimal silicon area, we followed a bottom-up approach. We synthesized the prototype watermarking encoder chip in a Xilinx FPGA using VIRTEX technology. For the custom IC design, we used various Cadence tools with $0.35\mu m$ complementary metal-oxide semiconductor (CMOS) technology. A typical watermarking chip has four distinct modules: watermark encoder, watermark decoder, watermark generator, and controller that controls the operations. Most of the invisible watermarking algorithms described in the current literature as well as the algorithm presented in this paper use pseudorandom numbers as watermarks. Therefore, we focused on the structural design aspects of a watermarking generator that uses linear feedback shift registers (LFSRs). The encoder chip implemented using $0.35\mu m$ CMOS technology consumes $2.0mW$ when operated at $3.3V$ supply and $0.5GHz$ frequency.

III. Related Research

The current literature is rich in watermarking algorithms developed for various types of media, such as image, video, audio, and text data, and their software implementations. The algorithms work in various domains like spatial, DCT, and wavelet and insert-extract different types of watermarks including invisible robust, invisible fragile, and visible. These watermarking algorithms primarily work off-line; i.e., the images are first acquired and then the watermarks are inserted before the watermarked images are made available to the user. Thus, in this approach, there is a gap between image capture and image transmission. The objective of this research work was to develop a hardware-based watermarking system to bridge this gap. The watermark chip will be fitted in any electronic component that acquires the images, which are then watermarked in real time while capturing. In this section, we briefly discuss the few hardware-based watermarking systems mentioned in the current literature. These hardware-based watermarking systems were designed and implemented on an FPGA board, Trimedia processor board, or custom IC using different CMOS technologies.

Strycker et al. [24] have proposed a real-time watermarking algorithm in the spatial domain for television broadcast monitoring. They address the implementation of a real-time watermark embedder and detector on the Trimedia TM-1000 very long instruction width (VLIW) processor developed by Philips Semiconductor. In the insertion procedure, pseudorandom numbers are added to the incoming video stream based on the luminance value of each frame, and watermark detection is based on the calculation of the correlation values. Mathai et al. [23] describe a VLSI chip designed with $0.18\mu m$ CMOS technology implementing the above video watermarking algorithm.

A DCT domain invisible watermarking chip is presented by Tsai and Lu [25]. The watermarking system embeds a pseudorandom sequence of real numbers with a selected set of DCT coefficients and is extracted without using the original image. The chip is implemented with TSMC $0.35\mu m$ technology and has a die size of $3.064 \times 3.064mm^2$ and 46,374 gates. The chip is estimated to consume $62.78mW$ of power when operated at $50MHz$ frequency with a $3.3V$ supply.

Garimella et al. [26] have proposed a watermarking VLSI architecture for invisible fragile watermarking in the

spatial domain. In this scheme, the differential error is encrypted and interleaved along with the first sample. The watermark can be extracted by accumulating the consecutive least significant bits (LSBs) of the pixels and then decrypting them. The extracted watermark is then compared with the original watermark for image authentication. The application specific integrated circuit (ASIC) is implemented using $0.13\mu m$ technology. The area of the chip is $3453 \times 3453\mu m^2$, and the chip consumes $37.6\mu W$ of power when operated at $1.2V$. The critical path delay of the circuit is $5.89ns$.

Mohanty et al. [22] have proposed another watermarking hardware architecture that can insert two visible watermarks in images in the spatial domain. This architecture can insert either of the two watermarks depending on the requirements of the user. The chip is implemented with $0.35\mu m$ technology and occupies an area of $3.34 \times 2.89mm^2$ and consumes $6.9286mW$ when operated at $3.3V$ and $292.27MHz$.

Fan et al. [27] have proposed a visible watermarking design based on an adaptive discrete wavelet transform (DWT). They propose efficiently reduced operational and resource-sharing techniques using an existing algorithm. Host image and watermark are transformed into three-level multi-resolution structures. The host image signal is divided into two sequences with the same pattern length. Processing time is reduced by using a two-path parallel processing architecture. The signal is sent to different processing elements by the demultiplexers. The watermark image is embedded by modifying the coefficients of the image.

In this paper, we describe a VLSI architecture that implements both Invisible robust and invisible fragile watermarking functionalities in the spatial domain. In invisible robust watermarking, a ternary watermark is embedded in the original image with an encoding function that involves addition of a scaled gray value of neighboring pixels. A binary watermark generated from pseudorandom numbers is XORed with the original image bit plane in the invisible fragile watermarking algorithm. The VLSI architecture is prototyped with a Xilinx FPGA and a custom IC design.

IV. Invisible Watermarking Algorithms

In this section, we present an invisible robust image watermarking algorithm and an invisible fragile image watermarking algorithm whose VLSI architecture and chips are described in subsequent sections. The algorithms selected are simple and effective and, with modifications, can result in high-performance hardware that can perform watermarking in real time. We discuss the insertion and detection methods in brief, with the modifications necessary to facilitate hardware implementation. The notations used in our description of the algorithms are listed and defined in Table I.

In both algorithms described in the following subsections, we used a binary pseudorandom number sequence as the watermark. Pseudorandom number sequences with large periods have excellent randomness and correlation properties [28], [29], [30]. We anticipate that the watermark created with such a pseudorandom number sequence will have several distinct advantages, including the following:

- An authorized user who knows the watermark key that includes the initial sequence and the number of cycles can exactly reconstruct the original watermark whenever needed.

- An attacker who tries to tamper with the watermark can only succeed in swapping the pseudorandom number sequence without affecting its correlation properties. Thus, meaningful watermark detection is still possible.
- The techniques used for generating such pseudorandom number sequences are all compatible with hardware implementation. Such implementations would be capable of online, real-time algorithm execution, the primary objective of this paper.

TABLE I

NOTATIONS USED TO EXPLAIN SPATIAL DOMAIN WATERMARKING ALGORITHMS

I	Original image assumed as a grayscale image
W	Watermark image, which is binary or ternary
(i, j)	A pixel location
I_W	Watermarked image, which is also a grayscale image
$N_I \times N_I$	Original image I or watermarked image I_W dimension
$N_W \times N_W$	Watermark image W dimension
E, E_1, E_2	Watermark encoding functions
D	Watermark detection function
D_R	Watermark detection ratio
D_T	Watermark detection threshold
I_D	Difference image
r	Neighborhood radius
I_N	Neighborhood image, which is a grayscale image
K	Digital watermark key needed for watermark generation
α_1, α_2	Scaling constants to determine watermark strength
$\|\|.\|\|$	Cardinality function

A. Invisible Robust Watermarking Algorithm

Invisible robust image watermarking is based on the widely acceptable algorithm in [31], [32]. The algorithm works in the spatial domain. This algorithm is claimed to be robust to various major attacks, including geometric attack, as evident from benchmark testing like Stirmark.

The watermark insertion process is demonstrated with the help of the block diagram shown in Fig. 1(a). In the first step of the insertion process, a pseudorandom number generator generates a random sequence of ternary data. The watermark W is constructed out of this random sequence; thus, W is a ternary image (i.e., an image with three levels of gray values) having pixel values $\{0,1 \text{ or } 2\}$. The pseudorandom number generator uses a digital watermark key K as the initial sequence. Using encoding functions E_1 and E_2, the watermark insertion is performed by altering the pixels of the original image I as follows:

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

to obtain the watermarked image I_W. The encoding functions E_1 and E_2 are functions of the original image I and its neighborhood image I_N. For watermark strength factors α_1 and α_2, these encoding functions are defined as follows:

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

where α_1 and α_2 satisfy $1 > \alpha_1 > 0$ and $-1 < \alpha_2 < 0$. These conditions guarantee that the encoded pixel always has a positive value. The scaling $(1-\alpha_1)$ is used to scale I_N to ensure that the watermarked image gray value I_W does not typically exceed the maximum gray value for 8-bit image representation corresponding to a pure white pixel. In rare occasions when $I(i,j)$ is close to 255, the watermarked pixel $I_W(i,j)$ may exceed the maximum value. In that case, the value of $I_W(i,j)$ is truncated to 255. Our experimental results indicate that this truncation does not create a perceptible change in the image.

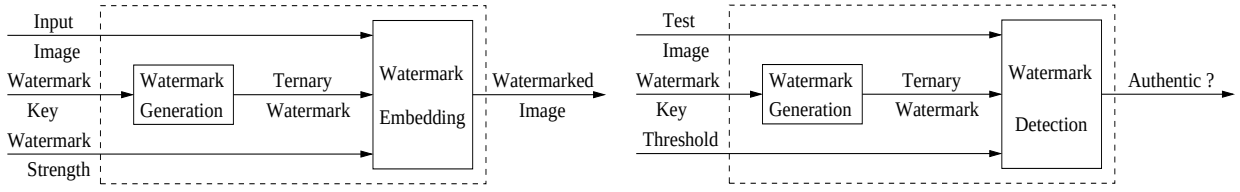
For a particular pixel location of the original image, the neighborhood image pixel gray value can be calculated from the gray values of the neighboring pixels. A given neighborhood radius r decides the number of neighborhood image pixel gray values to be used for its calculation. The neighborhood radius r determines the upper bound of the watermarked pixels in an image. For the smallest neighborhood radius, $r = 1$, it can be computed as the average of the three other pixel gray values:

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

This averaging requires division, which can be a costly operation. We instead aim to build fast and simple hardware. A division of four would be easily implemented using two right-shift by 1-bit operations. However, division by four of three neighborhood pixel gray values may reduce the accuracy of I_N. Thus, to ensure a proper tradeoff among accuracy, cost of computation, and cost of hardware, we propose the following function for the calculation of I_N:

$$I_N(i,j) = \left\{ \frac{\frac{I(i+1,j)+I(i+1,j+1)}{2} + I(i,j+1)}{2} \right\}. \tag{4}$$

The use of this averaging method simplifies hardware implementation because the division by two can be implemented by using a right-shift by 1-bit operation.



(a) Watermark Insertion (b) Watermark Detection

Fig. 1. Invisible Robust Watermarking in the Spatial Domain [31], [32]

The block diagram for the watermark detection is provided in Fig. 1(b). In the first phase of the detection process, the original ternary watermark W is generated with a pseudorandom number generator using the same watermark

digital key K and the same number of cycles used during the insertion process. The next step involves calculation of a difference image I_D using the detection function shown below:

$$I_D(i,j) = \begin{cases} 1 & \text{if } I_W(i,j) - I_N(i,j) > 0, \\ 2 & \text{if } I_W(i,j) - I_N(i,j) < 0. \end{cases} \tag{5}$$

In this function, I_W is the watermarked image under test, and I_N is the neighborhood image. This neighborhood image I_N is calculated from the original host image by using the function presented in Eqn. 4. After creation of the original watermark image W and the difference image I_D, the next step involves creation of a binary watermark image W^* as follows:

$$W^*(i,j) = \begin{cases} 1 & \text{if } W(i,j) \neq 0 \text{ and } W(i,j) \neq I_D(i,j), \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Using the original ternary watermark W and constructed binary watermark W^*, a detection ratio is determined as

$$D_R = 1 - \frac{||W^*||}{||W||}. \tag{7}$$

The detection ratio D_R is in essence the ratio of the correctly detected pixels to the sum of the watermarked pixels in the image. Ownership can be established when the detection ratio is larger than a predefined detection threshold.

Before proceeding with the development of the architecture and its corresponding FGPA and custom IC implementations, we performed simulations of the above algorithm. We used MATLAB as the simulation environment and tested the algorithm for various test images. For simplicity, we chose $\alpha_1 = -\alpha_2$ and performed our experiments with various values in the range $\alpha_1 = 0.98$ (weak watermark embedding) to $\alpha_1 = 0.1$ (very strong watermark embedding). The corresponding values of peak signal-to-noise ratio (PSNR) were in the range of $56dB$ to $21dB$ with no visual degradations.

B. Invisible Fragile Watermarking Algorithm

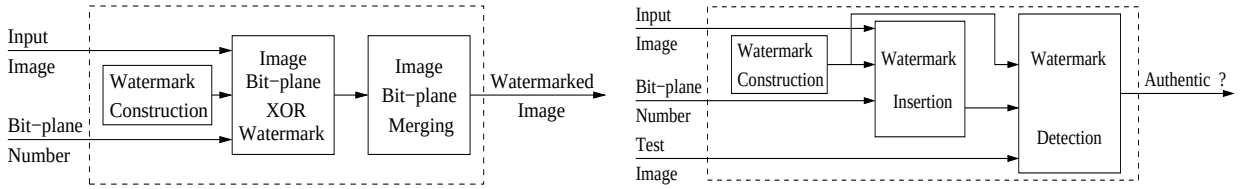
The invisible fragile watermark insertion was performed as presented in Fig. 2(a). The first step involved the generation of a pseudorandom binary sequence $\{0,\ 1\}$ of period N using a a pseudorandom number generator. The period N is equal to the number of pixels ($N_W \times N_W$) of the watermark image size that the user intends to create. The watermark image was constructed by arranging the binary sequence into 8×8 blocks. The size of the watermark image W was assumed to be the same as the size of the host image I. The bit planes of the grayscale input image were derived. A grayscale image I is represented as eight binary images $I[7]$, $I[6]$, $I[5]$, $I[4]$, $I[3]$, $I[2]$, $I[1]$, and $I[0]$, where "$I[0]$" is the LSB plane. In other words, a grayscale image I is represented as binary images $I[k]$, where k is an integer in the range $[0, 7]$, k denoting a specific bit plane. The binary watermark is inserted in the appropriate bit plane such that the PSNR is higher than a predefined threshold value. Assuming that the watermark insertion is to be performed in the k^{th} bit plane, the watermark insertion process is given by the

following expression:

$$\left.\begin{array}{lcl} I_W[0](i,j) & = & I[0](i,j), \\ ..... & & .....,\\ I_W[k-1](i,j) & = & I[k-1](i,j), \\ I_W[k](i,j) & = & I[k](i,j) \text{ XOR } W(i,j), \\ I_W[k+1](i,j) & = & I[k+1](i,j), \\ ..... & & .....,\\ I_W[7](i,j) & = & I[7](i,j), \end{array}\right\} \quad (8)$$

where the lower-order bit planes range from 0 to $k-1$ and the higher- order bit planes range from $k+1$ to 7, k being the inserting bit plane of the image.

After all the bit planes are merged, the watermarked image I_W is obtained. It may be noted that this algorithm, which involves conversion of grayscale to binary images, is inherently suitable for hardware implementation because in the hardware, the number system will be 2's complement binary. The identification of the candidate bit plane for watermark insertion is an iterative process. However, in order to develop area-efficient and high-performance hardware, we decided to select a particular bit plane and eliminate the iterative step. After running simulations with several hundred different images, we concluded that the $3^{rd}(k=2)$ bit plane is the best candidate for watermark insertion from the PSNR point of view. The typical PSNR for different test cases for k =0, 1, 2, and 3 was found to be $50dB$, $46dB$, $39dB$, and $33dB$, respectively. Thus, we decided to use the $3^{rd}(k=2)$ bit plane as the candidate for watermarking to avoid any perceptible impact on the image quality.



(a) Watermark Insertion (b) Watermark Detection

Fig. 2. Invisible Fragile Watermarking in the Spatial Domain

Fig. 2(b) highlights the steps of our fragile detection process. The first step consists of generating a pseudorandom binary sequence, followed by creating the binary original watermark W with the key and approach used during the insertion phase. We then created the watermarked image I_W following the same steps as in the watermark insertion process. Then, the cross-correlation of the watermarked image I_W, test watermarked image I_{W^*}, and binary watermark image W was calculated, followed by a test statistic computation. Depending on the value of the test statistic, the severity and extent of forgery of the the test watermarked image were determined. Following the approach in [30], we calculated the spatial cross-correlation function of images I and J as:

$$\chi_{IJ}(\alpha,\beta) = \sum_i \sum_j \{I(i,j) - J(i-\alpha, j-\beta)\}\,. \quad (9)$$

Assuming that W_k is the watermark image block, $I_{W\,k}$ is the watermarked image block, and I_{W^*} is the watermarked image block that might be forged, the test statistics for the block δ_k are calculated as follows [30]:

$$\delta_k = \chi_{I_W W}(0,0) - \chi_{I_{W^*} W}(0,0), \tag{10}$$

$$= \sum_i \sum_j \{I_W(i,j) - W(i,j)\} - \sum_i \sum_j \{I_{W^*}(i,j) - W(i,j)\}. \tag{11}$$

The average test statistics δ for all blocks is obtained as given below:

$$\delta = E\left[|\delta_k|\right] = \frac{1}{N}\sum_k \delta_k \text{ (where } N \text{ denotes the number of blocks)}. \tag{12}$$

In order to determine the values of δ to set a test paradigm, we performed extensive simulations with MATLAB. The watermarked image was tampered with by using the built-in functions of the image editing software and MATLAB. The average test statistics were calculated in all cases. Typical test statistics for the "Lena" image are shown in Table II for various quality factors (QFs) of the Joint Photographic Experts Group (JPEG). The testing paradigm can be established as follows:

- If $\delta < 9.0$, the watermarked image under test is perceptibly identical to the original watermarked image. It is fully authentic.
- If $9.0 \leq \delta < 50$, the watermarked image under test is forged. It is authentic.
- If $50.0 \leq \delta \leq 70$, the watermarked image under test is heavily forged. It is authentic.
- If $\delta > 70.0$, the watermarked image under test does not belong to the owner or has been severely tampered with.

V. Proposed VLSI Architectures for Invisible Watermarking

In this section, we discuss the architectures proposed in this paper for implementing the two algorithms discussed in Section IV. We first present individual architectures for the invisible robust algorithm and for the invisible fragile algorithm. Then we describe a unified architecture that can perform both invisible robust and invisible fragile watermarking. While developing the unified architecture, we shared the different modules so that individual modules could be used for performing either of the watermarking operations. Each of the proposed architectures (datapath and controller) has three distinct units or modules: watermark generation unit, watermark insertion unit, and controller unit that executes the overall watermarking process. We used embedded on-chip memory in each of the datapaths to temporarily buffer the data for computation. However, the datapaths can be developed without the memory units if they directly access off-chip memory.

A. Architecture for Robust Watermarking

The datapath for invisible robust watermarking is shown in Fig. 3. The image random access memory (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 the watermark data. The watermark data can either be generated using the linear feedback shift register (LFSR) or given as an external

TABLE II

TAMPERING OF A TEST IMAGE AND ITS TEST STATISTICS CALCULATION

Tampering Operations	Test Statistics (δ)	Observation
JPEG compression, QF 50%	12.6	Identical
JPEG compression, QF 25%	16.7	Identical
3 × 3 blur, JPEG compression, QF 25%	24.5	Identical
5 × 5 blur, JPEG compression, QF 25%	35.5	Heavily blurred
25% sharp, JPEG compression, QF 25%	17.2	Identical
50% sharp, JPEG compression, QF 25%	27.8	Identical
75% sharp, JPEG compression, QF 25%	43.1	Too sharp
3 × 3 spread, JPEG compression, QF 25%	71.3	Unrecognizable
Pixel size 2 × 2, JPEG compression, QF 25%	21.7	Identical
Pixel size 4 × 4, JPEG compression, QF 25%	32.1	Almost unrecognizable
Pixel size 6 × 6, JPEG compression, QF 25%	65.1	Unrecognizable
3 × 3 median filter, JPEG compression, QF 25%	22.4	Identical
5 × 5 median filter, JPEG compression, QF 25%	28.5	Blurred
7 × 7 median filter, JPEG compression, QF 25%	35.9	Too blurred
11 × 11 median filter, JPEG compression, QF 25%	51.6	Unrecognizable
Gaussian noise, mean = 0.0, variance 0.01	40.1	Identical
Gaussian noise, mean = 0.0, variance 0.02	56.2	Too noisy
Gaussian noise, mean = 0.0, variance 0.04	79.2	Too noisy

input by the user. In this hardware design, it is assumed that at any point, 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 the other regions (a total of four times for the assumed sizes). The region of the original image to be watermarked is described in terms of five parameters (top_left, top_right, center, bottom_left, and bottom_right), and address decoders are used to determine the proper locations.

The invisible robust watermark insertion algorithm involves adding (or subtracting) a constant times the image pixel gray value 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. 4 is computed as follows. First, the $I(i,j+1)$ and $I(i+1,j+1)$ are given to the adder 1 as input. The resulting sum and carryout from adder 1 are fed to adder 2 along with $I(i+1,j)$. The resulting sum of the adder 2 is the neighborhood function value. The division by two is performed by shifting the result 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 multipliers are fed to the adder/subtractor unit to perform the watermark insertion according to Eqn. 2. Because we are concerned only

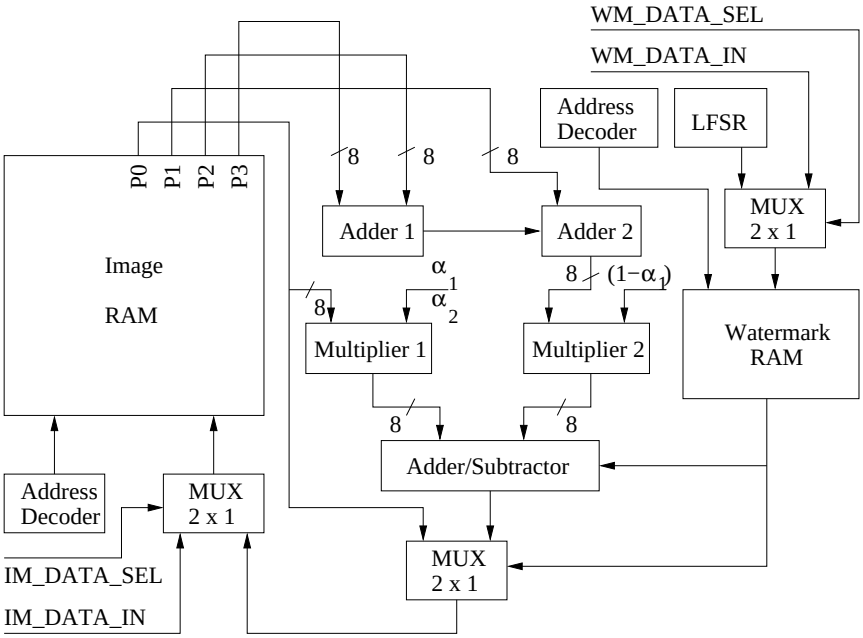


Fig. 3. Datapath for Invisible Robust Watermarking

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, according to Eqn. 1.

B. Architecture for Fragile Watermarking

The datapath for fragile watermark insertion is shown in Fig. 4. The original image is stored in the image RAM, and the watermark is created in the same way as in the case of robust watermarking described above and stored in the watermark RAM. For watermark insertion, the 3^{rd} bit line of the image pixels is fed as input to an XOR gate along with that of the watermark value. The output of the XOR gate is returned to the image RAM, and the 3^{rd} bit line is overwritten by selecting the appropriate control signals.

C. Watermark Generation Unit

Most of the invisible watermarking algorithms described in the current literature and the algorithm discussed in this paper insert pseudorandom numbers to host data. Therefore, we focused on the structural design aspects of watermarking generators using LFSRs. The ternary watermark is generated by a pseudorandom sequence generator. The watermark generation unit consists of an LFSR. The LFSR has a multitude of uses in digital system design and is a very crucial unit in watermark security and detection. It is a sequential shift register with combinational feedback logic around it that causes it to cycle pseudorandomly through a sequence of binary values. We have

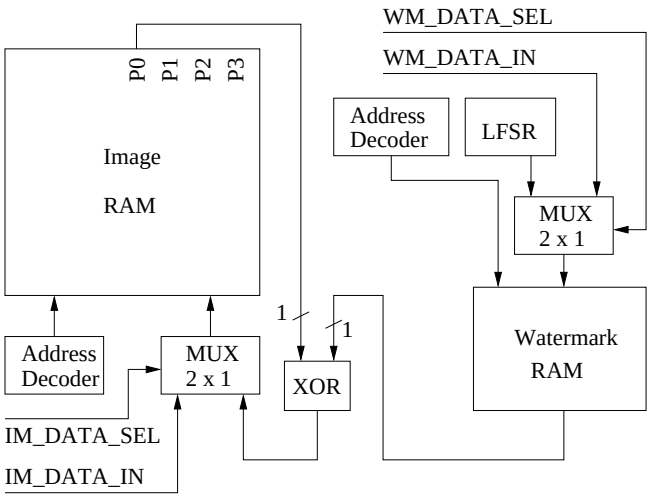


Fig. 4. Datapath for Invisible Fragile Watermarking

studied the challenges of an LFSR and have taken appropriate measures to ensure quality design [33], [34], [35]. The LFSR consists of flip-flops (FFs) as sequential elements with feedback loops. The feedback around an LFSR comes from a selected set of points called taps in the FF chain; these taps are fed back to the FFs after either XORing or XNORing.

The design aspects considered when modeling LFSRs are as follows [33], [34], [35].

- *Using XOR or XNOR feedback gates*: The feedback path may consist of either all XOR gates or all XNOR gates. They are interchangeable, and given particular tap settings, the LFSR will sequence through the same number of values in a loop before the loop repeats itself; the only difference is that the sequence will be different.
- *Choosing one-to-many or many-to-one feedback structure*: Both one-to-many or many-to-one feedback structures using XOR or XNOR gates can be implemented and use the same number of logic gates. A one-to-many structure will always have a shorter worst-case clock-to-clock path delay because it passes only through a single two-input XOR (XNOR) gate instead of a tree of XOR (XNOR) gates in the case of the many-to-one structure.
- *Avoiding prohibited or lockup state*: Using XOR gates, the LFSR will not sequence through the binary value where all bits are at logic zero. Should it find itself with all bits at logic zero, it will continue to shift all zeros indefinitely. Therefore, the LFSR should be prohibited from randomly initializing to all logic zeros during powerup. Similarly, the XNOR-based LFSR will not sequence through the binary values where all bits are at logic one and should be prohibited from randomly initializing to all ones during powerup. This random initialization can be overcome by using the following methodology: (1) using a reset to either preset or clear the individual register FFs to a known good value (in this case, the value is hardwired and cannot be changed; (2) providing a means of loading an initial seed value into the register, either in parallel or serially, and (3)

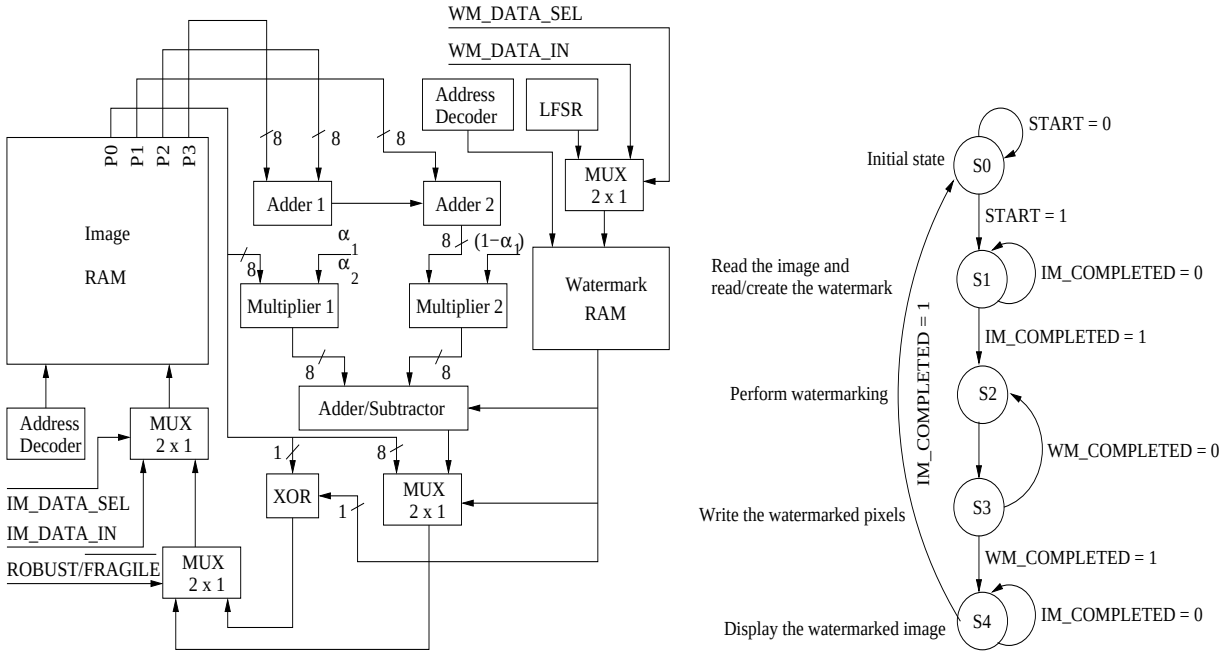
modeling extra circuitry that allows all 2^n values to be included in the sequence.

- *Ensuring a sequence of all 2^n values*: If taps provided for a maximal length sequence are used, the LFSR configurations described so far will sequence through (2^n-1) binary values. The feedback path can be modified with extra circuitry to ensure that all $2n$ binary values are included in the sequence.

An 8-bit LFSR is modeled so as to use a one-to-many feedback structure and has been modified for a 2^n looping sequence. It calculates and holds the next value of the LFSR, which is then assigned to the output signal WM_DATA after each clock edge. The NOR of all LFSR bits minus the most significant bit that is LFSR_REG (6:0) generates the extra circuitry needed for all 2^n sequence values.

D. Overall Architecture of the Proposed Watermarking Chip

The combined datapath for both robust and fragile watermarking is shown in Fig. 5(a). The data path is obtained by stitching the two datapaths from (Fig. 3 and Fig. 4) using multiplexers, which in turn give rise to additional control signals. The controller that drives the datapath is shown in Fig. 5(b). The control signals and their descriptions are given in Table III.



(a) Datapath (b) Controller

Fig. 5. Datapath and Controller for the Architecture of the Chip that Can Perform Both Invisible Robust and Invisible Fragile Watermarking in the Spatial Domain

VI. FPGA-Based Implementation of the Proposed Watermarking Chip

In this section, we describe FPGA-based prototyping of the architectures proposed in Section V. We followed a top-down modular design flow in performing the architectural design and simulation and FPGA prototyping as

TABLE III

CONTROL SIGNALS FOR SPATIAL DOMAIN INVISIBLE WATERMARKING CHIP

IM_ADDR_COUNT	Increment signal for the counters used to generate the address for the image
WM_ADDR_COUNT	Increment signal for the counters used to generate the address for the watermark
IM_$\overline{\text{READ}}$/WRITE	Image RAM read (1) or write (0)
WM_$\overline{\text{READ}}$/WRITE	Watermark RAM read or write
IM_DATA_SELECT	Select input or watermarked image
WM_DATA_SELECT	Select input or generate a watermark
IM_ADDR_SELECT	Select the location of the image
WM_ADDR_SELECT	Select the address of the watermark
START	Watermarking begins when set to 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 process continues
DATA_READY	High when the watermarked image is ready to be read
ROBUST/$\overline{\text{FRAGILE}}$	Choose between robust and fragile

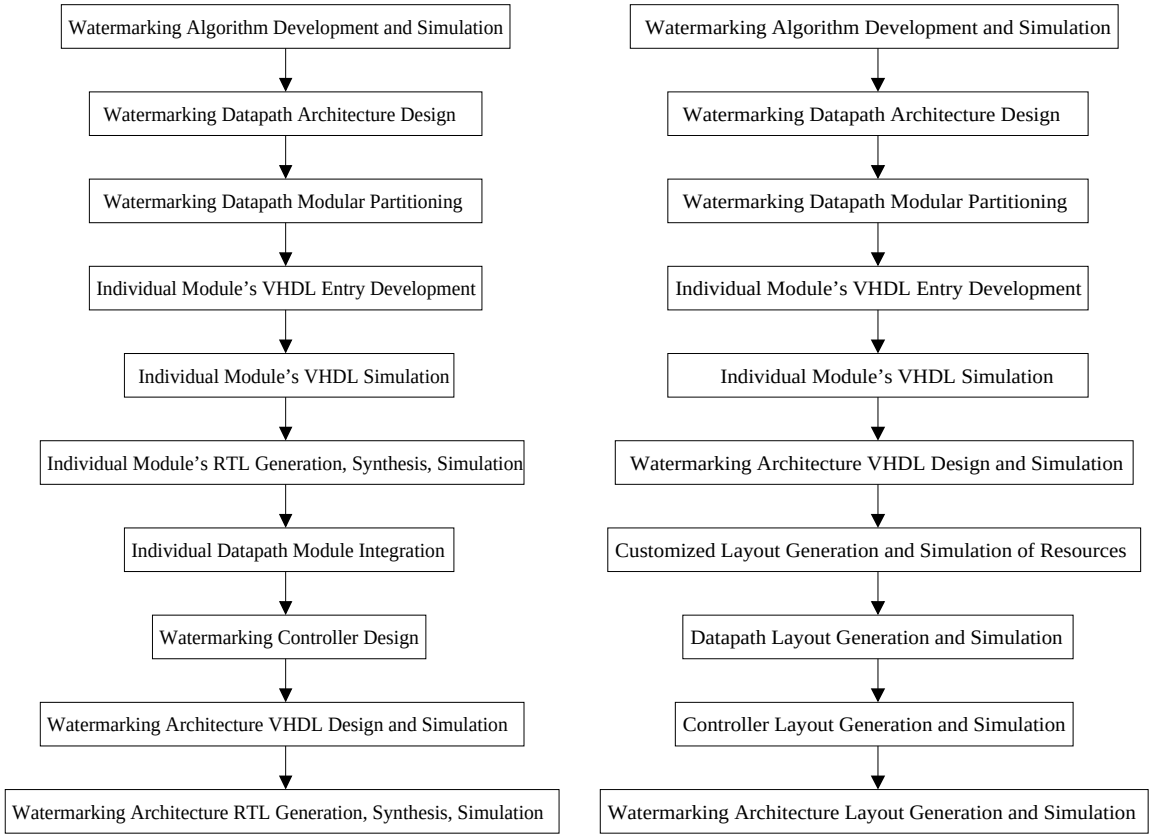
presented in Fig. 6(a). We logically and structurally divided an architectural unit into several modules first. Then we individually tested and verified these modules through simulation and synthesis of the VHSIC hardware description language (VHDL) and the register transfer language (RTL). Once the individual modules were tested and verified to be functionally correct, they were stitched together. Next, a controller design executed the datapath and ensured that the unit performed its operations.

TABLE IV

SUMMARY OF FPGA-BASED SYNTHESIS REPORT

Architectural Units	Period/Delay (ns)	Cell Usage
Watermark Insertion	15.526	122
Watermark Generation	4.916	43
Watermarking Encoder	19.842	838

Following the above approach, we divided the encoder units into two logical modules, the watermark insertion and the watermark generator. When the two modules were stitched together with a controller, they provided us with the complete encoder unit. All the modeling and design were performed with VHDL. The synthesis of the chip was performed with the Synplify Pro^{TM} tool targeting Xilinx VIRTEX-II technology with XCV50-BG256-6 target device. The simulations were performed with ModelSim. Individual RTLs were obtained after synthesizing the VHDL design for the watermarking insertion and watermark generation modules, respectively. An RTL schematic of the encoder synthesized from the VHDL when the watermarking insertion unit, watermarking generation unit, address decoders, temporary registers, and controllers were included was also obtained. The timing simulation was



(a) Approach for FPGA-Based Design (b) Approach Followed for Custom IC Design

Fig. 6. Design Flow Followed for the FPGA-Based Prototyping and Custom IC Design

performed to verify the functional correctness of the design. However, the RTL diagram or the simulation waveforms are not included here for brevity. During the simulation process, it was assumed that the image data was being read from, and stored to, the hard drive.

From the synthesis results, we derived the macro statistics and timing report of the units in Table IV. Minimum period is an indicator of the timing path from a clock to any other clock in the design. The minimum period is reported for both the generation unit and the encoder, whereas the critical path delay is reported for the insertion unit, which is fully combinational. The cell usage indicates all the logical cells that are basic elements of the technology.

VII. Custom IC Implementation of the Proposed Watermarking Chip

In this section, we describe a custom IC design of the proposed architecture that can perform both invisible robust and invisible fragile watermarking. As demonstrated in Fig. 6(b), we followed a modular design approach to generate the layout of the complete chip in which the logic design is top-down and the physical design is bottom-up. Our motivation was to use a custom layout to build the whole chip using minimal silicon. Following a hierarchical approach, we created the layout of various resources, such as adders, multipliers, etc. This approach was followed

by hierarchical creation of the layout for the insertion and generation units, encoder, etc. Finally, once the complete chip layout was generated, parasitic extraction and power, area, and performance analysis were performed on the post-layout (including parasitics) design.

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

The memory module involved two read/write memory structures, one for the original watermarked image (with a size of 256×256) and another for the watermark with a size of 128×128. The bit size for the image RAM was 8 bits and 2 bits for the watermark RAM. The basic building block for a memory module is a $6--$transistor static RAM cell available in the cell library. We chose static RAM instead of dynamic RAM because of its shorter read and write cycles. The memories were built as $n \times n$ arrays of static RAM cells and were addressed using row and column address decoders. Each decoder is implemented as an m-bit counter with additional AND logic to address 2^m cells.

The main component of the watermarking module (datapath), the insertion unit, consists of two 8-bit adders, two 8-bit multipliers, and an 8-bit adder/subtractor, each of which was built with standard structures. The carry inputs to the adder/subtractor and one of the inputs to the XOR gate were set to high whenever the watermark pixel value was 2 so that a subtraction was carried out as required for the robust watermarking encoding function (Eqn. 2).

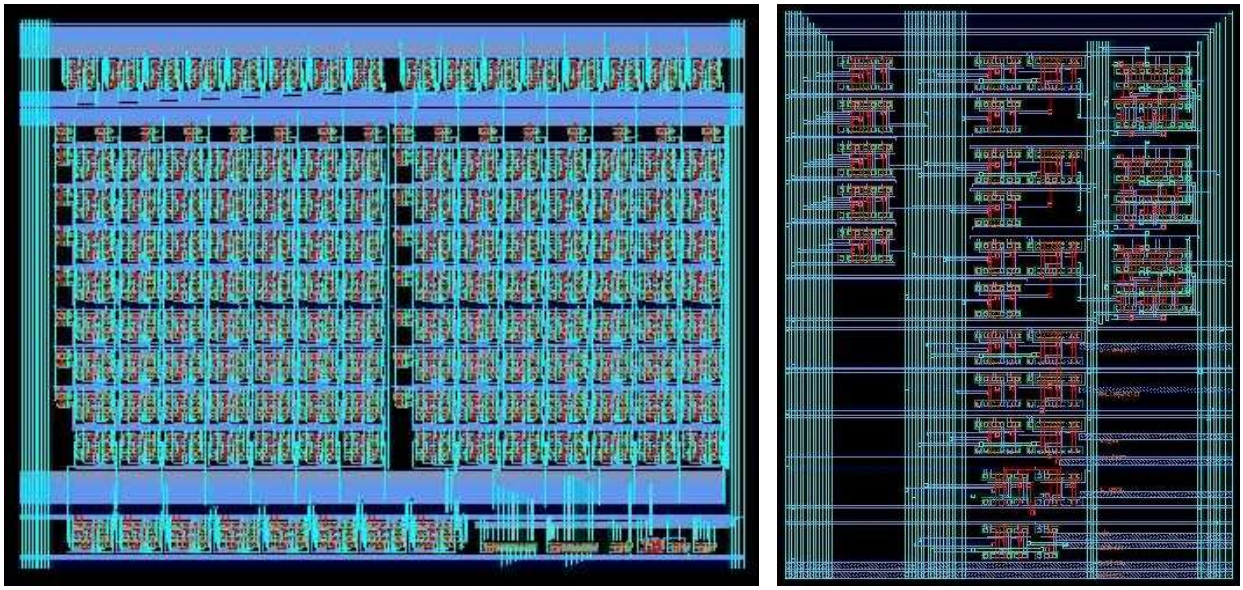
Several multiplexers were used at appropriate places in the design to select one of the incoming lines. Each of these multiplexers was implemented by using a combination of transmission gates. Three asynchronously resettable registers were designed to encode the five states of the controller depicted in Fig. 5(b). At any given time, the three registers could be reset by the user to return the controller to its initial state; and from there, the watermarking function could be started afresh.

TABLE V

POWER, AREA DETAILS FOR INDIVIDUAL UNITS FOR CUSTOM IC IMPLEMENTATION

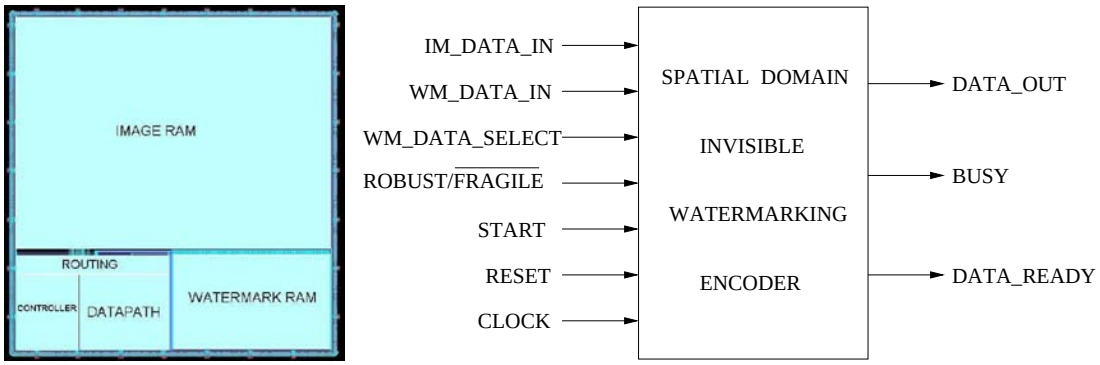
Modules	Gate Count	Power (mW)	Delay (ns)
Datapath	4547	1.1931	0.9158
Controller	233	0.0045	0.3901
RAM	1183,744	21.8012	2.3891

Each of the above-mentioned modules was implemented and tested separately and then connected to obtain the final chip. The gate count, power, and delay time of each module are listed in Table V for the $0.35\mu m$ SCN3M SCMOS technology from the Metal Oxide Semiconductor Implementation Service (MOSIS) and an operating



(a) Datapath Layout (b) Controller Layout

Fig. 7. Layout of the Datapath and the Controller of the Proposed Invisible Watermarking Chip



(a) Floorplan (b) Pin Diagram

Fig. 8. Floorplan and Pin Diagram for the Proposed Invisible Watermarking Chip

TABLE VI

OVERALL DESIGN STATISTICS OF THE WATERMARKING CUSTOM IC CHIP

Area (with RAM)	$15.012 \times 14.225 mm^2$
Number of gates (with RAM)	$1188K$
Number of gates (without RAM)	4820
Clock frequency (with RAM)	$151MHz$
Clock frequency (without RAM)	$545MHz$
Number of input/output pins	25
Power (with RAM)	$24mW$
Power (without RAM)	$2.0547mW$

voltage of $3.3V$. Functional verification of each block and generation of the data in Table V were performed by using transistor-level simulations in HSPICE. Full-chip functionality was verified with NanoSim. It is evident from the above statistics that the RAM consumed the most amount of power. If we assume that the proposed chip is to be used as a module within a complete JPEG encoder, then the memory module could be avoided in the watermarking datapath circuit. The layouts of the datapath and the controller are shown in Fig. 7(a) and Fig. 7(b), respectively.

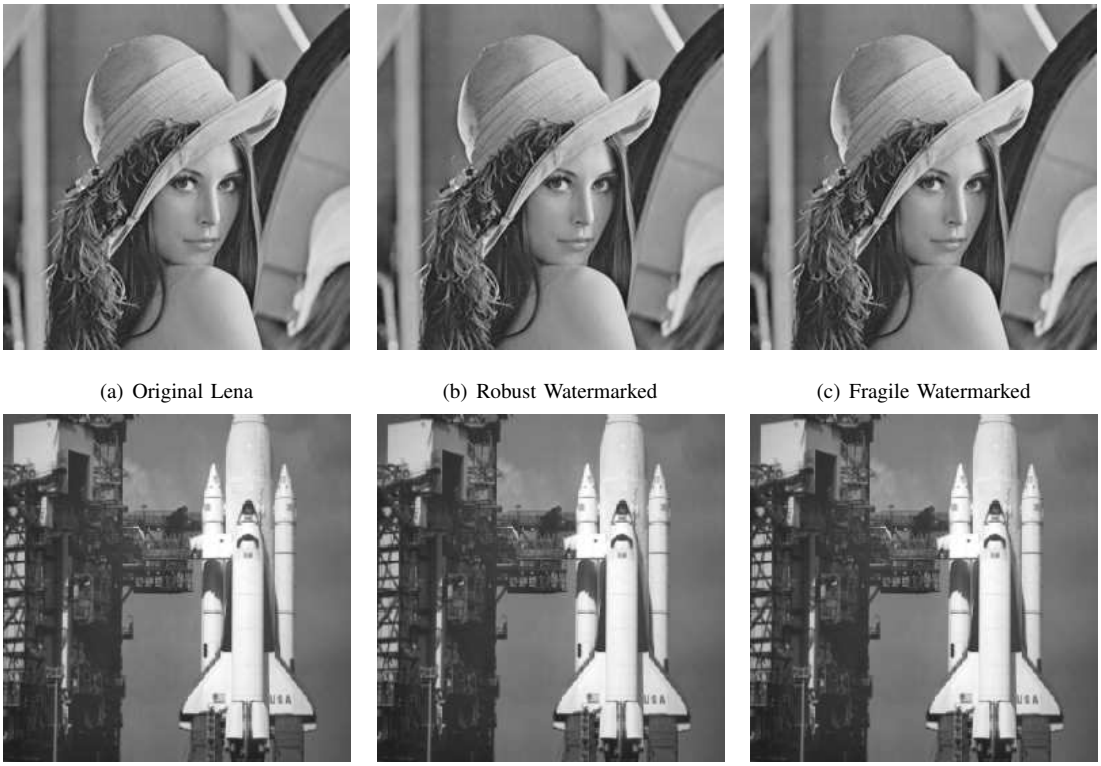
The complete layout and floorplan of the watermarking chip is given in Fig. 8(a). The pin diagram for the chip showing the inputs and outputs is given in Fig. 8(b). The choice between invisible robust watermarking and invisible fragile watermarking is made with the help of a robust/$\overline{\text{fragile}}$ line. When the robust/$\overline{\text{fragile}}$ line is high, it represents invisible robust watermarking; and when the robust/$\overline{\text{fragile}}$ line is low, it represents invisible fragile watermarking. During invisible fragile watermarking, the invisible robust watermarking part of the chip is disabled and vice versa. The invisible robust insertion datapath and the invisible fragile datapath share the same output pins. The overall design statistics of the prototype chip are provided in Table VI.

VIII. Experimental Results

Each individual unit in the chip was tested individually with HSPICE, and the complete encoder was simulated with NanoSim. The model file was obtained from MOSIS. A typical simulation time of $10,000ns$ was used. The simulation time depended on the module being tested. Random vectors were used for testing. The functionality and the delay of each module were verified with the help of simulation waveforms.

We carried out extensive simulations with various test images. The test images of size 256×256 were borrowed from [9], [16], [17] for the simulations. Selected examples of images are shown in Fig. 9. Visual inspection of the images illustrates the quality of the watermarking and demonstrates that the results matched with the software approach. As a quantitative measure of the perceptibility of the watermark, we again calculated the PSNR. We then compared the PSNR of the watermarked images obtained by using the proposed chip with that of the watermarked images obtained by using the software algorithms. The PSNR in both the hardware and the software algorithms was found to be approximately the same.

A broad perspective of various existing VLSI chips for watermarking is provided in Table VII along with the prototype chip described in this paper. The chip statistics provide data of technology, area, supply voltage, operating frequency, and power consumption, respectively. Our proposed chip performs the operations of both Tsai and Lu [25] and Garimella et al. [26], which perform invisible robust and invisible fragile watermarking, respectively. But, the power consumption is much lower than that of [25]. At the same time, our chip can operate at a frequency as high as $0.5GHz$, which is higher than any of the existing watermarking chips. It may be noted that it is not possible to provide a fair comparison because the different chips work in different domains and have different capabilities. It is a fact that DCT domain computations are more resource-intensive than spatial domain computations. However, the table is provided here not for comparison but as a guideline for the readers about similar architectures and chips performing watermarking.



(a) Original Lena (b) Robust Watermarked (c) Fragile Watermarked

(d) Original Shuttle (e) Robust Watermarked (f) Fragile Watermarked

Fig. 9. Spatial Domain Invisible Watermarked Test Images

TABLE VII

SUMMARY OF WATERMARKING CUSTOM IC HARDWARE DESCRIBED IN THE CURRENT LITERATURE FOR IMAGE WATERMARKING

Research Works	Watermarking Type	Processing Domain	Chip Statistics
Tsai and Lu [25]	Invisible robust	DCT	$0.35\mu m$, $3.064 \times 3.064 mm^2$, $3.3V$, $50MHz$, $62.78mW$
Garimella et al. [26]	Invisible fragile	spatial	$0.13\mu m$, $3.453 \times 3.453 mm^2$ $1.2V$, $100MHz$, $37.6\mu W$
Mohanty et al. [22]	Visible	spatial	$0.35\mu m$, $3.34 \times 2.89 mm^2$ $3.3V$, $292MHz$, $6.93mW$
Mohanty et al. [36]	Invisible robust Visible	DCT	$0.25\mu m$, 1.5 and $2.5V$ 70 and $280MHz$, $0.3mW$
This work	Invisible robust Invisible fragile	spatial	$0.35\mu m$, $3.3V$, $15.012 \times 14.225 mm^2$ $545MHz$, $2.0547mW$

IX. Conclusions

In this paper, we presented a watermarking encoder that can perform invisible robust, invisible fragile watermarking, and a combination of both in the spatial domain. To the best of our knowledge, this is the first watermarking architecture having both functionalities. The chip can be easily integrated in any existing JPEG encoder to watermark images right at the source end. The implementation of a low-power, high-performance version is currently in progress. Low-power VLSI features, such as multiple supply voltages, dynamic clocking, and clock gating will be considered. High-performance architectural implementations, such as pipeline or parallelism, are under investigation. The disadvantage of the watermarking algorithms implemented is that the processing needs to be performed pixel by pixel. In the future, we plan to investigate block-by-block processing. A low-power, high-performance watermarking decoder is also in the implementation stage. We are also planning to modify the architectures to handle color images. Because DRM systems need both encryption and watermarking, we believe that combining both the hardware and the data compression architectures will be necessary. Moreover, the on-chip encrypter can be used in storing the watermarking generator key in encrypted form, thus enhancing watermark security.

X. Acknowledgments

The authors would like to thank Ravi Namballa (graduate of the University of South Florida), C. Renuka Kumara (graduate of the Manipal Centre for Information Science, India), and Shridhar Nayak (faculty member at the Manipal Centre for Information Science, India) for their help.

References

[1] S. Emmanuel and M. S. Kankanhalli, "A Digital Rights Management Scheme for Broadcast Video," *ACM-Springer Verlag Multimedia Systems Journal*, vol. 8, no. 6, pp. 444–458, June 2003.

[2] D. Kundur and K. Karthik, "Digital Fingerprinting and Encryption Principles for Digital Rights Management," *IEEE Signal Processing*, vol. 52, 2004.

[3] A. M. Eskicioglu and E. J. Delp, "An Overview of Multimedia Content Protection in Consumer Electronics Devices," *Elsevier Signal Processing : Image Communication*, vol. 16, pp. 681–699, 2001.

[4] N. M. Kosaraju, M. Varanasi, and S. P. Mohanty, "A High Performance VLSI Architecture for Advanced Encryption Standard (AES) Algorithm," in *Proceedings of 19th IEEE International Conference on VLSI Design*, 2006, pp. 481–484.

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

[6] S. P. Mohanty, "Digital Watamerking of Images," M.S. thesis, Department of Electrical Engineering, Indian Institute of Science, Bangalore, India, 1999.

[7] N. Memon and P. W. Wong, "Protecting Digital Media Content," *Communications of the ACM*, vol. 41, no. 7, pp. 35–43, July 1998.

[8] F. Mintzer, G. Braudaway, and M. Yeung, "Effective and Ineffective Digital Watermarks," in *IEEE International Conference on Image Processing (ICIP-97)*, 1997, vol. 3, pp. 9–12.

[9] 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.

[10] I. J. Cox, J. Kilian, T. Shamoon, and T. Leighton, "Secure Spread Spectrum Watermarking of Images, Audio and Video," in *Proceedings IEEE International Conf on Image Processing*, 1996, vol. 3, pp. 243–246.

[11] I. J. Cox, J. Kilian, T. Shamoon, and T. Leighton, "A Secure Robust Watermarking for Multimedia," in *Proceedings of First International Workshop on Information Hiding*, 1996, vol. 1174, pp. 185–206.

[12] I. J. Cox, J. Kilian, 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.

[13] H. Berghel, "Watermarking Cyberspace," *Communications of the ACM*, vol. 40, no. 11, pp. 19–24, November 1997.

[14] M. Swanson, M. Kobayashi, and A. Tewfik, "Multimedia Data Embedding and Watermarking Technologies," *Proceedings of the IEEE*, vol. 86, no. 6, pp. 1064–1087, June 1998.

[15] S. Craver, N. Memon, B. L. Yeo, and M. M. Yeung, "Resolving Rightful Ownerships with Invisible Watermarking Techniques: Limitations, Attacks and Implications," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 4, pp. 573–586, May 1998.

[16] 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.

[17] 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.

[18] C. T. Li, "Digital fragile watermarking scheme for authentication of JPEG images," *IEE Proceedings - Vision, Image and Signal Processing*, vol. 151, no. 6, pp. 460–466, Dec 2004.

[19] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Information Hiding - A Survey," *Proceedings of the IEEE*, vol. 87, no. 7, pp. 1062–1078, July 1999.

[20] S. Voloshynovskiy, S. Pereira, T. Pun, J. J. Eggers, and J. K. Su, "Attacks on Digital Watermarks: Classification, Estimation-based Attacks and Benchmarks," *IEEE Communications Magazine*, vol. 39, no. 9, pp. 118–126, August 2001.

[21] 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*, vol. 17, no. 5, pp. 47–57, Sep 2000.

[22] 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 (TVLSI)*, vol. 13, no. 8, pp. 1002–1012, August 2005.

[23] 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.

[24] 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.

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

[26] 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. 283–288.

[27] 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*, 2005, pp. 399–403.

[28] F. J. MacWilliam and N. J. A. Sloane, "Pseudorandom Sequences and Arrays," *Proceedings of the IEEE*, vol. 64, no. 12, pp. 1715–1729, Dec 1976.

[29] D. V. Sarwate and M. B. Pursley, "Cross-correlation of Pseudorandom and Related Sequences," *Proceedings of the IEEE*, vol. 68, no. 5, pp. 593–619, May 1980.

[30] R. G. Wolfgang and E. J. Delp, "A watermark for digital images," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 1996, vol. 3, pp. 219–222.

[31] 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.

[32] F. Bartolini, A. Tefas, M. Barni, and I. Pitas, "Image Authentication Techniques for Surveillance Applications," *Proceedings of the IEEE*, vol. 89, no. 10, pp. 1403–1418, Oct 2001.

[33] 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.

[34] M. J. S. Smith, *Application-Specific Integrated Circuits*, Addison-Wesley Publishing Company, MA 01867, USA, 1997.

[35] D. J. Smith, *HDL Chip Design: A Practical Guide for Designing, Synthesizing and Simulating ASICs and FPGAs Using VHDL or Verilog*, Doone Publications, USA, 1998.

[36] S. P. Mohanty, N. Ranganathan, and K. Balakrishnan, "A Dual Voltage-Frequency VLSI Chip for Image Watermarking in DCT Domain," *IEEE Transactions on Circuits and Systems II (TCAS-II)*, vol. 53, no. 5, pp. 394–398, May 2006.