AN EFFECTIVE HIGH PERFORMANCE APPROACH OF REVERSIBLE IMAGE WATERMARKING USING MOUSETRAP PIPELINING

R.G.Krishna Bharathi
Final Year Student, Department of Computer Science and Engineering,
Raja College of Engineering and Technology, Madurai, Tamilnadu, India

E.Gopinath
Assistant Professor, Department of Computer Science and Engineering,
Raja College of Engineering and Technology, Madurai, Tamilnadu, India

S.Charulatha
Final Year Student, Department of Computer Science and Engineering,
Raja College of Engineering and Technology, Madurai, Tamilnadu, India

Abstract

Digital watermarking is act of hiding data within a digital image in order to resolve issues related to authentication and copyrights. Watermarking is suitable in sensitive applications like law enforcement, medical and military images without any distortions. The watermarking technique satisfying those requirements referred to 'Reversible Watermarking'. Reversible watermarking is designed so that it can be removed completely restore the original image. Considering the age of reversible watermarking which decade to count, it has fetched enormous attention of researches to boost of. In traditional watermarking techniques, main concern is to embed and recover watermark with minimum loss.

Index Terms: *Asynchronous, Clock–less, low power, Reversible watermarking, Reconfiguration*

1. Introduction

Digital watermarking is one way of protecting images, where secret information called watermark is embedded in digital image (or video or audio). This information can later be extracted to authenticate ownership. Literature highlights that there are plenty of software implementations of watermarking schemes defined in both spatial and transform domains. While transform domain techniques are robust, spatial domain methods are less complex.

The main drawback of software implementation of watermarking schemes is that there is very little flexibility and means to improve the speed and to reduce area. A hardware implementation provides a flexibility of optimizing the area, speed and power metrics of the watermarking system, as all the algorithms are implemented using full-custom circuitry. While there are plenty of software algorithms available for RW, the number of hardware implementations is limited. Reversible algorithm can be subdivided into two main categories, namely *fragile* and *semi-fragile*. Most of the developed techniques belong to the family of fragile that means that the inserted watermark disappears when a modification has occurred to the watermarked image, thus revealing that data integrity has been compromised. An inferior number, in percentage, are grouped in the second category of semi-fragile where with term it is intended that the watermark is able to survive to a

possible unintentional process the image may undergo, as for instance, a slight JPEG compression. *Spatial Domain.* This subsection is dedicated to present some of the main works implementing fragile reversible watermarking by operating in the spatial domain. One of the most important works in such a field has been presented by Tian [4, 5]. It presents a high-capacity, high visual quality, and reversible data embedding method for grayscale digital images. This method calculates the difference of neighboring pixel values and then selects some of such differences to perform a difference expansion (DE). *Semi-fragile Algorithms*: De Vleeschouwer et al. proposed in [20], a semi-fragile algorithm based on the identification of a robust feature of the luminance histogram for an image tile.

Literature highlights that there are plenty of software implementations of watermarking schemes defined in both spatial and transform domains. While transform domain techniques are robust, spatial domain methods are less complex. Coltucetal. [2] proposed a spatial domain RW technique based on integer transform and reversible contrast mapping. This scheme provides high data bit- rate and lower complexity. J Tian [3] proposed a difference expansion based method which doesn't need compression to embed the watermark. Wien Hong [4] proposed a modified histogram shifting method to provide a good image quality and low computational cost. Rajendraetal. [5] proposed a modified histogram shifting technique to enhance the capacity and quality.

For most of the above mentioned algorithms, estimation error may be quite high if the pixels are not in a uniform region. In order to address this issue, Dragoietal. [6] presented an adaptive interpolation scheme. The interpolated values are calculated in one of the two ways – Average of four neighboring pixels and average of horizontal or vertical pairs of pixels. To achieve minimum distortion, data is hidden in pixels with estimation error 'e' less than threshold *T*. If $e > T$, the pixels are grouped as "not embedded".

2. Existing Architecture

Power dissipation and clock distribution issues are becoming major constraints in IC's with smaller dimensions and higher operating speeds. The clock is a major component of any hardware design. But due to switching at every clock edge, a lot of power is dissipated in synchronous or clocked designs. Asynchronous designing helps to overcome this issue by completely eliminating clock from the design. The blocks switch only when necessary and not at every edge of clock pulse. Thus the clock elimination can make it simpler to design an IC and also reduce the area requirements. Thus asynchronous design style is more efficient than its synchronous counterpart. These asynchronous designs are Delay – Insensitive (DI) [13]. The advantages in such designs include low power, ease of integration with multi – speed circuits and reduced clock distribution problems.

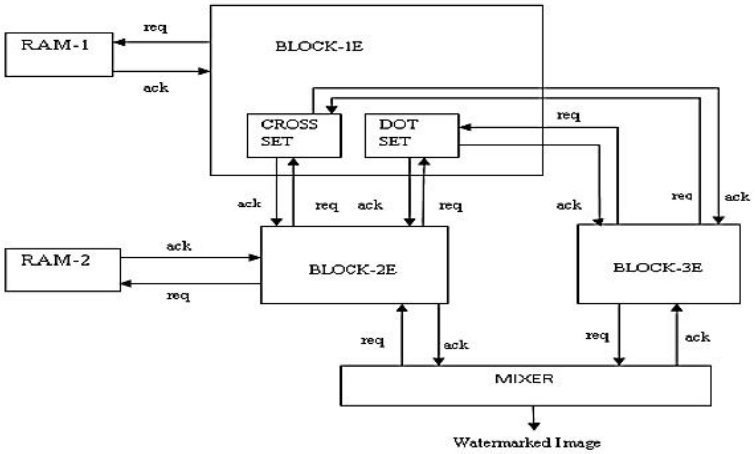
Pipeline [14]. The pipeline uses a 2 I/P XNOR gate and a level sensitive latch. These latches are transparent before a set of data arrives and they become opaque after that data has arrived. Fig. 4 shows a basic mousetrap pipeline implementation

Initially all the *req* and *ack* are in the same state. The XNOR gate acts as a Latch Controller. For the N^{th} stage, when req_N is high, the latch controller output is *'0'* and the latch becomes opaque. This

indicates a data in th e N^{th} stage. Once ack_N becomes high the latch becomes transparent again.

Figure below shows the VLSI Architecture for the watermark encoder for the proposed scheme. It consists of two blocks: Block

1E for sorting the pixels to *cross set* and *dot set* regions. Block 2E to embed watermark and block 3E to modify the "*not embedded*" pixels.

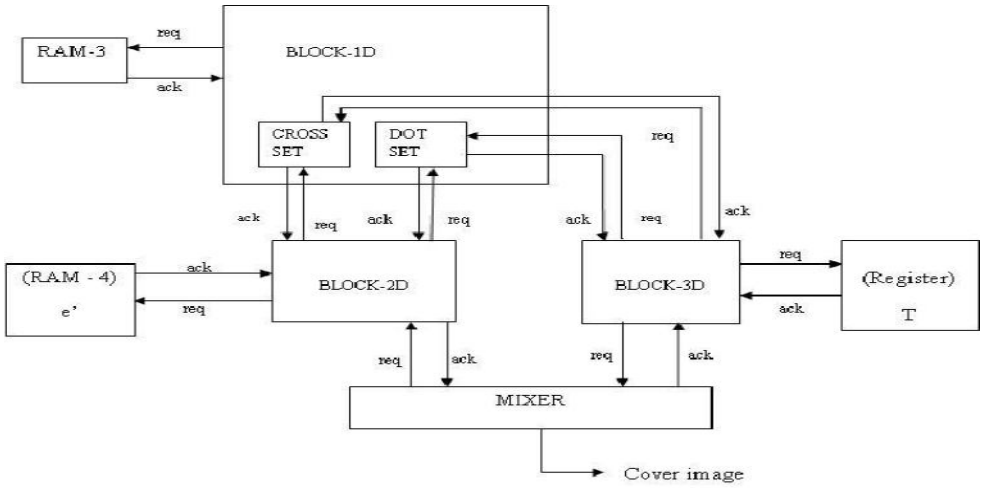


Watermarking Encode r Top – Level Implementation

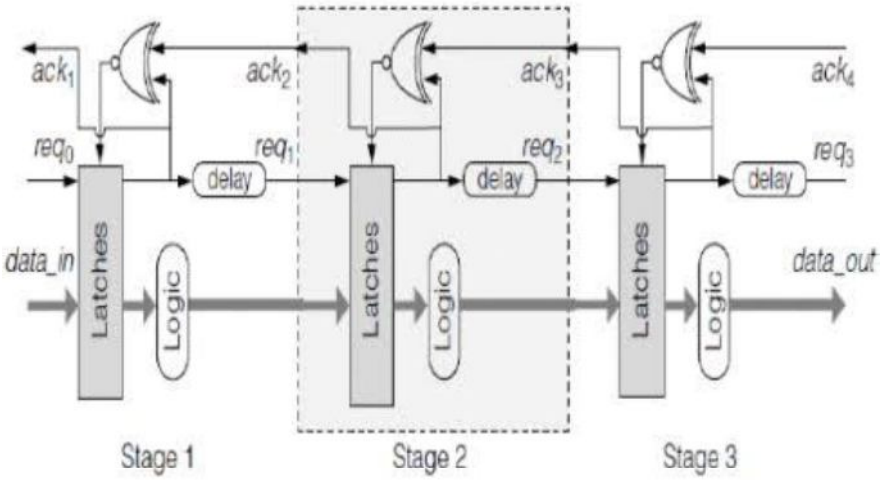
RAM - 1 stores the cover image. RAM- 2 stores the watermark. The architecture for decoder is as shown in Fig. 6 and is very similar to that of the encoder. Block 1D operation is similar to that of Block 1E. Block 2D restores the pixel values using 'e' to separate the embedded bit. Block 3D restores pixel values using *T*. Each module communicates through handshaking signals i.e. *req* and *ack* as shown in Fig. 3. The RAMs used are asynchronous memories. The cover image used is of 512 X 512 pixels and watermark is also of the same size.

Watermark Embedding

Watermark bit '*b*' is embedded into the pixel *i*



Watermarking Decoder Top – Level Implementation



Mouse Trap Pipelining

The steps for encoding are as follows:

1. The image is grouped as two parts - *S* (selected) region and *R* (reserved) region. *S* consists of the pixels chosen for embedding and *R* is used to send information needed at the decoding stage in the form of threshold values and flag bits.
2. To reduce large distortions, watermark is embedded in pixels with estimation error $e < T$, where *T* is the threshold.
3. Starting with $T = 1$, the embedding process is simulated. If the number of embeddable pixels doesn't suffice, *T* is incremented. This continues till the watermark is completely embedded.
4. The pixels with $e \_ T$ form the "*not embedded group*". These pixels are either modified or retained without change so that they provide higher error in comparison to the embedded pixels.
5. The pixels of the *S* region are further grouped as two sets – "the *cross set*" and "the *dot set*".
6. Starting from first row, all the pixels of even numbered columns are assigned to the *cross set* and the remaining pixels are assigned to the *dot set*. For second row, the pixels of even columns are put into the *dot set*, and those of odd columns are put into *cross set*. The distribution of pixels is thus alternated in the same manner.
7. While embedding the watermark bits, the *cross set* pixels are first checked if they are embeddable or not, followed by *dot set* pixels.

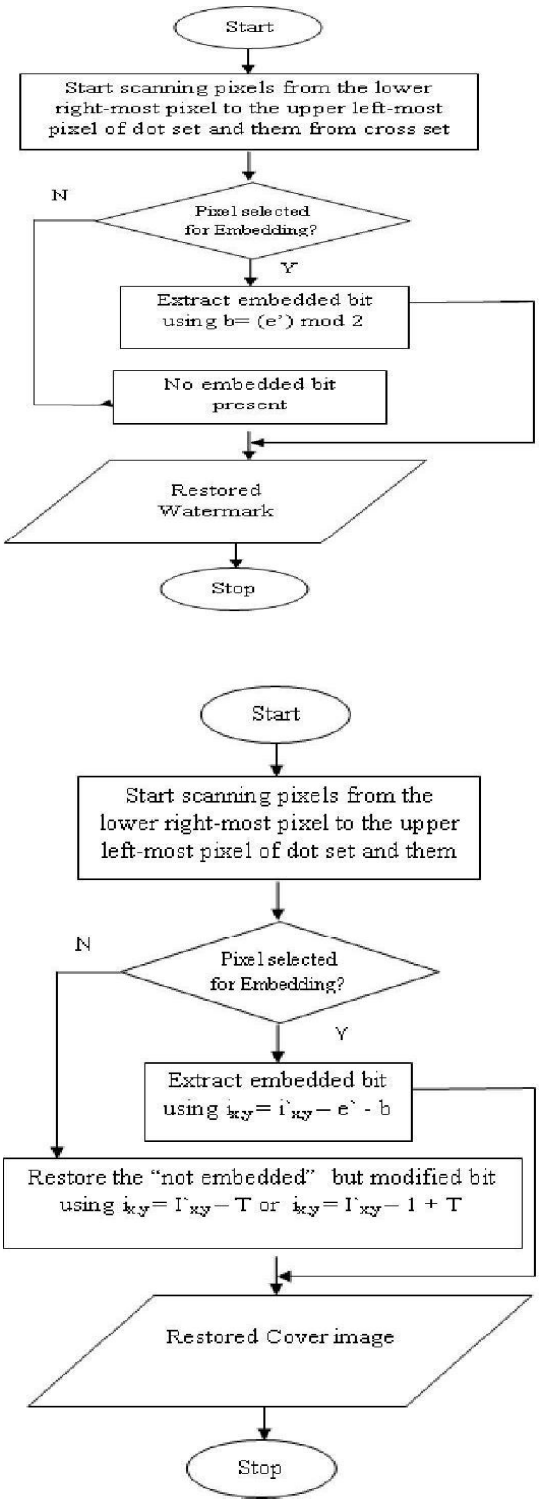
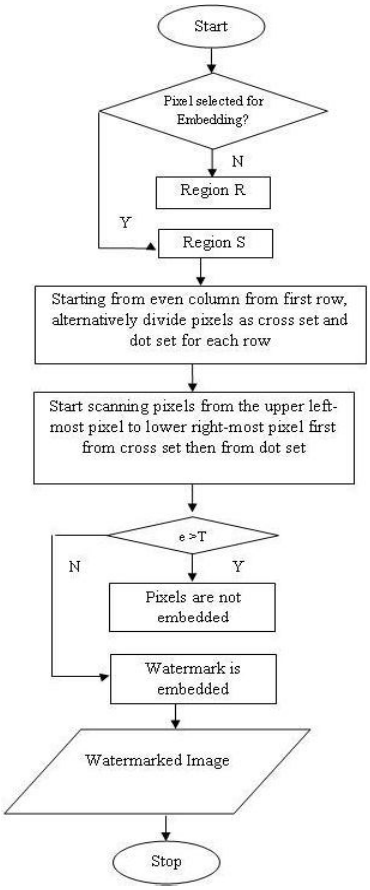


Fig2 - Watermark Extraction

B. Watermark Extraction

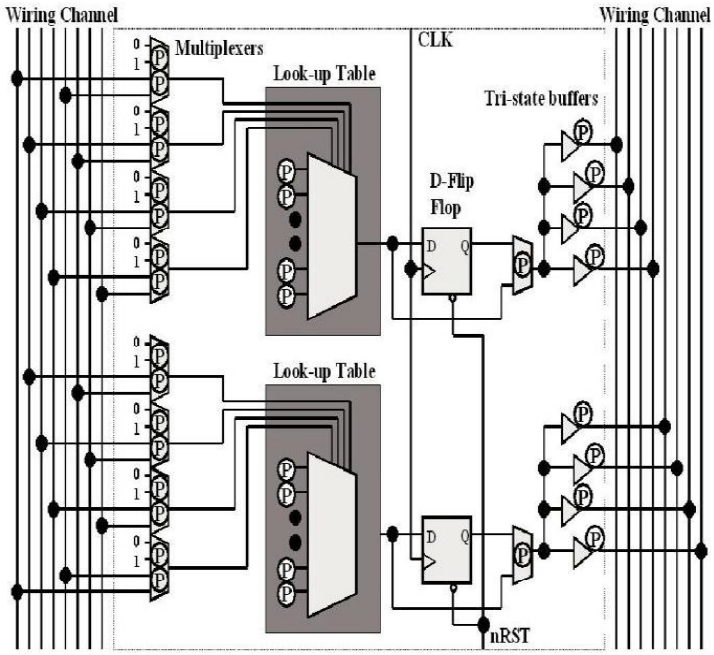
The extraction of watermark follows exact complement process of embedding. In embedding, the pixels are operated from top left to bottom right pixel whereas in extraction, the process is reverse. Depending on the values of error *e* and the pixel value, the

Since the encoding started from the *cross set*, decoding starts from *dot set*. Since the decoding is done in a direction opposite to encoding, in order to recover the embedded watermark, the bit stream has to be reversed.



III. Proposed Architecture

A block diagram of an optically reconfigurable logic block of the DORGA-VLSI chip is presented in Fig. 5. Each optically reconfigurable logic block consists of 2 four-input one-output look-up tables (LUTs), 10 multiplexers, 8 tri-state buffers, and 2 delay-type flip-flops with a reset function. The input signals from the wiring channel, which are applied through some switching matrices and wiring channels from optically reconfigurable I/O blocks, are transferred to LUTs through eight multiplexers. The LUTs are used for implementing Boolean functions. The outputs of an LUT and of a delay-type flip-flop connected to the LUT are connected to a multiplexer. A combinational circuit and sequential circuit can be chosen by changing the multiplexer, as in FPGAs. Finally, outputs of the multiplexers are connected to the wiring channel again through eight tri-state buffers. As a result, each four-input one-output LUT, multiplexer, and tri-state buffer has 16 photodiodes, 2 photodiodes, and 1 photodiode, respectively. In all, 58 photodiodes are used for programming an optically reconfigurable logic block. The optically reconfigurable logic block can be reconfigured perfectly in parallel. The CAD layout is depicted in Fig. 6. This is a standard-cell based design



Block Diagram of an Optically Reconfigurable Logic Block (ORLB)

The cell size is 294.0 × 186.5 μm^2. Wiring between cells was executed using the first to the third metal layers while avoiding the aperture area of the photodiode cell. Such optically reconfigurable logic block design is based on a standard cell design, except for custom designs of transmission gate cells and photodiode cells. Each photodiode is arranged at 42.0 μm horizontal intervals and at 12.0-21.0 μm vertical intervals.

IV. Result and Discussions

To implement them, clock-by-clock dynamically reconfigurable devices are desired. However, using current VLSI technologies, simultaneous realization of fast reconfiguration and numerous reconfiguration contexts is impossible. To realize such clock-by-clock dynamically reconfigurable devices, another technology must be developed. As one possibility, this chapter has introduced and described an optically reconfigurable gate array VLSI. Currently, the gate count and performance of such ORGA-VLSIs are insufficient. Nevertheless, such architecture presents the possibility of overcoming current VLSI limitations. Realizing a device to overcome those limitations remains as a subject for future works.

The hardware is implemented using the hardware description language – Verilog. The synthesis is performed using the synthesis tool from Xilinx – Vivado. The SoC used is Zynq – 7000 by Xilinx. The area and power requirements of both synchronous and asynchronous designs are compared in Table I. Comparison is made between the synchronous implementation developed by Sudip et al. [18] and proposed asynchronous scheme along the same lines and tabulated in Table II.

It can be seen that when compared with our synchronous implementation, on an average, the area has increased by about 4%. But this area penalty can be neglected when compared to an

approximate 33 % reduction in overall power consumption. Similarly, comparison of architecture of [18] and proposed asynchronous scheme shows that there is an area increase of 3% and power reduction of 33%.

Table 1: Comparison between Synchronous and Proposed Asynchronous Implementations

Parameter	Synchronous	Asynchronous	% Change
Encoder			
I/O	0.92 %	4.19 %	3.27 % increase
LUT	70.56 %	75.26 %	4.7 % increase
POWER	4.2 watt	1.3 watt	30.95% decrease
Decoder			
I/O	2.98 %	6.03%	3.81 % increase
LUT	80.59 %	84.13%	3.54 % increase
POWER	5.93 watt	2.07 watt	34.91% decrease

Table 2: Comparison between Synchronous Implementation of [18] and Proposed Asynchronous Implementations

Parameter	Synchronous [18]	Asynchronous	% Change
Encoder			
I/O	0.26 %	4.19 %	3.93 % increase
LUT	73.24 %	75.26 %	2.02 % increase
POWER	3.7 watt	1.3 watt	35.13% decrease
Decoder			
I/O	3.26 %	6.03%	2.77 % increase
LUT	82.82%	84.13%	1.31 % increase
POWER	6.7 watt	2.07 watt	30.89% decrease

References

1. B.Surekha, Dr.G.N.Swamy, “A semi-blind image watermarking based on Discrete Wavelet Transform and secret sharing,” IEEE International Conference on Communication, Information and Computing Technology, Sardar Patel Institute of Technology, Mumbai, 2012, pp. 1-5.
2. Coltuc, D., Chassery, J.M., “Very Fast Watermarking by Reversible Contrast Mapping”, IEEE Signal Processing Letters, vol. 14, pp. 255-258, 2007.
3. J. Tian, ”Reversible Data Embedding Using a Difference Expansion”, in IEEE Transactions on Circuits and Systems for Video technology, vol. 13, no. 8, pp. 890-896, 2003.
4. Wien Hong, Tung Shou Chen, Kai Yung Lin and Wen Chin Chiang, “A modified histogram

shifting based reversible data hiding scheme for high quality images", Asian Network for Scientific Information, Information Technology Journal, Vol.9, No.1, pp. 179-183, 2010.

5. Rajendra D. Kanphade and N.S. Narawade,"Forward Modified Histogram Shifting based Reversible Watermarking with Reduced Pixel Shifting and High Embedding Capacity", International Journal of Electrical and Computer Engineering, vol. 5, no. 2, pp. 185-191, 2012. Catalin Dragoi, Dinu Coltuc, "Improved rhombus interpolation for reversible watermarking by difference expansion", IEEE Transactions on EUSIPCO, pp. 1688-1692, 2012.
6. Garimella, A, Satyanarayana, M. V. V., Satish Kumar, R., Murugesh P. S., & Niranjan, U. C., "VLSI implementation of online digital watermarking technique with difference encoding for 8-bit gray scale images," in Proc. 6th International Conference on VLSI Design, 2003, pp. 283-288.
7. Mohanty. S. P., Ranganathan. N., & Balakrishnan. K.,"A dual voltage frequency VLSI chip for image watermarking in DCT domain," IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 53, no. 5, pp. 394-398, 2006
8. Mohanty. S. P., Kougianos. E., & Ranganathan. N., "VLSI architecture and chip for combined invisible robust and fragile watermarking," Computers & Digital Techniques, vol. 1, no. 5, 600-611, 2007.
9. S. Karmani, R. Djemal, R. Tourki, "Efficient hardware architecture of 20 scan based wavelet water-marking for image and video", Computer Standards and Interfaces, vol. 31, no. 4, 801-811, 2013.
10. Lad, T. C., Darji, A D., Merchant, S. N., & Chandorkar, A N., "VLSI Implementation of Wavelet Based Robust Image Watermarking Chip," In Proc. of International Symposium on Electronic System Design, 2011, pp. 56-61.
11. Gourav Thakur, Dr. Veena S. Chakravarthi, "Design and Performance Analysis of 8-bit Asynchronous Pipelined Processor," in VLSI Communication Advanced devices Signals & systems And Networking, July 2013.
12. Steven M. Nowick and Montek Singh, "High-Performance Asynchronous Pipelines: An Overview", IEEE Design & Test of Computers, vol. 28, pp. 8-22, 2011.
13. D. M. Thodi, J. J. Rodriguez, "Prediction-error based reversible watermarking", in Proc. International Conference on Image Processing, Oct. 2004, vol. 3, pp. 1549-1552.
14. Xuan, G.R., Yang, C.Y., Zhen, Y.Z., and Shi, Y.Q., "Reversible data hiding using integer wavelet transform and companding technique", in Proc. International Workshop on Digital-forensics and Watermarking, Seoul, Korea, 2004.pp. 1-11.
15. Darji, A D., Lad, T. c., Merchant, S. N., & Chandorkar, A N., "Watermarking Hardware Based on Wavelet Coefficients Quantization Method", Circuits, Systems and Signal Processing, vol. 32, no. 6, pp. 2559- 2579, 2013.
16. Sudip Ghosh, Nachiketa Das, Subhajit Das, Santi P Maity and Hafizur Rahaman, "FPGA and SoC Based VLSI Architecture of Reversible Watermarking Using Rhombus Interpolation By Difference Expansion", in Proc. Annual IEEE India Conference, 2014, pp. 1-6.