2016 IEEE 6th International Conference on Advanced Computing

Asynchronous Implementation of Reversible Image Watermarking Using Mousetrap Pipelining

Lakshmi H R
Research Scholar, Dept. of ECE
K. S. Institute of Technology
Bangalore, India
hrl.lakshmi@gmail.com

Surekha B
Professor, Dept. of ECE
K. S. Institute of Technology
Bangalore, India
borrasurekha@gmail.com

***Abstract*— Digital watermarking is act of hiding data within a digital image in order to resolve issues related to authentication and copyrights. Reversible watermarking is suitable in sensitive applications like medical images, as it can recover original image after extraction of the watermark. Many reversible watermark schemes have been proposed earlier but there are very few hardware implementations available. Further they are all synchronous clock – based and consume more than 40% of the total power. As the power – efficient designs are in demand, an asynchronous clock – less VLSI architecture for hardware implementation of Dragoi et al. method of reversible image watermarking is proposed in this paper. Results show that, by incorporating asynchronous pipelining, the power consumption is reduced considerably.**

***Index Terms*— Asynchronous, Clock–less, low power, Reversible Watermarking**

I. Introduction

With the increasing internet connectivity and rapid usage, digital data representation and transmission of images, video and audio are on rise. The chances of a particular digital image of being tampered and misused by unauthorized person through internet are also on rise. This is a violation of one's intellectual property rights. Hence, protection of digital images is necessary and inevitable.

Digital watermarking [1] 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. In few applications like medical, military imaging and law enforcement, it is necessary to retrieve the exact original image after extracting the hidden watermark. The class of watermarking schemes that satisfy these requirements is called Reversible Watermarking (RW).

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. Coltuc et al. [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. Rajendra et al. [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, Dragoi et al. [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".

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.

Mathai N.J. et al. [7] discussed the various issues involved in hardware implementation of watermarking and proposed a video watermarking algorithm namely JAWS focusing on area, power and timing constraints. Garimella A. et al. [8] presented a hardware implementation for a watermarking technique which is fragile. The area and power are calculated using CMOS 0.13 micro meter technology. To reduce power, Mohanty S. P. et al. [9] presented a VLSI implementation based on dual voltage, dual frequency and clock gating. This method is defined for both visible and invisible watermarks using DCT. The authors also extended their work [10] to achieve high performance. The algorithms are implemented and performance analyzed in both Xilinx FPGA and full custom design. S. Karmani et al. [11] described a two-dimensional scan – based frequency domain watermarking for both video and image. The design is implemented in Altera Stratix – II FPGA. Results show that this method is robust against malicious attacks. T. C. Lad et al. [12] proposed a VLSI implementation of watermarking based on wavelets. The design is implemented on FPGA as well as through full custom circuitry and to optimize the area and power.

The drawback of the existing hardware implementations of watermarking is that all the operations are synchronized to the clock or the derivative of the clocks. As a result, at each clock edge, different blocks of the design switch and consumes significant amount of power.

Since, the power efficient hardware designs of watermarking are in great demand, an asynchronous clock – less VLSI architecture is proposed in this paper. Watermarking technique chosen for implementation is Dragoi et al. [6] scheme. It is a reversible image watermarking defined in spatial domain and is based on enhanced rhombus interpolation and difference expansion.

Section II briefs Dragoi et al. [6] RW scheme, Section III presents the proposed asynchronous pipeline architecture and implementation. Section IV includes results and Section V concludes paper.

II. Dragoi et al. Watermarking Scheme

There are three main steps in any reversible watermarking process: Watermark embedding, Watermark extraction and original image recovery. The methodology followed by Dragoi et al. [6] for these processes are shown in Figures 1 - 3.

A. Watermark Embedding:

Figure. 1 shows the flowchart for embedding the watermark. Cover image I is first preprocessed by defining a rhombus neighborhood. The pixel $i_{x,y}$ is modified by taking the average of its four immediate neighbors.

$$\hat{i}_{x,y} = \lfloor (i_{x-1,y} + i_{x,y-1} + i_{x+1,y} + i_{x,y+1}) * 0.25 \rfloor \tag{1}$$

If pixel region is uniform, estimation is done using Eq. 1. Else, the pixel is considered to be native to the most homogeneous group. To know the homogeneity, between the pixels their distance is computed for both the groups.

$$dist_h = i_{x,y-1} - i_{x,y+1} \tag{2}$$

$$dist_v = i_{x-1,y} - i_{x+1,y} \tag{3}$$

In both Eq. 2 and Eq. 3, only the modulus value is considered. If the uniformity of the rhombus is less than a threshold value, then the pixel is estimated using Eq. 1. If the horizontal distance is more than vertical distance, then the average of vertical neighbors is considered. Else, the average of horizontal neighbors is considered.

To embed the watermark, the estimation error parameter 'e' is used:

$$e = i_{x,y} - \widehat{i_{x,y}} \tag{4}$$

Watermark bit 'b' is embedded into the pixel i,. to form i' as follows:

$$i'_{x,y} = e + b + i_{x,y} \tag{5}$$

The limiting condition for the pixel is [0, L-1]. Here, for an 8 bit grayscale image, the value L-1 = 255. After embedding, the new estimation error is computed as follows:

$$e' = 2e + b \tag{6}$$

The steps for encoding are as follows:

- 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.
- To reduce large distortions, watermark is embedded in pixels with estimation error $e < T$, where T is the threshold.
- 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.
- The pixels with $e \geq T$ 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.
- The pixels of the S region are further grouped as two sets – "the *cross set*" and "the *dot set*".
- 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.
- While embedding the watermark bits, the *cross set* pixels are first checked if they are embeddable or not, followed by *dot set* pixels.

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 decoder decides if the pixel is embedded or not.

The watermark bit b is then extracted from the error e as follows:

$$b = (e') \bmod 2 \tag{7}$$

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.

C. Cover Image Restoration:

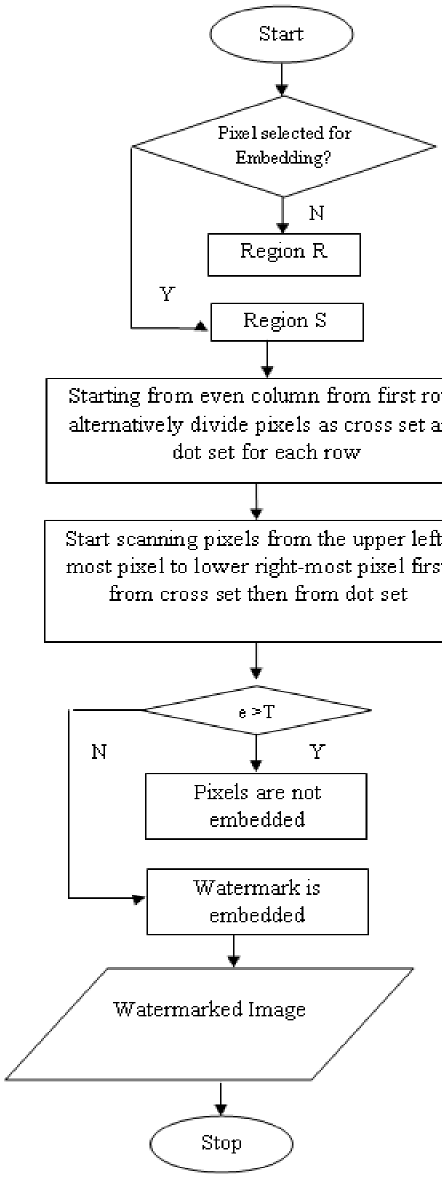
If a particular pixel was chosen for embedding, it is restored as,

$$i_{x,y} = i'_{x,y} - e' - b \quad (8)$$

If a particular pixel is "*not embedded*", its value is restored as:

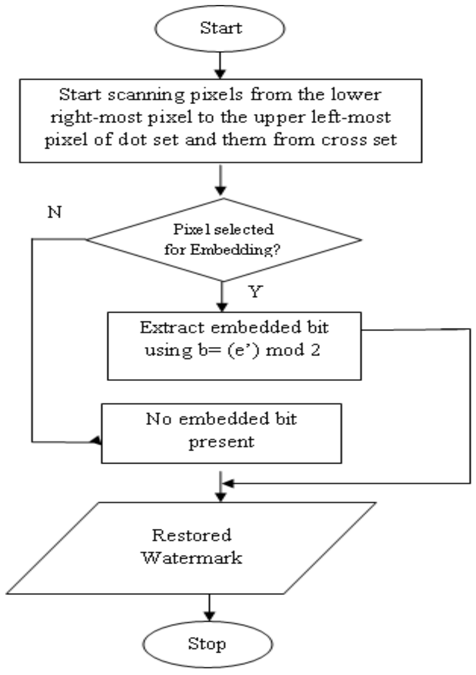
$$i_{x,y} = i'_{x,y} - T \text{ , If error '}e\text{' is positive} \quad (9)$$

$$i_{x,y} = i'_{x,y} - 1 + T \text{ , If error '}e\text{' is negative} \quad (10)$$



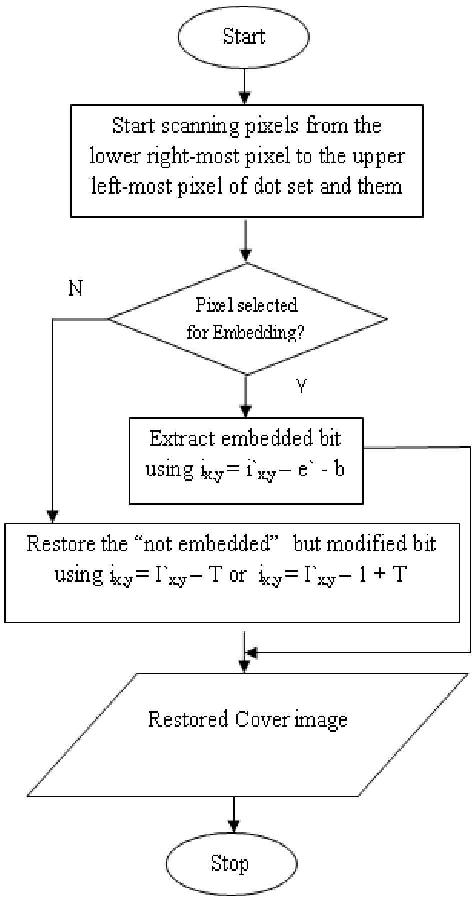
Start
Pixel selected for Embedding?
N
Region R
Y
Region S
Starting from even column from first row, alternatively divide pixels as cross set and dot set for each row
Start scanning pixels from the upper left-most pixel to lower right-most pixel first from cross set then from dot set
e >T
N
Y
Pixels are not embedded
Watermark is embedded
Watermarked Image
Stop

Fig. 1. Watermark Embedding



Start
Start scanning pixels from the lower right-most pixel to the upper left-most pixel of dot set and them from cross set
N
Pixel selected for Embedding?
Y
Extract embedded bit using b= (e') mod 2
No embedded bit present
Restored Watermark
Stop

Fig. 2. Watermark Extraction



Start
Start scanning pixels from the lower right-most pixel to the upper left-most pixel of dot set and them
N
Pixel selected for Embedding?
Y
Extract embedded bit using i_xy = i`_xy – e` - b
Restore the "not embedded" but modified bit using i_xy = I`_xy – T or i_xy = I`_xy – 1 + T
Restored Cover image
Stop

Fig. 3. Cover Image Restoration

III. Proposed 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.

In the absence of clocks, the synchronization between different modules is achieved through Request and Acknowledge handshaking protocols [14]. In this paper, an asynchronous two-phase handshaking scheme is chosen. It is called Mousetrap 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.

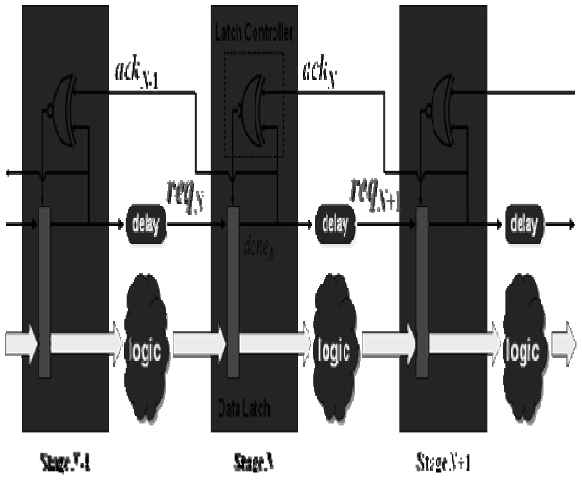


Fig. 4. Mousetrap pipeline

Initially all the *req* and *ack* are in the same state. The XNOR gate acts as a Latch Controller. For the Nth stage, when req_N is high, the latch controller output is '*0'* and the latch becomes opaque. This indicates a data in the Nth stage. Once ack_N becomes high the latch becomes transparent again.

Figure 5 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. 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.

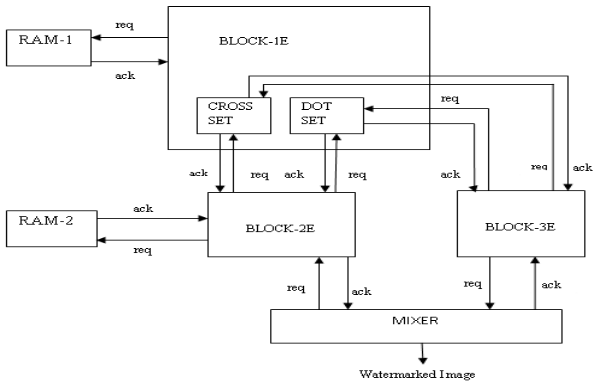


Fig. 5. Watermarking Encoder Top – Level Implementation

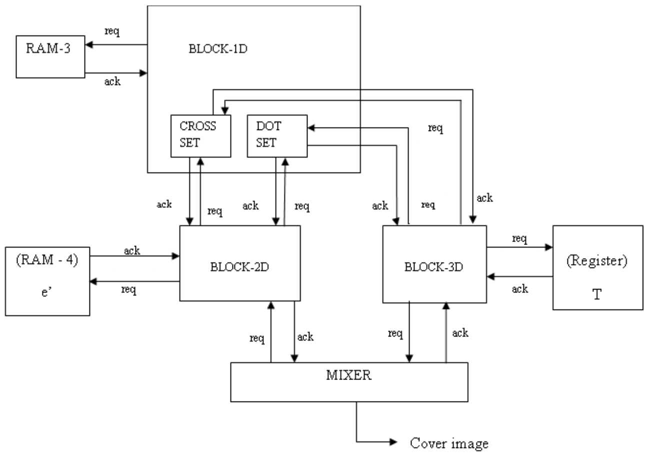


Fig. 6. Watermarking Decoder Top – Level Implementation

IV. Results And Discussions

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 I. 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 II. 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

V. Conclusion

To the best of our knowledge, this proposal is the first of its kind in asynchronous implementation of watermarking chips. The proposed architecture eliminates the hardware implementation issues of clock distribution and power dissipation due to clock switching. The design has lower power consumption compared to the existing synchronous architectures.

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.

[6] Catalin Dragoi, Dinu Coltuc, “Improved rhombus interpolation for reversible watermarking by difference expansion”, IEEE Transactions on EUSIPCO, pp. 1688-1692, 2012.

[7] N.J. Mathai, D. Kundur, A Sheikholeslami, "Hardware implementation perspectives of digital video watermarking algorithms," IEEE Transactions on Signal Processing, vol. 51 no.4, pp. 925-938, 2003.

[8] 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.

[9] 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

[10] 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, pp. 600-611, 2007.

[11] 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, pp. 801-811, 2013.

[12] 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.

[13] 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.

[14] Steven M. Nowick and Montek Singh, "High-Performance Asynchronous Pipelines: An Overview", IEEE Design & Test of Computers, vol. 28, pp. 8-22, 2011.

[15] 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.

[16] 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.

[17] 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.

[18] 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.