Novel Algorithm and Architectures for High-Speed Low-Power ConText-Based Steganography

Somayeh Timarchi
Department of Electrical Engineering
Shahid Beheshti University
Tehran, Iran
s_timarchi@sbu.ac.ir

Masoud Abbasi Alaei
Department of Electrical Engineering, Mathematics and Computer Science
University of Twente
Enschede, Netherlands
m.abbasialaei@utwente.nl

Hossein koushkbaghi
Department of Electrical Engineering
Shahid Beheshti University
Tehran, Iran
h.kooshkbaghi@mail.sbu.ac.ir

Abstract **- Least Significant Bit (LSB) insertion method is a popular type of steganographic algorithms in spatial domain. Nevertheless, in this approach essential measures should be considered to enhance the both visual quality and security properties. ConText is a revised version of LSB method to hide secret information in image carrier with enhanced visual imperceptibility. This paper introduces a novel algorithm based on ConText, called the Modified ConText (MCT). The proposed algorithm is based on using a threshold level to compare pixels in a sub-block which leads to faster and power efficient implementation. We strengthen the ConText algorithm which can embed data in a more noisy-like area to increase security and visual quality. Moreover, a high-speed hardware implementation of the MCT algorithm is also presented by employing faster comparisons. In addition to assigning threshold value that can lead to a more efficient architecture, the pre-computation low-power technique is also employed to reduce power consumption. The proposed architecture is synthesized by the ISE tool and implemented on a Spartan-3 FPGA device. The results imply that the proposed architecture outperforms the system frequency, the usage of FPGA resources, and power consumption by approximately 7%, 30%, and 64%, respectively.**

Index Terms - Steganography; LSB insertion; FPGA-based implementation; Pre-computation technique; Low-power design;

I. Introduction

In today's communications, protecting important information from unauthorized accesses and modifications is absolutely necessary [1]. There are various techniques to fulfill this desire including cryptography and steganography. While cryptography encodes and protects the content of data, steganography attempts to conceal the existence of data [2]. With the emergence of digital signal processing, steganography has been spread and emphasized in the digital domain. Even though all types of data, including image, video and text, can be used in steganography as cover (carrier) media, the digital image has been commonly utilized as the cover in steganographic systems [1, 3].

Steganography employs a wide range of techniques in order to hide important data in the cover media. One of the well-known approaches is to directly insert the data into the least significant bit. In this technique, the data are embedded in one or several least significant bit(s) of the image pixels. There are many articles about the spatial domain steganography in the literature which are elaborated in the following. Potdar et al. [4] used a method to produce a fingerprinted secret sharing steganography in order to resist against the cropping attack. Wang and his colleagues [5] offered a technique, in which the differences between the secret-embedded and original images are totally indistinguishable by human eyes. It also avoids falling off the boundary issue by utilizing the pixel value differencing method. Chan and Cheng [6] suggested a LSB technique based on optimal pixel adjustment. Lou and Liu [7] proposed a method that employs variable-size secret data and redundant Gaussian noise, in order to resist against the common-cover-carrier attack. In [8], an algorithm was presented on the basis of searching for regions with a great diversity of the gray-scale levels by dividing the image into some blocks and sub-blocks, and comparing the pixel values of the sub-blocks.

Although numerous steganographic techniques and algorithms have been published in many articles, the hardware implementation has not been adequately remarked till now. A few hardware implementations of LSB steganography have been presented since the last decade [10-18]. It is worthwhile to state that hardware implementation provides some benefits such as the capability of interacting with the client through the user interface, as well as some other advantages including the high speed and portability. FPGA-based implementation of steganographic algorithms seems to be an interesting option since its capacity for parallel processing could allow multi-channel processing [10]. FPGAs are reconfigurable, flexible and physically secure devices with high computational capabilities and offer a fast design cycle [10, 15].

In this paper, a new algorithm as well as hardware implementations is presented. In the proposed algorithm (called the Modified ConText, or shortly MCT), a threshold value is considered to specify the range of differences between two adjacent pixels in a 2×2 sub-block. So, a high-quality stego-image is attained. Moreover, the threshold assignment leads to a more efficient hardware implementation. A developed version of the MCT is also explores in order to achieve a higher quality stego-image with lower area and power consumption for hardware realization. Finally, two high-speed and low-power architectures are explored for MCT.

978-1-5386-4379-2/17/$31.00©2017 IEEE

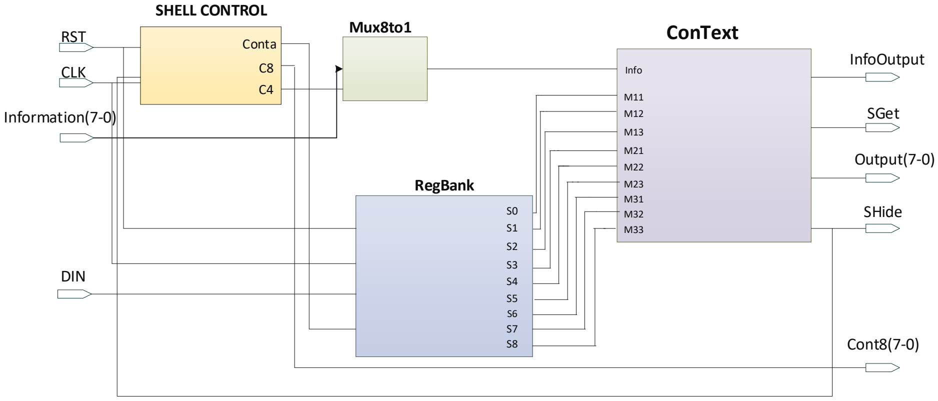


Fig.2. FPGA-based structure of the context algorithm in more detail [12].

The rest of this paper is organized as follows. In Section II, the ConText algorithm [8] and its hardware implementation [12] are described. In Section III, the novel ConText-based algorithm is developed and the advantages of the proposed algorithm are discussed. Two versions of MCT implementation are introduced in Section IV. Section V presents the simulation results and the statistical analysis of steganographic algorithms as well as the comparison between different hardware implementations. Finally, the paper is concluded in the last section.

II. ConText algorithm and its hardware implementation

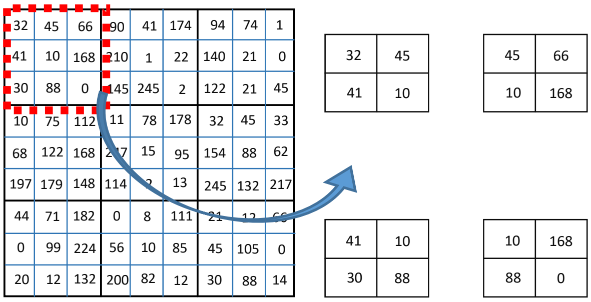
The ConText algorithm analyzes the image in the spatial domain and looks for an area with a greater diversity of gray-scale levels in order to embed the data. The steps of the ConText algorithms, shown in Fig. 1 are as follows [8]:

1) The cover image is divided into 3×3 blocks without overlapping

2) The 3×3 block is divided into four sub-blocks

3) If there are at least three different gray-scale levels in each sub-block as there is in the 3×3 block of Fig. 1, the block will be considered as a valid sub-block.

4) The LSB of the central pixel of the 3×3 block will be embedded, if the four sub-blocks are valid. After embedding the data bits, the validity of the four sub-blocks will be verified. If some sub-blocks are invalid, the data bit will not be inserted.



32	45	66	90	41	174	94	74	1
41	10	168	210	1	22	140	21	0
30	88	0	145	245	2	122	21	45
10	75	112	11	78	178	32	45	33
68	122	168	[illegible]	15	95	154	88	62
197	179	148	114	[illegible]	13	245	132	217
44	71	182	0	8	111	21	12	66
0	99	224	56	10	85	45	105	0
20	12	132	200	82	12	30	88	14

32	45
41	10

45	66
10	168

41	10
30	88

10	168
88	0

Fig.1. ConText algorithm. a) Dividing the cover image into 3×3 blocks. b) Dividing 3×3 block into four sub-blocks [8].

In the rest of this section, it is attempted to provide a general view for the hardware structure of the ConText algorithm. Fig. 2 presents the architecture, including the inputs and outputs and internal blocks. As shown in the figure, this architecture has four blocks: shell control, MUX 8 to 1, Reg bank, and ConText block [12].

The shell control block is used to control the process of embedding the data in the cover image. It operates as a finite-state machine. The MUX 8-to-1 is an 8-to-1 multiplexer that selects the information bit which requires to be hidden in the LSB of the pixel. The Reg bank stores 9 pixel values of the 3×3 block. Finally, the ConText block is the main part of the architecture, which performs the process of dividing the cover image into 3×3 blocks and 2×2 sub-blocks. The main computations are performed in this block and the most demanding operations refer to the comparisons made between the four values of the 2×2 sub-blocks.

III. PROPOSED MODIFIED CONTEXT (MCT) ALGORITHM

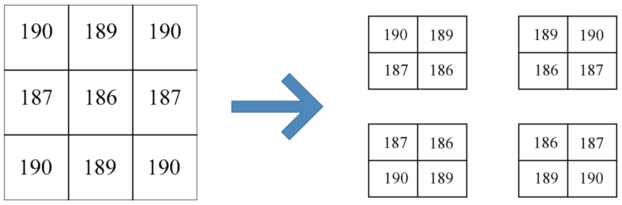
The proposed MCT algorithm is explored in this section. The proposed algorithm is based on the ConText algorithm with some modifications. The modifications have been organized in order to achieve a better quality of the output stego-image. Besides, the proposed hardware implementation outperforms the power consumption and the area occupation. The steps of the proposed five-step MCT algorithm are as follows:

1) The cover image is divided into 3×3 blocks without overlapping.
2) The 3×3 block is divided into four sub-blocks.
3) If there are, at least, three gray-scale levels in each sub-block, with level differences higher than the desired threshold level (δ), the sub-block will be considered as a valid sub-block.
4) The LSB of the central pixel of 3×3 block will be embedded, if its four sub-blocks are valid.
5) After insertion, the validity of the four sub-blocks will be verified. If there are less than three

different gray-scale levels considering the value δ, the data bit will not be inserted.

Generally, the new modification comprises employing a threshold level (δ). In order to verify the validity of the four sub-blocks, the differences must be higher than δ. Using threshold values leads to reduction in steganography capacity. This might be more distinguishable, for instance, when cover image has plain texture with a lower variety of grayscale levels. The Choice of proper threshold level will be studied in more detail in the next section. To have a clear understanding of the proposed algorithm, we give the following example .

Example: Let's assume the 3×3 block with nine values depicted in Fig. 3. Based on the ConText algorithm, the data bit can be inserted in the central pixel (186), but the block is not a noisy region, and is actually a smooth one. In order to increase the security of the conventional ConText algorithm, a threshold level, like 7 could be considered in the proposed MCT algorithm. By this assumption, the data bit will not be inserted because the difference between two adjacent values in the 2×2 sub-block is less than 7.



190	189	190
187	186	187
190	189	190

190	189
187	186

189	190
186	187

187	186
190	189

186	187
189	190

Fig. 3. Values of nine pixels in the 3×3 block (left) and four 2×2 sub-blocks (right)

To reduce the area overhead and achieve a fast implementation, the fifth step of the proposed algorithm could be eliminated. So that the four-step version of MCT algorithm is formed. This modification has a drawback as well. In fact, by eliminating the final step, the validity of the four sub-blocks will not be verified. As a result, after the insertion, some sub-blocks might become invalid, or in other words, some pixels values could fall below the threshold level (*δ*). However, this issue can be ignored, because a significant quality of the stego-image higher than the conventional ConText algorithm is obtained. On the other hand, without final verification performed after inserting data bit, the difference between two adjacent pixels is an issue. Therefore, the aforementioned issue is not a serious predicament.

It is essential to study the impacts of different threshold levels on the performance of the proposed algorithm and hardware implementation. Decision on an appropriate value used for *δ* needs to be made based on speed and power dissipation of the hardware implementation as well as capacity and cover image complexity. High threshold levels, for example *δ* = 7, lead to a higher PSNR (Peak Signal-to-Noise Ratio) and consequently a higher stego-image quality, along with a more efficient hardware implementation, i.e. higher speed and less power consumption. However, the cost of capacity reduction will be paid. On the other hand, choosing lower threshold value helps to achieve higher capacity, but stego-image quality as well as hardware implementation results will be degraded (slightly in speed and power consumption).

According to synthesis results, to achieve simultaneously reasonable results of hardware implementation and steganographic capacity, δ = 3 is chosen in this work. However, if designing priority is speed and steganographic capacity is less important, higher threshold levels can be selected.

In the rest of the paper we assume the four-step version of MCT algorithm (ignoring the fifth step) for implementation and comparison. Because of eliminating the validation step after insertion in the five-step version of MCT, less area and delay are achieved compared to the five-step version of the MCT block.

IV. High speed and low power Hardware Realization for MCT

In this section we proposed two versions of hardware implementations for MCT algorithm (MCT-s1 and MCT-s2). Both of them have their advantageous and disadvantageous. Higher speed, lower power consumption in conjunction with less usage of FPGA resources are the advantageous of the second implementation. On the other hand, the first one is more accurate, in cost of losing speed and power. The two structures will be discussed in the following.

The general structure of the MCT algorithm is explained. Fig. 4 depicts the general architecture of proposed MCT-S1 and MCT-S2 in details. According to this figure, the proposed structure contains four blocks, which are similar to the blocks offered in [12].

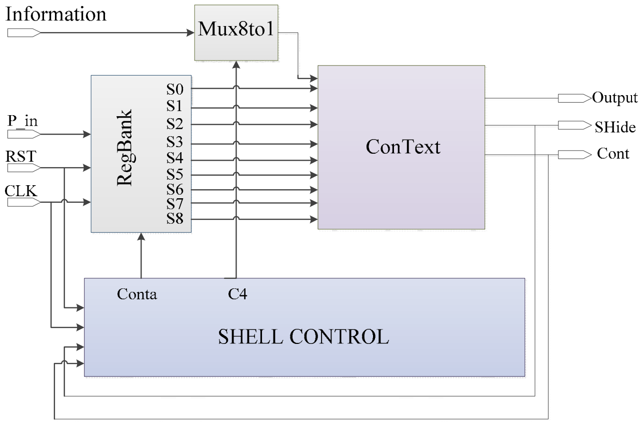


Fig. 4. The proposed MCT general structure in more details.

Fig. 5 shows the ConText block structure of the proposed MCT algorithm. M11, M12… M33 are nine input pixels. Output, Cont, and SHide are the output signals, which are similar to the ConText structure. Four comparators are needed to compare pixels in 2×2 sub blocks.

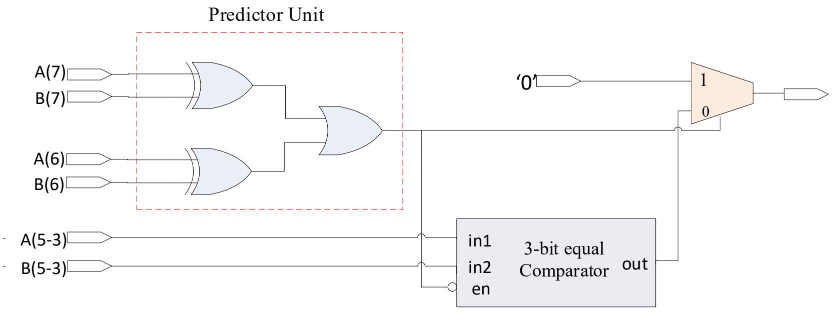


Fig.7. The proposed low-power equal comparator block of MCT-S2 structure employing the pre-computation technique.

Fig. 6 illustrates the comparator block in more details. This block is composed of four threshold-based comparators. A, B, C and D are the four pixel values of the 2×2 sub-blocks. Threshold-based comparator determines whether the difference of two pixels is above the threshold level (δ) or not. In this paper a threshold-based comparator is designed which is called MCT-S1.It is more accurate in making a comparison of two pixels compared to MCT-S2 but less efficient in speed and power consumption. The rest of this section explains the proposed architectures.

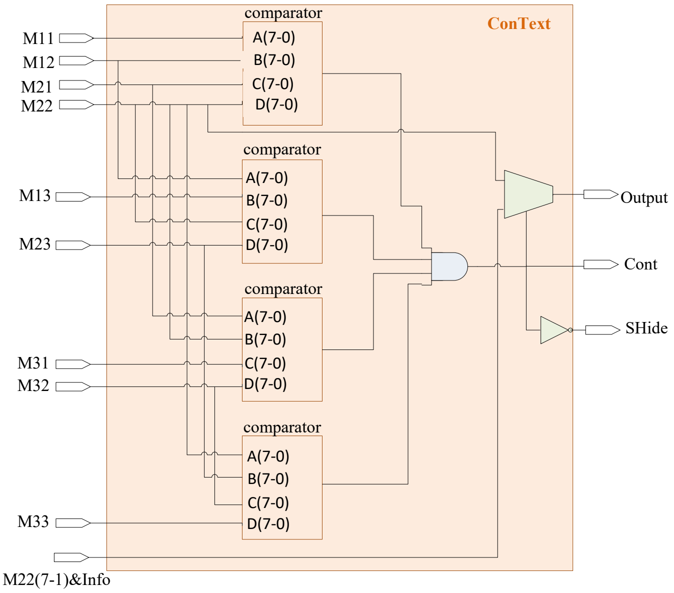


Fig. 5. Structure of *ConText* block of the proposed MCT-S1 and MCT-S2 architectures.

In the second implementation algorithm called MCT-S2 considerable alterations have been employed in the threshold-based comparator blocks. Instead of calculating the difference between the two adjacent pixels in a sub-block and comparing with threshold level, an alternative approach has been opted, explained as follows:

In this approach, according to the value of δ, some bits are ignored to speed up the comparison. Although this method cannot meet all of the states, it can fulfil our goal of fast and power efficient implementation. For δ = 3, we ignore 2 LSBs bits and compare the remaining ones. Three LSBs and four LSBs bits can be ignored for δ = 7 and δ = 15, respectively. Let's assume δ = 3. In order to evaluate the differences between two pixels in a sub-block, two LSBs of two pixels are neglected. Then, the rest of bits (i.e., 6 MSBs) are compared together.

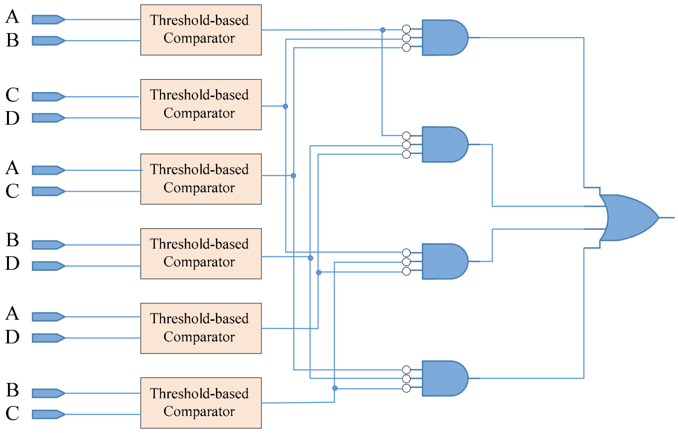

A
B
Threshold-based Comparator
C
D
Threshold-based Comparator
A
C
Threshold-based Comparator
B
D
Threshold-based Comparator
A
D
Threshold-based Comparator
B
C
Threshold-based Comparator

Fig. 6. The proposed *comparator* block of MCT-S1 and MCT-S2 architectures depicted in Fig. 5.

This approach leads to higher speed, less usage of FPGA resources and less power consumption. Comparing the results of proposed structure with other existing efficient one, illustrates that the improvement in speed and power consumption are remarkable. On the other hand, this strategy, fails in many cases. It is considered that some 2×2 blocks are valid, while they are not. In fact, the MCT-S2 is an appropriate implementation of the MCT algorithm. However, if efficient architecture is the first priority and accuracy comes second, it can be an ideal choice.

The proposed MCT-S2 architecture also employs a pre-computation technique in order to reduce the dynamic power consumption even more. In fact, the idea behind the pre-computation method is to selectively compute some logic outputs and make use of them for lowering the internal switching activity [15]. Based on the pre-computed values, the threshold-based comparator block can be either enabled or disabled. In the proposed structure, this technique is applied to the threshold-based comparators. By employing this technique, it is possible to use a smaller comparator.

The Threshold-based comparator block in the proposed structure (Fig. 6) consists of a predictor unit shown in Fig. 7 (δ = 3). Two MSBs of the input operands are employed to predict the comparison results. According to the truth table of predictor unit, only in 4 out of 16 states, the predictor output is zero, which enables the equal comparator block. In the remaining states, the result is produced during prediction phase. Using pre-computation technique makes a significant reduction in power consumption as discussed in the next section.

In the following, we first explore the first hardware implementation of the MCT algorithm called MCT-S1 which needs to compromise between accuracy and hardware efficiency. The only difference is in employing the strategy for comparing pixels in 2×2 sub-blocks. In the second approach, we use straightforward method to compare pixels (i.e. subtract two pixels and then compare the result with *δ*). It may look very time consuming, but by combining subtraction and comparison operations, the results will be improved.

V. SIMULATION AND COMPARISON

The results of the proposed MCT hardware implementations are presented in this section. The experimental findings can be divided into two major categories: the evaluation results of the proposed algorithms, and the experimental results of the proposed hardware implementations. Two experiments are carried out to evaluate the performance of the proposed algorithms, in terms of the embedding capacity and stego-image quality.

Four 512×512 gray-scale images are chosen as the cover images, as displayed in Fig. 8. Finally, the experimental results of hardware implementation are presented in order to demonstrate that the proposed architectures can perform more efficiently than the most efficient existing ones.

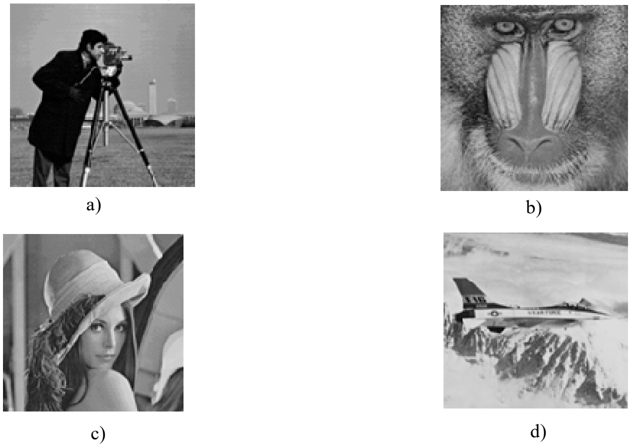

a)
b)
c)
d)

Fig. 8. Cover images. a) Cameraman b) jet plane c) Lena d) Mandrill.

As mentioned earlier in this section, two criteria are chosen in order to evaluate the proposed algorithms (embedding capacity and PSNR). Fig. 9 illustrates the effects of several values of δ on the embedding capacity for the proposed MCT algorithm. This figure depicts that a lower value of δ would lead to a higher capacity.

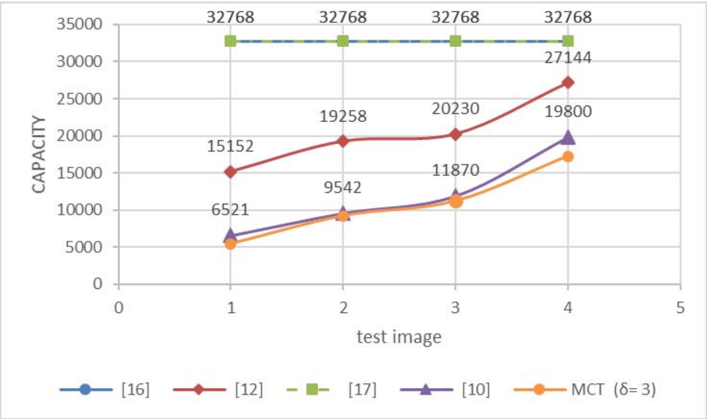


Fig.9. Embedding capacity of the steganography algorithms versus test images.

The second criterion for examining the performance of the proposed algorithms is the peak signal-to-noise ratio (PSNR). The results of comparing the proposed algorithms with the algorithms presented in [10, 12, 16, 17], are depicted in Fig. 10.

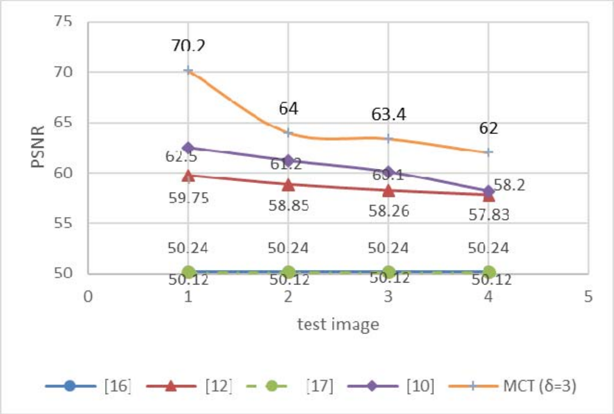


Fig. 10. Comparison of PSNR versus test images.

- Experimental results of hardware implementation

At first, the proposed architecture was modeled at the register transfer level by using the VHDL. The functional testing and simulation were performed using ModelSim 6.5. Additionally, ISE 14.2 was chosen to synthesize, map, and route on the Spartan-3 device family.

The results of the hardware implementations are shown in Table I. This table exhibits the FPGA results of the architecture based on the MCT-S1 and MCT-S2 architectures along with four efficient ones.

Table I also illustrates power consumption values obtained by Xilinx tools. Total on-chip power (static design + dynamic design) contains power dissipated on-chip from any supply source. Static power is the power when the device is configured and there is no switching activity. Dynamic power is the average power from logic utilization and switching activity. Significant impact of pre-computation technique on power consumption can be observed form Table I.

TABLE.I. RESULTS OF IMPLEMENTATION ON FPGA SPARTAN 3

	[26]	[22]	[27]	[20]	MCT-S1(δ=3)	MCT-S2(δ=3)
Clock (MHZ)	140	95	155	80	**125**	**167**
Number of slice Flip Flops	76	93	68	687	**62**	**45**
Number of 4-input LUTs	92	124	84	1245	**76**	**50**
Number of occupied slices	82	110	75	1240	**74**	**64**
Total on-chip power (W)	0.245	0.187	0.195	0.874	**0.121**	**0.067**

VI. CONCLUSION

In this paper, a novel algorithm (i.e., MCT) was proposed based on the ConText algorithm along with two hardware implementations. The algorithm employs a threshold value for deciding on the embedding process. The proposed algorithm enhances the quality of the stego-image by improving the PSNR criterion. Besides, the proposed approaches lead to a more efficient hardware implementation. The proposed architecture for the four-step MCT algorithm utilizes fewer resources than the most efficient existing steganographic architectures. Besides, the system frequency is improved. Employing a threshold value causes to ignoring some bits in the comparison stage and reducing the area occupation by half. Moreover, in order to decreasing the power consumption, the pre-computation technique was utilized by controlling the enable pin of the main part of computations for some inputs. The proposed MCT-S2 architecture outperforms the system frequency of the most efficient architecture by approximately 7% and reduces the usage of resources and power consumption by 30% and 64%, respectively.

REFERENCES

[1] Cheddad, A., “Digital image steganography: Survey and analysis of current methods” Signal Processing, vol. 90, pp. 727-752, 2010.

[2] Jung, K. and Yoo, K., “Data hiding method using image interpolation”, Computer Standards & Interfaces, vol. 31, no. 2, pp. 465-470, 2009.

[3] Liu, C. and Liao, S., “High-performance JPEG steganography using complementary embedding strategy”, Pattern Recognition, vol. 41, no. 2, pp. 2945-2955, 2008.

[4] Potdar, V.M., Han, S. and Chang, E., “Fingerprinted secret sharing steganography for robustness against image cropping attacks”, Proceedings of IEEE Third International Conference on Industrial Informatics (INDIN), Perth, Australia, pp. 717–724, 2005.

[5] Wang et al., “A high quality steganographic method with pixel-value differencing and modulus function”, Journal of Systems and Software, vol. 81, no. 1, pp. 150-158, 2008.

[6] Chan, C.K., Cheng, L.M., “Hiding data in images by simple LSB substitution”, Pattern Recognition, vol. 37, no. 3, pp. 469-474, 2004.

[7] Lou, D.C., Liu, J.L., “Steganographic method for secure communications”, Computers and Security, pp. 449–460, 2002.

[8] Herrera-Moro, Dulce R., Raúl Rodríguez-Colín, and Claudia Feregrino-Uribe, “Adaptive Steganography based on textures” In Electronics, Communications and Computers, 17th International Conference on, pp. 34-34. IEEE, 2007.

[9] Luo, W., Huang, F., & Huang, J. (2010). Edge adaptive image steganography based on LSB matching revisited. Information Forensics and Security, IEEE Transactions on, 5(2), 201-214.

[10] Laces, P., Antonio, W., & Garcia-Hernandez, J. J. (2015, June). FPGA implementation of a low complexity steganographic system for digital images. In Computer and Information Science (ICIS), 2015 IEEE/ACIS 14th International Conference on (pp. 319-324). IEEE.

[11] Kait, V. S., & Chauhan, B. (2015, April). BPCS steganography for data security using FPGA implementation. In Communications and Signal Processing (ICCSP), 2015 International Conference on (pp. 1887-1891). IEEE.

[12] Gomez-Hernandez, E., Feregrino-Uribe, C., and Cumplido, R., “FPGA Hardware Architecture of the Steganographic ConText Technique” 18th International Conference on Electronics, Communications and Computers IEEE, pp. 123-128, 2008.

[13] Hala A. F., and Magdy, S., “Design and Implementation of a Secret Key Steganographic Micro Architecture Employing FPGA”, Asia and South Pacific Design Automation Conference (ASP-DAC‟04), 2004.

[14] Amirtharajan, R., Balaguru, R., Ganesan, V., “Design and analysis of Prototype Hardware for Secret sharing using 2-D Image Processing”, International Journal of Computer Applications, Vol. 4, No. 4, pp. 0975-8887, 2010.

[15] Mahmoudpour,S., and Mirzakuchaki, S., “Hardware Architecture for a message Hiding Algorithm with novel Randomizer”, International journal of Computer Application, Vol. 37, No. 7, pp. 46-53, 2012.

[16] Sundararaman, R., “Stego System on Chip with LFSR based Information Hiding Approach,” International Journal of Computer Applications, vol. 18, 2011.

[17] Mohd, B. J., Abed, S., Al-Hayajneh, T., and Alouneh, S., "FPGA hardware of the LSB steganography method," International Conference on Computer, Information and Telecommunication Systems (CITS), pp. 1-4, 2012.

[18] Bhunia, Swarup, and Saibal Mukhopadhyay. Low-power variation-tolerant design in nanometer silicon. Springer, 2011.