FPGA Hardware Architecture of the Steganographic ConText Technique

Edgar Gómez-Hernández, Claudia Feregrino-Uribe, Rene Cumplido
Nacional Institute for Astrophysics, Optics and Electronics
Luis Enrique Erro No.1, Sta. María Tonantzintla, Puebla, México, 72840
{ed06mx, cferegrino, rcumplido}@inaoep.mx

Abstract

This work presents a hardware architecture of the ConText steganographic technique in a Cyclone II FPGA of the Altera family. The ConText technique takes advantage of noisy regions and those with abrupt gray levels changes in an image where the hidden information is very difficult to detect; the process to locate this region is highly repetitive and computationally expensive. The technique is implemented in an FPGA to increase the processing speed. The implementation results show a throughput of 61.5 Mbps.

1. Introduction

The term Steganography comes from the Greek words *stegos* (cover) and *graphy* (write), and it literally means *covered writing*. Contrary to the cryptography, the steganography does not change the message, just hides it from people whom it is not this directed to [1].

The steganography is the art and the science of the hidden communications. As the cryptography, the steganography has been practiced for many years. In the past, people made it with hidden tattoos, with invisible inks, wooden charts with wax, among many other methods. Currently the same principle applies when sending some message; the sender simply hides such a message by keeping it inside some other file, image, text, audio or video [3].

The elements of a steganographic system are the following:

- *Cover-object*: It is any object that may carry a hidden message, for example: images, audio, video, text, HTML files, etc.
- *Stego-object*: refers to the object which is carrying a hidden message. *Cover-object + message = stego-object*.

In the same way that the cryptoanalysis corresponds to cryptography, whose objective is to obtain the coded information, the steganalysis corresponds to steganography, which looks for the existence of hidden information in some object. The main strategy of the steganalysis is to look for strong changes among the original object (*cover-object*) and the object that is analyzed (*stego-object*). For such a reason there has been developed a type of steganographic algorithms that are based on incrusting the information depending on the peculiar characteristics of the cover-object, looking for regions in which the information does not produce detectable changes.

In the case of the digital images, these places are found where there are noise or borders, that is where there are abrupt changes in the pixels tones (gray scale).

Among the steganographic techniques that exist today for images are [4] :

- Masked and filtrate: Masking and filtering techniques, usually restricted to 24-bit and gray-scale images, hide information by marking an image, a manner similar to paper watermarks. Watermarking techniques may be applied without fear of image destruction due to lossy compression because they are more integrated into the image.
- Algorithms and Transformations (embedded in the transformed domain) Most of the work in this category has been concentrated on making use of redundancies in the DCT (discrete cosine transform) domain, which is used in JPEG compression. But there have been other algorithms which make use of other transform domains such as the frequency domain.
- Least Significant Bit (LSB) insertion (embedded in the space domain)

Insert in the LSB. This technique is the most popular, but the easiest of attacking. It takes an image, and reconstructs the original image in another, the least significant bit of the message to hide changes.

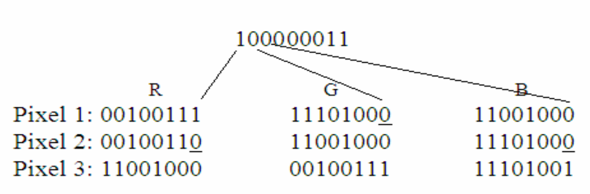
0-7695-3120-2/08 $25.00 © 2008 IEEE
DOI 10.1109/CONIELECOMP.2008.24

IEEE computer society

Example LSB: To insert an A : 100000011 in the 3 pixels RGB:

	R	G	B
Pixel 1:	00100111	11101001	11001000
Pixel 2:	00100111	11001000	11101001
Pixel 3:	11001000	00100111	11101001

The underlined bits are modified bits, the others are equal:



100000011

	R	G	B
Pixel 1:	00100111	11101000	11001000
Pixel 2:	00100110	11001000	11101000
Pixel 3:	11001000	00100111	11101001

The Context technique hides the information using this method.

There have been some hardware architectures of steganographic algorithms reported in the literature [5,6]. In [5], a steganographic algorithm that instead of using conventional substitution and translation operations on the plaintext characters uses simple plaintext hiding in a random bit string called the hiding vector is reported. And in [6], a secret key steganographic algorithm is described, that given a message aims to hide it into a cover such that even if an attacker detects the existence of the message, the attacker will not be able to recover it without the secret key that is known only to sender and receiver. A comparison of the architecture results is shown in Table 3.

In spite of the attractiveness of speeding up the insertion and recovering process in steganographic algorithms, not much has been done in hardware about steganography. This work aims to do that by presenting hardware architecture of the Context steganographic algorithm. The following section describes the ConText algorithm, as it is described in [2].

The paper is organized as follows. Section 2 provides an introduction to the ConText Technique as it is described in [2]. Section 3 shows the proposed architecture for the ConText algorithm and its modules are described in detail. Section 4 describes the implementation of the architecture, Section 5 shows the experimental results and finally Section 6 concludes.

2. ConText Technique

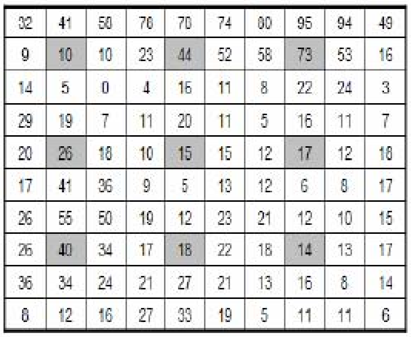
To be able to correctly identify the regions where the information will be inserted, the gray scale level is analyzed in the spatial distribution, looking for the areas with greater diversity of gray scale levels. The selection process of the pixel where the information will be inserted is described in the following steps [2]:

- Divide the image in non overlapping blocks of 3x3 pixels (Figure 1).
- The block of 3x3 is divided in four sub-blocks of 2x2 pixels (Figure 2).
- Each sub-block of 2x2 is considered valid if there are at least 3 values of gray scale levels.
- The information is inserted in the 3x3 block center if the four sub-blocks are valid.
- The validity of the four sub-blocks of 2x2 is verified after having inserted the information. If some 2x2 sub-block is not valid after hiding the information in the 3x3 block, then the inserted bit is not considered as hidden to avoid losing information during the recovery process.

In the Figures 1 and 2 this process can be appreciated.

3. ConText Hardware Architecture

As it can be appreciated from the ConText technique, the most demanding operation is the comparison of the pixel values of the 4 generated sub-blocks.



32	41	50	76	70	74	80	95	94	49
9	10	10	23	44	52	58	73	53	16
14	5	0	4	16	11	8	22	24	3
29	19	7	11	20	11	5	16	11	7
20	26	18	10	15	15	12	17	12	18
17	41	36	9	5	13	12	6	8	17
26	55	50	19	12	23	21	12	10	15
26	40	34	17	18	22	18	14	13	17
36	34	24	21	27	21	13	16	8	14
8	12	16	27	33	19	5	11	11	6

Figure 1. The image is divided in 3x3 blocks.

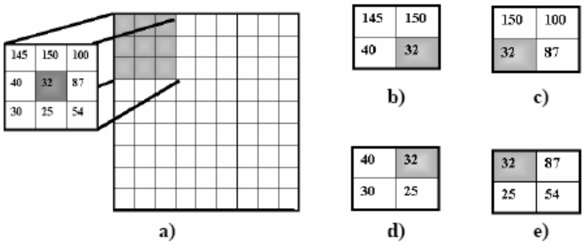


Figure 2. The 3x3 block is divided into four 2x2 sub-blocks.

The hardware implementation was based on the generation of blocks that perform simple operations, following the Top - Down methodology.

3.1 Architecture

Figure 3. shows the general block of the architecture, showing the input and output data for the ConText technique.

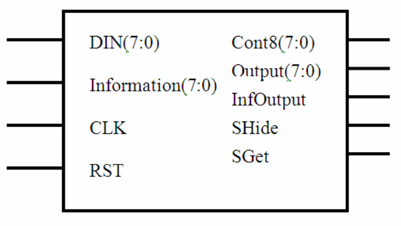


Figure 3. ConText Interface.

The inputs are:

- **Din**: Pixel value of the image.
- **Information**: The information that was incrusted in the image, if it is the case.

The outputs of ConText architecture are:

- **Cont**: It takes the control data input data to hide in the image (*cover-object*).
- **Output:** Central pixel value of the 3x3 matrix after inserting the information in the image (*cover-object*).
- **InfOutput**: Extracted information of the central pixel of the 3x3 matrix (*stego-object*).
- **SHide**: Output bit that is activated if it is selected to hide information in the central pixel of the 3x3 matrix (*cover-object*).
- **SGet**: Output bit that is set to 1 if information can be obtained from the central pixel of the 3x3 matrix (*stego object*).

Figure 4. shows the architecture in more detail. It can be seen that it has four blocks: 1) Shell_controll1, 2) Mux8to1, 3) Regbank, and 4) ConText. Next, these four blocks are described deeper.

1) Block Shell_controll1

This block is a state machine that controls the other blocks. It also keeps counters that control the input data flow (Din, Information). It inputs a signal (Ins) that indicates if it is possible to incrust information in the 3x3 matrix (stego object) and their outputs are:

- **Conta**: It controls the input pixels of the image (*stego object*) to be stored in the RegBank (Bank of Registers).
- **C8**: It controls the input of the 8 bits vectors information that will be incrusted in the image.
- **C4**: It selects from the 8 bits vector the bit to be inserted in the 3x3 matrix.

2) Block Mux8to1

Block that is used to select the information bit that was hidden in the 3x3 matrix.

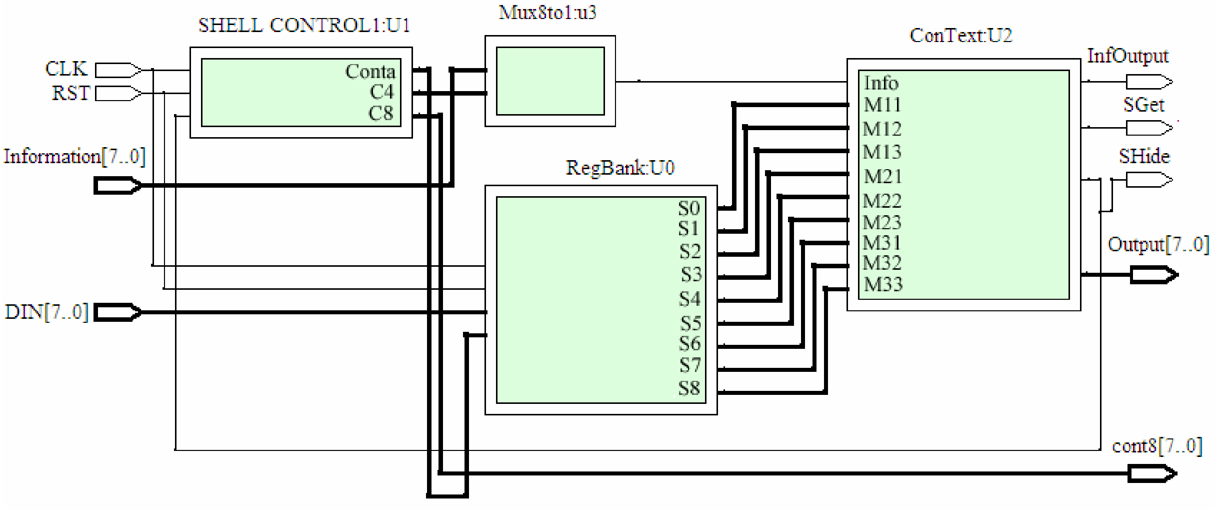


Figure 4 Implementation of the ConText technique.

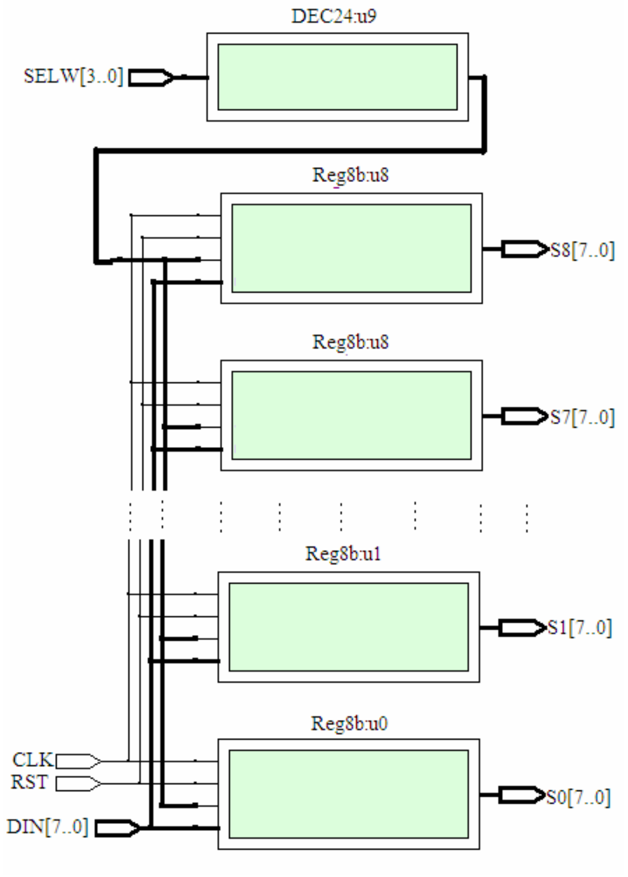


Figure 5. RegBank (Register's bank).

3) Block RegBank

This block is a registers bank that stores the values of the 3x3 matrix to be used by the ConText block (Figure 5).

4) Block ConText

This block carries out the main part of the architecture, which consists on the execution of the ConText technique, where the division of the 3x3 matrix in the four 2x2 sub-blocks is carried out and later on their comparison. As it can be appreciated in the Figure 6, the block consists of 8 sub-blocks. Each sub-block (Figure 7) compares the values of the 2x2 sub-blocks, the first four ones are the input matrix and the following ones are to compare the values of the 3x3 matrix with the incrusted bit of information. The purpose is to verify if the condition (if the four 2x2 sub-blocks are valid) is preserved. If the condition is lost when incrusting the bit of information, the incrusted bit is left there and it does not count as an inserted bit in order to avoid loss of information during the recovery process.

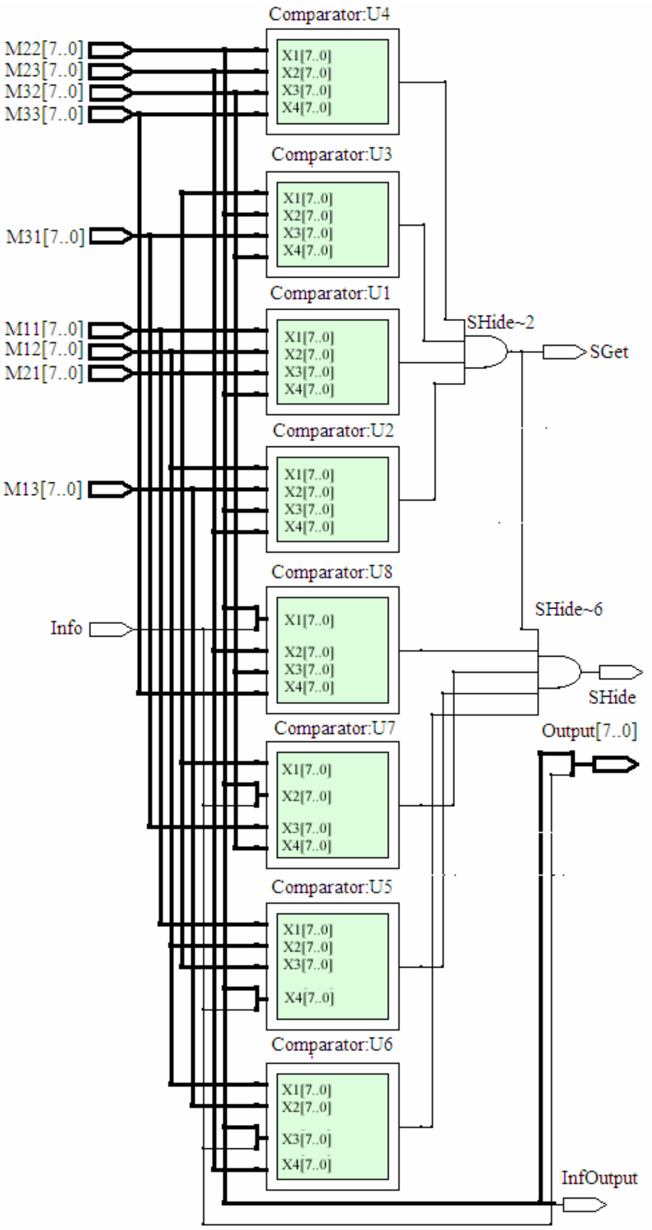


Figure 6. ConText Block.

4. Implementation

The hardware architecture of the ConText technique was developed using the description language VHDL (Vhsic Hardwre Description Language). The simulation tests were carried out with the ModelSim tool from Xilinx. For the synthesis and the place and route, the Altera software Quartus II was used. Lastly, to test the operation, a prototyping card with a Cyclone II EP2C35F672C6 from Altera, MatLab 7 with Simulink and DSP Builder version 6.1 from Altera were used.

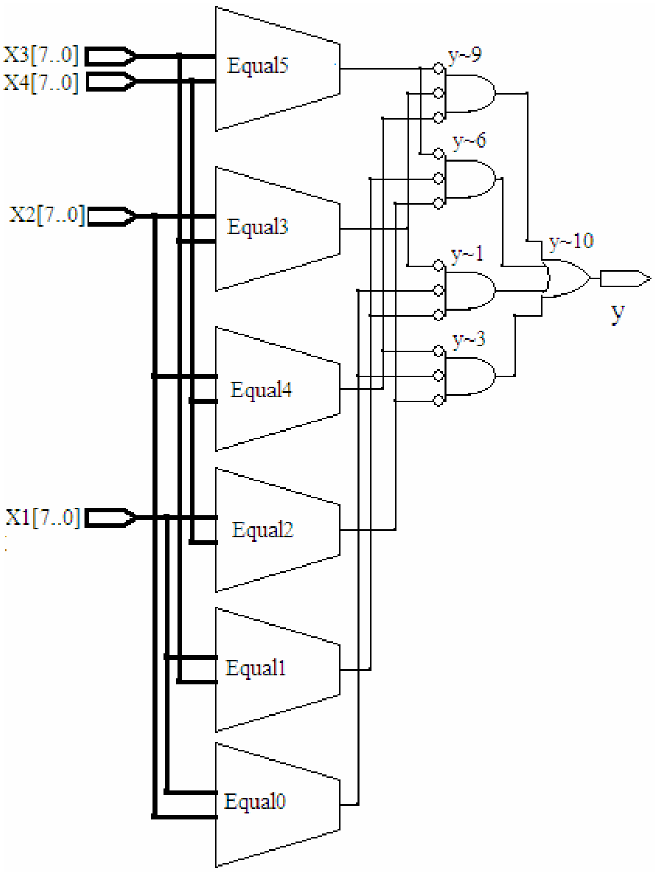
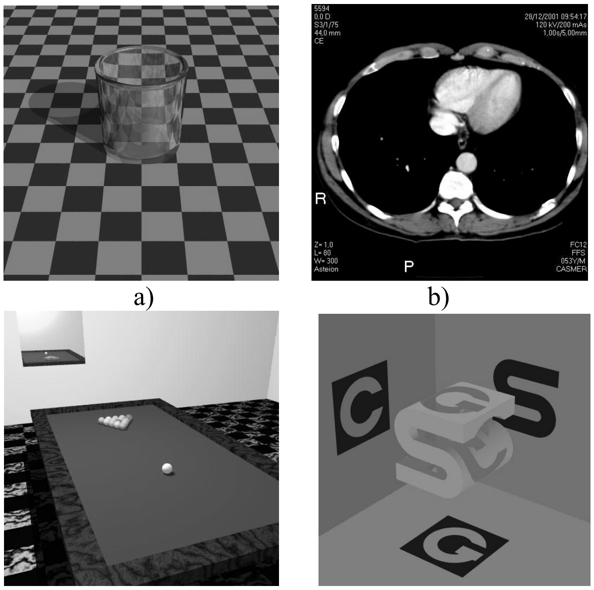


Figure 7. Context Sub-block



a) b)

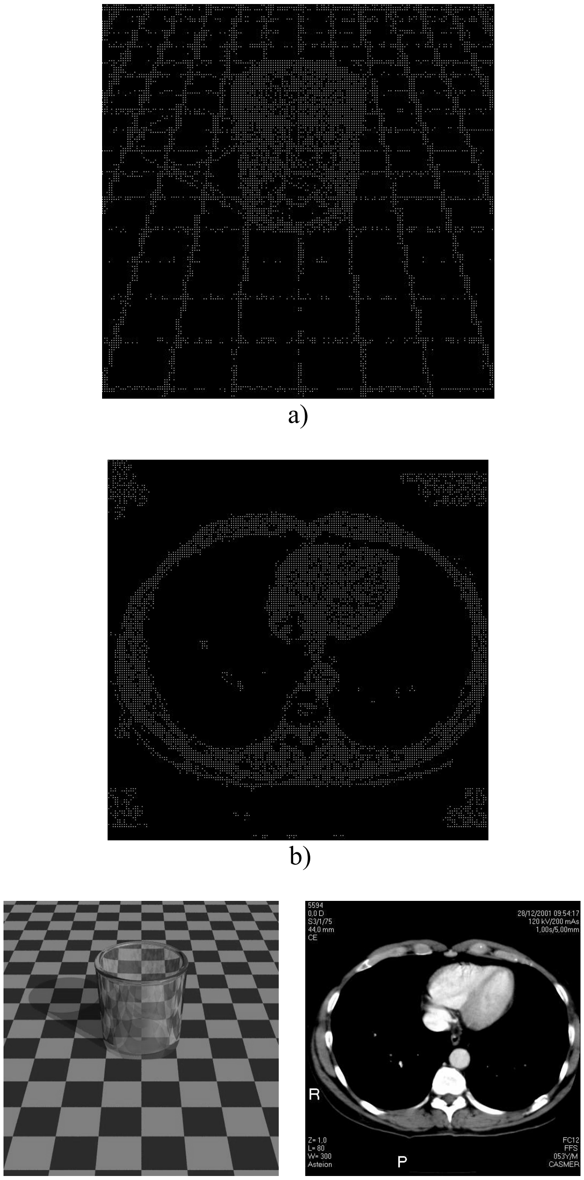
c) d)

Figure 8. Test images

5. Results

The images used for testing the architecture are shown in Figure 8. They include different gray scale images that range from geometric forms to medical magnetic resonance images.

Figures 9a) and 9b) show the pixels where the information will be hidden from Figures 8a) and 8b). Figures 9c) and 9d) show the stegoimages. From the figures, it can be appreciated that no change can be detected visually.



a)

b)

c) d)

Figure 9. Output Imagines

The synthesis results are shown in Table 1. It can be seen that the architecture may process eight million pixels per second, that is, 61.54 Mbps.

Logical elements	204
Maximum frequency (MHz)	106.75
Throughput (Mbps)	61.54
Yield (pixel/s)	8,066,000
Yield / Area (pixel/s / logical elements)	39, 539

Table 1. Results of the synthesis in FPGA

The tests of the algorithm implemented in software that served as the basis of this architecture [2] were carried out in a computer with a Celeron processor at 1.4 GHz and 512 MB of RAM memory, Windows XP operating system with Matlab version 7.0.

As it can be seen in Table 2, the software implementation runs on average in 8.089 seconds per image, whereas the hardware architecture achieves an average of 0.0325 seconds per image. That means, the hardware architecture perform 252 times faster than the software implementation.

Image	Size	Software	FPGA
a	512x512	6.5818	0.0325
b	512x512	5.1174	0.0325
c	512x512	10.6524	0.0325
d	512x512	10.0044	0.0325

Table 2. Comparison of times (seconds) of the implementations in Software and in FPGA.

	[5]	[6]
Throughput (Mbps)	95.532	1.576
Area (CLB)	168	
Functional Density (Mbps/CLB)	0.569	
Frequency (MHz)		35.4

Table 3. Performance report reference.

A comparison of the performance of the software implementation of the ConText algorithm against other similar algorithms can be consulted at the reference [2]. No other hardware implementation of the ConText algorithm has been reported in the literature, so it is not possible to make a comparison of performance. On the other hand, the comparison between architectures of algorithms different from ConText is not fair since they would not be compared under the same conditions; furthermore the algorithms are based on different techniques.

6. Conclusions

The FPGA hardware architecture of the ConText technique has shown a significant improvement over a Matlab implementation. Although it is not a fair comparison, it gives an idea of the gains achieved by hardware implementations in this type of steganographic algorithms. Even if a faster software implementation were available, this hardware implementation performs 8 comparisons in a single clock cycle, so, it is not expected that this hardware implementation can be outperformed by a software one.

As further work, the modules or blocks that perform simple and repetitive operations, such as the Context Block, could be parallelized in order to achieve that all the comparisons are executed in only one clock cycle, Care must be taken if a reduced area architecture is desirable.

7. References

[1] J. Fridich, and R. Du, "Secure Steganographics Methods for Palette Images", In Information Hiding, 3rd International Workshop, Springer 1999,pp. 47-60.

[2] Dulce R. Herrera-Moro, Raúl Rodríguez-Colín, Claudia Feregrino-Uribe, "Adaptive Steganography based on textures" Electronics, Communications and Computers, 2007 CONIELECOMP '07 17th international Conference Page(s): 34 - 34 .

[3] Mehdi Kharrazi, Husrev T. Sencar and Nasir "Image Steganography: concepts and practice." Memon. Polytechnic University. Broklyn, NY. USA. April 22, 2004 www.ims.nus.edu.sg/preprints/2004-25.pdf.

[4] Neil F. Johnson, Sushil Jajodia, "Exploring Steganography: Seeing the Unseen ".Journal Title: IEEE Computer. Date: 1998. Volume: 31. Issue: 2. p. 26 – 34.

[5] Farouk, H.A. Saeb, M. " Hybrid Hiding Encryption Algorithm (MHHEA) for Data Communication Security base on Hybrid Hiding Encryption Algorithm (HHEA) " Comput. Dept., Arab Acad. for Sci., Technol. & Maritime Transp., Alexandria, Egypt; Design, Automation and Test in Europe, 2005. 7-11 March 2005.

[6] Farouk, H. Saeb, M. " Design and implementation of a secret key steganographic micro-architecture employing FPGA " Dept. of Comput., Arab Acad. for Sci. Technol. & Maritime Transport, Alexandria, Egypt; Design, Automation and Test in Europe Conference and Exhibition, 2004. 16-20 Feb. 2004 Volume: 3, On page(s): 212- 217 Vol.3.