Hardware Architecture for a Message Hiding Algorithm with Novel Randomizers

Saeed Mahmoudpour
Department of electrical engineering,
Qazvin branch
Islamic Azad University

Sattar Mirzakuchaki
Department of electrical engineering
Iran University of Science and Technology,
Tehran, Iran

ABSTRACT
Steganography is the art of hiding information in a cover medium such that the existence of information is concealed. An image is a suitable cover medium for steganography because of its great amount of redundant spaces. One simple method of image steganography is the replacement of the least significant bit (LSB) of a cover image with a message bit. This represents a high embedding capacity but it is detectable by statistical analysis methods such as Regular-Singular (RS) and Chi-square analyses. Therefore, a new LSB algorithm is proposed here which can effectively resist statistical analysis. In this novel algorithm, every two sample's LSB bits are combined using addition modulo 2 which is compared to the secret message. If these two values are not equal, their difference is added to the second sample. Otherwise, no change is made. This paper proposes a hardware realization of this new algorithm. Furthermore, two scalable pixel interleaver and novel message bit randomizer with two different stego-keys are designed. Pixel interleaver can improve resistance against visual analysis by random selection of pixels.

Keywords
LSB steganography; hardware description language; FPGA; pseudo-random number generator.

1. INTRODUCTION
In the modern world, information is converted from paper type to digital information. Therefore, security improvement in data saving and exchanging is important.

Different techniques of cryptography are used for data encryption but all of these methods can be recognized by invaders. If the information can be embedded in a medium in such a way that it cannot be observable easily, it will not raise the suspicion of invaders. This is the main idea of steganography.

In other words, steganography is the art of embedding information in a cover medium such that the existence of data is concealed. Many different file formats can be used to achieve this purpose but digital images are the most popular carrier files. The way images are stored creates a great amount of redundant space which is the ideal place to hide information [1].

One of the most common methods of image steganography is to replace least significant bits (LSB) of the cover medium by message bits. For instance, a simple method proposed is to place the embedding data at the least significant bit of each pixel in the cover image.

The image formats used typically in such steganographical methods are lossless and the data can be directly manipulated and recovered. Since bmp images use lossless compression, one form of LSB attempts to use bmp images. However, other image formats are used as cover image as well.

LSB method represents a high embedding capacity while it is detectable by statistical analysis such as Regular-Singular (RS) and Chi-square analyses [2],[3]. Therefore, a new method of LSB steganography is proposed by Zhang in [4] which can resist against statistical analysis.

Presently, different methods of steganography need a computer to hide and extract the secret data. This poses serious restrictions on portability, cost, and speed performance of the system. Hardware model of steganographical algorithms can overcome these limitations. User can transfer the message via a portable memory like flash memory to the chip and after embedding the message in a cover image, stego-image will be sent back to the flash drive [5]. Designing a hardware model for this new algorithm of LSB which can resist statistical analyses is the major goal of our research.

In [6], four different steganographical algorithms are compared for their suitability in hardware implementation. They are: 1) Novel image hiding scheme based on block difference [7]. 2) Data hiding in images by adaptive LSB substitution based on the pixel-value differencing [8]. 3) LSB steganographic algorithm based on predictive neighbor pixels [9]. 4) Simple 2-bit LSB substitution steganography [10]. To choose the right scheme for hardware design, timing requirement and complexity in implementation is considered. In these four schemes, LSB algorithms (scheme 2, 3 and 4) are suitable for hardware implementation because of their acceptable complexity and short processing time.

Hardware implementation of the classic LSB algorithm is proposed in [5]. However, this algorithm is not stable against statistical analyses.

Zhang proposed an algorithm which can resist against statistical analyses and it is more convenient for implementation. Therefore, Zhang algorithm is selected for hardware implementation in this paper. Moreover, two randomizers are designed in this paper which can improve security. Consequently, the aim of this study is to implement Zhang algorithm which has unique advantages in comparison with classic LSB. Besides, two novel randomizers for pixel and message bits, which result in a safe and secure system, are also proposed here.

As mentioned earlier, in this design, two randomizers with two secret keys are used for security improvement. 1) A scalable pixel interleaver with a unique key selects pixels for message embedding. Interleaver can change its length according to the size of image. It means that the interleaver generates numbers according to the number of pixels and prevents redundant generations. 2) A new combinational message bit randomizer is used to change message bits before embedding phase by means of a user inserted key.

The paper is organized as follows:

In the next section, related works on hardware implementation of steganographical algorithms are surveyed. Classic LSB and its weak points and Zhang novel algorithm which can solve classic LSB problems are introduced in section three. Hardware design methodology and related modules are represented in section four. Simulation results and system behavior for different inputs are represented in section five. In section six, distortion analysis is done after message embedding phase and values of mean square error and peak signal to noise ratio are calculated for different numbers of message bits in each pixel and are then compared with a similar work. Finally, section seven draws a conclusion..

2. RELATED WORKS

In an active research domain such as image steganography, different methods and algorithms are presented continuously. In recent years, researchers attempt to increase message bit embedding capacity without dramatic changes in carrier quality and to improve resistant against steganalysis. Using pseudo random number generators is a solution for security improvement. In [11], LSB algorithm is represented with a pseudo random number generator which improves security against attacks. In [12] and [13], random number generator is used as a selector in placing the message bits in image randomly. In these two methods a secret key is shared between sender and receiver.

Different special purpose processors and hardware models are designed for different steganographical algorithms. A special purpose processor for steganography and its FPGA implementation is proposed in [14]. This architecture involves an embedded processor and an SDRAM controller. The embedded processor sends read-write commands to the memory. These commands will be arranged by SDRAM controller and await a memory confirmation. Furthermore, a pseudo random number generator with a secret key is designed for hiding messages in this work.

In [15], a hardware realization of a novel algorithm named "ConText technique" is proposed. This novel algorithm is implemented on Altera Cyclone II FPGA. Context technique uses noisy regions of image with dramatic changes in gray level. Selection process of these regions needs complicated calculations and so using an FPGA can increase the speed of calculations.

A novel hardware model for LSB message hiding method is proposed in [5]. Timing analyses of hardware model shows that the embedding and extraction processes are accomplished 740 times faster than software model. Also, the speed of the software algorithm entirely depends on the run time environment of the process. But, the response time of hardware is always constant for a given size of the cover image. This can be an advantage when we try to interface this module with other systems due to predictability of timing.

3. CLASSIC LSB AND ZHANG ALGORITHM

As mentioned earlier, in LSB method, least significant bits of the cover image are exchanged with message bits. To increase the capacity of message bits insertion in a cover image, each pixel can carry more than one message bit. For security considerations, pixels can be selected by a pseudo-random number generator. In this method, pixels are selected randomly such that message bits are situated all over the image which will increase the resistance against visual analysis.

These classic LSB steganography methods have a common weak point. The sample value changes asymmetrically. When the LSB of cover image sample value is equal to the message bit, no change should be made. Otherwise, we need to change the value 2n to 2n+1 or 2n+1 to 2n. But the changes from 2n to 2n-1 or from 2n+1 to 2n+2 will never emerge. In a grayscale 8-bit bmp image, each pixel is presented by a value between 0 and 255. For even numbers, if the message bit is equal to zero, no change is made while if the message bit is equal to 1, a change is made from value 2n to 2n+1. For odd valued pixels, message bit equal to 1 causes no change, while message bit equal to zero results in a change from 2n+1 to 2n. Steganalysis can take advantage of such asymmetry [4].

Zhang in [5] represented a novel LSB algorithm which can eliminate this asymmetry. Here, an 8-bit gray-scale bmp image is selected as the cover medium with the assumption that the embedded message is a random data.

3.1 Embedding Process

Consider $S = < x_0, x_1, \dots , x_n >$ be the set of pixels of an image which is selected by a pseudo-random number generator. Pseudo random number generator produces random numbers according to the value of seed (stego-key).

x is the gray value of each pixel. *n* is determined by the size of embedded message and the number of LSB bits in each pixel which can be used to embed messages. It can be calculated by:

$$n = \frac{k}{m} \quad (1)$$

Where *k* is the length of bit stream of embedded message, and *m* is the number of bits used to embed messages in each pixel. The bit stream of embedded message is divided into bit segment of *m* bit length and denoted with $E = < e_1, e_2, \dots , e_n >$. $e \in \{0, 1 ,\dots, 2^{m-1}\}$. Defining $LSB_m(x)$ to be the function to get the *m* bit LSB value from the *x*, we embed a message as follows:

$$\text{For } i = 1, 2, \dots, n$$

$$x_i = x_i + e_i - (LSB_m(x_{i-1}) + LSB_m(x_i)) \bmod 2^m$$

$$\text{if } x_i > 255 \text{ then}$$

$$x_i = x_i - 2^m;$$

$$\text{if } x_i < 0 \text{ then}$$

$$x_i = x_i + 2^m$$

3.2 Extraction Process

The same stego-key must be used to generate the pseudo-random number. Selecting pixels according to the pseudo-random number to construct $S = < x_0, x_1, \dots, x_n >$, the message can be extracted as follows:

$$For\ i = 1, 2, \ldots, n$$

$$e_i = (LSB_m(x_{i-1}) + LSB_m(x_i)) Mod 2^m ;$$

$E = < e_1, e_2, \ldots, e_n >$ can reconstruct the message [5].

This new algorithm has all the advantages of classic algorithm while increasing the resistance against statistical analysis. Therefore, hardware realization of this algorithm is useful for high capacity message transformation in a secure way with no suspicion raised. Because of high resistance of this algorithm against statistical attack, invaders cannot realize the existence of messages in image.

In the next section, we propose a hardware implementation for Zhang algorithm. Moreover, we design two novel randomizers in this implementation which improve message transformation security. The scalable pixel interleaver also improves security against visual analysis.

4. HARDWARE DESIGN PROCEDURE AND MODULES

Hardware implementation of Zhang algorithm consists of several modules. Verilog [16] is used as a hardware description language to design modules of this algorithm. There are three modules which construct the main parts of the hardware: message bit randomizer, scalable pixel Interleaver, and embedding module. The main core of randomizers is pseudo random number generator which is introduced in the next section.

4.1 Pseudo-Random Number Generator

Pseudo random number generator (PRNG) prevents invaders to find message bits easily. A secret key can be used as a seed for PRNGs. Using a seed causes PRNGs to generate the same random numbers on receiver side as on the sender side. In this paper, a linear feedback shift register (LFSR) is used as PRNG.

4.1.1 Implementation of LFSR

A LFSR is made of sequential shift-register with combinational feedback logic connected to it which can generate a sequence of binary values in a pseudo-random manner. A design modeled around LFSRs often has both speed and area advantages over a functionally equivalent design that does not use LFSRs.

Feedbacks around an LFSR's shift register are connected to the certain points (taps) of LFSR construction and constitute either XORing or XNORing these taps to provide taps back into the register.

The selection of taps determines how many values can be generated in a given sequence before the sequence is repeated. Certain tap arrangement lead to maximal length sequences of $(2^n - 1)$. These settings are calculated for different lengths of LFSRs and are represented in a reference table [17].

To prevent invaders' access to bit messages, a secret key is defined. Secret key works as a seed for LFSR. The initial content of the register is referred to as a seed.

Figure 1 shows a 4-bit LFSR constructed of D flip-flops and XOR gates in its shift path. The feedbacks are selected from taps 3 and 4 (3,4). The seed affects set and reset inputs of the individual flip-flops of the shift register. When the INT value is zero, the LFSR seed is asynchronously loaded into four flip-flops. Therefore, LFSR generates random numbers starting from an initial value selected by the user. This set of random numbers will be generated at receiver side if the user has this unique seed [18].

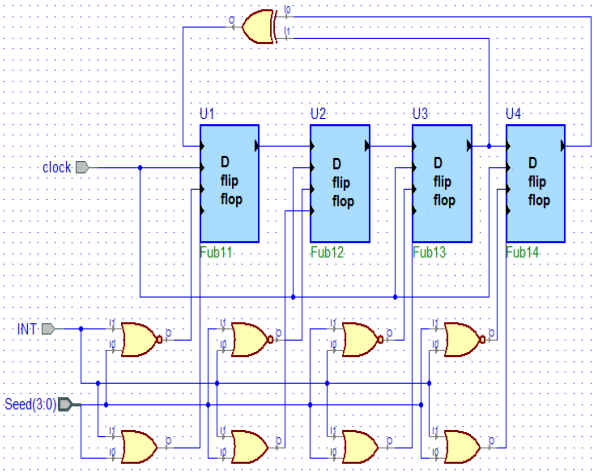

Fig 1: Four-bit LFSR with seed

The generated random numbers can be XORed with message bits and the results will be embedded in a cover image. In receiver side, the message bits will derive from XORing LSB of cover image and the same random bits which had been generated in sender side [19]. In this paper, LFSR is the main core of pixel interleaver and the massage bit randomizer modules.

4.2 Message Bit Randomizer Module

This part of design changes message bits so that if invaders find them, they cannot construct the message without access to secret key and the LFSR architecture. In each clock cycle, message bits must be XORed by a random number. The result of XOR operation will be embedded in pixels. At receiver side, extracted LSBs of each pixel must be XORed with the same random bits to construct the message bits.

A novel combinational randomizer is used for this reason. Figure 2 shows the architecture of this design. Message bits are embedded in least significant bits of a pixel. A 16-bit LFSR with a seed (*seed2* input) generates random values. As shown in Figure 2, for one, two, three and four message bit insertion, four outputs (y_1, y_2, y_3, y_4) are considered. Input *m* value is changed from one to four by user selection and each value of *m* creates one of the outputs.

Module SHIFT16 contains a 16-bit LFSR that randomly generates sixteen bits in each clock cycle. However, these random values will not be XORed with message bits directly. Random bits from SHIFT16 module will be inserted in another module named MYDES.

MYDES module contains several multiplexers and LFSRs. In this module, if m=1, sixteen bits insert a 16 to 1 multiplexer. Selector of this multiplexer is connected to a 4-bit LFSR. Therefore, multiplexer selects one of these sixteen inputs as output, randomly. The output will be XORed by one message bit. This process is repeated for every bit insertion. Figure 3 shows top level view of MYDES module which contains LFSR and multiplexer for one bit insertion.

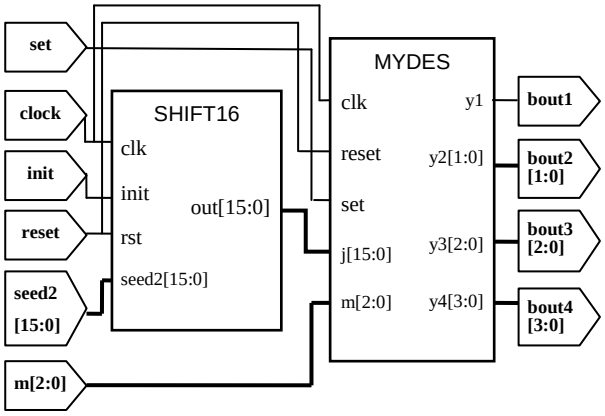

Fig 2: Top level view of message bit combinational randomizer

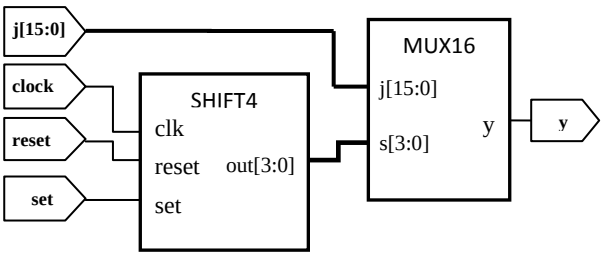

Fig 3: Top level view of MYDES module

If user input is m>1, MYDES module selects different multiplexers and LFSRs lengths.

For example, two 8 to 1 multiplexers with a 3-bit LFSR as selector are designed for two bit insertion. In each clock, sixteen bits from SHIFT16 are transmitted to input of MYDES module and will be situated on inputs of two multiplexers. 3-bit LFSR creates the address for selector of these two multiplexers. Therefore, two bits are selected randomly in each clock cycle.

For three bit insertion, one 8 to 1 multiplexer and two 4 to 1 multiplexers with a 3-bit and a 2-bit LFSRs are considered.

Four 4 to 1 multiplexers and four 2-bit LFSRs are considered in four bit insertion. All these designs are embedded in MYDES module but Figure 3 shows MYDES module for just one bit insertion to be brief.

4.3 Scalable Pixel Interleaver Module

Pixel interleaver selects pixels from memory in a random manner such that the message is embedded all over the image randomly. This will increase resistance against visual analyses and attacks. If message embedding is started from first pixel and is continued sequentially, changes can be recognized by visual analysis, but if messages are embedded in a random manner, visual analysis will be difficult. In this design, a pixel is extracted from RAM memory by "read" command in each clock period. Output of the pixel interleaver module is connected to address input of the RAM. Therefore, pixels will be selected via a random number which is generated by pixel interleaver.

After message embedding phase, pixels are sent back to pixel interleaver to be transformed to their primary address in RAM memory by "write" command. Figure 4 shows top level module of this interleaver.

Role of inputs and outputs are as follows:

- *N* and *M*: inputs that represent width and length of image. *N* multiply *M* shows the total number of pixels in each image.
- *Pixel_in*: input that contains the pixels selected from RAM.
- *Out*: an 8-bit output that transforms the pixel to embedding module for message embedding.
- *Seed1*: 18-bit input of secret key.
- *Steg_pixel*: pixel input which is connected to output of embedding module. Pixel will be sent back to interleaver module via this input.
- *Steg_out*: pixels will be sent back to their primary place on RAM memory via this output.
- *Address*: output that is connected to address input of RAM. A random address number will be applied to RAM memory via this output.

The main core of pixel interleaver is an 18-bit LFSR that can randomize 2^{18}-1 pixels with secret key input (*seed1*). However, generation of all these numbers for a small size image is a time consuming process because some of these generated numbers are larger than the total number of image pixels. Therefore, using an LFSR with lower length has more advantages in comparison with an 18-bit LFSR for all image sizes.

The scalable pixel interleaver which is implemented in this design can change its LFSR length from 6 to 18 bits according to the size of image. This can be done by buffer gates with additional control signals. These gates are used when a signal is to be driven only when the control signal is asserted [16]. Therefore, buffers can be used to connect or disconnect lines between D flip-flops and feedback taps and build diverse lengths of LFSRs.

Firstly, the total number of pixels is calculated by multiplying width and length of image by a multiplier. Then, the length of LFSR is selected according to the total number of pixels. Multiplication result is applied to LFSR and a comparator selects the length of LFSR in comparison with the number of pixels. Redundant D flip-flops and XOR feedbacks of LFSR are then deactivated. Therefore, an LFSR with an appropriate length is formed. For example, if the length of 18-bit LFSR changes to twelve, it works like a 12-bit LFSR and twelve D flip-flops with their feedback connections are activated and six most significant bits become zero.

The generated numbers are prepared to use as address inputs of RAM memory.

LFSR feedbacks are selected from certain taps (feedback connections) that can generate numbers in full length.

As mentioned earlier, taps must be selected such that an LFSR with length *n*, generates 2^n-1 numbers. This can be achieved by means of a reference table. Otherwise, LFSR cannot generate maximal length sequences of (2^n - 1) and some numbers will be discarded. Therefore, system cannot use all the pixels for embedding. For example, one of the tap selections for full length number generation of 18-bit LFSR is (7,18) [17]. It means that feedbacks are selected from output of 7^{th} and 18^{th} D flip-flops.

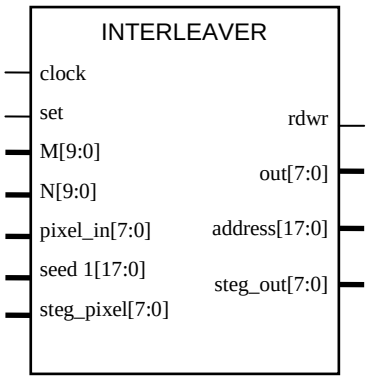

Fig 4: Scalable pixel interleaver module

4.4 Embedding Module

Zhang LSB algorithm needs continuous calculations. For one bit insertion, least significant bits of two adjacent pixels are combined using modulo 2 addition. Result of pervious stage will be combined by third pixel LSB. Modulo 2 addition and this process will be continued for all pixels. Embedding module for Zhang algorithm implementation is described in this section. One to four message bit(s) can be embedded in each pixel by user command. Top level module of this block is represented in Figure 5.

Role of inputs and outputs are as follows:

- *Pixel_in*: 8-bit input that delivers pixels to embedding module in each clock.
- *Stegset*: if stegset input turns into 1, steganography process is enabled.
- *In_message1* to *in_message4*: one to four bit(s) input for embedding one to four bit(s) in each pixel.
- *m*: a three bit input that selects the number of embedding bits in each pixel. Number of exchanging bits is selected by user. Therefore, user can adjust the number of embedding message bits according to image quality.
- *Out*: 8-bit output which transforms the stego-pixel to interleaver after embedding process.

Hardware design of the embedding module consists of three major phases.

Phase1: in each clock cycle, if *stegset*=1, an 8-bit pixel is transferred from interleaver to embedding module via *Pixel_in* input. Input *m* selects the number of embedding message bits in each pixel.

Phase2: *m* least significant bits of the first pixel input must be saved in a memory because they will be used in next clock cycle. For second pixel input, *m* least significant bits of this pixel must be XORed by memory bits and the result will be subtracted (via a subtractor) from pixel input value and will be added (via an adder) to message bit value.

Phase 3: a comparator will check the result of pervious phase. If the result is negative, value 2^m is added to it and if the result becomes more than 255, value 2^m is subtracted from it. Finally, the new pixel value will be sent to output and a copy of LSB of final result will be saved in memory to be used in next clock cycle. For other pixel inputs, this cycle will be repeated and the result will be represented from output *out.*

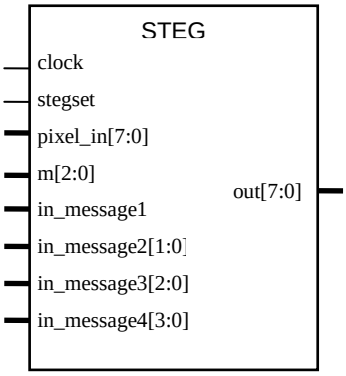

Fig 5: Embedding module

4.5 Top Level View of Steganography System

Embedding module is connected to pixel interleaver and message bit randomizer to construct the main steganography system. Top level view of this system is represented in Figure 6.

In each clock cycle, pixels are selected via interleaver module by first user key (*seed1*) from an external RAM. Pixels are transferred to STEG module for message embedding. In other side, random numbers are generated by message bit randomizer module (LFSNEW) with second user key (*seed2*) and are combined with message bits by XOR operation. XOR1 module consists of four inputs. One of these inputs is created according to the number of embedding message bits (*m*). In each clock, random numbers which are generated from LFSNEW module are XORed by message bits and will be sent to STEG module for embedding in pixels.

After embedding messages, pixels will be sent back to interleaver module to be situated in their primary position in RAM memory by write command. Synplify premier 9.6.1 is used to synthesize the verilog codes and generate these modules.

5. HARDWARE SIMULATION RESULT

In section four, main structure of system and roles of inputs and outputs in each block were introduced. Verilog codes of this design are simulated and compiled in Active-HDL 8.2 simulator. Inputs and outputs operations are tested and simulated and results of simulations are mentioned in this section.

5.1 Embedding Module Results

Embedding module consists of inputs and outputs which are connected to other parts of steganography system. In each clock, one pixel is inserted in this module via *pixel_in* input. In an 8-bit grayscale bmp image, each pixel has a value from 0 to 255 in decimal scale.

Figure 7 shows embedding module inputs and output for one bit insertion. Input *m* selects the number of bits embedded in each pixel and can be a number between 1 and 4.

The special feature of this new algorithm is symmetrical change in pixel values. In classic LSB, pixels with 2n value cannot flip to 2n-1 and pixels with 2n+1 value cannot flip to 2n+2. This asymmetry is eliminated by Zhang's algorithm. As shown in Figure 7, pixel with gray value of 98 has flipped to value 97 and in another pixel, change is made from 199 to 200

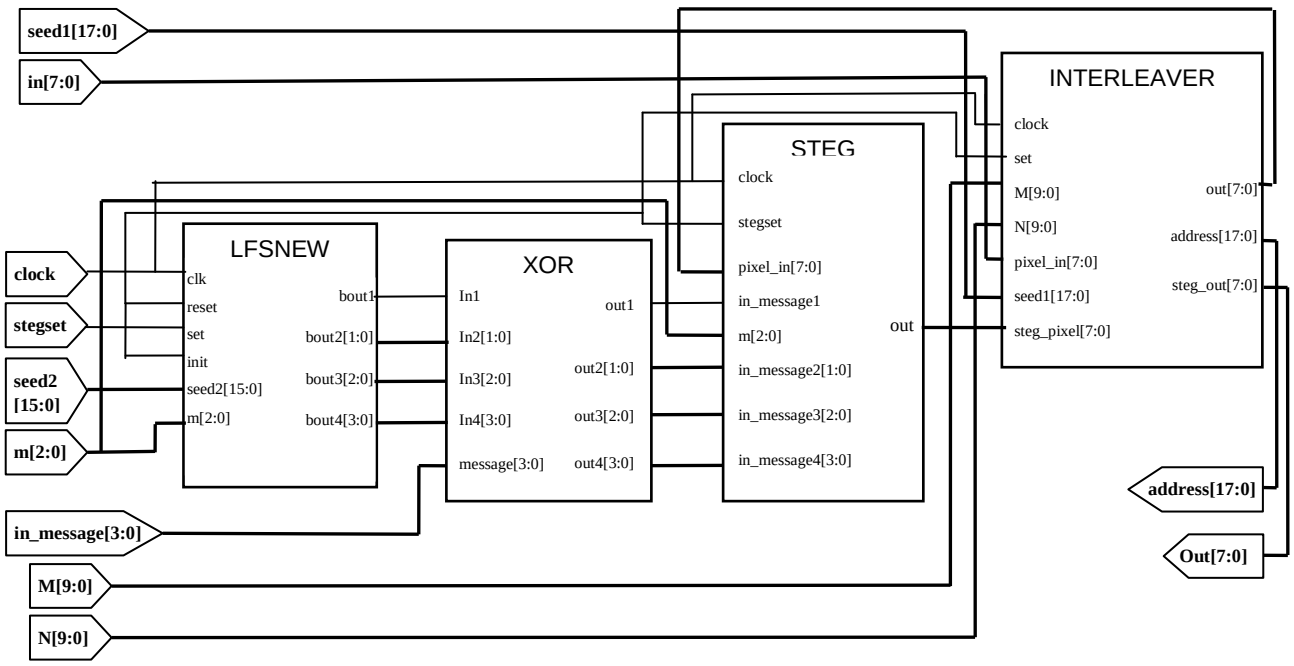

Fig 6: Top level view of steganography system

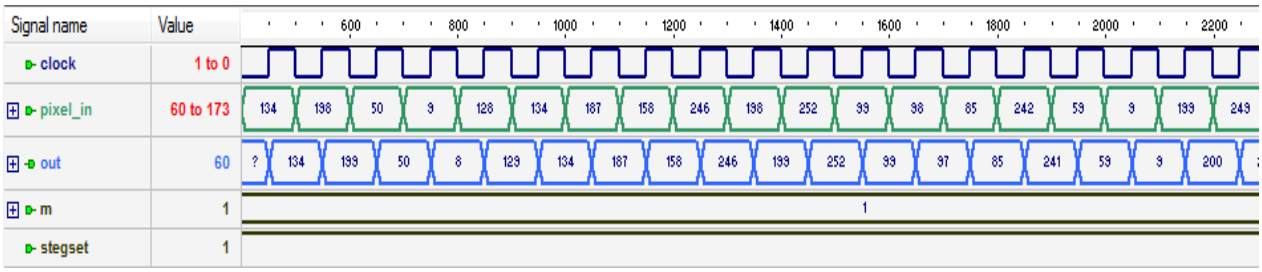

Fig 7: Waveforms of embedding module

and therefore hardware implementation shows proper response.

6. DISTORTION ANALYSIS

The images can be distorted in embedding process because of changing pixel bits.

Distortion is measured by means of two parameters namely, Mean Square Error (MSE) and Peak Signal to Noise Ratio (PSNR).

MSE can be calculated by (2) [20]:

$$MSE = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(X_{ij} - Y_{ij})^2 \quad (2)$$

M denotes the total number of pixels in the horizontal dimensions of the image. N shows the total number of pixels in vertical axis. Therefore, MN denotes the total number of pixels.

X_{ij} represents the gray value of pixels in the original image and Y_{ij}, represents the gray values of the stego-image. Lower MSE value means higher image quality.

The PSNR is calculated using (3) [20], [21]:

$$PSNR = 10\log_{10}\left\{\frac{I^2_{\max}}{MSE}\right\}dB \quad (3)$$

I_{max} is the maximum intensity value of each pixel which is equal to 255 for 8 bit gray scale images. Higher value of PSNR leads to better image quality.

Results of steganography for Lena and Baboon digital images is represented in Figures 8 and 9 respectively and full embedding capacity is considered for m=1 to 4.

The size of these images is 256*256 pixels. As shown here, message embedding is done with no dramatic changes in image quality.

Tables 1 and 2 show MSE and PSNR for various values of *m* in Lena and Baboon images. Results are compared with Amirthrajan work [5]. These tables show comparable results of MSE and PSNR with Amirtharajan work.

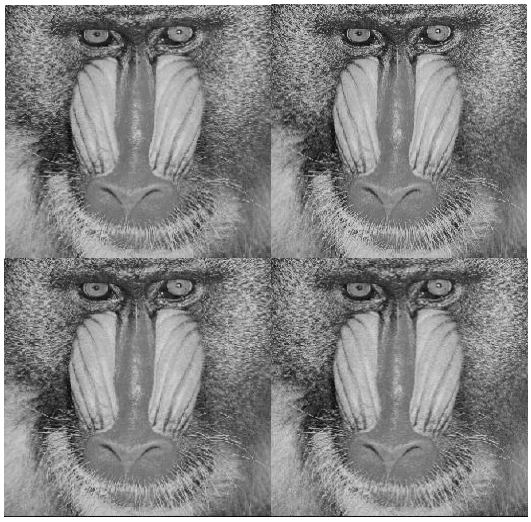

Fig 8: Steganography results for Baboon image. ((a) left-up), ((b) right-up)), ((c) left-down)), ((d), right-down)) for m=1, 2, 3 and 4 for full embedding capacity

Fig 9: Steganography results for Lena image. ((a) left-up), ((b) right-up)), ((c) left-down)), ((d), right-down)) for m=1, 2, 3 and 4 for full embedding capacity

7. CONCLUSION

Zhang LSB algorithm delivers several advantages. This algorithm can resist statistical analyses and enables the usage of LSB in a secure way. Therefore, it is more appropriate for hardware implementation. Furthermore, by using our pixel interleaver and message bit randomizer, protection against attacks is improved. User can select how many bits must be embedded in each pixel according to image quality and length of message. If the length of message is low, user can select one or two message bits to be embedded in each pixel. Otherwise, if the capacity is important and user needs to send more message bits, it can be satisfied by selection of more embedded bits in each pixel. This will allow the image quality and message capacity to be adjusted according to the user needs. In this design, two separate keys are used to improve security. MSE and PSNR calculation shows comparable results with similar work on classic LSB.

TABLE 1. Distortion Analysis of Baboon image and comparison with R.Amirtharajan work

Baboon	**No. Of Bits (m)**	**1**	**2**	**3**	**4**
Zhang LSB Algorithm Results	***MSE (no unit)***	0.4956	2.2755	10.032	38.1153
	PSNR dB	51.1796	44.56	38.331	32.6758
R. Amirtharajan Results	***MSE (no unit)***	0.5031	2.5699	10.3083	39.4619
	PSNR dB	51.1134	44.0316	37.9989	32.169

TABLE 2. Distortion Analysis of Lena image and comparison with R.Amirtharajan work

Lena	**No. Of Bits(m)**	**1**	**2**	**3**	**4**
Zhang LSB Algorithm Results	***MSE (no unit)***	0.4913	2.3244	10.2634	39.9374
	PSNR dB	51.217	44.4677	36.0179	32.6979
R. Amirtharajan Results	***MSE (no unit)***	0.5012	2.5761	10.2769	40.3637
	PSNR dB	51.131	44.021	35.7189	32.0708

8. REFERENCES

[1] Morkel, T., Eloff, JHP., and Olivier, MS., June/July 2005, "An Overview of Image Steganography," in HS Venter, JHP Eloff, L Labuschagne and MM Eloff (eds), Proceedings of the Fifth Annual Information Security South Africa Conference (ISSA2005), Sandton, South Africa.

[2] Bandyopadhyay, S.k., Bhattacharyya, D., Ganguly, D., Mukherjee, S., Das, D., 2008, "A Tutorial Review on Steganography" IC3, pp 105-114.

[3] Fridrich, J., Goljan, M., and Du, R., October-November 2001, "Detecting LSB Steganography in Color and Gray-Scale Images", Magazine of IEEE Multimedia Special Issue on Security, pp. 22-28.

[4] Zhang, H.J., TANG, H.J., August 2007, "a Novel Image Steganography Algorithm Against Statistical Analysis" Proceedings of the Sixth International Conference on Machine Learning and Cybernetics, Hong Kong, IEEE pp 3884-3888.

[5] Amirtharajan, R., Bosco Balaguru, R.J., Ganesan, V., July 2010 "Design and Analysis of Prototype Hardware for Secret Sharing Using 2-D Image Processing", International Journal of Computer Applications (0975 – 8887), Volume 4 – No.4, pp 17-22.

[6] Leung, H.Y., Cheng, L.M., Cheng, L.L., Chan, C.K., 2007, "Hardware Realization of Steganographic Techniques," Third International Conference on International Information Hiding and Multimedia Signal Processing (IIH-MSP 2007), iih-msp, vol. 1, pp.279-282.

[7] Li, S.L., Leung, K.C., Cheng, L.M., Chan, C.K., 2006, "A novel image hiding scheme based on block difference", Pattern Recognition 39, pp 1168-1176.

[8] Li, S.L., Leung, K.C., Cheng, L.M., Chan, C.K. "Data Hiding in Images by Adaptive LSB Substitution Based on the Pixel-Value Differencing" , icicic 2006, IS16-005.

[9] Chang, C.C., Lin, M.H., Hu, Y.-C., 2002, "A fast and secure image hiding scheme based on LSB substitution", Int. Journal of Pattern Recognit. And Artif. Intell. 16 (4), pp 399-416.

[10] Chan, C.K., Cheng, L.M., 2004, "Hiding data in images by simple LSB substitution", Pattern Recognition 37, pp 469–474.

[11] Wang, H., Wang, S, October 2004, "Cyber warfare: Steganography vs. Steganalysis", Communications of the ACM, 47:10.

[12] Sharp, T., 2001, "An implementation of key-based digital signal steganography", Proc. 4th International Workshop on Information Hiding, Springer LNCS, vol. 2137, pp.13-26.

[13] Rodrigues, J.M., Rios, J.R. and Puech, W., April 2004, " SSB-4 System of Steganography using Bit 4", Proc. 5th International Workshop on Image Analysis for Multimedia Interactive Services, (WIAMIS'04), Lisboa, Portugal.

[14] Farouk, H.A., Saeb, M., 2003, "An FPGA Implementation of a Special Purpose Processor for Steganography" 2003 IEEE International Conference on Field-Programmable Technology (FPT), Proceedings, pp 395-398.

[15] Gómez-Hernández, E., Feregrino-Uribe, C., Cumplido, R., 2008, "FPGA Hardware Architecture of the Steganographic ConText Technique" 18th International Conference on Electronics, Communications and Computers IEEE , pp 123-128.

[16] Palnitkar, S., 2003, "Verilg HDL: a Guide to Digital Design and Synthesize" 2nd, Prentice Hall PTR, ISBN 0-13-044911-3.

[17] Douglas, J.S., 1997, "HDL Chip Design" 3rd, Doone Publications, Madison, AL, USA, ISBN 0-9651934-3-8, , pp. 179-187.

[18] Navabi, Z., 2006 "Verilog Digital System Design" 2nd ed, New York: McGraw-Hill, pp. 163-166.

[19] M. Bhat, G., Mustafa, M., A. Parah, S., Ahmad, J., 2010, "Field programmable gate array (FPGA) implementation of novel complex PN-code-generator- based data scrambler and descrambler", Maejo Int. J. Sci. Technol., 4(01),pp. 125-135.

[20] Gibson, J. D., 2000, "Handbook of Image and Video Processing" Academic Press: A Harcourt Science and Technology Company Publication.

[21] Huynh-Thu, Q., Ghanbari, M., 2008 "Scope of Validity of PSNR in Image/Video Quality Assessment", IEEE Electronics Letters Journal, pp 800-801.