Defence Science Journal, Vol. 67, No. 5, September 2017, pp. 551-558, DOI : 10.14429/dsj.67.10177
© 2017, DESIDOC

A FPGA based Steganographic System Implementing a Modern Steganalysis Resistant LSB Algorithm

Kunjan Pathak* and Manu Bansal

Thapar University, Patiala - 147 004, India
**E-mail: pathak.kunjan@gmail.com*

ABSTRACT

Steganography differs from other data hiding techniques because it encodes secret message inside cover object in such a way that transmission of secret message also remains a secret. Widespread usage of digital images, lower computational complexity and better performance makes spatial domain steganographic algorithms well suited for hardware implementation, which are not very frequent. This work tries to implement a modern steganalysis resistant LSB algorithm on FPGA based hardware. The presented work also optimises various operations and elements from original one third probability algorithm with respect to hardware implementation. The target FPGA for the implementation is Xilinx SP605 board (Spartan 6 series XC6SLX45T FPGA). Stego images obtained by the implementation have been thoroughly examined for various qualitative and quantitative aspects, which are found to be at par with original algorithm.

Keywords: Steganography; Data security; Spatial domain algorithm; LSB algorithm; Optimisation for hardware; FPGA implementation

NOMENCLATURE

LSB	Least significant bit
FPGA	Field programmable gate array
TCP-IP	Transmission control protocol-internet protocol
HCF-COM	Histogram characteristic function-centre of mass
DCT	Discrete cosine transform
DFT	Discrete fourier transform
FSM	Finite state machine
MSE	Mean square error
PSNR	Peak signal to noise ratio
JPEG	Joint photographic experts group

1. INTRODUCTION

Steganography, derived from the Greek words stegos, meaning roof or covered and graphia which means writing, is the art and the science of hiding the fact that communication is taking place. The first known application of steganography is as ancient as Greek antiquity. Messages were tattooed on slaves' heads and then their hair were allowed to grow until the message would get covered. In more recent turn of events, US government has made a claim that attackers of 9/11 used steganographic algorithms to communicate via websites and other digital media using internet[1-3].

Steganography has found wide range of applications in data security. Numerous defence organisations have been using one or the other form of steganographic techniques for covert communications. Similar applications may be found in intelligence community. Use of steganography as a tool for copyright protection has been practiced as well. Steganography is also used in medical field, where patients' data is embedded within scanned picture result itself. In this way, confidentiality of the data can be ensured while achieving reduction in transmission time[4].

Steganography remains a promising domain in the field of data security. It has been proved to be as effective as cryptography and watermarking.

1.1 Classification of Steganography based on Cover Media

Any type of digital media including audio, video and image files can be used as a cover file for hiding a secret message. Recent studies also explore possibilities about utilising frames of protocols like TCP-IP as cover media[5]. However, digital images are the most widely used medium for steganography. Wide spread usage, lesser storage size and extensive research work done in the field of image processing makes digital images the most attractive option for cover in steganography.

Image steganography techniques are largely based on either spatial domain modifications or frequency domain modifications.

1.1.1 Spatial Domain Steganography

In spatial domain steganography, image is represented as bit planes. Generally, the bit plane's least significant bits are devoid of any significant information. Alteration in those planes does not affect image information considerably. Thus secret message is embedded in LSB planes. Major advantages of spatial domain methods includes high embedding capacity, high level of imperceptibility and lower computation

Received : 12 July 2016, Revised : 18 May 2017
Accepted : 23 May 2017, Online published : 19 September 2017

complexity[6], which makes these techniques more attractive for any hardware implementation.

However, they are vulnerable to statistical methods based steganalysis attacks.

1.1.2 Frequency Domain Steganography

In frequency domain steganography, cover image is transferred in frequency domains using transformations like DCT, DFT. The data is hidden inside the coefficients obtained by such operations. These methods demonstrate robustness against statistical methods based steganalysis attacks. On the other hand, they lack high embedding rate and involve operations of complex number multiplications repeating for multiple iterations which are computationally expensive[6].

2. MOTIVATION AND RELATED WORK

Spatial domain steganography techniques, like LSB substitution, offer better embedding capacity. The improvement in embedding capacity is achieved using operations which are computationally simpler than those used by frequency domain techniques. However, weakness of such techniques against statistical attacks like chi-square test has motivated development of LSB matching technique[7,8]. When value of cover image pixel LSB and message bit does not match, cover pixel value may be increased or decreased randomly in LSB matching. Thus probability of change for a pixel value is considered as 0.5. Steganalysis methods such as HCF-COM performs very well against LSB matching[9].

A revised version of LSB matching reduces the same to 0.37,[10] resulting in better correlation between cover and stego image. Sarreshtedari[11], *et al.* has proposed a newer algorithm which reduces the probability of change to 1/3. One third probability algorithm shows better resistance against HCF-COM attack while performing better than LSB matching and revised LSB matching[11].

In the field of steganography, hardware implementation using FPGA is not very frequent. There are a few examples of hardware implementation based on LSB replacement method and older spatial domain techniques. Mohd.[12], *et al.* and Farouk[13], *et al*. has implemented traditional LSB substitution algorithm. Laces[14], *et al.* have also made the similar effort. Major weakness of such algorithm against statistical attacks is obvious. Shahadi[15], *et al.* have implemented audio steganography algorithm. But, hardware implementation of a recent LSB based image steganography algorithm, which is resistant against modern steganalysis attacks, has not been accomplished yet. It is noteworthy that LSB based algorithms comes with lower computational complexity making them more suitable for hardware implementation.

Motivation for this work also lies in the fact that most of the operations explained in one third probability algorithm are not optimised or well supported in context of hardware implementation. Purpose of presented work is to modify the algorithm for hardware implementation and then implement the modified algorithm on FPGA. A FSM containing details of hardware implementation has been designed. Performance evaluation of modified algorithm has also been carried out using various quantitative and qualitative parameters.

3. MODIFIED ONE THIRD PROBABILITY ALGORITHM AND SYSTEM DESIGN

Elements and methods from Sarreshtedari[11], *et al.* are mentioned in following sub-section along with proposed modifications.

3.1 Basic Definitions

Minor Change: A minor change in any value x is an addition or a subtraction by one, such that ⌊x/2⌋ remains unchanged. Mathematically, it can be defined as[11]:

$$Minor(x) = floor\left(\frac{x}{2}\right)*4+1-x \tag{1}$$

However, optimised version of the operation in terms of hardware design can be obtained by considering given grey pixel value. If a grey pixel value is an even number, then minor function increments it by one and decrements it by one if number is odd.

Major Change: A major change in any x is an addition or a subtraction by one, such that ⌊x/2⌋ changes. It can be defined mathematically as[11]:

$$Major(x) = floor\left(\frac{x+1}{2}\right)*4+1-x \tag{2}$$

Similar to the minor change function, the major change function can also be optimised. If a grey pixel value is an even number, then minor function decrements it by one and increments the same by one if number is odd.

LSB Function: LSB function for any two grey pixel values can be defined as follows[11]:

$$f(x_i, x_{i+1}) = LSB\left(x_i + floor\left(\frac{x_{i+1}}{2}\right)\right) \tag{3}$$

However, LSB function can also be modified as:

$$f(x_i, x_{i+1}) = LSB\begin{pmatrix} x_i + (1 \text{ place arithmetically} \\ \text{right shifted value of } x_{i+1}) \end{pmatrix} \tag{4}$$

It is noteworthy that an arithmetic (unsigned) right shift by 1 place produces the same result as division by 2 and the floor function combined in Eqn. (3).

To determine operation on specific cover pixel group, result vector is defined as:

$$r = [r_i,\ r_{i+1}, r_{i+2}] = xor\begin{pmatrix} [m_i, m_{i+1}, m_{i+2}], \\ [f(x_i, x_{i+1}), f(x_{i+1}, x_{i+2}), f(x_{i+2}, x_i)] \end{pmatrix} \tag{5}$$

Optimisation of all three functions, which are performed repeatably for each cover pixel group, ensures reduced hardware utilisation by eliminating computationally complex operations like floor function, division, multiplication as well as excess number of additions and subtractions.

Use of floor function and division operation may necessitate final resultant value or intermediate variable to be a floating point number, which will require separate class support for required precision values. It may lead to 64 bit registers for each pixel floating point value instead of 8 bit unsigned integer value. Additionally, limited or none synthesis support is available for floating point type. Major and minor functions can simply be implemented using a multiplexer according to directive prescribed in this work. Multiplexers are readily available as an atomic unit in most FPGAs.

3.2 Embedding Steps

Step 1 Initialisation

According to the algorithm given by Sarreshtedari[11], *et al.*, the cover image x is copied into the stego image y. Pixels having extreme values of $N-1$ and 0 are decreased and increased by one, correspondingly. The $N \times N$ matrix C, where $N-1$ is the upper bound for permissible pixel values, is defined and all of the entries of C are set to zero.

However, synthesis of two dimensional array consisting signed integers has not been supported by majority of synthesis tools which generates RTL level net list. In this work, two dimensional array has been substituted by two single dimensional arrays. Control signals for each array determines to register changes in given pixel values to either lower triangle array or upper triangle array as shown in Fig.1. Each, lower triangular array *lt* and upper triangular array *ut* both have ($N*N$) no. of elements. All of these elements are also initialised with zero value.

Step 2 First Scan

The cover image is divided into groups of three pixels (x_i, x_{i+1}, x_{i+2}) in each group, which are called embedding units. (x_i, x_{i+1}, x_{i+2}) and (y_i, y_{i+1}, y_{i+2}) have similar values in the beginning.

Three secret bits (m_i, m_{i+1}, m_{i+2}) are fetched from the secret message corresponding to each embedding unit. (y_i, y_{i+1}, y_{i+2}) would contain values for corresponding pixels in the stego image after embedding operation.

The result for matching vector is calculated as given in Eqn. (5). This calculation is repeated until result vectors are derived for all embedding units. Transformations in embedding units are performed according to the Table 2.

If there is a change in $C(x,y)$ or $C(y,x)$ element according to the mandatory phase calculations as described in original one third probability algorithm, corresponding change in lower or upper triangle matrix from modified algorithm would occur as follows:

(a) For an entry in C matrix $C(a,b)$, if $a>b$, then corresponding change would happen in lower triangular matrix.

(b) For an entry in C matrix $C(a,b)$, if $a<b$, then corresponding change would happen in upper triangular matrix.

(c) For an entry in C matrix $C(a,b)$, if $a=b$, then corresponding change happens in diagonal elements of C matrix. It would correspond to no change case when r=[0, 0, 0]. So no action is needed.

Step 3 Mandatory Phase

For each embedding unit in cover image, cover image pixel values and corresponding message bits are fetched. The pixel values are transformed according to the following 8 cases as per mentioned in Table 1.

However modifications in change matrix C in original algorithm has been given as:

$$C(y_a, x_a) = C(y_a, x_a) + 1 \tag{6}$$

Table 1. Operations in mandatory phase of embedding including resultant stego pixels

r vector	Resultant stego pixel after change
[1, 0, 0]	$y_i = Minor(x_i)$
[0, 1, 0]	$y_{i+1} = Minor(x_{i+1})$
[1, 1, 0]	$y_{i+1} = Major(x_{i+1})$
[0, 0, 1]	$y_{i+2} = Minor(x_{i+2})$
[1, 0, 1]	$y_i = Major(x_i)$
[0, 1,1]	$y_{i+2} = Major(x_{i+2})$
[0, 0,0]	No Change
[1,1,1]	Save to the random list

$$C(x_a, y_a) = C(x_a, y_a) - 1 \tag{7}$$

where a equals to i, i+1, i+2 is according to r vector as mentioned in Table 1.

For example if r= [1, 0, 0] then value of $a=i$. Thus changes would occur in $C(y_i, x_i)$ and $C(x_i, y_i)$ as explained by Eqns. (6) and (7).

However changes in lower and upper triangular arrays follows relation between resultant stego pixel y and cover pixel x.

If $y>x$ then,

$$lt(y_a * x_a) = lt(y_a * x_a) + 1 \tag{8}$$

$$ut(x_a * y_a) = ut(x_a * y_a) - 1 \tag{9}$$

Similarly, for $x>y$,

$$lt(y_a * x_a) = lt(y_a * x_a) - 1 \tag{10}$$

$$ut(x_a * y_a) = ut(x_a * y_a) - 1 \tag{11}$$

where a follows similar notation as per Eqns. (8) and (9).

It is worth mentioning that no changes occur in either original change matrix C or derived lower and upper triangular arrays *lt* or *ut* when r = [0,0,0] or r = [1,1,1] during this phase.

Table 2. Hardware utilisation in Xilinx Spartan 6 series XC6SLX45T FPGA

Slice logic utilisation	Used	Available	Utilisation (%)
Number of slice registers	214	54,576	1
Number of slice LUTs	322	27,288	1
Number of slice LUT used as memory	0	6,408	0
Number of occupied slices	160	6,822	2
Number of bonded IOBs	171	296	57
Number of RAMB16BWERs	112	116	96
Number of DSP48A1s	9	58	15
Average fan-out of non-clock nets	5.52		
Number of LUT flip flop pairs used	392		

Step 4 Random Phase

In random phase, we just consider the case when result vector *r* is [1,1,1]. Three different sets of changes can be performed at this stage (i.e. one pixel undergoes major change, one pixel undergoes minor change and the third one remains unchanged.). We calculate 'score' for each of these three change sets. The score is calculated by adding signs from corresponding entries of the *lt* and *ut* arrays according to two possible alterations in original algorithm. For example, if pixel values have depth of 2 bits (values 0,1,2,3) and entries of the change matrix are as below just after the first phase of embedding. (Note that for level 0 to 3, matrix row and column numberings are considered as 1 to 4):

$$C = \begin{bmatrix} 0 & 4 & 0 & 0 \\ -4 & 0 & 1 & 0 \\ 0 & -2 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix} \tag{12}$$

Corresponding entries in lower triangular and upper triangular arrays, excluding diagonal elements, will be

$$\begin{aligned} lt[1{:}15] &= [0\ \ C(2*1)\ C(3*1)\ c(4*1)\ 0\ 0\ C(3*2) \\ &\quad 0\ C(4*2)\ 0\ 0\ 0\ C(4*3)\ 0\ 0\ 0\ ] \\ &= [0\ \text{-}4\ 0\ 0\ 0\ \text{-}2\ 0\ 0\ 0\ 0\ 0\ \text{-}1\ 0\ 0\ 0] \end{aligned} \tag{13}$$

$$\begin{aligned} ut[1{:}15] &= [0\ \ C(1*2)\ C(1*3)\ c(1*4)\ 0\ 0\ C(2*3) \\ &\quad 0\ C(2*4)\ 0\ 0\ 0\ C(3*4)\ 0\ 0\ 0\ ] \\ &= [0\ 4\ 0\ 0\ 0\ 2\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0] \end{aligned} \tag{14}$$

For embedding unit *x*= [1, 1, 2] and *m*= [0, 1, 1]

r = [0, 1, 1] *x* or LSB ([(1 + LSB (⌊1/2⌋)), (1 + LSB (⌊2/2⌋)), (2 + LSB ⌊1/2⌋)]) = [1, 1, 1]

1. Major($x1$):1 to 2, Minor($x2$):1 to 0 ⇒ Score(1) = sign(ut(2*3)) + sign(lt(2*1)) = 0
2. Major(x2):1 to 2, Minor (x3)) :2 to 3 ⇒ Score(2) = sign(ut(2*3)) + sign(ut(3*4)) = 2
3. Major(x3):2 to 1, Minor (x1):1 to 0 ⇒ Score(3) = sign(lt(3*4)) + sign(lt(2*1)) = −2

Thus the best decision is second case.

As grey level '1' and '2' are facing changes here, entry (3,2) and (4,3) are incremented and entry (2,3) and (3,4) are decremented by one in *C* matrix.

$$C = \begin{bmatrix} 0 & 4 & 0 & 0 \\ -4 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{15}$$

Corresponding changes in lt and ut arrays would be:

$$lt = [0\ \text{-}4\ 0\ 0\ 0\ \text{-}1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0] \tag{16}$$

$$ut = [0\ 4\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0] \tag{17}$$

Thus all the pixel sets, which had been stored in random change list, undergo above-mentioned changes and generation of stego image is finally accomplished.

3.3 Extraction Algorithm

The extraction process is similar to the original algorithm but modified form of LSB function would be utilised. The stego image is divided into embedded units, each consisting of three pixels (y_i, y_{i+1}, y_{i+2}).Three message bits are extracted from each embedded unit following the equation of LSB function. The process is continued until all embedded bits have not undergone extraction procedure.

$$m_i = f\left(y_{i,}y_{i+1}\right) = LSB\left(y_i + floor\left(\frac{y_{i+1}}{2}\right)\right) \tag{18}$$

$$m_{i+1} = f\left(y_{i+1,}y_{i+2}\right) = LSB\left(y_{i+1} + floor\left(\frac{y_{i+2}}{2}\right)\right) \tag{19}$$

$$m_{i+2} = f\left(y_{i+2,}y_i\right) = LSB\left(y_{i+2} + floor\left(\frac{y_i}{2}\right)\right) \tag{20}$$

4. METHODOLOGY AND IMPLEMENTATION

4.1 System Architecture

Figure 1 shows top level system architecture of proposed steganographic system. The block RAMs contains cover image *x* and secret message *m*. A monochrome image with only two levels, black and white (binary 0 and 1 respectively) has been chosen as secret message. Due to the constraints on target FPGA resources, image size is set at 128*128 pixels of and secret message has also been chosen to be of 128*128 bits. Each cover pixel contains 256 grey levels and has storage size of 8 bits. In this way, secret message (128*128 bits) is comprised of 1/8th portion of original cover image (128*128*8 bits). Thus, secret message embedding in the cover image has been done at its full capacity of 1 bit per pixel.

To perform the function of lower and upper triangular arrays, which assists in all over histogram compensating characteristics of one third probability algorithm, two Block RAMs containing 256*256 no. of signed numbers have been allocated. They are initialised as all zeroes values for each element.

Resultant stego image *y* has also the size of 128*128 pixel having 8 bits depth per pixel. It has also been assigned a block RAM and is initialised with all zeroes.

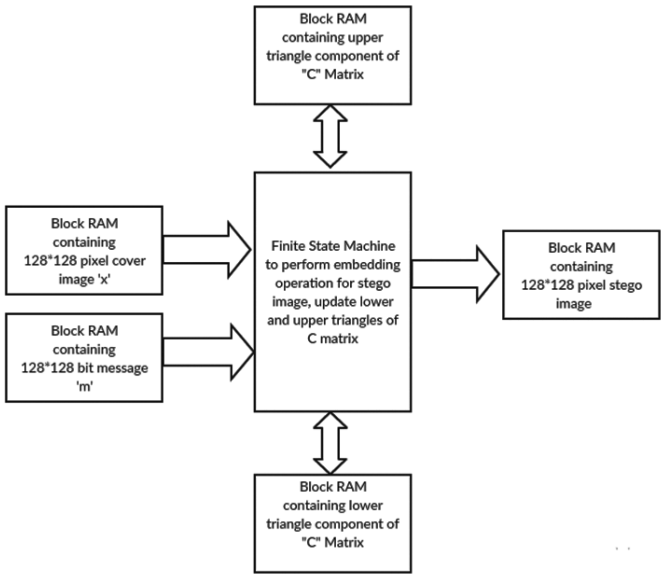

Block RAM containing upper triangle component of "C" Matrix
Block RAM containing 128*128 pixel cover image 'x'
Block RAM containing 128*128 bit message 'm'
Finite State Machine to perform embedding operation for stego image, update lower and upper triangles of C matrix
Block RAM containing 128*128 pixel stego image
Block RAM containing lower triangle component of "C" Matrix

Figure 1. Top level block diagram of steganographic system based on modified algorithm.

4.2 Finite State Machine for Final Hardware Design

The Finite State Machine, which is responsible for implementing embedding process, is explained in this sub-section. All transitions take place on the positive edge of clock. Functionality of the states is described in Fig. 2.

1. Reset: Entered at power on or when reset input is asserted. Address for all memories are initialised at 0.
2. State S2 to S6 are responsible for executing mandatory phase. They perform transformations in stego pixel values and changes in *lt* and *ut* matrices. Once all units have undergone mandatory phase, then only FSM enters random phase.
3. State S7 to S14 are responsible for random phase. Reading entries from *ut* and *lt* matrices, calculating score, decision of best random change has been carried out in these states. Following the completion of random phase embedding in all units, the FSM enters S15, which suggests completion of embedding phase execution.

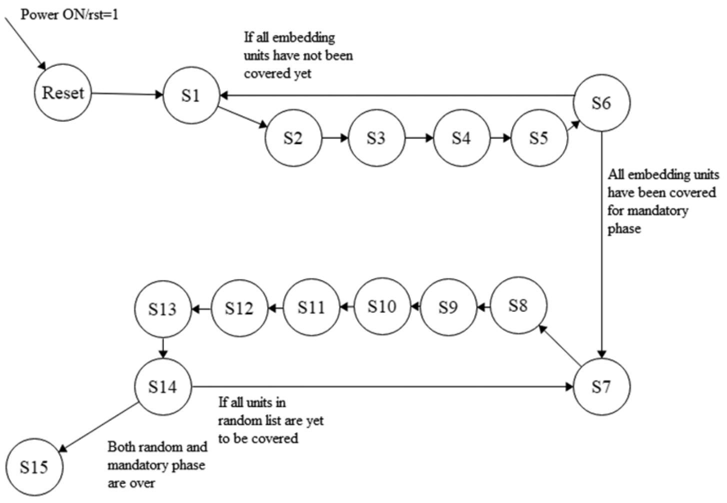

Power ON/rst=1
If all embedding units have not been covered yet
Reset
S1
S2
S3
S4
S5
S6
All embedding units have been covered for mandatory phase
S13
S12
S11
S10
S9
S8
S14
S7
If all units in random list are yet to be covered
Both random and mandatory phase are over
S15

Figure 2. Finite State Machine designed for embedding operation.

4.3 Methods and Tools used for Implementation

A HDL code for proposed design has been developed in Verilog HDL. Xilinx ISE tools have been used for synthesis, placement and routing operations. Final Target for the design is SP605 board, which includes Xilinx Spartan 6 Family FPGA.

Resultant stego image produced by FPGA has been converted to binary files. The files are imported into MATLAB for the quantitative analysis as well as image display and histogram calculations. Quantitative analysis procedures have been carried out using various functions in MATLAB.

4.4 Definition of Various Basic Parameters Employed for Qualitative Analysis

Mean Squared Error (MSE)

It is computed by performing pixel by pixel squared differences of the cover and stego images. The computation can be expressed as follows:

$$MSE = \frac{1}{m*n}\sum_m\sum_n\left(x_{mn} - y_{mn}\right)^2 \tag{21}$$

m number of rows in cover image

n number of columns in cover image

x_{mn} pixel value from cover image

y_{mn} pixel value from stego image

Higher value of MSE indicates greater dissimilarity between cover image and stego images.

Peak Signal to Noise Ratio (PSNR)

PSNR indicates subjective quality of the stego image in comparison with the cover image. It is measured in decibels. The higher value of PSNR points to better quality of stego image. PSNR is computed using the following Eqn (22).

$$PSNR = 10*\log_{10}\frac{255^2}{MSE} \tag{22}$$

where 255 is maximum grey level in an 8 bit pixel.

Correlation Coefficient

It reflects statistical similarity between resultant stego and cover image. It should be 1 ideally. It is calculated by given formulae.

$$r = \frac{\sum_m\sum_n\left(x_{mn} - \bar{x}\right)\left(y_{mn} - \bar{y}\right)}{\sqrt{\left(\sum_m\sum_n\left(x_{mn} - \bar{x}\right)\right)^2} * \sqrt{\left(\sum_m\sum_n\left(y_{mn} - \bar{y}\right)\right)^2}} \tag{23}$$

Here, x and y are cover and stego images, respectively with m rows and n columns. $\bar{x}$ and $\bar{y}$ are arithmetic means for both images.

Entropy of Image

Image entropy is a quantity which is used to describe the `busyness' of an image, i.e. the minimum amount of information which must be coded by any image processing algorithm. Ideally entropy for both cover and stego image should be equal and does not change significantly.

$$e = -\sum_i p_i \log_2 p_i \text{ bits per pixel} \tag{24}$$

Alternatively, image entropy suggests minimum number of bits that are required to encode given image information in a pixel of given image. Unit for image entropy is bits/pixel.

5. RESULTS AND DISCUSSION

Qualitative and quantitative parameters are obtained from implementation. Qualitative parameters include image appearance and its histograms. Quantitative parameters include mean square error, PSNR, cross correlation, entropy comparison between cover and stego image. Comparative analysis also covers original one third probability and other state of art LSB algorithm's performance characteristics for standard test images. To ensure real world operability, algorithm has also been applied on an image from internet (image 'girl') and another image from digital camera (image 'beach').

5.1 RTL Diagram and Device Utilisation Summary: Timing Summary

Speed Grade: -3

- Minimum period: 9.526 ns (Maximum Frequency: 104.971 MHz)
- Minimum input arrival time before clock: 4.614 ns
- Maximum output required time after clock: 4.743 ns

5.2 Resultant Images and their Histograms

Subjective examination of cover image and stego images reveal that there is no visible image distortion in result images. Histograms of both images shows very slight deviation during embedding process, which makes detection of message hiding difficult using statistical methods like HCF-COM are as shown in Figs. 3-4.

Quantitative analysis shows that correlation between cover and stego image is almost equal to 1, which is near ideal. Entropy of stego image does not deviate from cover image significantly. It suggests that minimum information required to be in cover image does not changes significantly by embedding algorithm. MSE and consequent PSNR levels are nearly equal or marginally better than result statistics of original one third probability algorithm and various other state of art LSB methods results stated in Sarreshtedari[11], *et al.* Results decisively shows that proposed method performs at par with original one third probability and other LSB algorithms in Table 3.

However, images obtained by any LSB steganography method is susceptible to noise and transmission errors. A robust error checking and correcting mechanism is required while transmitting in noisy environment[16]. Lossy compression techniques like JPEG are prevalent for images, which alters LSB bit planes heavily while compressing the image. Significant amount of redundancy and various supporting techniques are required in addition with proposed method to bring down error rate in compressed stego image[17].

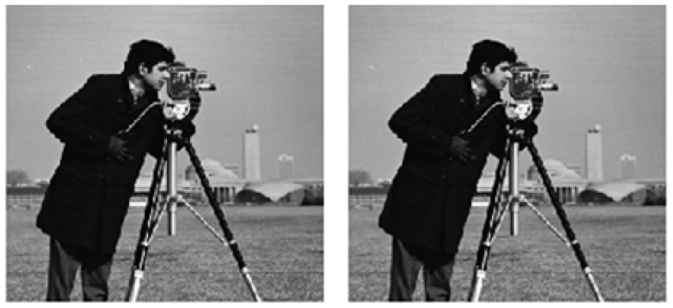
6. CONCLUSIONS AND FUTURE SCOPE

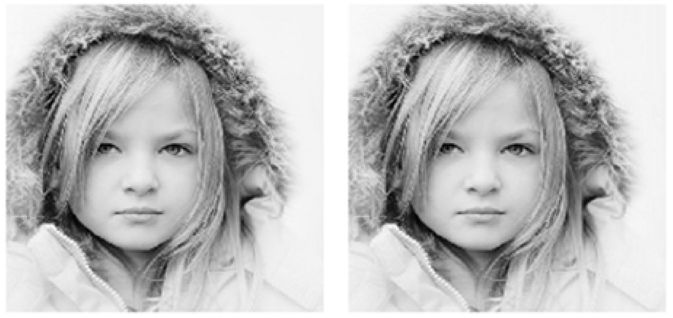
The FPGA implementation of modified one third probability algorithm shows optimised implementation of a steganographic system. It is one of the few examples of system level implementation of a novel spatial domain image steganographic algorithm, which is resistant to many steganalysis attacks. Many operations and elements of original algorithm have been optimised or modified in this work. The proposed system has been implemented on Xilinx SP605 board (Spartan 6 series XC6SLX45T FPGA). The qualitative and quantitative analysis of the results from the implemented system has been done. The performance of the system has been found to be at par with the original algorithm.

However, further hardware implementations for onion steganographic methods can be developed, which includes cryptography and other additional security layers. In the same implementation, power saving and performance enhancing techniques like pipelining and parallelism can be introduced.

(a)

(b)


(c)


(d)

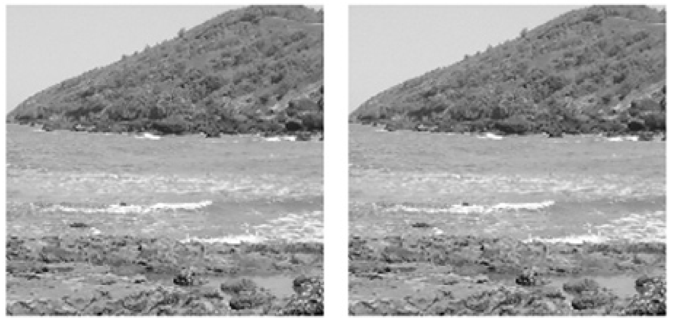

(e)

Figure 3. Various cover images and their resultant in order of top to bottom: (a) Lenna, (b) Peppers, (c) Cameraman, (d) Girl, and (e) Beach.

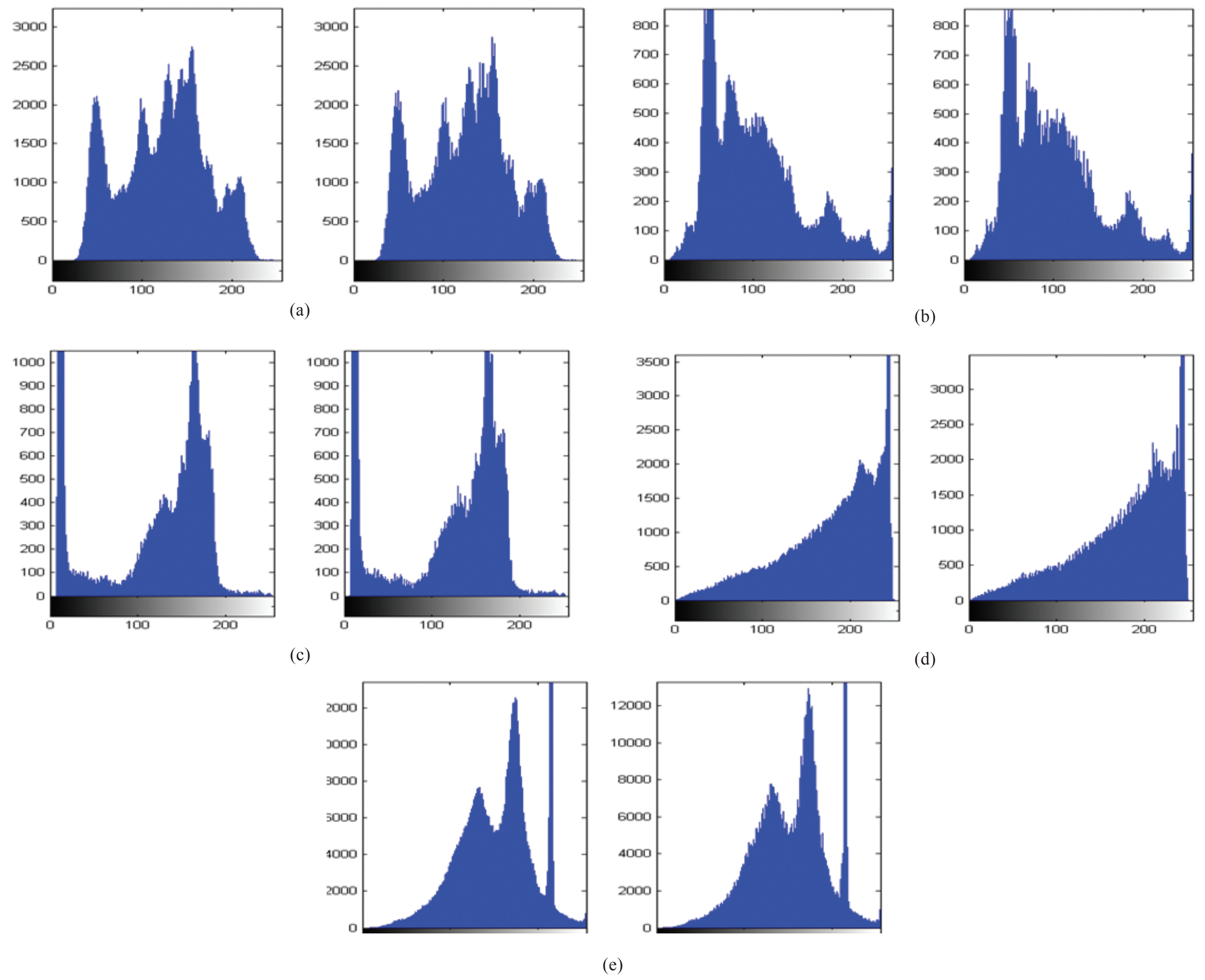


Figure 4. Histograms of various cover image and their resultant stego images of in order of top to bottom: (a) Lenna, (b) Peppers, (c) Cameraman, (d) Girl, and (e) Beach.

Table 3. Quantitative performance analysis of modified embedding algorithm

Image	MSE	PSNR(dB) from original algorithm[11]	PSNR (dB) LSBM[11]	PSNR (dB) LSBMR[11]	PSNR(dB)	Correlation coefficient	Cover entropy (bits/pixel)	Stego image entropy (bits/pixel)
Lenna	0.3334	52.9024	51.1444	52.3887	52.935	0.9999	7.4451	7.4437
Peppers	0.3331	52.8968	51.1439	52.3919	52.9389	0.9999	6.9917	6.9914
Cameraman	0.347	52.9310	51.1357	52.3831	52.9188	1	7.0097	7.0161
Girl	0.3417	NA	NA	NA	52.7941	0.9999	7.3242	7.3377
Beach	0.3370	NA	NA	NA	52.8542	0.9999	7.1698	7.1753

REFERENCES

1. Halenár, R. Steganography used for Copyright Protection in MATLAB environment. *European J. Sci. Theology,* 2014, **10**(1), 253-262.
2. Johnson, N.F. & Jajodia, S. Exploring steganography: Seeing the unseen. *IEEE Computer,* 1998, **31**(2), 26-34. doi: 10.1109/MC.1998.4655281
3. Krenn, J.R. Steganography and steganalysis. http://www.krenn.nl/univ/cry/steg/article.pdf (accessed on 21/03/2015)
4. Nirinjan, U.C. & Anand, D. Watermarking medical images with patient information. *In* the 20th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Hong Kong, China, 1998, pp. 703-06.

doi: 10.1109/IEMBS.1998.745518

5. Murdoch, S.J. & Lewis, S. Embedding covert channels into TCP/IP. Information Hiding Workshop, 2005. Retrieved 11 June 2015.
6. Saha, B. & Sharma, S. Steganographic techniques of data hiding using digital images. *Def. Sci. J.*, 2012, **62**(2), 11-18.
7. Westfeld, A. & Pfitzmann, A. Attacks on steganographic system. *In* Proceeding of Third International workshop on Information Hiding, 1999, **1768**, 22-28. doi: 10.1007/10719724_5
8. Fredrich, J.; Goljian, M. & Du, R. Detecting LSB steganography in color and greyscale images. *IEEE Multimedia*, 2001, **8**(4), 22-28. doi: 10.1109/93.959097
9. Ker, A.D. Improved detection of LSB steganography in grayscale images. *Lecture Notes Comput. Sci.*, 2005, **3200**, 583–592.
10. Mielikainen, J. LSB matching revisited, *IEEE Signal Process. Lett.,* 2006, **13**(5), 285–287. doi: 10.1109/LSP.2006.870357
11. Sarreshtedari, S. & Akhaee, M.A. One-third probability embedding: a new ±1 histogram compensating image least significant bit steganography scheme. *Image Processing, IET,* 2014, **8**(2), 78,89. doi: 10.1049/iet-ipr.2013.0109
12. Mohd. B.J.; Abed, S.; Al-Hayajneh, T. & Alouneh, S. FPGA hardware of the LSB steganography method. *In* International Conference Computer, Information and Telecommunication Systems (CITS), Amman, 2012, pp. 1-4. doi: 10.1109/CITS.2012.6220393
13. Farouk, H.A. & Saeb, M., An FPGA implementation of a special purpose processor for steganography. *In* Proceedings. 2003 IEEE International Conference on Field-Programmable Technology (FPT), 2003, pp. 395-398. doi: 10.1109/FPT.2003.1275785
14. Laces W.A.P.; Garcia-Hernandez, J.J.; FPGA implementation of a low complexity steganographic system for digital images. *In* IEEE/ACIS 14th International Conference on Computer and Information Science (ICIS), Las Vegas, NV, 2015, pp. 319-324. doi: 10.1109/ICIS.2015.7166613
15. Shahadi, H.S.; Jidin R. & Way, W.H. Concurrent hardware architecture for dual-mode audiosteganography processor-based FPGA. *Elsevier Comput. Electr. Eng. J.,* 2015. doi: 10.1016/j.compeleceng.2015.03.007
16. Harmsen, J.J. & Pearlman, W.A. Steganalysis of additive-noise modelable information hiding. *In* International Society for Optics and Photonics on Electronic Imaging, June 2013, pp. 131-142. doi: 10.1117/12.476813
17. Currie III, D.L. & Irvine, C.A. Surmounting the effects of lossy compression on steganography. Naval Postgraduate School Monterey CA Dept. of Computer Science, 1996.

CONTRIBUTORS

Mr Kunjan Pathak is MTech student in VLSI Design at Thapar University, Patiala. His research interests include : Data security, optimisation and applications of various algorithms and protocols on hardware. His previous work includes implementation of TDES on FPGA.
His contribution in the present study includes, evaluation of various steganographic algorithms for suitability of hardware implementation. Further, overall optimisation of one third probability algorithm and implementation details such as state machine design and FPGA implementation has been accomplished by him.

Mrs Manu Bansal is Assistant Professor at department of Electronics and communication, Thapar University, Patiala. Her research interests include : Digital VLSI design. Her previous works are based on minimisation algorithms in VLSI Design.
Her contribution in present study includes overall study of various steganographic algorithms and selection of optimum methodology for hardware implementation.