Journal of information and communication convergence engineering
http://jicce.org

J. Inf. Commun. Converg. Eng. 15(3): 151-159, Sep. 2017 **Regular paper**

FPGA Implementation of LSB-Based Steganography

Quang Do Vinh and Insoo Koo*, *Member, KIICE*

School of Electrical Engineering, University of Ulsan, Ulsan 44610, Korea

Abstract

Steganography, which is popular as an image processing technology, is the art of using digital images to hide a secret message in such a way that its existence can only be discovered by the sender and the intended receiver. This technique has the advantage of concealing secret information in a cover medium without drawing attention to it, unlike cryptography, which tries to convert data into something messy or meaningless. In this paper, we propose two efficient least significant bit (LSB)-based steganography techniques for designing an image-based steganography system on chip using hardware description language (HDL). The proposed techniques manipulate the LSB plane of the cover image to embed text inside it. The output of these algorithms is a stego-image which has the same quality as that of the original image. We also implement the proposed techniques using the Altera field programmable gate array (FPGA) and Quartus II design software.

Index Terms: Field programmable gate array, Hardware design, LSB steganography

I. INTRODUCTION

Steganography is a useful technique for concealing a crucial message inside a cover image, especially a bitmap image. It is famous for the ability to hide data without creating suspicion. Thus, developing an algorithm for hiding data plays an important role in creating a steganography system.

Steganography is usually applied in communication systems in which the required security level is high in order to protect not only the information but also the senders and receivers. The secret message is often embedded into a cover medium, such as a 24-bit digital image (which is thought to be the most popular cover message). Its high-redundancy characteristic ensures there will be enough space to insert data into the image without causing remarkable changes. Meanwhile, color images have now become a universal means of communication and can be found anywhere in the world at any time. Thus, data is protected.

Fig. 1 shows a typical example of a steganography system [1]. In such a system, there are always two main parts. The encoder hides a secret message in a cover image. The output of this process is a stego-image. The decoder retrieves the information from the received stego-image by using predefined rules based on an implicit agreement between the sender and the receiver, including the key and the stego algorithm.

There are several approaches to concealing a message inside an image so that any changes in the original image are undetectable and insensible [2], including least significant bit (LSB) insertion, masking & filtering, and transformation.

In the LSB insertion method, the LSB planes of the cover image are altered by the bits of the message, but conform to particular rules. The quality of the output stego-image is

Received 15 June 2017, **Revised** 18 June 2017, **Accepted** 12 July 2017
***Corresponding Author** Insoo Koo (E-mail: iskoo@ulsan.ac.kr, Tel: +82-52-259-1249)
School of Electrical Engineering, University of Ulsan, 93 Daehak-ro, Nam-gu, Ulsan 44610, Korea.

Open Access **https://doi.org/10.6109/jicce.2017.15.3.151** print ISSN: 2234-8255 online ISSN: 2234-8883

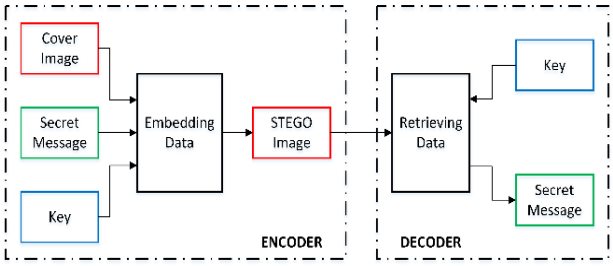

Fig. 1. Basic steganography system.

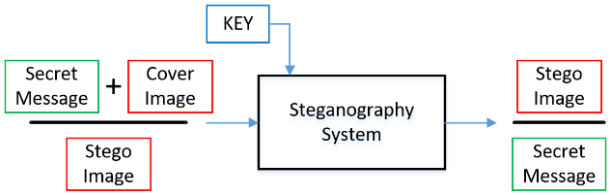

Fig. 2. LSB-based image steganography system.

maintained at the same level as the original. Carrying out this technique requires using an image lossless compression format so that the hidden data will not get lost, which usually happens with lossy compression [3]. The most common image format used in this case is the 24-bit color image.

The masking & filtering method is usually done on 24-bit and greyscale images. The message is embedded in the substantial fields of the image so that they are well mixed. This method resembles paper watermarks, in which the information is scattered throughout the cover image [2].

In the transformation method, which is more robust than LSB insertion, the message is embedded throughout the entire image by utilizing digital signal processing methods, such as discrete cosine transform (DCT), discrete Fourier transform (DFT), and wavelet transform. However, it is hard to implement due to complex mathematical functions [4].

In this paper, we propose two efficient LSB-based steganography techniques that can be used for designing an image-based steganography system using hardware description language (HDL). The basic idea of the system is described in Fig. 2. The input of such a system could be a cover image or a stego-image. If the input file is a cover image, the secret data are embedded into it with the protection of a key. If the input file is the stego-image, the secret information is retrieved by using a reasonable key. In both cases, the secret message is a text file or text entered from a keyboard. Meanwhile, the key is usually a short password that can be memorized easily.

This paper is organized as follows. In Section II, we illustrate the process of developing LSB steganography algorithms implemented in Matlab. Also in this section, a simple LSB insertion technique is simulated with Modelsim using Verilog HDL to verify its efficiency before being implemented using a field programmable gate array (FPGA). The procedure for hardware design is provided in Section III. Finally, in Sections IV and V, we present the system testing environment and the conclusion, respectively.

II. LSB STEGANOGRAPHY ALGORITHMS DEVELOPMENT

MATLAB is an effective mathematical tool in digital signal processing [5]. Thus, it is used for developing the algorithms, for testing the outcomes of the hardware simulation process, and for communicating with the hardware core processor in designing the steganography system.

As mentioned earlier, the basic idea of LSB-based steganography is to embed a message into the least significant bits of an image. Fundamentally, in a bit map image, each pixel is composed of three primary red, green and blue components. And each of them is represented by an eight-bit number which has a value in the range [0, 255]. Thus, every eight bits of the message can be embedded into 24 bits of a pixel [6]. The essential weakness of this technique is that the secret data could be extracted by using statistical analysis. Thus, to prevent the message from being stolen, algorithms are further developed by adding a password for level-2 protection.

In this paper, we propose two algorithms: *LSB insertion* and *watermarking*. These two solutions for steganography were created for the purpose of comparing their strengths and weaknesses in order to choose the most suitable algorithm for hardware design. Each of the two algorithms has pros and cons. For example, watermarking is much more complex than LSB insertion, but the data are protected well; LSB insertion is simple but data can be retrieved easily.

We use a bitmap image as the input cover medium to hide the data, relying on its high-redundancy characteristic and lossless compression. A bitmap image (which has the extension .bmp) is one of the most common image formats. Each bitmap image has a header, which should not be changed, and primary data. The header contains the file extension, the resolution, and the offset (representing the byte distance between the beginning of the bitmap file to the beginning of the pixel bits) [7]. The secret message can only be inserted into the primary data of the image.

A. Image Data in MATLAB and ModelSim

ModelSim is a verification and simulation tool for VHDL, Verilog HDL, System Verilog, and mixed language hardware

Fig. 3. Red, green and blue components in MATLAB.

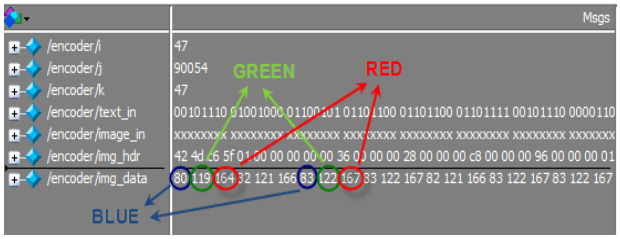

Fig. 4. The structure of the input image data.

designs. This section explains the way in which data are loaded into MATLAB, compared to ModelSim. An image with a resolution of M*N (pixels) is loaded into MATLAB in the form of an M*N*3 (bytes) matrix in which each primary color (red, green, and blue) occupies M*N (bytes). These matrices contain decimal values, as shown in Fig. 3.

Meanwhile, ModelSim reads the input bitmap image in the form of a binary value array. These numbers are converted to decimal values to compare with those in MATLAB. And the following image data are derived from ModelSim. One important thing to be noticed is the way the numbers are arranged in the data array, as can be seen in Fig. 4.

From now on, when we discuss embedding a message in an image, there is an implicit agreement that the message will be embedded into the pixel bits, not the header.

B. LSB Insertion Algorithm

Message and key are embedded together into the cover image to provide stronger protection. Thus, the embedded data are formed as expressed below:

Data = [l-key key Message]

In this formula, l-key is the length of the key that should be estimated in advance. The LSB-embedding algorithm substitutes the LSBs of the pixels of the cover image with the bit stream of the text to be hidden. More specifically, every eight bits of data will be embedded into every pixel, or 24 bits, of the image, as illustrated in Fig. 5.

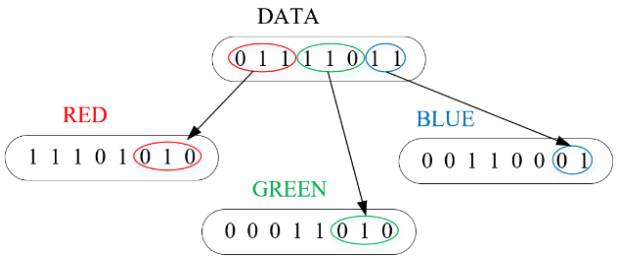

Fig. 5. A simple LSB algorithm inserts eight bits of data into one pixel (24 bits) of the cover image.

The stego-image is almost the same as the cover image because the variations in the LSBs of the image pixels do not result in any remarkable differences in the image. The detailed LSB-based data hiding algorithms are explained below.

1) Embedding Process

Step 1: Read the input cover image, then extract the image header and image data; decompose the image data into red (R), green (G), and blue (B) channels; meanwhile, the image header must remain unchanged for further use in Step 5.

Step 2: Clear the LSB planes of the R, G, and B that will be used for inserting data.

```
LSB_R = bitand(R,248);   %use the last three bits
LSB_G = bitand(G,248);   %use the last three bits
LSB_B = bitand(B,252);   %use the last two bits
```

Step 3: Insert every eight bits of data into every pixel of the image using the *bitor* operator.

Step 4: Recompose R, G, and B to create the stego-image.

2) Extraction Process

Step 1: Read the stego-image and extract the primary R, G, and B colors.

Step 2: Use the key to get the length of the secret text embedded in the stego-image.

Step 3: Find the secret text (in the form of a binary array) hidden inside the image.

Step 4: Convert ASCII numbers into strings, and save the output data into a text file called “message.txt.”

C. Watermarking Algorithm

Watermarking is the process of inserting proprietary information in a digital image by presenting alterations to its pixels with minimum perceptual disturbance [8]. In this paper, the fundamentals of the watermarking algorithm are almost the same as LSB insertion, meaning that this method also uses the LSBs of the cover image to hide the data.

RGB YCbCr

Fig. 6. RGB and YCbCr images.

However, the RGB image will be converted into a YCbCr image for embedding data. In this format, the luminance (Y) component represents the brightness of the image, while the chrominance (Cb and Cr) components store the color difference information. The main reason for the transformation of RGB to YCbCr is that the human eye has different sensitivity to color and brightness [9]. The Y element is more attractive to the human eye than the other two, whereas any changes taking place in the Cb and Cr elements will not affect the structure and quality of the image. Thus, the secret message will be covered in the Cb and Cr components of the image. Fig. 6 shows an example of the differences between RGB and YCbCr images.

The watermarking algorithm is illustrated in the following steps.

1) Encoding Process

Step 1: Read the input cover image; convert it to YCbCr format by using the following approximation [10]:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.499 \\ 0.499 & -0.418 & -0.0813 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \quad (1)$$

Step 2: Extract the LSBs of the Cb and Cr components by using the modulo-2 operation.

Step 3: Select sub-matrices from LSB_Cb and LSB_Cr, then specify their positions by using *(row j, column k)*.

Step 4: Insert every single bit of data into every LSB_Cb and LSB_Cr matrix pair using the following rules:

If mod(j+k,2) = 0 then
If data = 1: LSB_Cb = 0 and LSB_Cr = 1
If data = 0: LSB_Cb = 1 and LSB_Cr = 0
Else
If data = 1: LSB_Cb = 1 and LSB_Cr = 0
If data = 0: LSB_Cb = 0 and LSB_Cr = 1

An example of this algorithm is in Fig. 7.

Step 5: Recompose Y, Cb, and Cr components into the YCbCr image, then convert the image to bitmap format to create the stego-image.

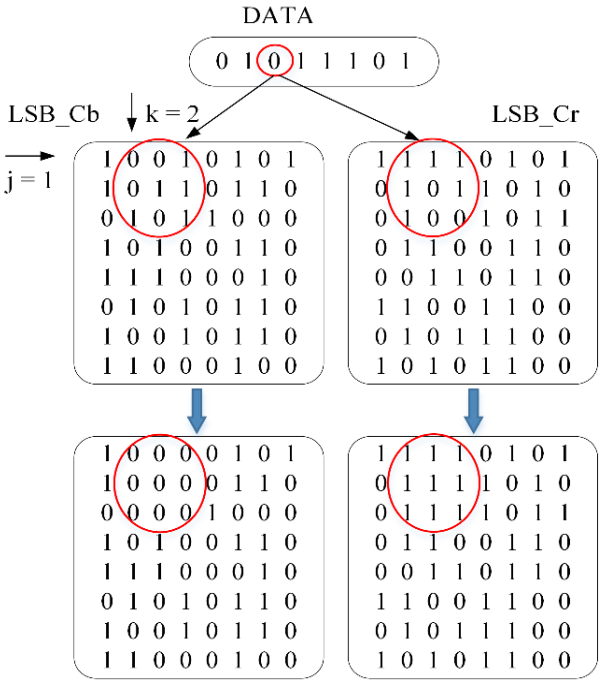

Fig. 7. An example of embedding one bit of data into two 3×3 matrices at position (1,2).

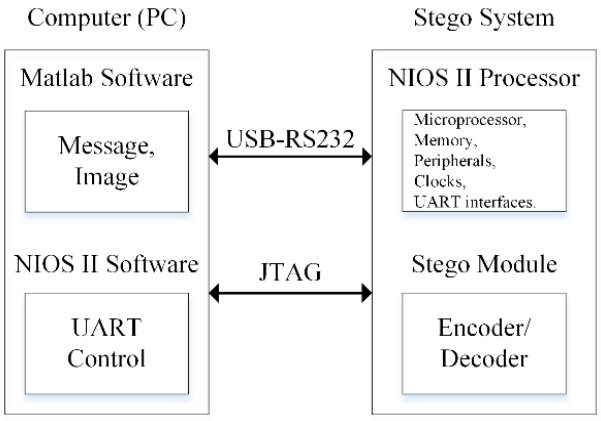

Fig. 8. Steganography (Stego) system model.

2) Decoding Process

The inverse procedure is performed to retrieve the data. However, due to the process of converting the image from RGB to YCbCr, and vice versa, the LSB matrices are changed a little bit, which could lead to changes in the data. The solution to this problem is to count the number of 0 and 1 bits in those selective sub-matrices to extract the value of the respective watermark data. For example, if the number of 0 bits in the LSB_Cb sub-matrix at position (1,2) is greater than the number of 1 bits, this LSB_Cb sub-matrix is supposed to be a *zeros* matrix, and thus, the watermark bit will be 0.

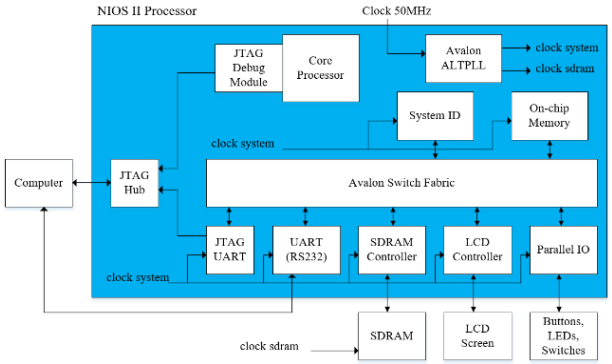

Fig. 9. The architecture of the NIOS II processor on an FPGA chip.

III. SYSTEM-ON-CHIP DESIGN

The main purpose of this paper is to use Verilog HDL to design a system-on-chip (SoC) that can execute an LSB-based image steganography technique. The algorithms for embedding data into an image have been developed and verified by using Matlab and Modelsim. In this section, Quartus II software is used to create the SoC. The Altera DE2 evaluation kit is used to implement this SoC using an FPGA chip. The simple model of the proposed steganography system is presented in Fig. 8.

SoC refers to a single integrated circuit (or chip) composed of all the components of an electronic system [11]. In this model, the stego system contains two main parts, including the Altera NIOS II processor and a stego module. This system will be able to communicate with a computer through a USB-to-RS232 cable (for testing purpose) and a Joint Test Action Group (JTAG) cable (for downloading the NIOS II software used to control the system).

A. NIOS II Processor

Fig. 9 shows the detailed specifications of the Altera NIOS II processor consisting of the following components:

- A core processor, system ID, on-chip memory
- Clock controller: Altera phase-locked loop (ALTPLL)
- Synchronous dynamic random-access memory (SDRAM) controller
- JTAG universal asynchronous receiver-transmitter (UART) and Serial UART (RS232)
- LCD controller
- Parallel input/output (IO) controller

SOPC Builder was used to create the processor. It is a powerful system development tool already included as part of the Quartus II software. We need to specify the system components, and SOPC Builder generates the interconnect logic automatically. In this system, there are several ready-to-use SOPC Builder components provided by Altera and third-party developers, including the NIOS II core processor (with system ID and on-chip memory), the Avalon ALTPLL, JTAG UART, SDRAM controller, and a parallel IO controller. Meanwhile, the serial UART and liquid crystal display (LCD) controllers are written using Verilog HDL and imported into SOPC Builder as custom components.

1) System ID

The system ID helps to specify its existence in the FPGA chip. This ID must be specified beforehand by the designer to keep communications between NIOS II software and the hardware stable.

2) SDRAM Controller

The SDRAM controller helps the system to store data, such as the message and the image in 8 MB of SDRAM.

3) Altera Phase-Locked Loop

The ALTPLL implements phase-locked loop circuitry, which operates by producing an oscillator frequency to match the frequency of the input clock (50 MHz).

4) UART (RS-232 Serial Port)

The system uses this port to communicate with a computer for transmitting and receiving data (message and image). The basic settings for this controller are:

- Parity: none
- Data bits: 8
- Stop bit: 1
- Baud rate: 38400 bps

5) Other Controllers

The remaining controllers are used for testing and demonstration purposes. The parallel IO controller helps to connect the system with external light emitting diodes (LEDs), seven-segment displays, and switches. The LCD controller will control the 16x2 character LCD to display information like the operation mode of the system (encoder or decoder).

B. Stego Module

The steganography module has two main parts: encoder and decoder. The encoder will perform the data embedding process. Meanwhile, the data retrieval process is carried out by the decoder. Fig. 10 describes the connection between the NIOS II processor and the stego module.

As seen in the figure, the processor receives data, including text and images, from a computer in the form of a binary array. These data will be stored in SDRAM and are pre-processed by NIOS II software before being sent to the

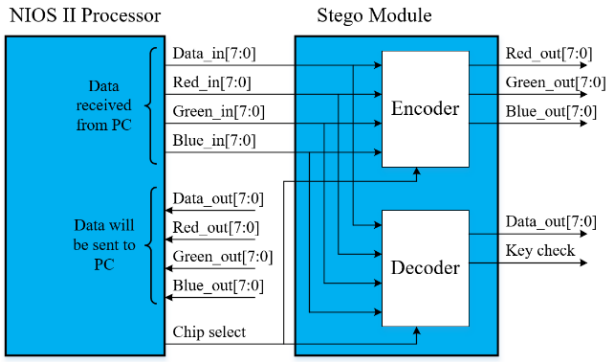

Fig. 10. The connection between the NIOS II processor and the stego module.

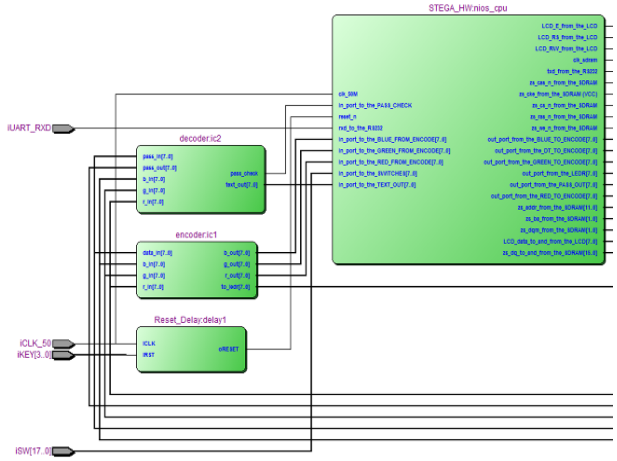

Fig. 11. Steganography system's basic connections at the register-transfer level.

stego module. There are four streams of eight-bit data, which will be sent to the stego module every clock cycle.

The chip-select signal is used to choose the operation mode (encode or decode). With the encoder, the inputs are Message and Key (Data_in) and the cover image (Red_in, Green_in, and Blue_in); the outputs are the bit streams of the stego-image. With the decoder, the inputs are Key and Stego-image; the outputs are the message bit stream and the key-check signal. This key-check signal will tell the processor whether the input key is correct or not. Fig. 11 presents the basic connections between the processor and the stego module at the register transfer level (RTL).

IV. SYSTEM TESTING

To test the efficiency of the system, data were transmitted from a PC to the stego system, which was implemented using an FPGA Cyclone II chip on the DE2 kit. The system will execute the steganography techniques on the input binary data, and then send the output data back to the PC. Communications between the PC and the system (on the DE2 kit) is carried out through a USB-to-RS232 cable at 38400 bps. A comparative analysis of the proposed algorithms was also investigated.

Prerequisites:

- Hardware: PC, USB-to-RS232 cable, DE2 kit.
- Software: Matlab, Quartus II, NIOS II.

A. Testing Environment

The testing environment for the encoder and decoder modules is presented in Figs. 12 and 13, respectively.

Regarding the encoding process, the cover image, the secret message, and the key are read into Matlab in the form of a binary array, which is then transmitted to the stego system through the USB-to-RS232 cable. The system embeds the message in the image, and then generates the stego-image. This image is sent back to the PC for further analysis. The decoding process, on the other hand, uses the stego-image and a key as its inputs. And the output will be the secret message if this key matches the key from the encoding process.

The encoder testing procedure is illustrated in Fig. 14.

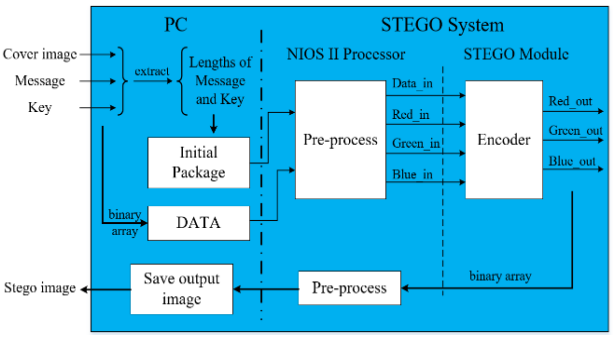

Fig. 12. Testing environment for the encoding process.

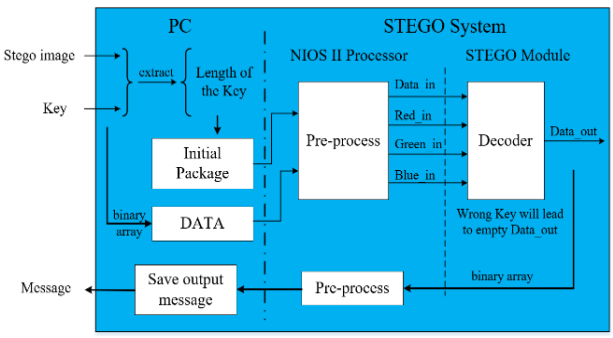

Fig. 13. Testing environment for the decoding process.

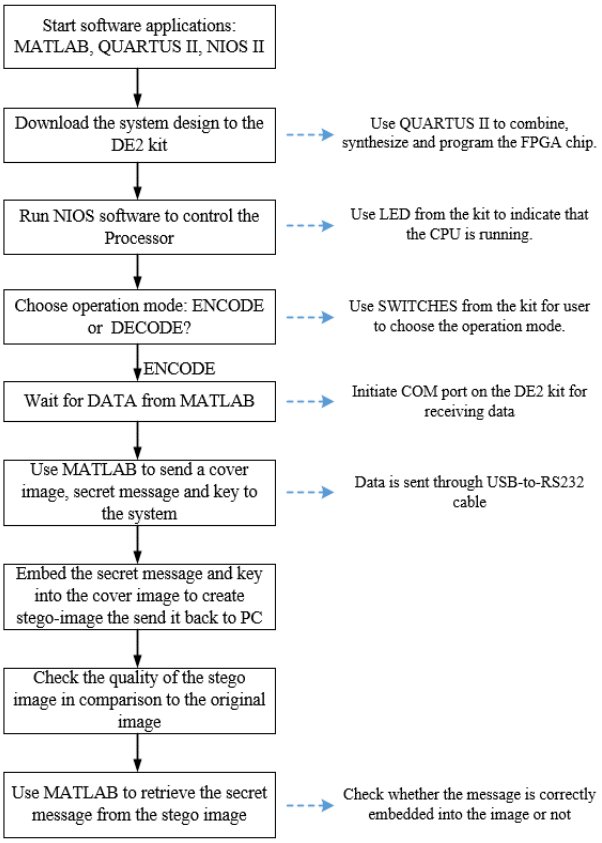

Fig. 14. Encoder testing procedure.

B. Results and Discussion

The length of the message that can be hidden inside an image should also be considered in evaluating the algorithm's efficiency. Regarding the LSB insertion algorithm, if the resolution of the cover image is M×N, then it can be used to insert up to M×N×8 bits of the secret message. For the watermarking technique, the total bits of the information that can be hidden in the M×N cover image is (M×N)/(A×B), such that A×B is the size of the submatrices that are selected for inserting each bit of the data. In the proposed LSB-based steganography algorithms, 150×200 color images were used as cover images. In theory, secret text of up to 30,000 characters can be hidden using the LSB insertion technique. And the maximum characters of a text message that can be embedded into this image using the watermarking technique are 150×200/4/8 ≈ 937 bytes, if the size of the selected sub-matrices is 2×2.

The images can be distorted in the embedding process due to changing pixel bits. Distortion is measured via two performance evaluation parameters: mean square error (MSE) and peak signal-to-noise ratio (PSNR). MSE and PSNR can be calculated using the following equations [12]:

$$MSE = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} \left(X_{ij} - Y_{ij} \right)^2 , \tag{2}$$

$$PSNR = 10 \log_{10} \left(\frac{I_{\max}^2}{MSE} \right) dB , \tag{3}$$

where M indicates the total number of pixels in the horizontal dimensions of the image. N shows the total number of pixels in the vertical dimension. I_{max} is the maximum intensity value of each pixel, which could be equal to 255 for eight-bit grayscale or color images.

A lower MSE value indicates a better image quality. On the other hand, a higher PSNR means better image quality. Comparisons between MSE and PSNR of the two proposed LSB algorithms, based on the effect of changing the length of the secret text, are shown in Figs. 15 and 16.

As can be seen from the figures, the increment in the length of the secret message causes the MSE to increase and PSNR to decrease, which leads to higher statistical distortion in the stego-image, in comparison to the original. Furthermore, due to its complexity, the watermarking technique produces higher MSE and lower PSNR values, which indicates that the quality of the stego-image is not as good as that generated by the LSB insertion algorithm.

Another metric that can be used to evaluate the robustness performance of the two steganography algorithms is RS steganalysis [13]. The RS steganalytic technique explores the statistics of the stego-image by dividing the image into many groups (G) of the same size (M). The variation of each group can be chosen as the discrimination function f:

$$f(x_1, x_2, ..., x_n) = \sum_{i=1}^{n-1} | x_{i+1} - x_i | . \tag{4}$$

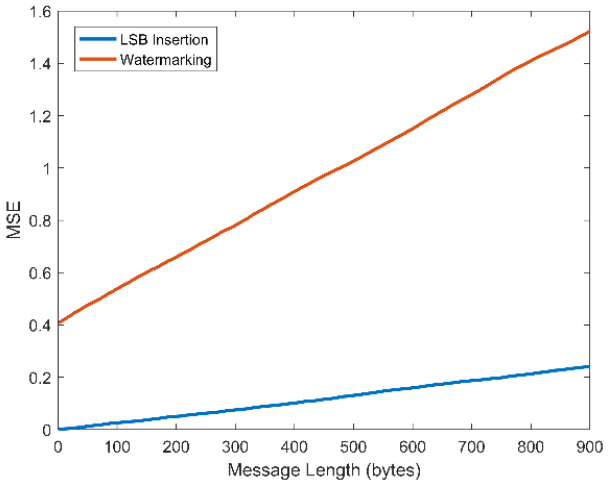

Fig. 15. The effect on MSE from changing the length of text for the LSB-based image steganography algorithms.

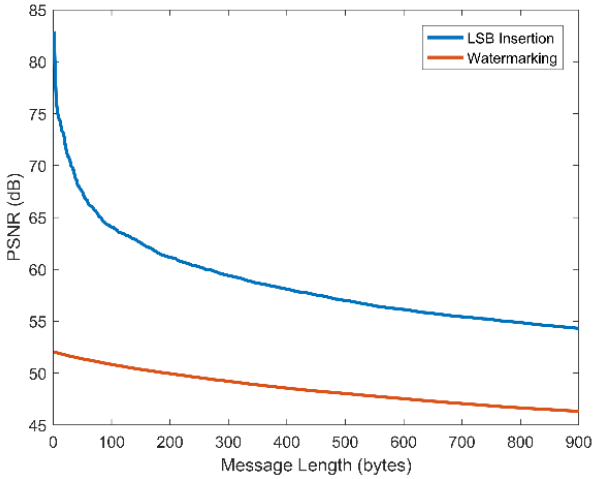

Fig. 16. The effect on PSNR from changing the length of text for the LSB image steganography algorithms.

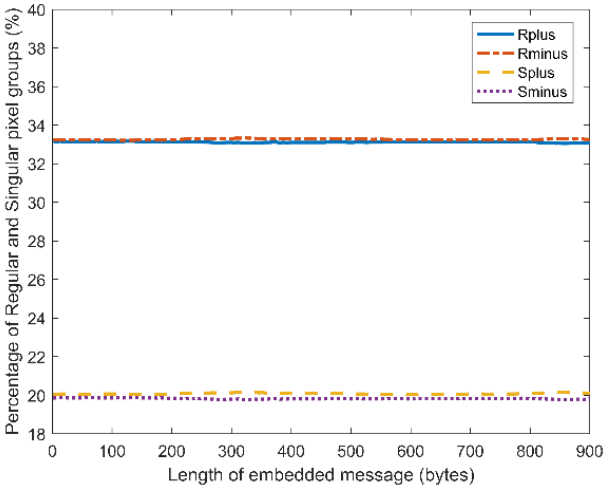

Fig. 17. RS-diagram of the stego-image created using LSB insertion algorithm.

It then defines two flipping operations $F_1 : 2n \leftrightarrow 2n+1$ and $F_{-1} : 2n-1 \leftrightarrow 2n$. The number of regular and singular groups are denoted as R_M and S_M, as follows.

$$R_M = \frac{\sum G \mid f(F_1(G)) > f(G)}{\sum G}, \tag{5}$$

$$S_M = \frac{\sum G \mid f(F_1(G)) < f(G)}{\sum G}. \tag{6}$$

R_{-M} and S_{-M} are defined similarly using F_{-1} function instead of F_1. The statistical hypothesis of RS steganalysis is that in a typical image with no hidden data, $R_M \cong R_{-M}$ and $S_M \cong S_{-M}$. In case of a stego-image, the difference between R_{-M} and S_{-M} increases faster than the difference between R_M and S_M.

Figs. 17 and 18 show RS diagrams for the two proposed LSB-based image steganography algorithms. As can be seen from the figures, the LSB insertion algorithm satisfies that the expected value of R_M and S_M equal the value of R_{-M} and S_{-M}, respectively; then it is robust to RS analysis attack. On the other hand, the watermarking technique seems to be vulnerable to RS analysis as the difference between R_{-M} and S_{-M} increases when the length of the input secret message increases. This is because a 900-byte embedded message is almost the peak hiding capacity of Watermarking algorithm when the 150×200 cover image is used. Meanwhile, it occupies only three percent of the total pixels of the cover image when it is embedded using the LSB insertion technique. Therefore, the LSB insertion is more suitable than the watermarking algorithm for implementing a steganography system using an FPGA.

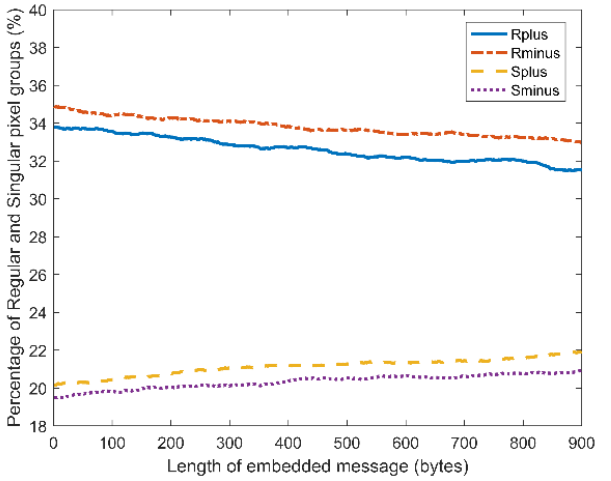

Fig. 18. RS-diagram of the stego-image created using watermarking algorithm.

V. CONCLUSION

In this paper, we propose two LSB-based image steganography techniques for embedding data in color images. The algorithms were first developed using Matlab software and were then applied to implement a steganography SoC using an FPGA. While the simple LSB insertion embeds each byte of the secret message into each pixel of the cover image, the watermarking hides every single bit of the data into selective LSB sub-matrices of the Cb and Cr components extracted from the original image. The simulation results show that PSNR decreases as the length of the secret message increases. The results also

indicate that the LSB insertion is more appropriate for hardware implementation.

ACKNOWLEDGEMENTS

This work was supported by the National Research Foundation of Korea funded by the MEST (Grant No. NRF-2015R1A2A11A15053452 and NRF-2015R1D1A1A09057077).

REFERENCES

[1] M. M. Amin, M. Salleh, S. Ibrahim, M. R. Katmin, and M. Z. I. Shamsuddin, "Information hiding using steganography," in *Proceedings of 4th National Conference on Telecommunication Technology (NCTT2003)*, Shah Alam, Malaysia, pp. 21–25, 2003.

[2] N. F. Johnson and S. Jajodia, "Exploring steganography: seeing the unseen," *IEEE Computer*, vol. 31, no. 2, pp. 26–34, 1998.

[3] L. Wang, J. Wu, L. Jiao, L. Zhang, and G. Shi, "Lossy to lossless image compression based on reversible integer DCT," in *Proceedings of 15th IEEE International Conference on Image Processing (ICIP2008)*, San Diego, CA, pp. 1037–1040, 2008.

[4] S. Owais, A. Zaidi, T. A. Khan, S. S. Hussain, and M. Hashmani, "A trend in global steganography and steganalysis approaches," *Asian Journal of Engineering, Sciences & Technology*, vol. 4, no. 1, pp. 30–33, 2014.

[5] G. Blanchet and M. Charbit, *Digital Signal and Image Processing using MATLAB*, 1st ed. New York, NY: Wiley, 2006.

[6] R. Chandramouli, M. Kharrazi, and N. Memon, "Image steganography and steganalysis: concepts and practice," in *Digital Watermarking*. Heidelberg: Springer, pp. 35–49, 2003.

[7] M. S. Al Rababaa, "Colored image-in-image hiding," in *Proceedings of the 10th International Conference (CADSM2009): The Experience of Designing and Application of CAD Systems Microelectronics*, Lviv-Polyana, Ukraine, pp. 445–450, 2009.

[8] W. C. Chu, "DCT-based image watermarking using subsampling," *IEEE Transactions on Multimedia*, vol. 5, no. 1, pp. 34–38, 2003.

[9] M. S. Nixon and A. S. Aguado, *Feature Extraction & Image Processing for Computer Vision*, 3rd ed. London: Academic Press, 2012.

[10] Y. Yang, P. Yuhua, and L. Zhaoguang, "A fast algorithm for YCbCr to RGB conversion," *IEEE Transactions on Consumer Electronics*, vol. 53, no. 4, pp. 1490–1493, 2007.

[11] T. Risset, "SoC (System on Chip)," in *Encyclopedia of Parallel Computing*. Boston, MA: Springer, pp. 1837–1842, 2011.

[12] S. Mahmoudpour and S. Mirzakuchaki, "Hardware architecture for a message hiding algorithm with novel randomizers," *International Journal of Computer Applications*, vol. 37, no. 7, pp. 46–53, 2012.

[13] J. Fridrich, M. Goljan, and R. Du, "Detecting LSB steganography in color and gray-scale images," *IEEE Multimedia*, vol. 8, no. 4, pp. 22–28, 2001.

Quang Do Vinh
received his B.E. degree from Ho Chi Minh City University of Technology, Vietnam, in 2009, and his M.Sc. degrees from RMIT University, Melbourne, Australia, in 2012. He currently enrolls in PhD program at the University of Ulsan, Korea since March 2017.

Insoo Koo
received his B.E. degree from Konkuk University, Seoul, Korea, in 1996, and his M.S. and Ph.D. degrees from Gwangju Institute of Science and Technology (GIST), Gwangju, Korea, in 1998 and 2002, respectively. From 2002 to 2004, he worked with the Ultrafast Fiber-Optic Networks (UFON) Research Center in GIST, as a research professor. For one year from September 2003, he was a visiting scholar at the Royal Institute of Science and Technology, Sweden. In 2005, he joined the University of Ulsan, where he is now a full professor. His research interests include next-generation wireless communication systems and wireless sensor networks.