Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 5216029, 8 pages
https://doi.org/10.1155/2018/5216029

Research Article

Embedded FPGA Design for Optimal Pixel Adjustment Process of Image Steganography

Chiung-Wei Huang,[1,2] **Changmin Chou**,[2] **Yu-Che Chiu**,[1] **and Cheng-Yuan Chang**[1]

[1]*Chung Yuan Christian University, Taoyuan City, Taiwan*
[2]*Chien Hsin University of Science and Technology, Taoyuan City, Taiwan*

Correspondence should be addressed to Cheng-Yuan Chang; ccy@cycu.edu.tw

Received 4 November 2017; Revised 5 February 2018; Accepted 19 February 2018; Published 22 March 2018

Academic Editor: Ahmed Refaey

We propose a prototype of field programmable gate array (FPGA) implementation for optimal pixel adjustment process (OPAP) algorithm of image steganography. In the proposed scheme, the cover image and the secret *message* are transmitted from a personal computer (PC) to an FPGA board using RS232 interface for hardware processing. We firstly embed k-bit secret message into each pixel of the cover image by the last-significant-bit (LSB) substitution method, followed by executing associated OPAP calculations to construct a stego pixel. After all pixels of the cover image have been embedded, a stego image is created and transmitted from FPGA back to the PC and stored in the PC. Moreover, we have extended the basic pixel-wise structure to a parallel structure which can fully use the hardware devices to speed up the embedding process and embed several bits of secret message at the same time. Through parallel mechanism of the hardware based design, the data hiding process can be completed in few clock cycles to produce steganography outcome. Experimental results show the effectiveness and correctness of the proposed scheme.

1. Introduction

In this digitized era, almost all kinds of information such as plain texts, voices, images, and videos can be digitally presented, leading to the birth of a wide range of digital media. Increased Internet applications also lead people frequently to deliver digitized information. In the information delivery process, the data hiding technology plays an important role in issues such as data security for digital communications, copyright protection for digital assets, the use of digital media to conceal communications, and certification of various electronic services.

Data hiding technique is to hide confidential messages into a cover media such that an unintended observer will not be aware of the existence of the hidden information. The confidential message can also be compressed and encrypted before embedding to an image. Embedding data into an image is one of the methods of data hiding to achieve data security. In this paper, 8-bit grayscale images are selected as the cover images. The cover images embedded with the secret message are called the stego images. For data hiding methods, the image quality refers to the quality of the stego images.

Many image steganography techniques have been proposed in the last two decades. These techniques can be classified into two domains: frequency domain techniques and image (spatial) domain techniques [1]. Most frequency domain techniques apply discrete cosine transform (DCT) [2, 3] or discrete wavelet transform (DWT) [4, 5] techniques on images to manipulate the coefficients of frequency transforms. The image domain techniques usually apply bit insertion and noise manipulation procedures on cover images [6].

The data hiding operation will result in distortion in the cover image. For some sensitive applications such as military or medical images, it is desirable to restore the original cover image from the stego image after extraction of the hidden data. The reversible data hiding technology has been identified as an effective way for integrity and copyright protection. Reversible data hiding techniques can be roughly classified into three types [7]: lossless compression based methods [8, 9], difference expansion (DE) methods [10–12],

and histogram modification (HM) methods [13, 14]. The lossless compression based methods make use of statistical redundancy of the cover image by performing lossless compression to create a spare space to accommodate additional secret data. The DE-based methods divide an image into pixel pairs and embed secret data into expanded difference values. The HM-based methods shift several or the maximum points in histogram bins of the original image to reserve spare space for data embedding.

Among image domain techniques, the last-significant-bit (LSB) replacement method [8, 9], which proposes to replace the least k-bit of the cover images by the secret message, is the most straightforward algorithm for embedding message into images. Since the LSB replacement method is easy to be detected and the secret message will be cracked by malicious attackers, more sophisticated methods such as the optimal moderately significant-bit replacement method [11] or the differences between original and predicted pixel values method [12] have been proposed and gotten certain degrees of success. However, most published techniques were implemented by using software schemes and few research implemented image steganography algorithms on hardware methods. Usually, hardware implementations are faster than software implementations in that the hardware logic gates can be processed in a parallel processing manner. The higher the complexity of the algorithm is, the more advantages the hardware implementations can get.

During recent years, field programmable gate array (FPGA) has become the dominant form of programmable logic [15]. In comparison to previous programmable devices like programmable array logic (PAL) and complex programmable logic devices (CPLD), FPGA can implement far larger logic functions than the others. In addition, FPGA not only supports sufficient logic arrays to implement more complicated systems and subsystems, but also exploits the increasing capacity of integrated circuits to provide designers with reconfigurable logic arrays that can be programmed on application specific basis. This drastically increases the flexibility in both the design process and the final manufacture by permitting a board-level design to perform many functions.

Among all presented techniques, optimal pixel adjustment process (OPAP) [11] is one of the well-defined algorithms for image data hiding. Some other data hiding algorithms may have more information capacity, but their hardware implementations will be much more complicated. In this paper, we proposed an OPAP design of circuit using FPGA with the hardware description language Verilog HDL (Hardware Description Language). The FPGA board, Xilinx Spartan-3E Starter Kit, is applied to fulfill the image data hiding technique. Using both software and hardware based schemes, two image hiding experiments are fulfilled to compare the correctness and efficiency of the proposed hardware based FPGA method.

The rest of the paper is organized as follows. Section 2 introduces the works about the LSB substitution method and OPAP technique. The steps to implement image steganography by FPGA are also described in detail. In Section 3, experimental evaluation and discussion about data hiding into images are presented to verify the enhancement of the hardware based image steganography technique.

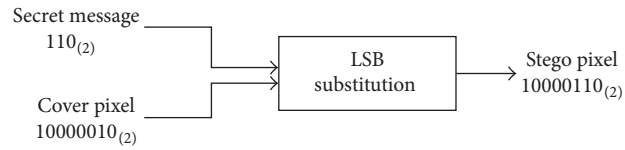

Figure 1: An example of the LSB substitution method.

2. Image Steganography by FPGA

In this section, we introduce the related works about the LSB method and the OPAP technique, which are essential to the data hiding process.

2.1. Related Works

2.1.1. The LSB Substitution Method. The LSB substitution method is a simple and easy image steganography method. This method replaces the k least significant bits (LSB) by secret messages in each pixel. In a 256-level gray image, the value of each pixel is usually represented in an 8-bit binary form:

$$p = \left(b_7 \times 2^7 + b_6 \times 2^6 + \cdots + b_0 \times 2^0\right), \tag{1}$$

where p is the value of a pixel, and b_i $(0 \le i \le 7)$ represents the 8-bit binary. As an example of the LSB substitution method, for a pixel value $130_{(10)}$, secret message $110_{(2)}$, and replacement width $k = 3$, we can try to embed the secret message into the pixel of image. Therefore, the original 8-bit binary value is 10000010 before the data hiding, and it becomes 10000110 after substituting the least 3 significant bits by the secret message $110_{(2)}$. The pixel value has changed from $130_{(10)}$ to $134_{(10)}$, and the difference of $4_{(10)}$ will not be noticed by human observation. Figure 1 illustrates this example of the LSB substitution method.

2.1.2. OPAP Algorithm. An OPAP method is proposed to enhance the image quality and security of the stego image obtained by the LSB substitution method [11]. Assume that p_i, p_i', and p_i'' are the corresponding pixel values of the ith pixel in the cover image, the stego image obtained by the LSB substitution method, and the refined stego image obtained by OPAP approach, respectively. Let $\delta_i = p_i' - p_i$ be the embedding error between p_i and p_i'. According to the embedding process of the LSB substitution method, p_i' is obtained by replacing the k least significant bits of p_i with k-bit secret message; therefore, the range of the embedding error is

$$-2^k < \delta_i < 2^k. \tag{2}$$

However, the value of δ_i can be further segmented into three intervals, such that

Interval 1: $2^{k-1} < \delta_i < 2^k$,

if $p_i' \ge 2^k$ then $p_i'' = p_i' - 2^k$

else $p_i'' = p_i'$

$$\begin{aligned}
&\text{Interval 2:} \quad -2^{k-1} \le \delta_i \le 2^{k-1},\\
&\qquad \text{when } p_i'' = p_i'\\
&\text{Interval 3:} \quad -2^{k} < \delta_i < -2^{k-1}\\
&\qquad \text{if } p_i' < 256 - 2^k \text{ then } p_i'' = p_i' + 2^k\\
&\qquad \text{else } p_i'' = p_i'.
\end{aligned} \tag{3}$$

In the above three cases, the absolute value of δ_i may fall into the range $2^{k-1} < |\delta_i| < 2^k$ only when $256 - 2^k \le p_i' < 2^k$. For other possible values of p_i', the embedding error δ_i will fall into range $0 \le |\delta_i| \le 2^{k-1}$.

2.2. FPGA Design with the OPAP Algorithm. In image data hiding process, researchers usually use the peak signal noise ratio (PSNR) to estimate the quality of the stego images. The PSNR is stated as follows:

$$\begin{aligned}
\text{MSE} &= \left(\frac{1}{M \cdot N}\right) \sum_{i=0}^{M} \sum_{j=0}^{N} \left(p_{i,j}' - p_{i,j}\right)^2,\\
\text{PSNR} &= 10 \cdot \log_{10}\left(\frac{255^2}{\text{MSE}}\right),
\end{aligned} \tag{4}$$

where M and N are the dimensions of the cover image and $p_{i,j}$ and $p_{i,j}'$ represent the pixels of the cover image and stego image, respectively. The higher PSNR value means the lower distortion and better image quality. Table 1 shows a typical result of embedding secret message to a cover image (Lena) by the OPAP algorithm. It shows that the PSNR of the stego image will be greater than 30 dB when the replacement width $k \le 4$. According to [11], 30 dB is a reasonable threshold to make the stego image nature enough. In other words, the secret messages will not be easily noticed or detected by malicious attackers when $k \le 4$. Thus, we set $k = 4$ in our experiments and thorough the following discussion.

Figure 2 shows the procedure of the proposed FPGA based image steganography method. First, the parameters M, N, and k are transmitted to the FPGA structure by RS-232, where M, N are the dimension of the cover image, and k is the replacement width of bits. That is, k bits of the secret message will be hided into the k LSB bits of one pixel. The bigger the k value, the larger the difference between the cover image and the stego image. The embedding scheme is processed pixel by pixel utilizing FPGA. In the proposed FPGA design, the OPAP algorithm for embedding a pixel is shown with the following five steps:

Step 1. Read the cover image pixel by pixel, the pixel p_i is called the cover pixel.

Step 2. Embed k-bit secret message into p_i by the LSB substitution method to get p_i'.

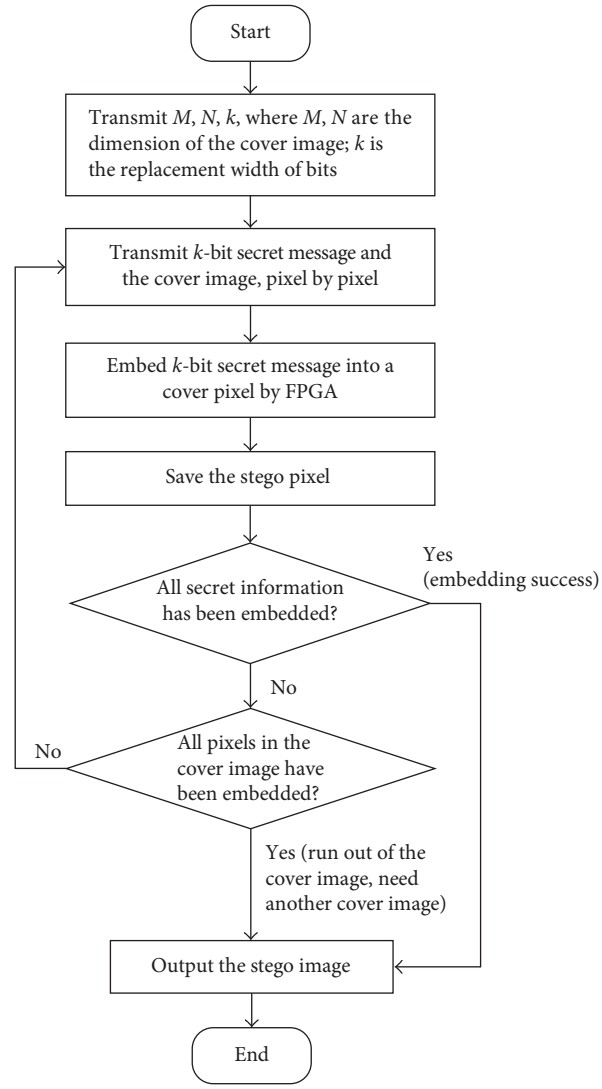

Figure 2: The flowchart of the proposed FPGA based procedure.

Step 3. Calculate the embedding error $\delta_i = p_i' - p_i$.

Step 4. Locate δ_i to its corresponding interval and process with associated calculations according to (3) to produce the stego pixel p_i''.

Step 5. Output the embedded stego pixel p_i'' and save it in memory.

After all the secret information has been embedded, we can save and output the stego image. The maximum storage of this embedding scheme is $M \times N \times k$ bits. Since the proposed procedure embeds the secret message to the cover image pixel by pixel, the time complexity is $O(M \times N)$.

2.3. Hardware FPGA Design Method. Figure 3 shows the structure of the hardware FPGA design, including the receiver, Stand_by, Din_Lsb_sub, D_emb, Transmitter, and Work_time modules. The function of each module is illustrated as follows.

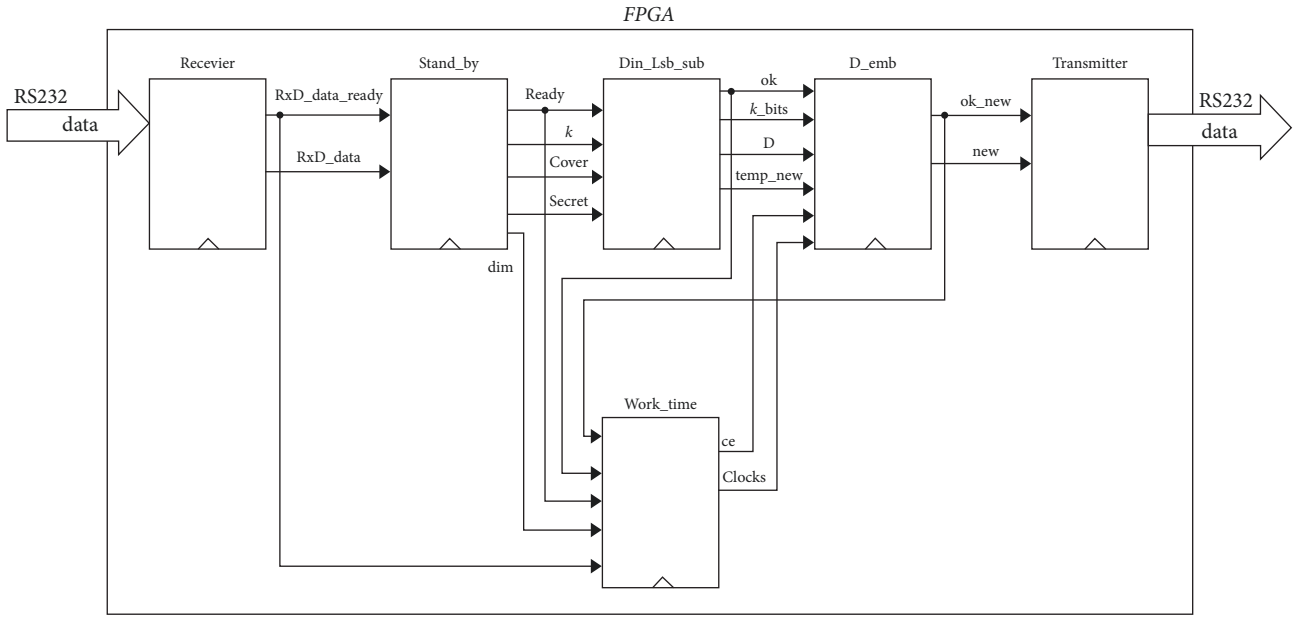

Figure 3: The structure of the hardware FPGA design.

2.3.1. Receiver Module. All information is transmitted from PC to the Receiver module of the FPGA board via RS232. The transmitted information consists of the cover image, the secret message, and the replacement width k. After receiving the information, the Receiver module removes the header from the cover image, followed by passing information to the next module. The output of the Receiver module consists of *RxD_data* and *RxD_data_ready*, where the *RxD_data* includes the raw data of the cover image, the secret message, and the replacement width k, the *RxD_data_ready* signal triggers the Stand_by module to receive data.

2.3.2. Stand_by Module. In the Stand_by module, a finite state machine (FSM) is designed to save the received data and transmit a set of embedding information to the next modules. There are four states in the FSM:

First state: it saves the dimension information, $M \times N$, of the cover image.

Second state: it saves the replacement width k.

Third state: it saves the cover image.

Fourth state: it saves the secret message and transmits all information to the next modules.

The output of the Stand_by module consists of five output signals: *ready*, k, *cover*, *secret*, and *dim*, which represented the data ready trigger, replacement width k, raw data of the cover image, the secret message, and the dimension $M \times N$ of the cover image, respectively. In our embedding procedure, the cover image is transmitted pixel by pixel, accompanied with k-bit secret message. The replacement width k, a cover pixel, and k-bit secret message will be transmitted to the Din_Lsb_sub module. The dimension information, $M \times N$, will be transmitted to the work_time module for calculating the overall processing time.

2.3.3. Din_Lsb_sub Module. The functions of the Din_Lsb_sub module are to operate the k-bit LSB substitution and calculate the embedding error. The Din_Lsb_sub module receives a cover pixel p_i, k-bit secret message, and the replacement width k in one clock cycle. The cover pixel p_i is firstly embedded with k-bit secret message by the LSB substitution method to get p_i'. After calculating the embedding error $\delta_i = p_i' - p_i$, the Din_Lsb_sub module transmits the replacement width k, the LSB substitution result p_i', and the embedding error δ_i to the D-emb module. The output of the Din_Lsb_sub module consists of four output signals: *OK*, *k_bits*, *D*, and *temp_new*, which represented the data ready trigger, replacement width k, the embedding error δ_i, and the substitution result p_i', respectively.

2.3.4. D_emb Module. The D_emb module uses a comparator to locate δ_i to its corresponding interval and process with associated calculations according to (3) to produce the stego pixel p_i'' and transmits the stego pixel p_i'' to the Transmitter module. The output of the D_emb module consists of two output signals: *ok_new and new*, which represented the data ready trigger and the stego pixel p_i'', respectively.

2.3.5. Transmitter Module. The Transmitter module transfers the stego pixel p_i'' into a bit stream and transmits the processing result to the PC via RS232.

2.3.6. Work_Time Module. The Receiver module, Stand_by module, Din_Lsb_sub module, and D_emb module keep sending *RXD_data_ready*, *ready*, *ok*, *and ok_new signals*, respectively, to the Work_time module thorough their execution period. The Work_time module records the execution time of each module and transfers the recorded data into a bit stream. After dealing with all pixels of the cover image,

Table 1: PSNR of Lena by using OPAP [11].

Cover image	k	PSNR (dB) OPAP	LSB
Lena	1	51.1410	51.1410
	2	46.3699	44.1519
	3	40.7271	37.9234
	4	34.8062	31.7808

the Work_time module will transmit the bit stream back to PC for analyzing the total execution time.

3. Experimental Results

To illustrate the performance of the proposed system, the following conditions are considered. The block diagram of the overall system is illustrated in Figure 3, where the FPGA is the controller and the input/output interfaces. Our FPGA is Spartan 3E Starter Kit by Xilinx, programmed with Verilog language to develop the proposed algorithm. The PC we used in our experiments is a Core i5, 4/4 (core/thread) CPU with 16 G RAM. The processed images are stored in a 7200 rpm hard disk.

We test the proposed FPGA structure on a series of cover images and secret messages. Experimental results show the difference between the proposed design and software implementation to compare the correctness and efficiency. In the first experiment, we use "Lena" with size 256 × 256 pixels to be the cover image, as shown in Figure 4(a). The secret message is an image "Mandril" in size 128 × 256 pixels, as shown in Figure 4(b). The stego images produced by software implementation and by the proposed FPGA implementation are shown in Figures 4(c) and 4(d), respectively. The secret messages extracted by software implementation and by the proposed FPGA implementation are shown in Figures 4(e) and 4(f), respectively. We used the replacement width $k = 4$ in Figures 4(c)–4(f) to compare the performance. Since a pixel is usually represented by 8 bits, a cover image can embed a secret image of its half size when the replacement width k is set to 4.

Another experiment is conducted to further examine the efficiency of the proposed method. The cover image shown in Figure 5(a) is "Girl" in size 256 × 256 pixels, and the secret message shown in Figure 5(b) is an image "Room," in size 128 × 256 pixels. The stego images produced by software implementation and by the proposed FPGA implementation are shown in Figures 5(c) and 5(d), respectively. The secret messages extracted by software implementation and by the proposed FPGA implementation are shown in Figures 5(e) and 5(f), respectively. All pictures in Figures 5(c)–5(f) are obtained by setting $k = 4$. Experimental results show that both the software and hardware implementations produce the same stego image.

Table 2 shows the execution time and speedup for embedding a secret image "Mandril" of size 128 × 256 pixels to a cover image "Lena" of size 256 × 256 pixels, by both the software based method and the proposed FPGA scheme. Table 3 shows the execution time and speedup for embedding a secret image "Room" of size 128 × 256 pixels to a cover image "Girl" of size 256 × 256 pixels, by both the software based method and the proposed FPGA scheme. Both tables show that the execution time for software based method increases with the larger replacement width k, and the execution time for FPGA scheme is almost the same with different values of k. In both cases, the speedup can be up to 15 when k is set to 4. Experimental results show that the outputs of these two implementation methods are identical.

Table 2: Comparison of execution time and speedup for embedding "Lena."

k	Execution time (seconds) software	FPGA	Speedup
1	0.116042	0.013110	≈8.85
2	0.120556	0.013110	≈9.20
3	0.143214	0.013110	≈10.92
4	0.199567	0.013110	≈15.23

Table 3: Comparison of execution time and speedup for embedding "Girl."

k	Execution time (seconds) Software	FPGA	Speedup
1	0.116482	0.013110	≈8.88
2	0.122311	0.013110	≈9.33
3	0.145358	0.013110	≈11.09
4	0.204881	0.013110	≈15.63

The device utilization used in these two experiments is shown in Table 4. Less than 10% of the hardware devices are used in the procedure of embedding a secret image of size 128 × 256 pixels to a cover image of size 256 × 256 pixels. Since the proposed scheme embeds secret image in a pixel-by-pixel manner, no extra hardware devices are needed even in embedding image of larger size. After observing the fact that the proposed pixel-wised system used less than 10% of the hardware devices, we further extend the proposed structure to a parallel structure which operates to embed 10 pixels at the same time. The parallel structure consists of one Receiver module, one Stand_by module, ten Din_Lsb_sub modules, ten D_emb modules, one Transmitter module, and one Work_time module. The device utilization of the 10-pixel parallel structure is shown in Table 4. We use larger size of images to test the efficiency of the parallel embedding structure. Figure 6 shows the experimental result of embedding the secret image "Field" of size 512 × 1024 into the cover image "Tulips" of size 1024 × 1024. Both the pixel-wise structure and the 10-pixel parallel structure produce the same stego image. The comparison of the execution time of the pixel-wise structure and the 10-pixel parallel structure is shown in Table 5.

4. Conclusions

We have proposed a hardware circuit design to carry out the image data hiding method on FPGA. Through the mechanism of the hardware design, the data hiding process

(a) (b) (c) (d) (e) (f)

Figure 4: The experimental results of embedding "Mandril" in "Lena." (a) The cover image "Lena" of size 256 × 256; (b) secret image "Mandril" of size 128 × 256; (c) the stego image produced by software implementation; (d) the stego image produced by the proposed FPGA design; (e) the secret message extracted by software implementation; (f) the secret message extracted by the proposed FPGA design.

Table 4: Device utilization of the proposed FPGA design.

System structure		Pixel-wised structure		10-pixel parallel structure	
Hardware devices	Available	Used	Utilization	Used	Utilization
Slices	4650	270	5%	1980	43%
Slice flip flop	9312	239	2%	2390	20%
4 input LUTs	9312	340	3%	3400	30%
Bounded IOBs	232	4	1%	40	10%
BUFGMUXs	24	2	8%	20	80%
MUFT18X18SIOs	20	2	10%	20	100%

(a) (b) (c) (d) (e) (f)

Figure 5: The experimental results of embedding "Room" in "Girl." (a) The cover image "Girl" of size 256 × 256; (b) secret image "Room" of size 128 × 256; (c) the stego image produced by software implementation; (d) the stego image produced by the proposed FPGA design; (e) the secret message extracted by software implementation; (f) the secret message extracted by the proposed FPGA design.

Table 5: The execution time of the pixel-wise structure and the 10-pixel parallel structure.

	Cover image "Girl" of size 256 × 256, with secret image "Room" of size 128 × 256			Cover image "Tulips" of size 1024 × 1024, with secret image "Field" of size 512 × 1024		
	Pixel-wise structure Execution time (seconds)	10-pixel parallel structure Execution time (seconds)	Speedup	Pixel-wise structure Execution time (seconds)	10-pixel parallel structure Execution time (seconds)	Speedup
$k = 4$	0.013110	0.001680	7.8	0.210302	0.025910	8.1

(a) (b) (c)

Figure 6: The experimental results of embedding "Field" in "Tulips." (a) The cover image "Tulips" of size 1024 × 1024; (b) secret image "Field" of size 512 × 1024; (c) the stego image produced by both hardware structures.

can be completed in fewer clock cycles to produce outcome much faster. Experimental results show the effectiveness and correctness of the proposed scheme. The speedup of the execution time is up to 15 for the proposed scheme over the software implementation scheme. The proposed scheme embeds secret image in a pixel-by-pixel manner. The advantage of this design is that there is no limitation on the size of the secret message and cover image, since no extra hardware devices are needed in embedding large-size images. The disadvantage of this design is that we cannot fully use the hardware devices to speed up the embedding procedure. Moreover, we have extended the basic pixel-wise structure to a parallel structure which can fully use the hardware devices to speed up the embedding process and embed several bits of secret message at the same time. The maximum embedding storage is constrained by the size of the cover image and the replacement width k. For an image of size $M \times N$ and replacement width k, the maximum embedding storage will be $M \times N \times k$. The limitation of the size of the cover image depends on the size of the memory to store the cover image. The future research includes applying the proposed prototype to develop a fully automated embedded system which is designed particularly for data hiding and independent from PC.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] R. Poornima and R. J. Iswarya, "An overview of digital image steganography," *International Journal of Computer Science & Engineering Survey*, vol. 4, no. 1, pp. 23–31, 2013.

[2] B. Kaur, A. Kaur, and J. Singh, "Steganographic approach for hiding image in DCT domain," *International Journal of Advances in Engineering & Technology*, vol. 1, no. 3, pp. 72–78, 2011.

[3] J. Song, Y. Zhu, and J. Song, "Steganography: an information hiding method base on logistic map in DCT domain," *Advances in Information Sciences and Service Sciences (AISS)*, vol. 4, no. 2, 2012.

[4] Q. Gu and T. Gao, "A novel reversible watermarking algorithm based on wavelet lifting scheme," *ICIC Express Letters*, vol. 3, no. 3, pp. 397–402, 2009.

[5] A. A. Sheju and U. L. Kulkarni, "A secure skin tone based steganography using wavelet transform," *International Journal of Computer Theory and Engineering*, vol. 3, no. 1, pp. 1793–8201, 2011.

[6] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for data hiding," *IBM Systems Journal*, vol. 35, no. 3-4, pp. 313–335, 1996.

[7] X. Zhang, "Reversible data hiding with optimal value transfer," *IEEE Transactions on Multimedia*, vol. 15, no. 2, pp. 316–325, 2013.

[8] R.-Z. Wang, C.-F. Lin, and J.-C. Lin, "Image hiding by optimal LSB substitution and genetic algorithm," *Pattern Recognition*, vol. 34, no. 3, pp. 671–683, 2001.

[9] C.-K. Chan and L. M. Cheng, "Hiding data in images by simple LSB substitution," *Pattern Recognition*, vol. 37, no. 3, pp. 469–474, 2004.

[10] W. Wang, J. Ye, T. Wang, and W. Wang, "Reversible data hiding scheme based on significant-bit-difference expansion," *IET Image Processing*, vol. 11, no. 11, pp. 1002–1014, 2017.

[11] C.-K. Chan and L. M. Cheng, "Improved hiding data in images by optimal moderately-significant-bit replacement," *IEEE Electronics Letters*, vol. 37, no. 16, pp. 1017-1018, 2001.

[12] C.-C. Chang, C.-C. Lin, and Y.-H. Chen, "Reversible data-embedding scheme using differences between original and predicted pixel values," *IET Information Security*, vol. 2, no. 2, pp. 35–46, 2008.

[13] W.-L. Tai, C.-M. Yeh, and C.-C. Chang, "Reversible data hiding based on histogram modification of pixel differences," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 6, pp. 906–910, 2009.

[14] P. Tsai, Y.-C. Hu, and H.-L. Yeh, "Reversible image hiding scheme using predictive coding and histogram shifting," *Signal Processing*, vol. 89, no. 6, pp. 1129–1143, 2009.

[15] J.-L. Zhang, Q. Wu, Y.-P. Ding et al., "Techniques for Design and Implementation of an FPGA-Specific Physical Unclonable Function," *Journal of Computer Science and Technology*, vol. 31, no. 1, pp. 124–136, 2016.