Development of a Secure Lossless Model for Image Steganography and its Hardware Realization for Real-Time Applications

Salah Harb

A Thesis
in
The Department
of
Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy (Electrical and Computer Engineering) at
Concordia University
Montréal, Québec, Canada

August 2023

CONCORDIA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Salah Harb

Entitled: Development of a Secure Lossless Model for Image Steganography and its Hardware Realization for Real-Time Applications

and submitted in partial fulfillment of the requirements for the degree of

Doctor Of Philosophy (Electrical and Computer Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. Carol Fung

_____ External Examiner
Dr. Chamseddine Talhi

_____ Examiner
Dr. Chunyan Wang

_____ Examiner
Dr. Wei-Ping Zhu

_____ Examiner, External to Program
Dr. Amr Youssef

_____ Thesis Co-Supervisor
Dr. M. Omair Ahmad

_____ Thesis Co-Supervisor
Dr. M.N.S. Swamy

Approved by _____
Dr. Jun Cai, Graduate Program Director

8/21/2023 _____
Dr. Mourad Debbabi, Dean
Gina Cody School of Engineering and Computer Science

Abstract

Development of a Secure Lossless Model for Image Steganography and its Hardware Realization for Real-Time Applications

Salah Harb, Ph.D.
Concordia University, 2023

Image steganography is a promising security technology through which secret data is securely embedded in an image and then retrieved at the receiver upon receipt of the resulting image. The secret data is concealed in the cover image in a way such that it is invisible, and the resulting stego image is transmitted to an authorized receiver. The receiver has a key that enables extraction of the secret data from the stego image through a technique called recovery process. The design of a good image steganographic scheme aims at achieving three desirable characteristics: high embedding rate, imperceptibility of the secret data in the stego image, and robustness against attacks. Imperceptibility refers to the ability to maintain the quality of the cover image in the generated stego image after embedding the secret data. Embedding rate refers to the amount of secret data that is embedded in a single pixel of the stego image. Robustness against attacks refers to the ability to resist noise that could affect the stego image or steganalysis, ensuring that unauthorized receivers are not able to recover secret data. Existing image steganographic schemes lack providing both high embedding rate and high-quality stego image simultaneously as these are two mutually conflicting requirements. Moreover, robustness against attacks in these schemes is achieved using a high-complexity cryptographic algorithm that increases the time complexity of the image steganography scheme.

In this thesis, our objective is to design and implement a secure and lossless model for image steganography that has the three desirable features mentioned above. To achieve this objective, in the first part of the thesis (Chapters 3 and 4), we develop two novel modulus-based image steganographic algorithms that combine the benefits of two fundamental modulus-based schemes, namely, EMD and DE, resulting in the embedding rate and the quality of the stego image that are superior to that of the individual schemes. In the first algorithm proposed, the two embedding schemes are combined to embed a block of the secret data, whereas in the second algorithm proposed, each block of the secret data is first divided into two parts and then the first part is concealed by using one scheme and the second part by using the other scheme. To enhance the robustness of the proposed algorithms, the pixels of the cover image are first permuted using a low-complexity permutation mechanism prior to the concealment of the secret data. In the second part of the thesis (Chapters 5 and 6), hardware realizations of the two steganographic schemes developed in the first part of the thesis are provided using a reconfigurable platform.

In the third part of the thesis (Chapter 7), a secure lossless image steganographic model, consisting of the modules for encryption, embedding, and recovery, is presented, and a scheme for its efficient hardware implementation is proposed. To this end, first efficient hardware realizations for the encryption and recovery modules are developed. In the encryption module, secret data is made more secure by encrypting it using the hardware implementation of the private-key advanced encryption standard

cryptosystem. In the recovery module, the quality of the cover image is preserved by encrypting the modified pixels and their original values using a hardware implementation of the public-key elliptic curve and Paillier cryptosystems. The hardware realizations of these two modules and that of the embedding module, designed in the second part of the thesis, are finally integrated together on an AMD Xilinx Zynq-7000 all programmable SoC platform. The efficiency and usefulness of the hardware implementation of the proposed steganographic model is demonstrated by considering the task of reducing the storage requirement for image data.

Acknowledgments

All praises to Almighty ALLAH who enabled me to embark on and successfully complete my PhD journey. This thesis would have never come to fruition without the will and blessings of ALLAH, the most gracious, the most merciful.

It has been an immense privilege and the greatest honor of my academic life to work closely with my esteemed thesis supervisors, Dr. M. Omair Ahmad and Dr. M.N.S. Swamy. Their guidance, expertise, and unwavering support throughout this research endeavor have been invaluable. I am deeply grateful for their patience, wisdom, and the countless hours they dedicated to shaping my thesis. Their profound knowledge and experience have enriched my understanding and have inspired me to achieve more than I ever thought possible.

I would also like to extend my heartfelt appreciation to the members of my advisory thesis committee. Their insightful comments, constructive criticism, and invaluable suggestions have significantly contributed to the refinement and improvement of this thesis. I am sincerely grateful for their time, effort, and commitment to my academic growth.

To my beloved family, I am eternally grateful for your unwavering support and encouragement throughout my academic pursuits and endeavors. To my mother Najah, my father Sami, my brothers Tariq, Khaled, Harb, and Hamza, and my sister Randa, you have been my pillars of strength. Your love, sacrifices, and belief in my abilities have been the driving force behind my success. I am forever indebted to you for instilling in me the values of perseverance, determination, and resilience.

Finally, I would like to express my gratitude to all my friends and colleagues who have been there for me, offering their support, encouragement, and camaraderie throughout this journey. Your friendship and companionship have made this experience even more meaningful and enjoyable.

Contents

List of Figures

List of Tables

Acronyms

AES	Advanced Encryption Standard
RSA	Rivest-Shamir-Adleman
DES	Data Encryption Standard
ECC	Elliptic Curve Cryptography
SPM	Scalar Point Multiplication
ECDH	Elliptic Curve Diffie-Hellman
ECDS	Elliptic Curve Digital Signature
GF	Galois Field
DSA	Digital Signature Algorithm
NIST	National Institute of Standard and Technology
LD	Lopez-Dahab
Rcon	Round Constant
LSB	Least Significant Bit
EXIF	Exchangeable Image File
XMP	Extensible Metadata Platform
PVD	Pixel Value Difference
HVS	Human Vision System
EMD	Exploiting Modification Direction
DE	Diamond Encoding
DFT	Discrete Fourier Transform
DWT	Discrete Wavelet Transform
IWT	Integer Wavelet transform
UMIS	Unified Modulus-based Image Steganographic
UMISPB	Unified Modulus-based Image Steganographic with Partitioned Block
BPP	Bit Per Pixel
PSNR	Peak Signal-to-Noise Ratio
SSIM	Structural Similarity Index Measure
FPGA	Field Programmable Gate Array
IP	Intellectual Property
SoC	System on Chip
APSoC	All-Programmable System on Chip
RP	Recovery Point
RISC	Reduced Instruction Set Computer
UART	Universal Asynchronous Receiver/Transmitter

Symbols

mod	Modulus Operation
GF	Galois Field
kP	Scalar Point Multiplication
$P(x, y)$	Point on Elliptic Curve
MUL	Multiplication Process
HW	Hamming Weight
$MontP$	Montgomery Product Operation
b^e	A Base b Raised to Exponent e
r	Random Integer
λ	Private Key
GCD	Great Common Divisor
LCM	Least Common Multiple
T	Execution Time
m	Number of Group of Pixels in EMD Scheme
k	Embedding Parameter in DE Scheme
$(2m+1)-ary$	Notational System of EMD Scheme
$(2k^2+2k+1)-ary$	Notational System of DE Scheme
F_{EMD}	Embedding Function in EMD Scheme
F_{DE}	Embedding Function in DE Scheme
F_{EXT_EMD}	Extracting Function in EMD Scheme
F_{EXT_DE}	Extracting Function in DE Scheme
bpp	Rate in Bit Per Pixel
dB	Peak Signal-to-Noise Ratio in Decibel
$P_1 = 2 \cdot P_0$	Point doubling Operation
$P_2 = P_0 + P_1$	Point Addition Operation
E_{curve}	Nonesingular Curve
Z_n	Set of Integers $mod\ n$
Z_n*	Multiplicative Group of Integers $mod\ n$

Chapter 1

Introduction

In recent years, the widespread digitization of content has permeated numerous sectors of the economy and social spheres, necessitating the addressing of potential security and privacy concerns. The proliferation of digitized content has specifically raised issues related to content authentication and the protection of ownership rights. Additionally, with the advancement of digital communication and computer technologies, the assurance of data confidentiality and privacy in internet communication channels has become increasingly challenging, thus highlighting the need for secure communication channels that utilize algorithms. This has given rise to the establishment of a new principle in information security known as information hiding. Steganography emerges as a novel addition to the field of information hiding, serving as a methodology for concealing secret data within a carrier in the form of digital media such as images, audio, or video. An image steganographic scheme, in particular, operates by concealing secret data within a digital image as the carrier. The image steganographic scheme holds significant importance in various real-time applications, including data privacy, content authentication, copyright protection, and biometric data protection.

1.1 Image Steganography

Transmitting secret data securely to authorized endpoints over public communication networks is a challenging task. Several algorithms have been introduced over the last few decades for this purpose. In contrast to cryptography, where data is changed (scrambled) from its original form (plaintext) into a meaningless form (ciphertext) [1], steganography is an information hiding methodology that ensures the secure transmission of data in digital media such as images, audio, or video [2]. A cryptographic algorithm mainly has two processes: encryption and decryption. In the encryption process, plaintext is changed into ciphertext, whereas in the decryption process, plaintext is recovered from the ciphertext, usually in a way that is reverse to the encryption process [1]. On the other hand, the image steganographic scheme comprises two processes: embedding and recovery. In the embedding process, the sender conceals the secret data within the cover image by performing specific operations, resulting in a stego image that is transmitted to the receiver through a communication channel. With the assistance of the shared private key with the sender, the receiver applies the recovery process by performing the inverse operations of the embedding process to extract the concealed secret data. Figure 1.1 shows the generic framework of the image steganographic scheme.

The design of an image steganographic scheme is concerned with achieving three key characteristics: imperceptibility, embedding capacity, and robustness against attacks. Imperceptibility refers to the ability of the generated stego image to maintain the visual quality of the cover image. Embedding capacity refers to the amount of secret data that

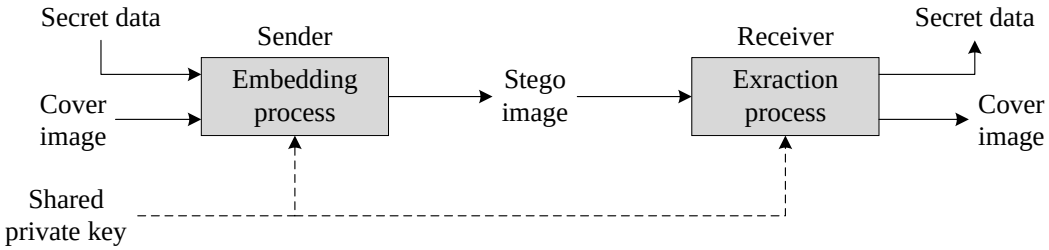


Figure 1.1: Generic framework of the image steganoghraphic scheme.

can be concealed in the stego image. Robustness against attacks could involve resilience to noise, allowing hidden secret data to be retrieved by authorized receivers even when the stego image is corrupted by noise, or being secure against steganalysis, preventing unauthorized receivers from detecting or retrieving the hidden secret data after it has been discovered. An image steganographic scheme is said to be lossless if the original cover image can be recovered after the extraction of the embedded data at the receiver.

In contrast to image steganography, steganalysis is the art of detecting whether a stego image contains hidden secret data and, if possible, extracting that data. During transmission, stego images may undergo various image processing operations, such as format conversion, compression, or malicious attacks by unauthorized receivers (i.e., attackers). Attackers launch these attacks to remove the hidden secret data and can do so by adding noise or cropping the image. These unauthorized operations typically occur when the attacker suspects that the stego image contains secret data.

In image steganography, digital images have been used in many practical real-time applications, which has created a strong need for image steganographic schemes to achieve the desired outcomes from these applications. An image steganographic scheme is evaluated, and its performance is assessed before being employed in an application. The performance metrics used to evaluate imperceptibility are the peak signal-to-noise ratio (PSNR) and the structural similarity index measure (SSIM) [3], whereas the bit per pixel (*bpp*) rate is used for evaluating embedding capacity. Higher PSNR/SSIM values indicate greater imperceptibility, while a higher *bpp* suggests the ability to conceal more data within a single pixel, meaning a larger capacity can be carried by a single cover image. Robustness against attacks can be evaluated by applying various steganalysis techniques. A successful attack indicates that the scheme is vulnerable; otherwise, the scheme is considered robust.

1.2 Literature Review

Over the last decade, there have been significant advancements in digital image processing and computational power, resulting in the creation of increasingly complex digital image processing systems. These advancements have led to an increasing need for steganographic schemes that can be used to conceal secret data within digital images. The ease of performing operations on digital images has further highlighted the need for such schemes. Steganographic schemes can be applied in various fields, including copyright protection and privacy, and have been the focus of extensive research since the 1990s. The literature contains a multitude of works that have presented various image steganography schemes, indicating the significance of this field and the ongoing efforts to enhance its efficiency and security. Moreover, with the growing importance of information security and privacy, the development of efficient and secure steganographic schemes has become even more critical in recent years.

An irreversible steganographic scheme refers to a scheme where the process of

concealing secret data into a cover image results in a permanent alteration of the image that cannot be fully reversed. Once the secret data is concealed, it becomes impossible to recover the original, unaltered cover image. On the other hand, a reversible steganographic scheme carries out an alternative approach. In these schemes, the secret data can be concealed within the cover image while preserving the possibility of recovering the original, unaltered image. Reversible schemes provide a valuable solution in applications where the preservation of the cover image is critical, allowing for the recovery of the original image without any loss of information.

Steganographic schemes operate either in the spatial or frequency domain. In the spatial domain, the scheme directly modifies the values of pixels to conceal secret data [4]. On the other hand, a frequency-based steganographic scheme transforms the cover image to obtain its frequency spectrum representation, and then conceals bits of the secret data within the frequency coefficients of the sub-bands [5, 6]. Although the embedding and recovery processes are more complex compared to spatial-based schemes, frequency-based steganographic schemes are highly secure and improve robustness against attacks. They also exhibit high resilience, as resulting stego images can defeat histogram-based attacks [7] and other possible alterations such as rotations, compression, and cropping during transmission. There are numerous frequency-based steganographic schemes in the literature that conceal secret data in the frequency domain [7–11]. Many of these schemes use discrete cosine transform (DCT) [9], integer wavelet transform (IWT) [7], and discrete wavelet transform (DWT) schemes [10], [11].

This literature review provides a detailed discussion of spatial-based steganographic schemes [12–47].In this thesis, steganographic schemes can be classified into three main categories: interpolation, encryption, and modulus-based schemes. One of the most fundamental spatial-based schemes is the LSB-based scheme, which provides a simple way of concealing secret data in a cover image by replacing each bit of the secret data with the LSB of a pixel value in the cover image [48]. Another form of LSB-based scheme is LSB matching, which involves decreasing or increasing the value of a pixel when its LSB does not match the bit in the secret data, and leaving it unchanged otherwise [49]. The LSB-based scheme is easy to implement and provides high embedding capacity with imperceptibility in the resulting stego image. Researchers have developed many techniques and methods for inserting secret data into the LSBs of pixel values, including embedding into multiple planes to deliver higher embedding capacities [50].

Carrying bits of secret data in the values of pixels in the cover image can also be achieved by calculating the differences between pixels instead of directly inserting into the LSBs of pixels. This scheme is called pixel value difference (PVD) [51], [52]. In the PVD steganographic scheme, the cover image is divided into non-overlapping groups of two neighboring pixels, and secret data is concealed in the difference between the two pixels within each group. PVD-based steganographic schemes have gradually gained attention and are regarded as a step forward in the information hiding field because pixels of the cover image and concealed secret data can be completely restored from the stego image in the recovery process.

The main concept of the PVD based schemes has been incorporated into another image processing technique called interpolation. In the interpolation based steganographic scheme [16,19,23], the resolution of the cover image is scaled up to create extra neighboring pixels to conceal the secret data into the interpolated pixels and keep the original pixels unchanged. Bits of the secret data are concealed into the interpolated pixels by calculating differences between interpolated and original neighboring pixels. The interpolation based steganographic schemes provide either high embedding rates or good image qualities for the resulting stego images, but not both. These schemes do not have a provision for a trade-off between the embedding rate and the quality of the stego

image. Also, the computational complexity is related to the interpolation technique used. Generally, since a complex interpolation method is employed in these schemes in order to maintain the quality of the cover image, the embedding time of these schemes is large.

The use of encryption-based steganographic schemes has emerged as a new approach to enhance the security of the secret data against attacks. In these schemes [24,30,34,35], the secret data is encrypted either before or during the embedding process to ensure its confidentiality. These schemes also offer a provision for balancing the embedding rate and the quality of the resulting stego image. However, due to the use of cryptographic algorithms [53, 54], the computational complexity of these schemes is generally high.

The third category of steganographic schemes [37, 39, 40, 43–45, 47] utilizes the modulus operation for the embedding process. In this category, each block of the secret data is converted into a secret digit using a predefined (l)-ary notational system, which is then embedded into a group of pixels in the cover image. Compared to the other two categories, the modulus-based steganographic schemes provide better image quality for the resulting stego images while still offering a trade-off between image quality and embedding rate. Additionally, these schemes have less computational complexity because the embedding and extraction processes use simple modulus functions. The modulus-based schemes are also resilient against attacks such as PVD histogram analysis. Overall, the modulus-based steganographic scheme shows considerable improvements in image quality, embedding rate, and robustness against attacks.

This literature review is concluded by the following observations:

- Reversible image steganographic schemes are crucial for specific applications such as medical and military systems. These schemes ensure that the original cover image can be fully recovered.
- LSB-based schemes are highly vulnerable to steganalysis, but they offer great flexibility in terms of integration with other schemes.
- Image steganographic schemes operating in the spatial domain achieve a higher embedding capacity than those used in the frequency domain. However, frequency domain-based steganographic schemes are more resistant to steganalysis attacks. Therefore, if high security is required, it is highly encouraged to use steganographic schemes operating in the frequency domain.
- Most image steganographic schemes exhibit a trade-off between the embedding rate and the image quality of the resulting stego image. Higher embedding capacities come at the cost of lower imperceptibility, and vice versa.

The superior quality of stego images in modulus-based schemes is achieved by limiting the changes in the pixel values of the cover image. With the same embedding rate, the image quality achieved by these schemes is higher than that of schemes in the other two categories: interpolation and encryption. Furthermore, the complexity of modulus-based schemes is lower, as no complex operations are required to conceal the secret data. Overall, modulus-based schemes are the most promising in all aspects, except for the fact that the secret data is not as secure as in schemes of the second category.

Exploiting modification direction (EMD) [37] and diamond encoding (DE) [39] modulus-based schemes are fundamental schemes in the modulus-based category that use modulus functions to conceal secret data. The EMD scheme provides the highest embedding rate of 1.16 *bpp*, which happens to be the lowest embedding rate of the DE

scheme. It is also observed that at this rate, both schemes have approximately the same image quality, with a PSNR value of about 52.1 dB.

1.3 Problem Statement and Motivation

The literature review suggests that the EMD and DE schemes in the modulus-based category provide stego images of superior quality compared to the stego images obtained from schemes in the other two categories interpolation and encryption-based schemes. For a given L, the EMD scheme gives a stego image of a certain quality and embedding rate, and similarly, the DE scheme also provides a stego image of a different quality and embedding rate. If the value of L is decreased, the image quality decreases and the embedding rate increases in the EMD scheme, whereas in the DE scheme, the change in the quality and embedding rate of the stego image is opposite to that of the EMD scheme.

In the EMD scheme, each block of size of L bits of the secret data is converted into E secret digits using a $(2b+1)$-ary notational system. If each secret digit is concealed in a group of m pixels in the cover image, then the value of b in the notational system is m. In the EMD scheme, for each secret digit, only one pixel in the group is modified by changing its value by ± 1 or remain unchanged. In the EMD scheme, higher values of m provide higher qualities of the stego images and lower embedding rates with maximum value of 1.16 *bpp* when m is 2. The DE scheme was proposed to deliver higher embedding rates than the rates achieved by the EMD scheme for large sizes of secret data blocks. In the DE scheme, a secret digit represented using the $(2k^2+2k+1)$-ary notation system is concealed in a group of two pixels only. Increasing L requires higher values of k. Higher values of k result in higher embedding rates with minimum value of 1.16 *bpp* when k is 1. However, a higher k deteriorates the quality of the stego image when there is change to the values of the two pixels by $\pm k$.

In both of the EMD and DE schemes, like any other steganographic scheme, as the quality of the stego image is improved, the embedding rate is decreased. In the EMD scheme, the maximum embedding rate is 1.16 *bpp*, while by using the DE schem it is possible to achieve an embedding rate higher than 1.16 *bpp*. On the other hand, the maximum image quality in terms of PSNR achievable by the DE scheme is 52.1 dB, a value that can be exceeded by using the EMD scheme.

The EMD and DE schemes are not robust against steganalysis attacks for several reasons. Firstly, these schemes employ modulo arithmetic operations to modify the pixel values of the cover image, which can introduce detectable statistical irregularities. Steganalysis techniques can leverage these irregularities to potentially uncover the concealed secret data. Additionally, the alterations made by the EMD and DE schemes are not specifically designed to withstand advanced steganalysis algorithms that are specifically crafted to detect and extract hidden data. These schemes lack additional layers of complexity that could provide stronger resistance against such attacks. Moreover, the simplicity of the EMD and DE schemes can render them more vulnerable to known steganalysis attacks. These attacks can exploit the specific patterns or properties of these schemes, resulting in higher accuracy in detecting the presence of hidden data.

The EMD and DE belong to the class of irreversible steganographic schemes. These schemes utilize modulo arithmetic operations to modify the pixel values of the cover image, enabling the concealing of a secret digit that represent the secret data block of size L. Consequently, the receiver is unable to recover the original cover image. Nevertheless, the receiver can successfully extract the concealed secret data. The irreversible nature of the operations employed in the EMD and DE steganographic schemes ensures the

concealment of the secret data, but at the cost of irreversibility of the original cover image.

Regarding the hardware implementation of the image steganographic schemes, the results of recent hardware implementations have indicated that more the complexity of a steganographic scheme is, more are the resources that it consumes [5], [55]. The hardware architectures of the steganographic schemes [5, 55–61] do not take full advantage of the optimization techniques available in FPGA platforms. As a result, they experience high delays and low throughputs. Techniques such as parallelism, where multiple tasks are executed simultaneously resulting in low delays [62], and pipelining, where stages of processing overlap to improve throughputs, are not fully leveraged in these architectures.

1.4 Objectives

In this thesis, our primary objective is to develop and implement a secure, lossless model for image steganography, utilizing a reconfigurable hardware-based platform. The model consists of three modules: embedding, encryption, and recovery. The main focus of this research is on the embedding module, which serves as the core of image steganography. In the embedding module, a key aspect of our investigation is to develop an image steganographic scheme that addresses three essential requirements of image steganography: (i) the ability to conceal a large amount of secret data, (ii) the generation of high-quality stego images, and (iii) the provision of robustness against steganalysis techniques. By fulfilling these requirements, our research would contribute to the advancement of the field of secure and reliable information hiding.

One effective strategy is to develop a steganographic scheme that achieves a balance between image quality and embedding rates while demonstrating resilience against steganalysis attacks. Additionally, it should outperform both the EMD and DE schemes in terms of image quality and embedding rates. This can be achieved by employing a fusion of the EMD and DE steganographic schemes. This fusion approach takes advantage of the unique strengths of each scheme, resulting in a novel hybrid scheme that offers improved performance compared to using either the EMD or DE schemes individually. By fusing the EMD and DE schemes, it becomes possible to achieve a balance between image quality and embedding rate. The EMD scheme prioritizes high image quality by focusing on preserving imperceptibility, while the DE scheme emphasizes a high embedding rate by optimizing the capacity for hidden data. The fusion of these two schemes allows for a synergistic integration of their advantages, resulting in an image steganographic scheme that provides both better stego image qualities and high embedding rates.

The resilience level of the EMD and DE steganographic schemes can be improved by adopting a mechanism that makes concealing the secret data into pixels of the cover image more secure, without relying on cryptographic algorithms. One such low-complexity mechanism, is permuting the locations of pixels in an image before the embedding process, which has been utilized by researchers primarily for breaking the high correlation between adjacent pixels [63–66].

An image steganographic scheme in this thesis goes through two phases: software development and hardware implementation. In the software development phase, the image steganographic scheme is developed and its performance is evaluated. In the hardware implementation phase, a reconfigurable hardware-based platform is utilized to implement the developed image steganographic scheme.

In this thesis, we develop two novel modulus-based steganographic schemes that combine the EMD and DE schemes. In the first modulus-based steganographic scheme, the secret data block of size L is divided into two parts: digits representing the first part

L_1 are concealed using the EMD scheme, and digits of the other part L_2 are concealed using the DE scheme. In the second modulus-based steganographic scheme, dividing secret data block is not required. It involves searching for the secret data block of size L in a pre-constructed lookup tables, and concealing the corresponding secret digits representing that secret data block.

Implementing the developed image steganographic schemes is carried out on the reconfigurable hardware-based platform. Reducing the computational delay and increasing the throughput are the most crucial tasks when processing image steganographic schemes in real-time applications. These critical issues are efficiently addressed by realizing these schemes on the reconfigurable hardware-based platform. The hardware realization of the image steganographic scheme has to: (i) provide high operating speeds, (ii) deliver high througputs, and (iii) consume less resources. To achieve that, an efficient hardware implementation of the image steganographic scheme should support the following:

1. Architectural pipelining techniques for higher operating speeds.
2. Parallelism for independent computations.
3. Resource sharing for a compact design (e.g., utilizing fewer resources).
4. Integrated high-speed communication interfaces for high transfer rates.

This thesis focuses on developing and implementing the embedding module of image steganographic schemes on a reconfigurable hardware-based platform. Once the embedding module is implemented, the encryption and recovery modules are subsequently implemented. The encryption module employs the private-key advanced encryption standard (AES) cryptosystem to secure the secret data before embedding. The encryption module strengthens the overall security of the secret data, preventing unauthorized access even if the stego image is subjected to steganalysis. The recovery module ensures a lossless model by registering the original and modified pixel values in a recovery list. By utilizing a specific public-key cryptosystem, the authorized receiver can securely restore the original pixel values, preserving the integrity and quality of the reconstructed image. The recovery module employs two public-key cryptosystems, elliptic curve (EC) and Paillier.

In this thesis, the secure lossless model, comprising three modules, is integrated and implemented on the reconfigurable AMD Xilinx Zynq-7000 all programmable system-on-chip (APSoC) platform.

1.5 Thesis Organization

The rest of the thesis is organized as following: Chapter 2 provides a detailed explanation of the EMD and DE modulus-based steganographic schemes, along with a comprehensive background on private and public-key cryptosystems. Additionally, it introduces the AMD Xilinx Zynq-7000 APSoC platform adopted in this thesis. In Chapter 3, we focus on the development of a novel modulus-based image steganographic scheme that does not involve dividing the secret data blocks. Chapter 4 describes the novel modulus-based image steganographic scheme that adopts a division approach for the secret data blocks. Chapter 5 introduces the hardware realization of the developed modulus-based image steganographic scheme proposed in Chapter 3. Chapter 6 introduces the hardware realization of the developed modulus-based image steganographic scheme proposed in Chapter 4. Chapter 7 presents the design of the reconfigurable embedded system for the secure lossless model of image steganography. This chapter integrates

all the reconfigurable hardware implementations, including the modulus-based image steganographic scheme proposed in Chapter 6, AES cryptosystem, and EC/Paillier cryptosystems. Finally, Chapter 8 concludes the thesis and discusses potential avenues for future research and improvements in the field of image steganography.

Chapter 2

Related Work

Exploiting modification direction (EMD) [37] and diamond encoding (DE) schemes [39] are two fundamental steganographic schemes belonging to the modulus-based category on which the steganographic schemes proposed in this thesis are based. This chapter first provides a more detailed review of these two schemes. Towards robustness of the proposed schemes in this thesis, locations of the pixels within the cover image are permuted by a chaos-based permutation mechanism. Therefore, a brief review of the chaos-based permutation mechanism is carried out. To make the model for image steganography more secure, secret data is encrypted using the private-key AES cryptosystem [67]. Hence, a brief review of the AES cryptosystem is presented in this chapter. To recover the original cover image after extracting secret data, modified pixels durinng the concealing process are stored and encrypted using public-key ellipic curve (EC) [68] and Paillier [69] cryptosystems. Therefore, a brief of the EC and Paillier cryptosystems is introduced. Since the proposed schemes of image stegonagrphy and the secure lossless model are implemented on a reconfigurable hardware platform, a description of the AMD Xilinx Zynq-7000 all-programmable system-on-chip (APSoC) platform is also included in this chapter.

2.1 Exploiting Modification Direction (EMD) Scheme for Image Steganography

The demand for a scheme that provides high embedding capacities, maintains the quality of a cover image, and shows robustness against attacks has always been a consideration in the research community. In 2006, a novel steganographic scheme emerged, introduced by Zhang and Wang in [37]. The scheme is called exploit modification direction (EMD) and uses modulo operations to conceal secret data. In the EMD scheme, the secret data is concealed using a $(2m+1)$-ary notational system. Each block of secret data is expressed as a secret digit, E, in the $(2m+1)$-ary notational system. The secret digit is hidden in a group of m consecutive pixels in a predetermined order of the pixels used for embedding. Only one of the pixels in this group is modified, with its value being increased or decreased by one or remaining unchanged. The pixel values in the group can be modified in $(2m)$ different ways. The embedding process in EMD starts by calculating the value of the weighted sum modulo function given by

$$F_{EMD} = \sum_{i=1}^{m} (i \ g_i) \ mod \ (2m+1) \tag{2.1.1}$$

where g_i is the pixel value of the ith pixel in a group of m pixels. The value of F_{EMD} thus obtained ranges between 0 to $2m$. Next, the difference, s, between E and F_{EMD} is

used to determine which pixel in the group is modified. The pixel modification is done as follows. (i) if the difference s is 0, then no modification is made to any of the pixels in the group; (ii) if s is less than or equal to m, then the pixel value g_s is increased by 1; and (iii) if s is greater than m, then the pixel value g_{2m+1-s} is decreased by 1.

Figure 2.1 shows an example of all possible values of F_{EMD} for $m = 2$, where the group contains only two pixels, and the values g_1 and g_2 are represented using 8-bit representation. Figure 2.2 shows all the possible modifications in the pixel values of a group of two pixels ($m = 3$) and three pixels ($m = 3$).

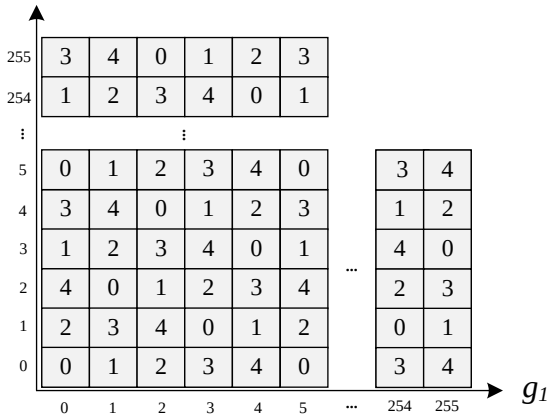
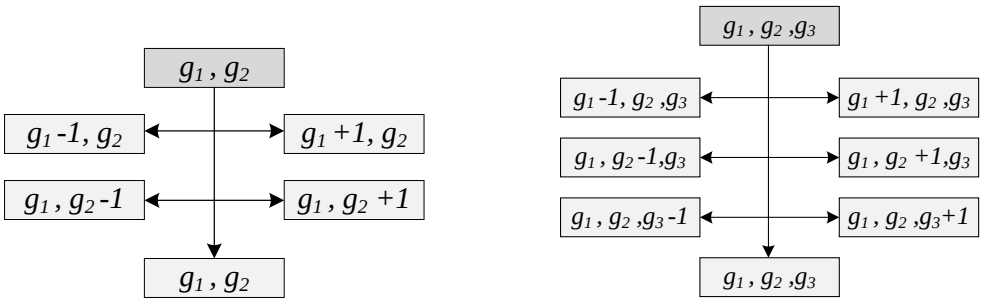


Figure 2.1: All possible F_{EMD} values when $m = 2$.



(a) $m = 2$ (b) $m = 3$

Figure 2.2: Different ways for pixels to get modified.

The secret digit E is extracted from the modified group by using the weighted sum modulo function given by

$$E = F_{EXT\_EMD} = \sum_{i=1}^{m} (i \; g_i^{'}) \; mod \; (2m + 1) \tag{2.1.2}$$

where $g_i^{'}$ represents the value of the ith pixel in the modified group.

In the EMD scheme, a suitable value of m must be chosen to achieve a good trade-off between the embedding rate and image quality [37]. The highest embedding rate of 1.16 *bpp* is achieved with the EMD scheme when $m = 2$, while maintaining an acceptable quality of image with a PSNR value of about 52.1 dB and SSIM of 0.9971.

The EMD scheme is a fundamental scheme that has been widely used in image steganography. Researchers in this area have explored this scheme further to achieve higher embedding rates for the secret data and improve the image quality of the resulting stego image [38, 39]. In [38], an improved EMD steganographic scheme is proposed, in

which the secret digit E is concealed in each of the pixels of the cover image instead of only one of the pixels in the group. In this scheme, the weighted modulo function is modified as follows:

$$F_{Improved\_EMD} = (g_i + x) \ mod \ (2m + 1) \tag{2.1.3}$$

where g_i is the original value of the ith pixel, and x is the modification that is to be made to the value of this pixel. The value of x that makes $F_{EMD} = E$ is chosen as the amount of change in the value of g_i, where $-m \leq x \leq m$. Then, the new value of the ith pixel becomes $g_i^{'} = g_i + x$. The secret digit E is extracted at the receiver as $(g_i^{'}) \ mod \ (2m + 1)$.

This improved steganographic scheme also provides the highest embedding rate of 2.32 *bpp* when $m = 2$, which is twice the highest rate delivered by the original EMD scheme. However, the quality of the resulting stego image is lowered to a PSNR value of 47.97 *dB* and an SSIM value of 0.9904.

2.2 Diamond Encoding (DE) Scheme for Image Steganography

A data hiding scheme with the diamond encoding (DE) method has been proposed in [39]. In this DE steganographic scheme, the secret digit E, represented by the $(2k^2 + 2k+1)$-ary notational system, is concealed in the values p and q of two consecutive pixels of the cover image when the pixels of the cover image are arranged in a pre-specified order. In contrast to EMD, in this scheme, the value of only one or both of the pixels can be modified by having their values increased or decreased by a number in the range $[-k, k]$, or remain unchanged.

In this steganographic scheme, for a given k, a set S_k is defined as the set of vectors, each representing the new values of the two pixels being modified. This set can be arranged in a diamond shape, with all the vectors appearing around the original vector (p, q). Figure 2.3 is an illustration of the diamond shape patterns for $k = 1$ and $k = 2$.

(a) S_1 (b) S_2

Figure 2.3: Diamond encoding patterns.

The embedding process in the DE scheme is started by calculating the value of a modulo function defined by

$$F_{DE} = ((2k + 1)p + q) \ mod \ (2k^2 + 2k + 1) \tag{2.2.1}$$

It should be noted that F_{DE} assumes the value in the range $[0, 2k^2 + 2k]$. Next, a distance d_k is defined as

$$d_k = (E - F_{DE}) \ mod \ (2k^2 + 2k + 1) \tag{2.2.2}$$

Therefore, d_k will have values that are also in the range $[0, 2k^2 + 2k]$. This set of values, denoted by D_k, can also be arranged in a diamond shape pattern similar to that of S_k. Each value of d_k in this pattern corresponds to a vector in S_k. Figure 2.4 illustrates the diamond shapes for the distance patterns D_1 and D_2.

(a) D_1 (b) D_2

Figure 2.4: Diamond encoding distance patterns.

The secret digit E is extracted from the modified values of (p, q), denoted by $(p^{'}, q^{'})$ in the stego image by using the modulo function given by

$$E = F_{EXT\_DE} = ((2k+1) \cdot p^{'} + q^{'}) \ mod \ (2 \cdot k^2 + 2 \cdot k + 1) \tag{2.2.3}$$

In DE scheme, a suitable k must be chosen for a good trade-off between the embedding rate and image quality [39]. Higher the value of k, higher is the achieved embedding rate and lower is the resulting image quality [39]. In the DE scheme, the best quality of the stego image is achieved when $k = 1$. For example, when the cover image is Lenna, the PSNR value of the resulting stego image is 52.1 dB with an embedding rate of 1.16 bpp. On the hand, for $k = 2$, the corresponding values are 47.8 dB and 1.85 bpp for the same cover image. It is seen that the minimum change in the value of k in the DE scheme can only be unity. However, with this minimum change, the image quality and the embedding rate change drastically. Hence, by using this scheme, it is not possible to achieve a balanced trade-off between the quality of stego image and embedding rate that one would like to have. In the DE scheme, overflowing and underflowing issues may occur to pixels which require adjusting the values of the pixels to keep them within the range of 0 to 255. However, the mechanism used in the DE scheme to keep the value of an underflowing or overflowing pixel within this desired range increases or decreases its value of the stego pixel in question more than what it is necessary, which affects the quality of the stego image [39].

2.3 Chaotic-based Permutation of Cover Image Pixels for Enhancing the Robustness of a Steganographic Scheme

Permutation is a widely used mechanism in steganography that enhances the robustness of hidden secret data against statistical attacks. By rearranging the locations of pixels, the hidden secret data becomes more difficult to detect since the correlations in the original cover image are disrupted. This technique is a key component in many steganographic schemes aimed at providing secure and reliable information hiding. In recent years, some researchers have adopted chaos-based permutation mechanisms for steganography and encryption purposes [65], [66]. This is because chaotic mechanisms offer better randomness characteristics, which are essential in any permutation mechanism aimed at increasing security.

In a dynamic system, the behavior can be complex and chaotic. The study of chaotic phenomena in dynamic systems in a deterministic way was pioneered by Robert May

and Mitchell Feigenbaum in computer experiments [70]. The state of a dynamic system evolves with time, and its behavior is shown to be sensitive to initial conditions. With an initial condition, a chaotic sequence can be obtained, which is known as a logistic map. Mathematically, the logistic map is a very simple model that represents chaos [63], and it can be expressed as follows:

$$x_{n+1} = \mu x_n(1 - x_n) \tag{2.3.1}$$

where x_0 is an initial condition in $[0, 1]$, and x_n represents the number of elements in the chaotic sequence, and μ is the rate of evolving for this sequence.

The sequence generated by the logistic map serves as the permutation key for the steganographic scheme. Each pixel in the cover image is assigned an order based on the permutation key, and the key is sorted to create a cipher sequence. The pixels are then rearranged based on their assigned order. This process of permuting the location of pixels in the cover image based on the permutation key is illustrated in Figure 2.5 . After the permutation process, a permuted cover image is generated, and it is ready to be used in the embedding process along with the secret data.

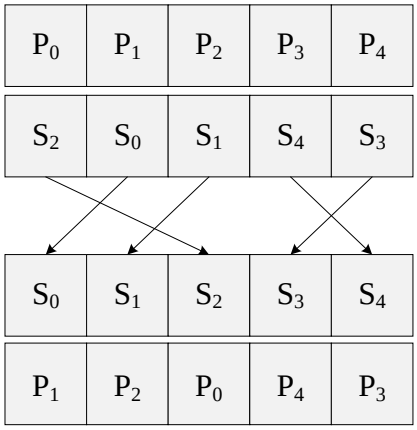


Figure 2.5: Permuting cover pixels using a cipher sequence.

After concealing the secret data, the permuted image is reordered to generate a stego image. This is done by using a decipher sequence obtained from the same permutation key used in the permutation process. The process of reordering the permuted stego pixels to their initial locations in the cover image using the decipher sequence is illustrated in Figure 2.6. The resulting image is the stego image, which contains hidden secret data and appears visually similar to the original cover image.

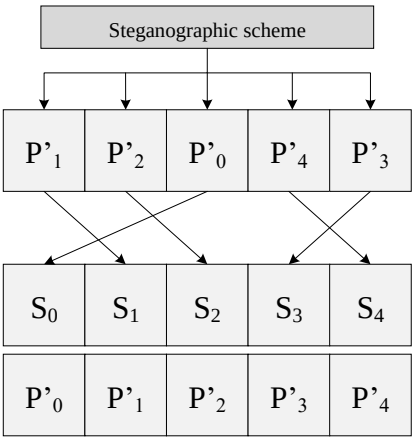


Figure 2.6: Reordering embedded pixels using a decipher sequence.

By leveraging the chaos-based permutation mechanism, the proposed steganographic scheme further enhances its security and robustness against steganalysis. The improved randomness characteristics of the permutation process make it more challenging for adversaries to detect the presence of hidden data within the stego image, thus increasing the overall resilience of the scheme. Integrating the permutation mechanism makes the image steganographic scheme to become more robust. Robustness against attacks has two forms: resilience to noise and resilience against steganalysis.

1. Resilience to noise can be checked by applying noise to an image. Salt and pepper noise may arise due to hardware failure or software crashes. In the transmitted image, some random pixels may change completely to black or white intensity values. Therefore, it is imperative to test the resilience of the steganographic scheme against salt and pepper noise. The robustness of the proposed scheme can be demonstrated by measuring its ability to recover the secret data despite the presence of this type of noise.

2. Resilience against steganalysis can be evaluated by applying different steganalysis techniques, such as pixel value difference (PVD) histogram analysis and machine learning-based anti-detection tests, on the steganographic scheme.

   (a) In the PVD histogram, the differences between corresponding pixel values in the cover image and its stego image are calculated. The resulting PVD values are then collected, and the frequency of their occurrence is counted across the entire image to form the PVD histogram. The underlying idea behind PVD histogram analysis is that any significant differences between the cover image and its stego image will be reflected in the histogram of the pixel differences, thus allowing the detection of the presence of hidden secret data.

   (b) In the machine learning-based anti-detection tests, an image is classified as a clean (cover) or suspicious (stego) image using a binary classifier. A misclassification of stego images in this test indicates that the hidden secret data is less detectable, and consequently, the steganographic scheme is more robust.

2.4 Private and Public-key Cryptosystems

Cryptography is the other side of information security, which deals with data encryption. It means the art of hiding and writing, where two processes are performed, encryption

and decryption processes [1]. In the encryption process, data is changed (scrambled) from its original form (plaintext) into a meaningless form (ciphertext). The decryption process is performed to recover the plaintext from the received ciphertext.

In cryptography, the encryption cryptosystems aim to deliver confidentiality, authentication, integrity, and nonrepudiation [1]. In confidentiality, unauthorized users are prohibited to access information. Authentication guarantees that the information is originated from the claimed sender. Integrity proves that data is unchanged and complete or that data is altered only by authorized users. Nonrepudiation indicates that the sender can not disown having sent the data or receiver cannot disown having received the data.

Encryption cryptosystems are generally categorized as symmetric or asymmetric cryptosystems based on the type of encryption keys. The symmetric cryptosystems are called by private-key encryption, and the asymmetric cryptosystems are called by public-key encryption. The symmetric cryptosystems requires only one key that is shared between the sender and the receiver. The asymmetric cryptosystems requires two keys, public and private. The public key is processed by the encryption process of the cryptosystem to encrypt the data, while the private key is used by the decryption process of the cryptosystem to decrypt data.

Advanced encryption standard (AES), data encryption standard (DES), and Rivest cipher 4 (RC4) are examples of the symmetric encryption cryptosystems. Rivest–Shamir–Adleman (RSA), elliptic curve (ECC), and Paillier are asymmetric cryptosystems. In fact, the asymmetric cryptosystems require more intensive processing compared to the symmetric cryptosystems. Researchers argue which encryption cryptosystem is more secure, yet the majority tends to adopt the symmetric cryptosystems that are characterized by speed and simplicity [71].

2.4.1 Binary Finite Fields $GF(2^n)$

Encryption cryptosystems perform arithmetic operations over prime (p) or binary (2^n) finite fields or so-called galois field (GF). The binary finite field $GF(2^n)$ is more suitable and efficient in hardware implementations because the arithmetic in the polynomial fields is a carry-less. In the prime finite fields $GF(p)$, finding points on an elliptic curve requires performing a square root algorithm [**?**, 72], while the process is much easier over $GF(2^n)$. In $GF(2^n)$, points can be found using generators for polynomials. There are $(2n-1)$ elements in $GF(2^n)$, which can be represented as binary polynomials. For example, the element in $GF(2^n)$ has a polynomial representation is expressed as follows:

$$a_{n-1} \cdot x^{n-1} + a_{n-2} \cdot x^{n-2} + a_{n-3} \cdot x^{n-3} + \cdots + a_1 \cdot x^1 + a_0 \cdot x^0 \tag{2.4.1}$$

where x^i represents the location of the ith term, and $a_i \in [0, 1]$ is the coefficient of the ith term.

Arithmetic operations are applied over these elements such as addition, multiplication, square, inversion, and division. Adding two elements $C = A + B$ is calculated by applying logic XOR function. Squaring an element A^2 is computed by padding 0s between two adjacent bits of the element. Multiplying two elements $C = A \cdot B$ is much harder and slower operation than addition and square.

Result of squaring an element or multiplying two elements can be out of the field. To reduce it, an irreducible polynomial of degree n, $(f(x))$, is used by applying the reduction (modular) step, such as $C = (A^2)\ mod\ f(x)$ or $C = (A \cdot B)\ mod\ f(x)$. Inversion operation is the most complex operation in terms of time and resources. Obtaining the inverse of an element A is the process of finding another element B that satisfies

$A \cdot B \equiv 1 \ mod \ f(x)$. Note that, the $f(x)$ has a major role in the performance of these operations. In this thesis, all curves and irreducible polynomials are chosen based on the recommendation of the NIST [73]. Next subsections introduce AES, EC, and Paillier encryption cryptosystems, and show how the finite fields are used to perform the modular arithmetic operations in these encryption cryptosystems.

2.4.2 Advanced Encryption Standard (AES) Cryptosystem

AES cryptosystem is a symmetric block cipher that encrypts, and decrypts 128 bits of a plaintext data using different cipher keys. AES cryptosystem supports three lengths of the cipher key, and they are 128, 192, and 256 bits. For each cipher key, number of encryption rounds are applied to generate a ciphertext. For the 128-bit, 192-bit and 256-bit cipher key, there are 10, 12, and 14 encryption rounds, respectively. AES cryptosystem contains two main parts, encryption/decryption processes and key expansion [67]. In the encryption/decryption processes, a 4×4 byte array, also called by a state (S), is the basic unit that transforms between the encryption/decryption rounds of AES cryptosystem. Figure 2.7 shows the encryption and decryption processes in AES cryptosystem.

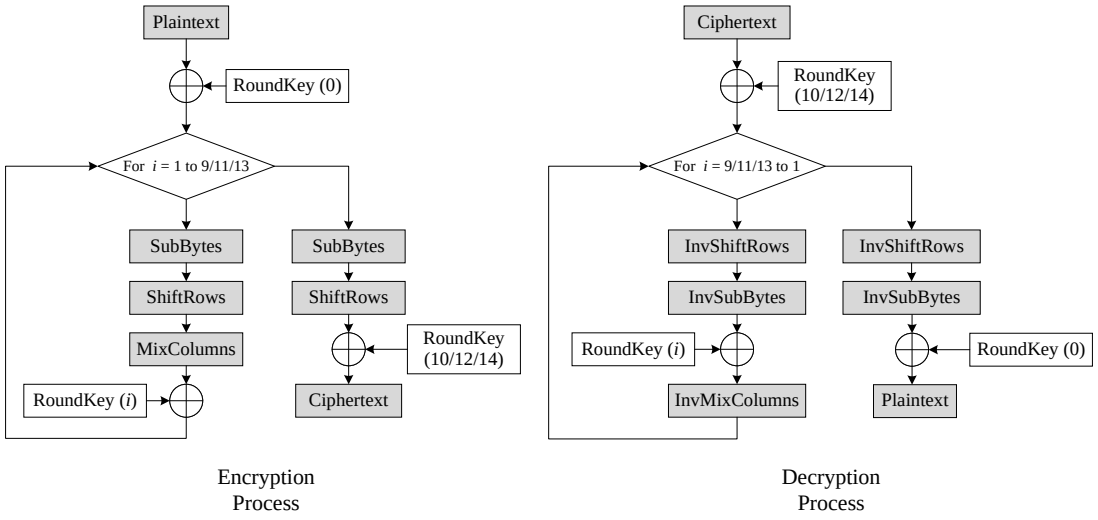


Figure 2.7: Encryption and decryption processes in AES cryptosystem.

In the encryption process, the plaintext is initially transformed into a state form. This state keeps changing until reaching the final round. Ciphertext is generated after performing the final round. The sender sends the ciphertext to the receiver of the communication channel. The receiver applies the decryption process, in which the rounds are in reverse order, to recover the plaintext. As shown in Figure 2.7 each encryption round has four types of byte-oriented transformations, SubBytes, ShiftRows, MixColumns, and AddRoundKey. In the decryption process, a decryption round consists of the inverse transformations, InvSubBytes, InvShiftRows, InvMixColumns, and InvAddRoundKey.

The encryption process begins with adding the initial cipher key to the plaintext, and the decryption process begins with adding the last subkey to the ciphertext. The encryption process starts with the SubBytes, ShiftRows, then MixColumns transformations and finally the subkey is added using XOR logic gate in the AddRoundKey transformation. In the last round, MixColumns is excluded, and the last subkey is added to generate the ciphertext. In the decryption process, the inverse transformations InvSubBytes, InvShiftRows, InvMixColumns, and InvAddRoundKey,

are performed. InvMixColumns transformation is excluded in the last round and the initial cipher key is added to recover the plaintext.

A) **SubBytes and InvSubBytes:**

The SubBytes transformation is a non-linear substitution performed over each byte in the state. The S-box is a table of 16×16 size which contains all possible 256 of 8-bit values. Each byte in the state has a range of values $\in (00, \ldots, ff)_8$. Constructing the S-box is done by (i) finding first the multiplicative inverse of the byte over the $GF(2^8)$ using the $(x^8 + x^4 + x^3 + x + 1)$ irreducible polynomial modulus, and (ii) applying the affine transformation over $GF(2^8)$ [67]. In the decryption process, the bytes are mapped back to their original form using the inverse substitution box (Inv-S-box). Inv-S-box is constructed by (i) applying the affine transformation over $GF(2^8)$, and (ii) finding the multiplicative inverse for a byte over $GF(2^8)$ mod $(x^8 + x^4 + x^3 + x + 1)$.

B) **ShiftRows and InvShiftRows:**

In ShiftRows transformation, the order of last three rows in the state is cyclically shifted to the left. The first row is left without any shifts while the second row shifts left one location, the third row two locations, and the fourth row three locations. The ShiftRows transformation is performed after SubBytes transformation. However, it can be performed before the SubBytes transformation without affecting the function of the encryption round due to the strong alignment in the AES algorithm [67]. In the decryption process, InvShiftRows transformation is applied by shifting order of the last three rows to right. The second row shifts right one location, the third row two locations, and the fourth row three locations.

C) **MixColumns and InvMixColumns:**

MixColumns transformation is applied after ShiftRows transformation for first 9 rounds during the encryption process, and it is not applied in the last round. Mixing columns of a state is done by performing $GF(2^8)$ multiplication. Each column (four bytes) of the state is multiplied by a constant matrix, which consists of four polynomials over $GF(2^4)$ mod $(x^4 + 1)$. The MixColumns constant matrix is defined as $(3 \cdot x^3 + 1 \cdot x^2 + 1 \cdot x + 2)$ mod $(x^4 + 1)$. The transformation in InvMixColumns is performed after AddRoundKey during the decryption process for first 9 rounds, and it is excluded from the last round. In the InvMixColumns transformation, an inverse matrix for the matrix in MixColumns is constructed over $GF(2^4) modulo (x^4 + 1)$. The InvMixColumns constant matrix is defined as $(b \cdot x^3 + d \cdot x^2 + 9 \cdot x + e)$ mod $(x^4 + 1)$.

- **Key Expansion:** In the key expansion, a process of generating subkeys (i.e., a key for each round) is performed using an initial cipher key. For the 128-bit cipher key, 10 of subkeys are generated with 16 bytes size for each. Figure 2.8 illustrates the key expansion process for generating the keys for the 10 encryption/decryption rounds using the 128-bit cipher key. The round constant (Rcon) is a constant array used in the key expansion process [67]. Table 2.1 lists the entries of the Rcon constant array for the 10 rounds.

2.4.3 Elliptic Curve (EC) Cryptosystem

Elliptic curve (EC) cryptosystem is a public-key cryptography, which was first proposed by Neal Koblitz and Victor Miller in the 1980s [68], [74]. Since then, many studies have

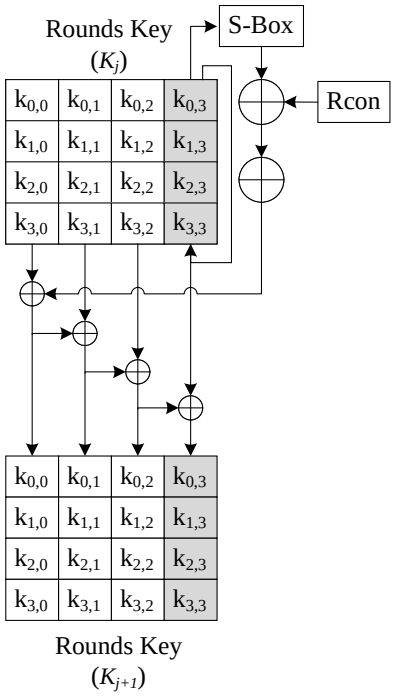


Figure 2.8: Key expansion process in AES cryptosystem.

Table 2.1: Rcon Entries for 10 AES Rounds.

Round Number	**Rcon Entry**	**Round Number**	**Rcon Entry**
1	01 00 00 00	**6**	20 00 00 00
2	02 00 00 00	**7**	40 00 00 00
3	04 00 00 00	**8**	80 00 00 00
4	08 00 00 00	**9**	1B 00 00 00
5	10 00 00 00	**10**	36 00 00 00

been conducted to explore its security levels against other public-key cryptosystems such as El-Gamal, RSA and digital signature algorithm (DSA) [75, 76], which are based on either the integer factorization or discrete logarithm problems [77].

Equivalent security levels with smaller sizes of keys, ease to implement, and resource savings, are reasons that give the EC cryptosystem to be very appealing and more dominant comparing with other public-key cryptosystems. EC cryptosystem has been standardized by IEEE and the NIST as a scheme in digital signature and key agreement protocols [78].

Elliptic Curves (ECs) are formulated by the so called Weiestrass equations, which can be performed over by normal or polynomial basis. In this thesis, polynomial basis in $GF(2^n)$ is chosen for its efficiency on the hardware platforms [72]. Equation 2.4.2 represents the general form of the nonesingular curve $E_{curve}(a, b)$ over $GF(2^n)$ [79].

$$y^2 + x \cdot y = x^3 + a \cdot x^2 + b \tag{2.4.2}$$

where $a, b \in GF(2^n)$ and $b \neq 0$. A set of affine points (x, y) satisfying the curve forms a group [79] with an identity point of that group. There are two fundamental elliptic curve operations, doubling and adding points. Point doubling is denoted as $P_1 = 2 \cdot P_0$, where P_1 is (x_1, y_1) and P_0 is (x_0, y_0) while point addition is denoted as $P_2 = P_0 + P_1$, where P_2 is (x_2, y_2), and $P_1 \neq P_0$. All points in the selected curve $E_{curve}(a, b)$ are

represented in the affine coordinates. However, there is another coordinate system that can be used to present points in the chosen curve such as protective coordinate system.

Adding two points $P_2 = P_0 + P_1$ on a curve $E_{curve}(a, b)$ is calculated as the following:

$$\lambda = \frac{y_1 + y_0}{x_1 + x_0} \tag{2.4.3}$$

$$x_2 = \lambda^2 + \lambda + x_0 + x_1 + a \ , \ y_2 = \lambda \cdot (x_0 + x_2) + x_2 + y_0 \tag{2.4.4}$$

While doubling a point $P_1 = 2 \cdot P_0$ on a curve $E_{curve}(a, b)$ is calculated as the following:

$$\lambda = x_0 + \frac{y_0}{x_0} \tag{2.4.5}$$

$$x_1 = \lambda^2 + \lambda + a \ , \ y_1 = x_0^2 + (\lambda + 1) \cdot x_1 \tag{2.4.6}$$

Scalar point multiplication (SPM) is the main important operation that dominants the EC based cryptosystems or protocols such as elliptic curve diffie-hellman (ECDH) [80] for key agreements and elliptic curve digital signature (ECDS) for digital signatures. SPM is process of adding a point k times, where k is a positive integer and P is a point on a selected curve. SPM is based on the implementation of the underlying point operations, where a series of doubling and adding points are performed to get $k \cdot P$ result. Finite fields operations are involved in the EC point operations such as addition, square, multiplication and inversion. Figure 2.9 presents the hierarchy of elliptic curve cryptosystem.

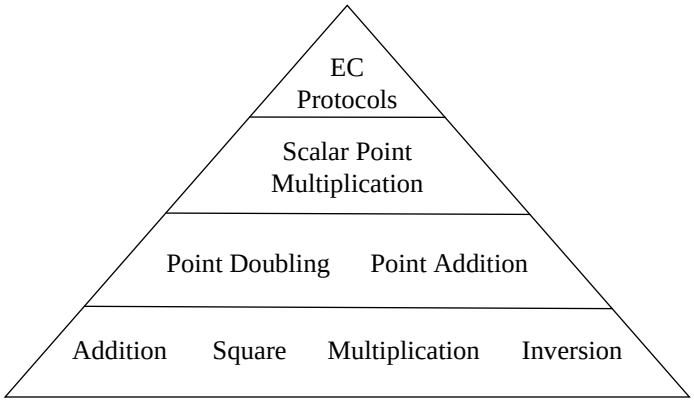


Figure 2.9: Hierarchy of elliptic curve cryptosystem.

Example 2.4.1. *Elliptic Curve Point Operations*

Assume construct a curve over $GF(2^3)$, and coefficients of the curve are $a = 8$, $b = 1$, and the irreducible polynomial is $f(x) = x^3 + x + 1$, The selected curve is the following:

$$E_{curve} = y^2 + x \cdot y = x^3 + 8 \cdot x^2 + 1 \tag{2.4.7}$$

To add a point $P_0 = (x, x^2 + x + 1)$, and $P_1 = (x^2 + 1, x^2 + x)$, the following is applied to get $P_2 = (x_2, y_2)$:

1. $\lambda = \dfrac{x^2 + x + 1 + x^2 + x}{x + x^2 + 1} = x^2.$
2. $x_2 = x^{2^2} + x^2 + x + x^2 + 1 + 8 = x^2 + x.$

3. $y_2 = x \cdot (x + x^2 + x) + x^2 + x + x^2 + x + 1 = x^2 + x + 1$.

4. $P_2 = (x^2 + x, x^2 + x + 1)$.

And to double a point $P_0 = (x+1, x^2+1)$, the following is applied to get $P_1 = (x_1, y_1)$:

1. $\lambda = x + 1 + \dfrac{x^2 + 1}{x + 1} = 0$.

2. $x_1 = 0^2 + 0 + 8 = x + 1$.

3. $y_1 = x + 1^2 + (0 + 1) \cdot (x + 1)$.

4. $P_1 = (x + 1, x^2 + x)$.

2.4.4 Paillier Cryptosystem

Paillier cryptosystem is a probabilistic public-key cryptography, which was invented by Pascal Paillier in 1999 [69]. It is based on the RSA computational-complexity public-key cryptosystem. Paillier cryptosystem has the $n-$th residue problem, where finding the composite $n-$th residue is believed to be computationally hard. The feature homomorphic property [69] in paillier cryptosystem makes it very appealing to be used and integrated in privacy-preserving applications such as transferring money and electronic voting campaigns.

The textbook version of RSA cryptosystem is a deterministic public-key cryptography (i.e., no random components) [76], which makes it vulnerable to the chosen-plaintext attacks by exploiting the multiplicative property [81,82]. In this attack, an adversary can distinguish between the ciphertexts, by challenging the sender, who has the private key, to encrypt one of two chosen plaintexts A and B to obtain the ciphertext C and send it back to the adversary. Using the public key, the adversary encrypts A and B to obtain $C_A = E_{k_{public}}(A)$ and $C_B = E_{k_{public}}(B)$. The adversary can check and distinguish which of C_A and C_B is equal to the received C. Since the RSA is is deterministic, one of them has to be equal to the C. This kind of RSA implementation is referred to as non-semantic cryptosystem. To resist this type of attacks, padding schemes for RSA is applied by concealing some random paddings into the plaintexts before starting encryption process [83]. This makes RSA is a semantic secure algorithm, which indicates that attackers can not distinguish between the ciphertexts.

Padded RSA cryptosystem does not support the homomorphic property since the randomness injected to the plaintext before encryption [84], hence RSA can be either semantically secure or homomorphic cryptosystem. On the other hand, Paillier cryptosystem is a semantic and homomorphic cryptosystem. However, the price to pay is that Paillier cryptosystem consumes more resources with equivalent security level with RSA cryptosystem [84]. More resources are required since Paillier cryptosystem involves more modular exponentiation and multiplications field operations than the RSA cryptosystem.

The following key generation, encryption, and decryption processes make up the Paillier cryptosystem:

2.4.4.1 Key Generation in Paillier Cryptosystem

The key generation process is performed to generate pair of keys, public and private, for the encryption and decryption processes [1]. In the Paillier cryptosystem, a user chooses two large primes, q and p, randomly and independently of each other, such that great common divisor (GCD) for q and p satisfy the following:

$$GCD = (q \cdot p, (q-1) \cdot (p-1)) = 1 \tag{2.4.8}$$

Then, compute the following:

$$n = q \cdot p \ , \ \lambda = LCM((q-1) \cdot (p-1)) \tag{2.4.9}$$

where LCM is the least common multiple.

Select random integer g in the $\mathbb{Z}^*_{n^2}$ field, and check the existence of the multiplicative inverse $\mu = (L(g^\lambda \ mod \ n^2))^{-1} \ mod \ n$, where $L(x)$ is defined as:

$$L(x) = \frac{(x-1)}{n}. \tag{2.4.10}$$

The user distributes his public key $(pubk)$ pair $(n, \ g)$ for encryption process, and the $(\lambda, \ \mu)$ is kept secret as the private key $(prvk)$ for decryption process.

2.4.4.2 Encryption and Decryption in Paillier Cryptosystem

Encryption: Let M be a plaintext to be encrypted where $M \in \mathbb{Z}_n$. Select a random integer R where $R \in \mathbb{Z}^*_n$. The ciphertext C is computed as follows:

$$C = E(M, pubk) = g^M \ \cdot \ R^n \ mod \ n^2 \tag{2.4.11}$$

Decryption: Let C be a ciphertext to be decrypted where $C \in \mathbb{Z}^*_{n^2}$. The plaintext M is computed as follows:

$$M = D(C, prvk) = L(C^\lambda \ mod \ n^2) \cdot \mu \ mod \ n \tag{2.4.12}$$

Example 2.4.2. *Paillier Cryptosystem*

A) **Key generation:**

(a) Assume $p = 7, q = 11$, then $n = p \cdot q = 77$.

(b) The integer $g \in \mathbb{Z}^*_{n^2}$ is selected. Let $g = 56552$.

(c) $(n, \ g) = (77, 56552)$ is the public key pair.

(d) Compute $\lambda = LCM((q-1) \cdot (p-1)) = LCM(6, 10) = 30$.

(e) Then $\mu = (L(g^\lambda \ mod \ n^2))^{-1} \ mod \ n = (L(56552^{30} \ mod \ 5929))^{-1} \ mod \ 77 = (L(3928))^{-1} \ mod \ 77 = (\frac{3928-7}{77})^{-1} \ mod \ 77 = 51^{-1} \ mod \ 77 = 74 \ mod \ 77$.

(f) Now the $(\lambda, \ \mu) = (30, 74)$ is the private key pair.

B) **Encryption Process:**

To encrypt a plaintext $M = 42$, where $42 \in \mathbb{Z}_{77}$. Choose random integer $R = 23$, where $23 \in \mathbb{Z}^*_{77}$. Compute $C = g^M \cdot R^n \ mod \ n^2 = 56552^{42} \cdot 23^{77} \ mod \ 5929 = 4624$.

C) **Decryption Process:**

To decrypt a ciphertext $C = 4624$. Compute $M = L(C^\lambda \ mod \ n^2) \cdot \mu \ mod \ n = L(4624^{30} \ mod \ 5929) \cdot 74 \ mod \ 77 = L(4852) \cdot 74 \ mod \ 77 = 42 = M$.

2.4.4.3 Homomorphic Properties in Paillier Cryptosystem

An interesting feature of the Paillier cryptosystem is its homomorphic properties. Assume there are two ciphertexts $C_1 = E(M_1, pubk) = g^{M_1} \cdot R_1^n \ mod \ n^2$ and $C_2 = E(M_2, pubk) = g^{M_2} \cdot R_2^n \ mod \ n^2$, where R_2 and R_2 are randomly chosen from $\mathbb{Z}_n^*$. Both ciphertexts are additively homomorphic on $\mathbb{Z}_n$ and this leads to the following properties:

- **Homomorphic Addition Property:** Decryption a product of two ciphertexts obtains the sum of their corresponding plaintexts as follows:

$$D(E(M_1, \ pubk) \cdot E(M_2, \ pubk) \ mod \ n^2) = M_1 + M_2 \ mod \ n \tag{2.4.13}$$

And that can be proved as the following:

$$\begin{aligned} E(M_1, pubk) \cdot E(M_2, pubk) &= (g^{M_1} \cdot R_1^n) \cdot (g^{M_2} \cdot R_2^n) \cdot mod \ n_2 && (2.4.14)\\ &= (g^{M_1} \cdot (g^{M_2}) \cdot R_1^n \cdot R_2^n) \cdot mod \ n_2 && (2.4.15)\\ &= g^{M_1+M_2} \cdot (R_1 \cdot R_2)^n \cdot mod \ n_2 && (2.4.16)\\ &= E(M_1 + M_2, \ pubk) && (2.4.17) \end{aligned}$$

- **Homomorphic Multiplication Property:** Raising a ciphertext to another plaintext obtains the product of the two plaintext as follows:

$$D(E(M_1, \ pubk)^{M_2} \ mod \ n^2) = M_1 \cdot M_2 \ mod \ n \tag{2.4.18}$$

And that can be proved as the following:

$$\begin{aligned} E(M_1, \ pubk)^{M_2} &= (g^{M_1} \cdot R_1^n)^{M_2} \ mod \ n_2 && (2.4.19)\\ &= g^{M_1 \cdot M_2} \cdot (R_1^{M_2})^n \ mod \ n_2 && (2.4.20)\\ &= E(M_1 \cdot M_2, \ pubk) && (2.4.21) \end{aligned}$$

Generally, a ciphertext is to power of a constant k obtains the product of the plaintext with the constant as follows:

$$D(E(M_1, \ pubk)^k \ mod \ n^2) = k \cdot M_1 \ mod \ n. \tag{2.4.22}$$

The power of Paillier cryptosystem comes from these properties, which makes it particularly a suitable cryptosystem in the design of electronic voting protocols, watermarking and secret sharing systems [85].

2.5 Xilinx Zynq-7000 APSoC Platform

In the early 2000's, a reconfigurable hardware-based platform, called by field programmable gate array (FPGA) has evolved to solve the issues that exist in the CPUs. FPGA demonstrates a powerful acceleration for developing and implementing algorithms by integrating high-speed communication interfaces for high transfer rates, fast digital signal processors (DSPs) for performing complex operations, supporting parallelism for

independent computations, and architectural pipelining for both resources reusability and high throughputs [86]. Although, FPGA introduces excellent hardware capabilities to deliver high performance, its main drawbacks are, the low-level of flexibility in the development phases, nonstandard drivers for the interfaces, and restricted scalability for large implementations. To resolve these challenges and to meet the requirements of high-performance applications, combining multiple chips into a single platform, which is called by system-on-chip (SoC), is a solution that will result in flexibility of the software and the high performance of the hardware. This type of heterogeneous platform has been adopted by Xilinx [87]. In 2011, Xilinx revealed the new platform as all-programmable SoC (APSoC), and they gave their new platform the name Zynq APSoC [87].

The new Zynq APSoC platform integrates the software programmability and high-performance of the ARM Cortex A9 hardcore processor with the hardware reconfigurability of the FPGA. Through this integration, Zynq APSoC supports software routines, operating system, and other services available in the ARM processor. Also, it provides an area for implementing logic, arithmetic operations, parallel processing, dynamic reconfigurations, and other capabilities delivered by FPGA. Therefore, functions and tasks in a designed system will be suitably divided between software and hardware.

The Zynq-7000 APSoC architecture consists of two main parts: a Processing System (PS) and Programming Logic (PL) [87]. The PS contains the dual-core ARM Cortex-A9 processor, while the PL is the side in which the FPGA device does locate. In the Zynq-7000 SoC, the architecture of the PS is fixed, and it contains processor and system memory, whereas the PL is fully flexible, offering a space, for the designer, to create or reuse peripherals. The Zynq-7000 APSoC also features integrated memory, a variety of peripherals, and high-speed communications interfaces. The communication links between PS and PL are made using the industry standard Advanced eXtensible Interface (AXI) connections [88]. In the PL, the reprogrammable part of the Zynq-7000 APSoC is predominantly featured as FPGA device [89]. It offers flexible and more customizable methods for performing and evaluating various reconfigurable hardware implementations. Figure 2.10 shows the general model of Zynq-7000 APSoC architecture.

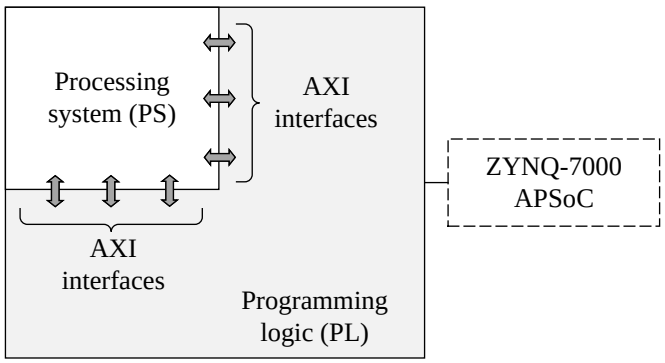


Figure 2.10: General model of the Zynq-7000 APSoC architecture.

The Zynq-7000 APSoC consists of two systems, hardware, and software. The hardware system has a processor (i.e., central component of the hardware system), memories, peripherals, and buses connecting all these components. In the software system, a stack of applications running one specific operating system (OS), and a software functionality for interfacing with the components in the hardware system. Figure 2.11 depicts the hardware and software systems in the Zynq-7000 APSoC. Peripherals are functional components that can be in form of (i) Intellectual Property

(IP) cores (coprocessors) for running specific tasks required by the central processor. (ii) cores for interacting with input/output interfaces. (iii) additional memories.

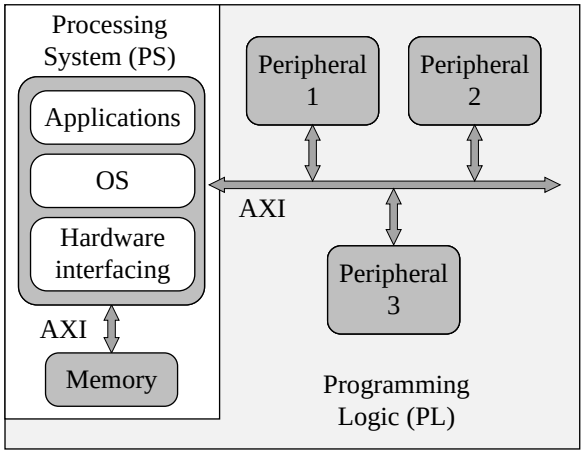


Figure 2.11: Hardware and software systems in the Zynq-7000 APSoC.

For a complete integrated embedded system, the key factor is not just in the properties of the PS and PL parts, but also the approach of bridging these two parts using an efficient AXI interconnections and interfaces. The AXI standard is a SoC communication bus protocol, which was developed by ARM in 1996 [88, 90]. Xilinx has extended and upgraded the AXI standard, and the current version is called by AXI4. AXI4 standard is supported by the Xilinx Zynq-7000 APSoC for providing an optimal interconnecting service between PS and PL parts. The AXI4 buses in the Zynq-7000 APSoC have three flexible implementations, and each represents a protocol for a particular connection between the processor and different blocks in the embedded system. The AXI4 buses are (i) AXI4 for many data burst transfer per address, and high-performance memory-mapped links, (ii) AXI4-Lite for single data burst transfer per address, simplified memory-mapped links, (iii) AXI4-Stream for unrestricted data burst transfer per address, high-speed streaming and none memory-mapped links.

The AXI4 interface is defined as point-to-point connection for sending/receiving data, addresses, and handshaking signals. The connections between the PS and PL are via a set of these interfaces, and these interfaces are managed by interconnect switches, which are located in the PS. There are two types of roles for the AXI4 interfaces, master (M) and slave (S). The master AXI4 interface indicates the either PS or PL is initiating and controlling the transactions, while the slave AXI4 interface indicates either PS or PL is responding to the master of that interface. In the Zynq-7000 APSoC, the PS-PL AXI4 interface is called by a general-purpose (GP) AXI4, which fits the low to medium rate of communications between PS and PL. The GP AXI4 interface is a direct and does not support buffering. There are four GP AXI4 interfaces, two master interfaces in the PS, and two slave interfaces in the PL. High-performance (HP) is another PS-PL AXI4 interface that is required in the high-rate communications between PL and the memories in the PS. There are four HP AXI4 interfaces, where all of them are slave in the PS and master in PL Each HP AXI4 interface has a buffer (i.e., FIFO). The last PS-PL AXI4 interface is the Accelerator Coherency Port (ACP), and it is a single interfacing connection between the PL and the Snoop Control Unit (SCU) within the Application Processing Unit (APU) in the PS. The ACP interface is responsible for realizing a coherency between cashes of the APU and the memories blocks of the master cores in the PL. Figure 2.12 shows the different types of AXI4 interfaces in the Zynq-7000 APSoC architecture.

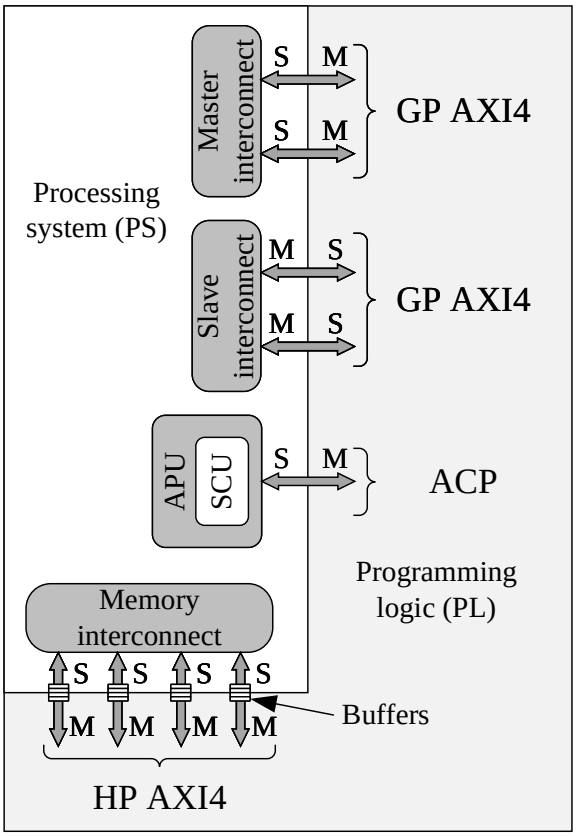


Figure 2.12: AXI4 interfaces in the Zynq-7000 APSoC architecture.

2.5.1 Hardware-based Platforms

In real-time applications, image steganographic schemes are designed and implemented using both software and hardware-based platforms. Software-based platforms offer convenience and flexibility for designing steganographic schemes. However, they do not support parallelism, and their computational time is longer than their hardware counterparts [91]. On the other hand, hardware-based platforms guarantee high computational speed, architectural optimization techniques, and massive parallelism support [92]. The performance of a hardware implementation for a steganographic scheme is evaluated by measuring the operating speed (MHz), utilized resources, throughput ($Mbps$), and power consumption ($Watt$). A hardware implementation is considered efficient if the operating speed is high, utilized resources are low, throughput is high, and power consumption is low. This can be achieved by applying appropriate architectural optimization techniques and utilizing resources effectively.

2.5.2 Field Programmable Gate Array (FPGA)

Field programmable gate array (FPGA) is an integrated circuit (IC) made to be configured by a user or a designer after manufacturing. FPGA is one of the preferable reconfigurable hardware platforms [93]. It offers flexible and customizable methods for performing and evaluating different hardware implementations and designs. Current FPGA devices consist of high-level elements such as reconfigurable logic blocks (CLBs), dedicated block memories (BRAMs), and digital signal processing (DSP) units. Reconfiguring the FPGA device involves constructing arithmetic and logic operations of an algorithm on the CLBs and other elements.

Elements of the FPGA device are linked via reconfigurable interconnects (i.e.,

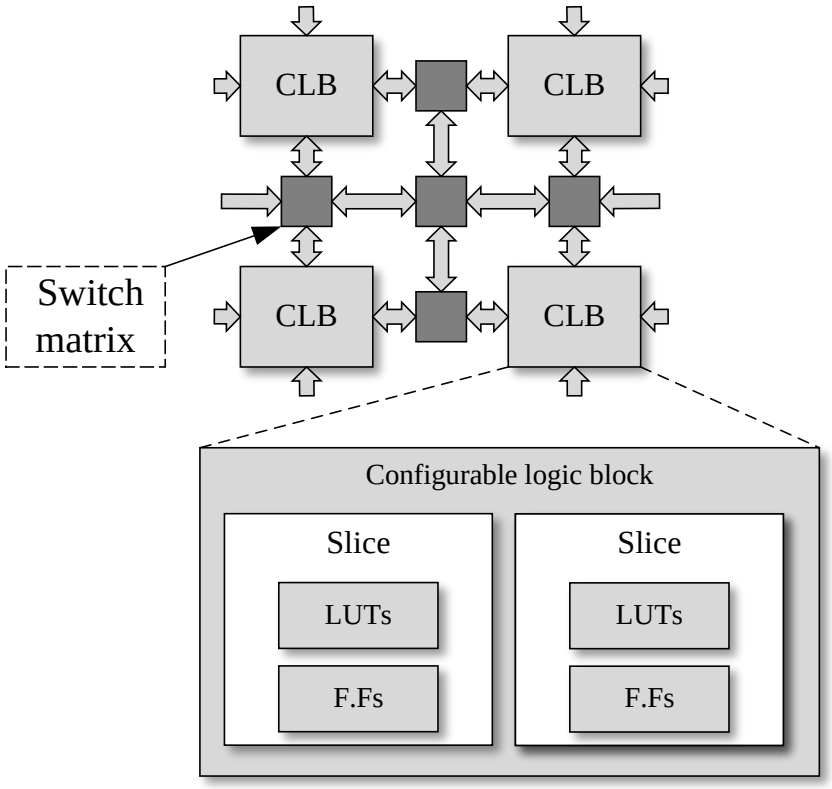


Figure 2.13: CLB structure in the FPGA device of the Xilinx Zynq-7000 APSoC.

routers) and switch matrices. Each CLB contains slices and carry logic components. The slices are used to perform two types of logic, sequential and combination. The combinational logic implements boolean functions. Flip flops performs the sequential logic like storing data (e.g., registers and latches) and passing values across CLBs. The carry logic is an arithmetic unit which is required to pass intermediate values through neighboring slices. The switch matrix locates next to each CLB, and it is considered as router for connecting elements of (i) a single CLB (ii) from CLB to other resources in the PL. In the PL part of the Xilinx Zynq-7000 APSoC, each CLB contains 2 slices, and a slice includes 4 lookup tables (LUTs) and 8 registers (i.e., flip flops (F.Fs)) [89]. Figure 2.13 represents a single CLB structure in the FPGA device of the PL part in the Xilinx Zynq-7000 APSoC.

2.5.3 Development Phases of an IP Core

Developing an IP core in the Xilinx Vivado design methodology involves six phases that ensure the IP core meets the target requirements:

1. Specification Phase: In this phase, the requirements and specifications of the IP core are determined. The specifications may include performance, power, and area requirements, as well as the interface specifications. The IP core requirements may come from the end-customer, market needs, or the IP core developer's own needs. The specifications must be well-defined, unambiguous, and complete to ensure that the IP core meets the target requirements.

2. Architecture Phase: In this phase, the architecture of the IP core is determined. The architecture may include the number of modules, the number of ports, the bit width of the data bus, and the type of interface. The architecture must be

optimized for performance, power, and area, and must meet the specifications determined in the previous phase. The architecture phase may also include algorithm selection, data flow, and pipeline optimization.

3. Design Phase: In this phase, the IP core is designed using high-level hardware description languages such as Verilog or VHDL. The design phase may include the creation of block diagrams, RTL coding, and simulation. The design phase is critical for ensuring the IP core meets the target specifications, and is also an opportunity to optimize the design for performance, power, and area.

4. Verification Phase: In this phase, the IP core is verified to ensure that it meets the target specifications. The verification phase may include simulation, emulation, and FPGA prototyping. The IP core must be tested for functionality, timing, and power. The verification phase is critical for ensuring the quality and reliability of the IP core.

5. Integration Phase: In this phase, the IP core is integrated into the target system or application. The integration phase may include hardware/software integration, IP core integration, and system-level integration. The IP core must be tested in the target environment to ensure that it meets the requirements and is compatible with other components in the system. The integration phase is critical for ensuring that the IP core works as expected in the target environment.

6. Deployment Phase: In this phase, the IP core is deployed to the target market. The deployment phase may include documentation, support, and training. The IP core must be well-documented to facilitate integration and use by customers. The support and training are critical for ensuring the successful adoption of the IP core by customers.

In summary, developing an IP core in the Xilinx Vivado design methodology involves several phases, including specification, architecture, design, verification, integration, and deployment. Each of these phases is critical for ensuring that the IP core meets the target specifications and works as expected in the target environment. The Xilinx Vivado design methodology provides a structured approach to IP core development, ensuring the quality and reliability of the IP core.

2.6 Summary

In this chapter, an in-depth review is provided on the fundamental steganographic schemes of EMD and DE on which the proposed schemes in this thesis are based. A mechanism of permutation using logistic maps for enhancing the robustness of the steganographic scheme is introduced. To make the model for image steganography secure and lossless, an introduction to the private and public-key cryptosystems and their main operations are reviewed. This chapter concludes with the discussion of the AMD Xilinx Zynq-7000 APSoC platform and its primary components on which both the proposed schemes and the secure lossless model are implement.

Chapter 3

UMIS: A Unified Modulus-based Image Steganographic Scheme

3.1 Introduction

In this chapter, we present a modulus-based steganographic scheme [94] that combines the EMD and DE schemes to hide secret data blocks. The proposed scheme does not require dividing the secret data blocks. Instead, we adopt a novel scheme where secret data blocks are searched for in two lookup tables, which contains all possible combinations of the secret data block of size L. By using the approach, four secret digits, that represent the corresponding to the secret data block of size L, are obtained. The concealment process for these digits is carried out as follows. The first two secret digits are concealed using the EMD scheme, which employs a $(2m+1)$-ary notational system. The other two secret digits are concealed using the DE scheme, which utilizes a $(2k^2+2k+1)$-ary notational system. For a robust scheme, we incorporate the chaotic-based permutation mechanism explained in Section 2.3 before starting the process of concealing secret data. The chaotic-based permutation mechanism utilized in this scheme is crucial for ensuring a robust and secure concealing process. By shuffling the pixel positions in the cover image, we introduce a significant level of complexity and randomness, thereby enhancing the scheme's resistance against steganalysis attacks.

Extensive experiments are conducted to evaluate the performance of the proposed scheme and compare it with that of the EMD and DE modulus-based schemes. Moreover, three types of steganalysis attacks are applied to demonstrate the robustness of the proposed scheme. The three attacks are PVD histogram analysis, salt and pepper noise analysis, and a machine learning-based anti-detection test.

3.2 Proposed Steganographic Scheme

In this section, we develop the novel modulus-based steganographic scheme. In this scheme, we choose four groups Q_0, Q_1, Q_2 and Q_3 of pixels in the cover image to conceal four secret digits S_0, S_1, S_2 and S_3, respectively. These four secret digits together constitute a secret data block B. In the embedding process, we use the DE scheme to conceal the two secret digits S_0 and S_2 in the pixel groups Q_0 and Q_2, respectively, while we use the EMD scheme to conceal the S_1 and S_3 in the groups the pixel groups Q_1 and Q_3, respectively. In order to enhance the security level of the proposed scheme, the chaotic-based permutation mechanism, explained in Section 2.3, is performed prior to the process of concealing secret data. A block diagram of the proposed scheme is shown in Figure 3.1.

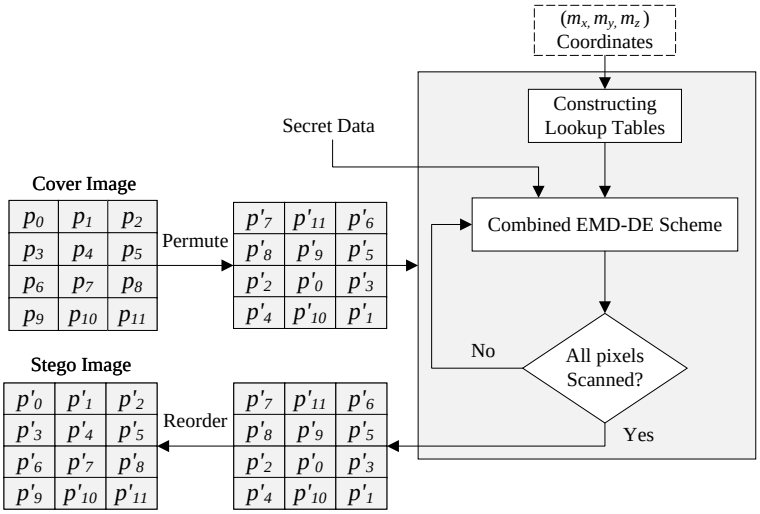


Figure 3.1: Block diagram of the efficient steganographic scheme.

3.2.1 Proposed Concealing Process

The inputs to the scheme consist of a cover image, parameters m_x, m_y, and m_z ($m_z < m_x < m_y$) that are used to construct two lookup tables T_1 and T_2, and a stream of secret data blocks, each denoted by B. The output of the scheme is a stego image, which is the modified version of the cover image with the secret data embedded in it. The first lookup table, T_1, is constructed using the parameters m_x and m_y. In this table, the x values range from 0 to $2m_x$, and the y values range from 0 to $2m_y$. Entries in this table are integers representing pointers to secret digits. The maximum value in this table is denoted by t_1^{MAX}, and therefore entries of table T_1 range from 0 to t_1^{MAX}. The value of t_1^{MAX} is calculated using

$$t_1^{MAX} = 2^{\lfloor \log_2((2m_x+1)(2m_y+1)) \rfloor} - 1 \tag{3.2.1}$$

Figure 3.2 shows the example of the lookup table T_1 when $m_x = 2$ and $m_y = 3$.

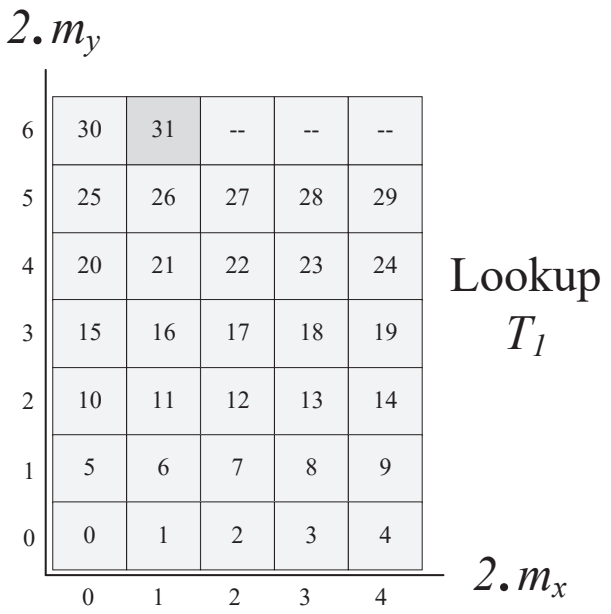


Figure 3.2: Lookup T_1 when $m_x = 2$ and $m_y = 3$.

Table T_2 is constructed based on the value of t_1^{MAX} in T_1. Values of x and y coordinates in T_2 are both in the range $[0, t_1^{MAX}]$. The value of the entries in T_2 represent all possible blocks of the secret data, and size of a single secret data block B is $2Log_2(t_1^{MAX}+1)$-bit. Figure 3.3 shows an example of the table T_2 when the value of t_1^{MAX} in T_1 is 31. For the example given in Figure 3.3, the size of the secret data block B is 10-bit. Give secret data block B, Table T_2 is searched to obtain its coordinates (t_{2_x}, t_{2_y}). Using these coordinates as the values of the entries of table T_1, we find the corresponding coordinates as $T_1(t_{2_x}) \rightarrow (t_{1_{x_0}}, t_{1_{y_0}})$ and $T_1(t_{2_y}) \rightarrow (t_{1_{x_1}}, t_{1_{y_1}})$. Values of $t_{1_{x_0}}$ and $t_{1_{y_0}}$ are considered as the secret digits $t_{1_{x_0}} \rightarrow S_0$ and $t_{1_{y_0}} \rightarrow S_1$, while values of $t_{1_{x_1}}$ and $t_{1_{y_1}}$ are the secret digits $t_{1_{x_1}} \rightarrow S_2$ and $t_{1_{y_1}} \rightarrow S_3$.

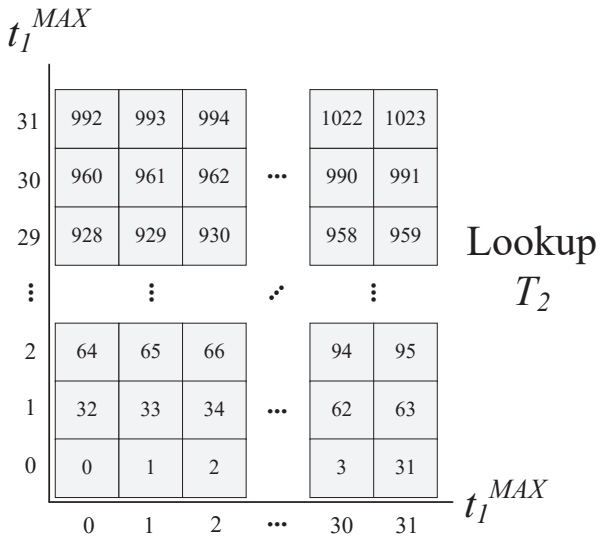


Figure 3.3: Lookup T_2 when $t_1^{MAX} = 31$.

These four secret digits are now concealed using the combined EMD and DE schemes as follows. The values of S_0 and S_2 are concealed by using the embedding function given by Eq. 2.2.1, whereas the values of S_1 and S_3 are concealed by using the embedding function given by Eq. 2.1.1. S_0 is concealed by the DE scheme using the $(2m_z^2 + 2m_z + 1) - ary$ notational system with the group Q_0 that consists of two pixels (p_0, q_0). S_1 is concealed by the EMD scheme using the $(2m_y + 1) - ary$ notational system with a group Q_1 that consists of m_y new pixels. S_2 is concealed by the DE scheme using the $(2m_z^2 + 2m_z + 1) - ary$ notational system with the group Q_2 consisting of other pixels (p_1, q_1), and S_3 is concealed by the EMD scheme using the $(2m_y + 1) - ary$ notational system with the group Q_3 of m_y new pixels. The four group of pixels in the stego image corresponding to the groups Q_0, Q_1, Q_2 and Q_3 of the cover image are denoted respectively as Q_0', Q_1', Q_2', and Q_3' after the embedding process. The above process is repeated for all the secret data blocks leading to finally in the construction of the stego image.

3.2.2 The UMIS Algorithm

The four secret digits S_0, S_1, S_2 and S_3, represent a single secret data block of size L, and four groups of pixels Q_0, Q_1, Q_2 and Q_3, are utilized to conceal them, respectively. The entire concealing process discussed in Section 3.2.1 is summarized in Algorithm 1.

We now demonstrate the proposed UMIS scheme by considering a specific example.

Algorithm 1 UMIS: Unified Modulus-based Image Steganographic Scheme.

Input: Cover image, parameters (m_x, m_y, m_z) where $(m_z < m_x < m_y)$, and a stream of secret data blocks B

Output: Stego image

Require: Lookup $T_1(0 : 2m_x, 0 : 2m_y)$ with entries $[0 : t_1^{MAX}]$

Require: Lookup $T_2(0 : t_1^{MAX}, 0 : t_1^{MAX})$ with entries $[0 : \left(t_1^{MAX} + 1\right)^2 - 1]$

while $Num(B) \neq 0$ **do**

$Q_0 = (p_0, q_0), Q_2 = (p_1, q_1)$

$Q_1 = m_y$-pixels, $Q_3 = m_y$-pixels

$(t_{2_x}, t_{2_y}) \leftarrow T_2(B)$ ▷ Coordinates of B in T_2

$(t_{1_{x_0}}, t_{1_{y_0}}) \leftarrow T_1(t_{2_x})$ ▷ Coordinates of t_{2_x} in T_1

$(t_{1_{x_1}}, t_{1_{y_1}}) \leftarrow T_1(t_{2_y})$ ▷ Coordinates of t_{2_y} in T_1

$S_0 \leftarrow t_{1_{x_0}}, S_1 \leftarrow t_{1_{y_0}}$

$S_2 \leftarrow t_{1_{x_1}}, S_3 \leftarrow t_{1_{y_1}}$

$DE[S_0, S_2]$ using (Q_0, Q_2) groups with $(2m_z^2 + 2m_z + 1)$

$EMD[S_1, S_3]$ using (Q_1, Q_3) groups with $(2m_y + 1)$

end while

Consider a grayscale cover image of 512×512 resolution, containing 262144 8-bit pixels. Using the constructed lookup tables T_1 and T_2 in figures 3.3 and 3.2, consider the secret data block B is $(001111000010)_2 = (962)_{10}$, $m_z = 1$ and values of the pixels in the cover image are $(77, 121, 128, 139, 150, 193, 199, 224, 221, 152)$. We obtain for the $x - y$ coordinate of the B in lookup table T_2. The (t_{2_x}, t_{2_y}) of the B is (2,30). Then we obtain the $x - y$ coordinate of t_{2_x} and t_{2_y} in lookup table T_1. The $(t_{1_{x_0}}, t_{1_{y_0}})$ of t_{2_x} is (2,0) and $(t_{1_{x_1}}, t_{1_{y_1}})$ of t_{2_y} is (0,6). Now these two coordinates are considered as the four secret digits: $S_0 \leftarrow 2$, $S_1 \leftarrow 0$, $S_2 \leftarrow 0$, and $S_3 \leftarrow 6$. The S_0 and S_2 are concealed by the DE scheme using (Q_0, Q_2) groups of pixels with 5-ary system. The S_1 and S_3 are concealed by the EMD scheme using (Q_1, Q_3) groups of pixels with 5-ary system. The groups are: $Q_0 \leftarrow (77, 121)$, $Q_2 \leftarrow (193, 199)$, $Q_1 \leftarrow (128, 139, 150)$, and $Q_3 \leftarrow (224, 221, 152)$. After applying the embedding functions of the EMD in Eq. 2.1.1 and DE Eq. 2.2.1, we get the stego pixels $(77, 121, 128, 139, 150, 192, 199, 224, 221, 151)$.

3.2.3 Extracting Process

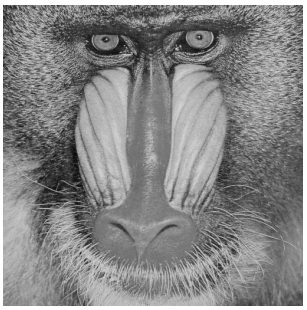
Extracting the secret data from the stego image received is done by reversing the process of embedding. The permutation key and the parameters m_x, m_y and m_z are assumed to be known to the receiver. The lookup tables T_1 and T_2 are constructed in the same way as constructed by the sender, and the correct order of stego pixels Q'_0, Q'_1, Q'_2, and Q'_3 are obtained using the cipher key. The values of S_0 and S_2 are extracted by using the extraction function given by Eq. 2.2.3, whereas the values of S_1 and S_3 are extracted by using the extraction function given by Eq. 2.1.2. The coordinate values $t_{1_{x_0}}$ and $t_{1_{y_0}}$ are obtained as $t_{1_{x_0}} \leftarrow S_0$ and $t_{1_{y_0}} \leftarrow S_1$ and the coordinate values $t_{1_{x_1}}$ and $t_{1_{y_1}}$ are obtained as $t_{1_{x_1}} \leftarrow S_2$ and $t_{1_{y_1}} \leftarrow S_3$. The values of the two entries at coordinates $(t_{1_{x_0}}, t_{1_{y_0}})$ and $(t_{1_{x_1}}, t_{1_{y_1}})$ in T_1 represent the coordinates t_{1_x} and t_{1_y} which represent the coordinates in T_2 of the secret data block B. The above process is repeated until all secret data blocks are extracted from the stego image.

3.3 Experimental Results and Analysis

The proposed scheme provides a high embedding rate and preserves the quality of the cover image in the resulting stego image. This is achieved through the use of a multilevel steganographic scheme where entries in one lookup table act as pointers for a secret data block in another lookup table. Also, concealing a smaller number of secret digits (as pointers), rather than a large number of secret digits representing a large secret data block, helps to maintain image quality.

Figure 3.4 represents a commonly-used cover images: Lena, Baboon, Airplane, and Cameraman. Stego images of the cover images resulting from the proposed scheme are shown in Figure 3.5. It is seen from this figure that the quality of the stego images is almost the same, clearly demonstrating the superiority of the multilevel steganographic scheme.

(a) Lena

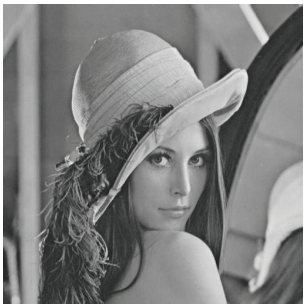


(b) Baboon

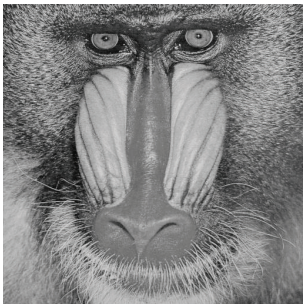
(c) Airplane

(d) Cameraman

Figure 3.4: Commonly-used cover images.



(a) Lena



(b) Baboon

(c) Airplane

(d) Cameraman

Figure 3.5: Resulting stego images: (a) PSNR 54.797 dB, (b) PSNR 54.88 dB,(c) PSNR 54.78 dB, (d) PSNR 55.08 dB.

The evaluation performance in terms of PSNR and embedding rate (bpp) for different sets of the parameter values in the lookup tables is presented in Figure 3.6 for the commonly used cover images. It is seen from this figure that our proposed modulus-based scheme achieves higher embedding rates and provides better image quality for the resulting stego images compared to that provided by using the EMD or the DE scheme alone.

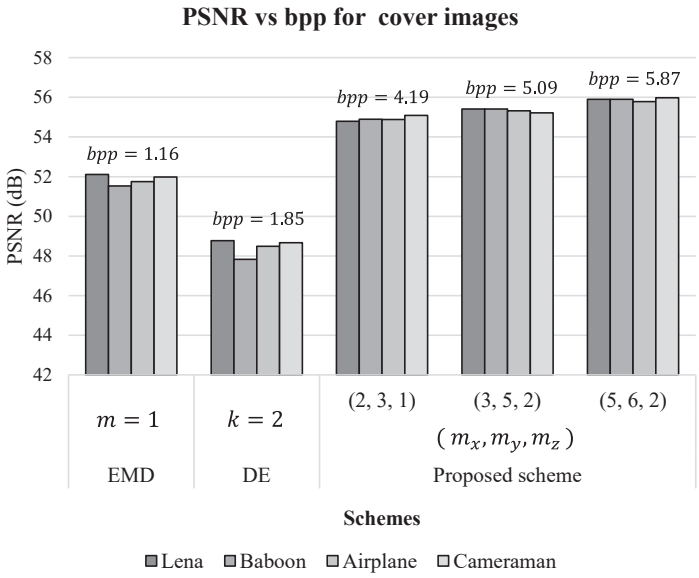


Figure 3.6: PSNR vs bpp evaluation for the Lena, Baboon, Airplane, and Cameraman cover images.

We present a comparison between our proposed UMIS scheme and other recent works in Table 3.1, using the same cover images with a 100% payload. The schemes are categorized into three groups: interpolation-based, encryption-based, and modulus-based schemes. We observe that our proposed steganographic scheme achieves the highest PSNR/SSIM values (least distortion) compared to all other schemes, in terms of the visual quality of the resulting stego images. Additionally, our scheme achieves the highest embedding rate compared to other works. When we use Lena as the cover image and select the parameters $m_x = 2$, $m_y = 3$ and $m_z = 1$, our scheme achieves an embedding rate that is 3.61 and 2.26 times higher than the embedding rate achieved by the EMD [37] and DE [39] schemes, respectively. This demonstrates that our combined steganographic scheme performs more efficiently than the original EMD and DE schemes.

Table 3.1: PSNR/SSIM and embedding rate comparisons of other recent works and the proposed scheme.

Scheme category	Scheme	Cover image								
		Lena			Baboon			Airplane		
		PSNR (dB)	SSIM	bpp	PSNR (dB)	SSIM	bpp	PSNR (dB)	SSIM	bpp
Interpolation based	[13] (2012)	22.32	0.867	1.32	21.24	0.651	1.32	23.76	0.797	1.34
	[16] (2018)	40	-	1.2	35.99	-	-	40	-	1.2
	[17] (2019)	46.88	0.9352	1.87	47.19	0.9922	1.88	39.55	0.914	1.445
	[19] (2019)	32.24	-	1.8	22.98	-	2.1	29.42	-	1.2
	[22] (2020)	34.18	-	1.41	24.69	-	2.4	32.84	-	1.27
	[23] (2021)	-	-	-	21.53	-	2.1	28.11	-	1.32
Encryption based	[24] (2016)	36.3	-	0.25	37.5	-	0.25	37.8	-	0.25
	[30] (2019)	44 .41	-	0 .5	27 .11	-	0 .5	44 .94	-	0 .5
	[35] (2021)	48	-	**2.59**	22	-	0.9	23	-	2.4
	[36] (2022)	-	-	-	28.29	0.97	0.063	45.7	**0.999**	0.125
Modulus based	[37] (2006)	51.8	**0.9971**	1.16	**51.8**	**0.9989**	1.16	**51.79**	0.9967	1.16
	[38] (2009)	47.92	0.9904	2.3	47.95	**0.9975**	2.3	47.97	0.9897	2.3
	[39] (2009)	**52.1**	0.9965	1.85	46.3	0.9962	1.85	46.7	0.996	1.85
	[41] (2018)	42.74	-	2.1	36.63	-	**2.6**	43.23	-	2.1
	[43] (2019)	51.8	-	1.16	-	-	-	51.54		1.15
	[44] (2020)	39.5	-	0.22	38.5	-	0.22	38.5	-	0.22
	[45] (2020)	48.66	-	2	48.52	-	2	48.52	-	2
	[46] (2020)	43.2	-	0.75	38.5	-	0.6	46.5	-	0.72
	[47] (2021)	43.74	-	2.5	43.73	-	2.5	43.74	-	**2.5**
	UMIS (Proposed scheme)	**54.112**	**0.9984**	**4.19**	**54.789**	**0.9989**	**4.19**	**54.782**	**0.9992**	**4.19**

The red color represents the 1^{st} best result and the green color represents the 2^{nd} best result.

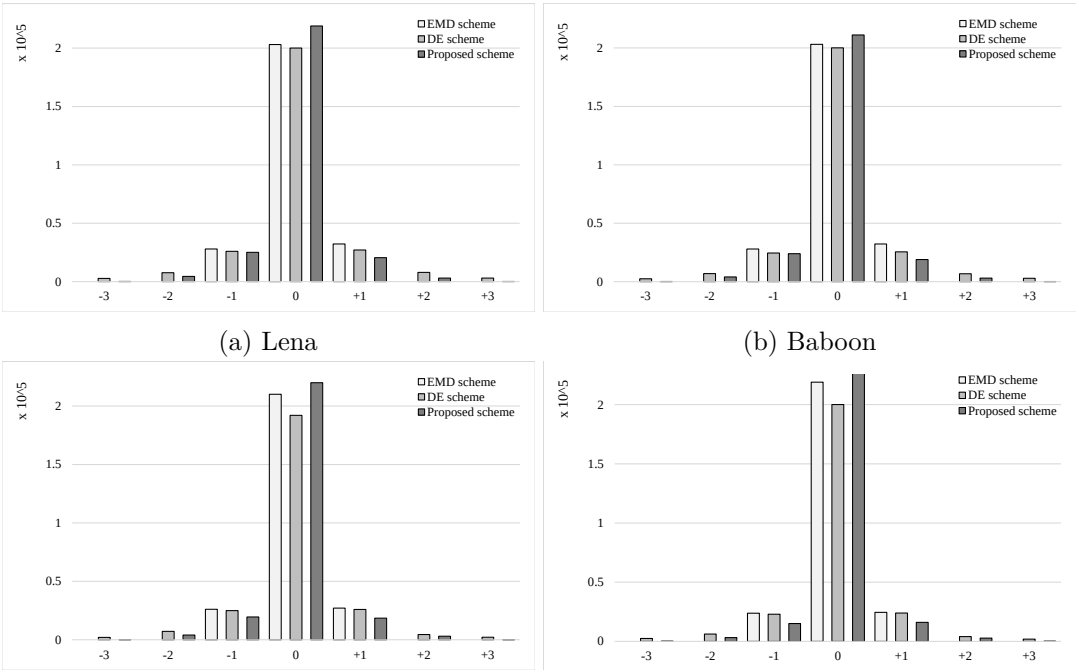
3.3.1 Digital Image Steganalysis

The objective of digital image steganalysis is to detect the presence of hidden secret data in an image. The robustness of the steganographic scheme against steganalysis attempts is a crucial evaluation metric. The steganalyst attempts to identify whether secret data exists in digital images. Blind steganalysis is a class of steganalysis techniques where the steganographic scheme is unknown to the steganalyst [95]. Blind steganalysis is considered the most effective class of steganalysis to attack a given stego image since it does not require knowledge of the applied steganographic scheme. Techniques in this class of steganalysis do not provide any information about the applied steganographic scheme. However, they can determine if an image has secret data or not. Three blind steganalysis attacks used in this paper are PVD histogram analysis, salt and pepper noise analysis, and a machine learning-based anti-detection test. It is worth mentioning that results obtained in these attacks are after applying the permutation process that uses the same cipher and decipher sequences.

3.3.1.1 PVD histogram analysis

In the PVD histogram steganalysis attack, histogram of the pixel differences between the cover and stego images is obtained. Figure 3.7 shows the PVD histogram using the

cover images and their stego images with 100% payload obtained by using the proposed UMIS scheme, as well as those obtained by EMD and DE schemes using the same cover image and the same payload. It is seen from this figure that the histogram value of the pixels with a zero difference is the highest, and the values with a non-zero difference are the lowest, corresponding to the proposed scheme. Therefore, it can be concluded that the proposed UMIS scheme is more robust against the PVD histogram steganalysis attack than EMD and DE schemes are.



(a) Lena (b) Baboon

(c) Airplane (d) Cameraman

Figure 3.7: The PVD histogram for cover images and their stego images generated by the EMD, DE and proposed schemes.

3.3.1.2 Salt and Pepper Noise Analysis

Evaluating the robustness of a steganographic scheme in a noisy environment is performed by applying salt and pepper noise. The noise, which changes some pixels to black or white, is added to the stego images at different densities, from 10% to 100%.

Figure 3.8 shows the PSNR values for the corrupted stego images, where the stego images themselves are obtained using the proposed scheme as well as that by using the EMD and DE schemes. It is seen from this figure that the proposed scheme provides the highest PSNR values at all the levels of the noise indicating that the attacker would be led to believe less that there is a secret data hidden in the stego image produced by the proposed scheme.

The bit error rate (BER) is the ratio of the total number of secret data bits extracted erroneously to the total number of embedded bits. A lower value of BER indicates that the receiver would be able to extract the secret data more successfully despite the fact that the stego image is corrupted by the salt and pepper noise. Figure 3.9 shows the BER values for the corrupted stego images, where the stego images themselves are obtained by using the proposed scheme as well as that by using the EMD and DE schemes. It is seen from this figure that the proposed scheme provides the lowest BER values at all levels of the noise.

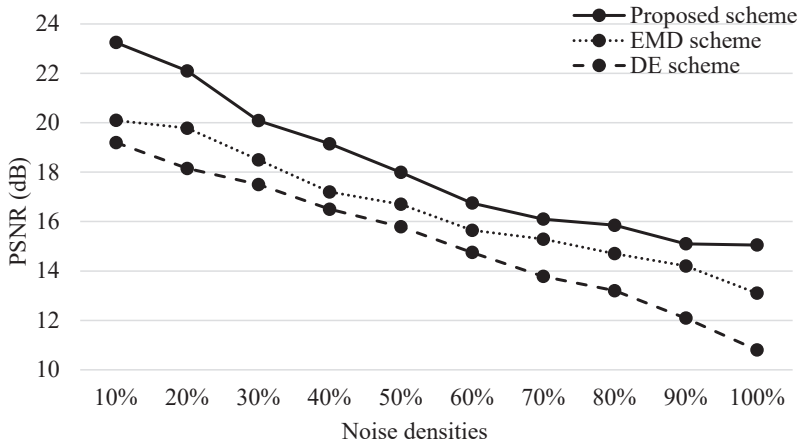


Figure 3.8: Comparison of the average PSNR for salt and pepper noisy images with noise densities ranging from 10% (0.1) to 100% (1.00).

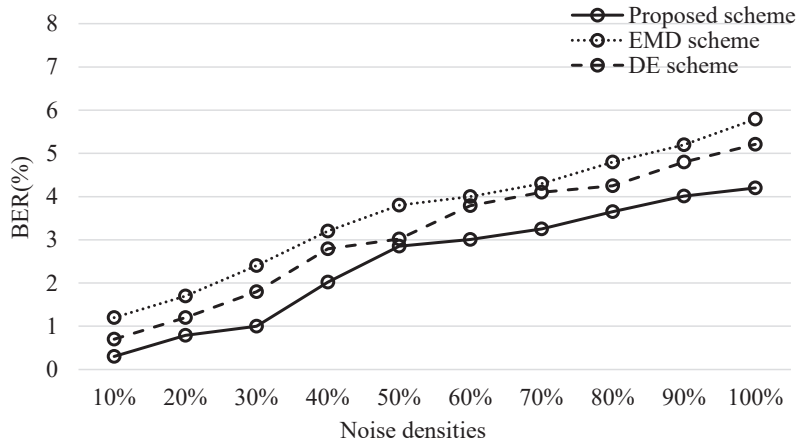


Figure 3.9: Comparison of the average BER for salt and pepper noisy images with noise densities ranging from 10% (0.1) to 100% (1.00).

3.3.1.3 Machine Learning-based Anti-detection Test

This blind steganalysis attack is based on using a classifier to make a decision [96]. The anti-detection test indicates that the steganographic scheme can defeat detection of hidden secret data in a way that makes suspicious stego images appear to be clean cover images. To classify stego and clean images into either suspicious or clean sets, a binary classifier is required. This requires a training set that contains statistical features for both cover and stego images. Statistical features can include histograms, contrast, mean, variance, skewness, entropy, and other statistical features. The binary classifier is trained using the extracted statistical features. After training, the classifier generates class labels using the statistical features of test images.

This anti-detection test comprises several steps. First, statistical features of images are extracted and stored in the dataset. Next, a support vector machine (SVM) binary classifier is created using supervised machine learning and optimized parameters for testing accuracy. Then, the classifier is trained on the statistical features in the defined dataset. Finally, the trained classifier is ready to predict the test images by labeling them as either cover or stego classes.

The proposed steganographic scheme is tested using the blind steganalysis attack. We generate 2000 grayscale stego images of size 512×512 using our scheme and used 1800 cover and stego images to train the SVM classification model. The remaining

200 images are used to test the model. To analyze the classification accuracy, we use a receiver operating characteristic (ROC) curve as a standard graphical representation to interpret the performance of the steganographic scheme. The ROC curve provides an indicator called the area under the curve (AUC), which is the total area under the curve. In digital image steganalysis, a larger AUC indicates that the steganalysis attack is strong and the steganographic scheme is vulnerable. Conversely, for the steganographic scheme, a smaller AUC and closer proximity to the diagonal indicates that it is more resilient against the attack [97]. Figure 3.10 shows the ROC performance curves for the SVM classifier for our proposed steganographic scheme, as well as for the EMD and DE schemes, where a 100% capacity is concealed into cover images. It can be seen from this figure, our proposed scheme has the smallest AUC and is closest to the diagonal, compared to the other steganographic schemes. This indicates that our proposed steganographic scheme has lower true positive rates (i.e., detection probabilities) because it does not disturb the statistical features of cover images, thereby avoiding detection using the SVM detector.

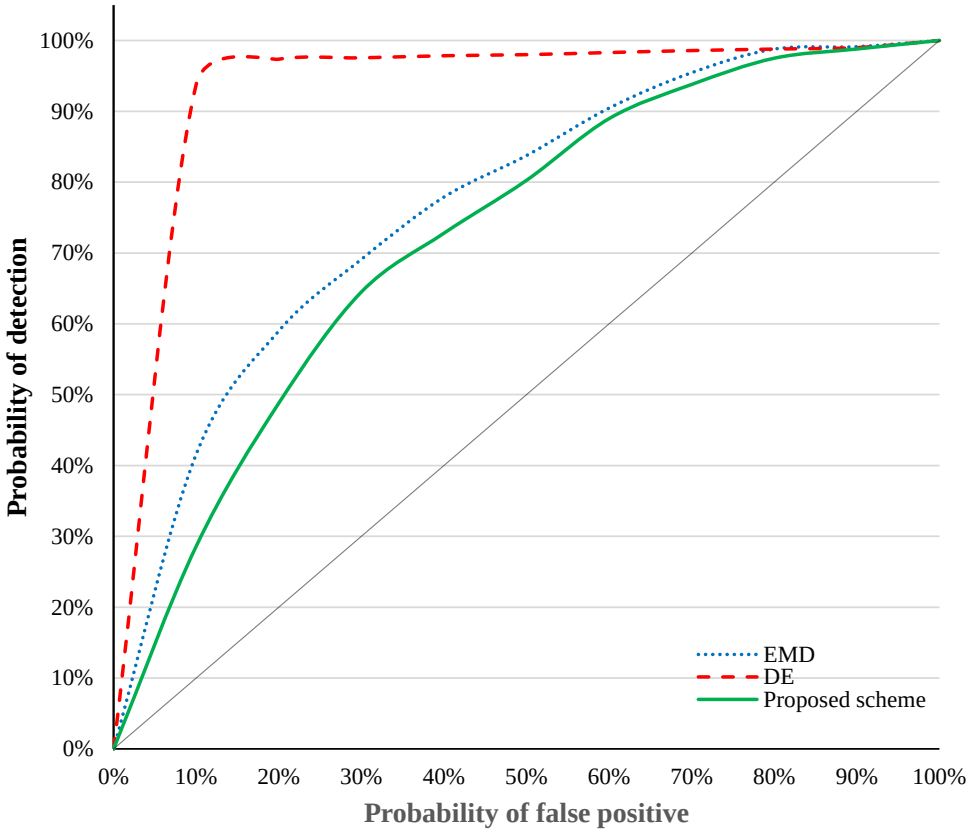


Figure 3.10: ROC curves of the EMD, DE and proposed schemes.

3.4 Summary

This chapter presents a novel image steganographic scheme that combines the EMD and DE schemes, aiming to achieve a high embedding rate and produce high-quality stego images simultaneously. The proposed scheme does not require dividing the secret data blocks. The secret data block is searched for in pre-constructed lookup tables, providing the corresponding four secret digits. These digits are then concealed within four groups of pixels in the cover image. The first and third secret digits are concealed using the DE scheme in the first and third groups of pixels, respectively. The second and fourth

secret digits are concealed using the EMD scheme in the second and fourth groups of pixels, respectively. The proposed scheme has been shown to provide high embedding rates and PSNR values. Moreover, experiments have demonstrated the resilience of this scheme against various types of steganalysis attacks.

Chapter 4

UMISPB: A Unified Modulus-based Image Steganographic Scheme with Partitioned Secret Data Blocks

4.1 Introduction

In this chapter, we present a modulus-based steganographic scheme [98] that combines the EMD and DE schemes to hide secret data blocks. The proposed scheme requires dividing the secret data block of size L into two smaller parts L_1 and L_2. The first part L_1 is concealed using the EMD scheme, and the second part L_2 is concealed using the EMD scheme.

In the EMD scheme, each block of size of L bits of the secret data is converted into E secret digits using a $(2b+1)$-ary notational system. If each secret digit is concealed in a group of m pixels in the cover image, then the value of b in the notational system is m. In the EMD scheme, for each secret digit, only one pixel in the group is modified by changing its value by ± 1 or remain unchanged. In the EMD scheme, higher values of m provide higher qualities of the stego images and lower embedding rates with maximum value of 1.16 *bpp* when m is 2. The DE scheme was proposed to deliver higher embedding rates than the rates achieved by the EMD scheme for large sizes of secret data blocks. In the DE, a secret digit represented using the $(2k^2 + 2k + 1)$-ary notation system is concealed in a group of two pixels only. Increasing L requires higher values of k. Higher values of k result in higher embedding rates with minimum value of 1.16 *bpp* when k is 1. However, a higher k deteriorates the quality of the stego image when there is change to the values of the two pixels by $\pm k$. For a given L, the EMD scheme gives a stego image of a certain quality and embedding rate, and similarly, the DE scheme also provides a stego image of a different quality and embedding rate. If the value of L is decreased, the image quality decreases and the embedding rate increases in the EMD scheme, whereas in the DE scheme, the change in the quality and embedding rate of the stego image is opposite to that of the EMD scheme.

In the view of the above observation, it would be worth studying as to whether we can develop a steganographic scheme that provides an embedding rate higher than the highest rate provided by the DE scheme and a quality of the resulting stego image superior to that achievable by the EMD scheme, by dividing each secret data block of size L into two parts L_1 and L_2, and concealing the two parts using the EMD and DE schemes, respectively, and continuing this process until all the secret data blocks are

concealed. To further enhance the image quality and embedding rate, two lookup tables of entries that represent the two parts L_1 and L_2 are utilized to conceal secret digits.

4.1.1 Analysis of Dividing Secret Data Blocks into Two Parts

For the analysis in this section, we first generate a single payload (PL) once and use the same PL for generating secret data blocks of various sizes and their different divisions. A single cover image size of 512×512 is used. Table 4.1 shows the performance evaluation of concealing a maximum payload by the EMD, DE and the merged EMD-DE schemes. It can be seen from this table that the minimum PL is concealed when the size of the secret data block is 5 using the EMD scheme. The minimum PL is obtained to guarantee that this payload can fit into the cover image using different sizes of the secret data blocks. The minimum PL is 163.840 *Kbits* of secret data. Table 4.1 indicates that the embedding rate and the image quality are improved when the size of the secret data block increases. Additionally, for better image quality, the size of L_1 must be larger than the size of L_2, while for a higher embedding rate the size of L_2 has to be larger than size of L_1.

Table 4.1: Performance evaluation for the EMD, DE, and the merged EMD-DE scheme.

EMD scheme				**DE scheme**			**Merged EMD-DE scheme**			
(L)	**Tot. PL** ($Kbits$)	(bpp)	**PSNR** (dB)	**Tot. PL** ($Kbits$)	(bpp)	**PSNR** (dB)	(L_1, L_2)	**Tot. PL** ($Kbits$)	(bpp)	**PSNR** (dB)
11	240299	0.92	58.568	240,299	0.92	32.845	(6, 5)	320,398	1.22	58.897
							(5, 6)	411,941	1.57	58.755
10	218453	0.83	56.443	262,144	1	34.911	(5, 5)	374,491	1.43	57.145
9	196608	0.75	54.576	235,930	0.9	37.921	(5, 4)	337,042	1.29	55.873
							(4, 5)	393,216	1.5	55.678
8	262144	1	54.019	262,144	1	40.012	(4, 4)	349,525	1.33	54.772
7	229376	0.88	53.323	229,376	0.88	42.976	(4, 3)	305,835	1.17	54.281
							(3, 4)	367,002	1.4	53.969
6	196608	0.75	53.057	262,144	1	45.07	(3, 3)	314,573	1.2	53.594
5	163840	0.63	52.575	218,453	0.83	47.819	(3, 2)	262,144	1	53.339
							(2, 3)	327,680	1.25	53.012
4	262144	1	52.1	262,144	1	52.1	(2, 2)	262,144	1	52.955

Table 4.2 shows the PSNR values of the resulting stego image obtained by concealing the minimum PL, namely 163.8 *Kbits*, by the EMD, DE and the merged EMD-DE schemes at a fixed embedding rate of 0.63 *bpp*. It is seen from this table that in the EMD scheme, the larger the size of the secret data block L, the higher the image quality. On the other hand, in the case of the DE scheme, the larger the size of the secret data block L, the lower the image quality.

From the analysis of these two tables, we conclude that dividing the secret data block into two parts, as equally as possible, and concealing them by using the EMD and DE schemes, respectively, lead to providing a performance in terms of both the image quality and the embedding rate, that is better than that achieved by concealing the entire secret data block using the EMD or DE scheme alone.

4.2 Proposed Concealing Process

In the proposed scheme, a secret data block of size L bits is divided into parts of sizes L_1 and L_2, respectively. The first part is represented into $(2n+1)$-ary notational system resulting in two secret digits each of which are concealed by the EMD scheme into a group of pixels of the cover image. The other part is represented into $(2k^2+2k+1)$-ary notational system resulting into a secret digit, which is concealed by the DE scheme into a group of two pixels of the cover image. For a robust scheme, the chaotic-based permutation mechanism, explained in Section 2.3, is performed prior to the process of concealing secret data.

Table 4.2: Image qualities for the EMD, DE, and the merged EMD-DE scheme using the minimum payload.

Tot. PL (*Kbits*)	(*bpp*)	(*L*)	EMD scheme			DE scheme			Merged EMD-DE scheme			
			PSNR (*dB*)	Consumed pixels/(*L*)	Levels (±)	PSNR (*dB*)	Consumed pixels/(*L*)	Levels (±)	(L_1, L_2)	PSNR (*dB*)	Consumed Pixels/(L_1, L_2)	Levels (±)
163.84	0.63	11	66.643	18	±1	38.925	2	±32	(9, 2)	67.523	10	±1
									(8, 3)	67.305	10	±2
									(7, 4)	67.187	10	±3
									(6, 5)	66.919	10	±4
									(5, 6)	66.775	8	±6
									(4, 7)	66.493	6	±8
									(3, 8)	66.312	3	±11
									(2, 9)	66.176		±16
		10	64.275	15		44.722		±23	(8, 2)	65.387	10	±1
									(7, 3)	65.223	10	±2
									(6, 4)	64.986	10	±3
									(5, 5)	64.824	8	±4
									(4, 6)	64.611	6	±6
									(3, 7)	64.399	3	±8
									(2,8)	64.155		±11
		9	62.865	12		47.903		±16	(7, 2)	63.633	10	±1
									(6, 3)	63.471	10	±2
									(5, 4)	63.242	8	±3
									(4, 5)	63.058	6	±4
									(3, 6)	62.515	3	±6
									(2, 7)	62.287		±8
		8	61.429	8		49.744		±11	(6, 2)	61.977	10	±1
									(5, 3)	61.856	8	±2
									(4, 4)	61.713	6	±3
									(3, 5)	61.505	3	±4
									(2, 6)	61.324		±6
		7	58.192	8		51.016		±8	(5, 2)	58.684	8	±1
									(4, 3)	58.467	6	±2
									(3, 4)	58.371	3	±3
									(2, 5)	58.053		±4
		6	56.644	8		53.038		±6	(4, 2)	56.917	6	±1
									(3, 3)	56.705	3	±2
									(2, 4)	56.513		±3
		5	54.442	6		54.388		±4	(3, 2)	55.699	3	±1
									(2, 3)	55.642		±2
		4	52.603	4		52.566		±3	(2, 2)	53.939	3	±1

4.2.1 Concealing the First Part of the Secret Data Block

Combining the EMD and DE schemes into a single steganographic scheme improves the performance in terms of embedding rates and maintaining image qualities. The proposed steganographic scheme uses a different strategy for concealing the secret data block. Each secret data block L is divided into two parts L_1 and L_2. For the first part L_1, the EMD scheme is performed by constructing an $x - y$ coordinate lookup table. The length of L_1 is denoted by the cut-off length using the EMD scheme (C_{EMD}). The entries of the lookup table represent all possible values of L_1, with values in the range $[0, 2^{C_{EMD}} - 1]$. The lookup table is situated on the x-coordinate and y-coordinate axes, with each axis having its own notational system for expressing secret digits. Specifically, the x-coordinate axis has all possible $2m_x$ values of the $(2m_x + 1)$ -ary system, while the y-coordinate axis has all possible $2m_y$ values of the $(2my + 1)$-ary system.

Once the $x - y$ coordinate lookup table is constructed, the coordinates (x, y) are obtained by searching for the value of L_1. The values of the coordinate (x, y) represent secret digits S_0 and S_1, where S_0 corresponds to the x-coordinate and S_1 corresponds to the y-coordinate, both representing the value of L_1. These two secret digits are concealed in two groups of pixels Q_0 and Q_1, respectively, using the EMD scheme. The Q_0 group requires m_x pixels to conceal the secret digit S_0, while the Q_1 group requires m_y pixels to conceal the secret digit S_1.

Figure 4.1 illustrates the $x - y$ coordinate lookup table when $C_{EMD} = 4$ bits, $m_x = 2$, and $m_y = 2$. Since $C_{EMD} = 4$ bits, the lookup table has 16 (2^4) entries, and values of these entries are in the range $[0, 2^4 - 1]$ or $[0000, 1111]_2$. Each value in the lookup table represents the first part L_1 of the secret data block of size L. It can be seen in this figure that the x-coordinate axis is in the range [0, 4] for a group of 2 pixels, and the y-coordinate axis is in the range [0, 4] for a group of 2 pixels. There is a redundancy in

the table of 9 items, which can be calculated as follows.

$$((2m_x + 1)(2m_y + 1) - 1) - (2^{C_{EMD}} - 1) \tag{4.2.1}$$

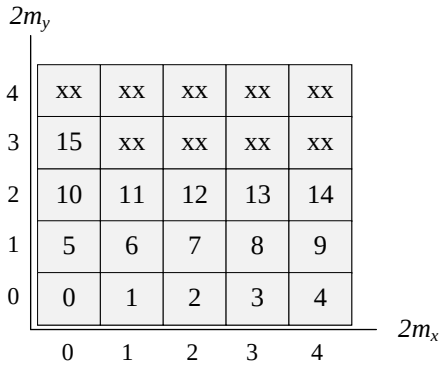


Figure 4.1: $x - y$ coordinate lookup table when $C_{EMD} = 4$ bits.

The redundancy in the $x - y$ coordinate lookup table can be reduced as much as possible by selecting an efficient combination of m_x and m_y. In general, the product of $2m_x$ and $2m_y$ should be equal to the number of entries $2^{C_{EMD}}$ in the lookup table.

4.2.2 Concealing the Second Part of the Secret Data Block

For the second part L_2, a new z-coordinate axis is added to conceal the secret digit S_2 which represents the L_2 using the DE scheme. The value of L_2 is denoted by the cut-off length using the DE scheme C_{DE}. The z-coordinate axis has the vector set of S_{m_z} in the $(2m_z^2+2_{m_z}+1)$-ary notational system. Searching for the L_2 in the distance pattern D_{m_z} determines which vector s_{m_z} is used to conceal in the third group Q_2 of 2-pixels (p, q). Figure 4.2 illustrates the z coordinate lookup table when $C_{DE} = 3$. For this cut-off, the suitable value for the $m_z = 2$. It can be seen from this figure that the z-coordinate axis also represents the DE distance pattern D_2 for the L_2. It is worth mentioning that after concealing is completed on the current set of groups of pixels, a new set of groups of pixels is introduced for concealing another block of secret data. The above process is repeated for all the secret data blocks leading to finally in the construction of the stego image.

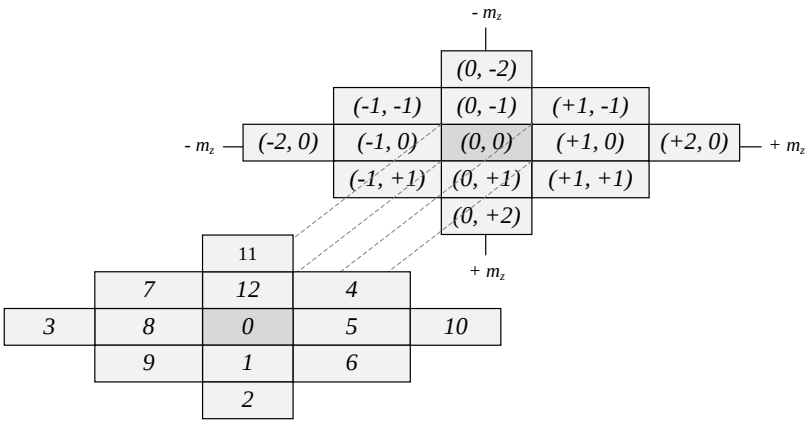


Figure 4.2: z-coordinate lookup table when $C_{DE} = 3$.

4.2.3 The UMISPB Algorithm

The three secret digits S_0, S_1 and S_2, represent a single secret data block of size L, and three groups of pixels Q_0, Q_1 and Q_2, are utilized to conceal them, respectively. The entire concealing process discussed in Sections 4.2.1 and 4.2.2 are summarized in Algorithm 2.

Algorithm 2 UMISPB: Unified Modulus-based Image Steganographic Scheme with Partitioned Secret Data Blocks.

Input: Cover image, cut-offs (C_{EMD}, C_{DE}), a stream of L-bit secret data blocks B;
Output: Stego image;
Require: $x-y$ coordinate lookup table $(0:2m_x, 0:2m_y)$;
Entries: $[0:2^{C_{EMD}}-1]$;
Require: $x-y-z$ coordinate lookup table $(-m_z:+m_z, -m_z:+m_z)$;
Entries: $[0:2m_z^2+2m_z]$;
while $Num(B) \neq 0$ **do**
$Q_0 = m_x - pixels, Q_1 = m_y - pixels, Q_2 = (p, q)$;
$(x, y) \leftarrow x-y$ *Lookup table*(L_1); ▷ Coordinates of L_1 in $x-y$ *Lookup table*
$D_{m_z} \leftarrow z$ *Lookup table*(L_2) ▷ D_{m_z} of L_2 in z *Lookup table*
$S_0 \leftarrow x, S_1 \leftarrow y, S_2 \leftarrow D_{m_z}$;
$EMD[S_0]$ *using* Q_0 *group with* $(2m_x+1)-ary$ *system*;
$EMD[S_1]$ *using* Q_1 *group with* $(2m_y+1)-ary$ *system*;
$DE[S_2]$ *using* Q_2 *group with* $(2m_z^2+2m_z+1)-ary$ *system*;
end while

In the EMD scheme, the secret data block of size L can be represented using $\lceil \frac{L}{\log_2(2m+1)} \rceil$ $(2m+1)$-ary secret digits. For example, if $L=7$, then $m=6$ and the number of secret digits required is $\lceil \frac{7}{\log_2(2m+1)} \rceil$ or 2 of 13-ary secret digits to represent a 7-bit secret data block. Each secret digit is concealed into a 6-pixel group, thus, 2 of 13-ary secret digits need 12 pixels to conceal the secret data block of 7-bit size. In the DE scheme, the secret data block of size L can be represented using a distance pattern D_k that contains $(2k^2+2k+1)$ secret digits. For example, if $L=7$, then $k=8$ and D_8 has 145 secret digits.

On the other hand, using the proposed scheme, if the size of the secret data block $L=7$ and the parameters for the $x-y-z$ coordinate lookup table are $m_x=2$, $m_y=2$, $m_z=2$, $C_{EMD}=4$, and $C_{DE}=3$, then the x-axis for the $x-y$ coordinate lookup table is $2m_x$ and it requires 2 pixels, and the y-axis in the table is $2m_y$ and it requires 2 pixels. Also, 2 pixels are required for the C_{DE} with $k=2$ and D_2 has 13 secret digits. This indicates that our scheme requires a total of 6 pixels to conceal the $L=7$, which is fewer pixels than the EMD scheme and has smaller distance patterns than the DE scheme. Table 4.3 shows the pixels required to conceal different sizes of L. Our proposed steganographic scheme consumes fewer pixels and has smaller sizes of distance patterns than the EMD and DE schemes, respectively.

The flowchart of the proposed steganographic scheme is shown in Figure 4.3. The entire embedding process is performed as follows: (i) Reshape the selected cover image into bitstreams. (ii) Permute the locations of the pixels in the cover image using the cipher sequence with the selected parameters (x_0, μ). (iii) Select the cut-offs C_{EMD} and C_{DE}. (iv) Select the suitable (m_x, m_y, m_z) for constructing the $x-y$ and z coordinate lookup tables. (v) Divide the secret data block L into two parts L_1 and L_2, then conceal both parts into the permuted pixels of the cover image using the coordinate lookup tables. (vi) Once all permuted pixels are scanned, reorder the permuted pixels of the generated stego image back to the original order using the decipher sequence.

Table 4.3: Number of pixels, size of the distance patterns for the EMD, DE, and the proposed schemes.

(L)	EMD scheme	DE scheme	Proposed scheme		
	Consumed pixels/(L) when m =2	Distance pattern D/(L)	(L_1, L_2)	Consumed pixels/(L_1)	Distance pattern $D/(L_2)$
11	10	2113	(6, 5)	6	41
			(5, 6)	6	85
10	10	1105	(5, 5)	6	41
9	8	545	(5, 4)	6	25
			(4, 5)	4	41
8	8	265	(4, 4)	4	25
7	8	145	(4, 3)	4	13
			(3, 4)	4	25
6	6	85	(3, 3)	2	13
5	6	41	(3, 2)	2	5
			(2, 3)	1	13
4	4	25	(2, 2)	1	5

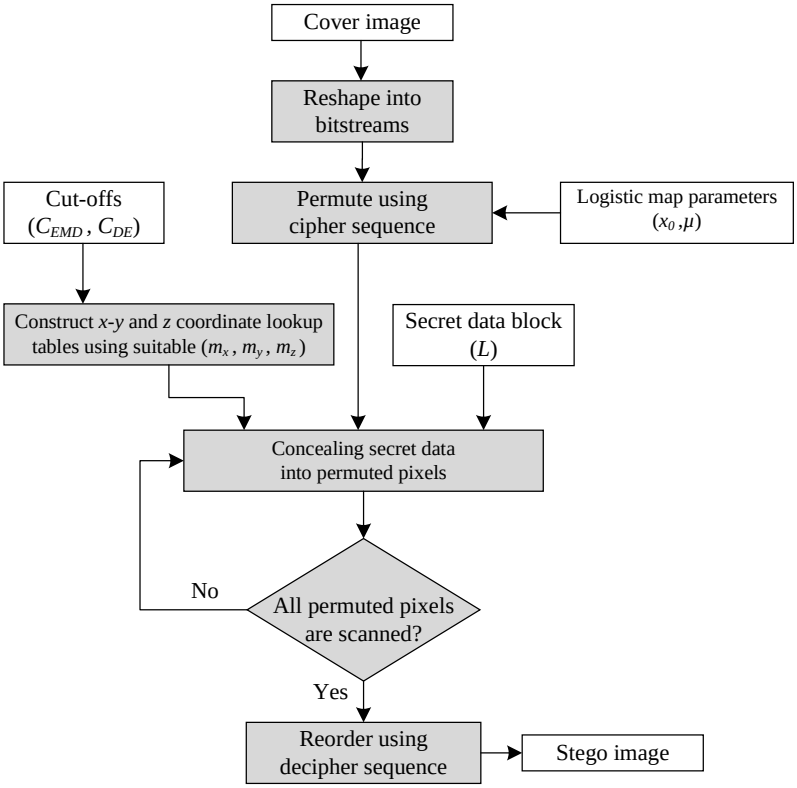


Figure 4.3: Flowchart of the proposed steganographic scheme.

The proposed scheme offers a simplified approach to the conversion of secret data blocks into l-ary digits. The conversion process is avoided, which eliminates the need for complex operations like multiplication and division, especially when the cut-offs C_{EMD} and C_{DE} are large. Instead, the secret data blocks are stored directly in the lookup tables, and the secret digits S_0, S_1, and S_2 are used for concealing. This approach provides an additional layer of security since the secret data are not directly involved in the concealing process.

The complexity of the proposed scheme is primarily limited to the construction of the lookup table. The construction process becomes more complex as the size of L increases and the cut-off lengths C_{EMD} and C_{DE} are increased. Larger cut-off values lead to more entries in the lookup tables, resulting in a more complex construction process. However, this process only needs to occur once at the beginning of the communication session for that specific application. In case the session is expired, or cut-offs are changed, the

lookup table can be reconstructed.

The proposed steganographic scheme is demonstrated using an example as follows. A grayscale cover image with a resolution of 512 × 512, containing 262144 8-bit pixels, is selected. The cover image is reshaped into bitstreams, and the permutation key parameters are set to $x_0 = 0.1$ and $\mu = 1.7$. The pixels in the cover image are permuted using the cipher sequence to generate a single row of permuted pixels. The $x - y$ and z coordinate lookup tables are constructed using the cutoff lengths $C_{EMD} = 4$ and $C_{DE} = 4$. The suitable parameters used are $m_x = 2$, $m_y = 2$, and $m_z = 3$. Figure 4.4 shows the lookup tables used in this example.

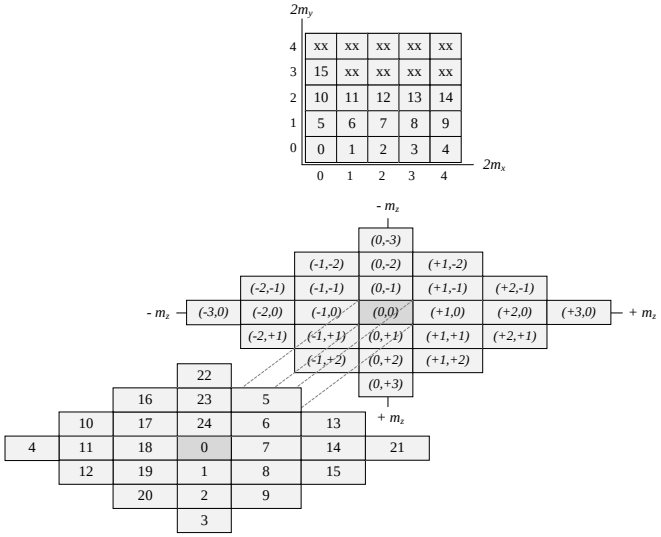


Figure 4.4: Lookup tables when $m_x = 2, m_y = 2, m_z = 4$.

Using the lookup tables in Figure 4.4, a secret data block of size $L = 8$ can be concealed using only 6 pixels. Let the secret data block be $(11101101)_2$ and assume the values of pixels in the cover image are (138, 140, 187, 120, 100, 199). Since $C_{EMD} = 4$ bits, the first 4 bits are considered as L_1, which is $(1110)_2$. To conceal L_1, we search for L_1 in the $x - y$ coordinate lookup table, and obtain the coordinates $(x = 4, y = 2)$. The values of the coordinates are considered the secret digits, with $S_0 \leftarrow x$ and $S_1 \leftarrow y$. We apply the EMD scheme to conceal $S_0 = 4$ in the first group Q_0 of pixels (138, 140). S_0 is concealed by using the embedding function given by Eq. 2.1.1. We calculate F_{EMD} using the 5-ary system. The F_{EMD} is 3, and the difference $s = (4 - 3)\ mod\ (5) = 1$, which indicates that the value of the first pixel is increased by one. The new values of the pixels in the group Q_0' are (139, 140). Next, we apply the EMD scheme again on the second group Q_1 of pixels (187, 120) to conceal $S_1 = 2$. S_1 is concealed by using the embedding function given by Eq. 2.1.1. The F_{EMD} using the 5-ary system. The F_{EMD} is 2, and the difference $s = (2 - 2)\ mod\ (5) = 0$, which means that there are no modifications to the values of the pixels in the group Q_1'.

Concealing the other part L_2 of the secret data block, which is $(1101)_2$, is done by the DE scheme. First, the F_{DE} is calculated on the last group Q_2 of pixels (100, 199) using by using the embedding function given by Eq. 2.2.1. The F_{DE} is 24, and the $d_{25} = (13 - 24)\ mod\ (25) = 14$. The d_{25} is considered as the secret digit S_3, or $S_2 \leftarrow d_{25}$. The d_{25} is searched to get its S_{25} in the z-coordinate lookup table. The corresponding vector is (+2, 0), which implies increasing the first pixel p by 2 and leaving the second pixel q with no changes. This makes the values of the pixels in the group Q_2' to be (102, 199). The new values of the pixels are (139, 140, 187, 120, 102, 199). This process continues until all pixels of the cover image are scanned, and stego pixels are permuted

back to the original order using the decipher sequence. Finally, the bitstream of pixels is reshaped into a 512×512 stego image.

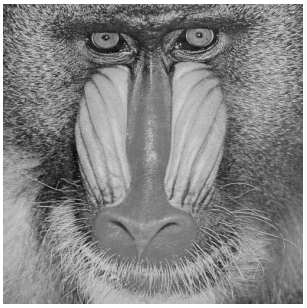
4.2.4 Extracting Process

To extract the secret data blocks from a received stego image, the receiver needs to apply the recovery process which is a reverse of the procedure of the embedding process. The permutation keys and parameters of the $x - y$ coordinate lookup table are known to the receiver. After reordering the locations of the pixels in the stego image, the receiver can extract (S_0, S_1) values using the extraction function given by Eq. 2.1.2. This involves two groups of pixels Q'_0 and Q'_1. The extracted secret digits are the values of the coordinate $x \leftarrow S_0$ and $y \leftarrow S_1$, which are the values coordinates that represent the first part of the secret data block L_1 in the $x - y$ coordinate lookup table. For the third group of two pixels Q'_2, the secret digit S_2 is extracted by using the extraction function given by Eq. 2.2.3 to retrieve the other part of the secret data block L_2. This process continues until all secret data blocks are extracted.

4.3 Experimental Results and Analysis

In this section, we present a demonstration and analysis of the results obtained from the experiments using the proposed steganographic scheme. Figure 4.5 and Figure 4.6 show a set of commonly used cover images Lena, Baboon, Airplane, Cameraman, and their resulting stego images. A maximum payload is concealed in these cover images. We evaluate our scheme using three performance metrics, which are (i) embedding rate (*bpp*) (ii) image quality, and (iii) robustness against steganalysis attacks. For image quality, we obtain the PSNR/SSIM values of the resulting stego images. The resulting stego images are generated after using the selected parameters of $(x_0 = 0.1, \mu = 1.7)$ for the cipher sequence generated by the logistic map, and $C_{EMD} = 5$, $C_{DE} = 4$ for the $x-y$ and z coordinate lookup tables. It can be seen from this figure that the imperceptibility is very high in the stego images after concealing the secret data. Higher PSNR/SSIM values indicate that artifacts can not be detected by the human visual system.

(a) Lena



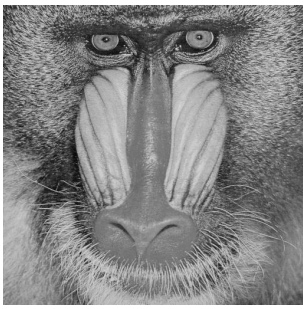
(b) Baboon

(c) Airplane

(d) Cameraman

Figure 4.5: Commonly-used cover images.

(a) Lena



(b) Baboon

(c) Airplane

(d) Cameraman

Figure 4.6: Resulting stego images: (a) PSNR 55.973 dB, (b) PSNR 55.939 dB,(c) PSNR 55.890 dB, (d) PSNR 55.898 dB.

Table 4.4 shows the PSNR values and the embedding rates of the proposed steganographic scheme using different parameters with 100% payload. Our proposed scheme combines two techniques to achieve a higher embedding rate. Specifically, we employ the EMD scheme in the x and y axes and the DE scheme in the z axis. To carry the secret data block of size L, we use three groups of pixels. The embedding rate and image quality are primarily determined by the cut-off lengths. We select the

parameters (m_x, m_y, m_z) to fit these cut-off lengths in the lookup tables. It can be seen from this table that our scheme provides higher embedding rates but with increased distortions when using higher cut-off lengths. Conversely, using fewer cut-off lengths results in lower embedding rates but with less distortion. It is also seen from this table that the balanced performance is when the cut-off sizes are approximately equal.

Table 4.4: PSNR (dB) and embedding rate (bpp) results of the proposed scheme using different parameters with 100% payload.

Cover image	C_{EMD} = 4			C_{EMD} = 5				C_{EMD} = 6			
	m_x=2; m_y=2			m_x =3; m_y =2				m_x =4; m_y =4			
	C_{DE}=2	C_{DE}=3	C_{DE}=4	C_{DE}=2	C_{DE}=3	C_{DE}=4	C_{DE}=5	C_{DE}=2	C_{DE}=3	C_{DE}=4	C_{DE}=5
	m_z=1	m_z=2	m_z=3	m_z=1	m_z=2	m_z=3	m_z=4	m_z=1	m_z=2	m_z=3	m_z=4
Lena	59.987	58.869	57.718	57.593	56.365	55.973	54.887	55.465	54.113	53.618	52.705
Cameraman	59.512	58.365	57.263	57.078	56.489	55.898	54.931	55.711	54.074	53.395	52.595
Baboon	59.321	58.123	57.983	57.589	56.312	55.939	54.822	55.779	54.149	53.431	52.684
Airplane	59.307	58.179	57.04	57.071	56.41	55.89	54.945	55.817	54.177	53.459	52.583
Gray (all pixels 128)	58.794	57.473	56.988	56.453	55.742	54.86	53.892	54.547	53.188	52.937	51.979
Embedding rate (*bpp*)	1.75	1.85	1.97	2.18	2.25	2.37	2.49	2.61	2.76	2.97	3.16

In terms of embedding rate, our proposed scheme combines two schemes, leading to a higher embedding rate. Specifically, the EMD scheme is employed in the x and y axes, and the DE scheme is employed in the z axis, and three groups of pixels are used to embed the secret data block. The parameters (m_x, m_y, m_z) play a dominant role in determining the embedding rates and image quality. It can be seen in Table 4.4, our scheme provides high embedding rates with increased distortions when higher cut-off lengths are used, while less cut-off lengths offer low embedding rates and less distortion.

We present a comparison between our proposed UMISPB scheme and other recent works in Table 4.5, using the same cover images with a 100% payload. The schemes are categorized into three groups: interpolation-based, encryption-based, and modulus-based schemes. We observe that our proposed steganographic scheme, which employs $x - y$ and z coordinate lookup tables, achieves the highest PSNR/SSIM values (least distortion) compared to all other schemes, in terms of the visual quality of the resulting stego images. Additionally, our scheme achieves the highest embedding rate compared to other works. When we use Lena as the cover image and select the parameters C_{EMD} = 5, $C_{DE} = 5$, our scheme achieves an embedding rate that is 2.15 and 1.35 times higher than the embedding rate achieved by the EMD [37] and DE [39] schemes, respectively. This demonstrates that our combined steganographic scheme performs more efficiently than the original EMD and DE schemes. Table 4.5 also includes the results of the proposed UMIS scheme when the size of the secret data block is 10 bits. It can be seen that the UMIS scheme provides a higher embedding rate than the one provided by the UMISPB scheme. On the other hand, the UMISPB scheme provides a higher image qualities for the resulting stego image than the one provided by the UMIS scheme.

Table 4.5: PSNR/SSIM and embedding rate comparisons of other recent works and the proposed scheme.

Scheme category	Scheme	Cover image								
		Lena			Baboon			Airplane		
		PSNR (dB)	SSIM	bpp	PSNR (dB)	SSIM	bpp	PSNR (dB)	SSIM	bpp
Interpolation based	[13] (2012)	22.32	0.867	1.32	21.24	0.651	1.32	23.76	0.797	1.34
	[16] (2018)	40	-	1.2	35.99	-	-	40	-	1.2
	[17] (2019)	46.88	0.9352	1.87	47.19	0.9922	1.88	39.55	0.914	1.445
	[19] (2019)	32.24	-	1.8	22.98	-	2.1	29.42	-	1.2
	[22] (2020)	34.18	-	1.41	24.69	-	2.4	32.84	-	1.27
	[23] (2021)	-	-	-	21.53	-	2.1	28.11	-	1.32
Encryption based	[24] (2016)	36.3	-	0.25	37.5	-	0.25	37.8	-	0.25
	[30] (2019)	44 .41	-	0 .5	27 .11	-	0 .5	44 .94	-	0 .5
	[35] (2021)	48	-	**2.59**	22	-	0.9	23	-	2.4
	[36] (2022)	-	-	-	28.29	0.97	0.063	45.7	0.999	0.125
Modulus based	[37] (2006)	51.8	0.9971	1.16	51.8	**0.9989**	1.16	51.79	0.9967	1.16
	[38] (2009)	47.92	0.9904	2.3	47.95	0.9975	2.3	47.97	0.9897	2.3
	[39] (2009)	52.1	0.9965	1.85	46.3	0.9962	1.85	46.7	0.996	1.85
	[41] (2018)	42.74	-	2.1	36.63	-	**2.6**	43.23	-	2.1
	[43] (2019)	51.8	-	1.16	-	-	-	51.54		1.15
	[44] (2020)	39.5	-	0.22	38.5	-	0.22	38.5	-	0.22
	[45] (2020)	48.66	-	2	48.52	-	2	48.52	-	2
	[46] (2020)	43.2	-	0.75	38.5	-	0.6	46.5	-	0.72
	[47] (2021)	43.74	-	2.5	43.73	-	2.5	43.74	-	2.5
	UMIS (Proposed scheme)	**54.112**	**0.9984**	**4.19**	**54.789**	**0.9989**	**4.19**	**54.782**	**0.9992**	**4.19**
	UMISPB (Proposed scheme)	**59.987**	**0.9992**	2.49	**59.321**	**0.9990**	2.49	**59.307**	**0.9994**	2.49

The red color represents the 1^{st} best result and the green color represents the 2^{nd} best result.

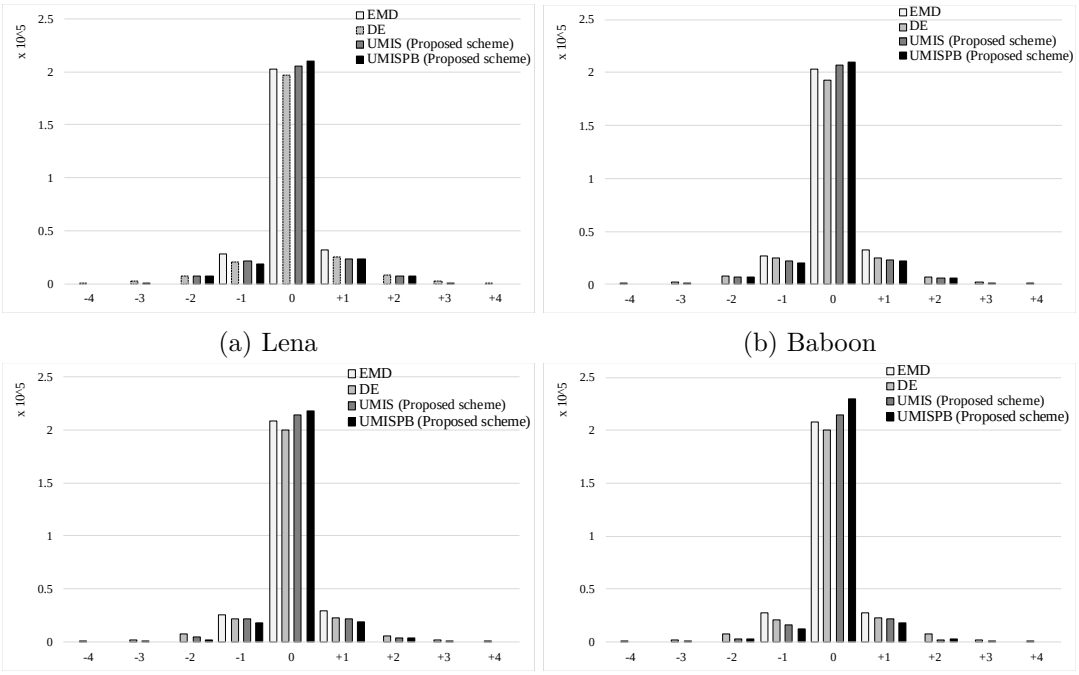
4.3.1 Digital Image Steganalysis

The objective of digital image steganalysis is to detect the presence of hidden secret data in an image. The robustness of the steganographic scheme against steganalysis attempts is a crucial evaluation metric. The steganalyst attempts to identify whether secret data exists in digital images. Blind steganalysis is a class of steganalysis techniques where the steganographic scheme is unknown to the steganalyst [95]. Blind steganalysis is considered the most effective class of steganalysis to attack a given stego image since it does not require knowledge of the applied steganographic scheme. Techniques in this class of steganalysis do not provide any information about the applied steganographic scheme. However, they can determine if an image has secret data or not. Three blind steganalysis attacks used in this paper are PVD histogram analysis, salt and pepper noise analysis, and a machine learning-based anti-detection test. It is worth mentioning that results obtained in these attacks are after applying the permutation process that uses the same cipher and decipher sequences.

4.3.1.1 PVD Histogram Analysis

The pixel value difference (PVD) histogram is a blind steganalysis attack used to detect the presence of hidden secret data in a digital image. It is obtained by calculating the differences between corresponding pixel values in the cover image and its stego image. The resulting PVD values are then collected, and the frequency of their occurrence is counted across the entire image to form the PVD histogram. The underlying idea behind PVD histogram analysis is that any significant differences between the cover image and its stego image will be reflected in the histogram of the pixel differences, thus allowing the detection of the presence of hidden secret data. Figure 4.7 displays the PVD histogram of each the cover image and its stego image obtained by using the proposed UMIS and UMISPB schemes as well as that by using the EMD and DE schemes with 100% payload. A higher number of zeros (bar at location 0) in the histogram indicates fewer modifications made to the cover image. The proposed UMISPB scheme provides a minimum amount of modifications to the cover images, implying a lower chance of detecting hidden secret data. The UMISPB scheme does not apply a consecutive type of concealing due to the permutation process in the pixels of the cover image, so

modifications (artifacts) in the PVD histogram are easily avoided. The PVD histogram is well preserved after embedding, and the zero bars are the highest. Consequently, the stego image is unlikely to be judged as suspicious. Our proposed UMISPB scheme is effective against PVD histogram analysis attack.



(a) Lena (b) Baboon

(c) Airplane (d) Cameraman

Figure 4.7: The PVD histogram for cover images and their stego images generated by the EMD, DE and proposed schemes.

4.3.1.2 Salt and Pepper Noise Analysis

Evaluating the robustness of a steganographic scheme in a noisy environment is performed by applying salt and pepper noise. The noise, which changes some pixels to black or white, is added to the stego images at different densities, from 10% to 100%.

Figure 4.8 shows the PSNR values for the corrupted stego images, where the stego images themselves are obtained using the proposed UMIS and UMISPB schemes as well as that by using the EMD and DE schemes. It is seen from this figure that the proposed UMISPB scheme provides the highest PSNR values at all the levels of the noise indicating that the attacker would be led to believe less that there is a secret data hidden in the stego image produced by the proposed scheme.

The bit error rate (BER) is the ratio of the total number of secret data bits extracted erroneously to the total number of embedded bits. A lower value of BER indicates that the receiver would be able to extract the secret data more successfully despite the fact that the stego image is corrupted by the salt and pepper noise. Figure 4.9 shows the BER values for the corrupted stego images, where the stego images themselves are obtained by using the proposed UMIS and UMISPB schemes as well as that by using the EMD and DE schemes. It is seen from this figure that the proposed UMISPB scheme provides the lowest BER values at all levels of the noise.

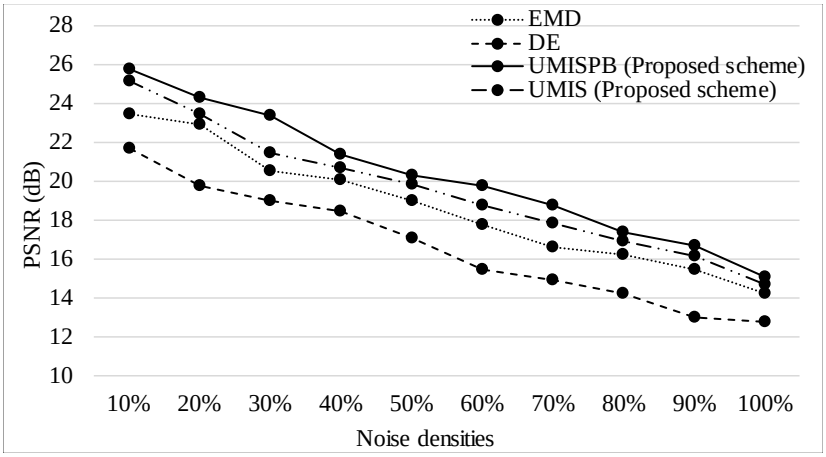


Figure 4.8: Comparison of the average PSNR for salt and pepper noisy images with noise densities ranging from 10% (0.1) to 100% (1.00).

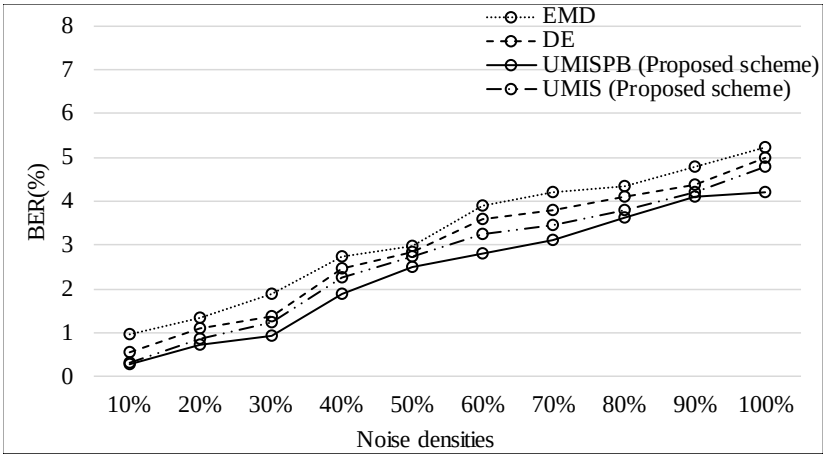


Figure 4.9: Comparison of the average BER for salt and pepper noisy images with noise densities ranging from 10% (0.1) to 100% (1.00).

4.3.1.3 Machine Learning-based Anti-detection Test

This blind steganalysis attack is based on using a classifier to make a decision [96]. The anti-detection test indicates that the steganographic scheme can defeat detection of hidden secret data in a way that makes suspicious stego images appear to be clean cover images. To classify stego and clean images into either suspicious or clean sets, a binary classifier is required. This requires a training set that contains statistical features for both cover and stego images. Statistical features can include histograms, contrast, mean, variance, skewness, entropy, and other statistical features. The binary classifier is trained using the extracted statistical features. After training, the classifier generates class labels using the statistical features of test images.

This anti-detection test comprises several steps. First, statistical features of images are extracted and stored in the dataset. Next, a support vector machine (SVM) binary classifier is created using supervised machine learning and optimized parameters for testing accuracy. Then, the classifier is trained on the statistical features in the defined dataset. Finally, the trained classifier is ready to predict the test images by labeling them as either cover or stego classes.

The proposed steganographic scheme is tested using the blind steganalysis attack. We generate 2000 grayscale stego images of size 512 $\times$ 512 using our scheme and used

1800 cover and stego images to train the SVM classification model. The remaining 200 images are used to test the model. To analyze the classification accuracy, we use a receiver operating characteristic (ROC) curve as a standard graphical representation to interpret the performance of the steganographic scheme. The ROC curve provides an indicator called the area under the curve (AUC), which is the total area under the curve. In digital image steganalysis, a larger AUC indicates that the steganalysis attack is strong and the steganographic scheme is vulnerable. Conversely, for the steganographic scheme, a smaller AUC and closer proximity to the diagonal indicates that it is more resilient against the attack [97]. Figure 4.10 shows the ROC performance curves for the SVM classifier for the proposed UMIS and UMISPB schemes as well as for the EMD and DE schemes, where a 100% capacity is concealed into cover images. It can be seen from this figure, the proposed UMISPB scheme has the smallest AUC and is closest to the diagonal, compared to the proposed UMIS scheme as well as the EMD and DE schemes. This indicates that our proposed UMISPB scheme has lower true positive rates (i.e., detection probabilities) because it does not disturb the statistical features of cover images, thereby avoiding detection using the SVM detector.

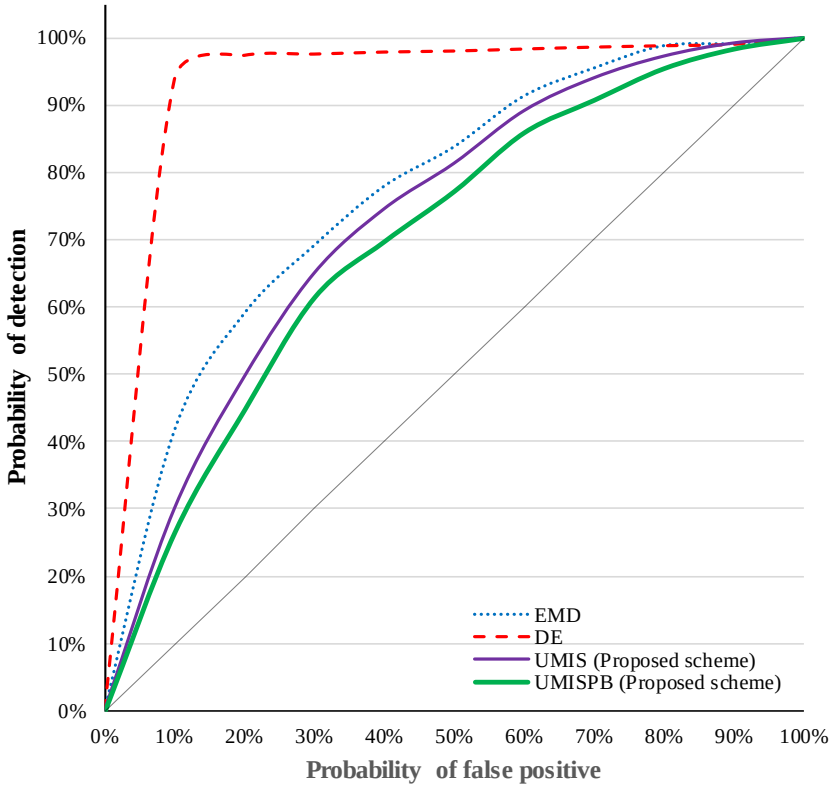


Figure 4.10: ROC curves of the EMD, DE and proposed UMIS, UMISPB schemes.

4.4 Summary

This chapter presents a novel unified modulus-based image steganographic with partitioned block scheme (UMISPB) that combines the EMD and DE schemes, aiming to achieve a high embedding rate and produce high-quality stego images simultaneously. The proposed UMISPB scheme involves dividing the secret data block into two parts and concealing them separately using the EMD and DE schemes, respectively. It has been shown that that dividing the secret data block into two parts, as equally as possible, and concealing them by using the EMD and DE schemes, respectively, lead to providing both

image quality and embedding rate, that are better than that achieved by concealing the entire secret data block using either the EMD or DE scheme alone. This observation has led to developing the proposed UMISPB scheme in which each block of secret data is hidden in three groups of pixels of the cover image. The first two groups are represented in the $(2n+1)$-ary notational system and the first part of the secret data is concealed in these two groups using the EMD scheme. The third group of pixels is represented in the $(2k^2+2k+1)$-ary notational system and the second part of the secret data is concealed in this group of pixels using the DE scheme.

Extensive experiments have been performed to obtain the embedding rates and PSNR values of stego images using the proposed UMISPB scheme, and the results obtained have been compared with that obtained using the proposed UMIS scheme (Chapter 3), the EMD and DE schemes, along with other recent state-of-the-art existing schemes. It has been shown that the proposed UMISPB scheme significantly outperforms the UMIS scheme and other recent schemes in terms of the quality of the stego image. However, it provides lower embedding rates than the proposed UMIS scheme. Nevertheless, the proposed UMISPB scheme demonstrates greater resilience compared to the UMIS, EMD, and DE schemes against all applied attacks. The high-performance and robustness of the proposed UMISPB scheme make it well-suited for a wide range of applications, such as secure military communications, confidential business communication, personal data protection, and other areas where secure communication and data transmission are critical.

Chapter 5

A Real-time Embedded System for the Steganographic Scheme UMIS and its Hardware Implementation

5.1 Introduction

Popular applications of image steganography encompass secure communication, image authentication [99], digital content distribution access control systems, data protection [100], secure mobile computing [101], banking systems, and IoT communication protocols [102]. Numerous steganographic techniques have been developed to maintain the quality of cover image while ensuring a high embedding rate [37, 39, 103, 104]. Image steganographic schemes can be classified into three main categories, interpolation-based [16, 19, 23], encryption-based [24, 30, 34, 35], and modulus-based schemes [37, 39, 40, 43–45, 47]. In the schemes of the first category, resolution of the cover image is scaled up to create extra neighboring pixels to conceal the secret data into the interpolated pixels and keep the original pixels unchanged. Bits of the secret data is concealed in the interpolated pixels by calculating the differences between the interpolated and original neighboring pixels, and the computational complexity depends on the interpolation technique used. Since, in order to maintain the quality of the cover image, a complex interpolation method is employed in these schemes, the embedding time of these schemes is large. In the encryption-based category, encryption cryptosystems are utilized to make the secret data more secure against attacks. However, in view of the use of cryptographic algorithms [53], [54], the computational complexity of these schemes is high. In the schemes of the third category, the embedding process is carried out using a modulus operation. Each block of the secret data is converted into a secret digit using a predefined $l-ary$ notational system and then embedded into a group of pixels of the cover image. These modulus-based steganographic schemes provide superior image qualities for the resulting stego images in comparison to that provided by the interpolation-based and encryption- based schemes. The modulus-based steganographic schemes have less computational complexity as the embedding and extraction processes employ simple modulus functions. The superior quality of the stego images in these schemes is achieved by limiting the changes in the pixel values of the cover image. With the same embedding rate, the image quality achieved by modulus-based schemes is higher than that achieved by the schemes in the other two categories.

Steganography can be implemented via software or hardware-based platforms. The

former offers design flexibility, but its computationally expensive. The latter offers high computational speed, architectural optimization capabilities and supports parallelism [91]. General-purpose CPUs were initially used, but they have the drawbacks low design portability and lack of pipelining optimization, even though they are faster than the software platform.

With the advancement of electronics, embedded digital image processing systems are becoming more prevalent [105]. ASIC, a hardware platform, delivers optimal performance as a dedicated platform, but has the disadvantages of complex design, high prototyping cost, and inflexibility during the developmental stage.

The development of Field-Programmable Gate Array (FPGA) offers a powerful platform for image processing algorithms, overcoming the limitations of CPUs and ASICs [106–108]. FPGA accelerates image processing through high-speed communication, the use of fast DSPs, parallelism, and pipelining [86]. Despite these capabilities, FPGA has drawbacks such as low flexibility, nonstandard interfaces, and limited scalability. To address these challenges, a solution called system-on-chip (SoC) has emerged, which combines multiple chips into a single platform, providing software flexibility and hardware performance [87]. AMD Xilinx adopted this platform that led to their so called All Programmable SoC (APSoC), Zynq-7000 APSoC, in 2011 [87].

The AMD Xilinx Zynq-7000 APSoC platform offers high-performance processing capabilities, the ability to support multiple interfaces and protocols, and consumes low power. The device's dual-core ARM Cortex-A9 processor, along with its programmable logic resources [109], provides the necessary processing power for hardware implementation. Additionally, it can support multiple interfaces and protocols, such as Ethernet, USB, and PCIe, which are necessary for seamless integration with other system components.

The results of recent FPGA hardware implementations have indicated that more the complexity of a steganographic scheme is, more are the resources that it consumes [5], [55]. The architectures in the existing hardware implementations schemes of [5, 55–61] do not take full advantage of the parallelism or pipelining, and thus results in high delays and low throughputs.

In this chapter, we present the hardware design [94] of the unified modulus-based image steganographic UMIS scheme, as summarized in Algorithm 1 of Section 3.2. Our aim is to design an efficient hardware architecture that can be effectively deployed on the versatile Zynq-7000 APSoC platform. To achieve high performance while optimizing resource utilization, the proposed design incorporates resource sharing, pipelining, and parallel processing techniques. The proposed design is developed and implemented based on the hardware-software co-design approach using the new AMD Xilinx Vivado 2020.1 design suite. By choosing a suitable number of pipelining stage, our design is capable of delivering high-throughput processing capabilities.

5.2 The Proposed Reconfigurable Embedded System of the Modulus-based Steganographic Scheme

The general schematic flow of the proposed reconfigurable Zynq-7000 APSoC for the steganographic scheme is represented in Figure 5.1. The general schematic has a top-bottom flow, where MATLAB is employed only for image representation and pixel-hex format conversion. The embedding process is performed as follows. (i) The two inputs to the Zynq-7000 APSoC platform are the cover image and the secret data. (ii) The pixels of the cover image are extracted and converted into a file of hex values using MATLAB along with the secret data. (iii) The reconfigurable Zynq-7000 embedded system receives the file containing pixel values in hex format and secret data in PS.

(iv) The locations of the hex values are permuted using a cipher key. The permutation module is implemented in the PS part of Zynq-7000 APSoC. (v) Once permutation is completed, the steganographic scheme starts to conceal the secret data into the permuted hex values of the pixels. (vi) The results, also in hex format, are permuted back to the original order using the decipher key. (vii) The hex values are converted into stego pixels using MATLAB to get the stego image.

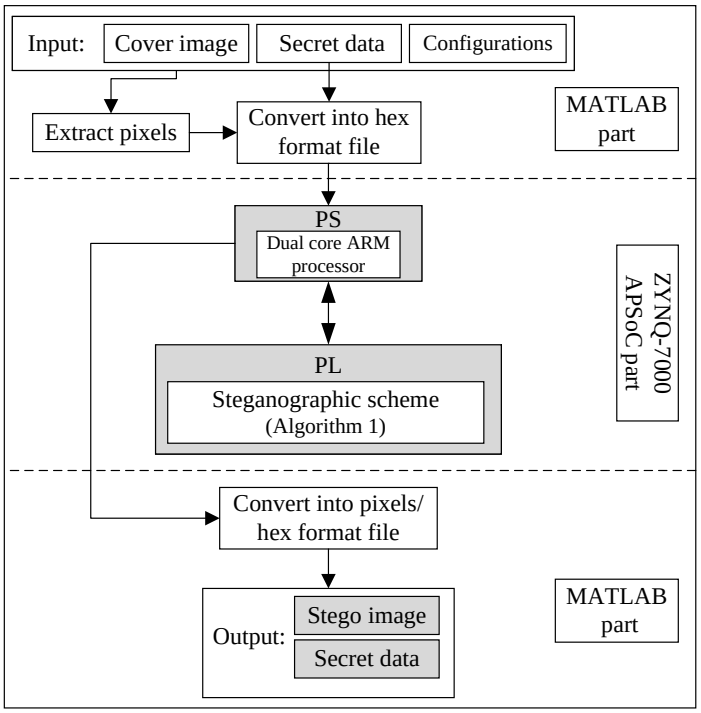


Figure 5.1: General schematic flow of the proposed reconfigurable Zynq-7000 APSoC (Algorithm 1).

The steganographic scheme summarized in Algorithm 1 is implemented in the PL part of Zynq-7000 APSoC. The implementation of the steganographic scheme has two functional modes, embedding and recovery. The embedding mode is for concealing secret data and generating the stego image, while the recovery mode is for extracting the concealed secret data from the stego image. In the embedding mode, concealing secret data is done as the hex values of the pixels are scanned. The modified (i.e., stego) permuted hex values are arranged back to the original order of hex values of the input cover image using a decipher key. The stego hex values are converted into stego pixels, and the stego image is generated using MATLAB. In the recovery mode, the secret data is extracted after permuting the hex values of the stego pixels using the same cipher key. Secret data gets extracted using the extraction functions in the way they got concealed during the embedding mode.

5.2.1 Overall System Architecture

The image steganographic system includes the steganographic module, storage module, I/O peripherals, and communication module. The steganographic module contains the AXI4 implementation IP core of the efficient steganographic scheme. The storage module consists of using the DDR3 RAM and SD card. The communication module includes the UART, the AXI4 interconnect, and GP AXI4 interfaces between PS and PL. The block diagram of the image steganographic system is shown in Figure 5.2.

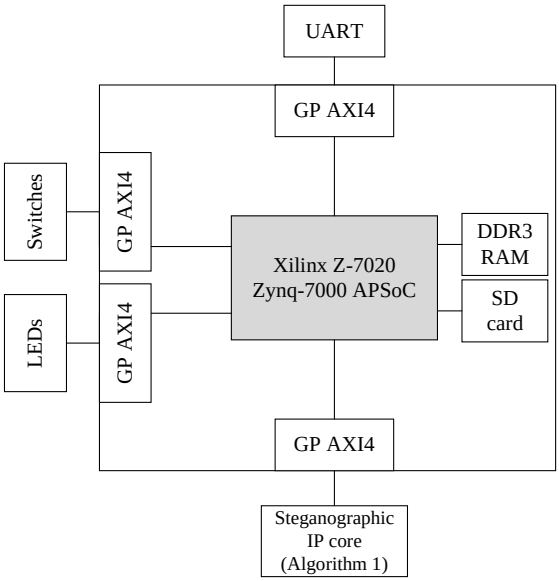


Figure 5.2: Block diagram of the image steganographic system (Algorithm 1).

The UART is another slave AXI4-Lite IP core used to facilitate a communication between Zynq-7000 APSoC and the host (PC) for debugging using a terminal application. The switches and LEDs are I/O slave AXI4-Lite IP cores and they are utilized for controlling and getting feedback from PS and PL. We develop the steganographic IP core with an architecture as shown in Figure 5.3.

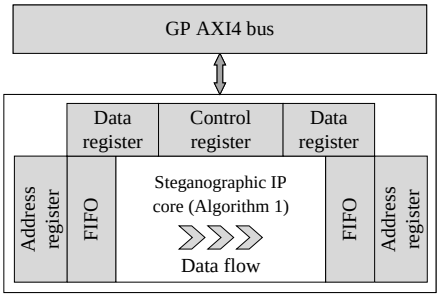


Figure 5.3: Block diagram of the steganographic IP core (Algorithm 1).

The GP AXI4 in PS is the master for this core. The steganographic IP core is interfaced with PS using the AXI4-Lite bus as a slave. The attached data registers can pass data by using the AXI4-Lite bus. These registers are software accessible in such a way that PS can send and read parameters to the IP core using a written C++ code. These registers can be classified as data registers, control registers, and address registers. Control registers are used for initializing the IP core. Data registers are used to send data for processing in the IP core. Address registers are used for buffering data in the FIFOs of the IP core. The steganographic IP core is adaptable to any image resolution by having generic parameters such as image size (width $\times$ height) and FIFOs depth. This feature of the steganographic IP core is essential to meet the requirements of various applications.

5.2.2 Steganographic Scheme Hardware Architecture

The hardware architecture of the steganographic scheme is represented in Figure 5.4. A block of data is read from the DDR3 RAM as input to the IP core. Each block contains

hex values of the pixels and secret data. The block is fed to an input FIFO, and the embedding process starts the moment the FIFO gets full. An interrupt signal is asserted, indicating the IP core is busy now and no more reads from the DDR3 RAM. A shift register reads from the input FIFO, and the process of either embedding (concealing) or recovery (extracting) secret data is started based on the Mode input signal.

The hardware implementation of the proposed steganographic scheme is carried out by implementing the two functional processes of embedding and recovery, as well as the lookup tables T_1 and T_2. The weighted sum modulo functions of the EMD and DE schemes are implemented in the FPGA using Lookup Tables (LUTs). The extraction functions of the EMD and DE schemes are also implemented using LUTs. The lookup tables T_1 and T_2 are implemented in the FPGA as LUT-based storages. Figure 5.4 shows the components for both embedding (Embed S_0, Embed S_1, Embed S_2, Embed S_3) and recovery (Recover S_0, Recover S_1, Recover S_2, Recover S_3). Each component represents the weighted sum modulo function for the corresponding scheme, with Embed S_0 and Embed S_2 representing the logic for the weighted sum modulo function of the DE scheme, and Embed S_1 and Embed S_3 representing the logic for the weighted sum modulo function of the EMD scheme.

Figure 5.4 shows the two paths of embedding and recovery processes. Data path of the embedding process is started by receiving both of the cover pixels (Q_0, Q_1, Q_2, Q_3) and secret data block B. The output of the lookup table T_2 is the coordinate values (t_{2_x}, t_{2_y}) of B and output of the lookup table T_1 is the four secret digits (S_0, S_1, S_2, S_3). These four secret digits are concealed into the cover pixels (Q_0, Q_1, Q_2, Q_3) by applying the embedding functions of EMD and DE schemes to generate the stego pixels (Q'_0, Q'_1, Q'_2, Q'_3). The data path of the recovery process is started by receiving the stego pixels (Q'_0, Q'_1, Q'_2, Q'_3). Secret digits (S_0, S_1, S_2, S_3) are extracted by applying the extracting functions of EMD and DE schemes. These secret digits are forwarded to the lookup table T_1 to get the coordinate values (t_{1_x}, t_{1_y}). These coordinate values are forwarded to the lookup table T_2 to locate the secret data block B.

In the embedding process, the hex values of the pixels from the cover image are arranged into four groups (Q_0, Q_1, Q_2, Q_3) for concealing four digits (S_0, S_1, S_2, S_3) that representing the secret data block. A stego pixel is generated for each group and stored in another shift register. Once all the secret digits are concealed, the output FIFO starts to receive the completed block of hex values that represent the stego pixels. These blocks are written back to the DDR3 RAM once the output FIFO is full. Once all the hex values of pixels from the cover image are scanned, the data in DDR3 RAM is saved in a text file and written in the SD card. MATLAB is utilized to convert the text file into pixels of the stego image. In the recovery process, the hex values of pixels from the stego image are arranged and the secret data is extracted and written back to the shift register. The output FIFO receives the extracted secret data and writes it back to the DDR3 RAM. The data in DDR3 RAM is saved in a text file and written in the SD card. MATLAB is used to convert the text file into a suitable format for the recovered secret data.

In our implementation, reading and writing operations are performed on a 64-bit data path that conforms with the size of 64-bit used in the registers, FIFOs, and RAM units. The FIFOs store 16 blocks of data, each with a width of 64 bits. The hardware implementation requires only two BRAMs for the FIFOs, one for inputting and the other for outputting the data. All the operations of the embedding and recovery processes are performed using LUTs and registers located in the configurable logic block (CLB) units of the FPGA device.

To improve the performance of our architecture, we use sub-pipelining to reduce the delay of the critical path. This technique is effective for architectures with an iterative

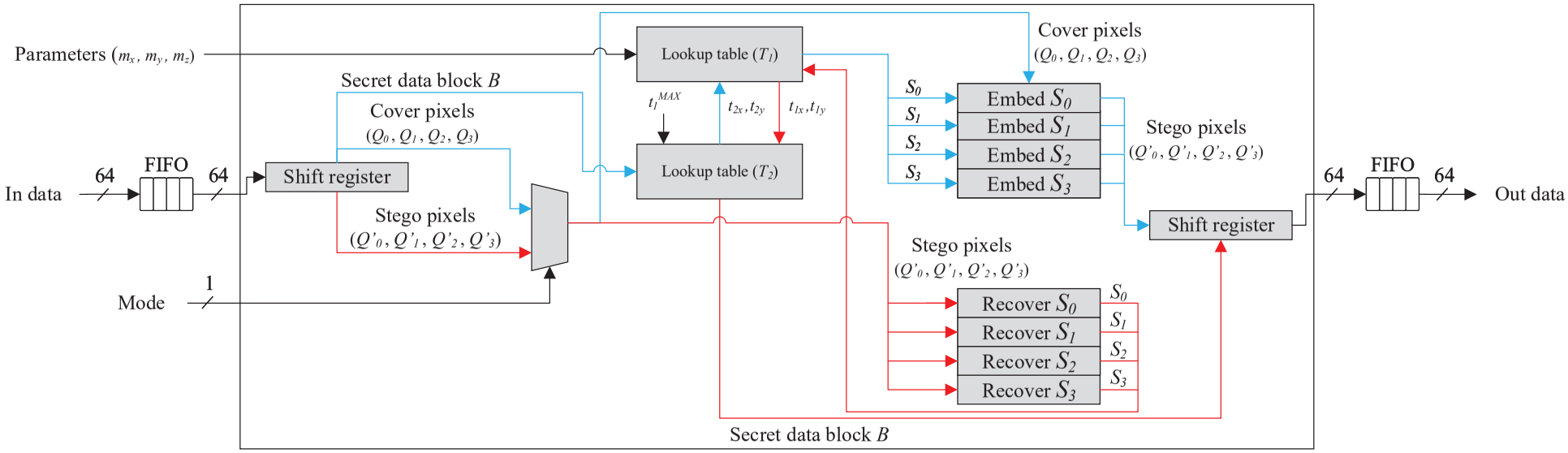


Figure 5.4: Hardware architecture of the steganographic scheme (Algorithm 1). Note: Connections in blue exist in embedding, connections in red exist only during recovery, and connections in black exist during both embedding and recovery.

behavior. By inserting an optimal number of sub-pipelining stages, we can achieve an efficient balance between the speed and area. Figure 5.5 shows the trade-off between delay and area when 1 to 4 sub-pipelining stages are inserted in our architecture. The best trade-off is achieved with two stages. The first stage is inserted after the lookup tables T_1 and T_2, and the second stage is inserted after the weighted sum modulo functions have been carried out. To further optimize the timing performance, register balancing (retiming) strategy is applied to the architecture.

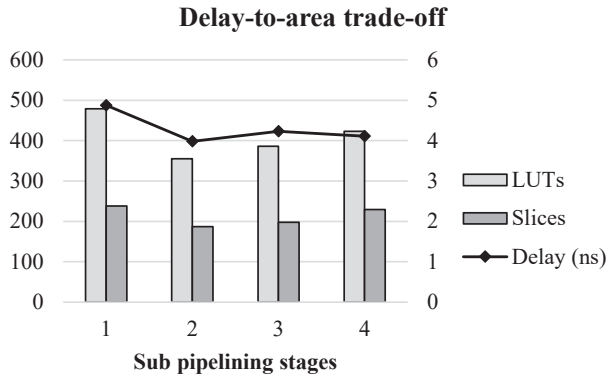


Figure 5.5: Delay-to-area (slices, LUTs) trade-off using different number of sub pipelining stages.

5.2.3 Finite State Machine of the Steganographic Scheme

An efficient finite state machine (FSM) model is employed in an iterative manner to achieve a compact design by sharing and reusing resources of the FPGA device for both the embedding and recovery processes. The FSM model comprises a specific number of states, each of which computes an output based on its input and forwards it to the next state. The FSM of the steganographic scheme's embedding and recovery processes is illustrated in Figure 5.6. The process of the embedding or recovery is initiated when a valid input signal is asserted by the input FIFO, indicating that the input data block is ready to be processed.

At the Start state, the lookup tables are constructed based on the parameters m_x, m_y and m_z. When the Mode signal is 1, the state Embedding process starts by concealing the corresponding four secret digits (S_0, S_1, S_2, S_3) that represent the secret data block in table T_2, and the pixels in the cover image get updated to generate the stego pixels in the state Update pixels process. When the Mode single is 0, the state Recovery process begins to extract the secret digits from the stego pixels. At the Done state, A done signal is asserted when the process of embedding or recovery has done the concealing or extracting the secret digits, respectively. Finally, the output data block is sent to the output FIFO once the valid output signal is asserted. Applying the FSM in the architecture of the steganographic scheme leads to have a latency (delay). Latency refers to the number of clock cycles required to process a data block from the input of a module to the output. The proposed architecture has a latency of 10 clock cycles. The total number of clock cycles becomes 15, if the AXI4-Lite bus read and write transactions (PS-PL communication) are included, where 2 and 3 clock cycles are required for the read and write transactions, respectively.

The proposed architecture also employs a parallelism strategy for both the embedding and recovery processes. In the embedding process, four secret digits are concealed in parallel, without any dependencies. Similarly, in the recovery process, four

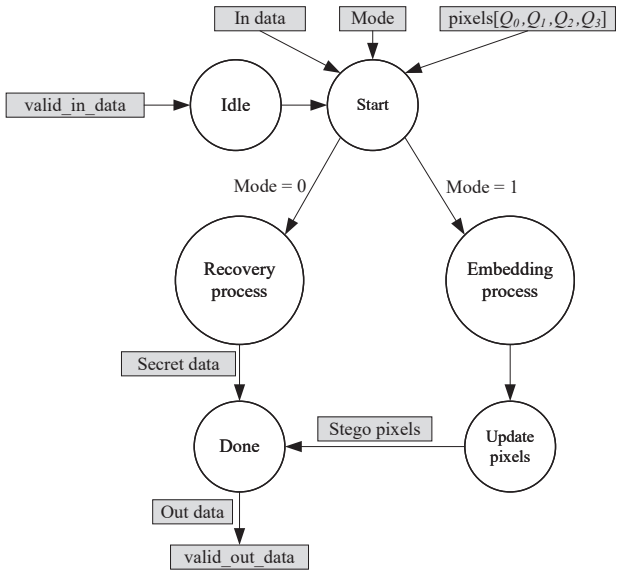


Figure 5.6: FSM of the embedding and recovery processes in the proposed steganographic scheme (Algorithm 1).

secret digits are extracted in parallel. The applied parallelism optimizes the critical data paths by minimizing the latency in terms of clock cycles.

5.2.4 Place and Route Implementation Results

Development of the steganographic IP core has three phases: design, implementation, and integration. In the design phase, RTL Verilog designs are created to implement an efficient steganographic scheme using an FPGA device. Functional and timing simulations for the Verilog designs are performed using the AMD Xilinx ISim simulator to ensure the proper functioning of the designs.

In the implementation phase, the functional designs are synthesized and implemented (placed and routed) on the FPGA device to generate reports on resource utilization, operating speed, and power consumption. To improve the synthesize and implementation processes, strategies such as register balancing, resource sharing, and combinational logic optimization are applied using the Vivado tool. A timing constraint is applied to all results to achieve the optimal balance between area and speed, with zero-timing errors. Table 5.1 shows the results of the place and route implementation of our design, compared to other recent FPGA implementations. It is seen from this table that our design achieves the highest performance in terms of speed, resource utilization, throughput, and power consumption. The implemented design provides high operating speed and high throughput due to the pipelining applied in the hardware architecture for the steganographic IP core. The AMD Xilinx power estimator (XPE) tool is used to perform a power analysis and estimate the power consumption of the IP core. The on-chip total power consumption is calculated by measuring the power consumption values of the system clock, logic blocks, and BRAMs in the FPGA device. It is worth noting that the values of the power consumption given in Table 5.1 represent the power consumed by the PL part (FPGA device) only. The implemented design has a small footprint, making it well-suited for low-powered, resource-constrained embedded systems such as wearable smart devices, radio frequency identification (RFID) cards, and network-attached storage devices. The place and route results indicate that our design is highly efficient in terms of the operating speed to LUT (MHz/LUT) ratio,

Table 5.1: Comparison of the implemented scheme with other implementations (place and route results).

Scheme	FPGA device	Slices	FFs	LUTs	BRAMs	Freq. (MHz)	Throughput (Mbps)	Efficiency (Freq./LUT)	Power (mW)*
IWT-based [5] (2014)	Cyclone II	-	7412	11222	-	50.00	-	0.004	-
Multilevel LSB [56] (2017)	Virtex- II Pro	-	-	-	-	**144.30**	144.3	-	-
EMD-based [55] (2019)	Artix-7	**296**	1890	1419	-	144.00	143.9	0.101	3807
Lifting-based [57] (2019)	Artix-7	476	555	350	128	130.14	**188.8**	**0.539**	**78.45**
Parallelism-based [58] (2021)	Virtex-6	-	-	-	-	118.66	-	-	-
Threshold-based [59] (2021)	Spartan-6	-	670	627	**7**	-	-	-	200
LSB matching [60] (2021)	Spartan-3A	-	647	668	-	-	-	-	369.3
Chaotic-based [61] (2022)	Cyclone IV	1100	**537**	1067	-	-	-	-	-
Our design	Artix-7	**187**	**629**	**355**	**2**	**250.54**	**1600**	**0.707**	**167**

* The values given are the powers consumed by only the PL parts of the various hardware implementations. The red color represents the 1^{st} best result and the green color represents the 2^{nd} best result.

with a value of 0.71. This ratio surpasses the highest value achieved by the fastest recent modulus-based hardware implementation, as reported in [55], which has the value of 0.10 for this ratio. Further, our hardware implementation achieves a higher throughput to LUT ($Mbps/LUT$) ratio, with a value of 4.51 as compared to the ratio 0.54 as reported in [57].

In the integration phase, the implemented design is packaged as an IP core using the AMD Xilinx IP package integrator. This tool is used to rapidly connect the design, implemented on PL, to PS using the AXI4 interface. The overall diagram of the image steganographic system using the Zynq-7000 APSoC is shown in Figure 5.7. It can be seen from the diagram that four IP cores are connected to PS in the Zynq-7000 APSoC. The steganographic IP core is considered a custom core and can be modified to add more features and support more functions. The UART and I/O IP cores, on the other hand, are produced by AMD Xilinx.

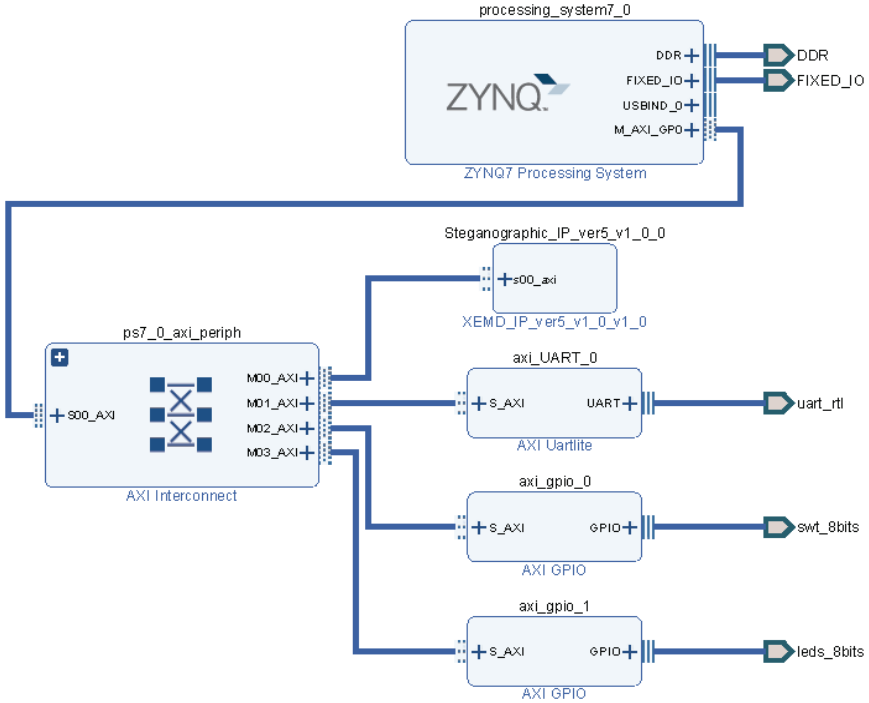


Figure 5.7: Overall diagram of the image steganographic system (Algorithm 1) using Zynq-7000 APSoC.

In an effort to achieve a real-time performance, all modulo arithmetic operations required for the embedding process are performed in real-time using the dedicated IP steganographic core that is optimized for this purpose. The output stego image is generated in real-time as the IP core receives the input cover image and the secret data. Also, a large set of cover images are pre-stored in the high-speed memory, DDR3 RAM, of the AMD Xilinx Zynq-7000 APSoC platform. This avoids any waiting time by the IP core to process the next cover image. In the PS part of the Zynq-7000 APSoC, two buffers are employed, one for reading the cover images and secret data, and the other to write the stego images to the DDR3 RAM. These buffers are accessed by the IP steganographic core for reading and writing data, which accelerates the embedding and recovery processes by allowing for cyclical processing of images. Also, the IP core is designed to perform parallel processing, which further improves the speed of image processing. Overall, these strategies allow for a high-speed, real-time implementation of the steganographic system.

5.3 Summary

This chapter presents the hardware implementation of the novel modulus-based image steganographic scheme. In particular, we focus on implementing the proposed scheme introduced in Chapter 3, which does not require dividing secret data blocks. The hardware implementation is carried out on the AMD Xilinx Zynq-7000 APSoC platform, which combines hardware and software design. An IP core is developed to incorporate the efficient steganographic scheme, utilizing minimal resources, low power, high speed, and high throughput. The implemented steganographic scheme exhibits high operating speeds, high throughput, and low power consumption.

Chapter 6

A Real-time Embedded System for the Steganographic Scheme UMISPB and its Hardware Implementation

6.1 Introduction

In this chapter, we present the hardware design of the unified modulus-based image steganographic with partitioned secret data blocks UMISPB scheme [98], as summarized in Algorithm 2 of Section 4.2. Similar to Chapter 5, our goal is to develop an efficient hardware architecture suitable for deployment on the versatile Zynq-7000 APSoC platform. To achieve high performance while optimizing resource utilization, we employ resource sharing, pipelining, and parallel processing techniques. The proposed design is implemented using the AMD Xilinx Vivado 2020.1 design suite, following the hardware-software co-design approach. By suitably choosing the number of pipelining stages, our proposed design achieves high-throughput processing capabilities.

6.2 The Proposed Reconfigurable Embedded System of the Modulus-based Steganographic Scheme

Figure 6.1 illustrates the overall schematic flow of the proposed reconfigurable Zynq-7000 APSoC for the proposed steganographic scheme in Section 4.2. The schematic follows a top-bottom flow, where MATLAB is utilized for image representation and pixel-hex format conversion. The embedding process is as follows. (i) The Zynq-7000 APSoC platform receives two inputs: the cover image and the secret data. (ii) MATLAB extracts the pixels from the cover image and converts them into a hex value file, along with the secret data. (iii) The reconfigurable Zynq-7000 embedded system receives the file containing hex values of pixels and secret data in the PS (Processing System). (iv) The locations of the hex values are permuted using a cipher key. The permutation module is implemented in the PS part of Zynq-7000 APSoC. (v) After permutation, the steganographic scheme conceals the secret data into the permuted hex values of the pixels. (vi) The resulting hex values, in permuted order, are reversed back to the original order using the decipher key. (vii) Finally, MATLAB converts the hex values back into stego pixels, resulting in the stego image.

The steganographic scheme summarized in Algorithm 2 is implemented in the PL part of Zynq-7000 APSoC. The implementation of the steganographic scheme has two

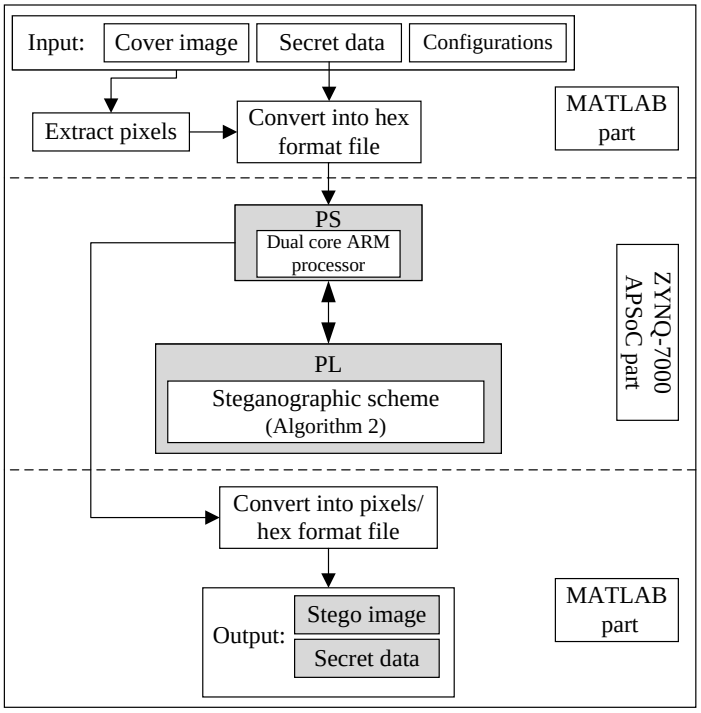


Figure 6.1: General schematic flow of the proposed reconfigurable Zynq-7000 APSoC (Algorithm 2).

functional modes, embedding and recovery. The embedding mode is for concealing secret data and generating the stego image, while the recovery mode is for extracting the concealed secret data from the stego image. In the embedding mode, concealing secret data is done as the hex values of the pixels are scanned. The modified (i.e., stego) permuted hex values are arranged back to the original order of hex values of the input cover image using a decipher key. The stego hex values are converted into stego pixels, and the stego image is generated using MATLAB. In the recovery mode, the secret data is extracted after permuting the hex values of the stego pixels using the same cipher key. Secret data gets extracted using the extraction functions in the way they got concealed during the embedding mode.

6.2.1 Overall System Architecture

The image steganographic system includes the steganographic module, storage module, I/O peripherals, and communication module. The steganographic module contains the AXI4 implementation IP core of the efficient steganographic scheme. The storage module consists of using the DDR3 RAM and SD card. The communication module includes the UART, the AXI4 interconnect, and GP AXI4 interfaces between PS and PL. The block diagram of the image steganographic system is shown in Figure 6.2.

The UART is another slave AXI4-Lite IP core used to facilitate a communication between Zynq-7000 APSoC and the host (PC) for debugging using a terminal application. The switches and LEDs are I/O slave AXI4-Lite IP cores and they are utilized for controlling and getting feedback from PS and PL. We develop the steganographic IP core with an architecture as shown in Figure 6.3.

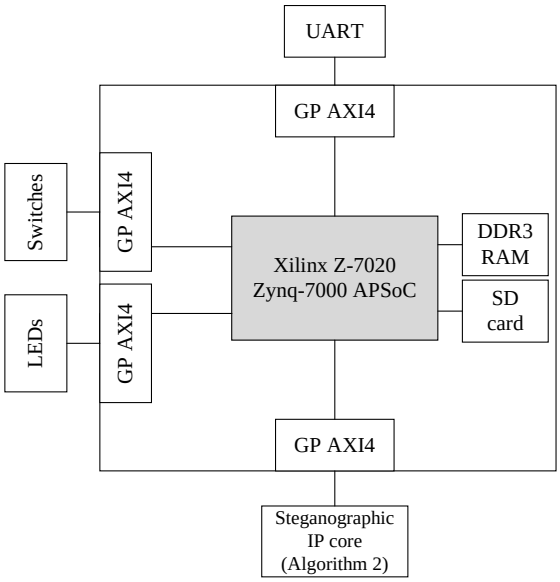


Figure 6.2: Block diagram of the image steganographic system (Algorithm 2).

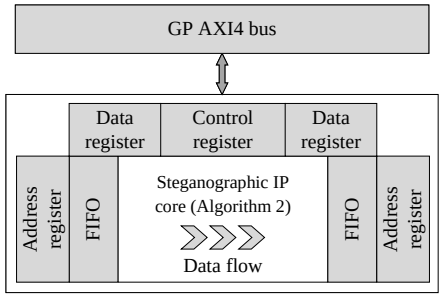


Figure 6.3: Block diagram of the steganographic IP core (Algorithm 2).

The GP AXI4 in PS is the master for this core. The steganographic IP core is interfaced with PS using the AXI4-Lite bus as a slave. The attached data registers can pass data by using the AXI4-Lite bus. These registers are software accessible in such a way that PS can send and read parameters to the IP core using a written C++ code. These registers can be classified as data registers, control registers, and address registers. Control registers are used for initializing the IP core. Data registers are used to send data for processing in the IP core. Address registers are used for buffering data in the FIFOs of the IP core. The steganographic IP core is adaptable to any image resolution by having generic parameters such as image size (width $\times$ height) and FIFOs depth. This feature of the steganographic IP core is essential to meet the requirements of various applications.

6.2.2 Steganographic Scheme Hardware Architecture

The hardware architecture of the steganographic scheme is represented in Figure 6.4. A block of data is read from the DDR3 RAM as input to the IP core. Each block contains hex values of the pixels and secret data. The block is fed to an input FIFO, and the embedding process starts the moment the FIFO gets full. An interrupt signal is asserted, indicating the IP core is busy now and no more reads from the DDR3 RAM. A shift register reads from the input FIFO, and the process of either embedding (concealing) or recovery (extracting) secret data is started based on the Mode input signal.

The hardware implementation of the proposed steganographic scheme is carried out

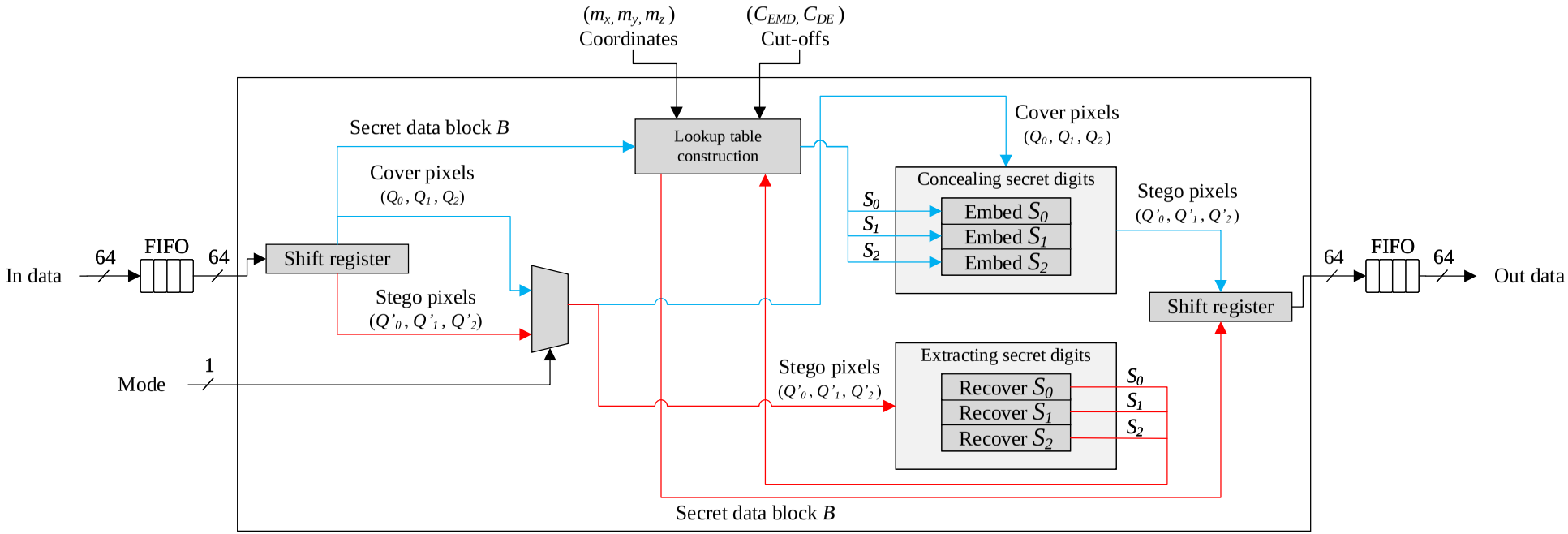


Figure 6.4: Hardware architecture of the steganographic scheme (Algorithm 2). Note: Connections in blue exist in embedding, connections in red exist only during recovery, and connections in black exist during both embedding and recovery.

by implementing the two functional processes of embedding and recovery, as well as the lookup table. The weighted sum modulo functions of the EMD and DE schemes are implemented in the FPGA using Lookup Tables (LUTs). The extraction functions of the EMD and DE schemes are also implemented using LUTs. The lookup table is implemented in the FPGA as LUT-based storages. Figure 6.4 shows the components for both embedding (Embed S_0, Embed S_1, Embed S_2) and recovery (Recover S_0, Recover S_1, Recover S_2). Each component represents the weighted sum modulo function for the corresponding scheme, with Embed S_0 and Embed S_1 representing the logic for the weighted sum modulo function of the EMD scheme, and Embed S_2 representing the logic for the weighted sum modulo function of the DE scheme.

Figure 6.4 shows the two paths of embedding and recovery processes. Data path of the embedding process is started by receiving both of the cover pixels (Q_0, Q_1, Q_2) and secret data block of size L. The secret data block of size L is divided into parts L_1 and L_2, where coordinate values of the L_1 is x, and y while the L_2 is z. The output of the lookup table is the coordinate values: $x \rightarrow S_0$, $y \rightarrow S_1$, and $z \rightarrow S_2$. These three secret digits are concealed into the cover pixels (Q_0, Q_1, Q_2) by applying the embedding functions of the EMD and DE schemes to generate the stego pixels (Q'_0, Q'_1, Q'_2). The data path of the recovery process is started by receiving the stego pixels (Q'_0, Q'_1, Q'_2). Secret digits (S_0, S_1, S_2) are extracted by applying the extracting functions of the EMD and DE schemes. These secret digits represents the two parts (L_1, L_2) of the secret data block of size L in the the lookup table.

In the embedding process, the hex values of the pixels from the cover image are arranged into three groups (Q_0, Q_1, Q_2) for concealing three digits (S_0, S_1, S_2) that representing the secret data block of size L. A stego pixel is generated for each group and stored in another shift register. Once all the secret digits are concealed, the output FIFO starts to receive the completed block of hex values that represent the stego pixels. These blocks are written back to the DDR3 RAM once the output FIFO is full. Once all the hex values of pixels from the cover image are scanned, the data in DDR3 RAM is saved in a text file and written in the SD card. MATLAB is utilized to convert the text file into pixels of the stego image. In the recovery process, the hex values of pixels from the stego image are arranged and the secret data is extracted and written back to the shift register. The output FIFO receives the extracted secret data and writes it back to the DDR3 RAM. The data in DDR3 RAM is saved in a text file and written in the SD card. MATLAB is used to convert the text file into a suitable format for the recovered secret data.

In our implementation, reading and writing operations are performed on a 64-bit data path that conforms with the size of 64-bit used in the registers, FIFOs, and RAM units. The FIFOs store 16 blocks of data, each with a width of 64 bits. The hardware implementation requires only two BRAMs for the FIFOs, one for inputting and the other for outputting the data. All the operations of the embedding and recovery processes are performed using LUTs and registers located in the configurable logic block (CLB) units of the FPGA device.

To improve the performance of our architecture, we use sub-pipelining to reduce the delay of the critical path. This technique is effective for architectures with an iterative behavior. By inserting an optimal number of sub-pipelining stages, we can achieve an efficient balance between the speed and area. Figure 6.5 shows the trade-off between delay and area when 1 to 4 sub-pipelining stages are inserted in our architecture. The best trade-off is achieved with two stages. The first stage is inserted after the lookup table, and the second stage is inserted after the weighted sum modulo functions have been carried out. To further optimize the timing performance, register balancing (retiming) strategy is applied to the architecture.

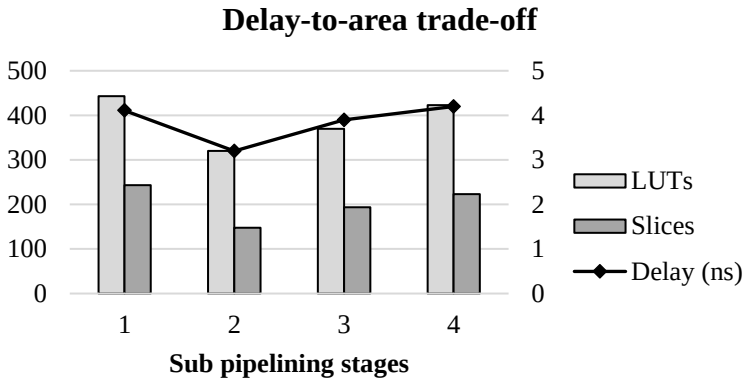


Figure 6.5: Delay-to-area (slices, LUTs) trade-off using different number of sub pipelining stages.

6.2.3 Finite State Machine of the Steganographic Scheme

An efficient finite state machine (FSM) model is employed in an iterative manner to achieve a compact design by sharing and reusing resources of the FPGA device for both the embedding and recovery processes. The FSM model comprises a specific number of states, each of which computes an output based on its input and forwards it to the next state. The FSM of the steganographic scheme's embedding and recovery processes is illustrated in Figure 6.6. The process of the embedding or recovery is initiated when a valid input signal is asserted by the input FIFO, indicating that the input data block is ready to be processed.

At the Idle state, the steganographic IP core does not perform any concealing or extracting operations, and it waits the assertion for the start signal. When start is 1, it transits to the Start state. At the Start state, the $x - y$ and z lookup tables are constructed based on the suitable parameters m_x, m_y, m_z, and the cut-offs (C_{EMD}, C_{DE}). The secret data block of size L, Mode, and the group of pixels (Q_0, Q_1, Q_2) are fetched from the shift register, and the IP core is waiting for the valid signal assertion before starting concealing or extracting secret data. When the Mode signal is 1, the state Embedding process starts by concealing the corresponding three secret digits (S_0, S_1, S_2) that represent the secret data block of size L, and the pixels in the cover image get updated to generate the stego pixels in the state Update pixels process. When the Mode single is 0, the state Recovery process begins to extract the secret digits from the stego pixels. At the Done state, A done signal is asserted when the process of embedding or recovery has done the concealing or extracting the secret digits, respectively. Finally, the output data block is sent to the output FIFO once the valid output signal is asserted.

Applying the FSM in the architecture of the steganographic scheme leads to have a latency (delay). Latency refers to the number of clock cycles required to process a data block from the input of a module to the output. The proposed architecture has a latency of 8 clock cycles. The total number of clock cycles becomes 13, if the AXI4-Lite bus read and write transactions (PS-PL communication) are included, where 2 and 3 clock cycles are required for the read and write transactions, respectively.

The proposed architecture also employs a parallelism strategy for both the embedding and recovery processes. In the embedding process, three secret digits are concealed in parallel, without any dependencies. Similarly, in the recovery process, three secret digits are extracted in parallel. The applied parallelism optimizes the critical data paths by minimizing the latency in terms of clock cycles.

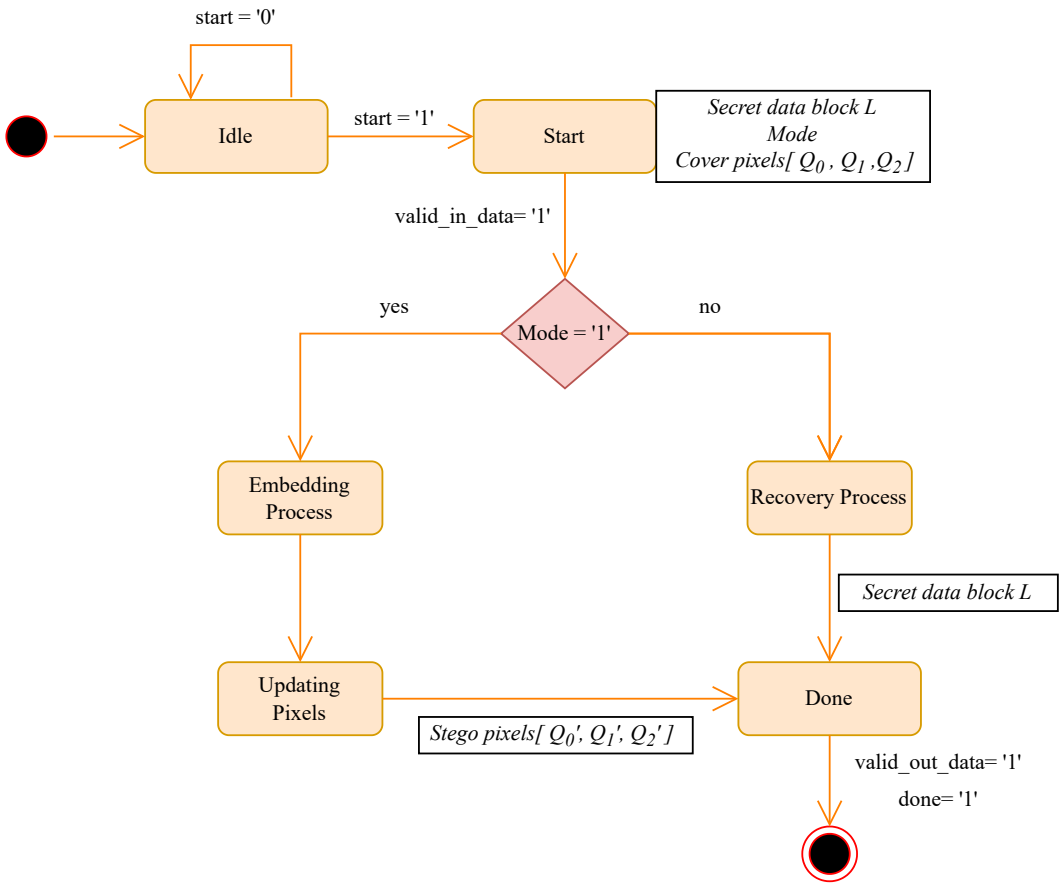


Figure 6.6: FSM of the embedding and recovery processes in the steganographic scheme Algorithm 2).

6.2.4 Place and Route Implementation Results

In the implementation phase, the functional designs are synthesized and implemented (placed and routed) on the FPGA device to generate reports on resource utilization, operating speed, and power consumption. To improve the synthesize and implementation processes, strategies such as register balancing, resource sharing, and combinational logic optimization are applied using the Vivado tool. A timing constraint is applied to all results to achieve the optimal balance between area and speed, with zero-timing errors.

Table 6.1 shows the results of the place and route implementation of the UMISPB design, compared to the UMIS design (Chapter 5) as well as other recent FPGA steganographic implementations. It is seen from this table that the hardware designs of the UMIS and UMISPB schemes achieve the highest performance in terms of speed, resource utilization, throughput, and power consumption compared to other hardware designs. Moreover, it is seen that the hardware design of the UMISPB scheme achieves better performance compared to that provided by the hardware design of the UMIS scheme. The implemented design provides high operating speed and high throughput due to the pipelining applied in the hardware architecture for the steganographic IP core. The AMD Xilinx power estimator (XPE) tool is used to perform a power analysis and estimate the power consumption of the IP core. The on-chip total power consumption is calculated by measuring the power consumption values of the system clock, logic blocks, and BRAMs in the FPGA device. It is worth noting that the values of the power consumption given in Table 6.1 represent the power consumed by the PL part

Table 6.1: Comparison of the implemented scheme with other implementations (place and route results).

Scheme	FPGA device	Slices	FFs	LUTs	BRAMs	Freq. (MHz)	Throughput (Mbps)	Efficiency (Freq./LUT)	Power (mW)*
IWT-based [5] (2014)	Cyclone II	-	7412	11222	-	50.00	-	0.004	-
Multilevel LSB [56] (2017)	Virtex- II Pro	-	-	-	-	144.30	144.3	-	-
EMD-based [55] (2019)	Artix-7	296	1890	1419	-	144.00	143.9	0.101	3807
Lifting-based [57] (2019)	Artix-7	476	555	350	128	130.14	188.8	0.539	**78.45**
Parallelism-based [58] (2021)	Virtex-6	-	-	-	-	118.66	-	-	-
Threshold-based [59] (2021)	Spartan-6	-	670	627	**7**	-	-	-	200
LSB matching [60] (2021)	Spartan-3A	-	647	668	-	-	-	-	369.3
Chaotic-based [61] (2022)	Cyclone IV	1100	537	1067	-	-	-	-	-
Our design (UMIS)	Artix-7	**187**	**629**	355	**2**	**250.54**	**1600**	**0.707**	167
Our design (UMISPB)	Artix-7	**148**	**523**	**320**	**2**	**289.7**	**2318**	**0.905**	**152**

* The values given are the powers consumed by only the PL parts of the various hardware implementations. The red color represents the 1^{st} best result and the green color represents the 2^{nd} best result.

(FPGA device) only. The implemented design has a small footprint, making it well-suited for low-powered, resource-constrained embedded systems such as wearable smart devices, radio frequency identification (RFID) cards, and network-attached storage devices. The place and route results indicate that our design is highly efficient in terms of the operating speed to LUT (MHz/LUT) ratio, with a value of 0.905. This ratio surpasses the highest value achieved by the fastest recent modulus-based hardware implementation, as reported in [55], which has the value of 0.10 for this ratio. Further, our hardware implementation achieves a higher throughput to LUT ($Mbps/LUT$) ratio, with a value of 7.24 as compared to the ratio 0.54 as reported in [57].

In the integration phase, the implemented design is packaged as an IP core using the AMD Xilinx IP package integrator. This tool is used to rapidly connect the design, implemented on PL, to PS using the AXI4 interface. The overall diagram of the image steganographic system using the Zynq-7000 APSoC is shown in Figure 6.7. As illustrated in the diagram, four IP cores are connected to PS in the Zynq-7000 APSoC. The steganographic IP core is considered a custom core and can be modified to add more features and support more functions. The UART and I/O IP cores, on the other hand, are produced by AMD Xilinx.

In an effort to achieve a real-time performance, all modulo arithmetic operations required for the embedding process are performed in real-time using the dedicated IP steganographic core that is optimized for this purpose. The output stego image is generated in real-time as the IP core receives the input cover image and the secret data. Also, a large set of cover images are pre-stored in the high-speed memory, DDR3 RAM, of the AMD Xilinx Zynq-7000 APSoC platform. This avoids any waiting time by the IP core to process the next cover image. In the PS part of the Zynq-7000 APSoC, two buffers are employed, one for reading the cover images and secret data, and the other to write the stego images to the DDR3 RAM. These buffers are accessed by the IP steganographic core for reading and writing data, which accelerates the embedding and recovery processes by allowing for cyclical processing of images. Also, the IP core is designed to perform parallel processing, which further improves the speed of image processing. Collectively, these strategies contribute to a high-speed, real-time implementation of the steganographic system, surpassing the performance of the hardware implementation in chapter 5 in terms of speed, resource consumption, throughput, and power consumption.

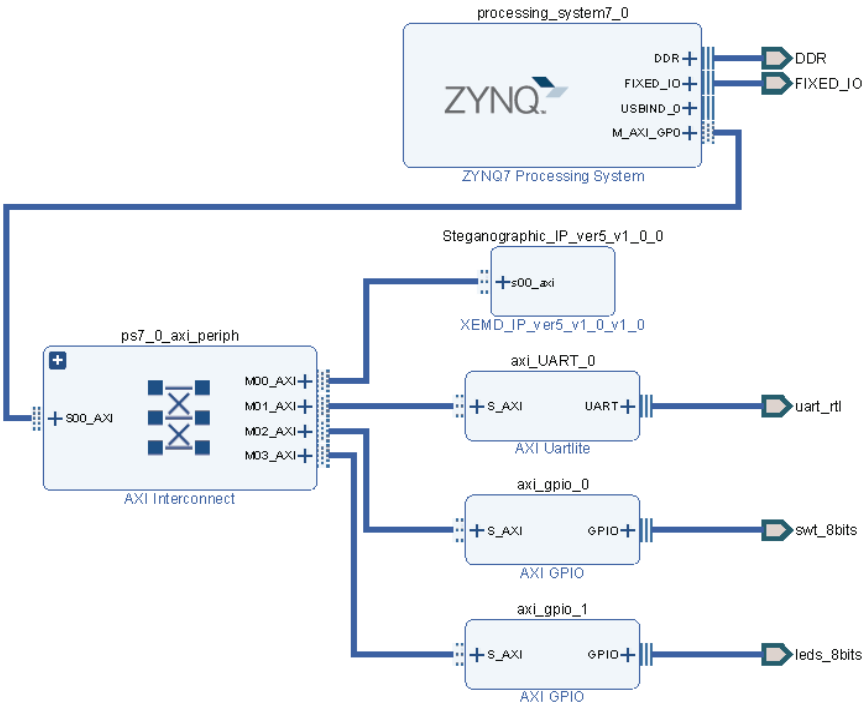


Figure 6.7: Overall diagram of the image steganographic system (Algorithm 2) using Zynq-7000 APSoC.

6.3 Summary

This chapter focuses on the hardware implementation of the novel UMISPB scheme proposed in Chapter 4, which involves dividing the secret data blocks into two parts. The hardware implementation is carried out on the AMD Xilinx Zynq-7000 APSoC platform, combining hardware and software design methodologies. An IP core is developed to incorporate the proposed steganographic scheme, making efficient use of resources while achieving high speeds, low power consumption, and high throughput. Comparing it with the hardware implementation of the UMIS scheme proposed in Chapter 5, the implemented UMISPB scheme demonstrates higher operating speeds, improved throughput, and reduced power consumption.

Chapter 7

Reconfigurable Embedded System of the Secure Lossless Model for Image Steganography

7.1 Introduction

In recent years, the increasing demand for efficient and secure communication has led to the development of various techniques for information hiding, such as image steganography. One of the challenges in image steganography is to ensure that the hidden data is undetectable and robust against various attacks. Reconfigurable embedded systems have emerged as a promising solution to address these challenges. These systems can be reprogrammed to perform various tasks, making them suitable for implementing image steganographic schemes.

Reconfigurable embedded systems combine the advantages of both hardware and software implementations, allowing for high performance, low power consumption, and flexibility in the design process. These systems can be optimized for specific tasks and can also be reprogrammed to accommodate changes in requirements, making them ideal for image steganographic applications. The use of reconfigurable embedded systems in steganography can enhance the security and efficiency of information hiding techniques.

The Xilinx Zynq-7000 APSoC platform is a hybrid system-on-chip that combines a dual-core ARM Cortex-A9 processor with programmable logic on a single chip. This platform offers a high level of flexibility and customization, allowing for the implementation of various applications in different domains, including image steganography. The programmable logic can be used to implement custom hardware designs, which can accelerate image steganographic schemes and enhance their performance. Moreover, the ARM Cortex-A9 processor provides a high level of processing power, which can be utilized for tasks that require high computational resources. The platform also includes a range of peripherals, such as Ethernet, USB, and memory interfaces, which can be used for data transfer and storage. These features make the AMD Xilinx Zynq-7000 APSoC platform a suitable choice for implementing the reconfigurable embedded system. The flexibility and performance of the platform can be leveraged to design a high-performance steganographic embedded system that is robust against various attacks and suitable for various applications.

This chapter introduces the reconfigurable embedded system designed to facilitate secure lossless image steganography. The system integrates multiple modules to enable efficient and reliable data hiding operations. One crucial component is the encryption module that utilizes the 128-bit AES cryptosystem for effective data protection. The recovery module incorporates the EC/Paillier cryptosystems to ensure cover image

restoration. The embedding module features the hardware implementation of the image steganographic scheme UMISPB presented in Chapter 4. The secure and lossless model architecture depicted in Figure 7.1 comprises the following modules:

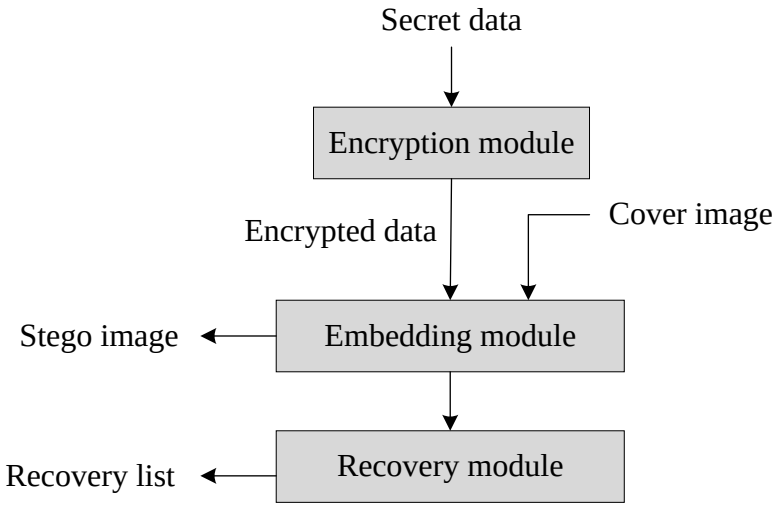


Figure 7.1: Modules of the secure lossless model for image steganography.

1. The encryption module utilizes the hardware implementation of the private-key 128-bit AES cryptosystem, as described in the work in [110].
2. The embedding module uses the novel modulus-based steganographic scheme UMISPB proposed in Chapter 4 and its hardware implementation described in Chapter 6.
3. The recovery module utilizes the hardware implementations of two public-key cryptosystems, EC and Paillier, as described in the works in [111], [112] and [113], respectively.

This integration of encryption, recovery, and embedding modules establishes a comprehensive solution for secure and lossless image steganography operations. This reconfigurable embedded system offers flexibility, adaptability, and high performance, allowing for effective and reliable image steganography operations while maintaining data integrity and confidentiality.

For implementing the secure lossless model on a reconfigurable embedded system, the ZedBoard Zynq-7000 board is selected [114]. There are many FPGA boards available in the market specifically suitable for steganographic embedded systems, such as the Nexys Video Artix-7 FPGA, AMD Xilinx Zynq-7000 SoC ZC706 Evaluation Kit, and ZedBoard Zynq-7000. Our criteria for choosing a board are that it should support RTOS-based design and image processing operations, have multiple I/O expansions (Ethernet, UART, SD card, Switches, Buttons, LEDs, PMODS) for easy access to the processing system and programmable logic, have a hardcore processor (ARM) rather than a reconfigurable softcore processor (MicroBlaze) for better performance in terms of operating speed and resource utilization, and be low-cost. The Nexys board, despite its capabilities in image processing, is not suitable for our steganographic embedded system due to its lack of a hardcore processor, resulting in poor performance in terms of operating speed and resource utilization. On the other hand, both the AMD Xilinx Zynq-7000 SoC ZC706 and ZedBoard Zynq-7000 provide similar performance in these areas. However, the ZedBoard Zynq-7000 stands out for its low cost, making it the preferred choice for the implementation of our proposed steganographic embedded system.

7.2 The Proposed Hardware Implementation of Advanced Encryption Standard (AES)

A high-performance hardware implementation of AES cryptosystem is introduced using the AMD Xilinx FPGA platform [110]. The hardware implementation provides a balanced performance in terms of speed and utilized resources. The balanced performance is achieved by exploiting the fully sub-pipelined architecture in the hardware implementation of AES cryptosystem. Measuring the critical data path delays between the modules of the implementation is done to find and determine the efficient locations and number of stages when inserting the pipeline registers. The proposed high-performance hardware implementation targets embedded systems that require high throughputs as those used in large bandwidth networks, network attached storages (NAS), internet of things (IoTs) network processors.

7.2.1 High-speed Hardware Implementation

Pipelining is one of the architectural optimization techniques that provides a high performance in terms of speed and throughput. The main concept of introducing the pipelining into any hardware design is to break the long critical data paths between resources of the FPGA device, mainly, between user-defined combination logic functions. This happens by inserting efficiently a specific number of pipeline registers between modules of the hardware design.

The S-box table is one of the crucial components in the AES, where the values of the table must be constructed initially before encrypting or decrypting data. There are two paths for generating the S-box table, either by implementing the logic functions for finding the multiplicative inverse of a byte over the $GF(2^8)$ using the $(x^8+x^4+x^3+x+1)$ irreducible polynomial [115] or employing the BRAMs to store precalculated values of the S-box. Also, the MixColumns transformation can be performed by obtaining the multiplication over $GF(2^8)$ results, and this requires developing a polynomial arithmetic module, therefore, consuming more resources.

It is known that logic functions in the combinational circuits tend to have long critical data paths (i.e., low frequencies) [116]. This can be optimized by applying fully sub pipelined architecture in expense of consuming large number of resources (slices, LUTs, and FFs) in the FPGA device. Utilizing the resources efficiently is achieved by our proposed pipelined design, where the SubBytes/InvSubBytes and MixColumns/InvMixColumns transformations are combined in the BRAMs of the FPGA device. The combination is done by substituting and mixing of columns at the same time. This happens by finding results of multiplying a substituted byte in the state column with the MixColumns constant matrix. That can be translated in the hardware by utilizing the FPGA BRAMs for substituting and mixing columns. In the FPGA device, there are BRAMs which each stores up to 36 $Kbit$ of data and can be configured as either two independents (dual port) 18 $Kbit$ BRAMs, or one 36 $Kbit$ BRAM [89].

To perform full substitution and mixing column, each byte in the state column is linked to a 18 $Kbit$ BRAM. Two bytes can be linked to a single dual port BRAM. For a single state column, 2 of dual port BRAMs are required..

The state has 16 bytes, 8 of dual port BRAMs are utilized for performing full substitution and mixing column. A single dual port BRAM contains two independent memories, each is an 8 width and a depth of 256 (e.g., 8x256 RAM). First memory consists of all combinations of the first byte in the state column being substituted and mixed with the MixColumns constant 2311, while the second memory has all

combinations for the second byte in the state column being substituted and mixed with the MixColumns constant 1231. Another single dual port BRAM is required for substituting and mixing the left 2 bytes in the state column with the MixColumns constants 1123, and 3112, respectively. Figure 7.2 shows a state being substituted and mixed using 8 dual port BRAMs. In the decryption process, the substitution and mixing columns are done by storing all combinations of substituting and field multiplying bytes with the InvMixColumns constants *ebd*9, 9*ebd*, *d*9*eb* and *bd*9*e* in 8 dual port BRAMs for the 4 columns in the state.

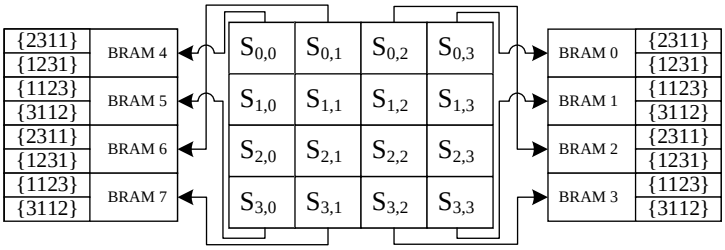


Figure 7.2: Combining substitution and column mixing into BRAMs.

In our design, we insert different numbers of pipelining stages. A single pipelining stage is inserted after each round. Two pipelining stages are inserted, one after each round, and the second after the combined substitution and column mixing BRAMs as sub pipeline register. Three pipelining stages are inserted. One after each round, one as sub pipeline register after the ShiftRows transformation, and the last sub pipeline register is inserted after combined substitution and column mixing BRAMs. The best delay-to-LUTs combination is the implementation with 3 pipelining stages are inserted. Figure 7.3 shows the proposed fully sub pipelined architecture for one AES encryption round using BRAMs.

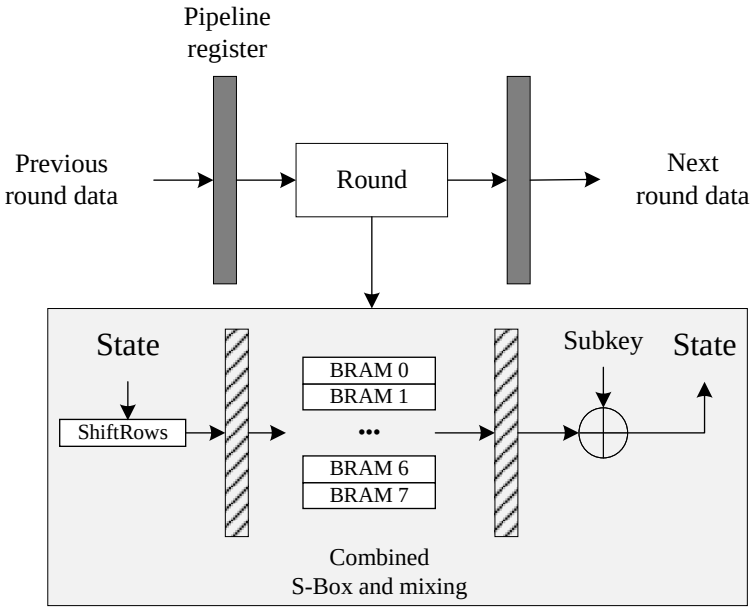


Figure 7.3: Fully sub pipelined AES encryption round.

7.2.2 Experimental Results and Comparisons

The proposed pipelined design for the AES algorithm with the fully sub pipelined architecture is evaluated on the FPGA hardware platform. The metrics for evaluating the performance of the proposed design are the speed (i.e. operating frequency (Hz)), utilized resources (LUTs, FFs, and BRAMs), and throughput (Mbps). Verilog Hardware Description Language (HDL) is employed to develop the proposed design. Validating the performance of the proposed design is achieved by using the AMD Xilinx FPGA devices Artix-7, Artix-7 (low voltage), Virtex-7, Kintex-7, Kintex-7 (low voltage), Spartan-7, and Kintex UltraScale. Artix-7, Virtex-7, Spartan-7, and Kintex-7 devices are fabricated using the 28nm technology, while the Kintex UltraScale is fabricated using the 20nm technology. The Kintex-7 devices target the complex systems such as 4G and 5G communications, while the Artix-7 FPGA devices are used in applications that are mid-performance in nature such as the vision cameras and low-end wireless networks. The Virtex-7 FPGA devices are optimized to deliver high performance by including advanced DSP units and large I/O bandwidths to complex applications such as networking systems (10G to 100G), portable radars, and prototyping circuits. The Spartan-7 FPGA devices offer dedicated security features and ideally suited for any to any connectivity, sensor fusion, and vision embedded systems.

The hardware implementation has been fully synthesized, translated, placed and routed using the new AMD Xilinx Vivado 2020 design suite [117]. The optimization goal of the implementation applies a balance design strategy. A time constraint is applied with a zero-timing error in the generated timing report after placing and routing the implementation. Table 7.1 shows the placed and routed results of the proposed fully sub pipelined architecture. Table 7.2 shows the experimental results comparison between our fully sub pipelined hardware implementation and other previous pipelined AES hardware implementations.

Table 7.1: Place and route results of the implemented AES design using different FPGA devices.

FGPA family	**Artix-7**	**Artix-7 Low voltage**	**Kintex-7**	**Kintex-7 Low Voltage**	**Virtex-7**	**Spartan-7**	**Kintex UltraScale**
Device	XC7A200TL	XC7A100TLF	XC7K160TFF	XC7K160TLFF	XC7V585TFF	XC7S75FGG	XCKU025
FFs	3537	3537	3537	3537	3537	3537	3,537
LUTs	1603	1622	1602	1627	1614	1640	1,701
Slices	1207	1420	1157	1360	1355	1430	1047
BRAMs	80	80	80	80	80	80	80
Freq. (MHz)	370	334	374	373	374	371	498
Throughput ($Mbps$)	47360	42752	47872	47744	47872	47488	63744
Efficiency ($Mbps$/**slice**)	39.3	30.1	41.4	35.1	35.4	33.3	60.9

Our high-speed hardware implementation outperforms the state-of-the-artwork. The results show that fully sub pipelined architecture benefits the design to achieve the least requirements in terms of resources, it performs on the fastest in operating frequencies, and it achieves the highest throughput. Table 7.2 shows that our design is 1.6 times faster in the frequency than that achievable by the design in [121]. Although our design consumes more slices than that consumed by [122], the operating frequency in [122] is very slow and it is 86% slower than our design. This slow performance in [122] is expected due to lack of any architectural optimization techniques such as pipelining. From Table 7.1 and Table 7.2, it is observed that the proposed pipelined design occupies less resources, offers high throughputs, and achieves high efficiencies compared with other existing works. This makes our design a well-suited for embedded systems that have large bandwidths, streaming services, high performance computing (HPC) clusters,

Table 7.2: Comparison of the proposed AES implementation with other implementations (place and route results).

Ref.	**LUTs**	**Slices**	**Freq.** (MHz)	**Throughput** ($Mbps$)	**Efficiency** ($Mbps$/**slice**)
[118] 2019	12640	13647	244	**31290**	2.29
[119] 2019	12640	1120	112.37	14383	1.14
[120] 2020	**3784**	1624	190.65	24320	**6.5**
[121] 2020	7758	-	**313**	20.3	2.85
[122] 2021	10110	**860**	69.21	1610.4	0.16
Our design	**1701**	**1047**	**498**	**63744**	**60.9**

The red color represents the 1^{st} best result and the green color represents the 2^{nd} best result.

IoTs networking, large NAS systems. Also, our design is suitable for embedded systems and applications that have a compact footprint, and limited resources.

7.3 The Proposed Hardware Implementation of Elliptic Curve Cryptosystem (ECC)

Scalar point multiplication (SPM) is the main operation that dominates the ECC-based cryptosystems [111], [112]. SPM is the process of adding a point k times, where k is a positive integer and P is a point on the selected curve. SPM is based on the implementation of the underlying point operations, where a series of doubling and adding points is performed by scanning the binary sequence of the scalar k, where $k = k_n k_{n-1} \cdots k_1 k_0$.

There are several methods and techniques to implement $k \cdot P$ efficiently: Binary Add-and-Double Left-Right, Binary Add-and-Double Right-Left as presented in [123], Non-Adjacent-Form (NAF) [124] and Montgomery ladder method [125]. Affine coordinate system is the default representation that some SPM methods use to introduce the points on the elliptic curves, while other SPMs utilize alternative projective coordinate systems for representing the points. The reason for applying different projective systems is to avoid the time/resource consuming inversion field operation. However, it increases the number of field multiplications. Generally, projective systems tend to provide efficient designs of ECC cryptosystem in terms of area/latency.

Lopez-Dahab (LD) [126] projective system is one of the most efficient projective coordinate systems. Points in affine system are mapped to LD projective system, where P at (x, y) coordinate is equal to P at $(X = x, Y = y, Z = 1)$ coordinate. In any SPM, the point multiplication algorithm must be selected first for performing the $k \cdot P$ operation. The next step is to define the projective coordinate operation system for representing the points. The final step is to choose algorithms for the finite field operations, mainly for the multiplication and inversion operations. Figure 7.4 illustrates the required components for constructing SPM in the proposed ECC cryptosystem. In the present work, Montgomery ladder algorithm is adopted as the point multiplication algorithm and the LD coordinate system to represent the points [111]. For the field multiplication and inversion operations, the Karatsuba-Ofman algorithm and the Itoh-Tsuji algorithm are, respectively, implemented.

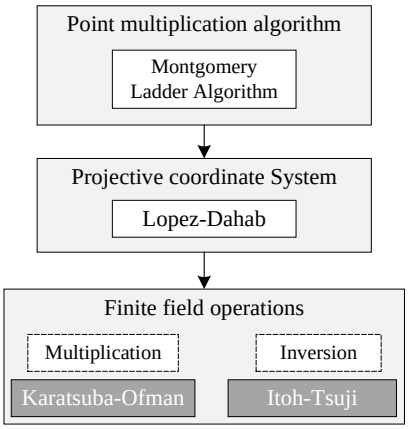


Figure 7.4: Main components of SPM.

7.3.1 High-speed Core for ECC over $GF(2^n)$

The proposed architecture of the high-speed ECC core is over $GF(2^n)$ using NIST binary recommended curves. Karatsuba-Ofman and Itoh-Tsuji algorithms are implemented for performing field multiplication and inversion operations, respectively. The LD coordinate system is used as a projective coordinate system to present the points. For SPM, a modified-pipelined Montgomery ladder method is implemented in such a way that an efficient number of pipelined stages are inserted.

The proposed high-speed area-efficient hardware design is shown in Figure 7.5. The general architecture has three units, an optimized finite-state machine control unit, a $GF(2^n)$ arithmetic unit containing a squarer and a Karatsuba-Ofman multiplier, and a control signal unit. Next, further details are given for these main units in the high-speed ECC core.

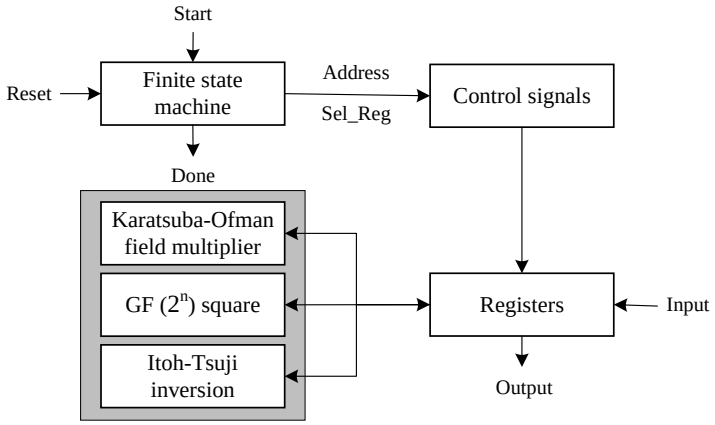


Figure 7.5: The proposed high-speed area-efficient hardware design.

7.3.1.1 FPGA Implementation: Results and Comparisons

The pipelining architectural approach is applied to the proposed ECC core for higher speed with an efficient utilized area in terms of both the working frequencies and the slices. Elliptic curve doubling and adding point operations are performed using a merged-improved Montgomery ladder scalar point multiplication algorithm. The proposed ECC core does not require any precomputed values or memories for calculations, thus resulting in an efficient design with fewer slices and clock cycles.

Table 7.3: FPGA results of the proposed high-speed ECC core (place and route).

GF (2^n)	Device	LUTs	FFs	Slices	Clock Cycles	Freq (MHz)	Time (μs)	Efficiency
163	Virtex-5	8,900	3,044	2,814	226	291	0.78	74584.42
233		15,672	4,333	4,585	327	217	1.51	33723.19
571		72,259	10,259	21,720	674	198	3.40	7722.93
163	Virtex-7	8,409	3,023	2,780	226	360	0.63	93397.85
233		16,003	4,323	4,762	327	310	1.05	46385.32
571		71,180	10,384	18,178	674	290	2.32	13515.38
163	Artix-7	8,417	3,016	2,693	226	191	1.18	51153.6
233		15,236	4,489	3,977	327	209	1.56	37445.44
163	Kintex-7	8,437	3,023	3,030	226	311	0.73	74028.16
233		15,221	4,483	4,915	327	309	1.06	44796.41
233	Virtex-6	14,402	4,203	4,143	327	239	1.451	38759.1

An effective FSM is developed to control the main ECC components, where a minimum number of states are used for performing the merged-improved Montgomery ladder SPM. To verify the performance of the proposed SPM, our high-speed ECC core is implemented over three finite fields, $GF(2^{163})$, $GF(2^{233})$ and $GF(2^{571})$, using several FPGA devices provided by AMD Xilinx: Virtex-5, Virtex-6, Virtex-7, Kintex-7 and Artix-7. The high-speed core has been synthesized, placed and routed using AMD Xilinx ISE 14.4 design suite [127].

In AMD Xilinx ISE tool, the place and route process reads a netlist of the synthesized design, extracts the components and logic elements, places them on the target FPGA device, and finally interconnects these components and logic elements using CLB routers. For better place and routing process, strategies in placing and interconnecting can be applied such as register balancing and combinational logic optimization. A time constraint is applied for all results to achieve better area-speed ratio with zero timing error. Table 7.3 presents the place and route results for the proposed high-speed ECC core. Table 7.4 represents our design compared with other previous ECC hardware implementations.

The proposed high-speed core provides higher speed in both Virtex-7 and Kintex-7 FPGA devices, since they are fabricated and optimized at 28nm technology. The Artix-7 device consumes fewer resources making it well suited for the battery-powered cell phones, automotive, commercial digital cameras and IP cores of SoCs.

The area-efficient hardware implementation in [129] consumes less area than the proposed core, but requires 95% more clock cycles. This large number of clock cycles is the result of the increased writing/reading in the distributed RAM-based memory and register shift operations. In [130], a pipelined architecture is applied to SPM, which achieves high performance in terms of the area and working frequencies. However, it requires 84% more clock cycles than that required in our proposed clock cycles, using the same device and field. Our high-speed core achieves greater efficiency than the design presented in [131]. Our design has 88% fewer clock cycles, 33% higher frequency, and 44% fewer slices than that in [131]. Using a Kintex-7 FPGA device, the design in [132] performs on 39% fewer slices than that in the proposed core over $GF(2^{233})$, but requires large number of clock cycles. In [132], an iterative-based architecture is adopted by all main SPM operations, where the binary (left-to-right) algorithm is used for scalar multiplication, an interleaved field multiplier is implemented for the multiplication operations, and a modified Extended-Euclidian is applied for the inversion operation.

Table 7.4: Comparison between our high-speed ECC core and other works over $GF(2^n)$.

	GF (2^n)	Device	LUTs / FFs	Slices	Clock Cycles	Freq (MHz)	Time (μs)	Efficiency
[128]	163	Vertix-5	- / -	5,768	-	343	5.08	5562.87
	233	Virtex-7	- / -	10,601	-	359	6.84	3213.32
[129]	163	Virtex-5	3,958 / 1,522	1,089	4168	296	14.06	10645.71
		Virtex-7	4,721 / 1,886	1,476	4168	397	10.51	64.47
[130]	163	Virtex-5	9,470 / 4,526	3,041	1,363	294	4.6	11652.35
	233		15,296 / 6,559	4,762	1,926	244	7.9	6193.55
[131]	163	Virtex-5	22,039 / -	6,059	1,591	200	8.1	3321.26
	233		28,683 / -	8,134	2,889	145	19.9	1439.46
	571		32,432 / -	11,640	44,047	126	348	140.97
[132]	233	Kintex-7	9,151 / 9,407	3,016	679776	255.66	2.66	29043.1
Ours	163	Virtex-5	8,900 / 3,044	2,814	226	291	0.78	74584.42
	233		15,672 / 4,333	4,585	327	217	1.51	33723.19
	571		72,259 / 10,259	21,720	674	198	3.40	7722.93
	163	Virtex-7	8,409 / 3,023	2,780	226	360	0.63	93397.85
	233		16,003 / 4,323	4,762	327	310	1.05	46385.32
	571		71,180 / 10,384	18,178	674	290	2.32	13515.38
	163	Kintex-7	8,437 / 3,023	3,030	226	311	0.73	74028.16
	233		15,221 / 4,483	4,915	327	309	1.06	44796.41

To summarize the comparisons: shift registers, segmented multipliers, or memory-based implementations result in an increased latency (clock cycle) as in [129] and [131]. Iterative architecture is not an efficient way to achieve higher speeds as mentioned in [132]. Pipelining architecture is a more practical way of achieving better performance and maintaining the balance between speed and area, as stated in [128] and [130]. The balance in the speed-area ratio can be achieved when the optimal number of pipelined stages are inserted. Our high-performance ECC core requires fewer slices, and offers decreased latencies with higher working frequencies. This high-speed area-efficient ECC core makes it very suitable for use in different kinds of real-time embedded systems such as cellphone banking services, health-care monitoring using smart watches, and accessing office networks and storage devices while abroad.

7.4 The Proposed Hardware Implementation of Paillier Cryptosystem

A new high-performance public-key image steganographic framework is developed and implemented using FPGA platform [113]. The Paillier public-key cryptosystem is used to encrypt pixels of the cover images [133]. The cover image is forwarded to the embedding process to conceal the secret data. The pixels are encrypted by the Paillier cryptosystem with a public key to generate encrypted stego image. In a reversible way, the receiver, who has the private part (i.e. private key) of the public key, can decrypt the received encrypted stego image. The result is a stego image that has the secret data. Extracting process is applied after. Reconstructing the cover image is done as a last step by sorting the decrypted pixels.

7.4.1 Proposed Hardware Architecture

In this section, the structure of the proposed image steganographic cryptosystem is presented. The structure consists of components that works together to embed-extract and encrypt-decrypt a single pixel. We should mention that all pixels of the cover image are within grayscale levels (e.g. 0 to 255). Each component at both encryption and decryption processes is controlled by an efficient finite state machine. Figure 7.6 represents the main components of the proposed steganographic cryptosystem, and Paillier cryptosystem is used for encrypting and decrypting pixels.

7.4.2 FPGA Implementation: Results and Comparisons

In this section, the results for the hardware implementation of the steganographic cryptosystem are presented. Verilog Hardware Description Language (HDL) is used to implement the proposed cryptosystem. Verifying the performance of the proposed design is achieved by using the FPGA platform devices which are provided by AMD Xilinx, where two FPGA families are used, Artix-7 and Kintex-7 devices [89]. Both FPGA devices are fabricated using the common 28nm technology. The Kintex-7 targets the high-density complex applications such as those in 3G and 4G communications, while the Artix-7 FGPA provides mid-performance for the applications running over power-sensitive systems including the vision cameras and low-end wireless networks. The hardware implementation has been fully synthesized, translated, placed and routed using the new AMD Xilinx Vivado 2018.2 design suite. A balance design strategy is applied as an optimization goal. A time constraint is applied, and the timing constraints report provides a zero timing error. Table 7.5 shows the results for the proposed steganographic cryptosystem after place and route.

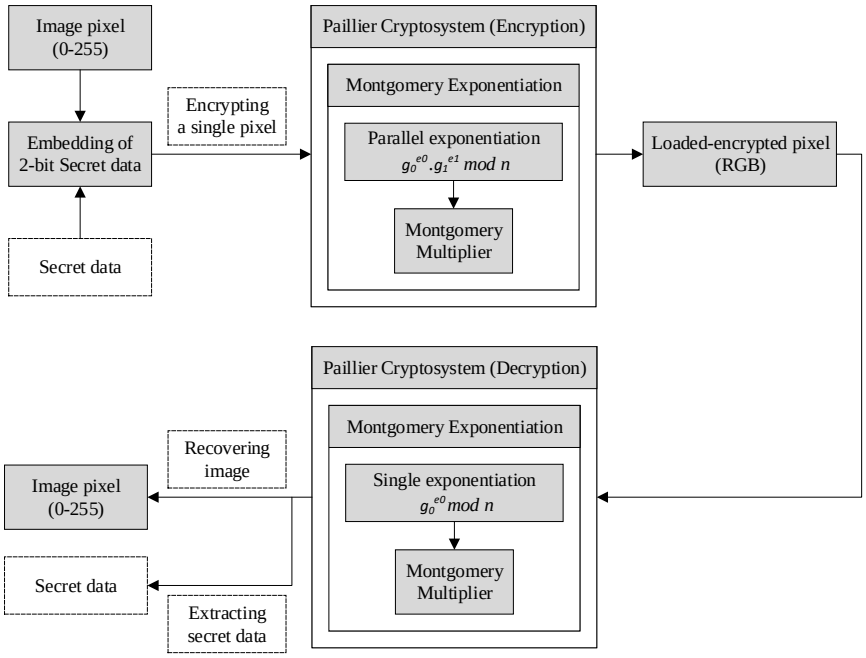


Figure 7.6: The main components of the proposed steganographic cryptosystem in encryption and decryption processes.

Table 7.5: Place and route results for encryption and decryption processes.

Size	Device	Encryption					Decryption				
		Slices	FFs	LUTs	BRAM	Freq. (MHz)	Slices	FFs	LUTs	BRAM	Freq. (MHz)
64	Artix-7	148	390	409	1.5	135.2	326	971	768	4	166.7
128		151	388	411	5	134.8	317	980	774	8.5	166.5
256		193	409	469	20	133.3	334	1000	819	34	166.6
64	Kintex-7	155	390	429	1.5	286	333	967	854	4	333.3
128		165	388	430	5	270.27	347	980	866	8.5	333.3
256		179	409	499	20	250	349	1000	850	34	333.3

Embedding-encryption module implementation provides a running frequency up to 135.2 MHz using 193 slices of the available resources in Artix-7 FGPA. In Decryption-extraction module implementation, up to 166.7 MHz of running frequency is achieved with 334 slices. Block RAMs (BRAMs) are used as a memory to read the cover image and write the loaded-encrypted stego image. More BRAMs are utilized when the size of the cover image is increased. The modules in Kintex-7 provides higher running frequencies between 250 MHz to 333.3 MHz in embedding-encryption and decryption-extraction, respectively.

The total on-chip power consumption is measured through calculating the power required by the resources in the hardware implementation such as clock, slices (i.e. LUTs and registers) and BRAMs unites at the maximum running frequencies. For the embedding-encryption module, our design consumes 0.139 $Watt$ for 64 $\times$ 64 cover image to 0.134 $Watt$ for 256 $\times$ 256 cover, while the decryption-extraction module takes up to 0.362 Watt of power consumption for 256 $\times$ 256 cover image. The results in Table 7.5 shows that our proposed cryptosystem is suitable for the limited-resources low-power embedded systems.

Table 7.6 represents a comparison between the proposed hardware implementation of the Paillier cryptosystem and other implementations. Our design outperforms the works in terms of embedding rate, utilized slices and frequency. The hardware implementation in [134] is designed using interpolation expansion method. The design achieved a high

performance in terms of the throughput, which is capable of processing about 7640 pixels in a single second using Virtex-6 FPGA device.

On the other hand, a large amount of resources as register and LUTs are utilized, where it requires around 10 times of the resource utilized by our design. Our design has twice the embedding capacity of the design in [134], and it consumes less power by 91.5%. The work in [137] represents methods of encapsulating secret data into a group n of pixels in a $(2n+1)-$ary notational system.

The hardware implementation offers high fps but with low PSNR, fixed 1-bit embedding rate and high utilized resources, where 16% of the total resources in Artix-7 FPGA device is roughly 12,000 logic cells. The power consumption in [137] is very high, it is 3.807 Watt, while our hardware implementation consumes between 0.139 Watt to 0.362 Watt of power.

Table 7.6: Performance comparison with existing hardware implementations for image steganography

Ref.	Image Size	Device	*bpp*	PSNR (dB)	Slices	LUTs	FFs	Freq (MHz)	TR ($Kpps$)
[135] (2014)	32 × 32	Spartan 3	0.46	29.46	9881	11291	9347	98	100.3
[134] (2015)	328 × 264	Virtex-5	1	-	-	9715	4883	96.4	7340.17
		Virtex 6	1	-	-	8564	4908	100.4	7640.63
[136] (2017)	128 × 128	Spartan 6	1	51.14	160	322	214	104.971	-
[137] (2019)	512 × 512	Virtex 7	1	45.11	16%	1%	2%	144	-
Ours	128 × 128	Artix-7	2	52.43	151	774	980	134.8	370
	256 × 256	Kintex-7	2	51.87	179	850	1000	250	680

7.5 Generating Recovery List Using the EC/Paillier Cryptosystem in the Recovery Module

EC and Paillier cryptosystems are used to generate the recovery point (RP) list in such a way that receiver can decrypt it to reveal the original values of the pixels for recovering the original cover image. Using EC, each pixel P is divided into two values X_0 and X_1. This point is considered as plaintext for the EC cryptosystem. Encryption process generates the encrypted point (C_{X_0}, C_{X_1}), which is registered into the RP list. This process goes on until all pixels are encrypted. On the other hand, Paillier cryptosystem takes the pixel P as plaintext M and encrypts it to get the encrypted cipher C. Figure 7.7 represents the mechanism of creating the RP list using the EC and Paillier cryptosystems.

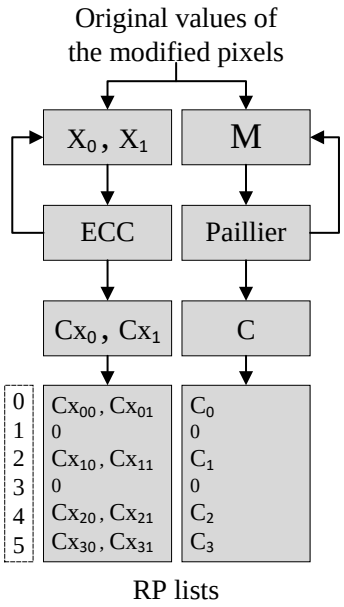


Figure 7.7: Generating RP list using EC and Paillier cryptosystem.

Encryption and decryption using EC cryptosystem follows the ElGamal cryptosystem procedure. Figure 7.8 illustrates the procedure using EC cryptosystem. Generating keys are done at the receiver side and the sender uses the public key to encrypt the messages. Note that E_p is the selected prime field, (a, b) are parameters for the selected elliptic curve, and r is a random number.

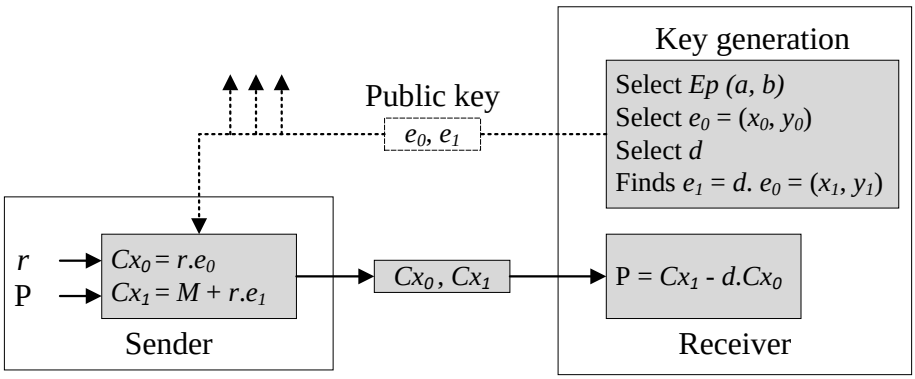


Figure 7.8: Encryption and decryption processes using EC cryptosystem.

The following example illustrates the process of generating a recovery point list when the EC cryptosystem is selected to encrypt the corresponding original values of the pixels that are modified in the embedding module.

Example: Encrypting Pixel using EC Cryptosystem:

- Assume the public key pair $e_0(2, 22)$ and $e_1(13, 45)$ using E_{67} prime field, and the private key d is 4.
- Let the original value of pixel $P_0 = 50$ at location 0 in the cover image, for a modified pixel $P'_0 = 53$.
- To encrypt P_0 using EC, divide it into two values such as $P_0 = X_{00} + X_{01}$.
- The P_0 can be divided to $X_{00} = 24$ and $X_{01} = 26$.
- The point $(24, 26)$ is encrypted by the sender to get $C_{X_{00}} = (35, 1)$ and $C_{X_{01}} = (21, 44)$ using the ElGamal procedure shown in Figure 7.8.
- These encrypted points are registered in the RP list at the corresponding location of 0.
- To decrypt the $C_{X_{00}}$ and $C_{X_{01}}$, the receiver finds $dC_{X_{00}}$ and inverse it to get $(23, 42)$.
- Then add it to the $C_{X_{01}}$ to get $(24, 26)$, which adding these values results in the $P_0 = 50$.

7.6 The Proposed Reconfigurable Embedded System using Zynq-7000 APSoC Platform

The general schematic flow of the proposed reconfigurable Zynq-7000 APSoC for the lossless secure model is represented in Figure 7.9. The general schematic has a top-bottom flow, where MATLAB is employed only for image representation and pixel-hex format conversion. There are three inputs to model, the cover image, the secret data, and configurations. The configurations are parameters that are entered by the user to control what the model is going to perform. The configurations are as follows. (i) parameters x_0 and μ for the permutation module. (ii) parameters m_x, m_y, m_z, C_{EMD}, and C_{DE} for the embedding module, where the steganographic scheme is implemented. (iii) 128-bit cipher key for the AES cryptosystem implemented in the encryption module. (iv) pair of keys (n, g) and (λ, μ) for the Paillier cryptosystem and a base point $P(x, y)$ for the EC cryptosystem, where both are implemented in the recovery module. (v) EM flag which has two values either 0 or 1, when the flag is 0, it means no encryption required, and when the flag is 1, it indicates the encryption is required. (vi) RM_2 flag which has three values, 00, 01, and 10. When the flag is 00, it means no recovery required, when the flag is 01, it indicates the recovery module performs the EC cryptosystem, and when the flag is 10, it indicates the recovery module performs the Paillier cryptosystem.

It can be seen that all modules are implemented in in the PL part of Zynq-7000 APSoC platform. The implementation of steganographic scheme in the embedding module has two functional modes, embedding and recovery. The embedding mode is for concealing secret data and generating the stego image, while the recovery mode is for extracting the concealed secret data from the stego image. In the embedding mode, the secret data gets concealed using the embedding functions of the EMD and DE schemes.

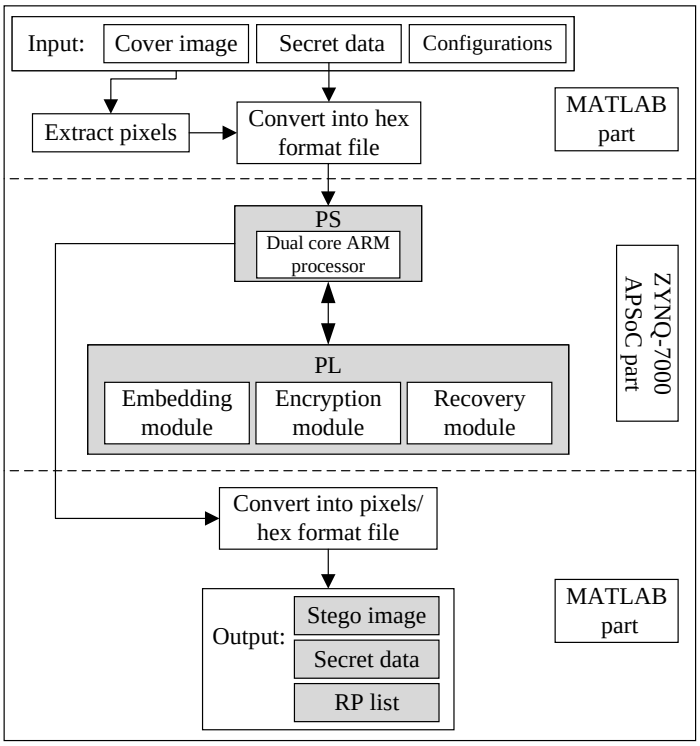


Figure 7.9: The general schematic flow of the proposed reconfigurable Zynq-7000 APSoC for the lossless secure model.

In the recovery mode, the secret data gets extracted using the extraction functions in the way they got concealed during the embedding mode.

In the embedding mode, concealing secret data is done as the hex values of the pixels are scanned. The modified (i.e., stego) permuted hex values are arranged back to the original order of hex values of the input cover image using the decipher key. The stego hex values are converted into stego pixels, and the stego image is generated using MATLAB. If EM is 1, the secret data is encrypted in the encryption module before starting concealing in the embedding module. If RM is 01 or 10, the modified (i.e., stego) permuted hex values are registered into the RP list using the EC or Paillier cryptosystem, respectively. the RP list is generated next to the stego image.

7.6.1 Overall System Architecture

The secure lossless image steganographic embedded system includes the implementation of the embedding module, and the implementations of the encryption and the recovery modules. The three modules are packed into AXI4 IP cores, embedding AXI4 IP core, encryption AXI4 IP core, and recovery AXI4 IP core. The embedded system also contains storage module, I/O peripherals, and communication module. The storage module consists of using the DDR3 RAM and SD card. The communication module includes the UART, the AXI4 interconnect, and GP AXI4 interfaces between PS and PL. The block diagram of the secure lossless image steganographic embedded system is shown in Figure 7.10.

The architecture of the three IP cores in the embedded system is similar to the IP core architecture designed in Figure 6.3. Figure 7.11 shows the three IP cores embedding, encryption, and recovery. These IP cores are implemented to access the GP AXI4 bus, enabling efficient communication and data transfer within the system. Each IP core serves a specific purpose and contributes to the overall operation of the steganographic

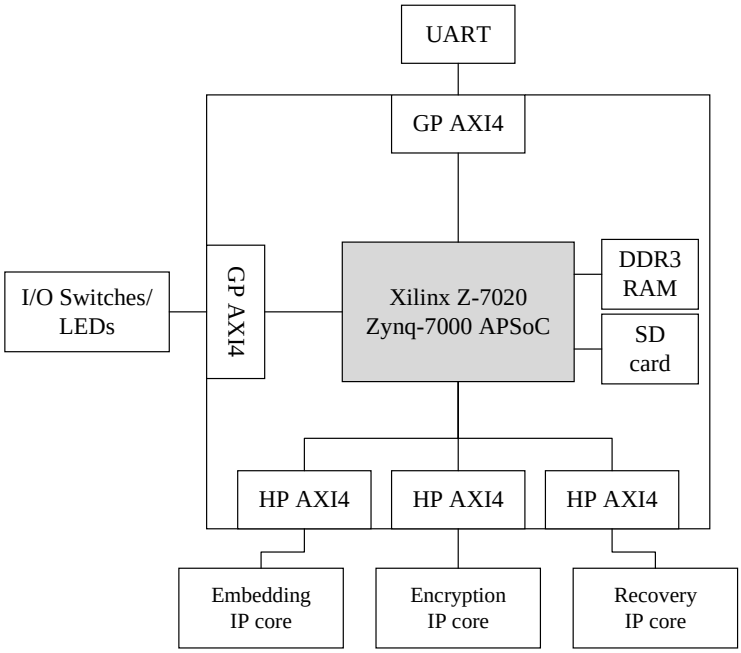


Figure 7.10: Block diagram of the secure lossless image steganographic embedded system.

embedded system.

7.6.2 Simulation: Embedding IP Core

In the field of digital design, simulation is a crucial step in verifying the functionality and performance of a design before it is implemented in hardware. The AMD Xilinx ISim simulator is a powerful simulation tool that allows designers to simulate and debug their designs before synthesizing and implementing them on an FPGA [138]. In this subsection, the embedding IP core is simulated using the AMD Xilinx ISim simulator, where a testbench written in VHDL is created for this purpose. During the simulation, the testbench supplies the cover image's pixels and secret digits to the embedding IP core, which in turn performs the embedding process to generate the stego pixels when the functional mode is 1. The verification is performed by comparing the captured and the expected results. If the results are equal, this means the IP core has successfully generated the correct data. By observing the simulation results, the designer can evaluate the effectiveness of the steganographic scheme and make any necessary adjustments or optimizations to the IP core's design. This iterative process ensures that the final embedding IP core delivers robust and reliable performance in real-world applications.

The AMD Xilinx ISim simulator generates a timing diagram that shows the behavior of the embedding IP core over time. The timing diagram is a graphical representation of the signals and their changes in the IP core during the simulation. It provides designers with a visual representation of the IP core's behavior under different input conditions and enables them to analyze the timing constraints and performance of the design. By analyzing the timing diagram, it can be easier to detect and fix timing errors, optimize the IP core's performance, and ensure that it meets the timing constraints of the target FPGA. Figure 7.12 shows the timing diagram of the simulation when the Mode signal is 1. It can be seen that processing three group of pixels (Q_0, Q_1, Q_2) require 8 clock cycles.

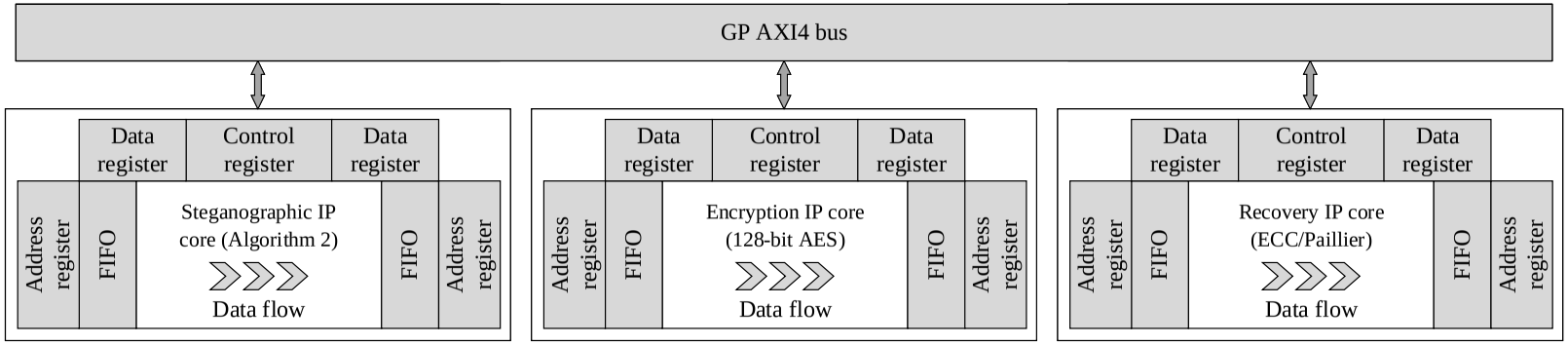


Figure 7.11: The embedding, encryption, and recovery IP cores.

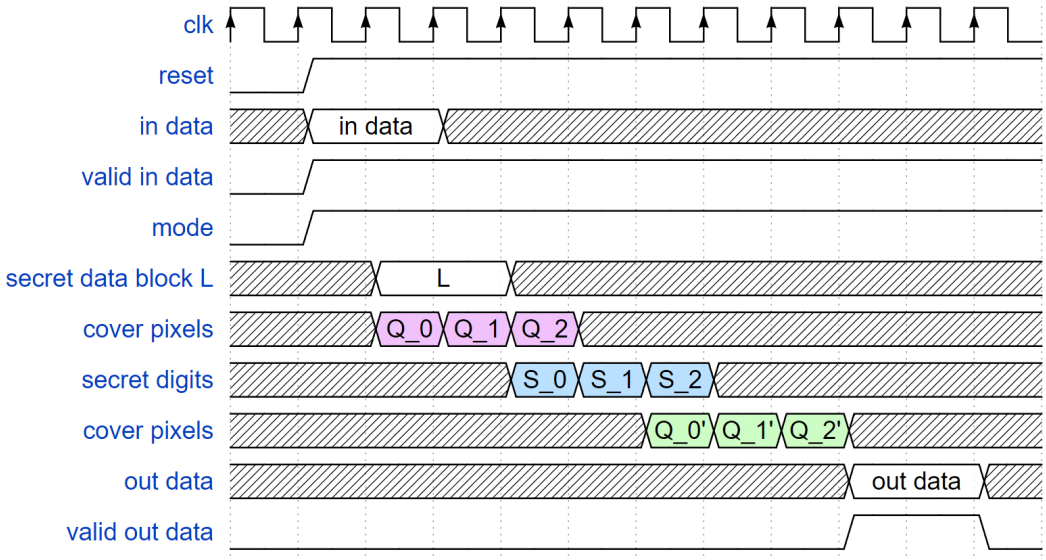


Figure 7.12: Timing diagram generated by AMD Xilinx ISim simulator.

7.6.2.1 RTL Schematic of the Embedding IP Core

The RTL schematic of the embedding IP core, shown in Figure 7.13, provides a visual representation of the core's digital circuit design. The RTL schematic shows the functional blocks, including the input and output interfaces using FIFOs, the data embedding and recovery logic that represents the embedding and extraction functions of the EMD and DE schemes, and the control logic that manages the embedding and recovery operations through the FSM. The schematic also shows the connections between the functional blocks and the data paths that carry the image data and the concealed or extracted data.

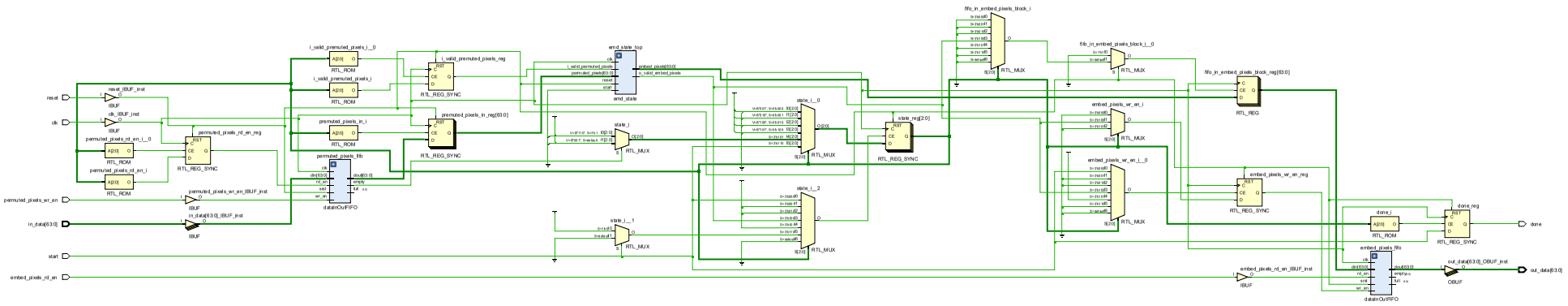


Figure 7.13: RTL schematic of the embedding IP core.

7.7 Finite State Machine of the Secure Lossless Image Steganographic Embedded System

Figure 7.14 shows the FSM of the implemented steganographic embedded system. There are seven states, Idle, Start, Encryption (128-AES), Embedding (Proposed scheme), Recover EC ($GF(2^{163})$), Recovery Paillier, and Done. When the start signal is asserted and the configurations are specified, the Start state is entered. In the Start state, locations of the pixels in the cover image are permuted using the cipher sequence. If encrypting the secret data is required ($EM =' 1'$), secret data is fed to the Encryption IP core, but when the encryption is not required ($EM =' 0'$), the secret data is forwarded directly to the Embedding IP core along with the permuted cover pixels. In the Embedding IP core, the steganographic scheme proposed in Subsection 7.6.1 starts to conceal secret data into the permuted cover pixels. If recovering modified pixels is required, one cryptosystem is performed; EC ($GF(2^{163})$) when ($RM =' 01'$) or Paillier when ($RM =' 10'$). This process keeps going until all secret data are concealed, and stegp pixels are permuted back to the original order using the decipher sequence. At the Done state, the stego pixels are written into a textfile along with RP list if it is required, and the done signal is asserted.

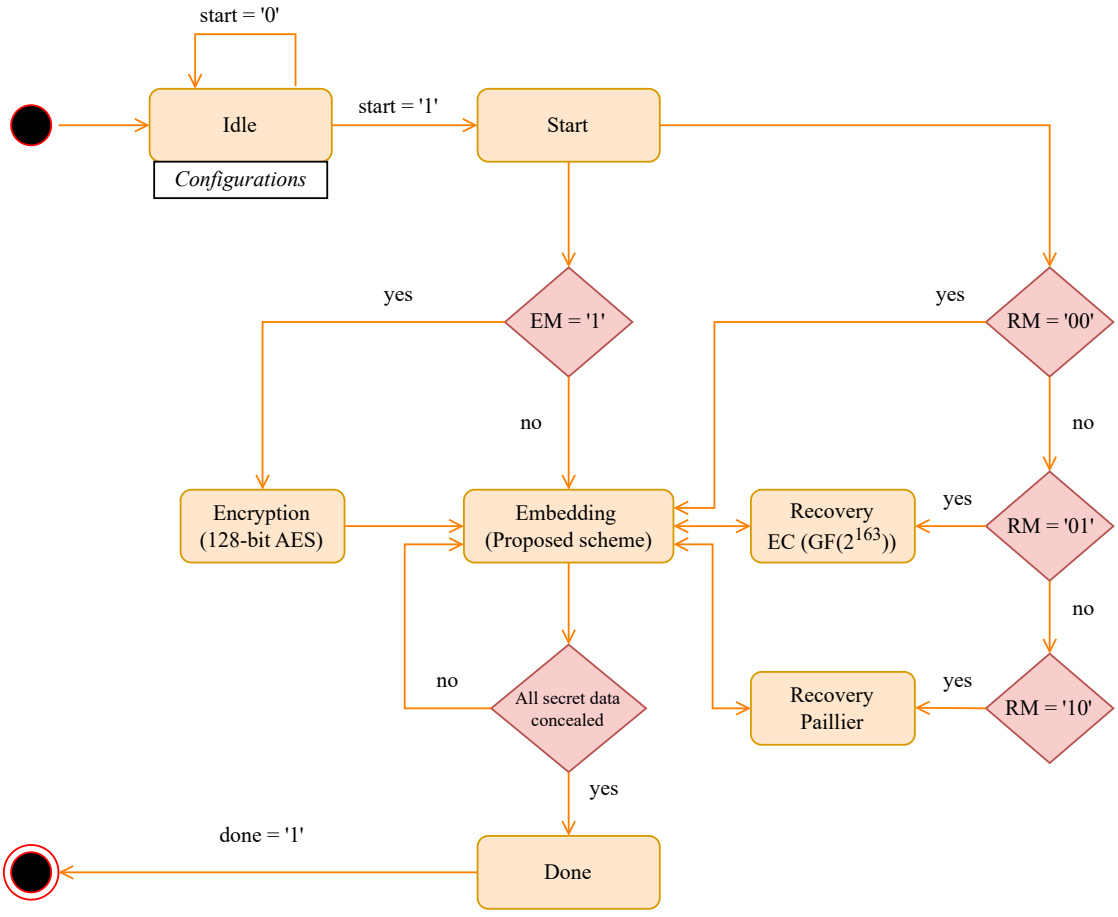


Figure 7.14: FSM of the secure lossless image steganographic embedded system.

7.8 Place and Route Results of the Secure Lossless Image Steganographic Embedded System

In the integration phase, the implemented modules are packaged as IP cores using the AMD Xilinx IP package integrator. This tool is used to rapidly connect the design, implemented on PL, to PS using the AXI4 interface. The overall diagram of the image steganographic system using the Zynq-7000 APSoC is shown in Figure 7.15. It can be seen from this figure that there are six IP cores connected to PS in the Zynq-7000 APSoC. The embedding, encryption, and recovery IP cores are considered custom cores and can be modified to add more features and support more functions. The UART and I/O IP cores, on the other hand, are produced by AMD Xilinx.

Table 7.7 presents the results of the place and route implementation of a secure lossless image steganographic embedded system. These results demonstrate that the implemented embedded system achieves an efficient performance in terms of speed, resource utilization, throughput, and power consumption. The implemented steganographic embedded system provides high operating speed and high throughput due to the pipelining applied in the hardware architectures of the IP cores.

Table 7.7: Performance metrics of the secure lossless image steganographic embedded system: Place and route results.

IP core	**Cryptosystem/Scheme**	**LUTs**	**FFs**	**Slices**	**Freq. (MHz)**	**Power (Watt)**
Embedding	Proposed scheme	320	523	148	289.7	0.167
Encryption	128-bit AES	1,603	3,537	1,207	370	0.97
Recovery	EC ($GF(2^{163})$)	8,417	2,016	2,693	191	0.801
	Paillier	409	390	148	135.2	0.139
Including UART and I/O IP cores						
Total embedded system		11,784	7,852	4,825	295.6	2.736

The XPE tool is used to perform a power analysis and estimate the power consumption of the IP cores. The on-chip total power consumption is calculated by measuring the power consumption values of the system clock, logic blocks, and BRAMs in the FPGA device. It is worth noting that the values of the power consumption given in Table 7.7 represent the power consumed by the PL part (FPGA device) only. The implemented design has a small footprint, making it well-suited for low-powered, resource-constrained embedded systems such as wearable smart devices, radio frequency identification (RFID) cards, and network-attached storage devices. By optimizing the performance and minimizing the resource utilization and power consumption, we can make systems and applications that are featuring the implemented embedded system to be more cost-effective and have a longer battery life.

7.8.1 Address Map for the IP Cores

Figure 7.16 shows the address map for all IP cores in the Zynq-7000 APSoC embedded system. The address map is divided into multiple regions, each of which is assigned to a specific IP core. Each IP core has a reserved space of 64k within the address map, which is used to access the core's registers and other memory-mapped resources. The regions are aligned with the memory space of the system's processor and can be accessed using memory-mapped I/O techniques. The address map is an important component of the system's design, as it ensures that the IP cores can be accessed efficiently and reliably by the processor and other components in the system.

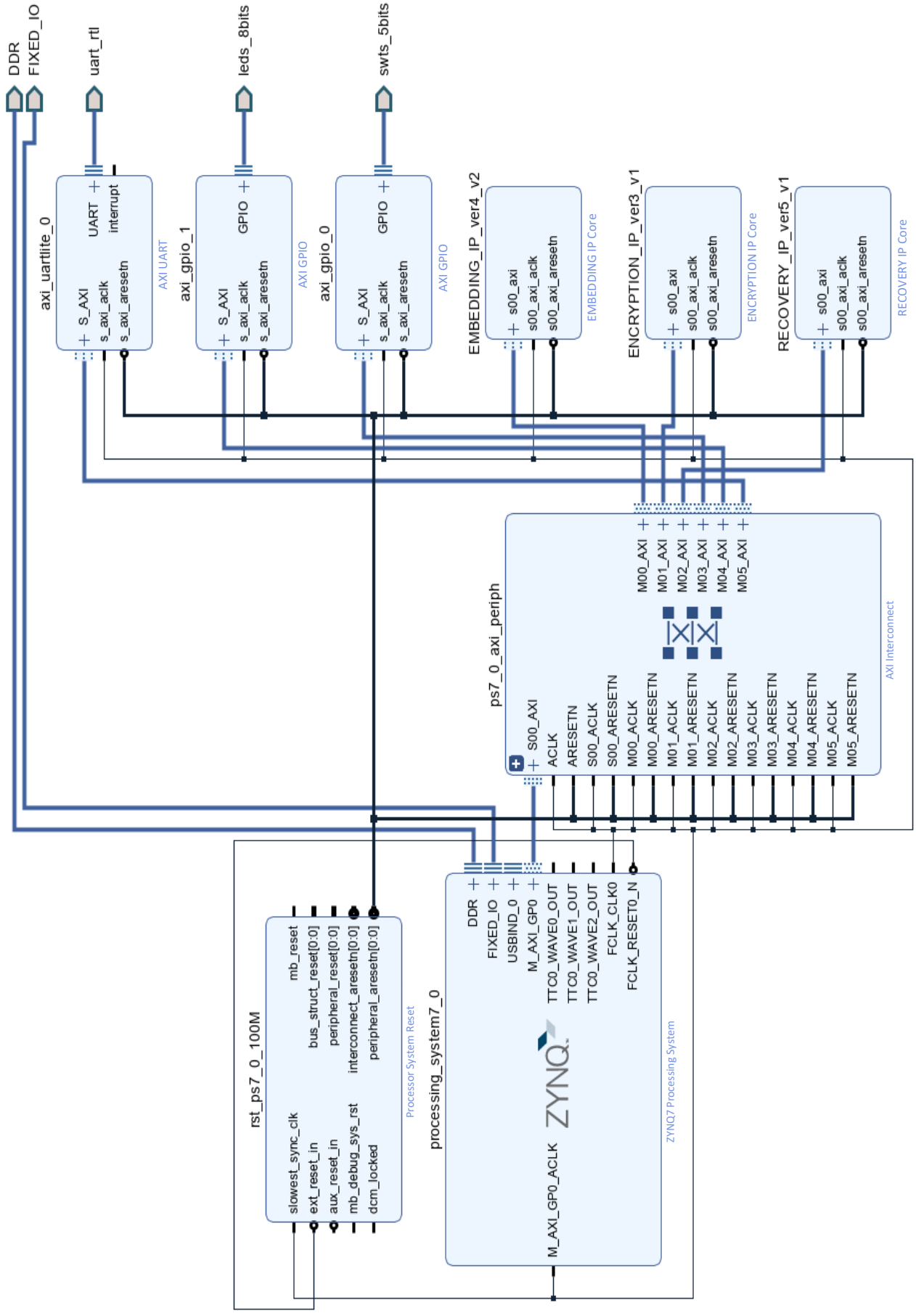


Figure 7.15: Overall diagram of the secure lossless image steganographic embedded system using AMD Xilinx Zynq-7000 APSoC platform.

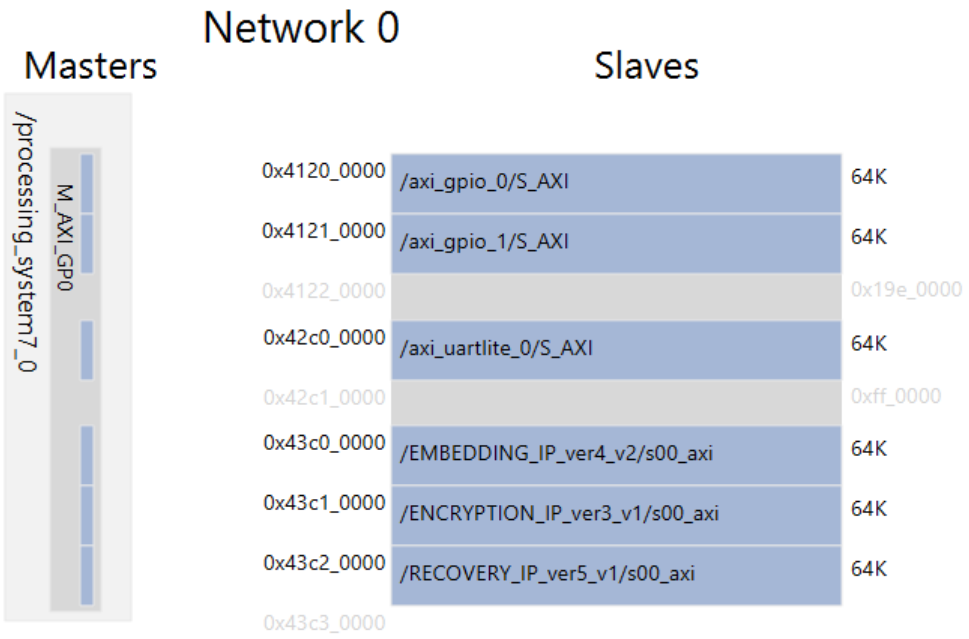


Figure 7.16: Address map for IP cores in the Zynq-7000 APSoC embedded system, with each core assigned a 64k memory space.

7.9 The Prototype: a Real-time Application

Image steganography is widely used in real-time applications to ensure the confidentiality and security of digital images transmitted over untrusted channels. By embedding sensitive data in digital images using steganographic schemes, the privacy and security of the data can be protected and unauthorized access to sensitive information can be prevented. The use of image steganography in real-time applications is becoming increasingly important as more and more sensitive information is transmitted over public networks.

Real-time applications that utilize image steganography include multimedia messaging applications, video conferencing systems, and medical imaging systems. In multimedia messaging applications, image steganography is used to embed sensitive information, such as location data or personal identification information, into images that are sent over public networks. By hiding this information, the risk of interception and unauthorized access to the data is minimized, ensuring the privacy and security of the communication. Video conferencing systems also utilize image steganography to secure the transmission of video streams over the internet. In these systems, sensitive data, such as financial information or personal identification data, can be embedded into the video stream using steganographic schemes. This ensures that the data is protected and cannot be intercepted by unauthorized parties. Additionally, steganography can also be used in medical imaging systems to hide sensitive patient data, such as patient identification information or health records, in medical images. This protects the privacy and security of the patient data and prevents unauthorized access to sensitive medical information.

To ensure a real-time performance, a dedicated IP core is optimized to handle all modulo arithmetic operations for the cryptosystems and schemes used in the encryption, recovery, and embedding modules. In the embedding IP core, the generation of the stego image occurs seamlessly in real-time as the IP core promptly receives the cover image and secret data. Upon reception, the encryption IP core swiftly encrypts the secret data.

Similarly, the recovery IP core generates the RP list once all modified pixels have been registered. In addition, a large set of cover images is pre-stored in the high-speed DDR3 RAM of the AMD Xilinx Zynq-7000 APSoC platform, effectively eliminating any wait time for the IP core to process subsequent cover images. In the PS part of the Zynq-7000 APSoC, two buffers are employed: one for efficient reading of cover images and secret data, and another for seamless writing of stego images to the DDR3 RAM. These buffers are accessed by the implemented IP cores, accelerating the embedding, encryption, and recovery operations within each IP core. The embedding IP core incorporates parallel processing techniques to further enhance the speed of image processing. The encryption IP core benefits from the applied pipelining architecture, enhancing operating speed and overall throughput. In the recovery IP core, sub-pipelining iterative architecture is employed for a balanced resource utilization and operating speed in both EC and Paillier cryptosystem. Overall, these strategies allow for a high-speed, real-time implementation of the steganographic embedded system.

7.9.1 Working using Xilinx Libraries

The implemented embedded system utilizes two primary libraries, namely xilffs and xparameters, both of which were developed by Xilinx. The Xilinx xilffs library is a versatile and efficient file system library that provides essential file management features for embedded systems using Xilinx FPGA and SoC devices. It is based on the widely adopted FatFs file system, which supports FAT12, FAT16, and FAT32 file systems. It is designed to work with different types of storage devices, including SD cards and USB flash drives. The library implements a file system that supports standard file operations, such as opening, reading, writing, and closing files. It also provides features such as directory support, file attributes, and error handling. One of the main advantages of using xilffs is that it provides a consistent and standardized interface for accessing files across different storage devices, which makes it easier to develop and maintain applications that use these devices.

The SD_Init() function is responsible for initializing and mounting the SD card to make it accessible to the system. It performs crucial tasks such as setting up the hardware interface, configuring the communication parameters, and establishing the file system on the SD card. Once the SD card is successfully mounted, it is ready for reading and writing operations. The SD_Init() function is typically called at the beginning of an application to ensure that the SD card is available for use throughout the program's runtime.

The SD_Reject() function is used to unmount the SD card and release any resources associated with it. This function is important for ensuring the proper handling of the file system and preventing potential data corruption or loss. When an application no longer needs access to the SD card, calling the SD_Reject() function allows the system to safely detach the card, perform necessary cleanup operations, and free up resources for other tasks. It is generally called at the end of an application or when the card is to be removed from the system.

The SD_ReadFile() function enables reading data from a text file stored on the SD card. This function takes the file's path, a buffer for storing the read data, and the number of bytes to read as its arguments. It opens the specified file, reads the requested data into the buffer, and closes the file once the operation is complete. The SD_ReadFile() function is essential for applications that need to access stored data, such as configuration settings, sensor readings, or logs. It simplifies the process of retrieving information from the SD card by abstracting the underlying file system operations.

The SD_WriteFile() function allows writing data to a text file on the SD card. It accepts the file's path, the data to be written, and the number of bytes to write

as its arguments. The function opens the specified file (or creates it if it does not exist), writes the provided data, and closes the file once the operation is complete. The SD_WriteFile() function is vital for applications that need to store data persistently, such as logging events, saving configuration changes, or recording sensor data. By providing an easy-to-use interface for writing data to the SD card, this function enables developers to focus on their application's core functionality while the xilffs library handles the complexities of file management.

The xparameters library is a component of the Xilinx SDK (Software Development Kit), which is a software suite for developing embedded applications on Xilinx devices. The xparameters library is a collection of pre-defined constants and data structures that represent the physical hardware components of the device, such as the memory map, interrupt controller, and peripheral devices. These parameters are used by the software application to interact with the hardware components of the device.

The xparameters library simplifies the process of developing software for Xilinx devices by providing a standardized way to access the hardware components. Instead of manually determining the address and configuration of each hardware component, developers can use the xparameters library to retrieve the required information. This reduces the chance of errors and makes it easier to maintain and modify the software application as the hardware configuration changes. The xparameters library is an important tool for developers working with Xilinx devices, and is included in the Xilinx SDK along with other software development tools and libraries.

7.9.2 An Application: Reducing the Storage Space of Digital Images

In this section, the implemented steganographic embedded system is employed in an application that aims to reduce the storage space of digital images. Reducing the storage space of the digital image is done by removing the attached extra information (metadata) from the file of the digital image and concealing it within the image. A digital image file has two parts: image data and metadata. The image data is the visual part of the human vision system in the form of pixels, and the metadata is auxiliary data that describes the image data itself. These metadata are useful for copyright protection, digital image classification via search engines, and image tracking. Image metadata is information inserted into an image in a specific format. A specific format precisely defines the structure of how the pixels, along with the metadata, are arranged.

7.9.2.1 Metadata Formats

Metadata is essentially data about data, and in the case of digital images, it can include information about the camera settings, location data,keywords, hashtags, owner's credits, captions, licensing data, and other details about the image itself. EXIF (Exchangeable Image File Format) is a metadata format that is commonly used in digital photography and is supported by most cameras and image editing software [139]. It allows for the storage of information such as the date and time the image was taken, the camera model and settings, and even GPS location data. XMP (Extensible Metadata Platform), on the other hand, is a more flexible metadata format that can be used with a variety of file types, including images, videos, and documents [140, 141]. It allows for the storage of a wider range of metadata, such as copyright information, keywords, and descriptions, and can also support custom fields for more specific data.

In addition to EXIF and XMP, there are other metadata formats used in different contexts. For example, IPTC (International Press Telecommunications Council) is a metadata format commonly used in the news industry for adding descriptive information to images [142]. It allows for the storage of information such as the photographer's

name, the publication or news agency, and a headline or caption for the image. Another example is Dublin Core [143], which is a widely used metadata format for describing digital resources such as web pages and documents. It includes elements such as title, creator, and subject, and is designed to provide a basic set of metadata that can be used across different domains and applications. Overall, choosing the right metadata format depends on the type of data being described and the context in which it will be used.

Typically, all metadata is located in the header of the image file as an opening tag, and the pixels come next. In the recent years, metadata in image files have become increasingly large in size, with more information being inserted. For example, the EXIF metadata format has more than 400 tags, each with a specific data type (i.e., string, integer, or double) [139].

An experiment on 100,000 top-visited websites was conducted in May 2022 using the HTTP archive crawl and BigQuery magic on the Google Cloud Platform [144]. These websites hosted around 168 GB of digital images. The purpose of this experiment was to find out as to how much metadata space these images consumed in the storage. We extracted and analyzed the metadata from the dataset generated by BigQuery. We found that around 36% of these images contained metadata, each consuming 17% of the total image storage space. This experiment implies that if each website had been visited once, around 10.3 GB of data could have been saved. In such a case, including metadata in these digital images consumes both storage space and network bandwidth.

7.9.2.2 Application Setup

The storage space of the digital image file can be reduced to enhance efficiency in both storage space and network bandwidth by employing the implemented steganographic embedded system. Figure 7.17 shows the general setup of the application of reducing the storage space of digital images using the ZedBoard Zynq-7000 APSoC evaluation board. In the experiment, a MATLAB code is used to extract metadata from a full high definition (FHD) image in red, blue, and green (RGB) channels. The image is stored in the TIFF file format and occupies a storage space of 6.84 MB. The metadata is written into a text file in hex format. The image is written to an SD card, along with its extracted metadata and configuration file. The SD card is inserted into the ZedBoard evaluation board. A Tera terminal emulator is used to establish a communication channel between the board and the host using a USB UART cable.

7.9.2.3 Software Application

The software application that runs on the ZedBoard Zynq-7000 APSoC evaluation board utilizes the xilffs and xparameters libraries. The software application begins by initializing the SD card to read the test image and its extracted metadata. The data is read using the SD_Read() function and is reformatted into two arrays (buffers), one for the pixels of the cover image and the other for the extracted metadata. After checking the configurations, if encrypting the metadata is required, the data is sent to to the encryption IP core which is accessed by its driver which is generated during the integration phase. The encrypted data is forwarded to the embedding IP core. The embedding IP core processes the received data in the input FIFO and writes the modified pixels into the output FIFO. If recovering the modified pixels is required, the recovery IP core is started to register the pixels by either the EC or Paillier cryptosystem. Once the output FIFO is full, the data is written to the SD card using the SD_Write() function along with the RP list. The process continues until all extracted metadata is concealed.

After running the application on the Z-7020 APSoC device, an image is generated and written back to the SD card. The generated image has a storage space of 5.97 MB,

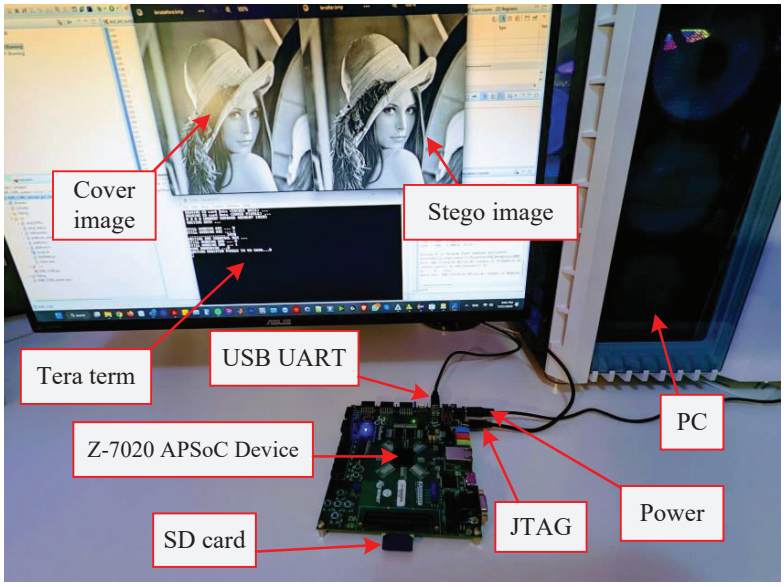


Figure 7.17: General setup of the application for reducing the storage space of digital images.

which is 0.9 MB smaller than the original image when the configurations are $EM =' 1'$ and $RM =' 00'$. In addition, it takes 0.022 seconds to conceal the encrypted metadata and write the image back to the SD card. Table 7.8 presents the performance of our design in terms of embedding rate, PSNR, and processing rate (FPS) compared to that of a number of other designs. It is seen from this table that our design outperforms the other designs, regardless of the format or resolution used for the images.

Table 7.8: Results of performance comparison in terms of the embedding rate, image quality, and processing rate.

Scheme	FPGA device	Image size	Encryption	Embedding rate (bpp)	Image quality PSNR (dB)	Processing rate (FPS)
IWT-based [5] (2014)	Cyclone II	128 × 128 (RGB)	No	2.00	51.53	-
Our design ($L = 9$)	Artix-7	128 × 128 (RGB)	Yes	**2.07**	**84.87**	**5627.69**
Our design ($L = 9$)	Artix-7	256 × 256 (RGB)	Yes	**4.14**	**84.23**	**1409.79**
Chaotic-based [61] (2022)	Cyclone IV	512 × 512 (RGB)	No	2.00	53.24	-
Our design ($L = 9$)	Artix-7	512 × 512 (RGB)	Yes	**8.28**	**83.54**	**351.55**
Our design ($L = 9$)	Artix-7	1920 × 1080 (RGB)	Yes	**16.56**	**83.86**	**44.43**
Multilevel LSB [56] (2017)	Virtex- II Pro	256 × 256 (Gray level)	No	1.00	77.84	183.48
Our design ($L = 9$)	Artix-7	256 × 256 (Gray level)	Yes	**1.38**	**82.66**	**4218.62**
EMD-based [55] (2019)	Artix-7	512 × 512 (Gray level)	No	1.16	45.36	549.00
Lifting-based [57] (2019)	Artix-7	512 × 512 (Gray level)	No	0.004	34.79	75.00
Threshold-based [59] (2021)	Virtex-6	512 × 512 (Gray level)	No	0.25	44.43	-
Parallelism-based [58] (2021)	Spartan-6	512 × 512 (Gray level)	No	0.26	42.99	-
LSB matching [60] (2021)	Spartan-3A	512 × 512 (Gray level)	No	1.00	51.37	-
Our design ($L = 9$)	Artix-7	512 × 512 (Gray level)	Yes	**2.76**	**81.92**	**1054.07**
Our design ($L = 9$)	Artix-7	1920 × 1080 (Gray level)	Yes	**5.52**	**82.55**	**133.25**

The application is able to save around 1 GB of data when 1000 RGB FHD test images (totaling 6.84 GB) are used. It takes 22 seconds to conceal the metadata in the images and write the storage-space-reduced images back to the SD card. In terms of frame per second (FPS), our real-time application can process a video at 44.43 FPS or a file with a storage space of 16.58 GB for a 1-minute RGB FHD video.

7.10 Summary

In this chapter, the secure lossless steganographic embedded system is implemented on the AMD Xilinx Zynq-7000 APSoC platform. The embedded system combines

hardware and software design approach, with a processing system using an ARM-based processor and an FPGA device. The system includes three IP cores: the embedding, encryption, and recovery. In the encryption IP core, the 128-bit AES cryptostyem is implemented. The embedding IP core has the hardware implementation of the modulus-based steganographic scheme. Finally, the recovery IP core has the implementations of two public-key cryptosystems, EC and Pailler. Place and route results have shown that all IP cores requires minimal resources, low power, high speed, and high throughput. The compact design of the embedded system makes it suitable for use in embedded systems and real-time processing engines, such as digital copyright control, secret communication, image search engine robustness, and feature tagging.

Chapter 8

Conclusion and Future Work

8.1 Concluding Remarks

This thesis consists of several chapters that explore various aspects of image steganography, from existing techniques and their limitations to proposing a scheme that aims to overcome these limitations and improve the performance of image steganography. Additionally, the thesis investigates the use of AES and ECC cryptosystems and their hardware implementations. Finally, a reconfigurable embedded system is presented that combines these various techniques to create a secure and lossless steganographic embedded system. The following summarizes the chapters in this thesis:

Chapter 1 provides an introduction to the concept of image steganography, outlines the objectives and motivations of the thesis, and defines the research contributions of the thesis by proposing a secure lossless model to overcome specific challenges. An overview of the subsequent chapters is also provided.

Chapter 2 provides a comprehensive review of image steganography, including framework components and fundamental schemes. Recent steganographic scheme performance is analyzed, with remarks and exploration of other schemes. Private and public-key cryptosystems and their main operations are also introduced. The chapter concludes with a discussion of the AMD Xilinx Zynq-7000 APSoC platform and its components.

Chapter 3 presents a novel modulus-based image steganographic scheme that combines the EMD and DE schemes. In the proposed scheme, the secret data block remains undivided. Instead, a lookup table is utilized to locate the corresponding four secret digits of the secret data block, and these four secret digits are concealed by the EMD and DE schemes as follows. Two digits are concealed using the EMD scheme, and the other two are concealed using the DE scheme.

In Chapter 4, another novel modulus-based image steganographic scheme is proposed. In the proposed scheme, the secret data block is divided into two parts. The EMD scheme is used to conceal the first part, while the DE scheme is employed for the second part.

These proposed modulus-based image steganographic schemes, whether involving divided or undivided secret data blocks, provide superior performance compared to the EMD, DE, and recent schemes. They demonstrate high embedding rates, high image qualities, and offer resistance against steganalysis attacks.

Chapter 5 introduces the hardware design of the modulus-based steganographic scheme proposed in Chapter 3. Chapter 6 introduces the hardware design of the modulus-based steganographic scheme proposed in Chapter 4. The proposed hardware architectures in these designs are deployed on the Zynq-7000 APSoC platform, and they incorporates resource sharing, pipelining, and parallel processing techniques. The

implemented scheme in Chapter 5 provides high operating speeds, high throughput, and low power consumption. On the other hand, The implemented scheme in Chapter 6 provides higher operating speeds, higher throughput, and lower power consumption than the hardware implementation in Chapter 5.

Chapter 7 presents the implementation of the proposed secure lossless steganographic embedded system on the ZedBoard Zynq-7000 board, which features the AMD Xilinx Zynq-7000 APSoC platform. The system includes the three IP cores: embedding, encryption, and recovery. The embedded system combines the hardware and software design approach, with the processing system using the ARM-based processor and the programmable logic using the FPGA device. The encryption IP core employs a private-key cryptosystem to encrypt 128-bit consecutive blocks of secret data. In the recovery IP core, two public-key cryptosystems are used to restore all original values of the pixels from the resulting stego, making the model lossless. The embedding IP core in the system adopts the implemented scheme introduced in Chapter 6. The embedding IP core provides high embedding rates, PSNR values, and operating speeds with low power consumption. The compact design of the embedded system makes it suitable for real-time processing applications, such as digital copyright control and feature tagging.

8.2 Future Work

While the proposed secure lossless steganographic embedded system presented in this thesis offers a robust solution to several challenges in image steganography, there are still areas that can be further explored and improved upon. In this section, we discuss potential areas for future research and development that can enhance the performance and applicability of the system.

1. Improved steganographic scheme: The proposed steganographic schemes in Chapter 3 and Chapter 4 have demonstrated high performance, but there is still room for improvement. Future work could explore new embedding techniques, such as deep learning-based approaches, to further enhance the embedding capacity and quality of the stego images.

2. Advanced machine learning-based steganalysis: The proposed steganographic schemes in Chapter 3 and Chapter 4 are resistant to a specific classifier in the machine learning-based anti-detection test, but it may not withstand advanced machine learning classifiers. Future work could investigate the use of advanced machine learning-based anti-detection techniques with a focus on fine-tuning for better classification accuracy.

3. Application to video steganography: The proposed steganographic schemes in this thesis focuses on image steganography. Future work could explore the extension of the proposed schemes to video steganography. This would involve addressing the unique challenges of video steganography, such as the need for temporal coherence and synchronization between frames.

4. Enhanced performance embedded system: The performance of the steganographic embedded system in Chapter 7 can be improved in terms of throughput by performing the concealing and extracting operations in a high data bus of 128-bit. Using the AXI4-Stream bus can improve the proposed architecture's performance by enabling fast transfer of secret data in bursts per address and providing efficient data transfer and control.

5. Power optimization: Another potential area of improvement is power optimization in the embedded system, which can be achieved by implementing techniques such as clock gating and power gating to reduce power consumption without compromising performance.

6. Hardware acceleration for the entire system: The hardware implementations for the individual modules were presented in Chapters 5, 6 and 7, the system as a whole could benefit from hardware acceleration. Future work could investigate the use of hardware acceleration techniques, such as FPGA-based accelerators, to improve the overall system performance.

Overall, the proposed secure lossless steganographic embedded system presented in this thesis lays a solid foundation for future research in the field. The system could be further improved to enhance its performance, robustness, and applicability to different domains.

References

[1] W. Stallings, *Cryptography and network security :principles and practice*, 5th ed. Prentice Hall, 2011.

[2] F. Y. Shih, *Digital Watermarking and Steganography: Fundamentals and Techniques*, 1st ed. CRC Press, Inc., 2007.

[3] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, April 2004.

[4] M. Hussain, A. W. A. Wahab, Y. I. B. Idris, A. T. S. Ho, and K.-H. Jung, "Image steganography in spatial domain: A survey," *Signal Processing: Image Communication*, vol. 65, pp. 46–66, July 2018.

[5] B. Ramalingam, R. Amirtharajan, and J. B. B. Rayappan, "Stego on FPGA: An IWT Approach," *The Scientific World Journal*, vol. 2014, p. e192512, Febuary 2014.

[6] P. C. Mandal, I. Mukherjee, G. Paul, and B. N. Chatterji, "Digital image steganography: A literature survey," *Information Sciences*, vol. 609, pp. 1451–1488, September 2022.

[7] A. Sukumar, V. Subramaniyaswamy, L. Ravi, V. Vijayakumar, and V. Indragandhi, "Robust image steganography approach based on RIWT-Laplacian pyramid and histogram shifting using deep learning," *Multimedia Systems*, vol. 27, no. 4, pp. 651–666, August 2021.

[8] X. Liao, K. Li, and J. Yin, "Separable data hiding in encrypted image based on compressive sensing and discrete fourier transform," *Multimedia Tools and Applications*, vol. 76, no. 20, pp. 20 739–20 753, October 2017.

[9] T. Rabie and I. Kamel, "High-capacity steganography: a global-adaptive-region discrete cosine transform approach," *Multimedia Tools and Applications*, vol. 76, no. 5, pp. 6473–6493, March 2017.

[10] V. Kumar and D. Kumar, "A modified DWT-based image steganography technique," *Multimedia Tools and Applications*, vol. 77, no. 11, pp. 13 279–13 308, June 2018.

[11] M. Fakhredanesh, M. Rahmati, and R. Safabakhsh, "Steganography in discrete wavelet transform based on human visual system and cover model," *Multimedia Tools and Applications*, vol. 78, no. 13, pp. 18 475–18 502, July 2019.

[12] K.-H. Jung and K.-Y. Yoo, "Data hiding method using image interpolation," *Computer Standards & Interfaces*, vol. 31, no. 2, pp. 465–470, February 2009.

[13] C.-F. Lee and Y.-L. Huang, "An efficient image interpolation increasing payload in reversible data hiding," *Expert Systems with Applications*, vol. 39, no. 8, pp. 6712–6719, June 2012.

[14] J. Hu and T. Li, "Reversible steganography using extended image interpolation technique," *Computers & Electrical Engineering*, vol. 46, pp. 447–455, August 2015.

[15] X. Zhang, Z. Sun, Z. Tang, C. Yu, and X. Wang, "High capacity data hiding based on interpolated image," *Multimedia Tools and Applications*, vol. 76, no. 7, pp. 9195–9218, April 2017.

[16] T.-C. Lu, "Interpolation-based hiding scheme using the modulus function and re-encoding strategy," *Signal Processing*, vol. 142, pp. 244–259, January 2018.

[17] M. A. Wahed and H. Nyeem, "Reversible data hiding with interpolation and adaptive embedding," *Multimedia Tools and Applications*, vol. 78, no. 8, pp. 10 795–10 819, April 2019.

[18] M. A. Wahed and H. Nyeem, "High capacity reversible data hiding with interpolation and adaptive embedding," *PLOS ONE*, vol. 14, no. 3, p. e0212093, March 2019.

[19] A. Shaik and T. V, "High capacity reversible data hiding using 2D parabolic interpolation," *Multimedia Tools and Applications*, vol. 78, no. 8, pp. 9717–9735, April 2019.

[20] A. Malik, G. Sikka, and H. K. Verma, "A Reversible Data Hiding Scheme for Interpolated Images Based on Pixel Intensity Range," *Multimedia Tools and Applications*, vol. 79, no. 25, pp. 18 005–18 031, July 2020.

[21] F. S. Hassan and A. Gutub, "Novel embedding secrecy within images utilizing an improved interpolation-based reversible data hiding scheme," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, pp. 2017–2030, July 2020.

[22] Y.-q. Chen, W.-j. Sun, L.-y. Li, C.-C. Chang, and X. Wang, "An efficient general data hiding scheme based on image interpolation," *Journal of Information Security and Applications*, vol. 54, p. 102584, October 2020.

[23] F. S. Hassan and A. Gutub, "Efficient Image Reversible Data Hiding Technique Based on Interpolation Optimization," *Arabian Journal for Science and Engineering*, vol. 46, no. 9, pp. 8441–8456, September 2021.

[24] X. Zhang, J. Long, Z. Wang, and H. Cheng, "Lossless and Reversible Data Hiding in Encrypted Images With Public-Key Cryptography," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 9, pp. 1622–1631, September 2016.

[25] M. Shah, W. Zhang, H. Hu, H. Zhou, and T. Mahmood, "Homomorphic Encryption-Based Reversible Data Hiding for 3D Mesh Models," *Arabian Journal for Science and Engineering*, vol. 43, no. 12, pp. 8145–8157, December 2018.

[26] L. Xiong and D. Dong, "Reversible data hiding in encrypted images with somewhat homomorphic encryption based on sorting block-level prediction-error expansion," *Journal of Information Security and Applications*, vol. 47, pp. 78–85, August 2019.

[27] Y. Fu, P. Kong, H. Yao, Z. Tang, and C. Qin, "Effective reversible data hiding in encrypted image with adaptive encoding strategy," *Information Sciences*, vol. 494, pp. 21–36, August 2019.

[28] Y.-C. Chen, T.-H. Hung, S.-H. Hsieh, and C.-W. Shiu, "A New Reversible Data Hiding in Encrypted Image Based on Multi-Secret Sharing and Lightweight Cryptographic Algorithms," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 12, pp. 3332–3343, December 2019.

[29] Y. Yang, X. Xiao, X. Cai, and W. Zhang, "A Secure and High Visual-Quality Framework for Medical Images by Contrast-Enhancement Reversible Data Hiding and Homomorphic Encryption," *IEEE Access*, vol. 7, pp. 96 900–96 911, July 2019.

[30] R. Zhang, C. Lu, and J. Liu, "A high capacity reversible data hiding scheme for encrypted covers based on histogram shifting," *Journal of Information Security and Applications*, vol. 47, pp. 199–207, August 2019.

[31] Y. Qiu, Q. Ying, K. Lv, Z. Qian, N. Tang, and X. Zhang, "Improved Scheme for Reversible Data Hiding for Encrypted Images using Interpolation Technique," in *Proc. IEEE International Conference on Signal, Information and Data Processing (ICSIDP)*, December 2019, pp. 1–7.

[32] Z. Yin, Y. Xiang, and X. Zhang, "Reversible Data Hiding in Encrypted Images Based on Multi-MSB Prediction and Huffman Coding," *IEEE Transactions on Multimedia*, vol. 22, no. 4, pp. 874–884, April 2020.

[33] D. Huang and J. Wang, "High-capacity reversible data hiding in encrypted image based on specific encryption process," *Signal Processing: Image Communication*, vol. 80, p. 115632, February 2020.

[34] S. Ayyappan, C. Lakshmi, and V. Menon, "A Secure Reversible Data Hiding and Encryption System for Embedding EPR in Medical Images," *Current Signal Transduction Therapy*, vol. 15, no. 2, pp. 124–135, August 2020.

[35] F. Wu, X. Zhou, Z. Chen, and B. Yang, "A reversible data hiding scheme for encrypted images with pixel difference encoding," *Knowledge-Based Systems*, vol. 234, p. 107583, December 2021.

[36] V. M. Manikandan and Y.-D. Zhang, "An adaptive pixel mapping based approach for reversible data hiding in encrypted images," *Signal Processing: Image Communication*, vol. 105, p. 116690, July 2022.

[37] X. Zhang and S. Wang, "Efficient Steganographic Embedding by Exploiting Modification Direction," *IEEE Communications Letters*, vol. 10, no. 11, pp. 781–783, November 2006.

[38] K.-H. Jung and K.-Y. Yoo, "Improved Exploiting Modification Direction Method by Modulus Operation," *International Journal of Signal Processing, Image Processing and Pattern*, vol. 2, p. 11, March 2009.

[39] R.-M. Chao, H.-C. Wu, C.-C. Lee, and Y.-P. Chu, "A Novel Image Data Hiding Scheme with Diamond Encoding," *EURASIP Journal on Information Security*, vol. 2009, no. 1, p. 658047, May 2009.

[40] W.-C. Kuo, C.-C. Wang, and H.-C. Hou, "Signed digit data hiding scheme," *Information Processing Letters*, vol. 116, no. 2, pp. 183–191, February 2016.

[41] Z. Li and Y. He, "Steganography with pixel-value differencing and modulus function based on PSO," *Journal of Information Security and Applications*, vol. 43, pp. 47–52, December 2018.

[42] Z. S. Younus and M. K. Hussain, "Image steganography using exploiting modification direction for compressed encrypted data," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, pp. 2951–2963, April 2019.

[43] Y.-X. Liu, C.-N. Yang, Q.-D. Sun, S.-Y. Wu, S.-S. Lin, and Y.-S. Chou, "Enhanced embedding capacity for the SMSD-based data-hiding method," *Signal Processing: Image Communication*, vol. 78, pp. 216–222, October 2019.

[44] R. Kumar and K.-H. Jung, "Robust reversible data hiding scheme based on two-layer embedding strategy," *Information Sciences*, vol. 512, pp. 96–107, February 2020.

[45] S. Saha, A. Chakraborty, A. Chatterjee, S. Dhargupta, S. K. Ghosal, and R. Sarkar, "Extended exploiting modification direction based steganography using hashed-weightage Array," *Multimedia Tools and Applications*, vol. 79, no. 29, pp. 20 973–20 993, August 2020.

[46] F. Peng, Y. Zhao, X. Zhang, M. Long, and W.-q. Pan, "Reversible data hiding based on RSBEMD coding and adaptive multi-segment left and right histogram shifting," *Signal Processing: Image Communication*, vol. 81, p. 115715, February 2020.

[47] H.-S. Leng, J.-F. Lee, and H.-W. Tseng, "A high payload EMD-based steganographic method using two extraction functions," *Digital Signal Processing*, vol. 113, p. 103026, June 2021.

[48] R. Tavares and F. Madeiro, "Word-Hunt: A LSB Steganography Method with Low Expected Number of Modifications per Pixel," *IEEE Latin America Transactions*, vol. 14, no. 2, pp. 1058–1064, Febuary 2016.

[49] A. K. Sahu and G. Swain, "Reversible Image Steganography Using Dual-Layer LSB Matching," *Sensing and Imaging*, vol. 21, no. 1, p. 1, December 2019.

[50] R. Dumre and A. Dave, "Exploring LSB Steganography Possibilities in RGB Images," in *Proc. 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, July 2021, pp. 1–7.

[51] N. M. (Ganguly), G. Paul, S. K. Saha, and D. Burman, "A PVD based high capacity steganography algorithm with embedding in non-sequential position," *Multimedia Tools and Applications*, vol. 79, no. 19, pp. 13 449–13 479, May 2020.

[52] M. Sahu, N. Padhy, and S. S. Gantayat, "Multi-directional PVD steganography avoiding PDH and boundary issue," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 10, Part A, pp. 8838–8851, November 2022.

[53] B. Schneier, *Applied cryptography: protocols, algorithms, and source code in C*. Wiley-India, 2007.

[54] A. J. Menezes, P. C. v. Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Boca Raton: CRC Press, May 2020.

[55] K. S. Shet, A. R. Aswath, M. C. Hanumantharaju, and X.-Z. Gao, "Novel high-speed reconfigurable FPGA architectures for EMD-based image steganography," *Multimedia Tools and Applications*, vol. 78, no. 13, pp. 18 309–18 338, July 2019.

[56] K. Sathish Shet, A. R. Aswath, M. C. Hanumantharaju, and X.-Z. Gao, "Design and development of new reconfigurable architectures for LSB/multi-bit image steganography system," *Multimedia Tools and Applications*, vol. 76, no. 11, pp. 13 197–13 219, June 2017.

[57] A. Phadikar, G. K. Maity, T.-L. Chiu, and H. Mandal, "FPGA Implementation of Lifting-Based Data Hiding Scheme for Efficient Quality Access Control of Images," *Circuits, Systems, and Signal Processing*, vol. 38, no. 2, pp. 847–873, February 2019.

[58] S. H. Seyed Dizaji, M. Zolfy Lighvan, and A. Sadeghi, "Hardware-Based Parallelism Scheme for Image Steganography Speed up," in *Proc. International Conference on Innovative Computing and Communications*, ser. Advances in Intelligent Systems and Computing. Singapore: Springer, July 2021, pp. 225–236.

[59] J. Wei, Z. Quan, Y. Hu, J. Liu, H. Zhang, and M. Liu, "Implementing a Low-Complexity Steganography System on FPGA," in *Proc. 9th International Conference on Intelligent Computing and Wireless Optical Communications (ICWOC)*, June 2021, pp. 64–68.

[60] S. Sinha Roy, A. Basu, A. Chattopadhyay, and T. S. Das, "Implementation of image copyright protection tool using hardware-software co-simulation," *Multimedia Tools and Applications*, vol. 80, no. 3, pp. 4263–4277, January 2021.

[61] J.-y. Sun, H. Cai, Z.-b. Gao, C.-p. Wang, and H. Zhang, "A novel non-equilibrium hyperchaotic system and application on color image steganography with FPGA implementation," *Nonlinear Dynamics*, vol. 111, no. 4, pp. 3851–3868, Febuary 2023.

[62] S. Hussain, N. Sheybani, P. Neekhara, X. Zhang, J. Duarte, and F. Koushanfar, "Faststamp: Accelerating neural steganography and digital watermarking of images on fpgas," in *Proc. 41st IEEE/ACM International Conference on Computer-Aided Design*, ser. ICCAD '22, no. 41, October 2022, pp. 1–9.

[63] M. Martelli, M. Dang, and T. Seph, "Defining Chaos," *Mathematics Magazine*, vol. 71, no. 2, pp. 112–122, April 1998.

[64] Jui-Cheng and J.-I. Guo, "A new chaotic key-based design for image encryption and decryption," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 4, May 2000, pp. 49–52.

[65] D. Qi, J. Zou, and X. Han, "A new class of scrambling transformation and its application in the image information covering," *Science in China Series E-Technological Sciences*, vol. 43, no. 3, pp. 304–312, June 2000.

[66] S. K.u. and A. Mohamed, "Novel hyper chaotic color image encryption based on pixel and bit level scrambling with diffusion," *Signal Processing: Image Communication*, vol. 99, p. 116495, November 2021.

[67] N. F. Pub, "197: Advanced encryption standard (AES)," *Federal information processing standards publication*, vol. 197, no. 441, p. 0311, 2001.

[68] V. S. Miller, "Use of elliptic curves in cryptography," in *Proc. Conference on the theory and application of cryptographic techniques*. Springer, 1985, pp. 417–426.

[69] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 1999, pp. 223–238.

[70] R. M. May, "Simple mathematical models with very complicated dynamics," *Nature*, vol. 261, no. 5560, pp. 459–467, June 1976.

[71] M. B. Yassein, S. Aljawarneh, E. Qawasmeh, W. Mardini, and Y. Khamayseh, "Comprehensive study of symmetric key and asymmetric key encryption algorithms," in *Proc. International Conference on Engineering and Technology (ICET)*, August 2017, pp. 1–7.

[72] E. Wenger and M. Hutter, "Exploring the design space of prime field vs. binary field ECC-hardware implementations," in *Proc. Nordic Conference on Secure IT Systems*. Springer, 2011, pp. 256–271.

[73] P. Gallagher, "Digital signature standard (DSS)," *Federal Information Processing Standards Publications, volume FIPS*, pp. 186–183, 2013.

[74] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proc. Annual International Cryptology Conference*. Springer, 1999, pp. 388–397.

[75] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, July 1985.

[76] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, February 1978.

[77] D. McGrew, K. Igoe, and M. Salter, "Fundamental elliptic curve cryptography algorithms," Internet Engineering Task Force (IETF), Tech. Rep. 2018, 2011. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6090.txt.

[78] I. International Organization for Standardization (ISO), *Cryptographic techniques based on elliptic curves*, August 2017, vol. 20. [Online]. Available: https://www.iso.org/standard/69726.html

[79] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to elliptic curve cryptography*. Springer Science and Business Media, 2006.

[80] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, November 1976.

[81] Y. Desmedt and A. M. Odlyzko, "A chosen text attack on the rsa cryptosystem and some discrete logarithm schemes," in *Proc. Conference on the Theory and Application of Cryptographic Techniques*. Springer, January 1985, pp. 516–522.

[82] E. Kiltz, A. O'Neill, and A. Smith, "Instantiability of RSA-OAEP Under Chosen-Plaintext Attack," *Journal of Cryptology*, vol. 30, no. 3, pp. 889–919, July 2017.

[83] G. Goos, J. Hartmanis, J. van Leeuwen, J.-S. Coron, D. Naccache, and J. P. Stern, "On the Security of RSA Padding," in *Advances in Cryptology — CRYPTO' 99*. Berlin, Heidelberg: Springer Berlin Heidelberg, December 1999, vol. 1666, pp. 1–18.

[84] X. Yi, R. Paulet, and E. Bertino, *Homomorphic Encryption and Applications*, ser. Springer Briefs in Computer Science. Springer International Publishing, January 2014. [Online]. Available: https://www.springer.com/gp/book/9783319122281

[85] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A Survey on Homomorphic Encryption Schemes: Theory and Implementation," *ACM Computing Surveys*, vol. 51, October 2017.

[86] S. Biookaghazadeh, M. Zhao, and F. Ren, "Are FPGAs Suitable for Edge Computing?" *In USENIX Workshop on Hot Topics in Edge Computing*, January 2018.

[87] *Zynq-7000 SoC - Design Overview*, Accessed 07 March 2022, available at: https://www.xilinx.com/support/documentation-navigation/design-hubs/dh0050-zynq-7000-design-overview-hub.html.

[88] *AMBA AXI4 Interface Protocol*, Accessed 17 July 2022, available at: https://www.xilinx.com/products/intellectual-property/axi.html.

[89] *Xilinx: 7 Series FPGAs Data Sheet: Overview (DS180)*, Accessed 03 April 2022, available at: https://docs.xilinx.com/v/u/en-US/ds180_7Series_Overview.

[90] *An introduction to AMBA AXI*, Accessed 04 July 2022, available at: https://developer.arm.com/documentation/102202/0200/AXI-protocol-overview.

[91] M. Qasaimeh, K. Denolf, J. Lo, K. Vissers, J. Zambreno, and P. H. Jones, "Comparing Energy Efficiency of CPU, GPU and FPGA Implementations for Vision Kernels," in *Proc. IEEE International Conference on Embedded Software and Systems (ICESS)*, June 2019, pp. 1–8.

[92] W. Diehl, F. Farahmand, P. Yalla, J.-P. Kaps, and K. Gaj, "Comparison of hardware and software implementations of selected lightweight block ciphers," in *Proc. 27th International Conference on Field Programmable Logic and Applications (FPL)*, September 2017, pp. 1–4.

[93] S. Kilts, *Advanced FPGA design: architecture, implementation, and optimization*. John Wiley & Sons, 2007.

[94] S. Harb, M. O. Ahmad, and M. N. S. Swamy, "An efficient image steganographic scheme for a real-time embedded system and its hardware implementation on amd xilinx zynq-7000 apsoc platform," *Journal of Real-Time Image Processing*, vol. 20, no. 3, p. 46, April 2023.

[95] J. Fridrich, *Steganography in Digital Media: Principles, Algorithms, and Applications*. Cambridge: Cambridge University Press, 2009.

[96] "Beyond binary classification," in *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*, P. Flach, Ed. Cambridge: Cambridge University Press, 2012, pp. 81–103.

[97] J. Fridrich and J. Kodovsky, "Rich models for steganalysis of digital images," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 868–882, May 2012.

[98] S. Harb, M. O. Ahmad, and M. N. S. Swamy, "A novel modulus-based steganographic scheme for improved stego image quality and high embedding rate," *under review by the Journal of Signal Processing: Image Communication*.

[99] S. Shivani, S. C. Patel, V. Arora, B. Sharma, A. Jolfaei, and G. Srivastava, "Real-time cheating immune secret sharing for remote sensing images," vol. 18, no. 5, pp. 1493–1508, October 2021.

[100] S. Tyagi, P. Anurag, and S. N. Pai, "Multilevel Steganography for Data Protection," in *Proc. Ambient Communications and Computer Systems*, ser. Advances in Intelligent Systems and Computing, Y.-C. Hu, S. Tiwari, K. K. Mishra, and M. C. Trivedi, Eds. Springer, 2019, pp. 461–472.

[101] P. Nawrocki, J. Pajor, B. Sniezynski, and J. Kolodziej, "Modeling adaptive security-aware task allocation in mobile cloud computing," *Simulation Modelling Practice and Theory*, vol. 116, p. 102491, April 2022.

[102] A. Alarood, N. Ababneh, M. Al-Khasawneh, M. Rawashdeh, and M. Al-Omari, "IoTSteg: ensuring privacy and authenticity in internet of things networks using weighted pixels classification based image steganography," *Cluster Computing*, vol. 25, no. 3, pp. 1607–1618, June 2022.

[103] Z. Qu, Z. Cheng, W. Liu, and X. Wang, "A novel quantum image steganography algorithm based on exploiting modification direction," *Multimedia Tools and Applications*, vol. 78, no. 7, pp. 7981–8001, April 2019.

[104] O. Elharrouss, N. Almaadeed, and S. Al-Maadeed, "An image steganography approach based on k-least significant bits (k-LSB)," in *Proc. IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, Febuary 2020, pp. 131–135.

[105] A. HajiRassouliha, A. J. Taberner, M. P. Nash, and P. M. F. Nielsen, "Suitability of recent hardware accelerators (DSPs, FPGAs, and GPUs) for computer vision and image processing algorithms," *Signal Processing: Image Communication*, vol. 68, pp. 101–119, October 2018.

[106] D. R. Menaka, D. R. Janarthanan, and D. K. Deeba, "FPGA implementation of low power and high speed image edge detection algorithm," *Microprocessors and Microsystems*, vol. 75, p. 103053, June 2020.

[107] X. Wang, C. Li, and J. Song, "Motion image processing system based on multi core FPGA processor and convolutional neural Network," *Microprocessors and Microsystems*, vol. 82, p. 103923, April 2021.

[108] F. Spagnolo, S. Perri, and P. Corsonello, "Design of a real-time face detection architecture for heterogeneous systems-on-chips," *Integration*, vol. 74, pp. 1–10, September 2020.

[109] S. Sinha, N. K. Goyal, and R. Mall, "Reliability and availability prediction of embedded systems based on environment modeling and simulation," *Simulation Modelling Practice and Theory*, vol. 108, p. 102246, April 2021.

[110] S. Harb, M. O. Ahmad, and M. N. S. Swamy, "A high-speed fpga implementation of aes for large scale embedded systems and its applications," in *Proc. 13th International Conference on Information and Communication Systems (ICICS)*, July 2022, pp. 59–64.

[111] S. Harb, M. O. Ahmad, and M. N. S. Swamy, "High-performance pipelined fpga implementation of the elliptic curve cryptography over gf (2n)," in *Proc. International Conference on Security and Cryptography (SECRYPT)*, vol. 2, August 2019, pp. 15–24.

[112] S. Harb, M. O. Ahmad, and M. N. S. Swamy, "A reconfigurable implementation of elliptic curve cryptography over gf (2n)," *E-Business and Telecommunications*, vol. 1247, pp. 87–107, August 2019.

[113] S. Harb, M. O. Ahmad, and M. N. S. Swamy, "Design and hardware implementation of a separable image steganographic scheme using public-key cryptosystem," in *Proc. 9th International Conference on Cryptography and Information Security (CRYPIS 2020)*, vol. 10, May 2020, pp. 227–240.

[114] *ZedBoard Zynq-7000 ARM/FPGA SoC Development Board - Digilent*, Accessed 09 April 2022, available at: https://digilent.com/shop/zedboard-zynq-7000-arm-fpga-soc-development-board/.

[115] V. Rijmen, "Efficient implementation of the rijndael s-box," *Katholieke Universiteit Leuven, Dept. ESAT. Belgium*, 2000.

[116] J.-P. Deschamps, J. L. Imana, and G. D. Sutter, *Hardware implementation of finite-field arithmetic*. McGraw-Hill Education, 2009.

[117] *Vivado ML Overview*, Accessed 04 May 2022, available at: https://www.xilinx.com/products/design-tools/vivado.html.

[118] S. Chen, W. Hu, and Z. Li, "High Performance Data Encryption with AES Implementation on FPGA," in *Proc. IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity)*, May 2019, pp. 149–153.

[119] C. Arul Murugan, P. Karthigaikumar, and S. Sathya Priya, "FPGA implementation of hardware architecture with AES encryptor using sub-pipelined S-box techniques for compact applications," *Automatika*, vol. 61, no. 4, pp. 682–693, October 2020.

[120] R. P. and M. H., "Design and implementation of power and area optimized AES architecture on FPGA for IoT application," *Circuit World*, vol. 47, no. 2, pp. 153–163, January 2020.

[121] S. J. H. Pirzada, M. N. Hasan, Z. W. Memon, M. Haris, T. Xu, and L. Jianwei, "High-Throughput Optimizations of AES Algorithm for Satellites," in *Proc. International Symposium on Recent Advances in Electrical Engineering & Computer Sciences (RAEE & CS)*, vol. 5, October 2020, pp. 1–6.

[122] T. M. Kumar, K. S. Reddy, S. Rinaldi, B. D. Parameshachari, and K. Arunachalam, "A Low Area High Speed FPGA Implementation of AES Architecture for Cryptography Application," *Electronics*, vol. 10, no. 16, p. 2023, January 2021.

[123] S. Harb and M. Jarrah, "Fpga implementation of the ecc over gf (2 m) for small embedded applications," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 18, no. 2, p. 17, March 2019.

[124] S. Moon, "A binary redundant scalar point multiplication in secure elliptic curve cryptosystems." *IJ Network Security*, vol. 3, no. 2, pp. 132–137, January 2006.

[125] P. L. Montgomery, "Speeding the pollard and elliptic curve methods of factorization," *Mathematics of computation*, vol. 48, no. 177, pp. 243–264, January 1987.

[126] J. López and R. Dahab, "Improved algorithms for elliptic curve arithmetic in gf (2 n)," in *Proc. International Workshop on Selected Areas in Cryptography*. Springer, January 1998, pp. 201–212.

[127] I. Xilinx, "Ise in-depth tutorial, complete guide (ug695)," Xilinx, Inc., Tech. Rep. 14.1, April 24, 2012.

[128] B. Rashidi, S. M. Sayedi, and R. R. Farashahi, "High-speed hardware architecture of scalar multiplication for binary elliptic curve cryptosystems," *Microelectronics Journal*, vol. 52, pp. 49–65, June 2016.

[129] Z. U. Khan and M. Benaissa, "Throughput/area-efficient ecc processor using montgomery point multiplication on fpga," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, no. 11, pp. 1078–1082, July 2015.

[130] L. Li and S. Li, "High-performance pipelined architecture of elliptic curve scalar multiplication over gf (2^m)," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 4, pp. 1223–1232, August 2016.

[131] G. D. Sutter, J.-P. Deschamps, and J. L. Imaña, "Efficient elliptic curve point multiplication using digit-serial binary field operations," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 1, pp. 217–225, January 2013.

[132] M. S. Hossain, E. Saeedi, and Y. Kong, "High-speed, area-efficient, fpga-based elliptic curve cryptographic processor over nist binary fields," in *Proc. Data Science and Data Intensive Systems (DSDIS), 2015 IEEE International Conference on*. IEEE, February 2015, pp. 175–181.

[133] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, January 1999, pp. 223–238.

[134] W. A. P. Laces and J. J. G. Hernandez, "Fpga implementation of a low complexity steganographic system for digital images," in *Proc. 14th International Conference on Computer and Information Science (ICIS)*. IEEE, June 2015, pp. 319–324.

[135] H. K. Maity and S. P. Maity, "Fpga implementation of reversible watermarking in digital images using reversible contrast mapping," *Journal of Systems and Software*, vol. 96, pp. 93–104, October 2014.

[136] K. Pathak and M. Bansal, "A fpga based steganographic system implementing a modern steganalysis resistant lsb algorithm," *Defence Science Journal*, vol. 67, no. 5, pp. 551–558, September 2017.

[137] K. S. Shet, A. Aswath, M. Hanumantharaju, and X. Gao, "Novel high-speed reconfigurable fpga architectures for emd-based image steganography," *Multimedia Tools and Applications*, pp. 1–30, January 2019.

[138] *ISE Simulator (ISim)*, Accessed 13 July 2022, available at: https://www.xilinx.com/products/design-tools/isim.html.

[139] *EXIF Tags*, Accessed 14 May 2022, available at: https://exiftool.org/TagNames/EXIF.html.

[140] *Exchangeable Image File Format - an overview*, Accessed 06 August 2022, available at: https://www.sciencedirect.com/topics/computer-science/exchangeable-image-file-format.

[141] *Extensible metadata platform (XMP), ISO 16684-1:2012*, Accessed 27 April 2022, available at: https://www.iso.org/standard/57421.html.

[142] J. R. Smith and P. Schirling, "Metadata standards roundup," *IEEE MultiMedia*, vol. 13, no. 2, pp. 84–88, April 2006.

[143] S. L. Weibel and T. Koch, "The dublin core metadata initiative," *D-lib magazine*, vol. 6, no. 12, pp. 1082–9873, December 2000.

[144] *HTTP Archive*, Accessed 27 May 2022, available at: https://httparchive.org/.