

A Project Report
on
**SPATIAL IMAGE STEGANOGRAPHY USING
FPGA AND ITS HARDWARE
VERIFICATION**

Submitted by,
J.SWARNA KAMAKSHI (114004211)
S.R.ASHWINI (114004018)
SWETHA.R (114004212)

In partial fulfillment for the award of the degree of
Bachelor of Technology
in
Electronics and Communication Engineering



School of Electrical & Electronics Engineering
SASTRA University
Thanjavur, India.

April, 2014

BONAFIDE CERTIFICATE

Certified that the project entitled **“SPATIAL IMAGE STEGANOGRAPHY USING FPGA AND ITS HARDWARE VERIFICATION”** submitted to SASTRA University, Thanjavur by J.SWARNA KAMAKSHI (Reg.No:114004211), S.R.ASHWINI (Reg.No:114004018), SWETHA.R (Reg.No:114004212) in partial fulfillment for the award of the degree of Bachelor of Technology in Electronics and Communication Engineering is the work carried out independently under my guidance during the period Dec 2013 – April 2014.

Project Guide

**[R.SUNDARARAMAN]
[AP III, ECE, SASTRA University]**

External Examiner

Internal Examiner

Submitted for the University Exam held on _____

DECLARATION

We submit this project entitled “**SPATIAL IMAGE STEGANOGRAPHY USING FPGA AND ITS HARDWARE VERIFICATION**” to SASTRA University, Thanjavur in partial fulfillment of the requirements for the award of the degree of “**Bachelor of Technology**” in “**Electronics & Communication Engineering**”. We declare that it was carried out independently by us under the guidance of R.SUNDARARAMAN, AP III, ECE, SASTRA University.

J.SWARNA KAMAKSHI

(Reg No: 114004211)

S.R.ASHWINI

(Reg No: 114004018)

SWETHA.R

(Reg No: 114004212)

Date :

Signature : 1.

Place :

2.

3.

ACKNOWLEDGEMENTS

Firstly, I would like to express my special thanks of gratitude to **Prof.R.Sethuraman**, Vice Chancellor, SASTRA University who encouraged us throughout the entire course of study.I extend my heartfelt thanks to **Dr.G.Balachandran**, Registrar, SASTRA University for providing the opportunity to pursue this project.

I dedicate my whole hearted thanks to **Dr. B. Viswanathan** , Dean, SEEE, SASTRA University and **Prof.M.Narayanan** , Dean-Student affairs, SEEE, SASTRA University, for their moral support. I also thank **Dr. K. Thenmozhi**, Associate Dean (ECE) who motivated me during the project work.

I owe a debt of deepest gratitude to my mentor **R.Sundararaman, AP-III, Sastra University** for his continuous support and guidance throughout the process during the pursuit of my project work. His deep insight in the field and invaluable suggestions helped me in making progress through my project work.

ABSTRACT

Information security is an indispensable requirement in the present scenario, with increasing instances of unauthorized copying, reproduction and fabrication of information. Although many secret information-sharing schemes have been proposed in the recent years, they are vulnerable to impersonation and tapping. Steganography is a promising and popular method that conceals the secret in a multimedia carrier such as image, audio or video, such that it is unintelligible to unintentional recipients. The existing authentication schemes for image steganography do not do well to protect the integrity of the message. Consequently, it is difficult to recover the entire message without distortion and loss, leading to poor image quality and the benefit of doubt to the hacker.

We propose to address the issue of weak authentication, combined with the optimization of available hardware and software resources. Spatial Image Steganography intends to operate based on the physical location of the pixels in an image. The pixels of the cover image are randomized into 2x2 blocks and stored in the SRAM of Altera DE1 Board with EP2C20F484C7 Cyclone II FPGA. The authentication bit which is a single bit is obtained by performing logical operations. This bit is then embedded in the LSB of the pixels in each cover block along with the secret message bits, using hardware. The RTL Schematic and Technology Map Viewer results are consolidated upon compilation.

The final stego-image is reconstructed from the blocks, followed by simulation-based hardware architecture verification. FPGA is preferred for embedding the secret, owing to its high-speed

block processing, high intrusion resistance, huge internal storage and better performance in comparison to software platforms.

The image acquisition and processing are done using LabVIEW software. The DE1 Control Panel application is used to write/read the contents to/from the 256K x 16 bits *IS61LV25616* CMOS Static RAM. Quartus II software is used to embed the secret and parity bits into the image blocks stored in the FPGA's SRAM. The reconstruction of the image is accomplished using LabVIEW. Finally, the architecture is verified by Timing Simulation in Quartus II.

The proposed project has myriad applications in the field of medicine, defense, politics, banking etc. where confidentiality, integrity and authentication are of utmost necessity. The future feasible additions to this project might include multi-level nesting of secret images in the cover and successful recovery. E.g.) Hiding two or more barcodes in one cover image. There is also a good prospect for works including an element to ensure non-repudiation.

CONTENTS

Chapter No.	Chapter Name	Page Nos.
1.	Introduction	1
1.1	Need For Information Security	1
1.2	A Comparative Analysis of secret hiding techniques	3
1.3	Steganography	5
1.3.1	Mechanism as a rule of thumb	5
1.3.2	Techniques	6
1.3.3	Old Vs New	6
2.	FPGA Implementation	9
2.1	FPGA Vs ASIC's	9
2.2	Overview of advantages	10
3.	LSB Substitution	11
4.	Proposed Methodology	13
5.	LabVIEW	14
5.1	LabVIEW in Image Processing	14
5.2	Image Acquisition and Processing using LabVIEW	15
6.	DE1 Board	20
6.1	Components	20

6.2	CMOS SRAM	23
6.2.1	Pin Configuration	24
6.2.2	Features	25
7.	DEI Control Panel	26
7.1	Activation of control panel application	26
7.2	Accessing SRAM in DEI Board	28
7.2.1	SRAM Write	28
7.2.2	SRAM Read	30
8	Information Hiding	31
8.1	Info-Hiding Algorithm	32
8.2	Design using Quartus-II	33
8.2.1	Pin Planning	34
8.2.2	RTL Schematic	35
8.2.3	Technology Map Viewer	36
8.2.4	Synthesis Report	37
8.2.5	Compilation Report- Flow Summary	37
8.2.6	Programmer	38
	Reconstruction Of The Image	40
9.1	Procedure	40

	Results	44
10.1	LabVIEW Simulation Results	44
10.1.1	Division of image pixels into 2×2 blocks and storing in text file	44
10.1.2	Reconstruction of image after embedding using FPGA	45
10.2	MSE and PSNR values of test images	46
10.3	Timing Analysis	50
10.4	Conclusion	53
	References	54

LIST OF FIGURES

Sl.No.	Figure No.	Figure Name	Page No.
1.	1.1	Steganography-Illustration	5
	1.2	Steganography Techniques	6
2.	2.1	Design Flow in a) ASIC b) Reconfigurable Hardware	10
3.	3.1	LSB Substitution-Example	12
4.	4.1	Block Diagram of Proposed Methodology	13
5.	5.1	Operations done in LabVIEW	15
	5.2	Original Image matrix	17
	5.3	Division of image to blocks	17
	5.4	2x2 Blocks generated by Mathscript	17
	5.5	Column-wise division of blocks	18
	5.6	Embedding procedure	19
6.	6.1	DE1 Board-Components	20
	6.2	A snapshot of programming the FPGA Chip	23
	6.3	Functional Block Diagram	24
	6.4	Pin Configuration	24
7.	7.1	Selection of USB Blaster for JTAG Interface	27

	7.2	DEI Control Panel Interface	27
	7.3	Selection of SRAM Tab	28
	7.4	Selection of source file for write operation	29
	7.5	Progress of write operation	29
	7.6	Selection of text file for read operation	30
8.	8.1	Schematic-Information Hiding	31
	8.2	Secret Hiding Algorithm	32
	8.3	Schematic Diagram of Clock Circuit	33
	8.4	VHDL Approach to embedding	34
	8.5	Pin planner	35
	8.6	RTL Schematic- Bird's eye view	36
	8.7	Technology Map Viewer- Bird'd eye view	36
	8.9	Compilation Summary	37
	8.10	Output Verification using Control Panel Application	38
9.	9.1	Reconstruction Procedure	42
10.	10.1	Division of image pixels into 2 x 2 blocks and storing in text file	44
	10.2	Reconstruction of image after embedding	45
	10.3	Image Name- Lena a) Original Cover Image b) Reconstructed Stego Image c) Histogram Report of cover image d)	46

		Histogram Report of stego image	
	10.4	Image Name- Pepper a) Original Cover Image b) Reconstructed Stego Image c) Histogram Report of cover image d) Histogram Report of stego image	47
	10.5	Image Name- Baboon a) Original Cover Image b) Reconstructed Stego Image c) Histogram Report of cover image d) Histogram Report of stego image	48
	10.6	Image Name- Kristen a) Original Cover Image b) Reconstructed Stego Image c) Histogram Report of cover image d) Histogram Report of stego image	49
	10.7	Timing Analysis Summary	51
	10.8	Vector Waveform file	52

LIST OF TABLES

S.No	Table No.	Table Name	Page No.
1	1.2	A comparative analysis of Secret Hiding Techniques	3
2	6.4	Pin Description	25
	6.5	SRAM Truth Table	25
3	8.1	Utilization of SRAM memory based on image size	35
4	8.2	Flow summary for various image sizes	38

Chapter 1

1. Introduction

1.1. Need For Information Security

In the 1990s, internet was an exclusive communication tool for military personnel, business contractors and universities. However, over the years, it has made transitions from private, restricted use to widespread, phenomenal public usage. This medium of communicating information spurred the growth of industries and other urban agglomerations with its reduced cost, timeliness and availability. It has also given the common man an enormous access to free flow of information, as promptness and speedy access allow for the thriving of banks, new businesses, healthcare and other government agencies. However, the downside is that information is more vulnerable and prone to threats.

Some of the common threats posed on secret information sharing are:

- Unauthorized access, modification and obliteration of information.
- Embezzlement or misuse of information upon authorized access.
- Malicious software programs (Trojans/ viruses) that can hamper vital data.
- Inadvertent alteration or destruction of information.
- Social Engineering, which is a fraudulent collection of sensitive personal information by psychologically manipulating the people into believing that it is a genuine procedure. E.g) Phishing, pre-texting, Quid pro quo (something for something) where the attacker offers to help and in turn, launches malware, extracts password, etc.

- Utility failure, such as power, heat, etc. that can stop a system from functioning and erase data altogether.

The identifiable weakness of an information system can be termed as its vulnerability. It proves to be the root-cause for threats and puts a system at risk. Even if information successfully reaches a receiver from a transmitter, its legitimacy should be verified to know if it has been falsified by an interceptor.

The theft of Intellectual Property (IP) by a potential competitor can render a person out of business. Tapping of sensitive financial, family or medical information by unwanted parties can leave the victim bankrupt and worst yet, make him/her prey to identity theft. Thus, an innocent citizen's name, address, social security number etc. can be used by perpetrators for committing crimes that can be heinous, costly and contentious.

When important military information from a government agency is ambushed by an adversary, it puts the national security at stake, compromising the lives of millions of citizens. Also, the medical or military information changed by a hacker can lead to unforeseen circumstances and deadly outcomes. Thus, information security is the need of the hour.

Five security services should be incorporated when transferring information over any channel or network, to arrest data-abuse and ensure successful reception. They are as follows:

- 1. Message Confidentiality:** It aims to make the message sensible only to the intended recipients. To the rest, it is just some unintelligible data. Confidentiality is a must for bank transactions, emails containing legally privileged information, etc.

2. **Message Integrity:** This ensures that the message received is the same as that sent by the sender, without any modification or concoction by a middle-man. The message should not be subject to any change before reception, either accidentally or maliciously.
3. **Message Authentication:** It deals with validating the identity of the true sender. If the message comes from an imposter and not the rightful sender, it needs to be detected. If an unprivileged user furnishes deposit transactions to a bank's central computer and leads it to believe that it is coming from another branch, then all access-control schemes are for naught and he can make easy wealth in a day. So, authentication protects integrity and verifies the data origin.
4. **Message Non-repudiation:** It guarantees that the person who sent a message cannot later deny sending it. The receiver is bound to maintain the record of the request/ command from the sender. E.g) A person requesting money to be transferred from his account to another and later repudiating his claim.
5. **Entity Authentication:** This feature gives the user or entity access to system resources after verifying his/her identity. This is in the best interest of both the organization as well as the individual.

1.2. A Comparative Analysis of Steganography, Digital Watermarking and

Cryptography

Sl. No.	ATTRIBUTES	STEGANOGRAPHY	DIGITAL WATERMARKING	CRYPTOGRAPHY
1.	DEFINITION	Covered Writing- Derived from Greek, steganos:covered Graphia : writing	Embedding message on a host signal.	Secret Writing- Derived from Greek, kruptos : secret Graphen : writing

2.	CARRIER	Any Digital media	Audio, images, formatted text, video, 3D models.	Mostly associated with text and rarely image-based.
3.	SECRET	Payload	Watermark	Plain-text
4.	KEY	Optional	--	Mandatory
5.	RESULT	Stego-file	Watermarked file	Cipher-text
6.	NATURE OF ATTACK	Steganalysis	Image Processing	Cryptanalysis
7.	DETECTION	Blind	Coherent Detection – knowledge of cover or watermark is a necessity	Blind
8.	PURPOSE	Secret communication	Proof of ownership by embedding copyright information	Data protection
9.	NUMBER OF INPUTS	Minimum of two, except in the case of self-embedding	--	One
10.	VISIBILITY	Never- the very existence of message is concealed in plain sight	Maybe visible or invisible	Always visible
11.	CONCERN	Perceptibility, Capacity of secret	Robustness against signal processing operations like spatial filtering, lossy compression, scanning and geometric distortions	Robustness, confidentiality, integrity, authentication, non-repudiation of data origin (data accountability)
12.	IS BROKEN WHEN	Attacker detects the use of steganography and is able to read/modify/remove hidden data.	It is removed or replaced.	De-ciphered
13.	RELATION TO COVER	Not related. Message is more important than cover	Becomes an aspect of the cover image.	--

14.	PLIABILITY	Any cover can be chosen.	Limited choice of cover.	--
15.	HISTORY	About 2500 years old with a variation of its digital version in the recent times.	Over 1000 years old but systematic study started a100 years ago.	Modern Era.
16.	COMMUNICATION	Point-to-point in a covert channel	One-to-many	One-to-one / One-to-many

1.3. Steganography

1.3.1. Mechanism as a rule of thumb:

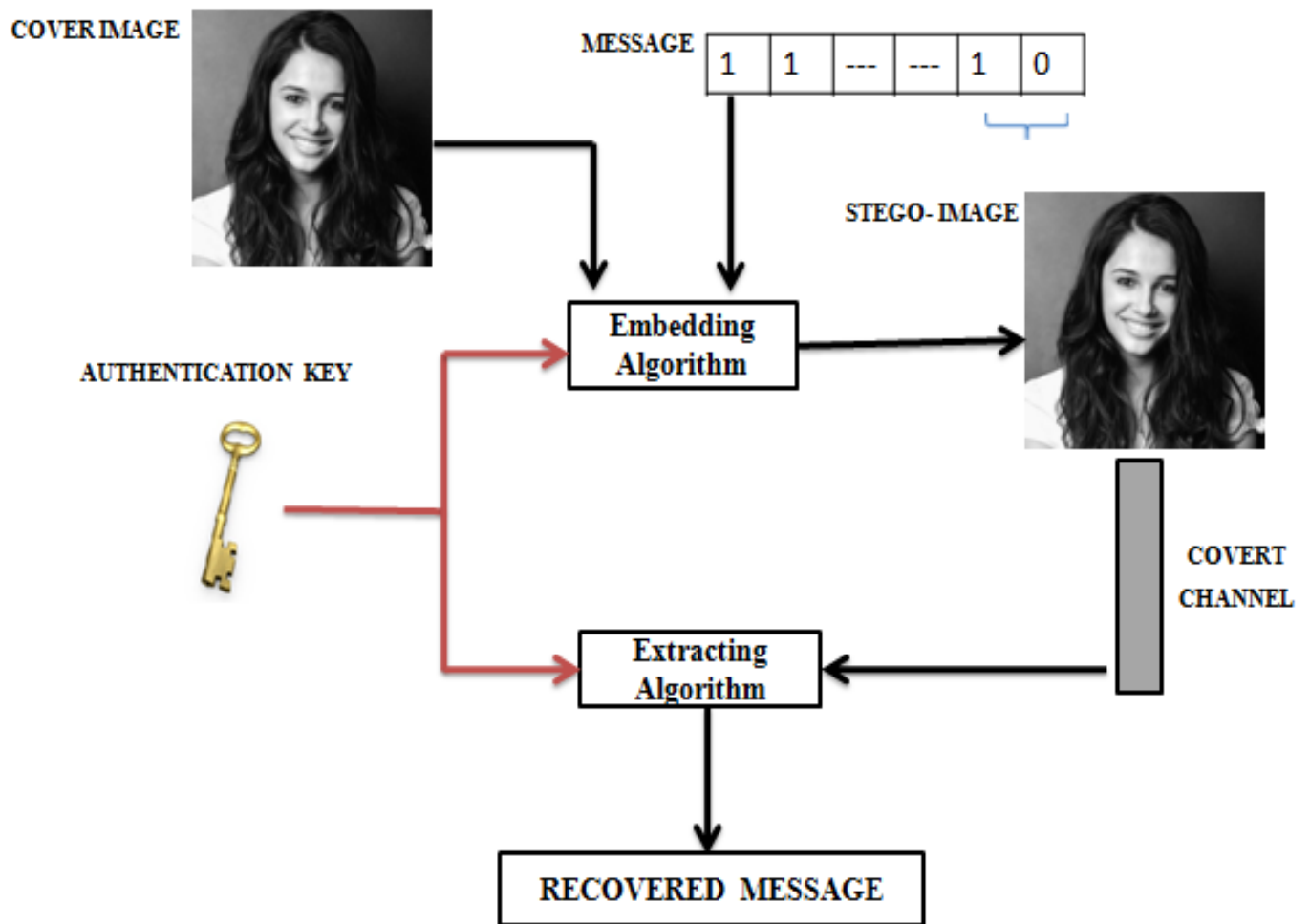


Fig.1.1 Steganography - Illustration

1.3.2. Steganography Techniques

The following are some of the popular techniques of achieving steganography :

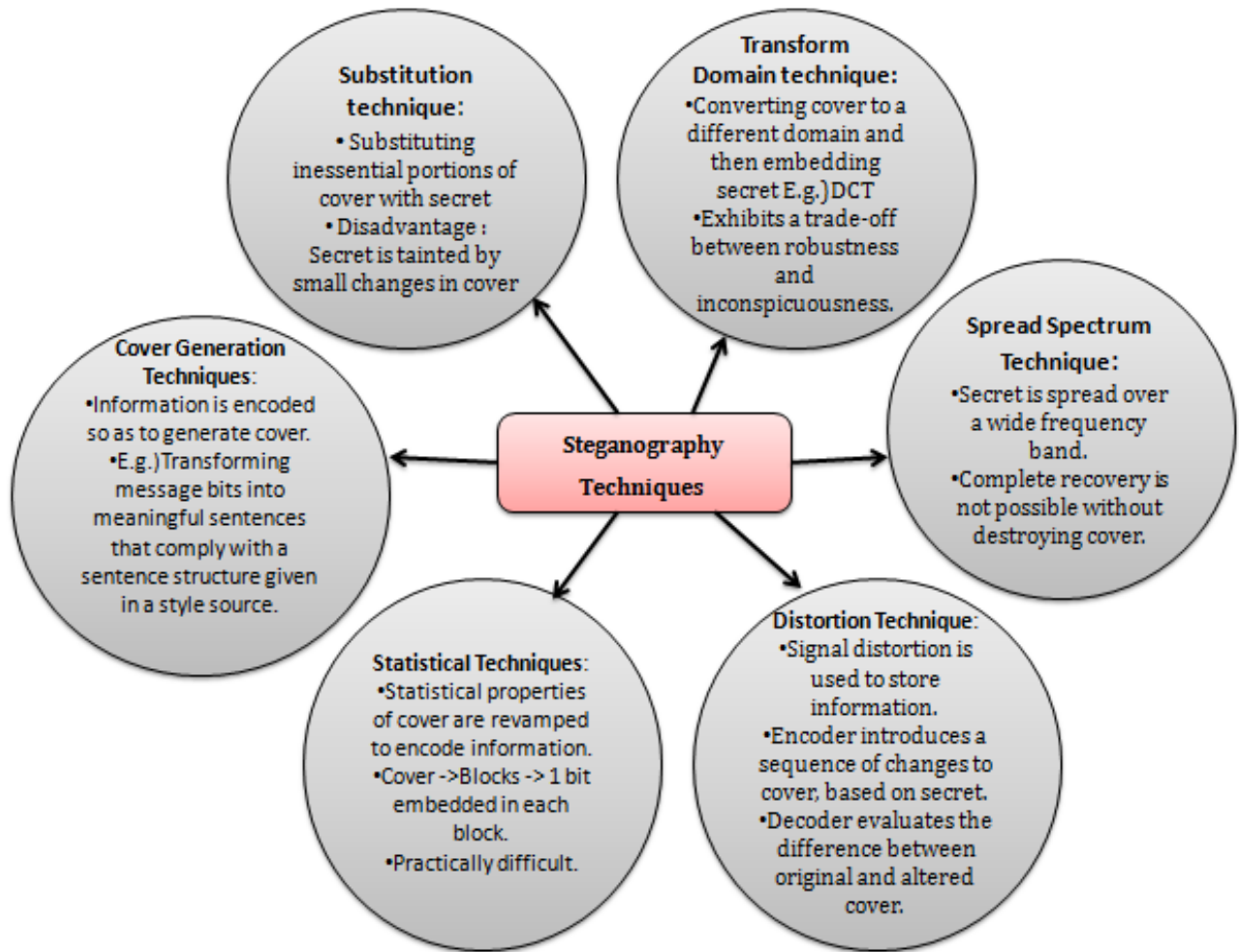


Fig.1.2 Steganography Techniques

1.3.3. Old vs. New

Some of the earliest visual secret sharing schemes include thresholding concept of Shamir and Blackley. Here, the secret is distributed to a group of 'n' participants. Each participant gets a share of the information. The secret can be retrieved only on combining sufficient number of shares from different participants, say from a group of 't'(threshold) or more. Recovering the

secret from individual shares is impractical. This system is popularly called **(t,n) Threshold Scheme**, such that $t \leq n$. Less than 't' players cannot reconstruct the secret information.

The drawback in this was that

- It was suitable for binary images only and called for generation of (t-1) random bits to disperse 1-bit secret among 't' participants. So, in order to distribute a secret message of length 'l', $(t-1) * l$ random bits are required.
- Also, the generated shadows or shares had noise-like properties which were difficult to tackle and easily caught the attention of attackers.

Yet another famous method involves **Variable Size Quantization** wherein compressed secret image is split into shadows based on a polynomial function with degree (t-1).

Disadvantages of this method are

- Variable size quantization yields stego-images which are smaller than the cover image. The large quantization value profoundly distorts the stego-image and is susceptible to attacks.
- Upon reconstruction, the entire secret cannot be recovered. Furthermore, its weak authentication allows even some falsified stego-images.

One of the recent realization of integrity and authentication is through adoption of **Chinese Remainder theorem**, which generates secret shares through a unique integer Y satisfying the congruence of residue R_i modulo integer m_i , where m_i for $1 \leq i \leq r$ is a set of pair-wise relatively prime moduli.

$$\text{i.e. } Y \equiv R_i \pmod{m_i}$$

On solving the system of congruences by application of Chinese Remainder Theorem, the secret can be attained.

- Although this method promises good image quality and authentication, the method of recovery of secret is complex.
- Intense, time-consuming computations are required for acquiring the secret.

Chapter 2

2. FPGA Implementation

2.1. FPGA Vs. ASICs

Although there are a legion of articles and research papers on spatial and transform domain steganography implemented in various software platforms, very few deal with implementation of the same on reconfigurable hardware. Reconfigurable hardware offers umpteen benefits- high processing speed, complete reprogrammability, enormous process control with multi-core processors, flexibility, real-time operation, concurrency, additional performance gain with software, etc.

Conventional ASICs demand protracted and elaborate floor-planning, placement and routing, verification of functionality at a specific timing and many other complex stages, which are eliminated by FPGAs. Firstly, functional specifications are enumerated and a code for the same is written in a Hardware Description Language. Through simulation, behavior or conduct of various components and modules are verified. This is followed by synthesis, which generates a schematic out of the textual code. After synthesis, placement and routing, timing analysis is performed to study the propagation delay between the nodes and understand how to optimize the components for better performance. Finally, the code is burnt on the IC and verified in circuit.

In ASICs, synthesis is followed by static timing analysis and equivalence checking, after which the system is ready for placement and routing, which is estimated to take at least 1-2 months on

handing over to the foundry. In FPGAs, there is no such constraint and the circuit is ready in an instant, if code is written.

The following flowchart describes the design flow in FPGAs:

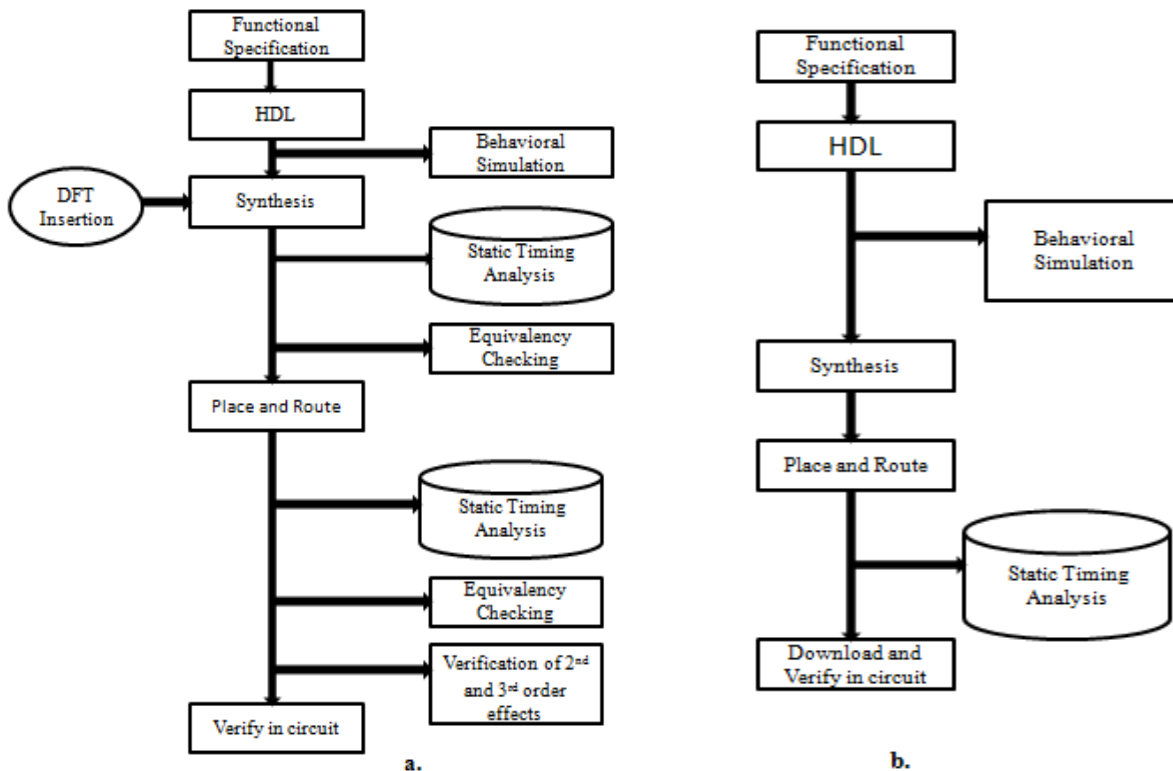


Fig.2.1 Design Flow in a) ASIC

b) Reconfigurable Hardware

2.2. An Overview of Advantages

FPGAs offer high *intrusion resistance* that offers protection from malicious interceptors. Additionally, they also *compute block units at a very high speed* and *provide huge internal RAM* storage, facilitating a heavy payload.

Chapter 3

3. LSB Substitution

Many steganographic algorithms deal with concealing secret more in the edges than in smooth portions of the cover image. This causes abrupt variations at the edges, causing severe distortion. To control these undesirable changes in image edges, LSB (Least Significant Bit) substitution is employed in many modern algorithms, with subtle changes, to achieve good resolution of stego-images.

Here, bit wise modulus operandi is implemented on the superfluous bits i.e. the LSB of image pixel values and the mean square error and peak signal to noise ratio are calculated to evaluate the performance rate.

Let P be the original 8-bit grayscale cover image of size $m_c \times n_c$.

Then, P can be represented as,

$$P = \{a_{ij} \mid 0 \leq i \leq m_c, 0 \leq j \leq n_c\}; \quad a_{ij} \in \{0,1,2 \dots .255\}$$

Let M be a secret message of s -bits denoted by

$$M = \{m_i \mid 0 \leq i \leq s, m \in (0,1)\}$$

Assuming that this s -bit message is embedded to k -least significant bits of the cover image, then the resulting embedded message M' can be expressed as:

$$M' = \{m'_i \mid 0 \leq i \leq s', m'_i \in \{0,1, \dots 2^k - 1\}, \text{ where } s' < m_c \times n_c.$$

The relation between original message M and embedded message M' can be expounded as

$$m'_i = \sum m_{i*k+j} * 2^{k-1-j}$$

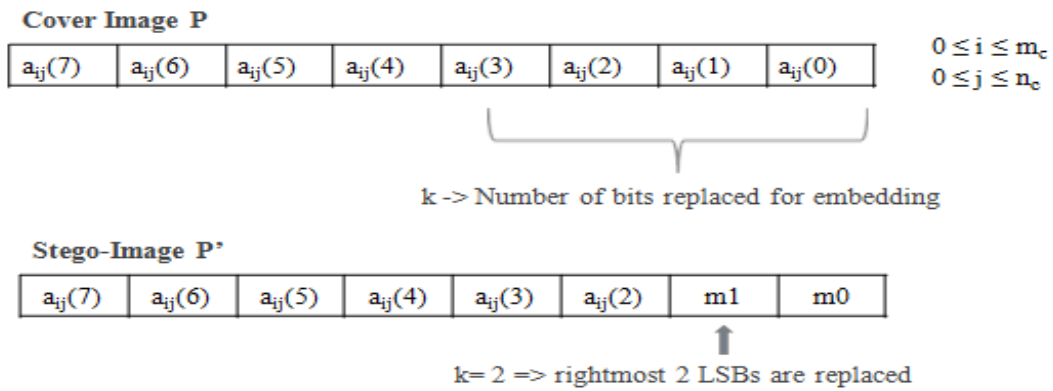
Now, a subset of n' pixels $\{a_{11}, a_{12}, \dots, a_{1s'}\}$ is selected from the cover P in a particular sequence. 'k' LSBs of P are replaced by m'_i on embedding, thereby forming a stego-pixel P' having a set of pixel values $\{a_{11}', a_{12}', \dots, a_{1s'}'\}$ containing the secret message bits.

$$a_{li}' = a_{li} - a_{li} \bmod 2^k + m'_i$$

The embedded message can be retrieved using the formula : $m'_i = a'_{li} \bmod 2^k$

Alternatively, we can also accomplish the same as follows. The cover P is ANDed with an 8-bit value such that the least significant bits to be embedded are made zero. If last one bit is to be embedded, it should be ANDed with FE (11111110). For two bits, FC (11111100), and so on.

This is followed by ORing the message bits to it: $P' = (P \& FC) \wedge m'_i$



Example :
Pixel value =159
Binary Equivalent = 10011111
k=2 ; Message = 10
Stego-pixel (binary rep)= 10011110
Stego-pixel (Decimal rep) = 158

Fig.3.1 LSB Substitution-Example

Chapter 4

4. Proposed Methodology

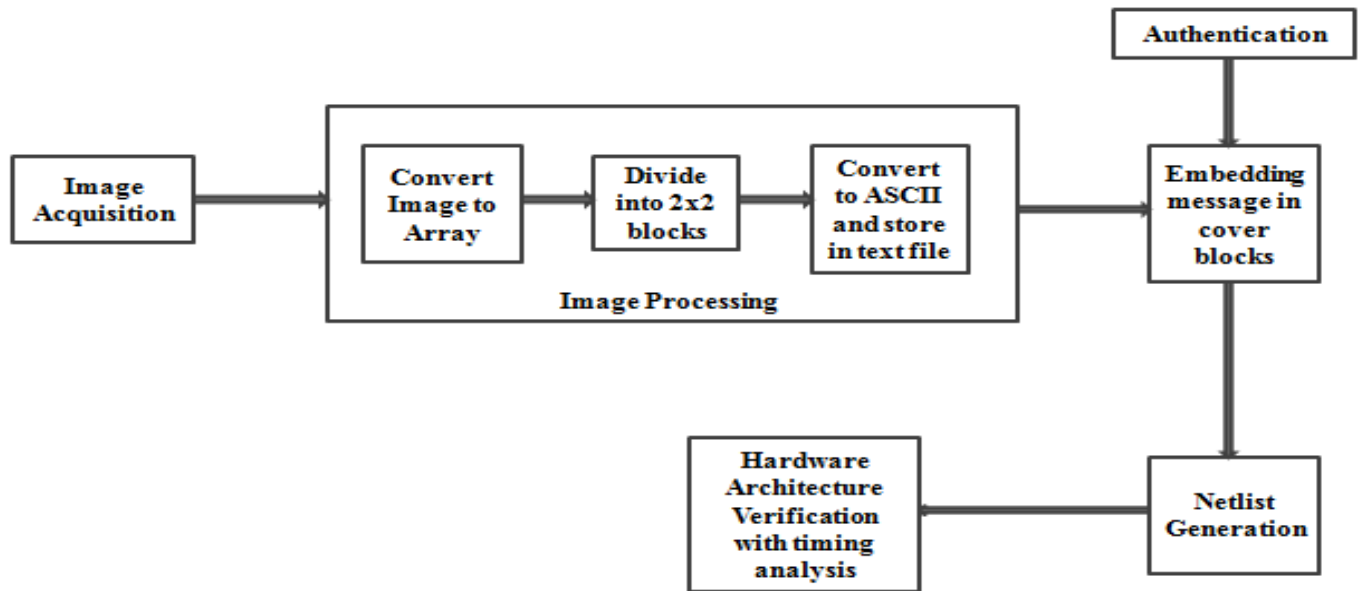


Fig.4.1 Block Diagram of Proposed Methodology

First, a grayscale image is acquired using LabVIEW and converted to an equivalent decimal array. This array is divided into blocks of 2, to increase the payload capacity and ease the information hiding procedure. The 2×2 blocks are converted to ASCII format and stored in a text file. By employing the appropriate secret hiding mechanism, both secret and authentication bits are embedded into the image cover using hardware. The design is generated and netlist is created to estimate the number of logic elements required. RTL and Technology Map Viewer schematic are done to confirm the correct logic and connections between the registers and combinational blocks in the design. Additionally, timing analysis is done to calculate the timing required to embed the secret message.

Chapter 5

5. LabVIEW

5.1. LabVIEW in Image Processing

Laboratory Virtual Instrumentation Engineer's Workbench (LabVIEW) is a graphical development platform or software for visual programming, developed by the National Instruments. It is preferred over other softwares owing to its interactive graphical user interface that is ideal for parallel processing, data acquisition and analysis, estimation, instrument control, automation. Thus, it enhances productivity in comparison to conventional programming languages. IMAQ Vision, a part of the Vision Development Module, is a library of LabVIEW VIs that can be extensively used for image processing applications.

An image is a two-dimensional array representing light intensity. It can be denoted as $f(x, y)$, where f is the brightness of a point (x, y) , x and y represent the spatial coordinates of the *picture element or pixel*. The pixels are real or complex and hold the visual information in discrete form. They can be represented by a finite number of bits. Top, left corner of an image is generally represented by coordinates $(0, 0)$.

Digital image processing is a pre-requisite because numerical values are required for further manipulation and it is not plausible to operate directly on raw pictorial data.

As our application calls for embedding secret in an image, a grayscale image is preferred over colour images. All gray levels ranging from black to white constitute grayscale image. This is because grayscale images are monochromatic while colour images are built of several stacked

colour channels. For instance, RGB image contains three channels (Red, Green, Blue) and CMYK images, four channels (Cyan, Magenta, Yellow, Black) etc. The number of quantization levels is an integral power of 2. i.e. $L = 2^b$, where 'b' is the number of bits used to represent the pixel value. The number of bits used to define pixel values is also called Bit Depth. Greater the bit depth, greater is the number of quantization levels and thus the tones that can be represented in a digital image.

So, the easiest way to alter a digital image is by applying changes to its grayscale values which are 8 bits long (unsigned integer type) with 256 possible gray levels, and pixel values ranging from 0 (Black) to 255 (White). This choice also guarantees a higher processing speed.

5.2. Image Acquisition and processing using LabVIEW

A standard 8-bit BMP (Bitmap) grayscale image- *Cameraman* has been used for this project. The size of the image is 256×256. So it has a total of 65536 pixel values. The following procedure is to be adopted for processing the image using LabVIEW.

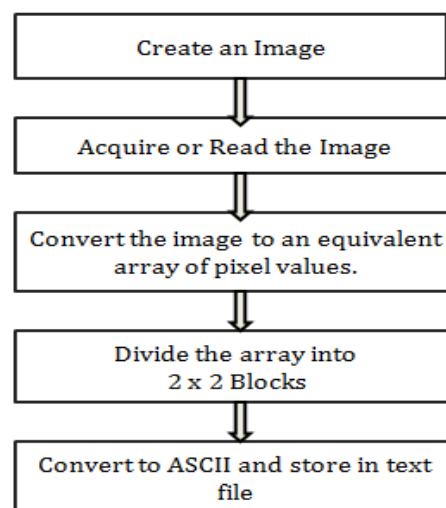


Fig.5.1 Operations done on image via LabVIEW

The cover image array is divided into blocks as FPGA can process blocks quicker than the entire array. Also, greater the number of blocks, great is the amount of secret embedded. Thus 2×2 blocks offer maximum capacity for holding secret data. Furthermore, if the entire secret is embedded in one portion of the image, it gives the interceptor the benefit of doubt and hence the secret can be tapped easily.

The resultant 2×2 blocks are converted to ASCII file format to offer optimization of memory. As bit '7' in ASCII representation is always zero, the ASCII character set takes up only half of the possible character codes in an 8-bit byte. So, if the file size is 128 KB in decimal and hexadecimal file formats, when converted to ASCII, it would only have a size of only 64 KB, as each character occupies only 1 byte.

The division of image array into 2×2 blocks and storage in a text file is achieved by Mathscript node, which helps combine the benefits of both text-based math and graphical programming. The MathScript is an embedded LabVIEW feature that connects the text-based input and output variables with the inputs and outputs of LabVIEW.

The blocks are divided column-wise. Let the image pixel values be represented by $a[i, j]$, where i and j represent the row and column number. As the image has 256 rows and 256 columns, the first block contains elements $a[1,1]$, $a[2,1]$, $a[1,2]$, $a[2,2]$. The second block contains $a[3,1]$, $a[4,1]$, $a[3,2]$, $a[4,2]$ and so on. So, finally there are 128×128 blocks of 2×2 in total.

Each cover block thus contains four 8-bit values that are stored consecutively in the text file in ASCII format, without any spacing or indentation, as follows:

$b_{1,1} b_{2,1} b_{1,2} b_{2,2} b_{3,1} b_{4,1} b_{3,2} b_{4,2} \dots$ where $b_{i,j}$ is the ASCII equivalent of $a_{i,j}$.

a[1,1]	a[1,2]	a[1,3]	a[1,4]...a[1,256]
a[2,1]	a[2,2]	a[2,3]	a[2,4]...a[2,256]
a[3,1]	a[3,2]	a[3,3]	a[3,4]		
a[4,1]	a[4,2]	a[4,3]	a[4,4]		
.....
.....
.....
a[256,1]	a[256,2]	a[256,3]a[256,256]

Fig.5.2 Original Image matrix

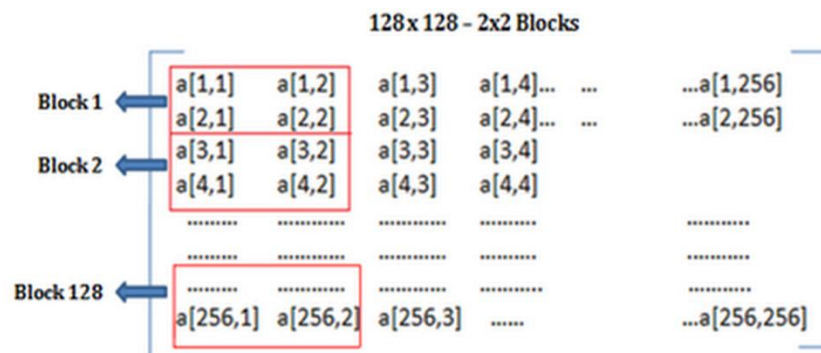


Fig.5.3 Division of image to blocks

```

MATLAB Command Window
File Edit View Window Help
[Icons] ?

111 112
103 128

new_f(:, :, 123, 128) =
    119    98
    136    119

new_f(:, :, 124, 128) =
    130    97
    112    136

new_f(:, :, 125, 128) =
    139    96
    106    96

new_f(:, :, 126, 128) =
     84    100
    126    106

new_f(:, :, 127, 128) =
    125    120
    137    114

Ready NUM

```

Fig.5.4 2x2 Blocks generated by Mathscript

The 16383 (128×128), 2×2 blocks of a 256×256 image generated by LabVIEW Mathscript node are displayed in the MATLAB Command Window as shown in fig.5.4.

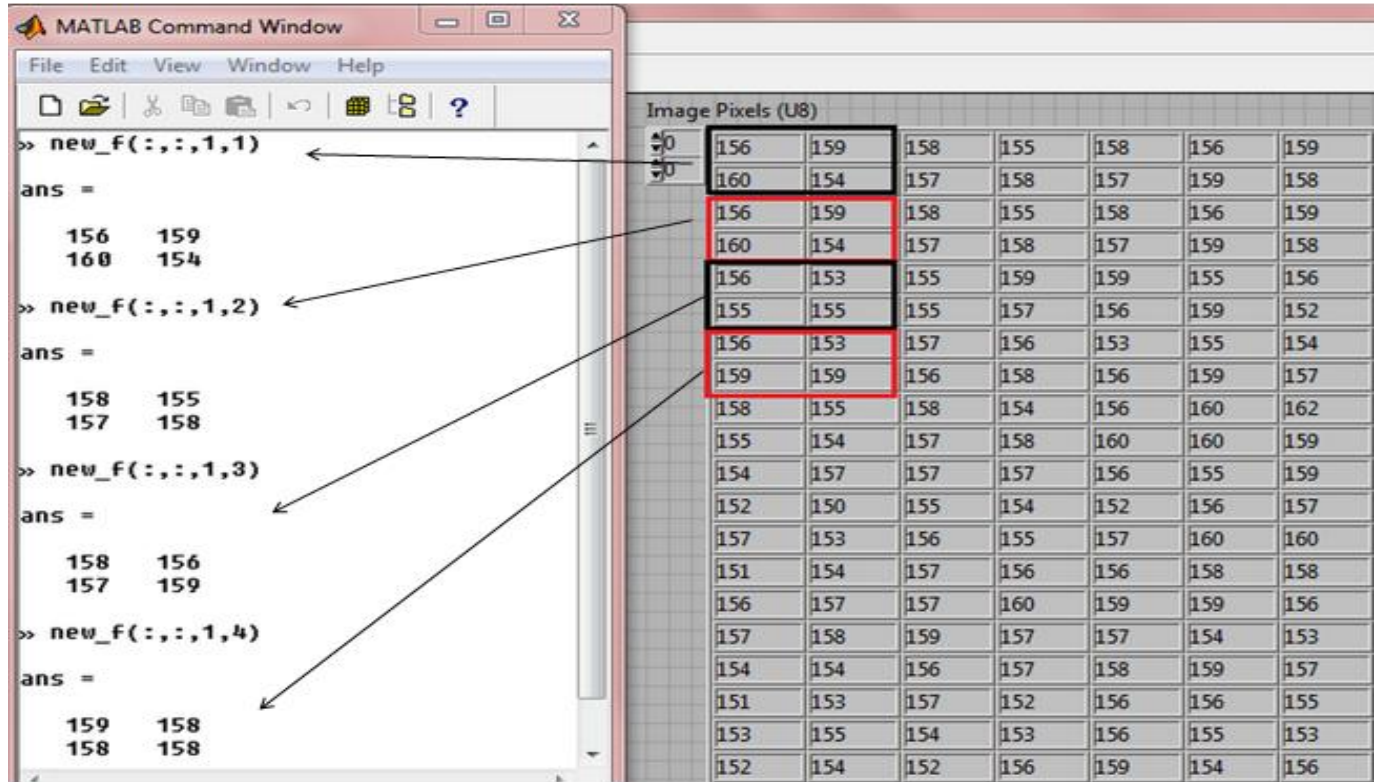


Fig.5.5 Column-wise division of blocks

To check the value of any particular 2×2 block, you can type the name of the array, followed by the position as given in fig.5.5. Here, **new_f** is the name of the array containing 2×2 blocks.

This figure illustrates the division of the original image array into 2×2 blocks column-wise. These blocks are stored in a text file, to be later accessed by the hardware for the purpose of embedding secret.

The LabVIEW implementation of steganography without division of image pixels into blocks is illustrated by the following image. The algorithm is implemented in software to test its working and ease its implementation in hardware. LED indicators prove an eye-pleasing approach to understanding the logic of operation.

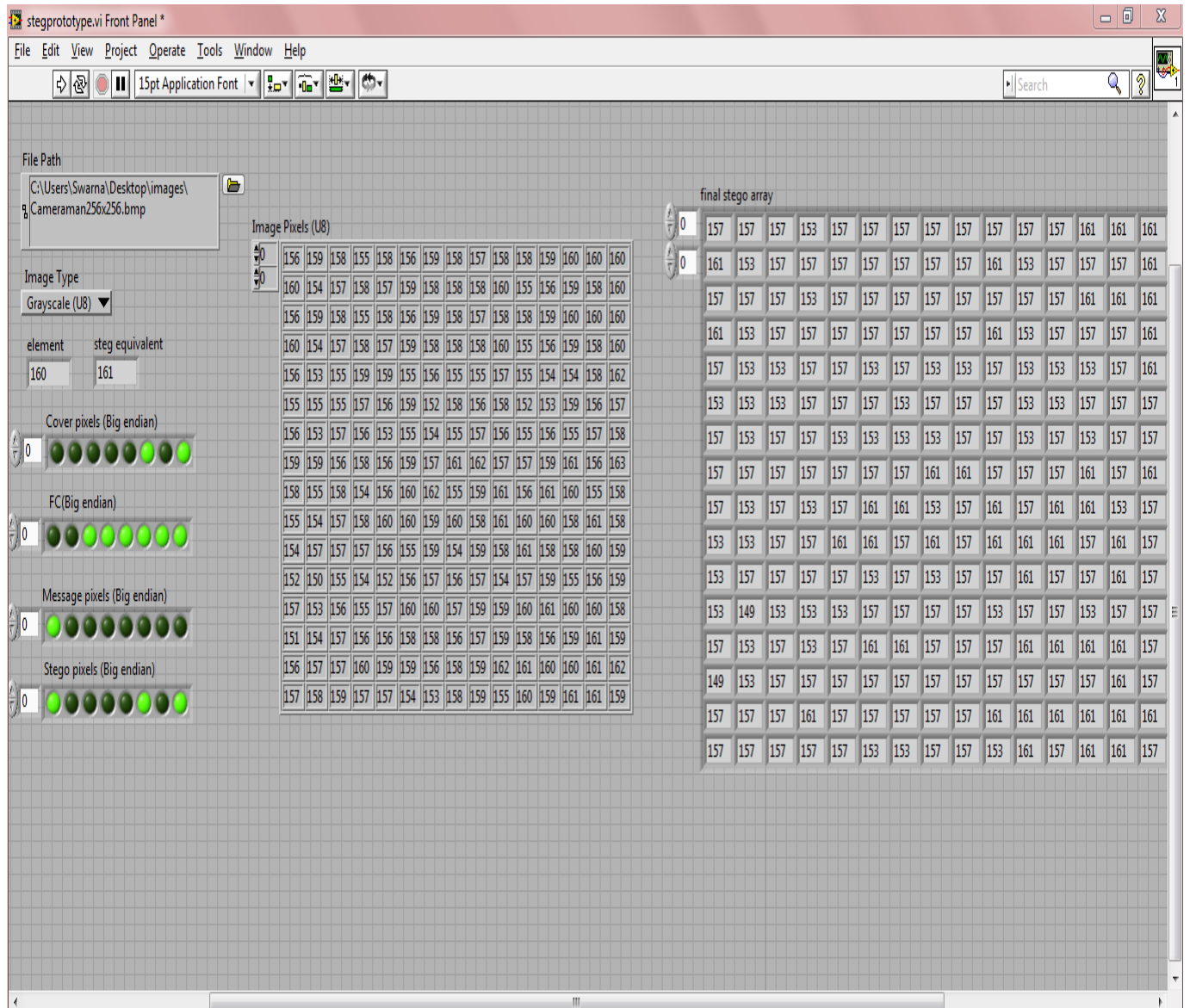


Fig.5.6 LabVIEW Simulation of embedding procedure without division of pixel matrix to blocks

In LabVIEW, the binary numbers are represented in Big Endian format, that is, LSB first and ending with MSB. In the above example, the last two bits of the pixel 160 are embedded with message bits “10”to obtain the stego-pixel.

Chapter 6

6. DE1 Board

6.1. Components

DE1 package consists of versatile components that succour the use of DE1 board in conjunction with computers having the Microsoft Windows operating system. The layout of the board is depicted below with all pivotal components and connectors.

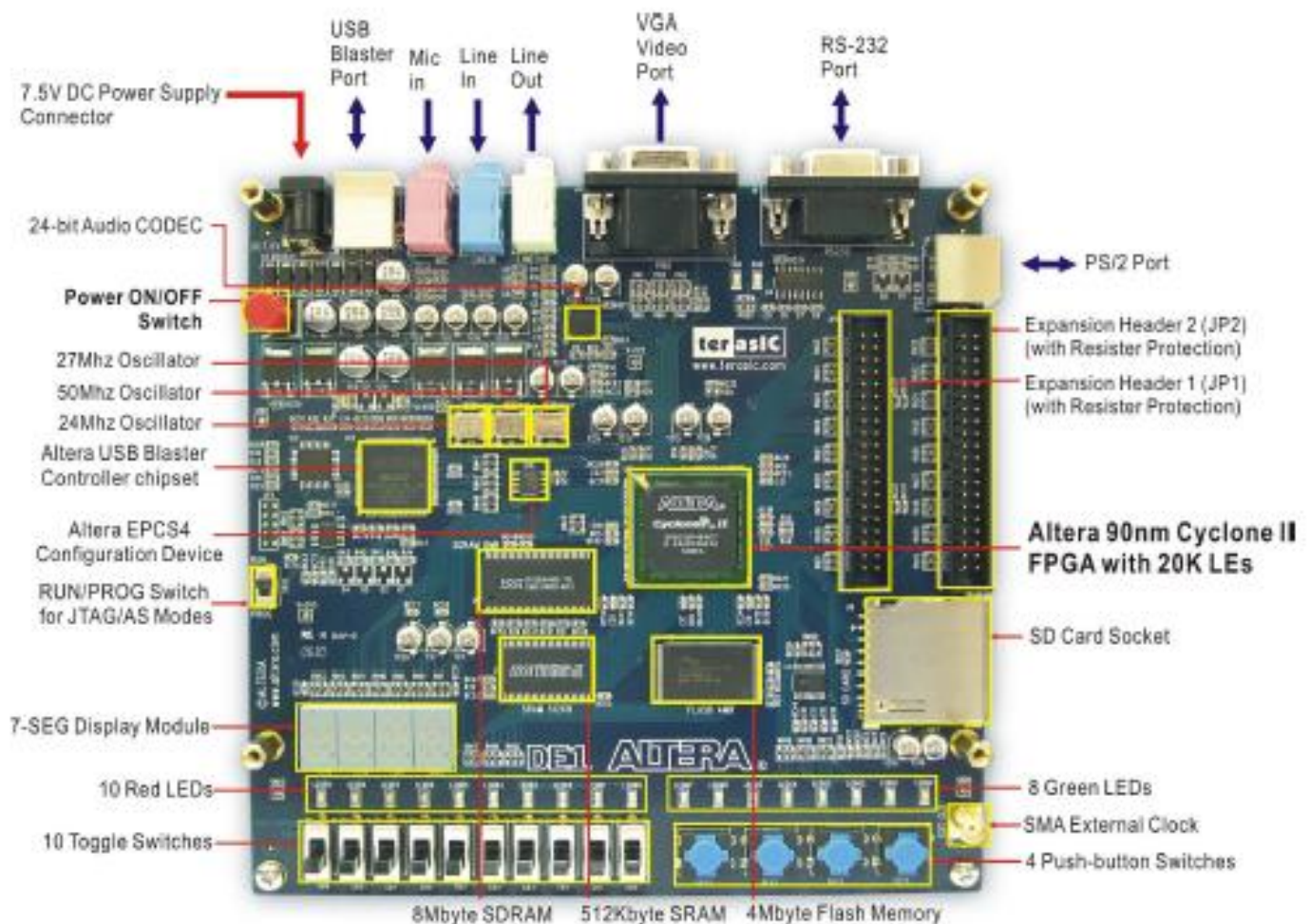


Fig.6.1. DE1 Board-Components

Some of the key components and their functionality are described below:

1. Altera Cyclone II 2C20 FPGA Device

- Altera Cyclone II FPGA family is the second generation series of FPGAs manufactured by Altera to offer cost-effective solution to multifarious applications.
- It uses the state-of-art, 90 nm process technology which substantially reduces the core operating voltage to 1.2V, compared to 1.5V of the 130 nm processes.
- I/O blocks demand a voltage of 3.3V maximum.
- Furthermore, multiple threshold voltages can be set to optimize the speed and power consumption of the transistors.
- So, this FPGA delivers a high performance which ensuring low power consumption at a cost that potentially challenges the Application Specific Integrated Circuits (ASICs).

2. On-Board USB Blaster

- In-circuit reconfiguration of FPGAs can be accomplished using USB Cables. The cables aid in downloading the design changes directly onto the device, allowing quick succession of multiple design iterations and easy prototype generations.
- Configuration and program files are driven from PC to Altera devices through USB BlasterTM.
- The cable can be interfaced with PCs using standard USB ports.
- Additionally, it helps to communicate and debug Nios II Embedded processors. Voltages supported are 1.8, 2.5, 3.3 and 5V.
- It supports both JTAG and Active Serial programming modes (AS).

3. 512 Kbyte SRAM

- Organized as 256K×16 bits.
- Can be accessed using Nios II Processor and DE1 Control Panel.

4. 8 Mbyte SDRAM

- Organized as 1M×16 bits×4 banks.
- Accessible using both DE1 Control Panel and Nios II Processor.

5. 4 Mbyte Flash Memory

- NOR Flash Memory
- Data bus is 8 bits wide.
- Accessible by both Nios II Processor and DE1 Control Panel.

6. Clock

- 50 MHz oscillator.
- 27 MHz oscillator.
- 24 MHz oscillator.
- SMA External Clock input.

7. 10 toggle switches

8. SD Card socket

9. 10 red user LEDs

10. 8 green user LEDs

11. 4 pushbutton switches

12. Altera Serial Configuration device – EPCS4

13. PS/2 mouse/keyboard connector

14. VGA DAC (4-bit resistor network) with VGA-out connector

- 15. Two 40-pin Expansion Headers with resistor protection
- 16. 24-bit CD-quality audio CODEC with line-in, line-out, and microphone-in jacks
- 17. RS-232 transceiver and 9-pin connector

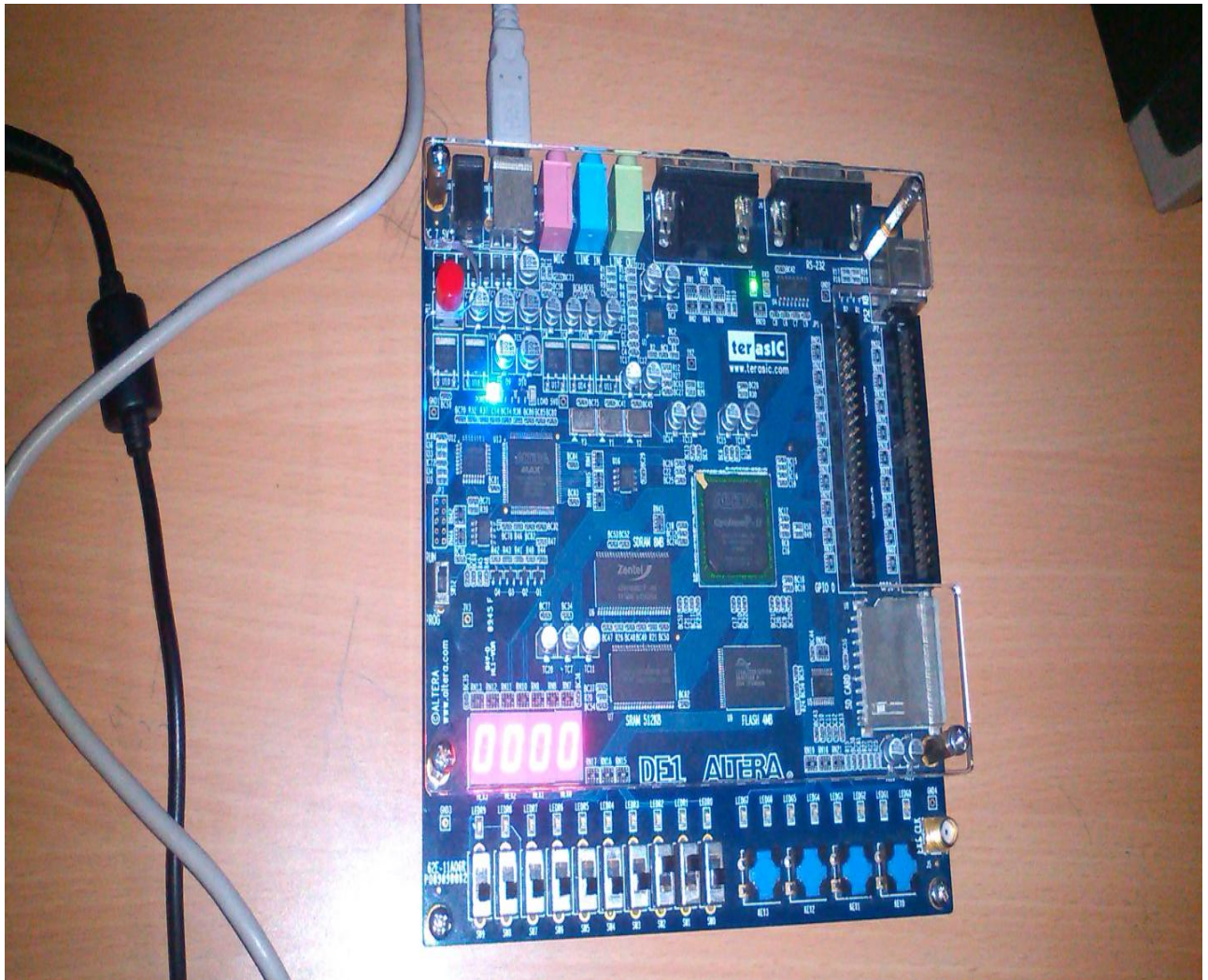


Fig.6.2 A snapshot of DE1 Kit used in the lab with USB Cable when programming the FPGA Chip

6.2. SRAM

The Altera DE1 Board of Cyclone II FPGA contains a 256×16-bit IS61LV25616 high-speed asynchronous CMOS Static RAM.

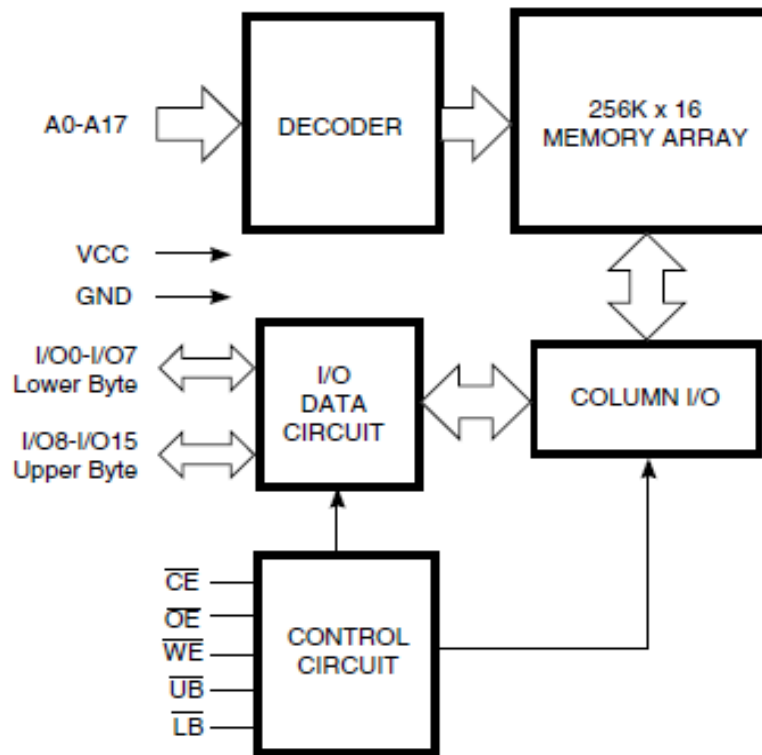


Fig.6.3 Functional Block Diagram

6.2.1 Pin Configuration

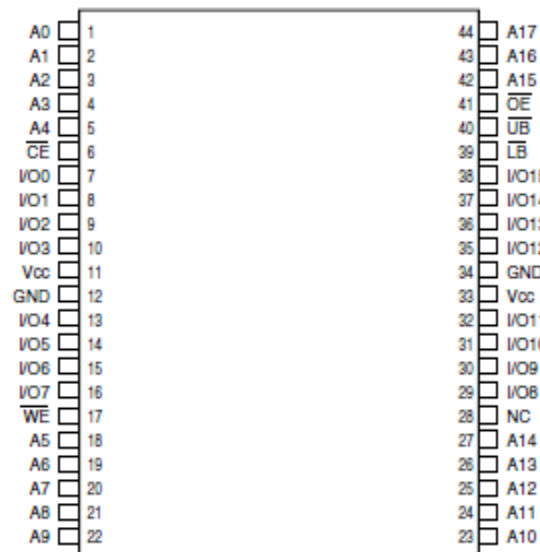


Fig.6.4 Pin Configuration

It has 18 Address lines and 16 bidirectional data lines.

Pin	Description
A0-A17	Address Inputs
IO 0-IO 15	Data Inputs/Outputs
CE(Active Low)	Chip Enable Input
OE (Active Low)	Output Enable Input
WE(Active Low)	Write Enable Input
LB(Active Low)	Lower Byte control(IO 0- IO 7)
UB(Active Low)	Upper Byte control(IO 8- IO 15)
NC	No Connection
GND	Ground

6.2.2 Features

- Can be accessed at a high speed (10,12 and 15 ns)
- Operates at low power
- Stand-by power of less than 5 mA is required. (typically)
- Requires a single 3.3V Power supply.
- Fully static operation : No clock or refresh required.
- Three state outputs – LOW, HIGH, HIGH IMPEDANCE

The following truth table is employed to configure the SRAM of Altera DE1 Board of Cyclone II FPGA for Data read and write operations.

Fig.6.5 Truth Table

Mode	I/O PIN						Vcc Current	
	\overline{WE}	\overline{CE}	\overline{OE}	\overline{LB}	\overline{UB}	I/O0-I/O7		I/O8-I/O15
Not Selected	X	H	X	X	X	High-Z	High-Z	I _{SB1} , I _{SB2}
Output Disabled	H	L	H	X	X	High-Z	High-Z	I _{CC}
	X	L	X	H	H	High-Z	High-Z	
Read	H	L	L	L	H	DOUT	High-Z	I _{CC}
	H	L	L	H	L	High-Z	DOUT	
	H	L	L	L	L	DOUT	DOUT	
Write	L	L	X	L	H	DIN	High-Z	I _{CC}
	L	L	X	H	L	High-Z	DIN	
	L	L	X	L	L	DIN	DIN	

When CE is HIGH (deselected), the device assumes a standby mode at which power dissipation can be reduced down with CMOS input levels.

Chapter 7

7. DE1 Control Panel

Using Control Panel application The DE1 Board comes with a Control Panel application that helps read and write contents from/to the SRAM on the board using a USB connection from a host computer.

Before embedding the message and authentication bits into the 8-bit pixel values stored in the SRAM, it is necessary to run the Control Panel application, in order to configure the corresponding circuit in the cyclone II FPGA.

7.1. Activation of Control Panel

1. The supplied USB cable is connected to the USB Blaster port.
2. The RUN/PROG switch is set to the RUN position
3. The Quartus II software is started.
4. **Tools > Programmer** is selected to reach this window. **Add File** is clicked and in the pop-up window that appears, the *DE1_USB_API.sof* file is selected.
5. The executable *DE1_control_panel.exe* on the host computer is started. The Control Panel user interface will appear.
6. The USB port is opened by clicking **Open > Open USB Port 0**. The DE1 Control Panel application will list all the USB ports that connect to DE1 boards. The DE1 Control Panel can control up to 4 DE1 boards using the USB links. The Control Panel will occupy the USB port until you close that port; you cannot use Quartus II to download a configuration file into the FPGA until you close the USB port.

7.The Control Panel is now ready for use.

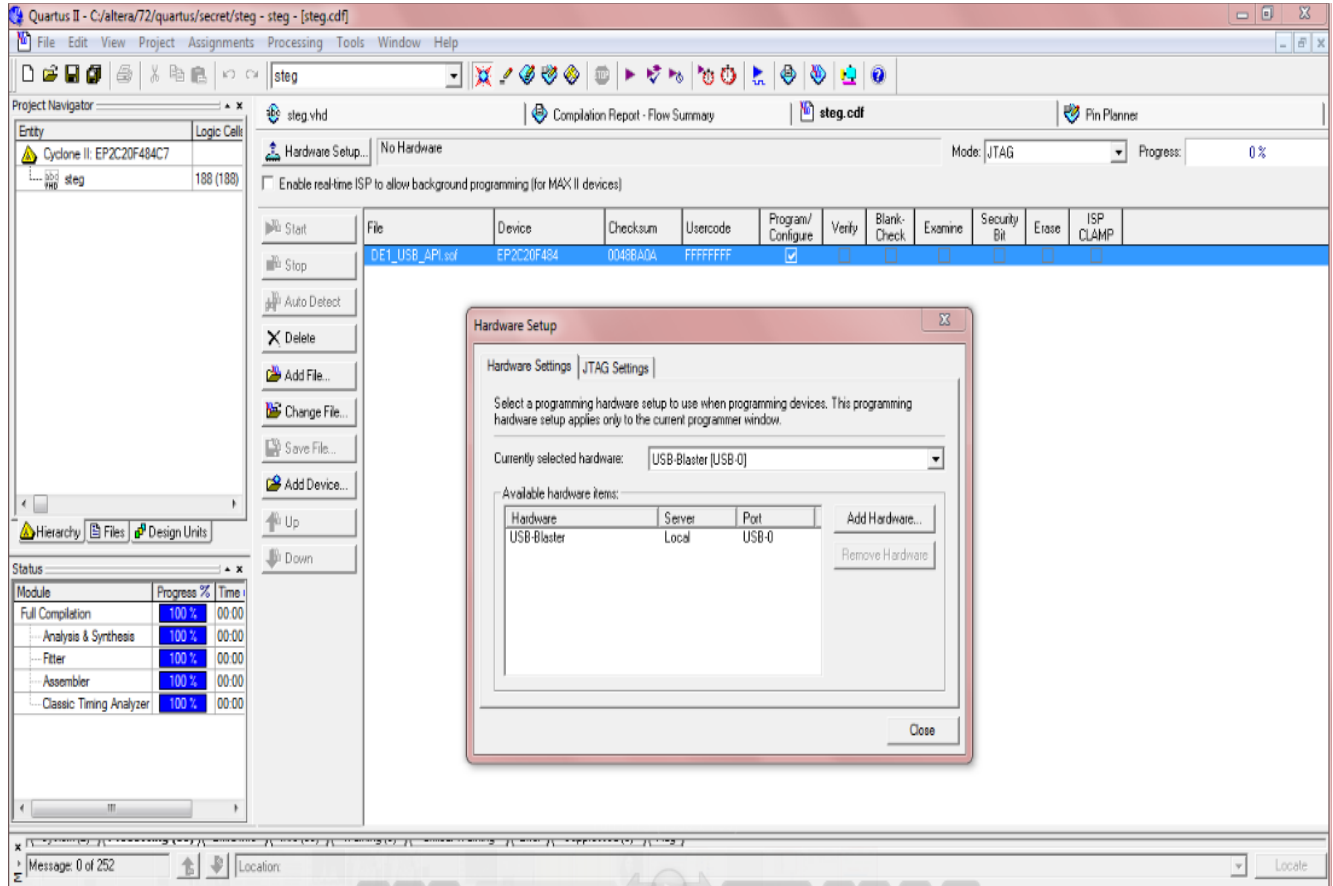


Fig.7.1 Selection of USB Blaster for JTAG Interface

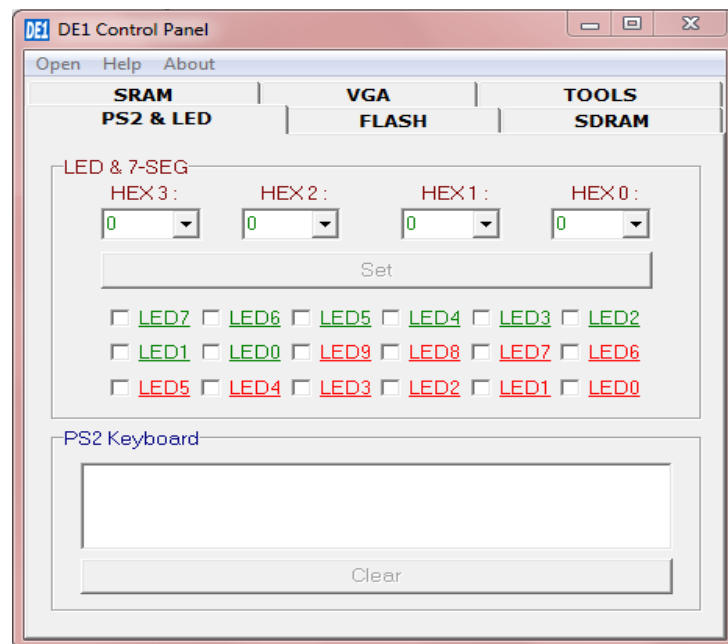


Fig.7.2 DE1 Control Panel Interface

7.2. Accessing SRAM in DE1 Board

SRAM tab is clicked in the DE1 Control Panel application. The following window appears.

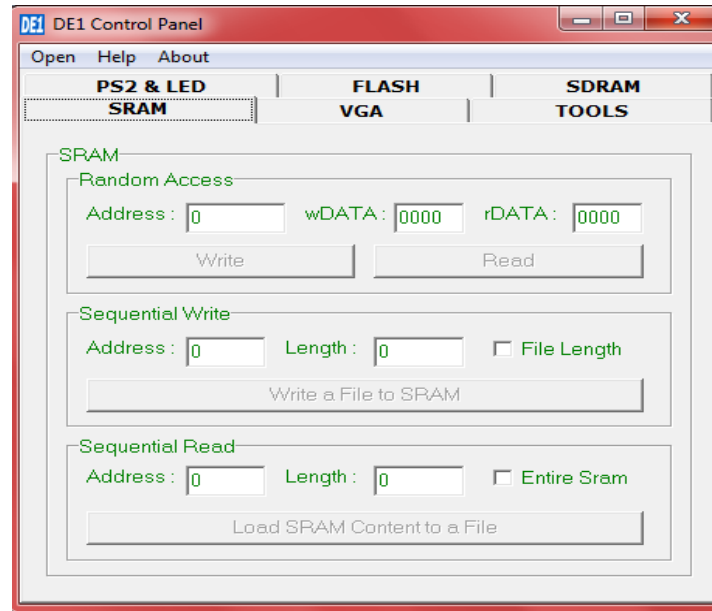


Fig.7.3 Selection of SRAM Tab

7.2.1. SRAM Write

In order to store the 65536 8-bit values sequentially, i.e. 128×128 blocks of 2×2 each,

- Starting address is specified in the **Address** Box. (Starting address 1h, Length 10000h). Each location holds 2 bytes.
- The number of bytes to be written (in Hex) is specified in the **Length** box. As the entire file is to be loaded, a checkmark is placed in the **File Length** box instead of specifying the number of bytes.
- To initiate the writing of data, the **Write a File to SRAM** button is clicked.
- The desired **source file** is specified when the Control Panel application responds with the standard Windows dialog box asking for source file.

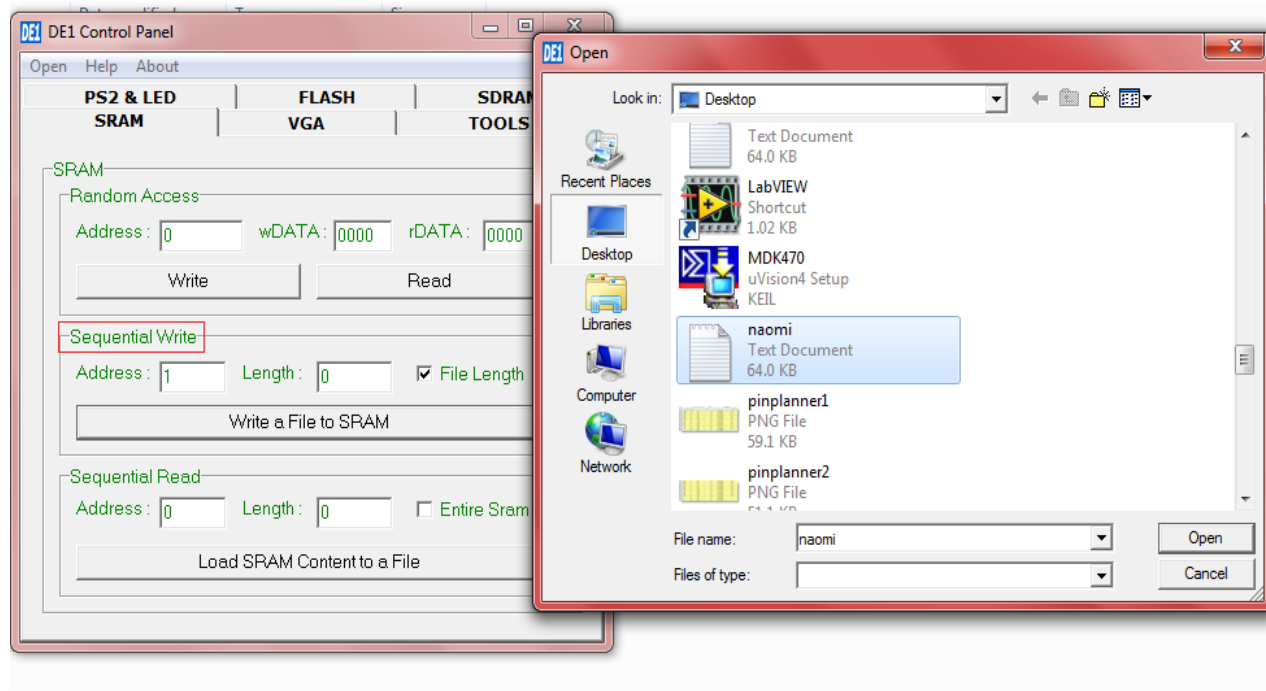


Fig.7.4 Selection of source file for write operation

When writing is done, the following window appears, indicating the progress of the data write operation.

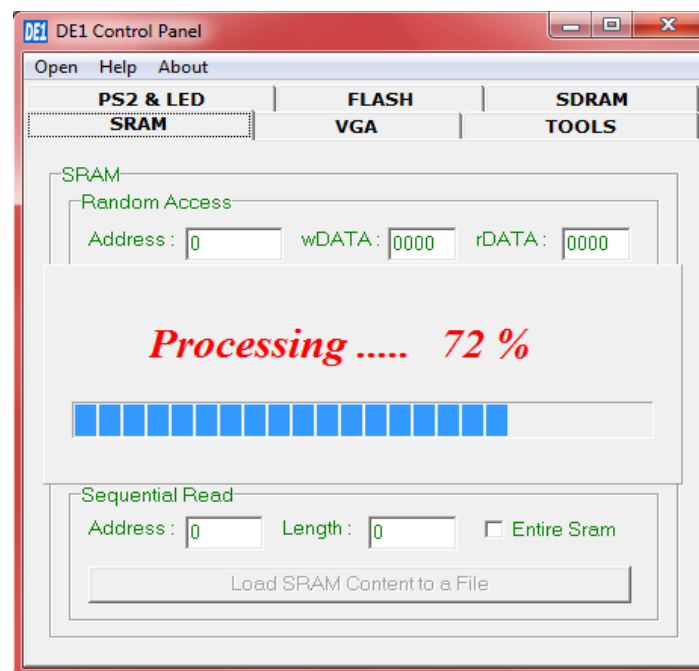


Fig.7.5 Progress of write operation

After embedding is done, the image pixels are stored back in SRAM in a different set of memory locations (say, 8000h to 10000h).

7.2.2. SRAM Read

The values can be read into a text file as follows:

- The starting address is specified in the **Address** box.
- The number of bytes to be copied into the file is specified in the **Length** box.
- **Load SRAM Content to a File** button is pressed.
- When the Control Panel responds with the standard Windows dialog box asking for the destination file, the desired file is specified.

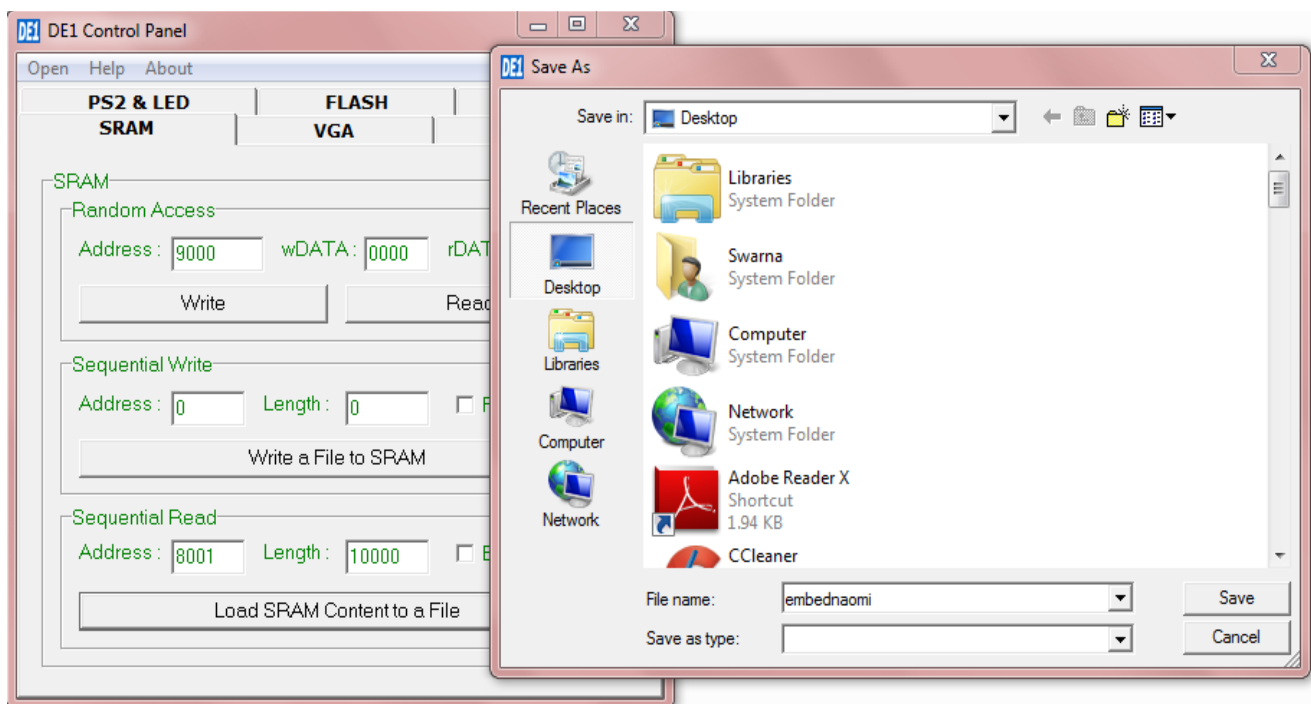


Fig.7.6 Selection of text file for read operation

To verify the 16-bit values from individual addresses, the address is specified and **Read** button is pressed.

Chapter 8

8. Information Hiding

The core functionality of embedding secret in the cover blocks is realized using FPGA because hardware promises better efficiency in comparison to other software platforms. DE1 board is a sterling choice in comparison to other software implementations, owing to its huge RAM Space of 512 Kbytes and its capability for high speed computation of block units.

8.1. Info-Hiding Algorithm - An Illustration

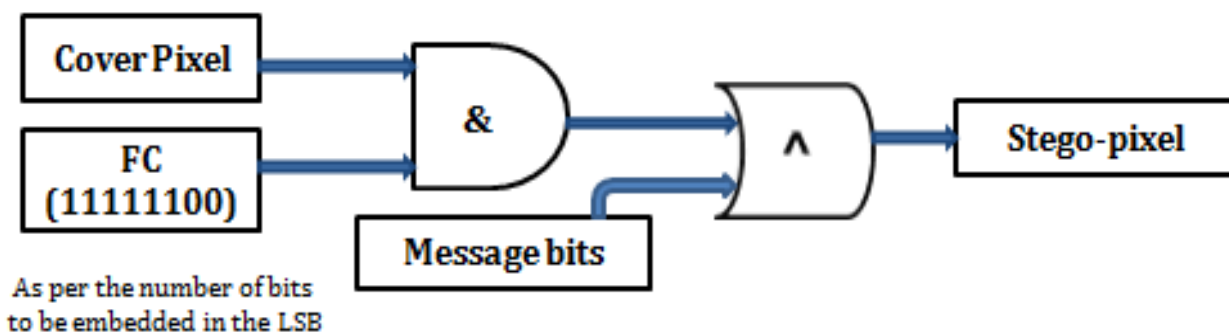


Fig.8.1 Schematic-Information Hiding

Authentication is yet another security measure for protecting confidential data. Authentication substantially reduces the frequency of security breaches.

Each cover block has four pixel values, say, $A_{m,n}$, $B_{m,n}$, $C_{m,n}$ and $D_{m,n}$, where m and n denote the row number and column number of the cover block in a total of 128×128 , 2×2 blocks. The CRC byte is formed by logically EX-ORing the four 8-bit pixel values. The individual bits of the resultant 8-bit value are EX-ORed to obtain the CRC or authentication bit. Thus, authentication is achieved here by parity check in each cover block, to ensure that the message is intact and integrity is maintained.

The message bits are embedded as illustrated in the above schematic. The authentication bit is concatenated to the LSB of one of the four 8-bit pixel values in each cover block.

$$\text{CRC byte} = \text{Ex-OR} (A_{m,n}, B_{m,n}, C_{m,n}, D_{m,n})$$

$$\text{CRC bit} = \text{Ex-OR} (\text{CRC byte (7), CRC byte (6), CRC byte (5), CRC byte (4), CRC byte (3), CRC byte (2), CRC byte (1), CRC byte (0)})$$

The following flow-chart briefly elucidates the procedure adopted in embedding the secret and parity bits. The data stored in SRAM is accessed using DE1 Control Panel Application.

A VHDL code is written for the below algorithm using Altera's Quartus II Design software Version 7.2 web edition and is used for hardware implementation. The hardware used is Altera Cyclone II FPGA EP2C20F484C7 present in the DE1 Board.

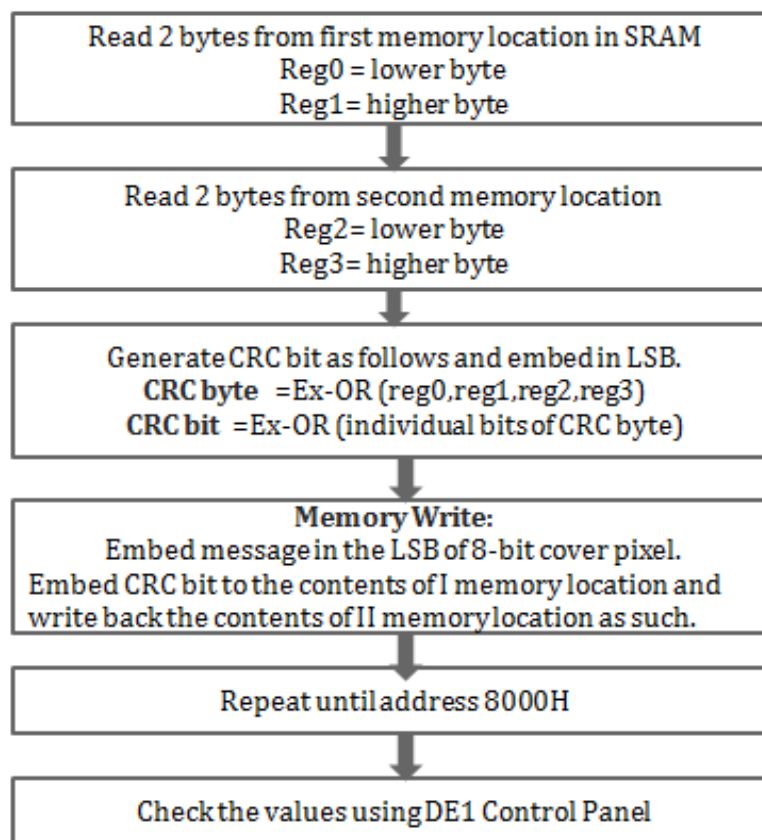


Fig.8.2 Secret Hiding Algorithm

8.2. Design in Quartus II

First, a new project(.qpf file) is created from **File > New Project Wizard**, followed by the specification of working directory of the project and project name. The Design Entry/Synthesis, Simulation and Timing Analysis EDA tools are specified appropriately in the EDA Tools setting page of New Project Wizard.

A new VHDL file(.vhd) is created using **File > New > VHDL File**. After creation of the file, the entity name and file name should be the same. The created file is now in the file section of the Project Navigator Window. It is right-clicked and the option “Set as top level entity” is chosen from the list of options. Now, the appropriate VHDL Code is written to embed secret data and authentication bit in the 2×2 image blocks. The control pins of SRAM are set to appropriate values as demanded by the operation. i.e. the Output Enable, Lower Byte Control, Upper Byte Control and Chip Enable are set to zero as they are active low. Write Enable is High for Reading and Low for Writing operation. A 50 MHz Clock is chosen and Read-Modify-Write operations are performed on the rising edge of the clock.

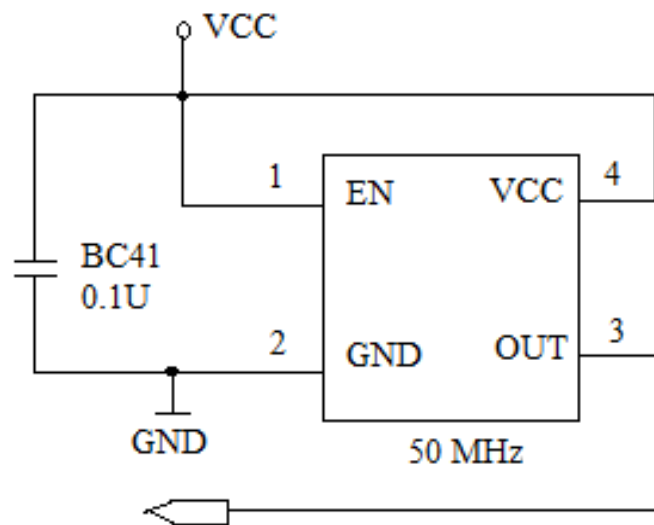


Fig.8.3 Schematic Diagram of Clock Circuit

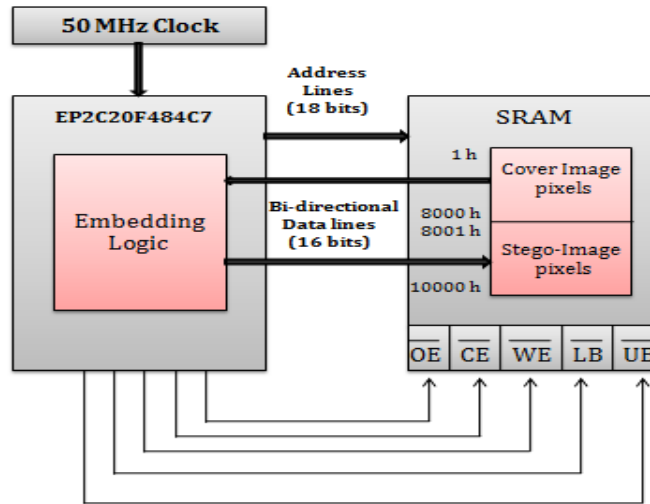


Fig.8.4 VHDL Approach to embedding

8.2.1. Pin Planning

After completion of the VHDL Coding, **pin planning** is done where each element of the entity is assigned appropriate pin values to map the design on the FPGA.

steg.vhd								
Compilation Report - Flow Summary								
steg.cdf								
Pin Planner								
Named: [] Edit: [X] [✓]								
Filter: Pins: all								
	Node Name	Direction	Location	I/O Bank	Vref Group	I/O Standard	Reserved	Group
1	addr[17]	Output	PIN_Y5	8	B8_N1	3.3-V LVTTTL (default)		addr[17..0]
2	addr[16]	Output	PIN_Y6	8	B8_N1	3.3-V LVTTTL (default)		addr[17..0]
3	addr[15]	Output	PIN_T7	8	B8_N1	3.3-V LVTTTL (default)		addr[17..0]
4	addr[14]	Output	PIN_R10	8	B8_N0	3.3-V LVTTTL (default)		addr[17..0]
5	addr[13]	Output	PIN_U10	8	B8_N0	3.3-V LVTTTL (default)		addr[17..0]
6	addr[12]	Output	PIN_Y10	8	B8_N0	3.3-V LVTTTL (default)		addr[17..0]
7	addr[11]	Output	PIN_T11	8	B8_N0	3.3-V LVTTTL (default)		addr[17..0]
8	addr[10]	Output	PIN_R11	8	B8_N0	3.3-V LVTTTL (default)		addr[17..0]
9	addr[9]	Output	PIN_W11	8	B8_N0	3.3-V LVTTTL (default)		addr[17..0]
10	addr[8]	Output	PIN_V11	8	B8_N0	3.3-V LVTTTL (default)		addr[17..0]
11	addr[7]	Output	PIN_AB11	8	B8_N0	3.3-V LVTTTL (default)		addr[17..0]
12	addr[6]	Output	PIN_AA11	8	B8_N0	3.3-V LVTTTL (default)		addr[17..0]
13	addr[5]	Output	PIN_AB10	8	B8_N0	3.3-V LVTTTL (default)		addr[17..0]
14	addr[4]	Output	PIN_AA5	8	B8_N1	3.3-V LVTTTL (default)		addr[17..0]
15	addr[3]	Output	PIN_AB4	8	B8_N1	3.3-V LVTTTL (default)		addr[17..0]
16	addr[2]	Output	PIN_AA4	8	B8_N1	3.3-V LVTTTL (default)		addr[17..0]
17	addr[1]	Output	PIN_AB3	8	B8_N1	3.3-V LVTTTL (default)		addr[17..0]
18	addr[0]	Output	PIN_AA3	8	B8_N1	3.3-V LVTTTL (default)		addr[17..0]
19	ce	Output	PIN_AB5	8	B8_N1	3.3-V LVTTTL (default)		
20	clk	Input	PIN_L1	2	B2_N1	3.3-V LVTTTL (default)		





















21		data[15]	Bidir	PIN_U8	8	B8_N1	3.3-V LVTTTL (default)	data[15..0]
22		data[14]	Bidir	PIN_V8	8	B8_N1	3.3-V LVTTTL (default)	data[15..0]
23		data[13]	Bidir	PIN_W8	8	B8_N1	3.3-V LVTTTL (default)	data[15..0]
24		data[12]	Bidir	PIN_R9	8	B8_N0	3.3-V LVTTTL (default)	data[15..0]
25		data[11]	Bidir	PIN_U9	8	B8_N0	3.3-V LVTTTL (default)	data[15..0]
26		data[10]	Bidir	PIN_V9	8	B8_N1	3.3-V LVTTTL (default)	data[15..0]
27		data[9]	Bidir	PIN_W9	8	B8_N0	3.3-V LVTTTL (default)	data[15..0]
28		data[8]	Bidir	PIN_Y9	8	B8_N0	3.3-V LVTTTL (default)	data[15..0]
29		data[7]	Bidir	PIN_AB9	8	B8_N0	3.3-V LVTTTL (default)	data[15..0]
30		data[6]	Bidir	PIN_AA9	8	B8_N0	3.3-V LVTTTL (default)	data[15..0]
31		data[5]	Bidir	PIN_AB8	8	B8_N0	3.3-V LVTTTL (default)	data[15..0]
32		data[4]	Bidir	PIN_AA8	8	B8_N0	3.3-V LVTTTL (default)	data[15..0]
33		data[3]	Bidir	PIN_AB7	8	B8_N1	3.3-V LVTTTL (default)	data[15..0]
34		data[2]	Bidir	PIN_AA7	8	B8_N1	3.3-V LVTTTL (default)	data[15..0]
35		data[1]	Bidir	PIN_AB6	8	B8_N1	3.3-V LVTTTL (default)	data[15..0]
36		data[0]	Bidir	PIN_AA6	8	B8_N1	3.3-V LVTTTL (default)	data[15..0]
37		lb	Output	PIN_Y7	8	B8_N1	3.3-V LVTTTL (default)	
38		oe	Output	PIN_T8	8	B8_N1	3.3-V LVTTTL (default)	
39		ub	Output	PIN_W7	8	B8_N1	3.3-V LVTTTL (default)	
40		we	Output	PIN_AA10	8	B8_N0	3.3-V LVTTTL (default)	

Fig.8.5 Pin planner

This is followed by synthesis of the VHDL file. Synthesis is the process of conversion of textual Hardware Description Language to schematic form. The synthesis is performed by choosing **Processing > Start > Start Analysis & Synthesis**. After synthesis, the netlist viewers help analyze and debug the design.

Image size	Utilized SRAM memory locations
128 × 128	16384
256 × 256	65536
512 × 512	262144

Table8.1 Utilization of SRAM Memory based on image size

8.2.2. RTL Schematic

RTL Viewer is used to view the initial synthesis results to check if the necessary logic has been created and the register-level transfers, other logic and corresponding connections are interpreted

correctly by the Quartus II software. This helps debug design errors at an earlier stage in design process. RTL Schematic is viewed by choosing **Tools > Netlist Viewers>RTL Viewer**.

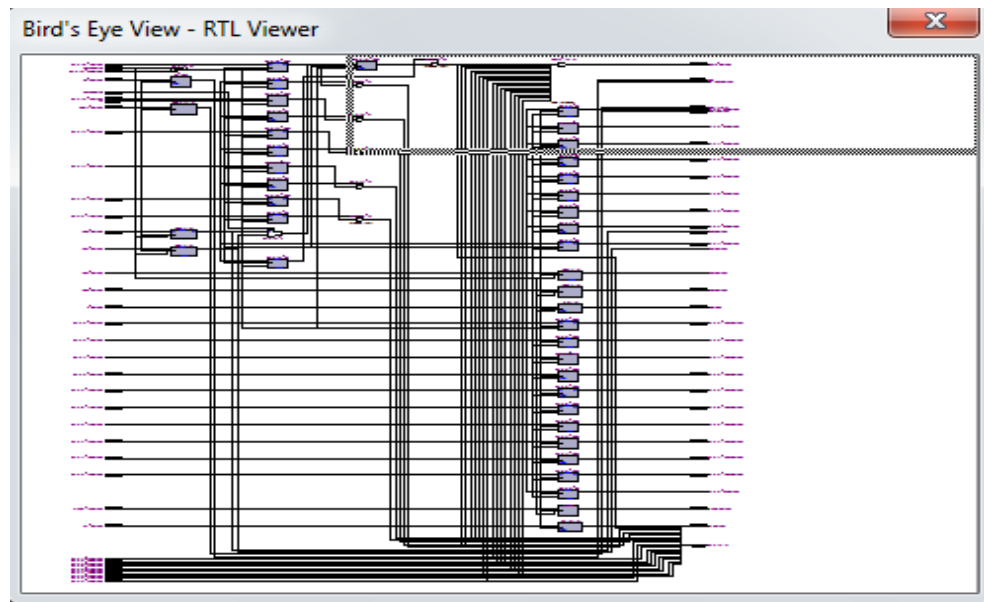


Fig.8.6 RTL Schematic- Bird's eye view

8.2.3. Technology Map Viewer

Technology Map Viewer results are collected to view the end results of Analysis and Synthesis.

This viewer elucidates the design with a technology-specific, graphical representation.

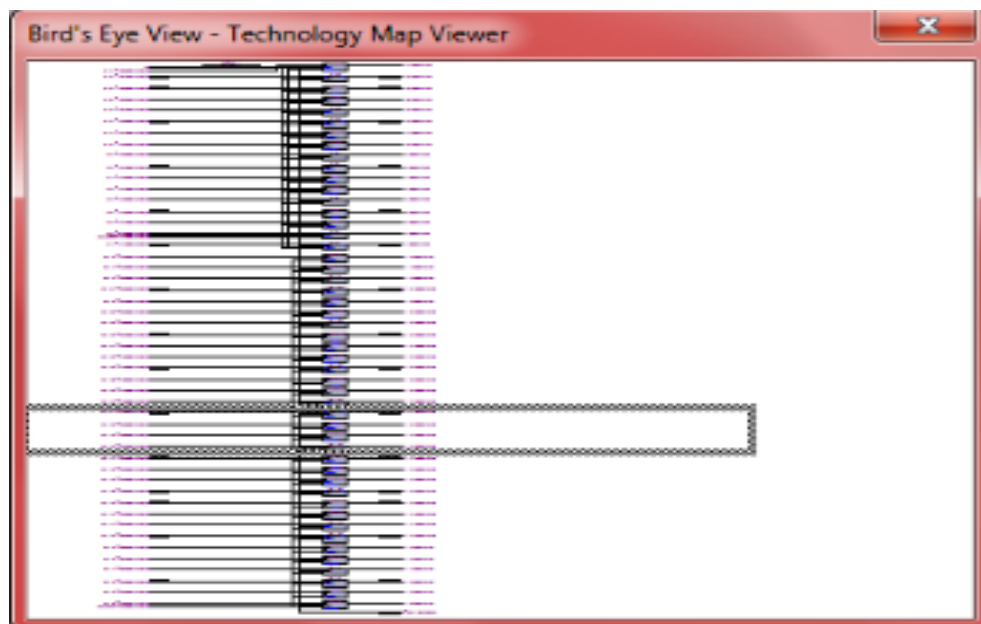


Fig.8.7 Technology Map Viewer- Bird'd eye view

8.2.4. Synthesis Report

The synthesis report shows that hardware consumption is less when performing steganography on even large sized grayscale images. The image data is stored in external SRAM and not Block RAM.

Total Memory bits in External SRAM = 256 K x 16 bits

Total logic Elements = 18,752

Total Combinational functions = 18,752

Available Dedicated logic registers = 18,752

Total pins = 315

8.2.5. Compilation Report- Flow Summary

The design is compiled by choosing **Processing > Start Compilation** and compilation report is as follows:

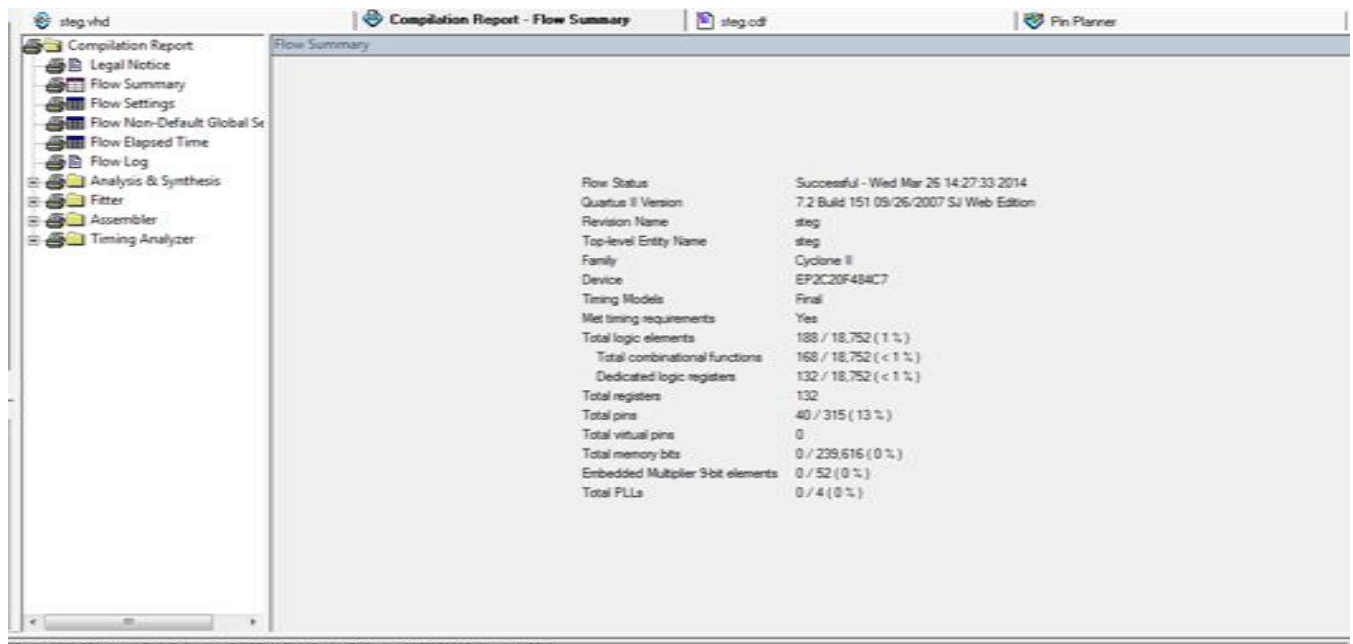


Fig.8.9 Compilation Summary

Image Size	Utilized logic elements	Utilized combinational functions	Utilized dedicated logic registers	Total pins used
128 × 128	191	171	132 (< 1 %)	40 (13%)
256 × 256	188	168	132 (<1 %)	40 (13%)
512 × 512	186	166	132 (< 1%)	40 (13%)

Table 8.2 Flow summary for various image sizes

8.2.6. Programmer

After successful compilation, an SRAM Object File (.sof) is created. The design is now ready for hardware implementation. The programmer can be selected from **Tools > Programmer. Add File** option is present to add .sof file. **Start** button is pressed to burn the program into the FPGS. USB Blaster is used for FPGA programming. The DE1 board does not require additional power source for programming as it takes the power from USB cable itself.

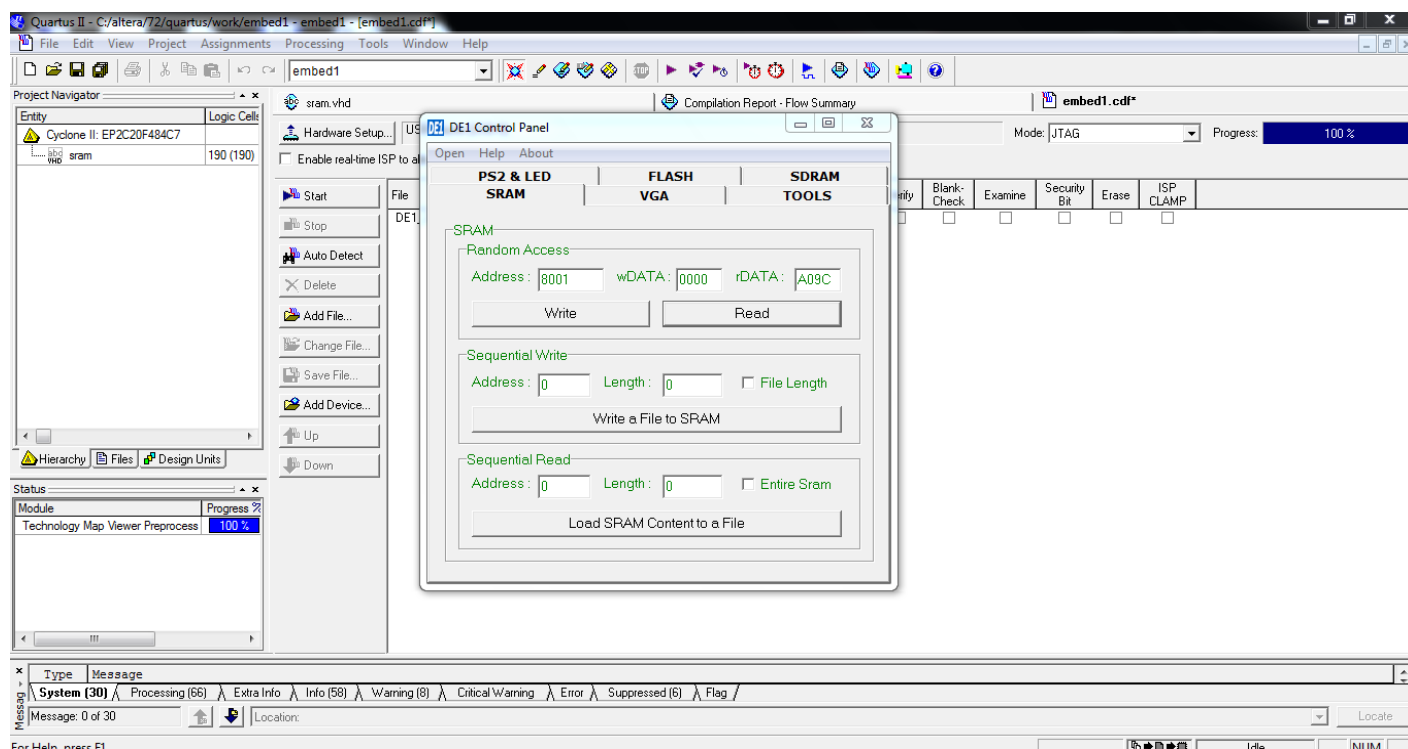


Fig.8.10 Output Verification using Control Panel Application

The Altera USB Blaster has to be installed during the first time of programming the device. In subsequent device programming, the Altera USB Blaster will be automatically recognized by the system (Personal Computer). The programming mode is JTAG. After embedding, the stego-blocks in SRAM are verified using Control Panel application and then stored in a text file.

Chapter 9

9. Image Reconstruction

9.1. Procedure

The image is to be reconstructed from the stego-blocks stored in the text file. The following steps are followed to achieve the reconstructed image:

- The text file is read in LabVIEW and converted to an equivalent decimal array. This array contains pixel values in form of blocks, divided column-wise. The blocks should be reshuffled to get the correct image array.
- Let $s = m \times n$, be the size of the image. Here n denotes the number of rows and columns. Since 2 bytes occupied one memory location when data was stored in SRAM, the resultant decimal array of blocks is divided into two arrays, one containing elements of even columns and the other, odd columns. i.e. first two values (2 bytes) are stored in even column array, next two (2 bytes) in odd column array, next two in even and so on, until all 's' values are reached.

Even column array is extracted using the formula: $4i$ and $4i+1$, where 'i' denotes the loop iteration.

So, when $i=0$, the 0^{th} and 1^{st} element are extracted (i.e., the first two bytes),

when $i=1$, the 4^{th} and 5^{th} element are extracted and so on.

Example:

For a 256×256 image, $s=65536$, $m=256$ and $n=256$.

So, the even column array contains elements (0, 1), (4, 5), (8, 9), (12, 13)..... where 'i' ranges from 0 to 32768.

Odd column array is extracted using the formula: $4i+2$ and $4i+3$.

Example:

For a 256×256 image, the odd column array contains elements (2, 3), (6, 7), (10, 11)...where $0 \leq i \leq 32768$.

The even column and odd column array contain 32768 elements each.

Alternatively, this can be done in two steps too. Firstly, the decimal block array is divided into two arrays, one containing the even elements (1 byte each) and the other containing odd elements (1 byte each). Then the 0^{th} , 2^{nd} , 4^{th} ...etc. elements (even position values) from both even and odd arrays are stored in **even column array**. Then 1^{st} , 3^{rd} , 5^{th} ,...etc. elements (odd position values) from both arrays are stored in **odd column array**.

- Now, 256 values from even and odd column arrays are alternately stored in a new array, until all 's' values are stored.
- The new array, which is a 1-D array, is converted to 2-D array of 256 rows and 256 columns.
- This array has elements arranged column-wise. In order to obtain the final image array, this array has to be transposed.
- IMAQ Vision tools aid in displaying the image from this final image array.
- The Mean Square Error and Peak Signal to Noise Ratio are computed for both the original and reconstructed image and the results are compared.

$$MSE = \frac{1}{m \times n} \sum_{i=1}^m \sum_{j=1}^n (x_{ij} - y_{ij})^2$$

where, MSE is the mean square error,

$m \times n$ is the size of the cover image,

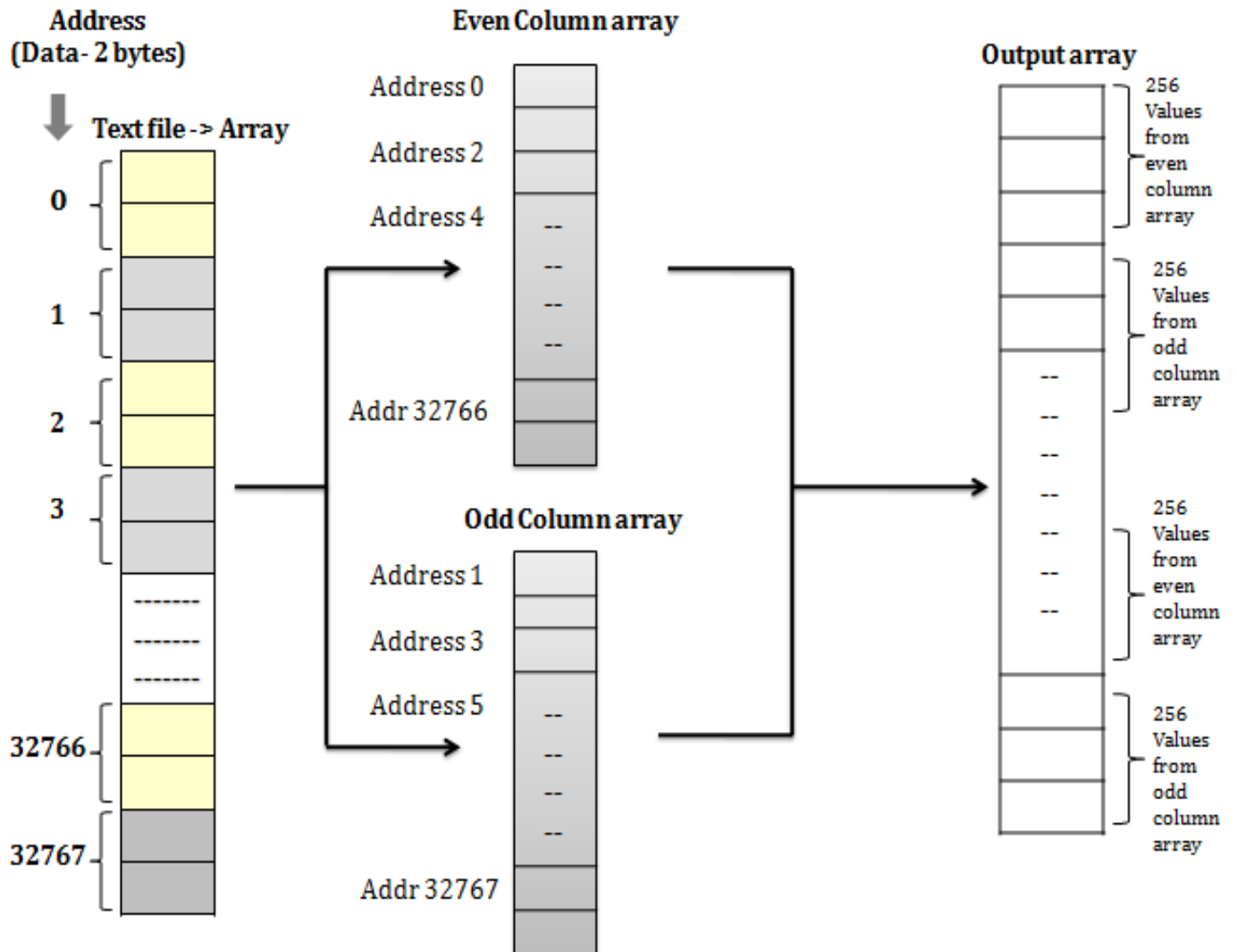
x_{ij} refers to cover pixel values and

y_{ij} refers to stego pixel values.

$$PSNR = 10 \times \log_{10} \frac{y_{max}^2}{MSE} dB$$

where, PSNR is the peak signal-to-noise ratio, which is an important criterion for the visual quality of stego-images.

y_{max} is 255 for a 256 x 256 image.



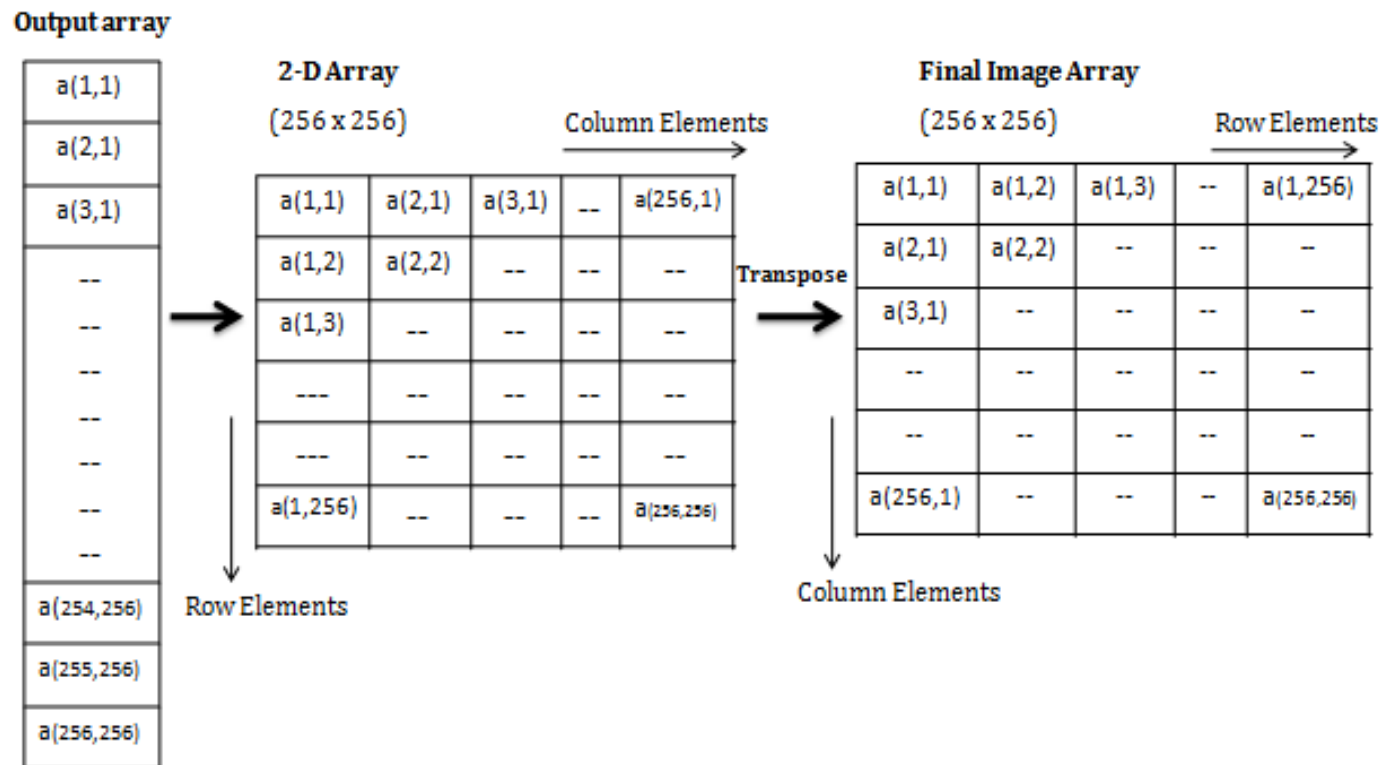


Fig.9.1 Reconstruction Procedure in a nutshell

Chapter 10

10. Results

10.1. LabVIEW Simulation Results

10.1.1. Division of image pixels into 2×2 blocks and storing in text file

The following snapshot shows the original cover image and its histogram before division into 2×2 blocks.

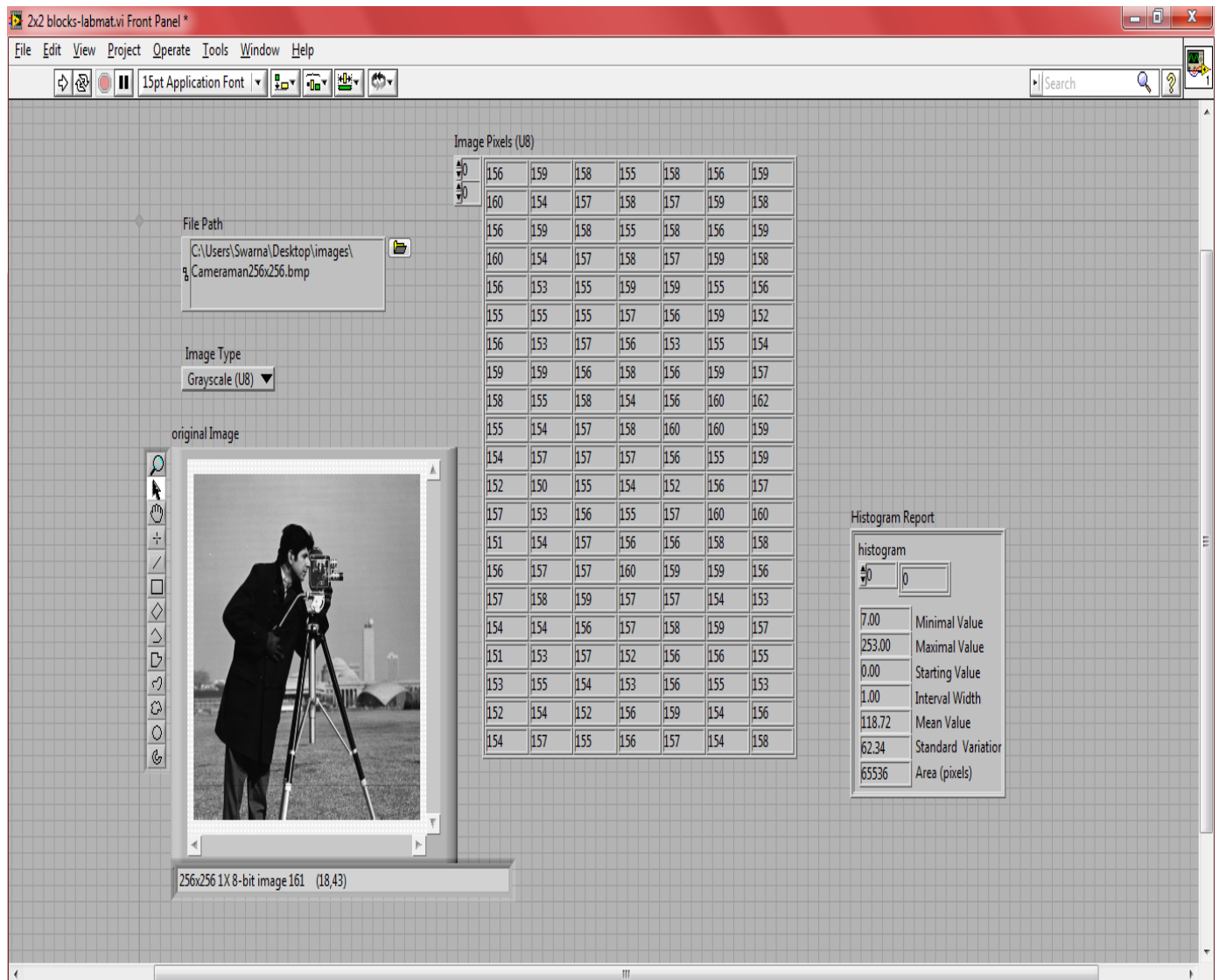


Fig.10.1

10.1.2. Reconstruction of image after embedding using FPGA

The image reconstructed from the blocks and its corresponding histogram are shown in the below figure.

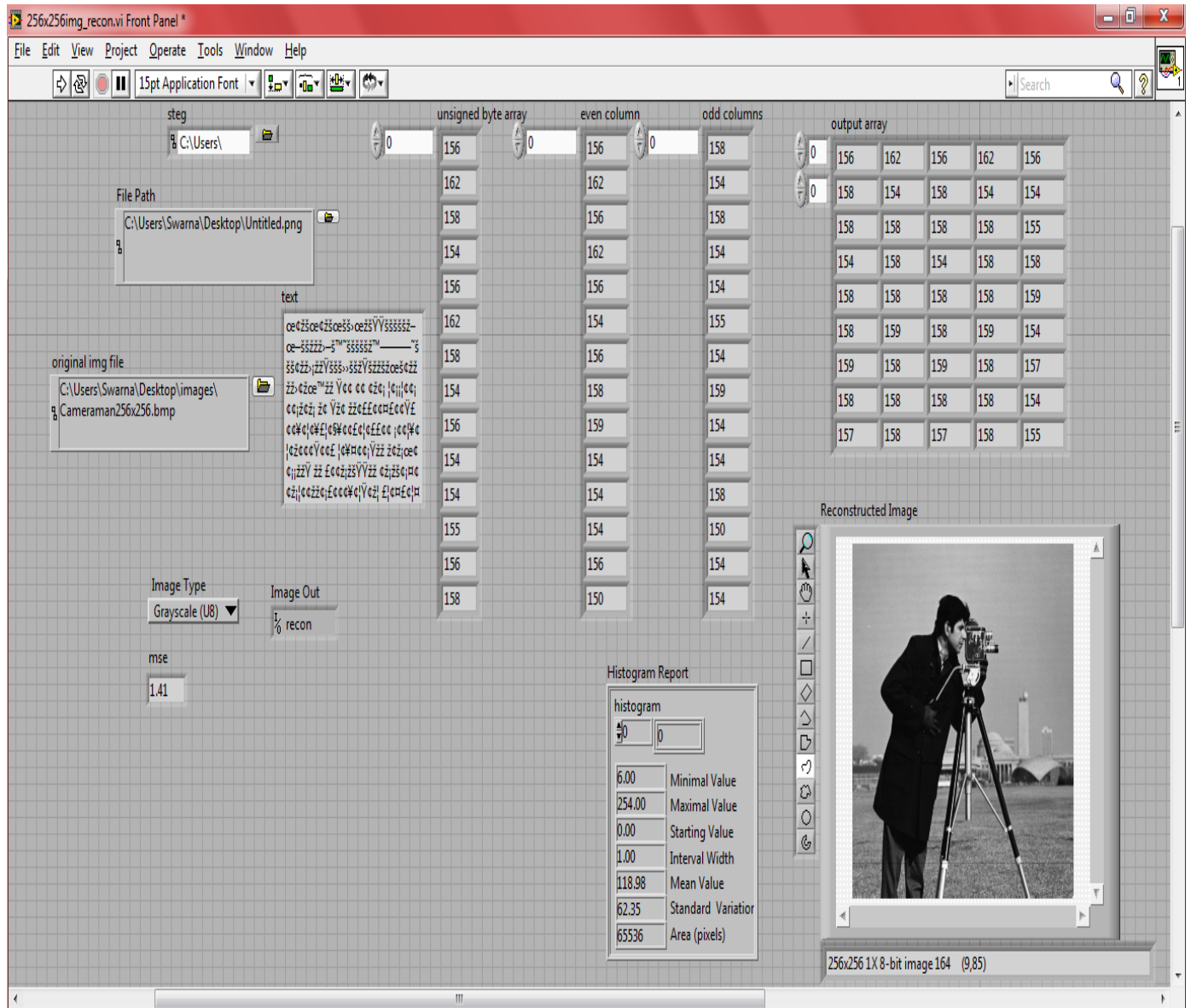


Fig.10.2

As observed from the two images, the reconstructed image is hardly any different from the original image with a very low MSE of 1.4. Thus, the presence of secret is concealed efficiently by this methodology.

10.2. Mean Square Error and PSNR values of test images

Mean Square error and PSNR values are calculated for a set of test images. Also, histogram reports are generated for both cover image and stego-image using LabVIEW. The image histogram represents the number of pixels corresponding to each tonal value. Tonal value is the distinguishable variation from white to black i.e. each discernible gray level.

It is also indicative of contrast. Narrow histograms denote low contrast whereas broad ones imply high contrast.

The detailed report displays Mean value and Standard Deviation. The mean value returns the mean of all the pixel values considered. Standard deviation is yet another statistical parameter denoting the deviation from average value. Higher standard deviation values connote better distribution of values in the image.

STANDARD IMAGES

Fig .10.3: **Image Name :** Lena

Size : 128 x 128

MSE : 0.44

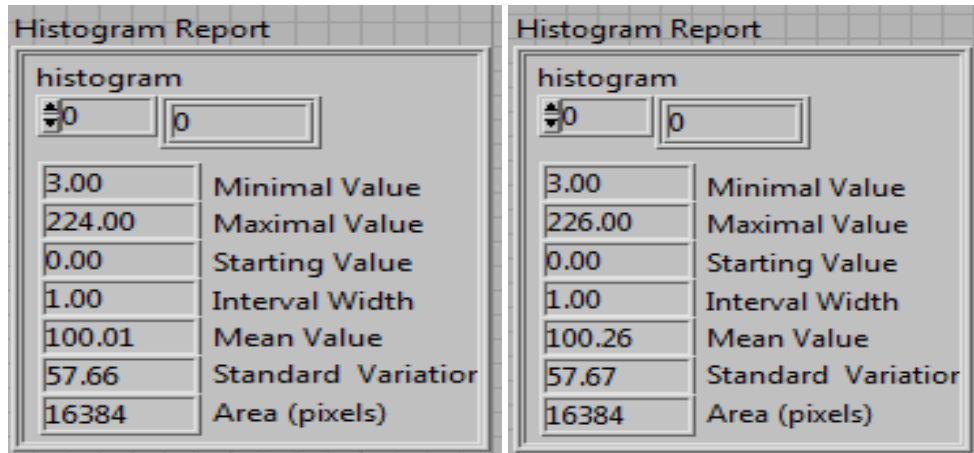
PSNR : 45.6415dB



a) Original Cover Image

b) Reconstructed Stego Image

Histogram Report



c) Cover image

d) Stego Image

Fig : 10.4: Image Name : Pepper

Size : 256 x 256

MSE : 0.89

PSNR : 48.636dB

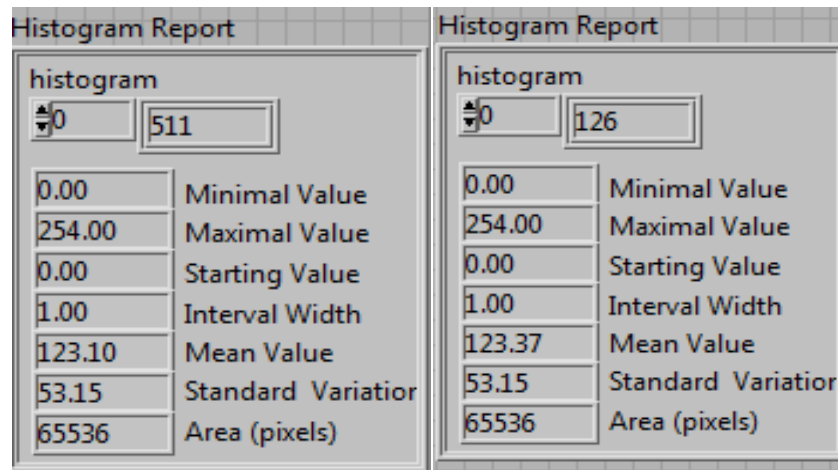


a) Original Cover Image



b) Reconstructed Stego Image

Histogram Report



c) Cover Image

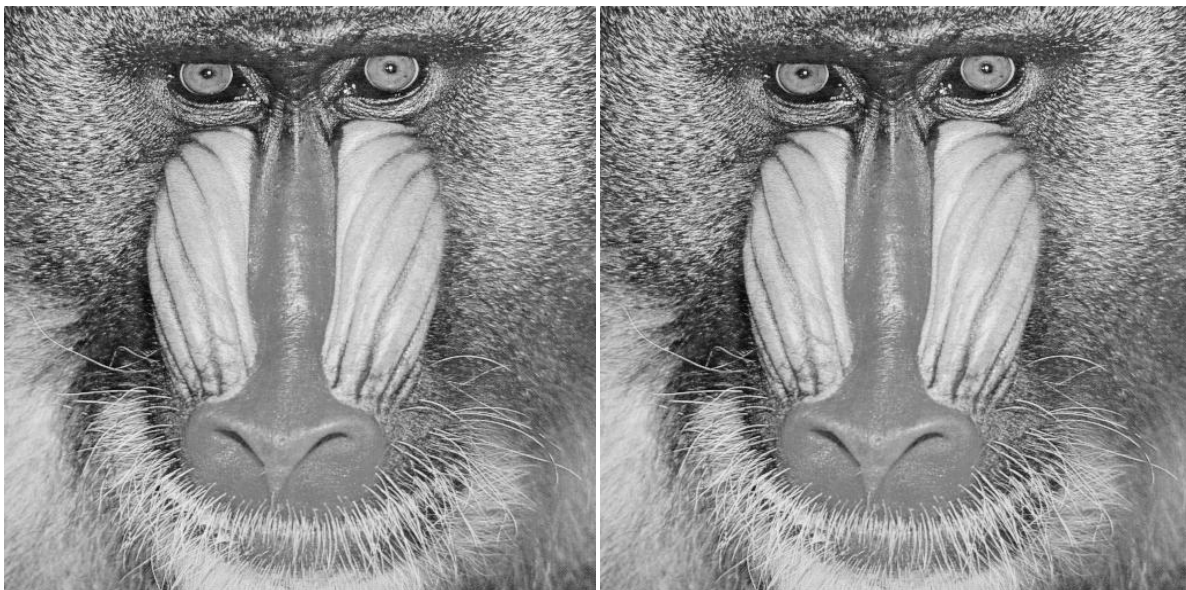
d) Stego Image

Fig : 10.5: **Image Name :** Baboon

Size : 512 x 512

MSE : 1.43

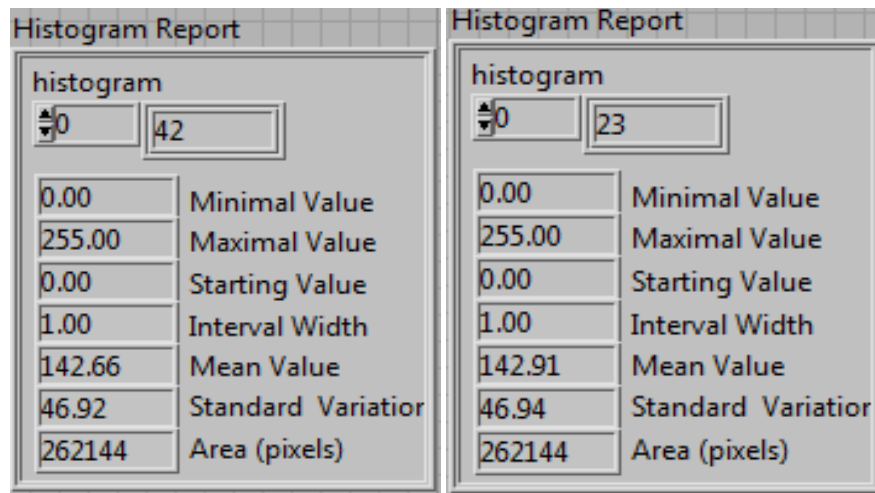
PSNR : 52.615 dB



a)Original Cover Image

b) Reconstructed Stego Image

Histogram Report



c) Cover Image

d) Stego Image

NON-STANDARD IMAGE

Fig : 10.6: Image Name : Kristen

Size : 256 x 256

MSE : 1.19

PSNR : 47.37dB

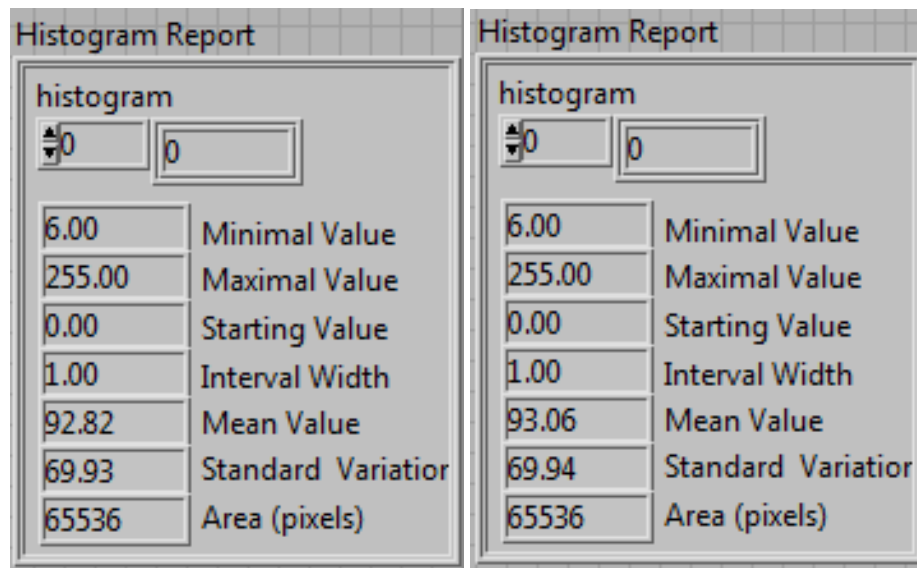


a) Original Cover Image



b) Reconstructed Stego Image

Histogram Report



c) Cover Image

d) Stego Image

10.3. Timing Analysis

Timing analysis is an important step in validating the performance of a design. If timing violations go unnoticed, they may lead to incorrect circuit operation.

So, the overall design operation is to be tested by using both functional simulation and timing analysis.

The below figure offers a comprehensive timing analysis report with t_{su} , t_h and t_{co} values. The **Clock setup time** (t_{su}) is the time duration for which data must be present at the input of the register that it feeds through its data or enable inputs, before the clock signal that clocks the register is asserted at the clock pin. **Clock hold time** (t_h) is the time duration for which data

should be retained at the input pin of a register, after clock signal clocking the register is asserted at the clock pin. **Clock-to-Output Delay** (t_{CO}) is the maximum time required to get a valid output at the output pin of a register after the clock transition at the input pin that clocks the register.

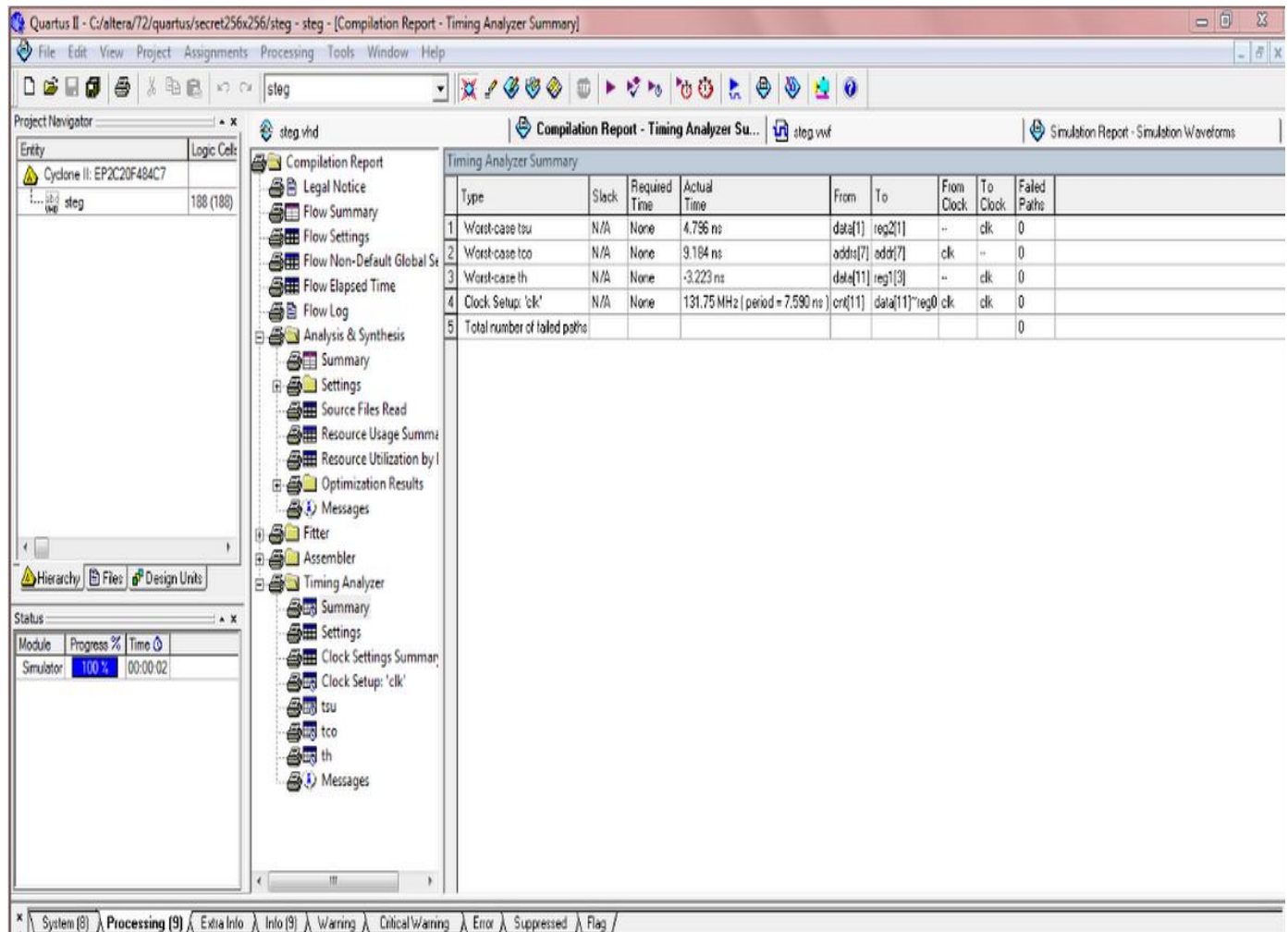


Fig.10.7 Timing Analysis Summary

Additionally, a vector waveform file (.vwf) can be created to draw the test signals for simulation and to choose which signals should be shown in the simulation results.

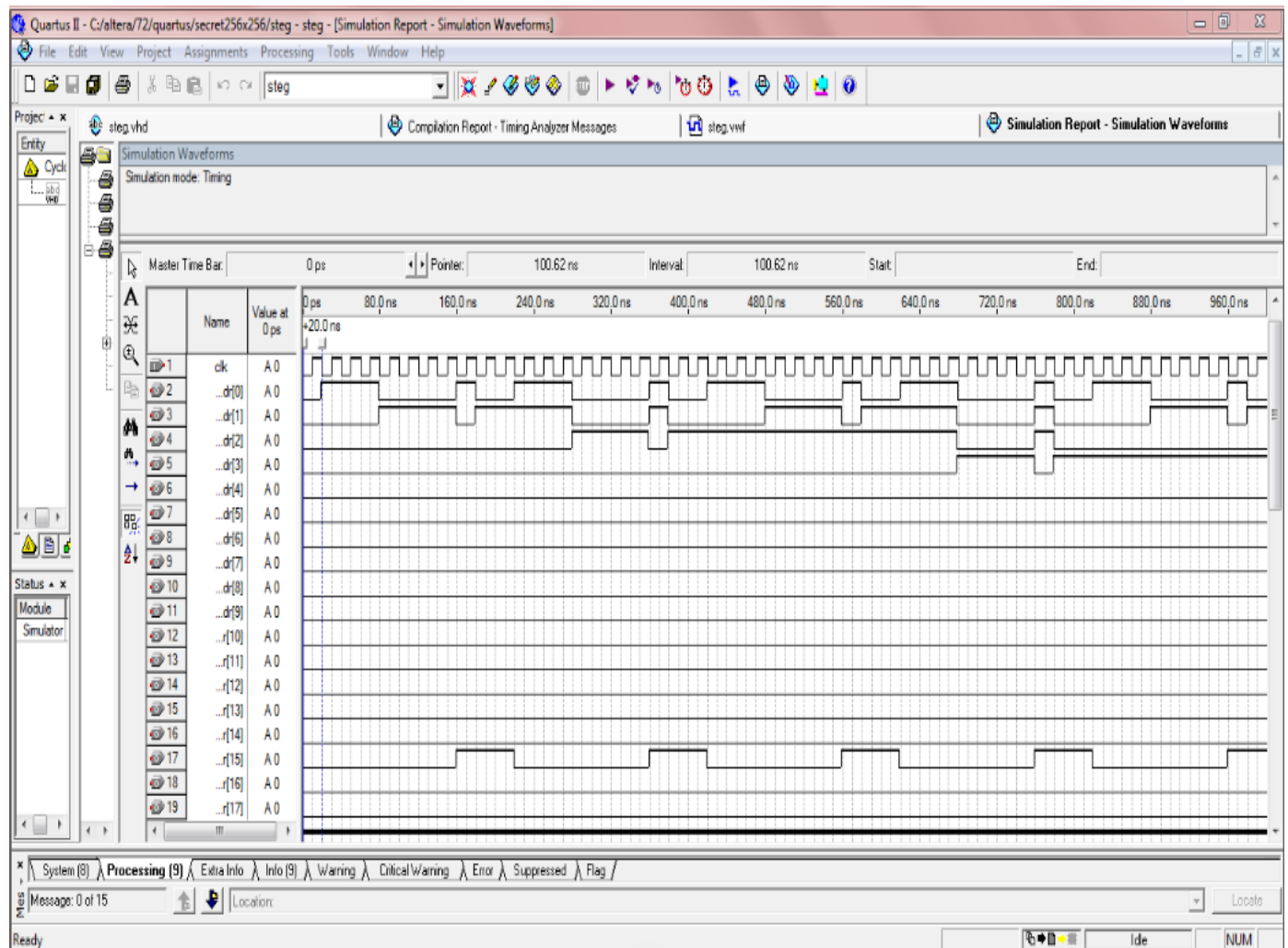


Fig.10.8 Vector Waveform file

Timing Report

Master clock frequency = 50 MHz

Clock cycles required to hide 2 bit = 2×3 cycles of 50 MHz = 120 ns (60 ns \times 2)

To embed 5 bits in 3 pixels, 300 ns is required.

Number of 2×2 blocks = 16384

Total time taken for embedding information = 16384×300 ns = 4.9 ms

10.4. Conclusion

This methodology of steganography offers a very high message capacity and very little likelihood for degradation of the carrier, i.e. cover image. For a 256 x 256 image, each cover block can hold a maximum of 15 message bits and 1 authentication bit. So the carrier has a message capacity of 2,45,760 (16384 x 15) message bits in total. As the algorithm is implemented on reconfigurable hardware, the overall design time and cost of implementation is significantly lower than when implemented by Application Specific Integrated Circuits (ASICs).

The proposed scheme ensures good optimization of hardware and software modules available and provides for fast operation. The stego-images yielded by our algorithm has excellent imperceptibility, which is proven by the calculated PSNR and MSE values. Complementing the functionality is its superior timing performance, supported by the timing analysis report. This method can also be associated with other domain steganographic approaches. Further additions to this project can include hiding multiple secret images in a single cover image or inclusion of some feature to assure message non-repudiation.

References

1. Chin-Chen Chang, Yi-Pei Hsiehb, Chia-Hsuan Lina, Sharing secrets in stego images with authentication, Pattern Recognition 41 (2008) 3130-3137.
2. Chi-Kwong Chan, L.M. Cheng, Hiding data in images by simple LSB substitution, Pattern Recognition 37 (2004) 467-474.
3. Abbas Cheddad, Joan Condell, Kevin Curran, Paul McKevitt, Digital image steganography: Survey and analysis of current methods, Signal Processing 90 (2010) 727-752
4. R.Sundararaman, Dr.Har Narayan Upadhyay, Stego system on Chip with LFSR based Information Hiding Approach, International Journal of Computer Applications (0975-8887), Volume 18-No.2, March 2011
5. Sundararaman Rajagopalan, Siva Janakiraman, Har Narayan Upadhyay,K.Thenmozhi, Hide and seek in silicon – Performance analysis of Quad block Equisum Hardware Steganographic systems, Procedia Engineering 30 (2012) 806-813
6. Mehdi Hussain, Mureed Hussain, A Survey of Image Steganography Techniques, International Journal of Advanced Science and Technology, Vol.54, May 2013
7. <http://userpages.umbc.edu/~tbenja1/umbc7/santabar/vol1/lec2/2-3.html>