



Department of Electronic Engineering

FINAL YEAR PROJECT REPORT

BEngCE-2006/07-<LMC>-<01>

<Digital Watermarking Device>

Student Name: Leung Hon Yin

Student ID:

Supervisor: Dr. L M Cheng

Assessor: Dr. L L Cheng

Bachelor of Engineering (Honours) in
Computer Engineering

Student Final Year Project Declaration

I have read the student handbook and I understand the meaning of academic dishonesty, in particular plagiarism and collusion. I declare that the work submitted for the final year project does not involve academic dishonesty. I give permission for my final year project work to be electronically scanned and if found to involve academic dishonesty, I am aware of the consequences as stated in the Student Handbook.

Project Title: Digital Watermarking Device

Student Name: Leung Hon Yin

Student ID:

Signature:

Date: 26-4-2006

No part of this report may be reproduced, stored in a retrieval system, or transcribed in any form or by any means – electronic, mechanical, photocopying, recording or otherwise – without the prior written permission of City University of Hong Kong.

Acknowledgements:

I need to express my thank to my supervisor, Dr. L .M Cheng and my assessor Dr. L. L Cheng and Dr. C K Chan who guided me through out this year. Without their strong support, I cannot fully understand those algorithms and complete this work. And I also need to thank my family.

Finally, I would like to thank MaCaPS staff, who helped me solve many technical problems.

Table of Content:

Acknowledgements:.....	4
Abstract:.....	7
1. Introduction.....	8
2. Aims and Objectives.....	9
2.1 Project Aims.....	9
2.2 Project Objectives.....	9
3. Project Environment.....	10
3.1 Project Scope.....	10
3.2 Development Tools.....	11
4. Background.....	12
4.1 Background of Watermarking.....	12
4.1.1 Definition of Watermarking ^[3]	12
4.1.2 Watermarking Applications ^[3]	12
4.1.3 Requirement of Digital Watermarks ^[3]	13
4.1.4 Principle of Watermarking.....	14
4.1.5 Performance Evaluation of Watermark System ^[8]	15
4.2 Algorithm of a novel image hiding scheme based on block difference ^[1]	17
4.2.1 Embedding Process ^[1]	17
4.2.2 Extracted Process ^[1]	18
4.3 Algorithm of Data Hiding in Images by Adaptive LSB Substitution Based on the Pixel-Value Differencing ^[2]	19
4.3.1 Data Embedding ^[2]	19
4.4 NTSC format.....	23
4.5 ITU-R BT 656 Video Data Format.....	23
4.5.1 Introduction.....	23
4.5.2 Common signal format of the interfaces.....	23
4.6 VGA Video Signal ^[8]	26
4.7 Xilinx XUP Virtex-II Pro Development Board ^[8]	27
5. Software Evaluation.....	29
5.1 User Interface.....	31
5.2 Evaluation of algorithm (A novel image hiding scheme based on block difference).....	32
5.3 Evaluation of Algorithm (Data Hiding in Images by Adaptive LSB Substitution Based on the Pixel-Value Differencing).....	33
5.4 Selection of different watermarking techniques.....	35
6. Hardware Implementation.....	36
6.1 Hardware Equipment.....	37
6.2 Block diagram of Watermarking Hardware.....	38
6.2.1 Flow in the block diagram.....	39
6.3 Software of Xilinx ISE 8.2i and ImPACT.....	41
6.3.1 Xilinx ISE 8.2i.....	41
6.3.2 ImPACT.....	43
6.4 The Hardware Flow-Diagram.....	44
6.5.1 Video Capture Module.....	45
6.5.2 If_decode module.....	46
6.5.3 Line Buffer Module.....	46
6.5.4 Neg_Edge_Detect Module.....	47

LMC 01

6.5.5 Pipe_Line Delay Module	47
6.5.6 Special SVGA Timing Generation Module	48
6.5.7 Watermark Rom Module	48
6.5.8 Vp422_444_dup module.....	49
6.5.9 YCrCb2RGB module.....	50
6.5.10 Encode module.....	50
6.5.11 Decode module	51
6.6 Demonstration of Watermarking	52
6.7 Hardware Implementation with three selected algorithms	55
6.7.1 Algorithm of LSB substitution based on predictive approach.....	55
6.7.2 Algorithm of 2 LSB substitution watermarking	57
6.7.3 Algorithm of data hiding in images by adaptive LSB substitution based on the pixel-value differencing	59
7. Discussion	61
7.1 Difficulty Encountered.....	61
7.2 Contribution	62
7.3 Limitation.....	62
8. Further Development	63
9. Conclusion	63
10. Reference	64
11. Appendix.....	65

Abstract:

With the recent rapid growth of networked multimedia systems, digital watermarking techniques have widely been used in high security documents and for integrity checking.

Digital watermarking is a technique to add hidden copyright notices or secret messages to digital audio, video, or image forms.

This project was divided into two parts. In the first part, two watermarking algorithms "A novel image-hiding scheme based on block difference" and "Data Hiding in Images by Adaptive LSB Substitution Based on the Pixel-Value Differencing" have been studied and implemented on software. These two algorithms are both lossless image hiding techniques and produce good image quality of the stego-image. In the second part, FPGA was used to implement digital watermarking techniques with the video signal.

Furthermore, Visual C#.net and Verilog HDL have been used to implement this project.

1. Introduction

Today, audio, video, image or many other things can be represented in digital form. It is much easier for someone to make a perfect copy, which will lead to extensive unauthorized copying which may weaken the music, film, book and software publishing industries. Sometimes forgery digital media or documents become an important issue for identification, authentication and law enforcement. These concerns triggered many researches to find ways to hide the copyright messages and serial number into digital media.

Digital watermarking is one of the data hiding techniques. There are diverse applications of digital watermarking -from counterfeiting and piracy deterrence, media management and identification and authentication to monitoring and mobile e-commerce. This project aims to develop digital watermarking device for authorization and copyright identification.

2. Aims and Objectives

2.1 Project Aims:

This project aims to develop digital watermarking device for authorization and copyright identification. This device consists of three components - a FPGA board, a VCD player and a display monitor. The VCD player provides the video signal to the FPGA. And the FPGA performs the watermarking process by encoding a watermark signal to the video signal in each frame securely and then transfer the video signal to the monitor.

2.2 Project Objectives:

1. Investigate feasible watermarking techniques that can be implemented in real time for document, image and video applications
2. Use Field Programmable Logic Gate (FPGA) to implement these watermarking techniques
3. Use video devices for testing the performance

3. Project Environment

3.1 Project Scope:

This project is to develop a dedicated digital watermarking device based on FPGA, for implementing the watermarking algorithm to the video signal.

The scope of work will include the followings:

1. Study the background of watermarking
2. Study the latest watermarking algorithm
3. Verify the watermarking algorithm with high-level language
4. Study the Verilog HDL syntax
5. Study the Xilinx XC2VP3O development board datasheet
6. Study the use of Xilinx ISE 8.0
7. Implement the algorithm by Verilog HDL
8. Study the video data format ITU-R BT656
9. Study the VGA timing
10. Configure the FPGA and show the watermarked image on VGA Monitor

3.2 Development Tools:

The software using in this project includes Microsoft Visual Studio 2005, Hardware Description Language (Verilog), Matlab (for initial study), Quartus 6.0 (for initial study), Xilinx ISE 8.2i, while the hardware includes Xilinx XC2VP30 development board, Video Decoder 1 board, VCD player with NTSC standard, Video cable, Jtag cable, Monitor and the power cable of the board.

4. Background

4.1 Background of Watermarking

4.1.1 Definition of Watermarking^[3]:

Digital watermarking means embedding information into digital material in such a way that it is imperceptible to a human observer but easily detected by computer algorithm. A digital watermark is a transparent, invisible information pattern that is inserted into a suitable component of the data source by using a specific computer algorithm.

There are two type of watermark including Digital watermarks are signals added to digital data (audio, video, or still images) that can be detected or extracted later to make an assertion about the data.

4.1.2 Watermarking Applications^[3]:

Digital watermarking is a feasible method for the protection of ownership rights of digital media such as audio, image, video and other data types. The application includes digital signatures, fingerprinting, broadcast and publication monitoring, copy control, authentication, and secret communication.

As a signature, the watermark identifies the owner of the content and can be used as a fingerprint to identify content consumers. For example, a specific watermarking technique is planned to be used to secure passports against counterfeiting in the United States. Broadcast and publication monitoring describes the area of computer systems which automatically monitors television and radio broadcast to track the appearance of distributed material. Several commercial systems already exist that make use of this technology. The watermark is often designed in such a way that any alteration either

LMC 01

destroys the watermark or creates a mismatch between the content and the watermark, which can easily be detected.

4.1.3 Requirement of Digital Watermarks^[3]:

Digital watermarks can be measured on the basis of certain characteristics and properties that depend on the type of application. These characteristics and properties include the difficulties of notice, the survival of common distortions and resistance to malicious attacks, the capacity of bit information, the coexistence with other watermarks, and the complexity of the watermarking method.

In general, they are described as fidelity, robustness, fragility, tamper resistance, data payload, complexity, and other restrictions. Digital watermarks must fulfil the following requirements.

Robustness:

It may not be possible without knowledge of the watermark algorithm or secret key to remove the watermark or to make it illegible. Robustness means the resistance ability of the watermark against the watermark attacks or modifications made to the original file. After modifications, resizing, file compression, rotation, and common operations, the watermark can still be detected and demonstrate a good quality.

Non-perceptibility:

It means that the brought bit sample of the watermark does not produce perceptible changes acoustically or optically. A perfect non-perceptible bit sample is present if data material marked with watermark and the original cannot be distinguished from each other.

Non-detectable:

It is always true that brought watermark information in data material is non-detectable if it is consistent with the origin data.

Undeletable:

The watermark must be hard to remove or even unable to remove by any attackers.

Complexity:

Complexity describes the cost to detect and encode the watermark information. One of measurement technique could be the amount of time. It is a good design to make watermarking algorithm and procedure as complex as possible.

Capacity:

Capacity refers to the amount of information that can be stored in a data source.

Unambiguous:

The extracted watermark is equivalent to the embedded watermark.

A trade-off has to be taken between the above-mentioned criteria for an optimal watermarking application.

4.1.4 Principle of Watermarking:

All watermarking techniques share the same generic build blocks: a watermark embedding system and a watermark decoder system.

As for the above example, the secret image S embeds into the original image L through the encoder and watermarked image L' is created. The secret image can be extracted from the watermarked image through the decoder. Different watermarking methods can be adopted in the encoder and decoder.

LMC 01

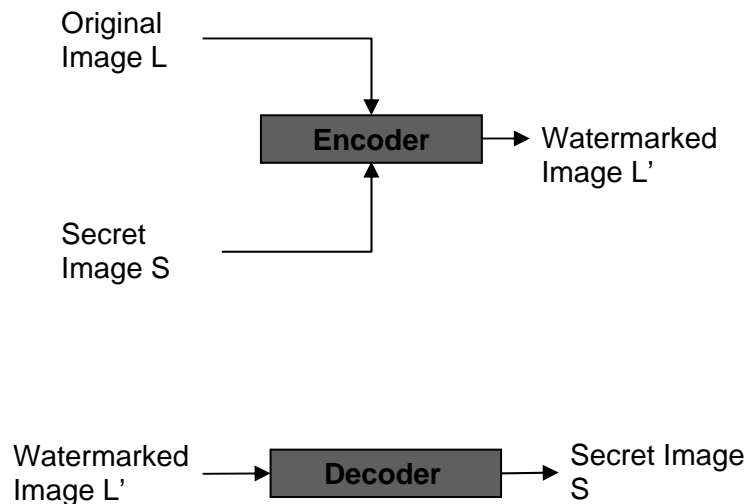


Figure 1: Watermarking for Images

4.1.5 Performance Evaluation of Watermark System ^[8]:

The robustness of watermarks usually depends on:

- 1) Amount of embedded information: more data need to embed, the lower the watermark robustness.
- 2) Watermark embedding strength and size: A trade-off need to be made between the watermark embedding strength and watermark perceptibility.
- 3) Size and nature of data: Size will directly affect the robustness of the embedded watermark

There exists the trade-off between the watermark perceptibility and the watermark robustness.

In order to evaluate the perceptibility of the watermark, subjective tests or a quality metric can be used. The most common use of quality metric for distortion measures in the field of image and video coding and compression are the signal-to-noise ratio (SNR), and the peak signal-to-noise ratio (PSNR). It may be useful to use those distortion metric adapted to the human visual and auditory system.

LMC 01

Signal-to-noise ratio: $SNR = \frac{\sum_{x,y} p^2_{x,y}}{\sum_{x,y} (p_{x,y} - \tilde{p}_{x,y})^2}$

Peak signal-to-noise ratio: $PSNR = XY \max_{x,y} p^2_{x,y} / \sum_{x,y} (p_{x,y} - \tilde{p}_{x,y})^2$

Where $p_{x,y}$ represents a pixel whose coordinates are (x, y), $\tilde{p}_{x,y}$ represents a pixel whose coordinates are (x, y) in the watermarked image. X and Y are the number of rows and columns respectively.

4.2 Algorithm of a novel image hiding scheme based on block difference^[1]:

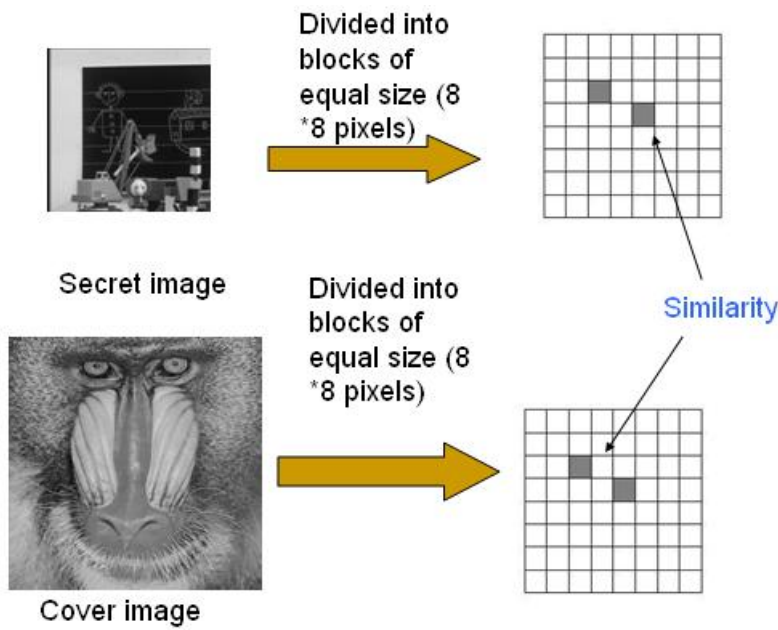


Figure 2: Image Block Partitioning

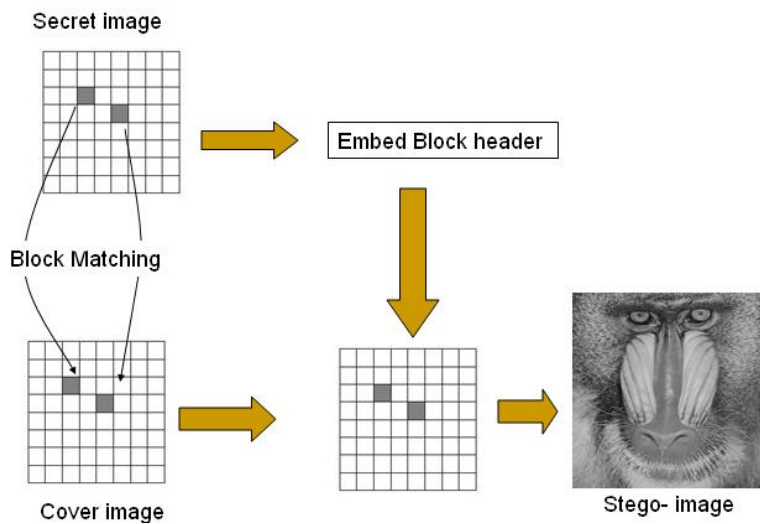


Figure 3: Block Matching and Block Header Embedding

4.2.1 Embedding Process^[1]:

Each cover-image block H_i can only be a reference block of one secret-image block S_j .

LMC 01

Step 1: For each secret-image block S_j , a cover-image block H_i is selected which having the smallest difference-degree DD_{H_i,S_j} amongst all the cover-image blocks, as a reference block of that secret-image block S_j .

Step 2: The error-matrix EM_{H_i,S_j} between the selected cover-image block H_i and the secret-image block S_j was computed.

Step 3: Then the normalized-error-matrix NEM_{H_i,S_j} between the selected cover-image block H_i and the secret-image block S_j is computed.

Step 4: Quantized-error-matrix QEM_{H_i,S_j} between the selected cover-image block H_i and the secret-image block S_j was computed

Step 5: The cover-image was then modified.

Step 6: The block header information was embedded into the unreferenced cover-image block's LSB

The additional information including the reference-block-index of S_j , the quantized-error-matrix QEM_{H_i,S_j} and the minimum element in the error-matrix EM_{H_i,S_j} , should also be embedded into the cover-image H as a form of block header by using conventional LSB substitution methods.

Reference-Block-Index	Quantized-Error-Matrix	Minimum Element in the Error-Matrix min(EM)
-----------------------	------------------------	---

Fig.4 The format of the block header

4.2.2 Extracted Process ^[1]:

The block header information need first be extracted so as to extract the secret image from the stego-image. Finally, the secret-image can be recovered by extracting all the secret-image blocks.

The detail information of this algorithm can refer to the paper

4.3 Algorithm of Data Hiding in Images by Adaptive LSB Substitution Based on the Pixel-Value Differencing^[2]

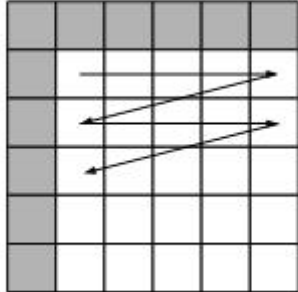
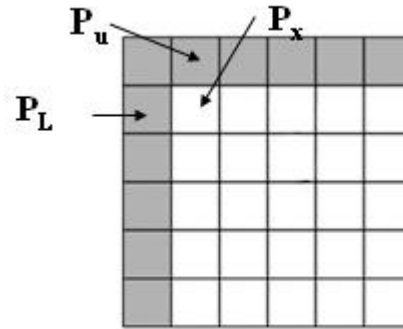


Fig. 5 The raster-scan order

Fig. 6: Positions of P_U , P_X and P_L

4.3.1 Data Embedding^[2]:

Let P_U and P_L be the upper and left neighbouring pixels of the input pixel P_X and g_u , g_l and g_x be the gray value of the upper pixel P_U , left pixel P_L and input pixel P_X respectively.

The secret data bit stream is embedded into the cover-image by the raster-scan order while the pixels located in the first row and first column are not considered to be embedded. The raster-scan order is illustrated in Figure 1.

The gray value difference d is defined as

$$d = g_u - g_l. \quad (1)$$

The embedding capacity of the input pixel P_X depends on the value of d .

Let n be the number of bits which can be embedded in the input pixel P_X and can be calculated by.

$$n = \begin{cases} 1, & \text{if } -1 \leq d \leq 1; \\ \lfloor \log_2 |d| \rfloor, & \text{if } d > 1 \text{ or } d < -1. \end{cases} \quad (2)$$

When $n > 4$, it set to be 4. Otherwise, the stego-image quality will be greatly reduced.

LMC 01

Then, an integer b which represents n bits secret data is placed in the n least significant bits of g_x and the new gray value g'_x is computed as

$$g'_x = g_x - g_x \bmod 2^n + b. \quad (3)$$

Optimal pixel adjustment process:

After replacing n -rightmost LSBs of g_x , the optimal pixel adjustment process will be applied to reduce the embedding error between g_x and g'_x . Let δ_x be the embedding error between g_x and g'_x and it is defined as

$$\delta_x = g'_x - g_x, \quad \text{where } -2^n < \delta_x < 2^n. \quad (4)$$

The value g'_x is then changed to the new gray value \hat{g}_x as follows:

$$\hat{g}_x = \begin{cases} g'_x - 2^n, & \text{if } 2^{n-1} < \delta_x < 2^n \text{ and } g'_x \geq 2^n; \\ g'_x, & \text{if } 2^{n-1} < \delta_x < 2^n \text{ and } g'_x < 2^n; \\ g'_x, & \text{if } -2^{n-1} \leq \delta_x \leq 2^{n-1}; \\ g'_x, & \text{if } -2^n < \delta_x < -2^{n-1} \text{ and } g'_x \geq 2^n; \\ g'_x + 2^n, & \text{if } -2^n < \delta_x < -2^{n-1} \text{ and } g'_x < 2^n \end{cases} \quad (19)$$

By using OPAP, the absolute embedding error between pixels in the host-image and stego-image is limited to $0 \leq |\hat{g}_x - g_x| \leq 2^{n-1}$, so that the quality of the stego-image is enhanced.

4.3.2 Data Extraction [2]:

The secret data is extracted from the stego-image by the raster-scan order while the pixels located in the first row and first column are not considered. Let P_U'' and P_L'' be the upper and left neighbouring pixels of the input pixel P_X'' and g_u'' , g_l'' and g_x'' be the gray value of the upper pixel P_U'' , left pixel P_L'' and input pixel P_X'' respectively. The gray value difference d'' is defined as

LMC 01

$$d'' = g_u'' - g_l'' \tag{5}$$

Let n'' be the number of bits which can be extracted from the input pixel P_X'' . The value

n'' is calculated by

$$n'' = \begin{cases} 1, & \text{if } -1 \leq d'' \leq 1; \\ \log_2 |d''|, & \text{if } d'' > 1 \text{ or } d'' < -1. \end{cases} \tag{6}$$

The value n'' is bounded to 4 if $n'' > 4$ and the value b is calculated by

$$b = g_x'' \bmod 2^{n''} \tag{7}$$

Finally, n'' bits secret data is then obtained by converting the value b to a binary string.

Example:

10	11	10	16	24	45	51	64
9	12	26	19	26	55	60	55
14	18	15	24	40	50	80	56
16	18	22	25	51	50	67	78
11	23	37	51	68	26	56	57
12	49	55	60	42	58	75	109
26	90	78	87	46	49	125	103
72	92	95	98	99	100	104	94

Fig.7: Part of the secret image

		P_U''	P_X''	P_R''				
P_L''	16	20	10	16	24	40	51	64
	12	12	24	19	26	58	60	55
	14	13	16	24	40	16	69	56
	14	17	22	29	51	78	68	78
	18	22	37	56	68	75	78	57
	24	35	55	64	22	78	75	109
	49	64	78	87	103	121	120	101
	72	92	95	98	99	100	103	99

Fig.8: Part of the cover image

LMC 01

Embedding Process:

1) The secret image's pixels will form a secret bit stream.

Take the first line of secret image as an example, the bit stream will be

00001010	00001011	00001010	00010000	00011000	00101101	00110011	01000000
----------	----------	----------	----------	----------	----------	----------	----------

The gray difference between P_U and P_L of cover-image is $20-12=8$.

The embedding capacity of P_X of cover-image is 4 bits according to Eq.16.

The gray value of the first pixel of the cover image is $10 = (00001010)_2$.

After replacing 4-rightmost LSBs of g_x , g_x will change from 12 $(00001100)_2$ to 0

$(00000000)_2$ as 4 LSBs are used for storing the secret data.

The embedding error will be $0-12 = -12$ according to Eq.18.

$$\because -16 < \delta_x < -8 \text{ and } \delta_x = 4 \text{ and } g'_x < 256 - 16$$

$$\therefore \hat{g}_x = 0 + 2^4 = 16$$

After taking the optimal pixel adjustment process, according to Eq.19, the new pixel gray value of P_x will be 16 and the new pixel gray value will then be used to calculate the embedding capacity of P_R .

Extraction Process:

In order to extract the secret bit stream of P_x from the stego-image, the gray difference between P_U and P_L of stego-image was first calculated and then conventional LSB was extracted according to that gray difference.

The gray difference between P_U and P_L of stego-image is $20-12 = 8$.

The number of secret data bits need to be extracted from P_x is 4 bits according to Eq.21.

After combining the entire secret data bit stream, the secret image can be recovered.

4.4 NTSC format

NTSC represents the National Television Standards Committee.

It is a video signal standard used by the colour television industry in the United States and Japan.

Most NTSC video frames consist of two interlaced fields. Each field is displayed as alternating horizontal lines across the screen.

The frame aspect ratio used by the NTSC standard format is 4:3.

4.5 ITU-R BT 656 Video Data Format

4.5.1 Introduction:

This Recommendation describes the method of interconnecting the digital television equipment operating on the 525-line or 625-line standards and complying with the 4:2:2 encoding parameters as defined in Recommendation ITU-R BT.601.PART 1

4.5.2 Common signal format of the interfaces

General description of the interfaces:

The interfaces provide a unidirectional connection between one source and one destination.

The data signals represent in the form of binary information coded in 8-bit or 10-bit words which are video signals, timing reference signals, and ancillary signals.

Video data:

Coding characteristics:

LMC 01

The video data is in compliance with Recommendation ITU-R BT.601, and with the field-blanking definition shown in Table 1.

Field interval definitions

Table 1

		625	525
V-digital field blanking			
Field 1	Start (V = 1)	Line 624	Line 1
	Finish (V = 0)	Line 23	Line 20
Field 2	Start (V = 1)	Line 311	Line 264
	Finish (V = 0)	Line 336	Line 283
F-digital field identification			
Field 1	F = 0	Line 1	Line 4
Field 2	F = 1	Line 313	Line 266

NOTE 1 – Signals F and V change state synchronously with the end of active video timing reference code at the beginning of the digital line.

NOTE 2 – Definition of line numbers is to be found in Recommendation ITU-R BT.470. Note that digital line number changes state prior to OH as described in Recommendation ITU-R BT.601 (Part A).

NOTE 3 – Designers should be aware that the “1” to “0” transition of the V-bit may not necessarily occur on line 20 (283) in some equipment conforming to previous versions of this Recommendation for 525-line signals.

Video data format:

The video data words are conveyed as a 27 Mword/s multiplex in the following order:

$$C_B, Y, C_R, Y, C_B, Y, C_R, \text{ etc.}$$

where the word sequence C_B, Y, C_R , refers to co-sited luminance and colour-difference samples and the following word, Y , corresponds to the next luminance sample.

Video timing reference codes (SAV, EAV)

There are two timing reference signals, one at the beginning of each video data block (start of active video, SAV) and one at the end of each video data block (end of active video, EAV) as shown in Fig. 1.

Each timing reference signal consists of a four word sequence in the following format:

FF 00 00 XY. The first three words are a fixed preamble. The fourth word contains

LMC 01

information defining field 2 identification, the state of field blanking, and the state of line blanking. The assignment of bits within the timing reference signal is shown in Table 2.

Table 2
Video Timing Reference Codes:

Data bit number	First word (FF)	Second word (00)	Third word (00)	Fourth word (XY)
9 (MSB)	1	0	0	1
8	1	0	0	F
7	1	0	0	V
6	1	0	0	H
5	1	0	0	P ₃
4	1	0	0	P ₂
3	1	0	0	P ₁
2	1	0	0	P ₀
1 (Note 2)	1	0	0	0
0	1	0	0	0

NOTE 1 – The values shown are those recommended for 10-bit interfaces.

NOTE 2 – For compatibility with existing 8-bit interfaces, the values of bits D₁ and D₀ are not defined.

F = 0 during field 1
1 during field 2

V = 0 elsewhere
1 during field blanking

H = 0 in SAV
1 in EAV

P₀, P₁, P₂, P₃: protection bits (see Table 3)

MSB: most significant bit

Table 1 defines the state of the V and F bits.

Bits P₀, P₁, P₂, P₃, are dependent on the states of the bits F, V and H as shown in Table

3. At the receiver this arrangement allows one-bit errors to be corrected and two-bit errors to be detected.

Table 3
Protection bits

Protection bits						
F	V	H	P3	P2	P1	P0
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	1	1
1	0	1	1	0	1	0
1	1	0	1	1	0	0
1	1	1	0	0	0	1

Ancillary data:

The ancillary signals should comply with Recommendation ITU-R BT.1364.

4.6 VGA Video Signal ^[8]

In this project, the watermarked frames will be displayed on a VGA monitor.

The XUP Virtex-II Pro Development System includes a video DAC and 15-pin high density D-sub connector to support XSGA output. The video DAC can operate with a pixel clock of up to 180 MHz. The 15-pin D-sub VGA connector consist of 15 pins including ground, red, green, blue, horizontal synchronization, vertical synchronization and the other pins that are no connected. The colour information of the screen will be controlled by Red, Green and Blue signals, while the horizontal and vertical synchronization will be controlled by the Horizontal, Vertical Syncs signals. These five signals are used to form images to be displayed on the VGA monitor.

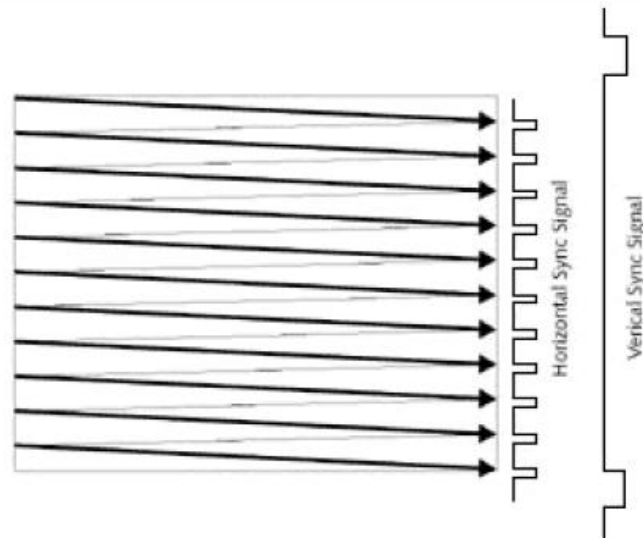


Fig.9: The diagram showing the timing of horizontal and vertical synchronization signal

4.7 Xilinx XUP Virtex-II Pro Development Board ^[8]

The XUP Virtex-II Pro Development System provides an advanced hardware platform that consists of a high performance Virtex-II Pro Platform FPGA equipped with a comprehensive collection of peripheral components that can be used to build up a complex system and to demonstrate the capability of the Virtex-II Pro Platform FPGA. As the project needs to process the video signal and this board contains the high-speed expansion connector which can be connected to the video daughter board using for the interface between the Development Board and the VCD player, this development board is suitable for using in this project. This board also contains the XSGA port which can output the video signal through the VGA cable to the monitor.

LMC 01

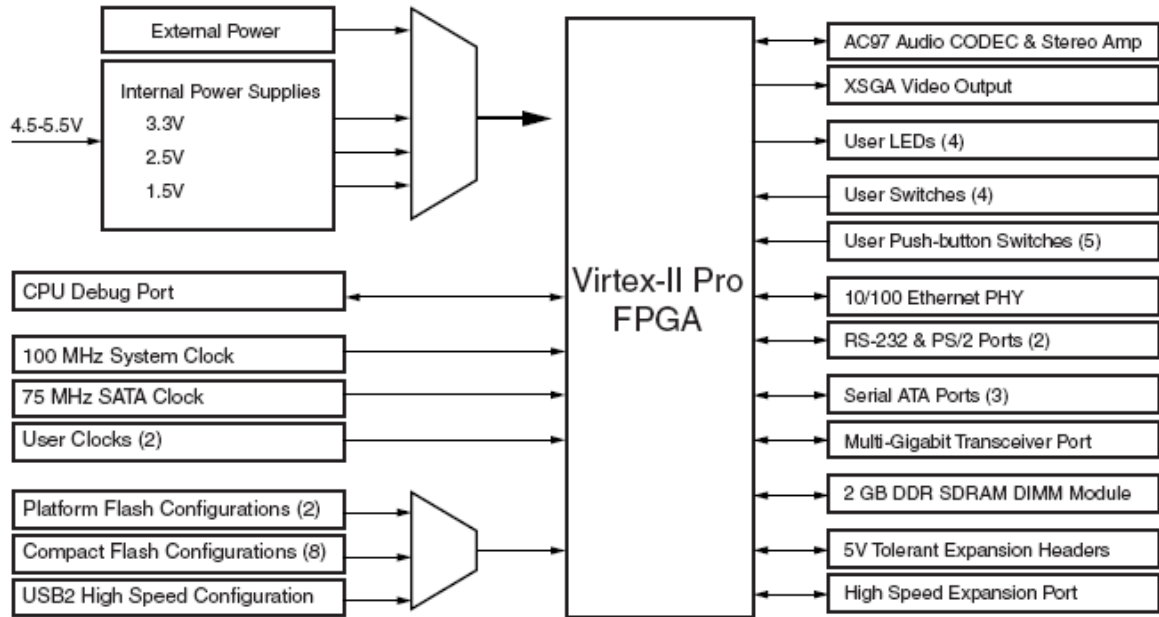


Fig. 10: The board diagram of the XUP Virtex-II Pro Development System

5. Software Evaluation

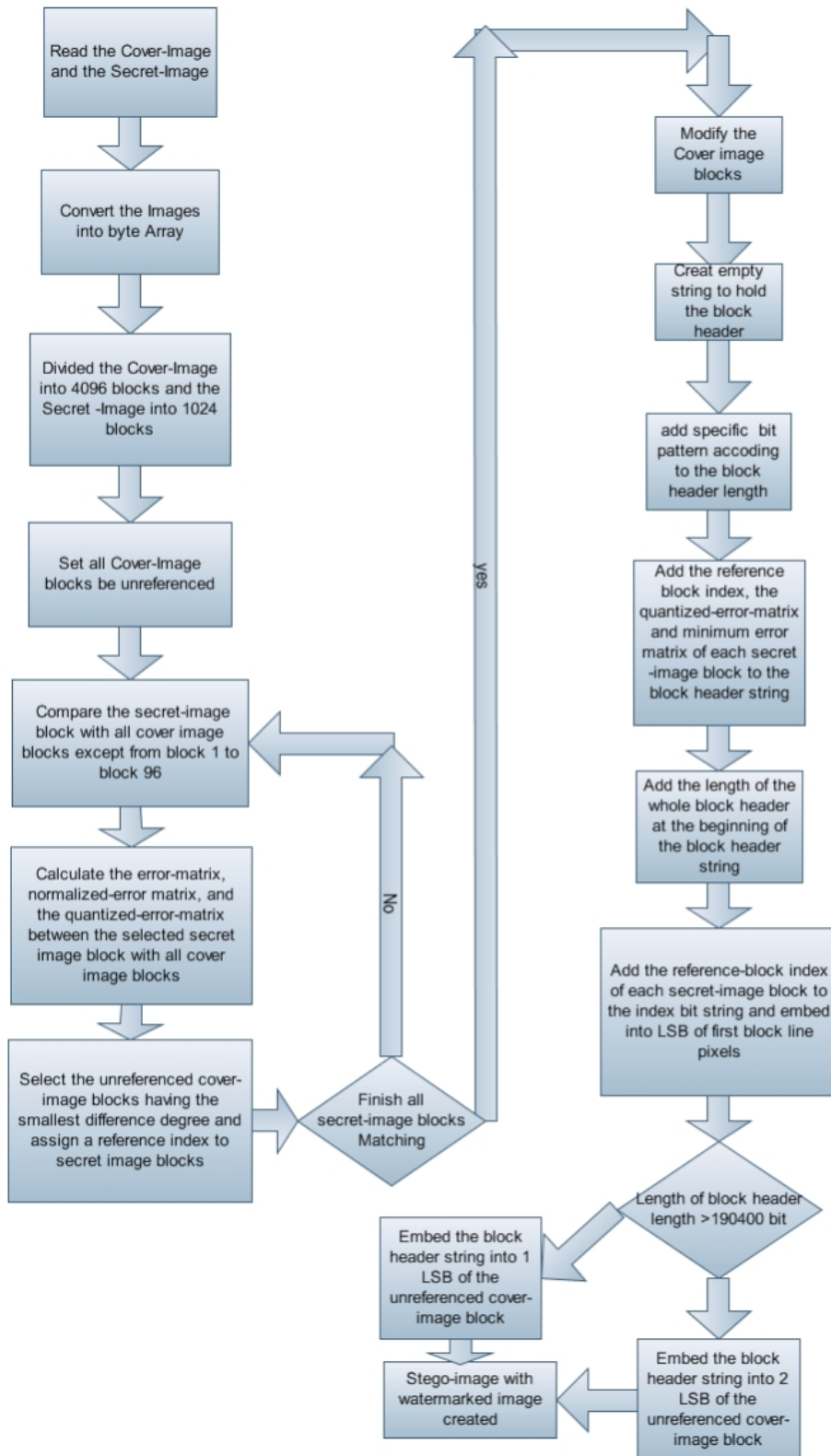


Fig.11 Flow chart of implementing the embedding algorithm of a novel image-hiding scheme based on the block difference

LMC 01

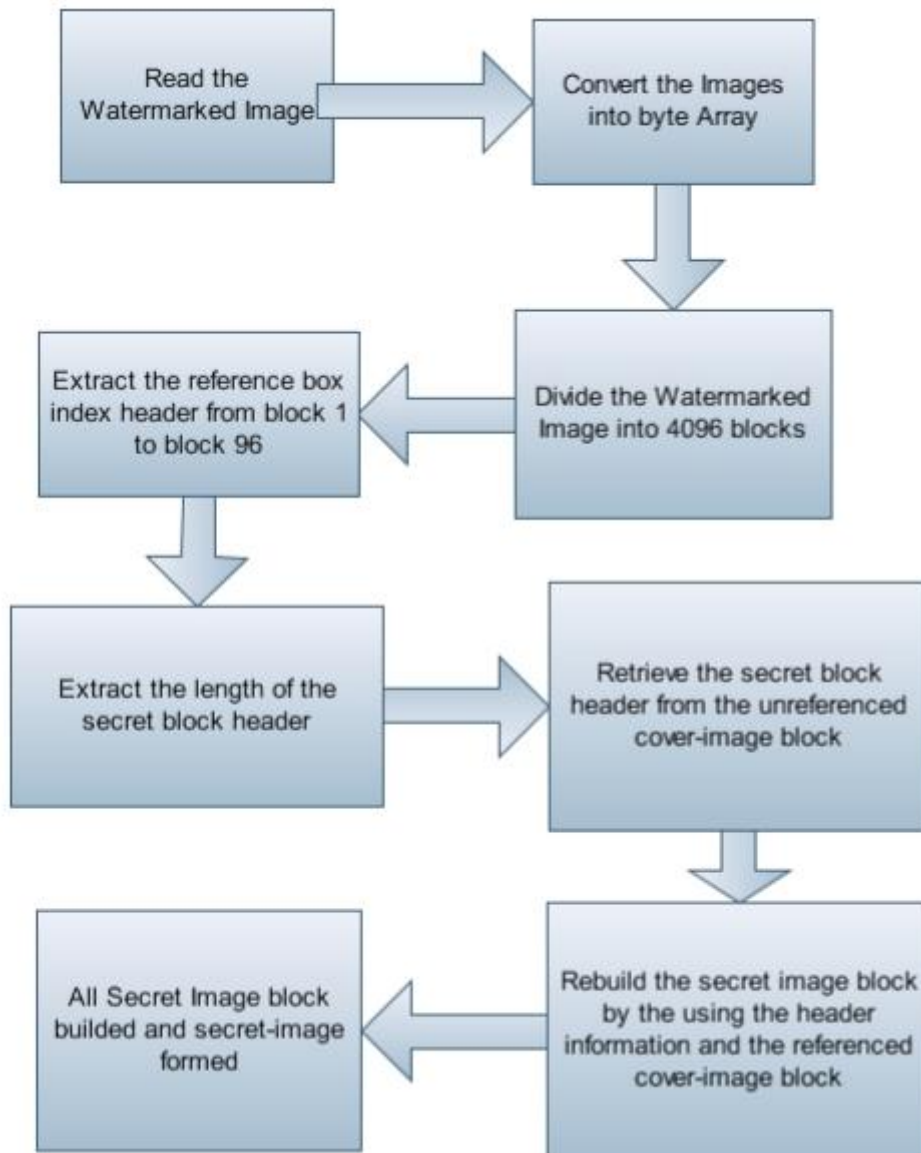


Fig.12 Flow chart of implementing the extraction algorithm of a novel image-hiding scheme based on the block difference

C# was used for implementing the algorithm's evaluation in this project.

As C#.net offers extensive support for Windows application development, it was quite simple for me to build up the user interface in the window form of the program.

There are four watermarking algorithms selected for implementing in hardware:

- 1) A novel image hiding scheme based on block difference
- 2) Data Hiding in Images by Adaptive LSB Substitution Based on the Pixel-Value

Differencing

LMC 01

3) Watermarking based on predictive neighbor pixels

4) 2 LSB substitution watermarking

Since first two algorithms are quite complex in structure, they need to be verified by in software. Raw image file format will be used for cover-image and the secret image.

5.1 User Interface

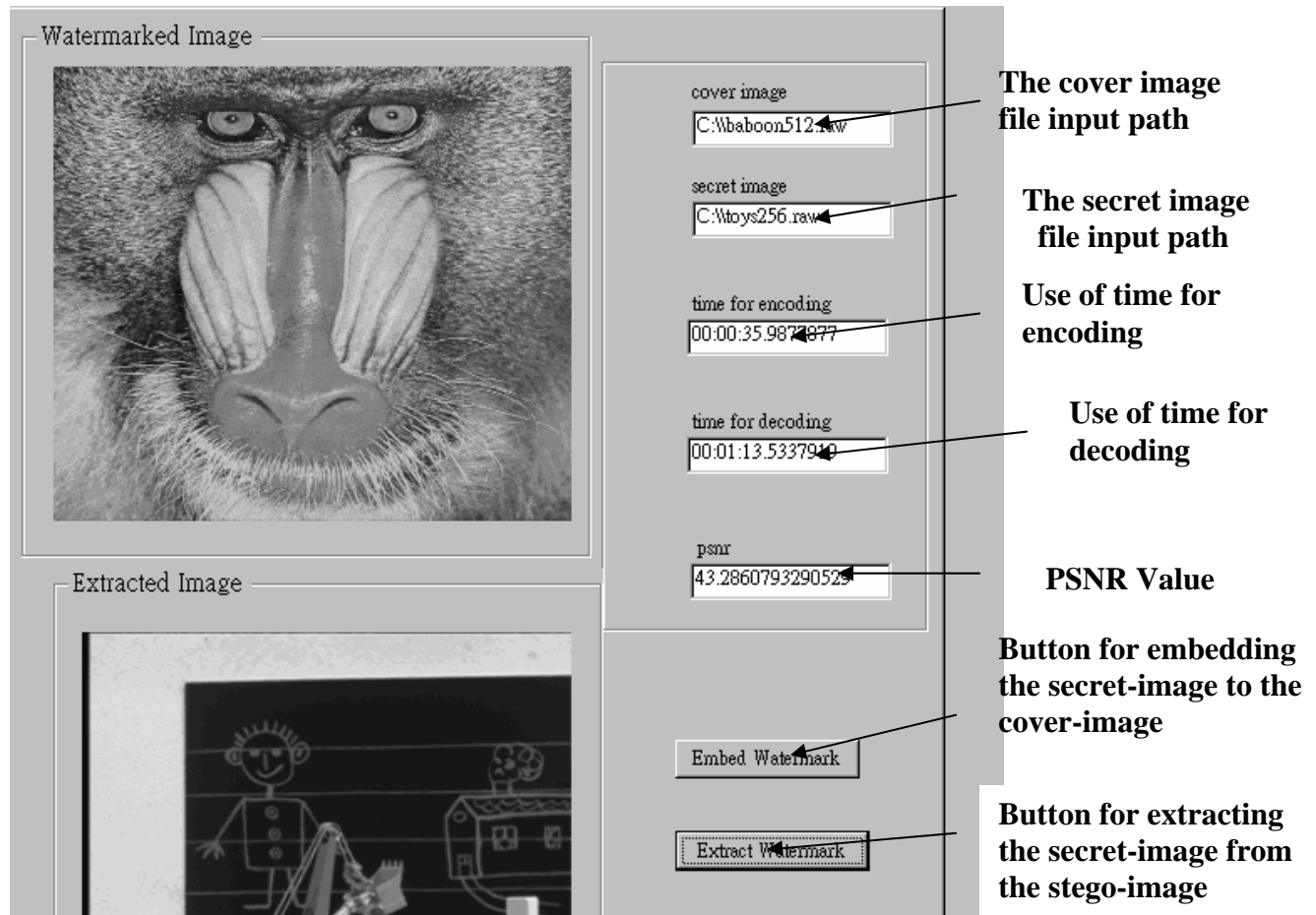


Fig.13: The user interface

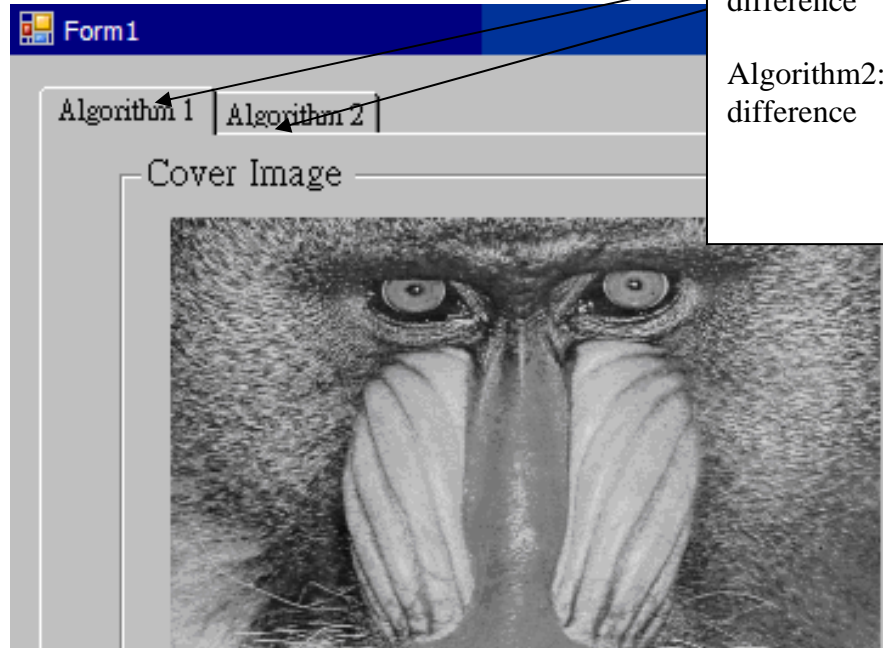


Fig.14: The user interface

5.2 Evaluation of algorithm (A novel image hiding scheme based on block difference):

In the evaluation test, both of the host-image and secret-image are 8 bit-grayscale image.

The host image size is 512×512 and the secret image size is 256×256.

The peak signal-to-noise ratio (PSNR) is employed to evaluate the stego-image quality.

For an n-bit image of size M × M pixels, the PSNR value is defined as follows:

$$PSNR = 10 \times \log_{10} \frac{(2^n - 1)^2}{1/M \times M \sum_{i=1}^M \sum_{j=1}^M (p_{ij} - \overline{p_{ij}})^2}$$

Test result by using different images:

Table 4

Host image	Secret image	Timing for encoding	Timing for decoding	PSNR Value
Baboon512	Toys256	35.99 s	1:13 s	43.286 dB
Airplane512	Toys256	27.6s	38.4s	45.29dB
Pepper512	Jet256	28.4s	42.3s	44.568 dB

LMC 01

Lena512	Bridge256	41.94s	1:33s	43.005dB
Pepper512	Lena256	31.5s	52.4s	44.321 dB

The result shows that this watermarking algorithm used a comparatively long time to run, but the quality of the stego-images obtained are quite good. And the mean PSNR value is 44.094 dB, which means the stego-images have a small distortion.

5.3 Evaluation of Algorithm (Data Hiding in Images by Adaptive LSB Substitution Based on the Pixel-Value Differencing):

In this algorithm evaluation test, both of the host-image and secret-image are 8 bit-grayscale image. The host image size is 512×512 and the secret image size is 128×128 .

Test result by using different images:

Table 5

Host image	Secret image	Timing for encoding	Timing for decoding	PSNR Value
Toys512	Jet128	3.36s	13.59s	44.219dB
Pepper512	Lena128	3.37s	11.82s	41.606dB
Lena512	Toys128	3.36s	10.76s	41.936dB
Bridge512	Baboon128	3.39s	7.05s	38.754dB
Airplane512	Lena128	3.37	16.62s	44.516dB

Since the embedding capacity of this algorithm is not good as the algorithm of a novel image hiding scheme based on block difference, the image size of the secret image need to be much smaller. Otherwise, it cannot extract the secret image correctly.

The result shows that this watermarking algorithm used a short time to run due to the smaller secret-image size and the quality of the stego-images obtained are quite good.

And the mean PSNR values are about 42.2062dB, which is lower than the first algorithm.

LMC 01

In order to evaluate the feasibility of the algorithm, there are 2 requirements need to be considered:

1. Timing Requirements
2. Complexity in Implementation

Timing Requirements means the program running time is need to be as short as possible and the complexity of the program cannot be hard to implement in hardware.

Evaluation Results of Various Watermark Techniques:

Table 6

Algorithm(s)	Timing Requirements	Complexity in Implementation
A novel image hiding scheme based on block difference	Long	Computation frame by frame and need a large frame buffer
Data Hiding in Images by Adaptive LSB Substitution Based on the Pixel-Value Differencing	Medium	Computation line by line
LSB substitution based on predictive approach	Short	Computation using three pixels only
2 LSB substitution watermarking	Short	No memory requirements

Comparison of Difference Techniques for Real-time Hardware Implementation

Table 7

Algorithm(s)	Advantage(s)	Disadvantage(s)	Hardware Implementation
A novel image hiding scheme based on block difference	A robust watermark	Long processing time	unsuitable
Data Hiding in Images by Adaptive LSB substitution Based on the Pixel-Value Differencing	Short computation time	Require several line buffers and small embedding capacity	suitable

LMC 01

LSB substitution based on predictive approach	Short processing time	Not stable	suitable
2 LSB substitution watermarking	Short processing time	Not robust enough	suitable

5.4 Selection of different watermarking techniques:

A novel image hiding scheme based on block difference:

Since the algorithm of a novel image hiding scheme based on block difference need to compute a lot of matching processes for each secret-image block with the cover-image block and compute the block header, the processing time of watermarking will be longer. Moreover, this algorithm requires a great deal of memory to store the whole secret-image pixel values and the whole cover-image values for matching; it is not desirable to implement this algorithm in hardware.

Data Hiding in Images by Adaptive LSB substitution Based on the Pixel-

Value Differencing:

The memory requirement and the timing requirement of this algorithm are not strict. It only requires several line buffers to store two line pixels value for watermarking. Therefore, it can be developed for hardware to implement.

LSB substitution based on predictive approach:

The memory requirement and the timing requirement of this algorithm are also not strict. It requires three pixels buffers for watermarking process. This algorithm is suitable to implement in the hardware.

2 LSB substitution watermarking:

It is the simplest algorithm amongst the selected algorithms. It replaces the 2 LSB of the pixel value with the secret data. This algorithm is not robust enough, but it is easier to implement in hardware.

6. Hardware Implementation:

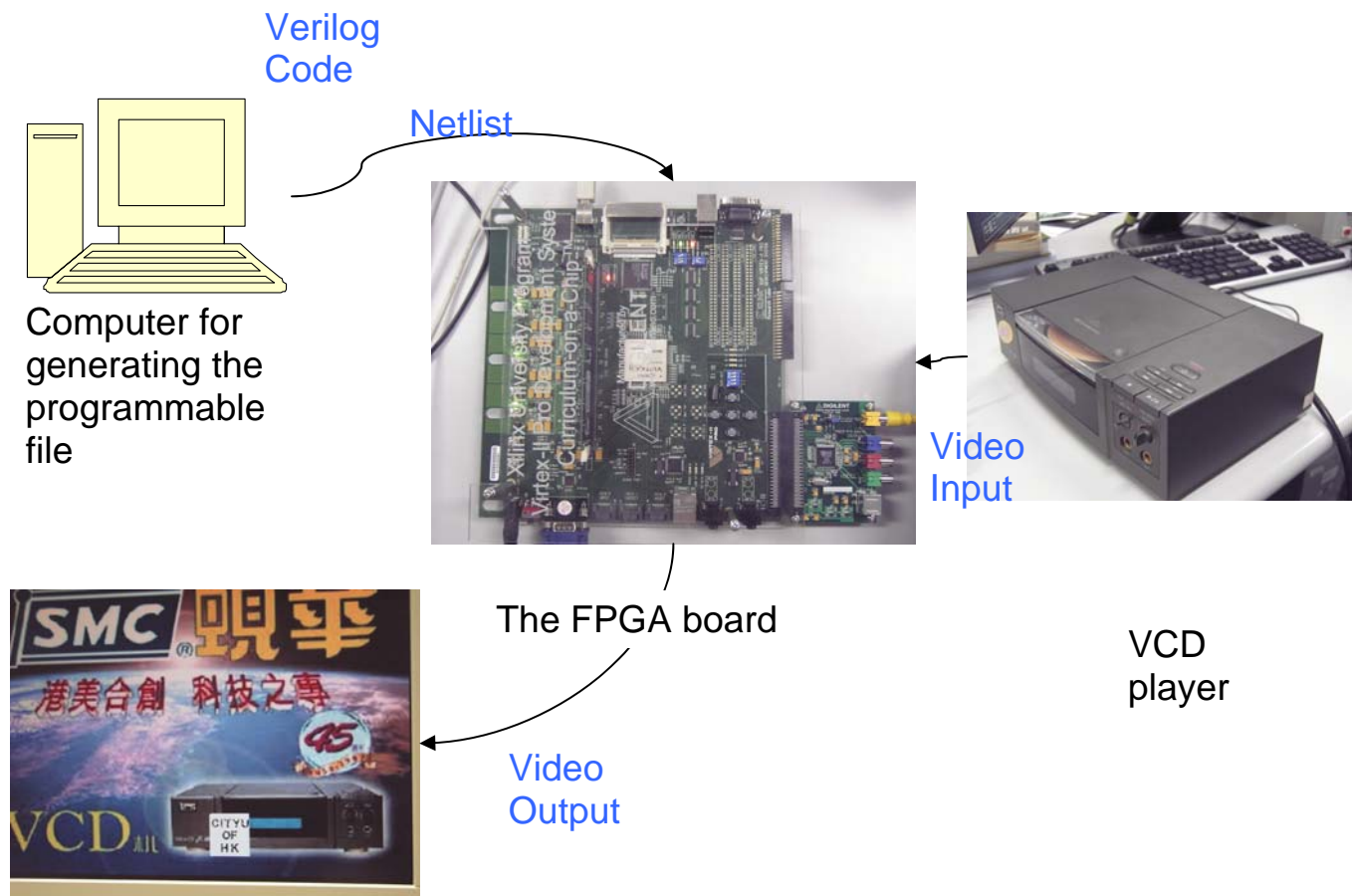


Fig 15: Hardware Implementation

LMC 01

6.1 Hardware Equipment:

The required hardware included:

- 1) Xilinx XC2VP30 development board
- 2) The Video Decoder 1 board (VDEC1) with the ADV7183B Video Decoder chip
- 3) VCD player with NTSC TV standard
- 4) Video cable
- 5) Jtag cable
- 6) Monitor
- 7) Power cable of the board

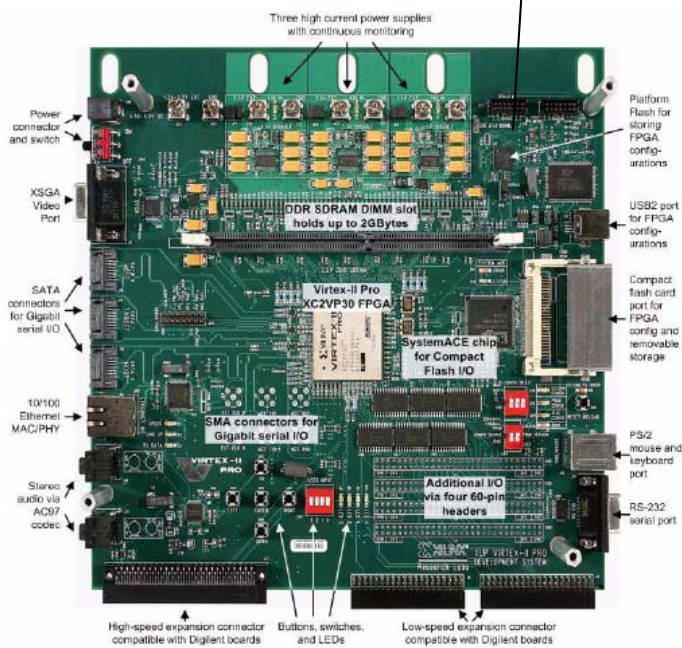


Fig17: Xilinx XC2VP30 development board

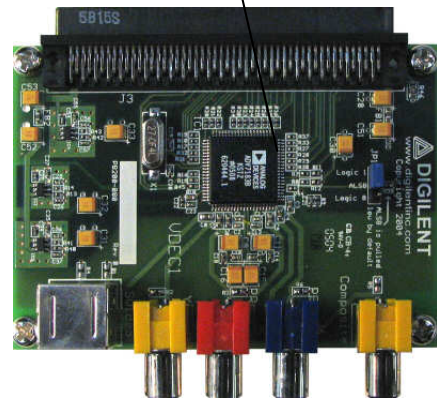
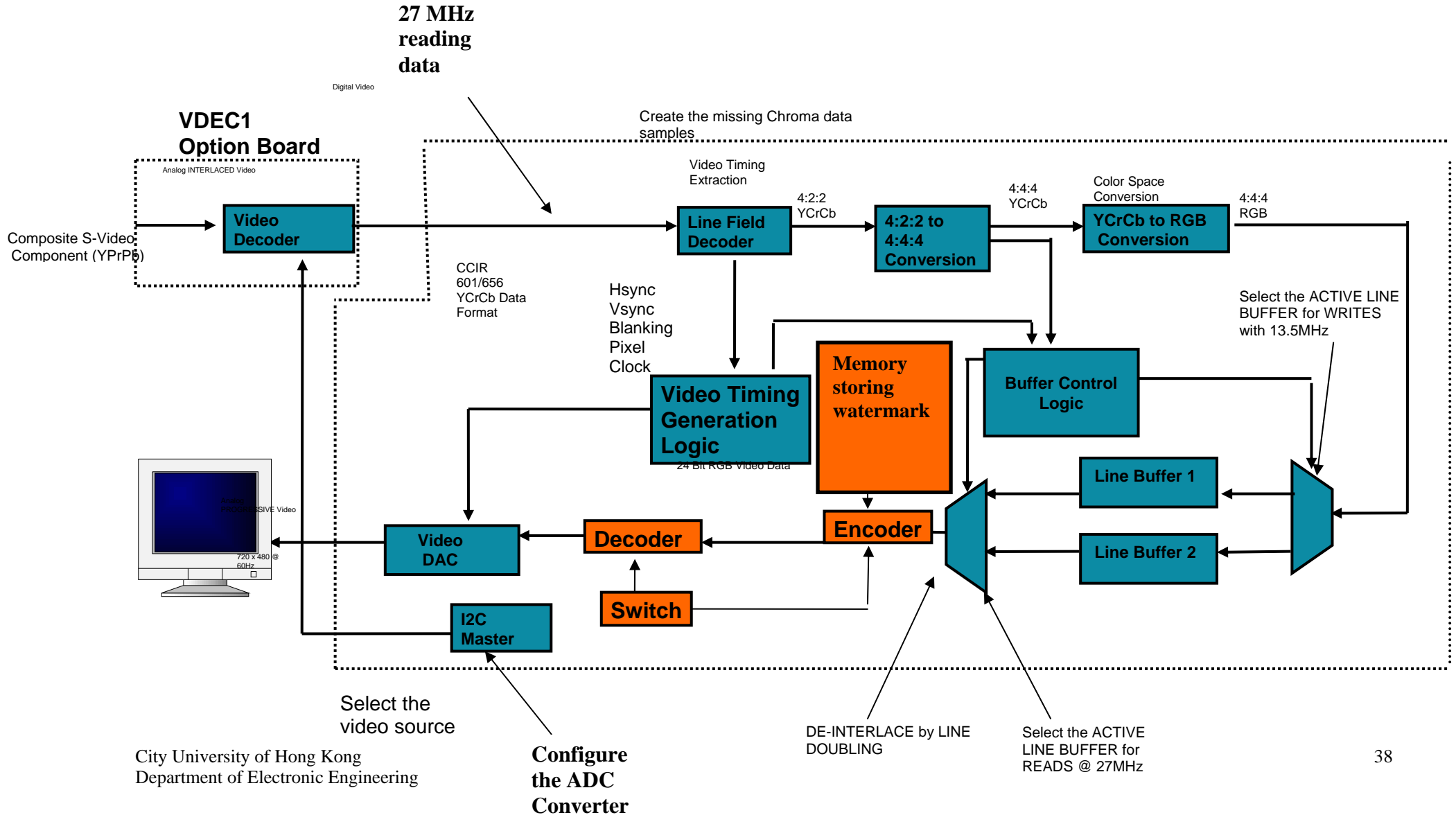


Fig.16: Video Decoder 1 board

6.2 Block diagram of Watermarking Hardware

Fig.18 Block diagram of Watermarking Hardware



6.2.1 Flow in the block diagram:

The video signal from the VCD player first passes into the VDEC board (video decoder board) and convert the analog signal to digital signal. This process was configured by the I2c master.

The digital signal generated from the video decoder board is in CCIR Ycrb format and the pixels signal are sampling at 27 MHz. When the Line field decoder receives the data, it extracts the specific timing information from the video stream and passes the video stream to the 4:2:2 to 4:4:4 converter and timing information to the video timing generation logic. The video data stream will change the video format from 4:2:2 Ycrb to 4:4:4 Ycrb via the 4:2:2 to 4:4:4 converter. The converter will buffer the data one time to retrieve enough information to perform the conversion. The clock speed of the video data stream will then be 13MHz.

After that, the 4:4:4 Ycrb format converts to the RGB format. And the RGB video data steam writes into the Line buffer 1 and Line buffer 2 by turn.

As the NTSC format is interlaced scanning, reading the original line by two times can compensate the missing line information. Therefore, write clock (13.5MHz) of both line buffers is half of the read clock (27 MHz).

The function of buffer control logic is to control line buffer reading and writing.

When RGB video stream enters into the encoder module, only red colour data bits were chosen for watermarking. Modules with orange colour are involved in watermarking process. According to different algorithms used, internal structure of the encoder will be changed. The encoder is connected to the memory storing the watermark.

When the encoder implements the watermarking, the encoder will provide the address of required data to the memory. After that, the memory will respond it by sending the

LMC 01

required data and the encoder will modify specific red colour data bits. The entire watermarking process will be controlled by a switch.

In order to show the success of watermark extraction, decoder first decodes the video signal and stores the extracted secret data to its memory between line 1 to line 399 in the screen. After line 399, the decoder extracts the stored data and replaces the RGB value of the video signal to be the watermark data in specific area. Without enable signal, decoder will not performed any data extraction. The watermarked signal will then pass to the video DAC module. The timing generation module generated the timing signals such as hsync, vsync, blanking, and pixel clock signal to the video DAC module.

Data signals and timing signals passes into video DAC module and convert into analog signal and then deliver to the monitor.

LMC 01

6.3 Software of Xilinx ISE 8.2i and ImPACT:

6.3.1 Xilinx ISE 8.2i:

The Xilinx ISE 8.2i is used for compiling the Hardware Description Language (verilog or VHDL program) to generate the programmable file (bit file format).

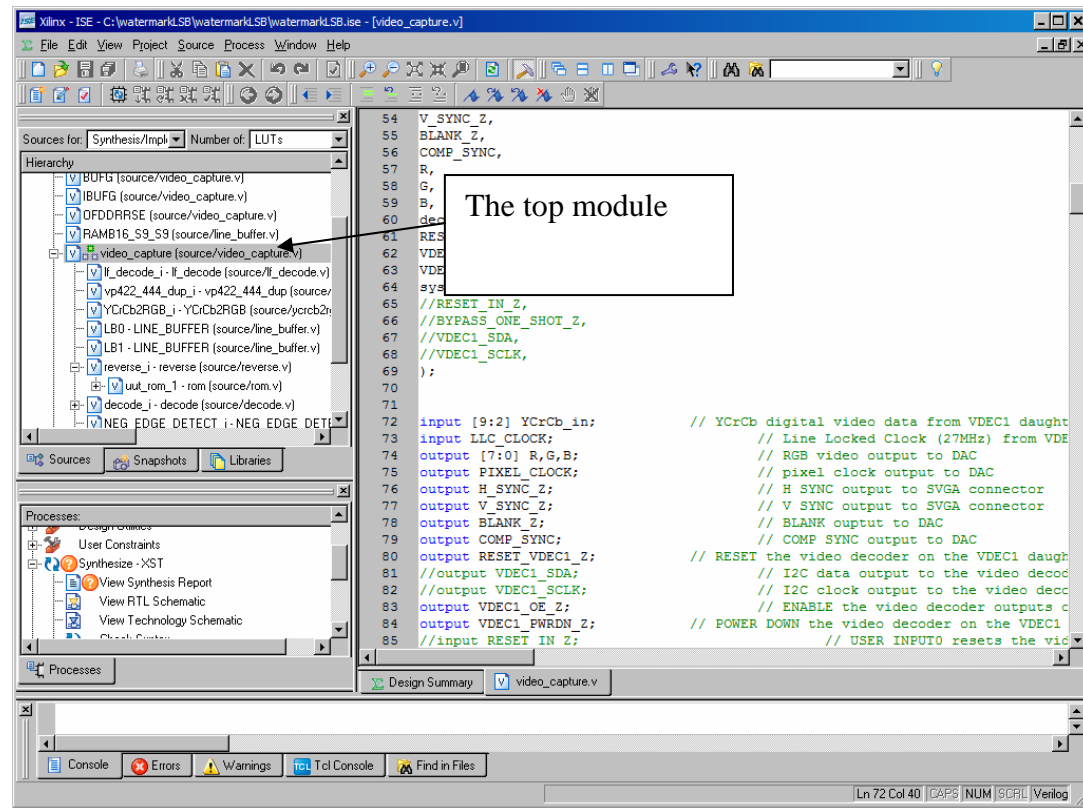


Fig.19: Xilinx ISE 8.2i interface

LMC 01

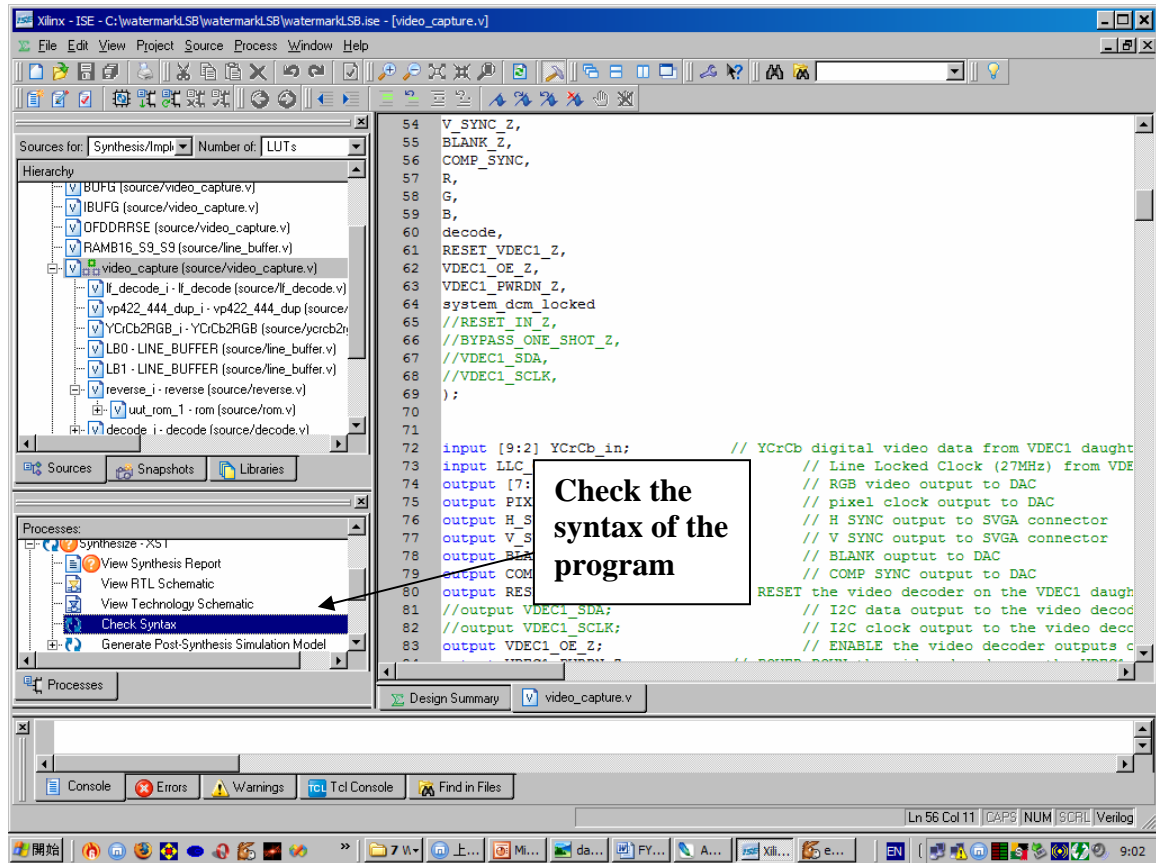


Fig.20: Xilinx ISE 8.2i interface

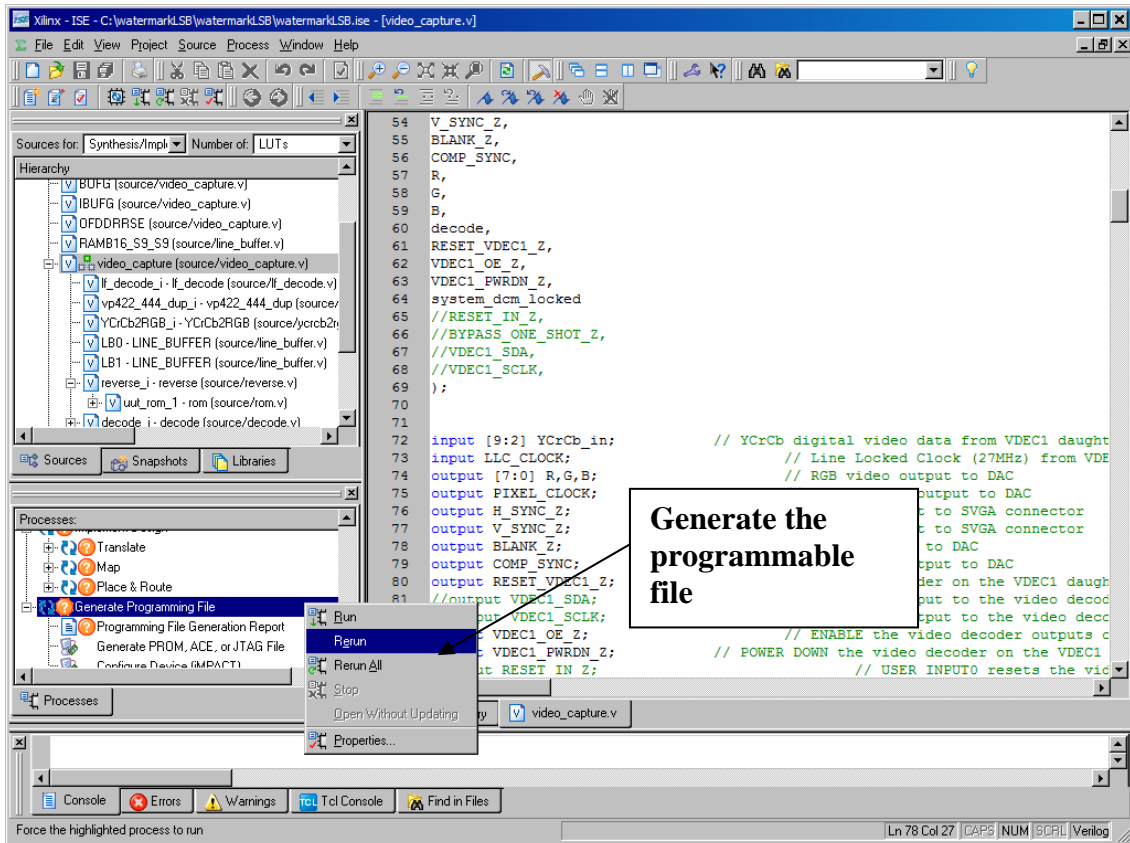


Fig.21: Generating the programmable file by Xilinx ISE 8.2i

LMC 01

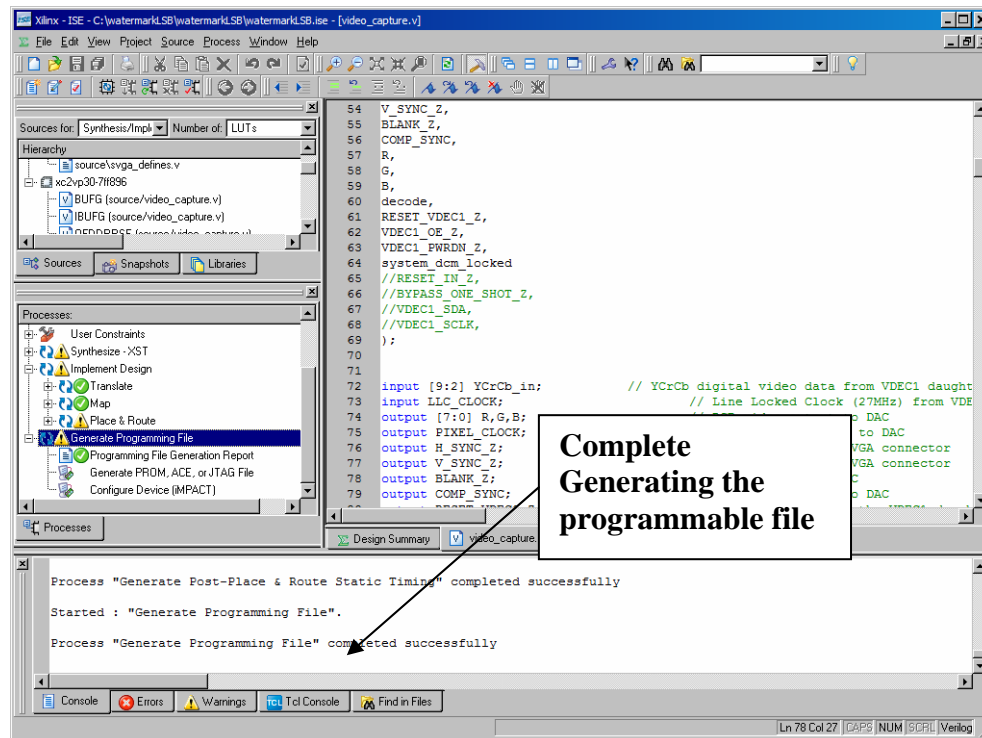


Fig.22: Finished generating the programmable file by Xilinx ISE 8.2i

6.3.2 ImPACT:

The software “imPACT” is used for downloading the generated programmable bit file to the FPGA board for configuring the FPGA

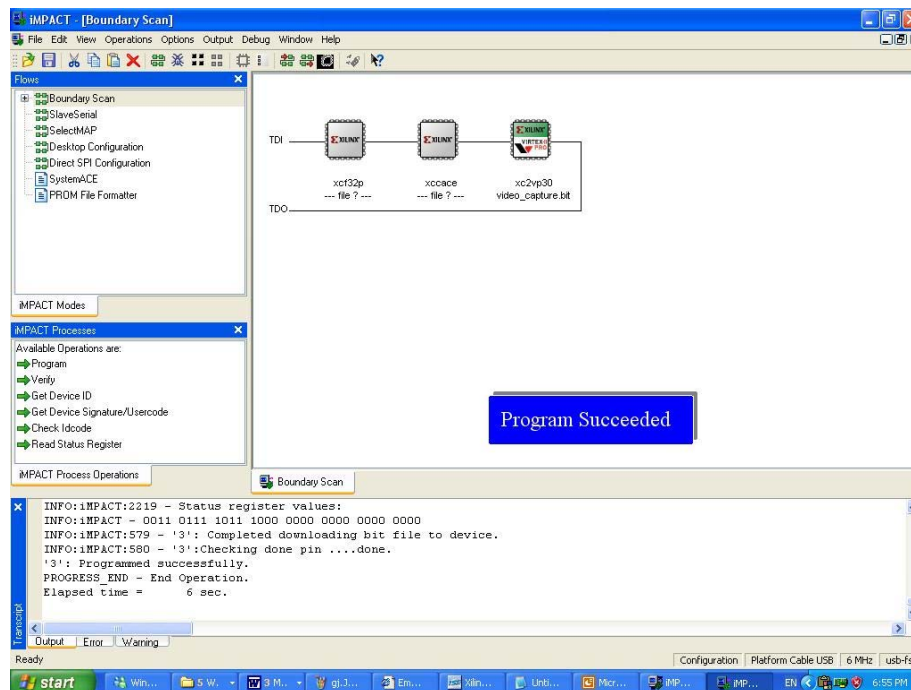


Fig.23 The video capture bit file has been successfully download to the FPGA board

LMC 01

6.4 The Hardware Flow-Diagram:

This project aims to develop a digital watermarking device for authorization and copyright identification.

In this project, the VCD player will first transmit the video signal to the video daughter board connected to the FPGA through the video cable. Meanwhile, the computer will compile the verilog code and synthesis. After that, it generates the programmable file and then downloads the netlist to configure the FGPA board and the video daughter board. Then the FPGA will execute the watermark program and deliver the watermarked signal to the computer monitor though the VGA cable.

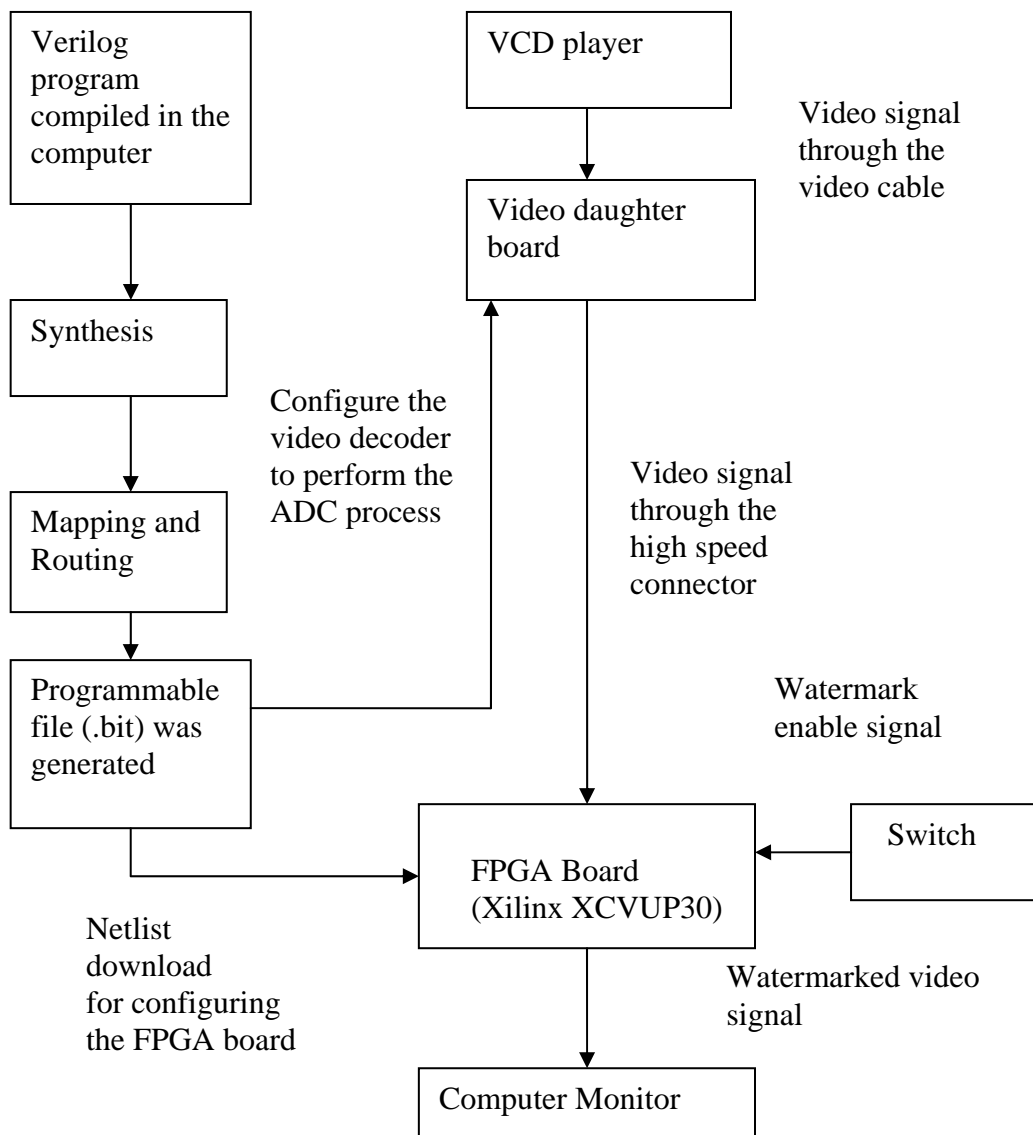


Fig.24 The flow chart of watermarked hardware

6.5 The Hierarchy of the Watermarked System

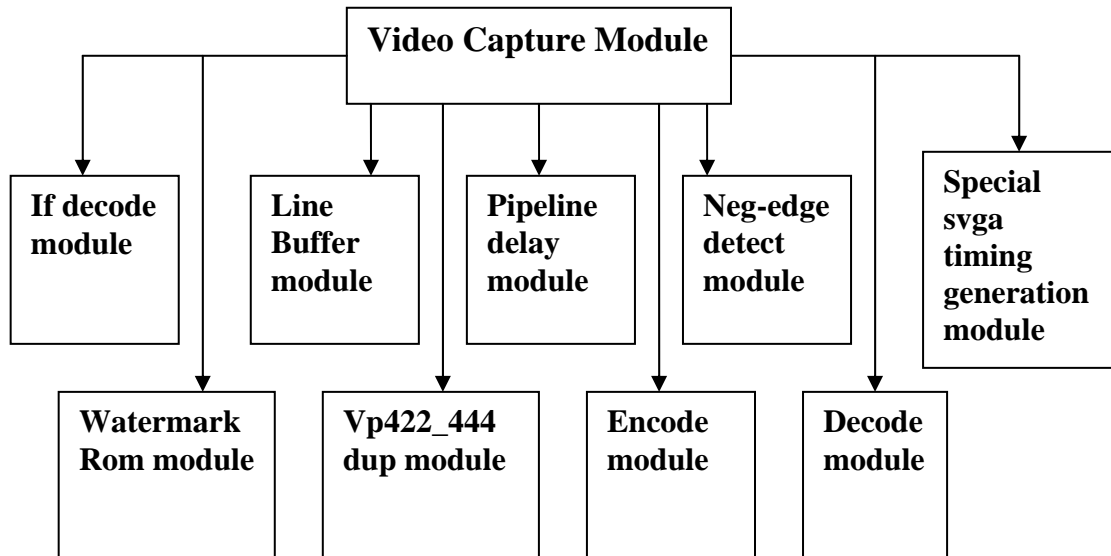


Fig.25 The hierarchy of the watermarked system

6.5.1 Video Capture Module:

Function: This module captures 656 digital video data signal and displays the video to a computer monitor. This is the top module in the hierarchy and consists of all sub-modules in the design.

Input:

Bits Length	Signal Name	Comment
10 bits	YCrCb_in	YCrCb digital video data from VDEC1 daughter board
1 bit	LLC_CLOCK	Line Locked Clock (27MHz) from VDEC1 daughter board
1 bit	system_dcm_locked	
1 bit	decode	The decode input

LMC 01

Output:

Bits Length	Signal Name	Comment
8 bits	R	Red video signal
8 bits	G	Green video signal
8 bits	B	Blue video signal
1 bit	H_SYNC_Z	H SYNC output to SVGA connector
1 bit	V_SYNC_Z	V SYNC output to SVGA connector
1 bit	COMP_SYNC	COMP SYNC output to DAC
1 bit	PIXEL_CLOCK	Pixel clock output to DAC
1 bit	BLANK_Z	BLANK output to DAC
1 bit	RESET_VDEC1_Z	RESET the video decoder on the VDEC1 daughter

6.5.2 If_decode module:

Function: This module identifies the format as NTSC or as PAL and to retrieve specific video timing information from the video signal.

Input:

Bit Length	Signal Name	Comment
1 bit	rst	Reset and Clock signal
1 bit	clk	27MHz for SDTV
10 bits	YCrCb_in,	Data from the input video stream

Output:

Bit Length	Signal Name	Comment
10 bits	YCrCb_out	Data delayed by pipe length
1 bit	NTSC_out	High = NTSC format detected
1 bit	Fo	High = field one (even)
1 bit	Vo	High = vertical blank

6.5.3 Line Buffer Module:

Function: This module defines the video line buffer RAM.

Input

Bit Length	Signal Name	Comment
1 bit	read_clk	Read clock
1 bit	read_enable	Active high
11 bits	read_address	Buffer address
1 bit	write_clk	Write clock
8 bits	write_red_data	Red data input

LMC 01

8 bits	write_green_data	Green data input
8 bits	write_blue_data	Blue data input
11 bits	write_enable	Write enable

Output

Bit Length	Signal Name	Comment
8 bits	read_red_data	Red data output
8 bits	read_green_data	Green data output
8 bits	read_blue_data	Blue data output

6.5.4 Neg_Edge_Detect Module:

Function: This module creates a one clock wide pulse on the negative transition of the

"data_in" signal. This is used to reset the vertical line counter in the

SPECIAL_SVGA_TIMING_GENERATION module on the transition of the "FIELD" bit

in the timing reference code.

Input

Bit Length	Signal Name	Comment
1 bit	clk	Clock
1 bit	data_in	Data input
1 bit	reset	Reset

Output

Bit Length	Signal Name	Comment
1 bit	one_shot_out	signal will be high for one clock cycle after the input transitions high to low

6.5.5 Pipe_Line Delay Module:

Function: This module aligns the sync and blank signals with the video data.

Input

Bit Length	Signal Name	Comment
1 bit	clk	Read clock
1 bit	rst	Active high
1 bit	hsync_in	Horizontal synchronization signal

LMC 01

1 bit	vsync_in	Vertical synchronization signal
1 bit	blank_in	Blanking signal
1 bit	comp_sync_in	Composite synchronization signal

Output

Bit Length	Signal Name	Comment
1 bit	hsync_out	Delayed horizontal synchronization signal
1 bit	vsync_out	Delayed vertical synchronization signal
1 bit	blank_out	Delayed blanking signal
1 bit	comp_sync_out	Delayed composite synchronization signal

6.5.6 Special SVGA Timing Generation Module:

Function: This module creates the timing and control signals for the VGA output and provides the control signals for the DAC and the VGA output connector. The screen size chosen for this project to implement is 800*600.

Input

Bit Length	Signal Name	Comment
1 bit	pixel_clock	pixel clock
1 bit	reset	reset
11 bits	hsync_in	Horizontal synchronization signal
1 bit	vsync_in	Vertical synchronization signal
8 bits	blank_in	Blanking signal
8 bits	comp_sync_in	Composite synchronization signal

Output

Bit Length	Signal Name	Comment
1 bit	h_synch_delay	Horizontal synch for VGA connector
1 bit	v_synch_delay	Vertical synch for VGA connector
1 bit	comp_synch	Composite synch for DAC
1 bit	blank	composite blanking
3 bits	char_line_count	line counter for char gen rom
14 bits	char_address	character mode address
11 bits	pixel_count	Count pixels in a line
10 bits	line_count	Count lines in a frame

6.5.7 Watermark Rom Module:

Function: This module provides the watermark rom.

Input

LMC 01

Bit Length	Signal Name	Comment
12 bits	addr	Memory address
1 bit	clk	Input clock

Output

Bit Length	Signal Name	Comment
8 bits	dout	Memory output

6.5.8 Vp422_444_dup module:

Function: This module provides the conversion of 4:2:2 to 4:4:4 by creating the missing Cr and Cb components. This module duplicates the Cr and Cb information to compensate missing Cr and Cb components.

Input

Bit Length	Signal Name	Comment
1 bit	rst	Reset and Clock input
1 bit	clk	27Mhz for SDTV
10 bits	ycrcb_in	data from the line field decoder
1 bit	ntsc_in	from Line field decoder
1 bit	fi	“FIELD” bit from Line field decoder
1 bit	vi	“VERTICAL BLANK” bit from Line field decoder
1 bit	hi	“HORIZONTAL BLANK” bit from Line field decoder

Output

Bit Length	Signal Name	Comment
8 bits	dout	Memory output
1 bit	ceo	output enable valid out put 1/2 ycrcb_in rate
1 bit	ntsc_out_o	NTSC format detected delayed to match 422-444 pipe length
1 bit	fo	high = field one (even) delayed to match 422-444 pipe length
1 bit	vo	high = vertical blank delayed to match 422-444 pipe length
1 bit	ho	low = active video delayed to match 422-444 pipe length
10 bits	y_out	4:4:4 luma data
10 bits	cr_out	4:4:4 chroma data

LMC 01

10 bits	cb_out	4:4:4 chroma data
---------	--------	-------------------

6.5.9 YCrCb2RGB module:

Function: This module converts the Y Cr Cb video data into the RGB color space.

Input

Bit Length	Signal Name	Comment
1 bit	clk	Synchronize Clock Signal
1 bit	rst	Reset Signal
10 bits	Y	Red data signal
10 bits	Cr	Green data signal
10 bits	Cb	Blue data signal

Output

Bit Length	Signal Name	Comment
8 bits	read_red_data	Red data output
8 bits	read_green_data	Green data output
8 bits	read_blue_data	Blue data output

6.5.10 Encode module:

Function: This module encode the watermark into the video signal.

Input

Bit Length	Signal Name	Comment
8 bits	R	Red data input
8 bits	G	Green data input
8 bits	B	Blue data input
11 bits	pixelcount	Pixel position
10 bits	linecount	Line position
1 bit	pixelclock	Pixel clock

Output

Bit Length	Signal Name	Comment
8 bits	Rout	Red data output
8 bits	Gout	Green data output
8 bits	Bout	Blue data output

LMC 01

6.5.11 Decode module:

Function: This module decode the watermark from the video signal and place it into the video frame.

Input

Bit Length	Signal Name	Comment
8 bits	R	Red data input
8 bits	G	Green data input
8 bits	B	Blue data input
11 bits	pixelcount	Pixel position
10 bits	linecount	Line position
1 bit	pixelclock	Pixel clock

Output

Bit Length	Signal Name	Comment
8 bits	Rout	Red data output
8 bits	Gout	Green data output
8 bits	Bout	Blue data output

6.6 Demonstration of Watermarking

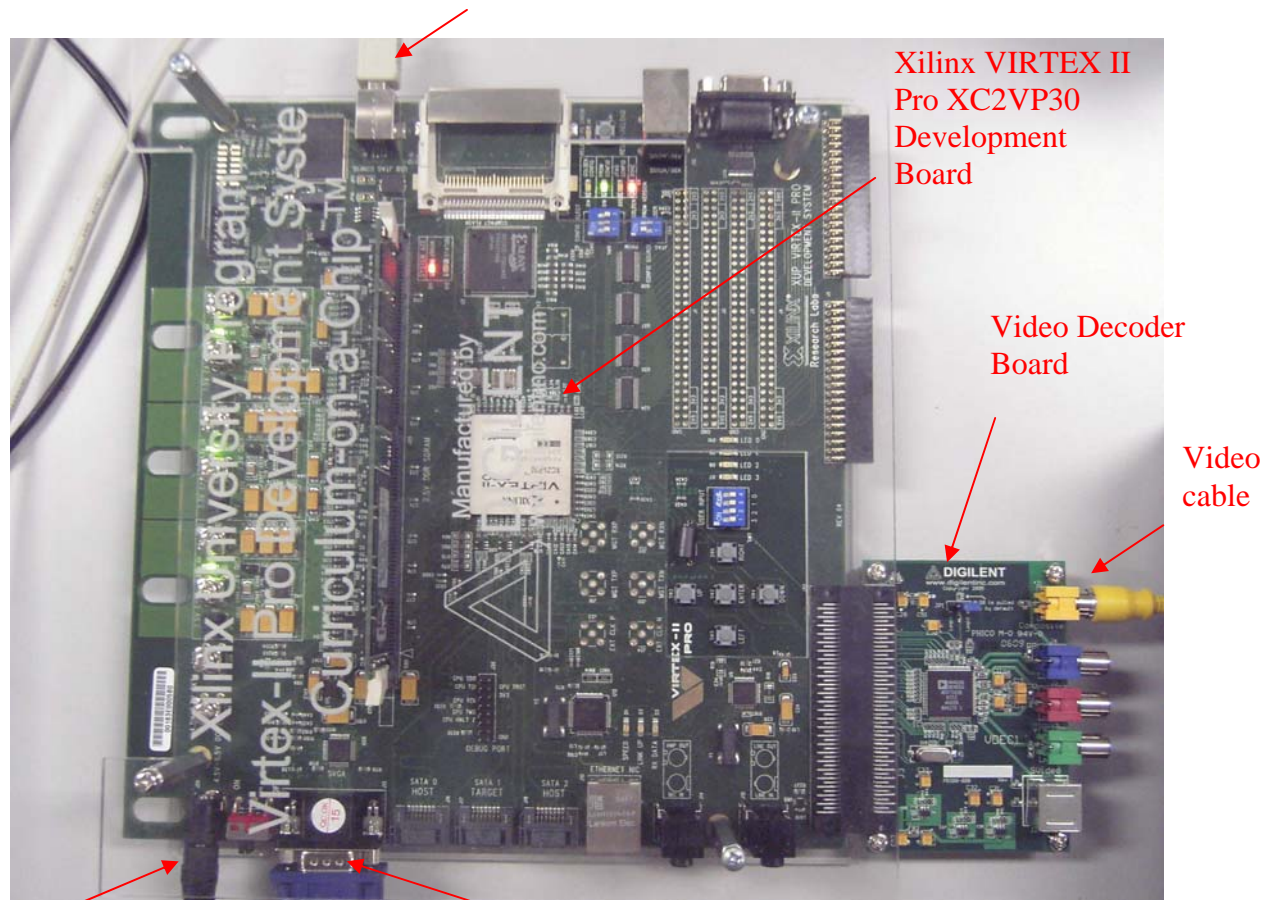


Fig.26: The Xilinx XC2VP30 Development Board and its peripheral

Power Cable

VGA cable

To configure the video decoder board, button (up) need to be pressed

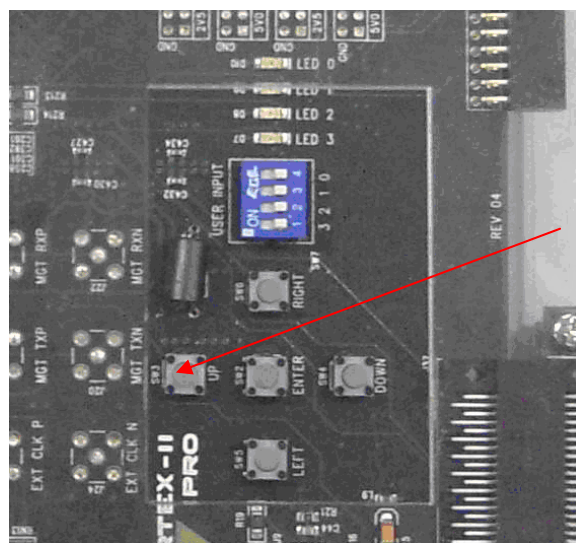


Fig.27: the position of button (up)

LMC 01

After downloading the programmable bit file (adv7183b_i2c.bit) to the FPGA and pressing Button (up), the LED 2 will light.

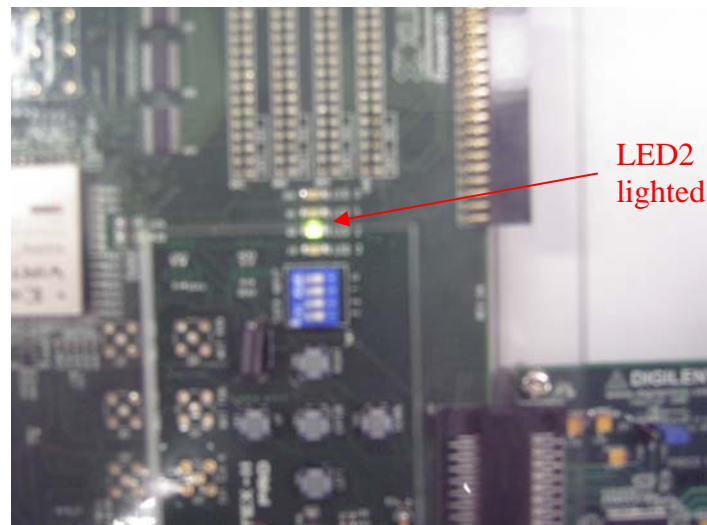


Fig.28: The lighted LED 2

The programmable file for watermarking can then be downloaded to the FPGA. The watermarked VCD video will then displayed on the monitor screen via the FPGA board.



Fig.29: The monitor screen contains invisible watermark

To verify the correctness of the watermark, switch '3' can be turned on.

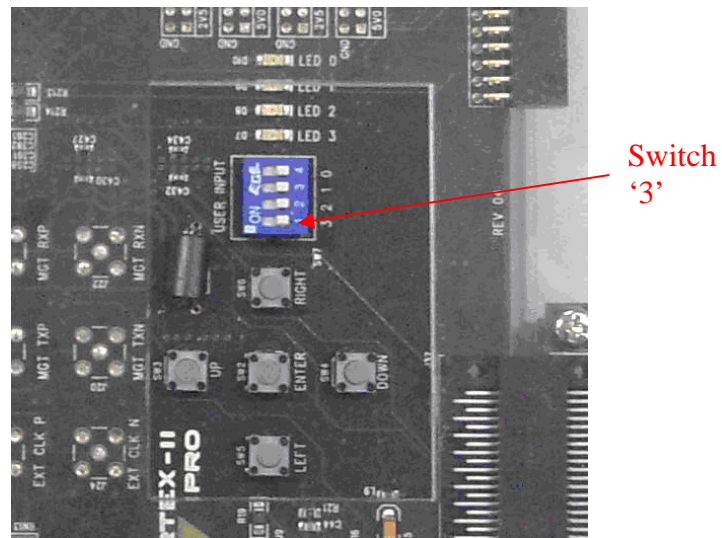


Fig.30: The position of switch '3'



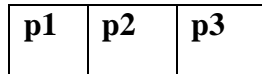
Fig.31: The watermark shown in the screen

After turning on the switch '3', a watermark logo "CITYU OF HK" can be seen on the screen. The demonstration of watermarking hardware has completed.

6.7 Hardware Implementation with three selected algorithms:

6.7.1 Algorithm of LSB substitution based on predictive approach:

This algorithm requires the additional memory for the decoder to store the watermark.



This algorithm uses three pixels at some specific positions for watermarking.



Fig.32: Red lines define the specific positions

The flow of this algorithm implementing in verilog language:

Embedding Process:

Step 1: Choose the specific positions of pixels and neighbour pixels to carry out the watermarking.

For example, Red lines are the specific pixels position of p2 and the neighbour pixels around it are p1 and p3.

Step 2: For each pixel **p2**, store its value and its neighbouring pixels values for comparison.

Step 3: Calculate the minimum value and the maximum value among p1, p2 and p3.

Step 4: The initial read address of the watermark memory is 0.

LMC 01

If memory content of the watermark equals to 8 bit “00000000”, p2 will set to the minimum value among p1, p2 and p3. Otherwise, p2 will set to the maximum value among p1, p2 and p3.

After that, the read address will be incremented by one until reaching the last address.

In the hardware, the encode module will buffer several pixels at specific positions and modify the pixel values according to the memory content of the watermark memory.

Extraction Process:

Step 1: Extract the pixels values from the specific positions in each line

Step 2: Store these three pixels values for comparison.

Step 3: Compare the central pixel with neighbor pixels. If it is greater than the neighbors, the decoder’s memory will store as 8 bit “11111111”. Otherwise, the memory will store as 8 bit “00000000”.

Step 4: Stop to extract secret data when the memory is full.

Step 5: After 400 lines in the screen, the watermark is ensured to be stored in the decoder’s memory. Therefore, it can be extracted from the decoder’s memory and displayed to the monitor by setting the RGB values equal to it.



Fig.33: The frame displayed with a watermark using the algorithm of LSB substitution based on predictive approach

6.7.2 Algorithm of 2 LSB substitution watermarking:

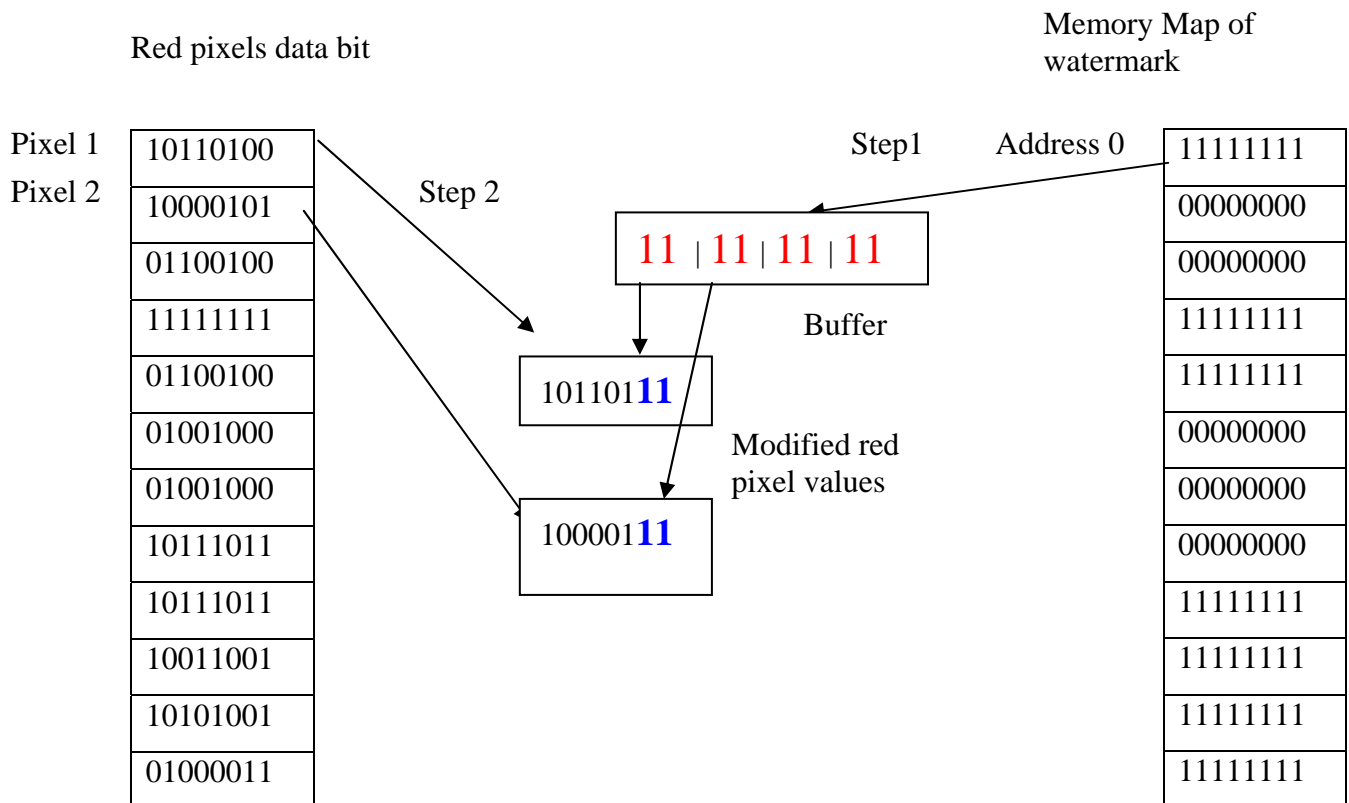


Fig.34: The flow of the algorithm implementing in verilog language

LMC 01

Embedding process:

Step 1: Divide each of 8 bit memory contents into 4 groups

Step 2: Replace the 2 least significant bits of Red data bits with each group by turn until reaching the last address of the memory

Extraction process:

Step 1: Extract the 2 least significant bits of Red data signals and store the 2 secret data bits into a temporary buffer.

Step 2: Shift the data bits in temporary buffer into a memory storing for watermark

Step 3: After 400 lines in the screen, it is ensured that the watermark has stored in the memory. Therefore, it can be extracted from the decoder's memory and displayed to the monitor by setting the all RGB values equal to the watermark value.



Fig.35: The frame displayed with a watermark using the algorithm of 2 LSB substitution watermarking.

LMC 01

The advantage of using 2 LSB substitution watermarking are high embedding capacity and low computational complexity, while the disadvantages is that this algorithm is not robust enough to protect watermark information.

6.7.3 Algorithm of data hiding in images by adaptive LSB substitution based on the pixel-value differencing:

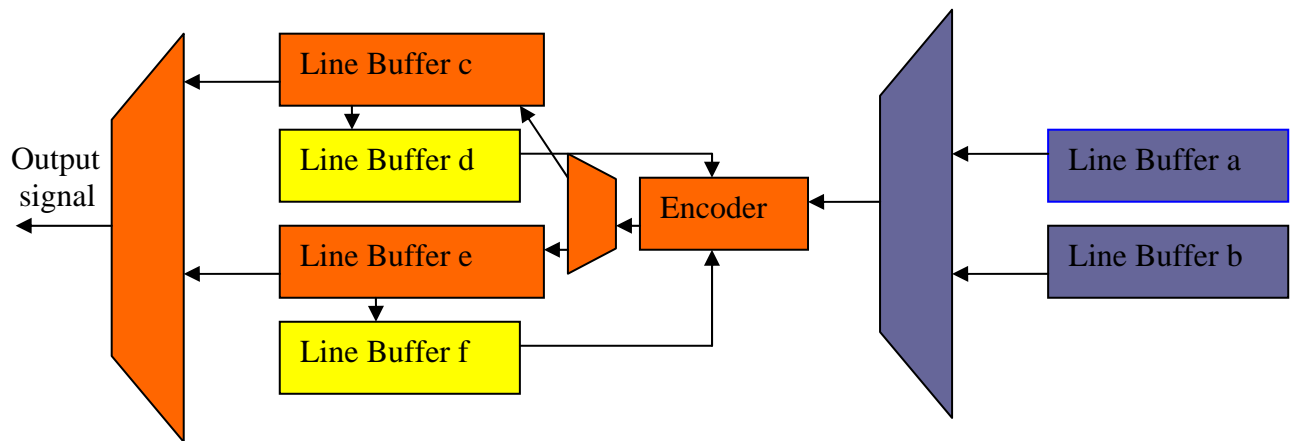


Fig.36 The design of implementing the watermark embedding algorithm

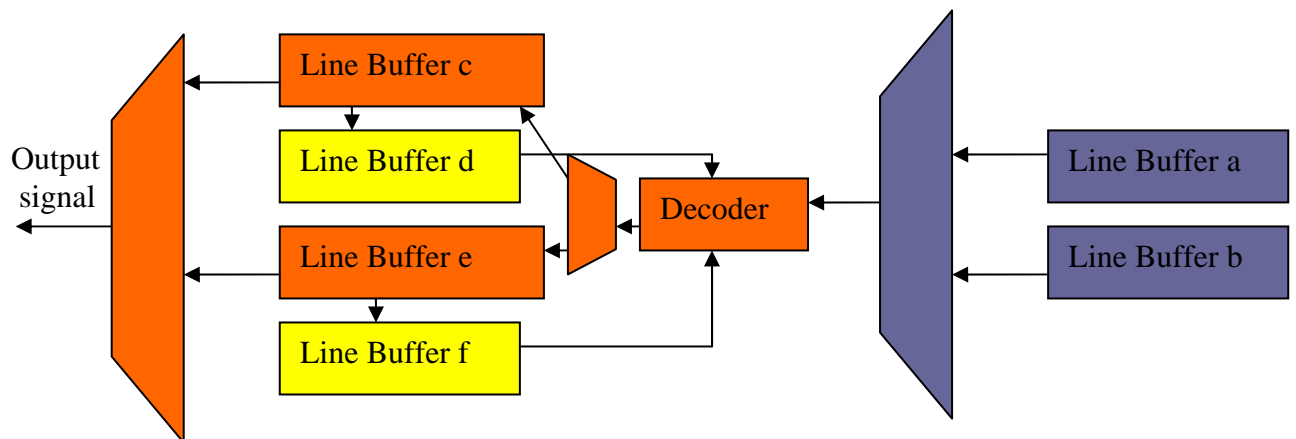


Fig.37: The design of implementing watermark extraction algorithm

In order to adopt this algorithm, the structure of the whole watermarking system part has to be modified.

LMC 01

Since the algorithm needs to store two lines for computation, two line buffers are added as line buffer c and line buffer e. As the required pixel position of encoding process and the output pixels are not the same, additional line buffers were required. One is for encoding process, and other one is for transmitting to outside. Line buffer d and line buffer f was added to duplicate line buffer c and line buffer e respectively.

When line buffer c is reading by the ADC module, pixel value stored in line buffer d and the coming video signal can be used for encoding and the watermarked data signals will pass into line buffer e and line buffer f.

The line buffer c and line buffer e will be read and written by turns, which are controlled horizontal synchronization signal.

In order to meet the time requirement of this algorithm, the watermarking process only need to process the odd pixels. The rest time can be used for computation process.

However, this algorithm has not implemented successfully yet, as the time-delay adjustment problems has not been solved.

7. Discussion:

7.1 Difficulty Encountered:

During working out this project, I have encountered different kinds of problems. Some of them seriously affect my plan.

Problem 1:

When I first learned C#, I found the difficulty to choose appropriate function to implement the algorithm. Therefore, I took much time to familiarize with different function of C#. Moreover, the compiler of Microsoft Visual Studio 2005 is also slow to execute the watermarking process. Therefore, the debugging process was like a time-consuming process.

Problem 2:

At the beginning of developing the watermark hardware, the Altera Cyclone II board was used. Before writing verilog HDL to implement the algorithm, I found sample programs about video capture written in VHDL language, which is similar to my project.

As I have not learned VHDL before, it was a very strange thing for me to familiar the difference between VHDL and Verilog HDL. Moreover, the flow of that program was also unclear to me, because some of program codes were generated by some mega function and are Altera HDL and the length of that sample codes are not short. A lot of syntax words were hard to understand. Due to these reasons, it made me so confusing by looking into these codes. It was quite hard for me to trace the structure of different modules and their relationship.

After spending a long time on understanding the code, I found a specific manual about the detail of that example and then knew more about the background of the program and its flow.

Problem 3:

LMC 01

The sample program required a lot of tools to modify including Matlab, DSP builder, Simulink. Therefore, I spent some time on studying how to use these tools. The video capture sample program required a lot of tools, so the compiler of Quartus 6.0 always runs for a long time and even lasts for an hour.

At the end of Week 5 in semester B, I changed to use the Xilinx FPGA board instead of Altera FPGA board. Datasheet and software study starts from very beginning again. It was quite a big problem for me to study datasheet and learn how to use Xilinx ISE in a short time.

Problem 4:

When I was going to study some sample program about video capture, it made me quite confusing. Verilog codes describing video data format and video capture part were complicated to understand. After studying a long time and searching relevant information on the internet, I have understood the meaning of the verilog code.

Problem 5:

During developing the watermarking algorithm in hardware, I found the difficulty of debugging due to the lack of debugging tools in verilog language. It was hard to trace the error source in a large program. Time is not enough for me to develop the last algorithm data hiding in images by adaptive LSB substitution based on the pixel-value differencing implementing in hardware.

7.2 Contribution:

A Real-Time Digital Watermarking Chip has been developed for copyright Protection and ownership authentication and ready for ASIC fabrication and commercialization.

7.3 Limitation:

In this project, the watermarking algorithm only implements the data hiding technique.

LMC 01

It does not involve the ability of withstanding the watermark attack. Therefore, the watermark will not be successfully extracted from the frame. In order to compensate this weakness, further development can be made.

8. Further Development:

In order to enhance the robustness of the watermarking technique, data communication error-correcting techniques can be used to protect from insider attacks. It can also utilize the performance of the chip by using DDR memory controller.

9. Conclusion:

Appropriate algorithms have been selected to implement by hardware. Two of three watermarking algorithms have successfully been implemented in hardware. The hardware implementation of the digital watermark chip was completed. Moreover, future development into commercial chip is viable.

10. Reference:

- [1] Suk-Ling Li, Kai-Chi Leung, L.M. Cheng, Chi-Kwong Chan, A novel image-hiding scheme based on block difference, *Pattern Recognition* 39 (2006) 1168-1176
- [2] Suk-Ling Li , Kai-Chi Leung, L.M. Cheng, Chi-Kwong Chan, Data Hiding in Images by Adaptive LSB Substitution Based on the Pixel-Value Differencing
- [3] Seitz, Juergen, *Digital Watermarking for Digital Media*, Hershey, PA, USA: Information Science Publishing, 2005. p. 8-p. 17.
- [4] <http://www-inst.eecs.berkeley.edu/~cs150/Documents/ITU656.doc>
- [5] James M. Lee, Verilog Quickstart, 3rd edition
- [6] Jesse Liberty, Programming C#,
- [7] XUPV2P User Guide
- [8] Stefan Katzenbeisser, Fabien A.P Petitcolas, Information Hiding Techniques for Steganography and Digital Watermarking, p.109-112

11. Appendix:

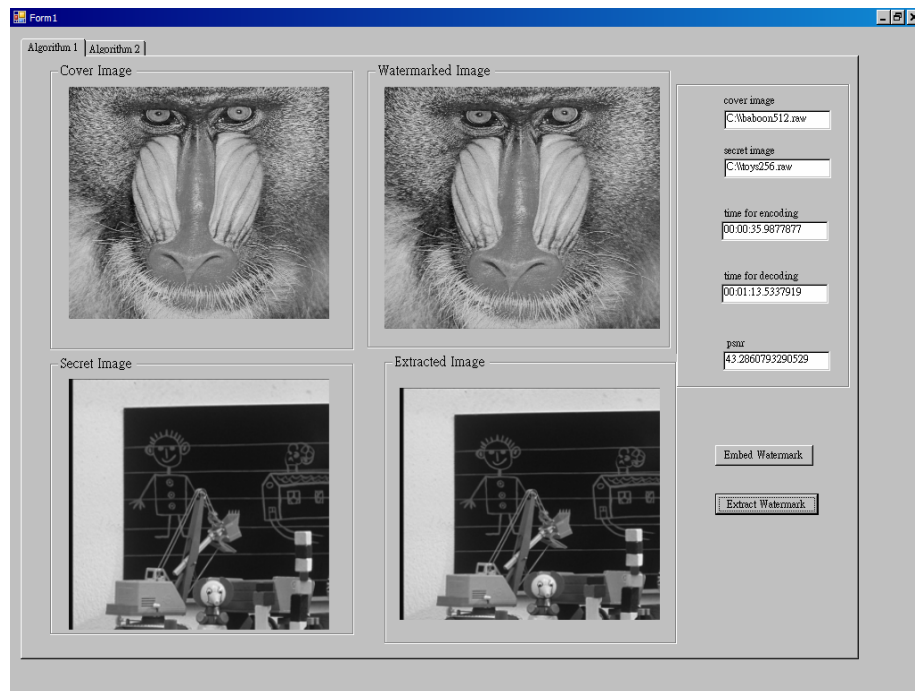


Fig.38: The program implementing the algorithm of a novel image hiding scheme based on block difference

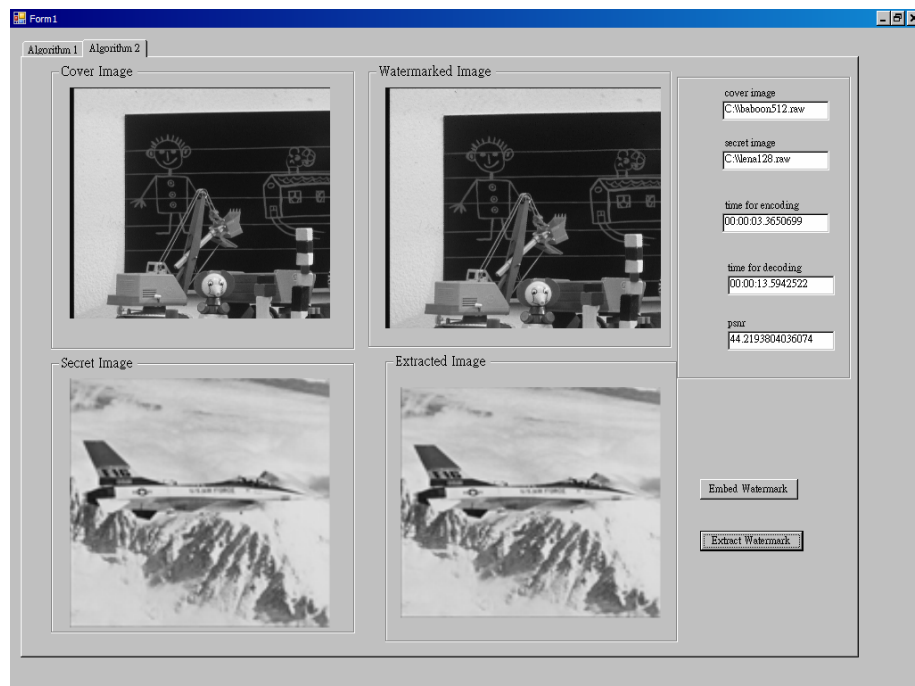
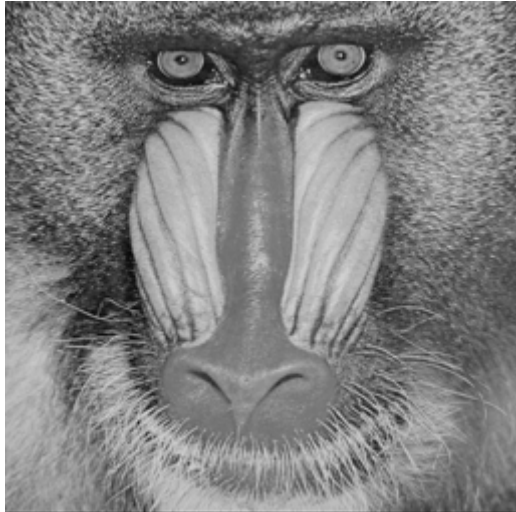


Fig.39: The program implementing the algorithm of Data Hiding in Images by Adaptive LSB Substitution Based on the Pixel-Value Differencing

LMC 01

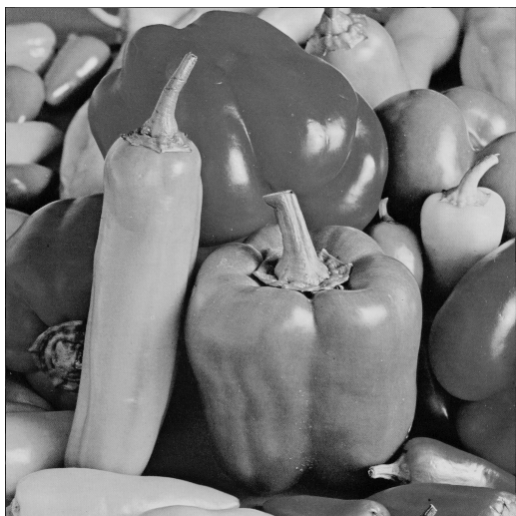
Raw images using for the algorithm of a novel image hiding scheme based on block difference:



(a)



(b)



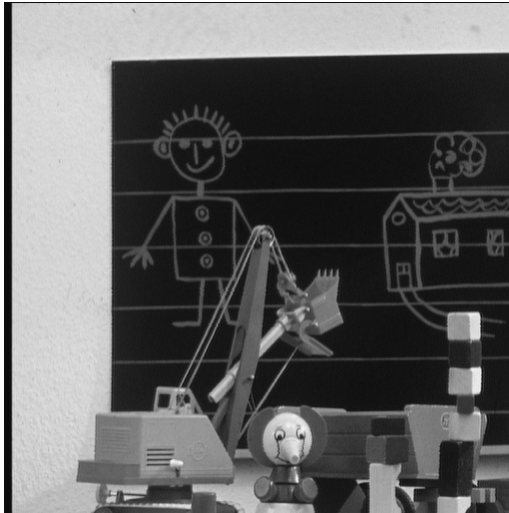
(c)



(d)

Fig. 40 :The cover-images of size 512 *512 pixels: (a) Baboon512; (b) Airplane512; (c) Pepper512; (d) Lena512.

LMC 01



(a)



(b)



(c)



(d)

Fig. 41: The secret-images of size 256 *256 pixels: (a) Toys256; (b) Jet256; (c) Bridge256; (d) Lena256.

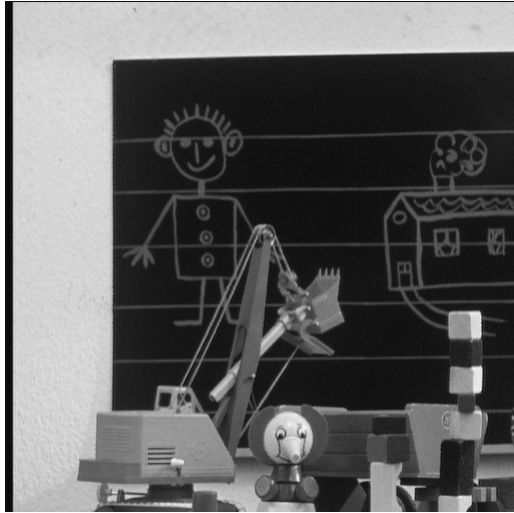
LMC 01



Fig.42: One of the watermarked images (Pepper512.raw) with secret image (Lena256.raw) by algorithm a novel image hiding scheme based on block difference

LMC 01

Raw images using for the algorithm of Data Hiding in Images by Adaptive LSB Substitution Based on the Pixel-Value Differencing:



(a)



(b)



(c)



(d)

LMC 01



(e)

Fig. 43: The cover-images of size 512 *512 pixels: (a) Toys512; (b) Pepper512; (c) Lena512; (d) Bridge512. (e) Airplane512

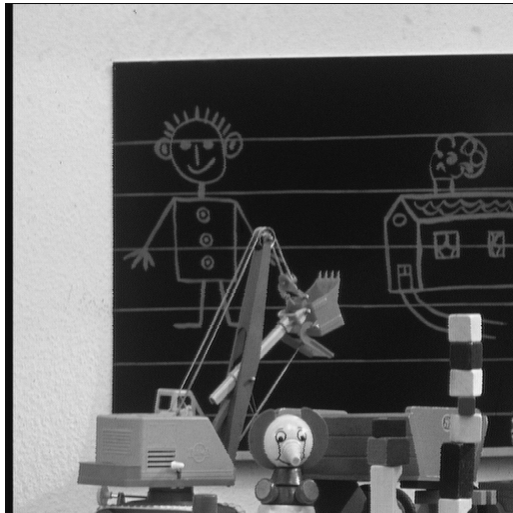


(a)

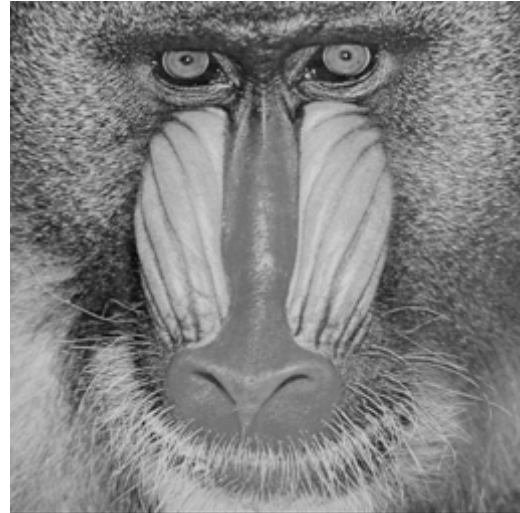


(b)

LMC 01



(c)



(d)

Fig.44: The secret-images of size 128 *128 pixels: (a) Jet128; (b) Lena128; (c) Toys128; (d) Baboon128



Fig.45: One of the watermarked images (Lena512.raw) with secret image (Toys128.raw) by Data Hiding in Images by Adaptive LSB Substitution Based on the Pixel-Value Differencing