Final Year Project Report

LMC 01

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
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
       4.1.4 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
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 XC2VP30 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\textsuperscript{[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 types 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\textsuperscript{[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
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.
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.
Signal-to-noise ratio: \[ SNR = \sum_{x,y} p_{x,y}^2 / \sum_{x,y} (p_{x,y} - \tilde{p}_{x,y})^2 \]

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

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](image)

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

<table>
<thead>
<tr>
<th>Reference-Block-Index</th>
<th>Quantized-Error-Matrix</th>
<th>Minimum Element in the Error-Matrix $\min(EM)$</th>
</tr>
</thead>
</table>

Fig.4 The format of the block header

4.2.2 Extracted Process [11]:

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 \textsuperscript{[2]}

![Fig. 5 The raster-scan order](image1)

![Fig. 6: Positions of $P_U$, $P_X$ and $P_L$](image2)

4.3.1 Data Embedding \textsuperscript{[2]}:

Let $P_u$ and $P_l$ be the upper and left neighbouring pixels of the input pixel $P_X$, and $g_u$ and $g_l$ 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

$$d = g_u - g_l.$$  \hfill (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.

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

When $n>4$, it set to be 4. Otherwise, the stego-image quality will be greatly reduced.
Then, an integer \( b \) which represents \( n \) bits secret data is placed in the \( n \) least significant bits of \( g_s \) and the new gray value \( g'_s \) is computed as
\[
g'_s = g_s - g_s \mod 2^n + b.
\] (3)

**Optimal pixel adjustment process:**

After replacing \( n \)-rightmost LSBSs of \( g_s \), the optimal pixel adjustment process will be applied to reduce the embedding error between \( g_s \) and \( g'_s \). Let \( \delta_s \) be the embedding error between \( g_s \) and \( g'_s \) and it is defined as
\[
\delta_s = g'_s - g_s, \quad \text{where } -2^n < \delta_s < 2^n.
\] (4)

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

\[
\hat{g}_s = \begin{cases} 
  g'_s - 2^n, & \text{if } 2^{n-1} < \delta_s < 2^n \text{ and } g'_s \geq 2^n; \\
  g'_s, & \text{if } 2^{n-1} < \delta_s < 2^n \text{ and } g'_s < 2^n; \\
  g'_s + 2^n, & \text{if } -2^n < \delta_s < -2^{n-1} \text{ and } g'_s \geq 2^n; \\
  g'_s, & \text{if } -2^n < \delta_s < -2^{n-1} \text{ and } g'_s < 2^n.
\end{cases}
\] (19)

By using OPAP, the absolute embedding error between pixels in the host-image and stego-image is limited to \( 0 \leq |\hat{g}_s - g_s| \leq 2^{n-1} \), 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
Let $n^*$ be the number of bits which can be extracted from the input pixel $P_x$. The value $n^*$ is calculated by

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

(6)

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

$$b = g_x^* \mod 2^n.$$  

(7)

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

**Example:**

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

Fig. 7: Part of the secret image

Fig. 8: Part of the cover image
**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.

$\therefore -16 < \delta_x < -8$ and $\delta_x = 4$ 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:
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>
<thead>
<tr>
<th>Video data format:</th>
</tr>
</thead>
<tbody>
<tr>
<td>The video data words are conveyed as a 27 Mword/s multiplex in the following order:</td>
</tr>
<tr>
<td>[ C_B, Y, C_R, Y, C_B, Y, C_R, \text{ etc.} ]</td>
</tr>
<tr>
<td>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.</td>
</tr>
</tbody>
</table>

| 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 |

### Table 1

<table>
<thead>
<tr>
<th>Field interval definitions</th>
<th>625</th>
<th>525</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>V-digital field blanking</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field 1 ((V = 1))</td>
<td>Start</td>
<td>Line 624</td>
</tr>
<tr>
<td>Field 2 ((V = 0))</td>
<td>Finish</td>
<td>Line 23</td>
</tr>
<tr>
<td><strong>F-digital field identification</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field 1</td>
<td>F = 0</td>
<td>Line 1</td>
</tr>
<tr>
<td>Field 2</td>
<td>F = 1</td>
<td>Line 313</td>
</tr>
</tbody>
</table>

**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 \( O_{470} \) 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.
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:**

<table>
<thead>
<tr>
<th>Data bit number</th>
<th>First word (FF)</th>
<th>Second word (00)</th>
<th>Third word (00)</th>
<th>Fourth word (XY)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 (MSb)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>F</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>V</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>H</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>P_3</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>P_2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>P_1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>P_0</td>
</tr>
<tr>
<td>1 (Note 2)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**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 D_3 and D_4 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 FAV

P_0, P_1, P_2, P_3: protection bits (see Table 3)

MSb: most significant bit

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

Bits P_0, P_1, P_2, P_3, 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

<table>
<thead>
<tr>
<th>Protection bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

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.
4.7 Xilinx XUP Virtex-II Pro Development Board

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.
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
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
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

![User Interface Diagram]

Fig.13: 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 $M \times M$ pixels, the PSNR value is defined as follows:

$$PSNR = 10 \times \log_{10} \frac{(2^n - 1)^2}{\frac{1}{M \times M} \sum_{i=0}^{M-1} \sum_{j=0}^{M-1} (p_{ij} - \hat{p}_{ij})^2}$$

**Test result by using different images:**

<table>
<thead>
<tr>
<th>Host image</th>
<th>Secret image</th>
<th>Timing for encoding</th>
<th>Timing for decoding</th>
<th>PSNR Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baboon512</td>
<td>Toys256</td>
<td>35.99 s</td>
<td>1:13 s</td>
<td>43.286 dB</td>
</tr>
<tr>
<td>Airplane512</td>
<td>Toys256</td>
<td>27.6 s</td>
<td>38.4 s</td>
<td>45.29 dB</td>
</tr>
<tr>
<td>Pepper512</td>
<td>Jet256</td>
<td>28.4 s</td>
<td>42.3 s</td>
<td>44.568 dB</td>
</tr>
</tbody>
</table>
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 $512 \times 512$ and the secret image size is $128 \times 128$.

**Test result by using different images:**

<table>
<thead>
<tr>
<th>Host image</th>
<th>Secret image</th>
<th>Timing for encoding</th>
<th>Timing for decoding</th>
<th>PSNR Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toys512</td>
<td>Jet128</td>
<td>3.36s</td>
<td>13.59s</td>
<td>44.219dB</td>
</tr>
<tr>
<td>Pepper512</td>
<td>Lena128</td>
<td>3.37s</td>
<td>11.82s</td>
<td>41.606dB</td>
</tr>
<tr>
<td>Lena512</td>
<td>Toys128</td>
<td>3.36s</td>
<td>10.76s</td>
<td>41.936dB</td>
</tr>
<tr>
<td>Bridge512</td>
<td>Baboon128</td>
<td>3.39s</td>
<td>7.05s</td>
<td>38.754dB</td>
</tr>
<tr>
<td>Airplane512</td>
<td>Lena128</td>
<td>3.37</td>
<td>16.62s</td>
<td>44.516dB</td>
</tr>
</tbody>
</table>

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

**Comparison of Difference Techniques for Real-time Hardware Implementation**

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

The FPGA board

Video Output

Video Input

VCD player

The FPGA board

Netlist

Verilog Code

Computer for generating the programmable file

Video

Output

Input

VCD

player

Fig 15: Hardware Implementation
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

Fig.16: Video Decoder 1 board

Fig17: Xilinx XC2VP30 development 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 Ycrcb 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 Ycrcb to 4:4:4 Ycrcb 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 Ycrcb 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
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.
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).
Check the syntax of the program

Generate the programmable file

Fig.20: Xilinx ISE 8.2i interface

Fig.21: 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.
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 FPGA board and the video daughter board. Then the FPGA will execute the watermark program and deliver the watermarked signal to the computer monitor through the VGA cable.

![Diagram of the hardware flow](Fig.24 The flow chart of watermarked hardware)
6.5 The Hierarchy of the Watermarked System

![Diagram of 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:**

<table>
<thead>
<tr>
<th>Bits Length</th>
<th>Signal Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 bits</td>
<td>YCrCb_in</td>
<td>YCrCb digital video data from VDEC1 daughter board</td>
</tr>
<tr>
<td>1 bit</td>
<td>LLC_CLOCK</td>
<td>Line Locked Clock (27MHz) from VDEC1 daughter board</td>
</tr>
<tr>
<td>1 bit</td>
<td>system_dcm_locked</td>
<td></td>
</tr>
<tr>
<td>1 bit</td>
<td>decode</td>
<td>The decode input</td>
</tr>
</tbody>
</table>
Final Year Project Report

Digital Watermarking Device

LMC 01

Output:

<table>
<thead>
<tr>
<th>Bits Length</th>
<th>Signal Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 bits</td>
<td>R</td>
<td>Red video signal</td>
</tr>
<tr>
<td>8 bits</td>
<td>G</td>
<td>Green video signal</td>
</tr>
<tr>
<td>8 bits</td>
<td>B</td>
<td>Blue video signal</td>
</tr>
<tr>
<td>1 bit</td>
<td>H_SYNC_Z</td>
<td>H SYNC output to SVGA connector</td>
</tr>
<tr>
<td>1 bit</td>
<td>V_SYNC_Z</td>
<td>V SYNC output to SVGA connector</td>
</tr>
<tr>
<td>1 bit</td>
<td>COMP_SYNC_Z</td>
<td>COMP SYNC output to DAC</td>
</tr>
<tr>
<td>1 bit</td>
<td>PIXEL_CLOCK</td>
<td>Pixel clock output to DAC</td>
</tr>
<tr>
<td>1 bit</td>
<td>BLANK_Z</td>
<td>BLANK output to DAC</td>
</tr>
<tr>
<td>1 bit</td>
<td>RESET_VDEC1_Z</td>
<td>RESET the video decoder on the VDEC1 daughter</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Bit Length</th>
<th>Signal Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 bit</td>
<td>rst</td>
<td>Reset and Clock signal</td>
</tr>
<tr>
<td>1 bit</td>
<td>clk</td>
<td>27MHz for SDTV</td>
</tr>
<tr>
<td>10 bits</td>
<td>YCrCb_in</td>
<td>Data from the input video stream</td>
</tr>
</tbody>
</table>

Output:

<table>
<thead>
<tr>
<th>Bit Length</th>
<th>Signal Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 bits</td>
<td>YCrCb_out</td>
<td>Data delayed by pipe length</td>
</tr>
<tr>
<td>1 bit</td>
<td>NTSC_out</td>
<td>High = NTSC format detected</td>
</tr>
<tr>
<td>1 bit</td>
<td>Fo</td>
<td>High = field one (even)</td>
</tr>
<tr>
<td>1 bit</td>
<td>Vo</td>
<td>High = vertical blank</td>
</tr>
</tbody>
</table>

6.5.3 Line Buffer Module:
Function: This module defines the video line buffer RAM.

Input:

<table>
<thead>
<tr>
<th>Bit Length</th>
<th>Signal Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 bit</td>
<td>read_clk</td>
<td>Read clock</td>
</tr>
<tr>
<td>1 bit</td>
<td>read_enable</td>
<td>Active high</td>
</tr>
<tr>
<td>11 bits</td>
<td>read_address</td>
<td>Buffer address</td>
</tr>
<tr>
<td>1 bit</td>
<td>write_clk</td>
<td>Write clock</td>
</tr>
<tr>
<td>8 bits</td>
<td>write_red_data</td>
<td>Red data input</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Bit Length</th>
<th>Signal Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 bit</td>
<td>clk</td>
<td>Clock</td>
</tr>
<tr>
<td>1 bit</td>
<td>data_in</td>
<td>Data input</td>
</tr>
<tr>
<td>1 bit</td>
<td>reset</td>
<td>Reset</td>
</tr>
</tbody>
</table>

Output

<table>
<thead>
<tr>
<th>Bit Length</th>
<th>Signal Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 bit</td>
<td>one_shot_out</td>
<td>signal will be high for one clock cycle after the input transitions high to low</td>
</tr>
</tbody>
</table>

6.5.5 Pipe_Line Delay Module:
Function: This module aligns the sync and blank signals with the video data.

Input

<table>
<thead>
<tr>
<th>Bit Length</th>
<th>Signal Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 bit</td>
<td>clk</td>
<td>Read clock</td>
</tr>
<tr>
<td>1 bit</td>
<td>rst</td>
<td>Active high</td>
</tr>
<tr>
<td>1 bit</td>
<td>hsync_in</td>
<td>Horizontal synchronization signal</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Bit Length</th>
<th>Signal Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 bit</td>
<td>pixel_clock</td>
<td>pixel clock</td>
</tr>
<tr>
<td>1 bit</td>
<td>reset</td>
<td>reset</td>
</tr>
<tr>
<td>11 bits</td>
<td>hsync_in</td>
<td>Horizontal synchronization signal</td>
</tr>
<tr>
<td>1 bit</td>
<td>vsync_in</td>
<td>Vertical synchronization signal</td>
</tr>
<tr>
<td>8 bits</td>
<td>blank_in</td>
<td>Blanking signal</td>
</tr>
<tr>
<td>8 bits</td>
<td>comp_sync_in</td>
<td>Composite synchronization signal</td>
</tr>
</tbody>
</table>

### Output

<table>
<thead>
<tr>
<th>Bit Length</th>
<th>Signal Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 bit</td>
<td>h_synch_delay</td>
<td>Horizontal synch for VGA connector</td>
</tr>
<tr>
<td>1 bit</td>
<td>v_synch_delay</td>
<td>Vertical synch for VGA connector</td>
</tr>
<tr>
<td>1 bit</td>
<td>comp_synch</td>
<td>Composite synch for DAC</td>
</tr>
<tr>
<td>1 bit</td>
<td>blank</td>
<td>composite blanking</td>
</tr>
<tr>
<td>3 bits</td>
<td>char_line_count</td>
<td>line counter for char gen rom</td>
</tr>
<tr>
<td>14 bits</td>
<td>char_address</td>
<td>character mode address</td>
</tr>
<tr>
<td>11 bits</td>
<td>pixel_count</td>
<td>Count pixels in a line</td>
</tr>
<tr>
<td>10 bits</td>
<td>line_count</td>
<td>Count lines in a frame</td>
</tr>
<tr>
<td>Bit Length</td>
<td>Signal Name</td>
<td>Comment</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>12 bits</td>
<td>addr</td>
<td>Memory address</td>
</tr>
<tr>
<td>1 bit</td>
<td>clk</td>
<td>Input clock</td>
</tr>
</tbody>
</table>

**Output**

<table>
<thead>
<tr>
<th>Bit Length</th>
<th>Signal Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 bits</td>
<td>dout</td>
<td>Memory output</td>
</tr>
</tbody>
</table>

**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**

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

**Output**

<table>
<thead>
<tr>
<th>Bit Length</th>
<th>Signal Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 bits</td>
<td>dout</td>
<td>Memory output</td>
</tr>
<tr>
<td>1 bit</td>
<td>ceo</td>
<td>output enable valid out put 1/2 ycrcb_in rate</td>
</tr>
<tr>
<td>1 bit</td>
<td>ntsc_out_o</td>
<td>NTSC format detected delayed to match 422-444 pipe length</td>
</tr>
<tr>
<td>1 bit</td>
<td>fo</td>
<td>high = field one (even) delayed to match 422-444 pipe length</td>
</tr>
<tr>
<td>1 bit</td>
<td>vo</td>
<td>high = vertical blank delayed to match 422-444 pipe length</td>
</tr>
<tr>
<td>1 bit</td>
<td>ho</td>
<td>low = active video delayed to match 422-444 pipe length</td>
</tr>
<tr>
<td>10 bits</td>
<td>y_out</td>
<td>4:4:4 luma data</td>
</tr>
<tr>
<td>10 bits</td>
<td>cr_out</td>
<td>4:4:4 chroma data</td>
</tr>
</tbody>
</table>
6.5.9 YCrCb2RGB module:

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

**Input**

<table>
<thead>
<tr>
<th>Bit Length</th>
<th>Signal Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 bit</td>
<td>clk</td>
<td>Synchronize Clock Signal</td>
</tr>
<tr>
<td>1 bit</td>
<td>rst</td>
<td>Reset Signal</td>
</tr>
<tr>
<td>10 bits</td>
<td>Y</td>
<td>Red data signal</td>
</tr>
<tr>
<td>10 bits</td>
<td>Cr</td>
<td>Green data signal</td>
</tr>
<tr>
<td>10 bits</td>
<td>Cb</td>
<td>Blue data signal</td>
</tr>
</tbody>
</table>

**Output**

<table>
<thead>
<tr>
<th>Bit Length</th>
<th>Signal Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 bits</td>
<td>read_red_data</td>
<td>Red data output</td>
</tr>
<tr>
<td>8 bits</td>
<td>read_green_data</td>
<td>Green data output</td>
</tr>
<tr>
<td>8 bits</td>
<td>read_blue_data</td>
<td>Blue data output</td>
</tr>
</tbody>
</table>

6.5.10 Encode module:

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

**Input**

<table>
<thead>
<tr>
<th>Bit Length</th>
<th>Signal Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 bits</td>
<td>R</td>
<td>Red data input</td>
</tr>
<tr>
<td>8 bits</td>
<td>G</td>
<td>Green data input</td>
</tr>
<tr>
<td>8 bits</td>
<td>B</td>
<td>Blue data input</td>
</tr>
<tr>
<td>11 bits</td>
<td>pixelcount</td>
<td>Pixel position</td>
</tr>
<tr>
<td>10 bits</td>
<td>linecount</td>
<td>Line position</td>
</tr>
<tr>
<td>1 bit</td>
<td>pixelclock</td>
<td>Pixel clock</td>
</tr>
</tbody>
</table>

**Output**

<table>
<thead>
<tr>
<th>Bit Length</th>
<th>Signal Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 bits</td>
<td>Rout</td>
<td>Red data output</td>
</tr>
<tr>
<td>8 bits</td>
<td>Gout</td>
<td>Green data output</td>
</tr>
<tr>
<td>8 bits</td>
<td>Bout</td>
<td>Blue data output</td>
</tr>
</tbody>
</table>
Final Year Project Report

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

<table>
<thead>
<tr>
<th>Bit Length</th>
<th>Signal Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 bits</td>
<td>R</td>
<td>Red data input</td>
</tr>
<tr>
<td>8 bits</td>
<td>G</td>
<td>Green data input</td>
</tr>
<tr>
<td>8 bits</td>
<td>B</td>
<td>Blue data input</td>
</tr>
<tr>
<td>11 bits</td>
<td>pixelcount</td>
<td>Pixel position</td>
</tr>
<tr>
<td>10 bits</td>
<td>linecount</td>
<td>Line position</td>
</tr>
<tr>
<td>1 bit</td>
<td>pixelclock</td>
<td>Pixel clock</td>
</tr>
</tbody>
</table>

Output

<table>
<thead>
<tr>
<th>Bit Length</th>
<th>Signal Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 bits</td>
<td>Rout</td>
<td>Red data output</td>
</tr>
<tr>
<td>8 bits</td>
<td>Gout</td>
<td>Green data output</td>
</tr>
<tr>
<td>8 bits</td>
<td>Bout</td>
<td>Blue data output</td>
</tr>
</tbody>
</table>
6.6 Demonstration of Watermarking

To configure the video decoder board, button (up) need to be pressed.
After downloading the programmable bit file (adv7183b_i2c.bit) to the FPGA and pressing Button (up), the LED 2 will light.

![LED2 lighted](image1)

**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.

![Monitor screen](image2)

**Fig.29: The monitor screen contains invisible watermark**

To verify the correctness of the watermark, switch ‘3’ can be turned on.
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.
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:

Red pixels data bit

<table>
<thead>
<tr>
<th>Pixel 1</th>
<th>10110100</th>
<th>10000101</th>
<th>01100100</th>
<th>11011111</th>
<th>01100100</th>
<th>01001000</th>
<th>10111011</th>
<th>10111011</th>
<th>10011001</th>
<th>10101001</th>
<th>01000011</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pixel 2</td>
<td>10110100</td>
<td>10000101</td>
<td>01100100</td>
<td>11011111</td>
<td>01100100</td>
<td>01001000</td>
<td>10111011</td>
<td>10111011</td>
<td>10011001</td>
<td>10101001</td>
<td>01000011</td>
</tr>
</tbody>
</table>

Step 1

Step 2

Buffer

Modified red pixel values

Memory Map of watermark

City University of Hong Kong
Department of Electronic Engineering

Fig. 34: The flow of the algorithm implementing in verilog language
**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.
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](image)

![Fig.37: The design of implementing watermark extraction algorithm](image)

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


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

Fig. 40: The cover-images of size 512 * 512 pixels: (a) Baboon512; (b) Airplane512; (c) Pepper512; (d) Lena512.
Fig. 41: The secret-images of size 256 * 256 pixels: (a) Toys256; (b) Jet256; (c) Bridge256; (d) Lena256.
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
Raw images using for the algorithm of Data Hiding in Images by Adaptive LSB Substitution Based on the Pixel-Value Differencing:
Fig. 43: The cover-images of size 512 *512 pixels: (a) Toys512; (b) Pepper512; (c) Lena512; (d) Bridge512. (e) Airplane512
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