

# 逢 甲 大 學

## 資 訊 工 程 學 系 專 題 報 告

### 電 子 相 框 影 像 控 制

學 生 ： 彭 德 鈞 （ 資 電 四 甲 ）

指 導 教 授 ： 王 益 文 老 師

中 華 民 國 九 十 三 年 十 二 月

# 摘要

在網路上，JPEG是很流行的影像格式，理由很簡單，在視覺上，可以達到我們人的需求，容量上，由於容量小的關係，可以說是帶給人們在網路上傳輸上的方便，在這中間的壓縮過程，如何將未壓縮的圖片濾掉不必要的資訊是我們感興趣的。了解JPEG，就可以知道圖片的資訊，像是可以知道是幾乘幾的圖片，可以透過圖檔的內容擷取出來，並加以利用，在圖片顯示在LCD上，計算出位置是有一定的必要。在按鍵上，也必須了解他的電路，在某個按鍵被觸發，必須判斷說該執行哪些功能。在圖片放大縮小上，要考慮的是如何在LCD上放大和縮小，在寫入資訊到LCD的過程中是需要一些技巧的。

我們知道，圖片可以做許多種轉變，經過某種演算法可以模糊化、柔化之類的，另外圖片的格式方面，相信大家都知道種類相當多，如果要將此專題做的更大一點，可以考慮多研究各種不同的格式的圖片，讓不同的格式的圖片放入記憶體，都可以依據不同圖片的header去判斷應該用哪一種decoder去解出來並顯示在LCD上，在寫入LCD的過程中，可以根據自己的需求做些改變，顯示出自己想要的效果，我想這是在未來，可以考慮做的東西。

關鍵詞：JPEG，LCD，SOPC Builder，Nios，PIO，SDRAM，RGB，push-button。

# 目錄

第一章 動機與目的.....	1
第二章 JPEG 介紹.....	2
第三章 設計環境.....	13
第四章 硬體配置.....	15
4.1 按鍵.....	15
4.2 LCD.....	18
4.3 System on a Programmable Chip.....	18
第五章 軟體部分.....	20
5.1 程式功能簡介.....	20
5.2 上下左右移動.....	21
5.3 放大縮小.....	21
5.4 其他功能.....	23
第六章 心得感想.....	24
6.1 心得.....	24
6.2 未來展望.....	24
附錄：	
成品圖片.....	25

## 圖表目錄

圖 1 JPEG 編碼系統架構.....	2
圖 2 色相轉換圖.....	4
圖 3 411 取樣圖.....	5
圖 4 211 取樣圖.....	6
圖 5 二維影像頻率分布圖.....	7
圖 6 亮度的量化矩陣.....	8
圖 7 彩度的量化矩陣.....	8
圖 8 編碼流程圖.....	9
圖9 DC霍夫曼編碼表.....	10
圖10 Zig-Zag掃描順序.....	11
圖11 AC係數的亮度霍夫曼表(部分) .....	12
圖 12 Nios Stratix Development Board.....	13
圖13 外加的4個push-button switches.....	14
圖14 120×160 16bits的LCD.....	14
圖15 button 電路圖.....	15
圖16 SOPC Builder PIO設定圖.....	16
圖17 Stratix內建按鈕pin.....	17
圖18 Expansion Prototype Connector - J15.....	17
圖19 Assignment Editor.....	17
圖20 Nios system settings.....	18

圖21 Nios SDK Shell.....	19
圖22 LCD 示意圖.....	20
圖23 像素擴展.....	21
圖24 像素相對座標.....	22
圖25 像素縮減.....	22
圖26 原圖.....	25
圖27 縮小.....	25
圖28 放大.....	25
圖29 成品照片.....	26

# 第一章 動機與目的

## 1.1 動機

在網路上，大部分圖片都是以 JPEG (Joint Photographic Experts Group) 格式，由於 JPEG 可以將 BMP (Bit Map) 格式圖片大量壓縮，以及容量小的優點，使得網路上傳輸上方便，再加上看到學長將 JPEG 解出來，再經由  $120 \times 160$  的 LCD 顯示出來，就覺得很有趣，於是就想了解 JPEG 的壓縮原理，進而去修改學長的程式，做出其他不一樣的效果，和一些控制上的功能。

## 1.2 目的

修改學長的程式使得任意大小的 jpeg 的圖片放入 Altera Nios 的 SDRAM，可以在  $120 \times 160$  的 LCD 顯示出來，並經由板子上的按鈕作控制，可以使得圖片在 LCD 上的畫面可以做上下左右的移動，還有的功能就是將圖片做放大和縮小，還有就是換下一張圖片的按鈕，以及額外再做個可以將彩色圖片轉成黑白的功能按鈕。

## 第二章 JPEG 介紹

### 2.1 JPEG 編碼架構

一開始我們先來了解一下 JPEG 的編碼系統架構，圖 1 為 JPEG 的編碼系統架構圖，然而此架構的逆向過程即可形成 JPEG 的解碼系統。

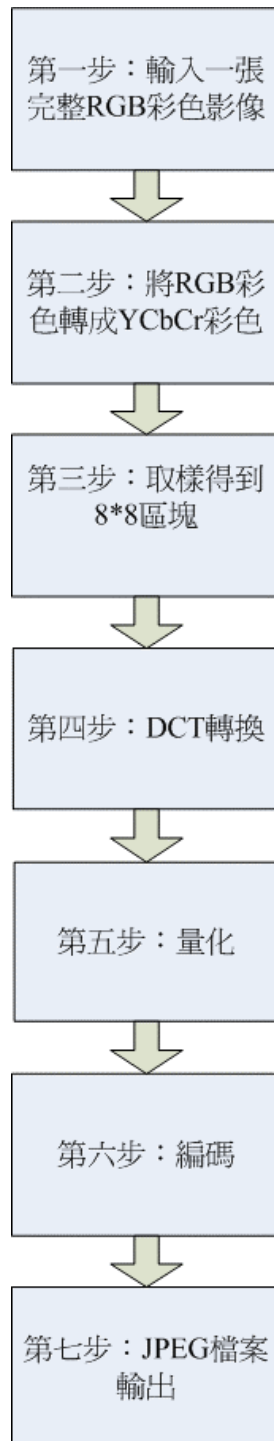


圖 1 JPEG 編碼系統架構

JPEG 壓縮技術不是只有一種模式，它可以細分為四種模式，分別為基本壓縮模式(baseline sequential coding)、漸進式壓縮模式(progressive coding)、無損編碼模式(lossless coding)、階層模式(hierarchical coding)。然而在這個專題中，我們討論的都是基本壓縮模式，其他三個模式都是透過基本壓縮模式，再加以改良而來的。

## 2.1.1 RGB 轉成 YCbCr

RGB 就是 Red, Green, Blue 三原色，彩色系統中除了 RGB，另外還有好幾種，如：YIQ、YUV、YCbCr 等等。任兩種彩色系統彼此之間都存在某種數學關係，如：

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.144 \\ 0.576 & -0.275 & -0.321 \\ 0.212 & -0.528 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

由式中彩色圖形可以視為多個分量組合，需分別處理。

假設輸入一張全彩的影像，包含 R、G、B 三個分色，然後 JPEG 會利用色相轉換的方法將其轉變為 Y、Cb、Cr 三種不同的分量(Y 表示亮度，Cb、Cr 表示彩度)。由於 R、G、B 三分量的關連性很高，同樣的資料常常出現在每個分量中，產生了空間上的浪費，再加上人的眼睛對彩度的感覺不像亮度這麼敏銳，所以用彩度和亮度分離的模式，也因此我們可以去掉許多彩度的資料，保留完整的亮度資料，這樣就叫 sampling，由於資料會有部份被刪除的關係，所以它是一種失真的壓縮演算法。轉換公式如下：

$$Y = 0.299R + 0.587G + 0.114B$$

$$Cb = -0.168R - 0.331G + 0.499B$$

$$Cr = 0.500R + 0.419G - 0.081B$$



## 2.1.2 取樣得到 $8 \times 8$ 區塊

在 JPEG 中，最小解碼單位皆為  $8 \times 8$  的大小。假設原圖為一張  $16 \times 16$  像素的基本區塊，經過色相轉換後會得到圖 2 中的 Cb、Cr、Y 三個區塊。

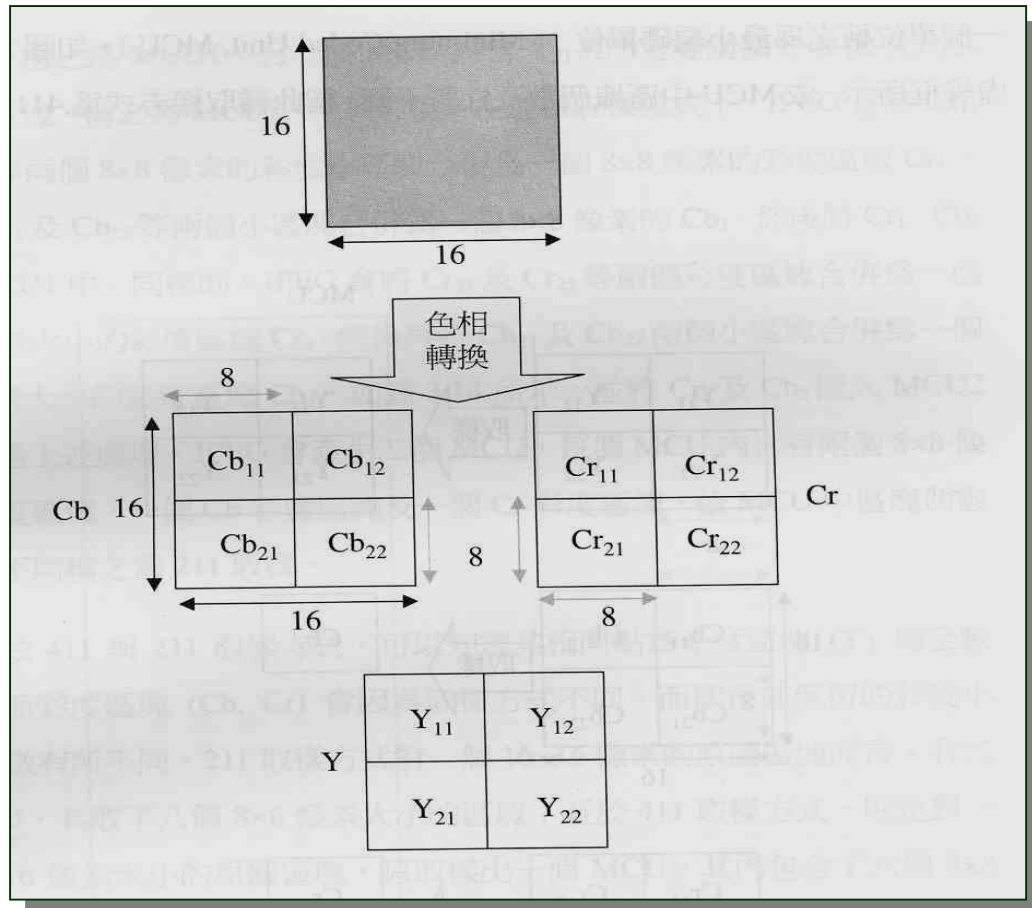


圖 2 色相轉換圖

JPEG 的取樣方式有兩種，一種叫做 411 取樣法，另一種叫做 211 法。若以 411 取樣時，可以得到 6 個  $8 \times 8$  矩陣，如圖 3，這幾個矩陣在 JPEG 中稱為 MCU (Minimum Coded Unit)，MCU 表示 JPEG 檔案中儲存壓縮資料的基本單位。其中 Y 的資料，也就是亮度的資料，就如前述，資料不與以取樣，保持不變，而 MCU 裡的 Cb1 是由  $Cb_{11}$ 、 $Cb_{12}$ 、 $Cb_{21}$ 、 $Cb_{22}$  這四塊  $8 \times 8$  矩陣中 64 個係數平均算出來的，同理 Cr1 也是用此法得出。

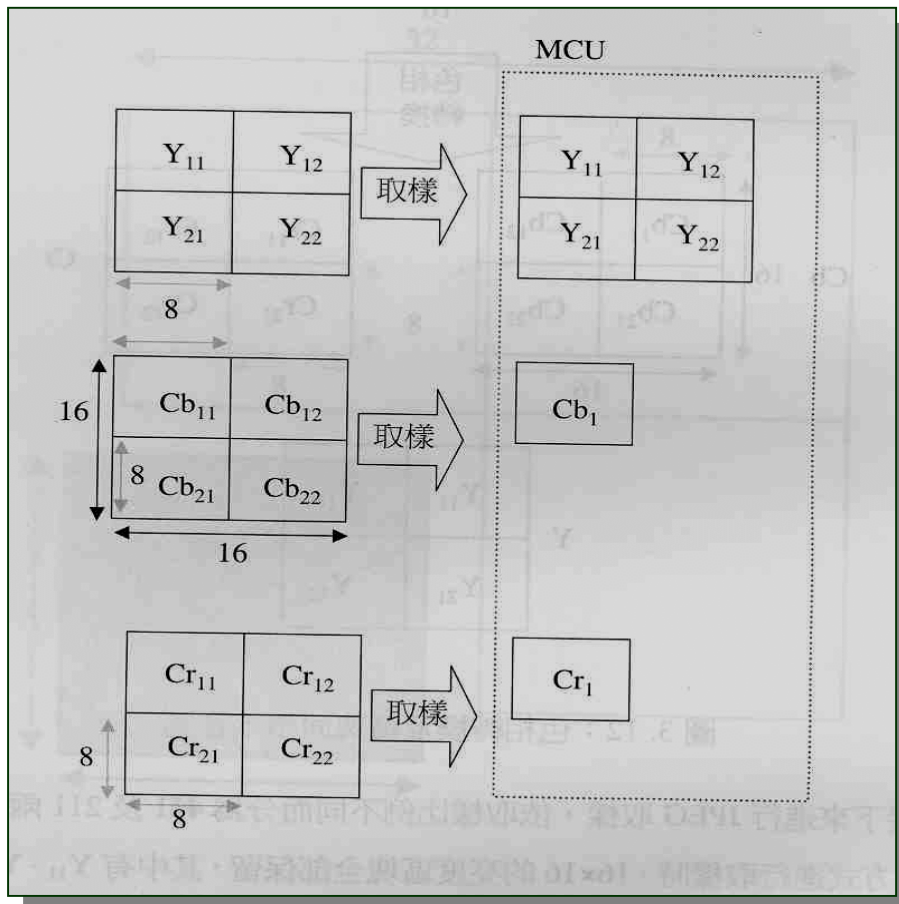


圖 3 411 取樣圖

若以 211 法取樣，如圖 4，16×16 的矩陣會產生兩個 MCU，Y11 和 Y12 資料不與以修改，放入 MCU1，Y21 和 Y22 放入 MCU2，MCU1 中的 Cb1 由 Cb11 和 Cb12 兩個 8×8 矩陣中的 64 個係數平均算出的，同理，Cr1 也是這樣算出的。同樣的，Cb2 由 Cb21 和 Cb22 合併而成，Cr2 由 Cr21 和 Cr22 合併而成，然後放入 MCU2。

從上面的 411 和 211 法的取樣過程中，我們可以知道用 411 法會有 6 個 8×8 矩陣，而用 211 法會有 8 個 8×8 矩陣，也因為 211 法的取樣出來的矩陣區塊比 411 多，所以在品質上 211 法會比 411 法的方式好，但相對，211 的取樣法會使用較多的空間。

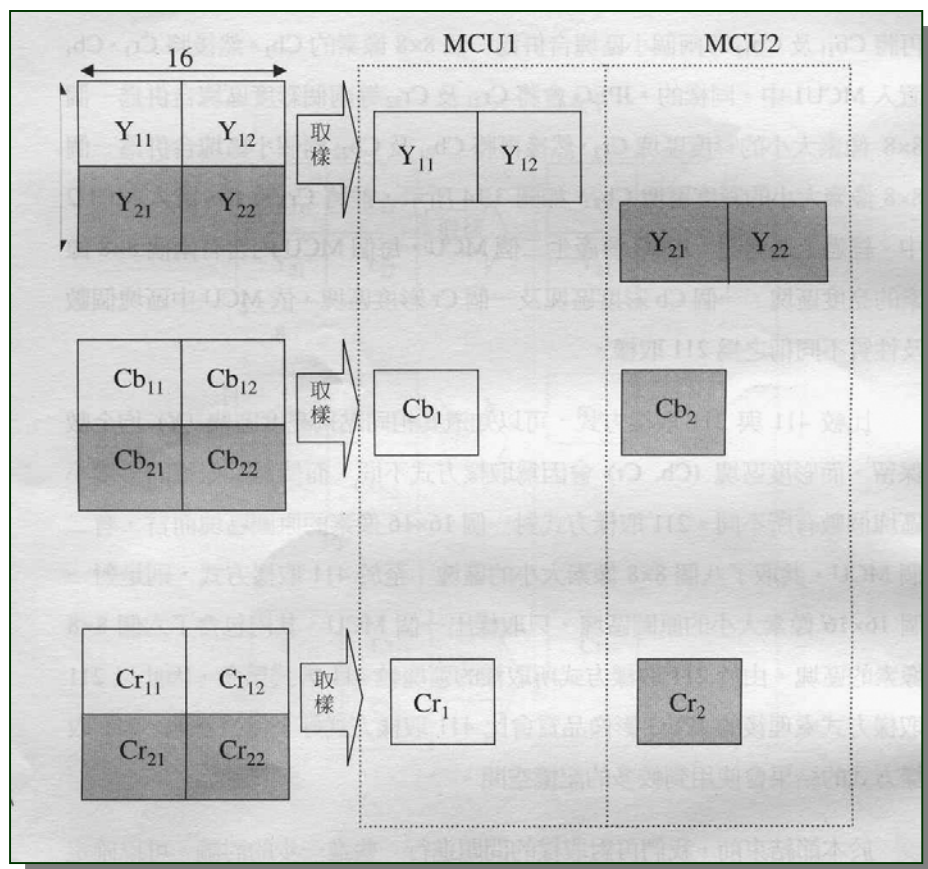


圖 4 211 取樣圖

## 2.1.3 DCT 轉換

在 MCU 中每一個  $8 \times 8$  的矩陣，一開始先將矩陣中的 64 個係數，都減去 128，這個動作的主要目的是要將原本介於 0 到 255 的灰階像素值，調整成介於 -128 到 127 之值，以利後續的影像處理。接著再進行 DCT 的轉換，所謂 DCT 轉換，就是透過下面的公式將空間域轉換成頻率域：

$$F(u, v) = \frac{1}{4} C(u) C(v) \sum_{x=0}^7 \sum_{y=0}^7 f(x, y) \cos \frac{((2x+1)u\pi)}{16} \cos \frac{((2y+1)v\pi)}{16}$$

$$C(u) C(v) = \frac{1}{\sqrt{2}}, \text{ 當 } u=v=0 \text{ 時,}$$

$$C(u) C(v) = 1, \text{ 當不是在 } u=v=0 \text{ 的情況下,}$$

其中  $x, y$  為原影像的矩陣座標， $u, v$  為 DCT 後頻率空間的座標。

DCT 轉換後每個  $8 \times 8$  的像素矩陣會產生一個 DC 係數，位在矩陣的  $(0,0)$  位置，剩下 63 個位置就是 AC 係數，如圖 5，經過 DCT 轉換後可大致分為低頻、中頻、高頻，越靠近 DC，頻率越低，資料的重要性也越高，因為人類對高頻的東西比較不敏感，所以在之後的量化技巧，就是要保留重要的資料，也就是頻率較低的部份，把高頻資料刪除，以達到壓縮的效果。

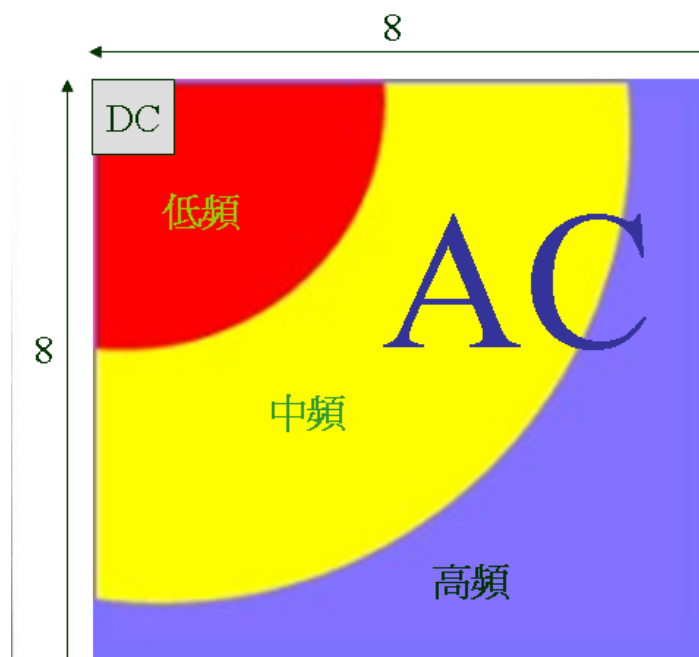


圖 5 二維影像頻率分布圖

## 2.1.4 量化

進行量化是要將  $8 \times 8$  DCT 轉換後的亮度及彩度頻率區塊再繼續做處理的動作，由 DCT 轉換那節我們可以知道，重要的資料主要集中在 DC 和 DC 附近，所以量化矩陣有個特色，就是矩陣中的係數，見圖 6、圖 7，左上的部份係數會比較小，越往右下的係數會越來越大，而量化的作法就是將 DCT 矩陣中的 64 個係數個別除以量化矩陣的整數，相除後四捨五入得整數商，這個商就是我們要的結果，我們記做  $DCT^Q$ 。

比較圖 6 和圖 7，我們可以知道虛線內範圍的係數，數值都比較大，數



值越大，壓縮的越多，當然品質會下降，再來就是圖 7 的虛線範圍之所以會比較大的原因，是因為人類對亮度比較敏感，所以壓縮的部份會比較少，保留更多亮度的資訊，至於彩度就可以壓比較多。

在量化的過程中，因為有做四捨五入的關係，會捨棄一些資訊，一定會有誤差，這也是 JPEG 是失真壓縮的原因之一。另外還有要注意的是，每一個 JPEG 影像所用的量化表都必須存放在檔頭中，以做為將來解壓縮的依據。

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

圖 6 亮度的量化矩陣

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

圖 7 彩度的量化矩陣

## 2.1.5 編碼

經過量化的處理後，接著 JPEG 就進入下圖 8 的程序，首先，先來看

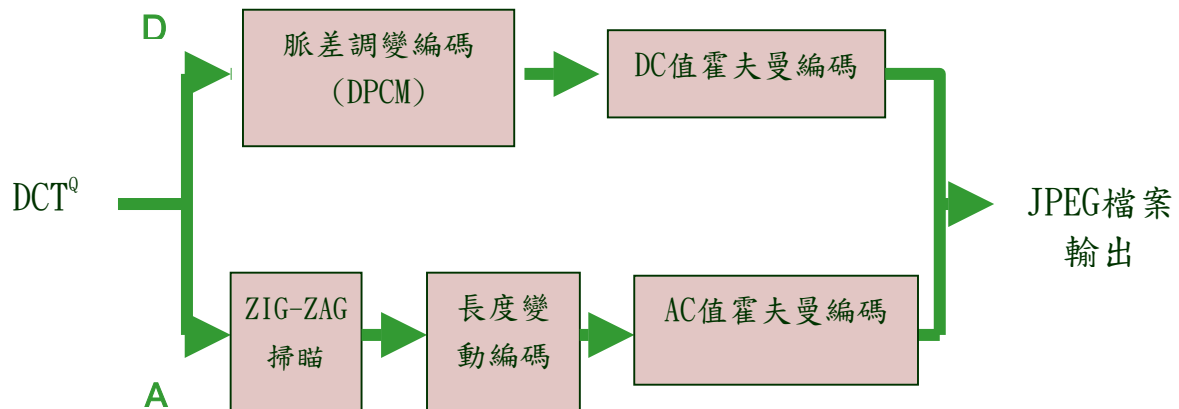


圖 8 編碼流程圖

DC 的部份，第一步要說明的是 DPCM。由於 DC 係數在影像中是很重要的資訊，所以經過量化後，通常是不為零的數值，而其餘的 AC 係數通常經過量化處理過後會出現較多的零，這也是要分開處理的原因。DPCM (Differential Pulse Code Modulation) 是取每個子影像分量的 DC 值與前一個 DC 值的差，它的運算式如下：

$$\text{DiffDC}(i) = \text{DC}(i) - \text{DC}(i-1)$$

至於第一個 DC 無前一個 DC 供運算，所以我們假設它的前一個 DC 值為 0。

Ex: DC value: 1 2 3 4 5

$$\begin{array}{r} -) 0 \ 1 \ 2 \ 3 \ 4 \\ \hline \end{array}$$

DiffDC: 1 1 1 1 1

經 DPCM 後的 DC 差值會遠比原 DC 值小，這是因為連續區塊中的像素值通常很相似，所以 DPCM 有效的將 DC 變小，接下來是霍夫曼編碼，霍夫曼編碼的精神主要是將一個數列中，經常出現的數值以較少位元數來表示，而不常出現的數值，則以較長的位元數來表示，如此來達到壓縮的效果。

JPEG 的霍夫曼編碼是依查表的方式來進行編碼，編碼 DC 的霍夫曼表是固

定的，如圖9所示，其中DiffDC之位元長度是指DiffDC的數值以二進位的1的補數進行編碼後所需的最小位元數。若DiffDC來自亮度區塊的DC差值，他會以<亮度編碼字，DiffDC>的格式輸出，同理，來自彩度區塊的DC差值，他會以<彩度編碼字，DiffDC>的格式輸出，例如：假設說來自彩度區塊的DiffDC值為30，以二進位來表示為 $(11110)_2$ ，由30去查表，可知彩度編碼字為 $(11110)_2$ ，所以輸出值為 $(1111011110)_2$ 。

DiffDC之位元長度	DiffDC	亮度編碼字的位元長度	亮度編碼字	彩度編碼字的位元長度	彩度編碼字
0	0	2	00	2	00
1	-1,1	3	010	2	01
2	-3,-2,2,3	3	011	2	10
3	-7,-4,4,7	3	100	3	110
4	-15,-8,8,15	3	101	4	1110
5	-31,-16,16,31	3	110	5	11110
6	-63,-32,32,63	4	1110	6	111110
7	-127,-64,64,127	5	11110	7	1111110
8	-255,-128,128,255	6	111110	8	11111110
9	-511,-256,256,511	7	1111110	9	111111110
10	-1023,-512,512,1023	8	11111110	10	1111111110
11	-2047,-1024,1024,2047	9	111111110	11	11111111110

圖9 DC霍夫曼編碼表

接下來是處理AC的部份，由於AC值沒有像DC值那樣的重要，所以AC值在量化時，許多AC係數都變成零，以達到壓縮的效果，由圖8，我們可以知道AC的壓縮是以長度變動編碼來完成的，在編碼前，必須將二維 $8 \times 8$ 像素大小的DCT係數區塊轉換成一維的係數陣列，而JPEG的作法是以Zig-Zag掃描，見圖10，從圖中可以知道，它是以左上角漸漸向右下角移動，依序存入陣列的順序，越先掃描的AC係數越重要，越後面被掃描的AC係數的重要性越低。







接下來下一步就是AC的霍夫曼編碼，基本上它和DC霍夫曼編碼是一樣的，都是用查表方式來做，不同的地方是AC霍夫曼編碼是以<R,L length><L>來進行編碼，而DC霍夫曼編碼是以DiffDC來進行編碼。

在進行AC霍夫曼編碼時，JPEG是以<R, L length>進行查表，以得到編碼字，然後再將這個編碼字與L合併起來，就是我們所要輸出的資料。

Ex：<R, L length><L>格式：<0, 2><00>，經由圖11的查表

⇒ 輸出值：0100 （在這裡是假設處理亮度的資料）。

R, L length	字碼長度	編碼字
0, 0 (EOB)	4	1010
0, 1	2	00
0, 2	2	01
0, 3	3	100
0, 4	4	1011
0, 5	5	11010
0, 6	7	1111000

圖11 AC係數的亮度霍夫曼表(部分)

至於解壓縮的部份就是壓縮流程的反過程，所以可以說是差不多的東西，在這裡就不多加介紹。有了以上JPEG的觀念，修改學長的程式也比較有概念。

## 第三章 設計環境

### 3.1 使用平台

- Stratix EP1S10F780C6 device
- 1 Mbytes of static RAM
- 16 Mbytes of SDRAM
- JTAG Connectors to Altera devices via Altera download cables
- One RS-232 DB9 serial ports
- Four push-button switches connected to Stratix user I/O pins
- 一個120×160 16bits的LCD
- 額外加上4個push-button switches

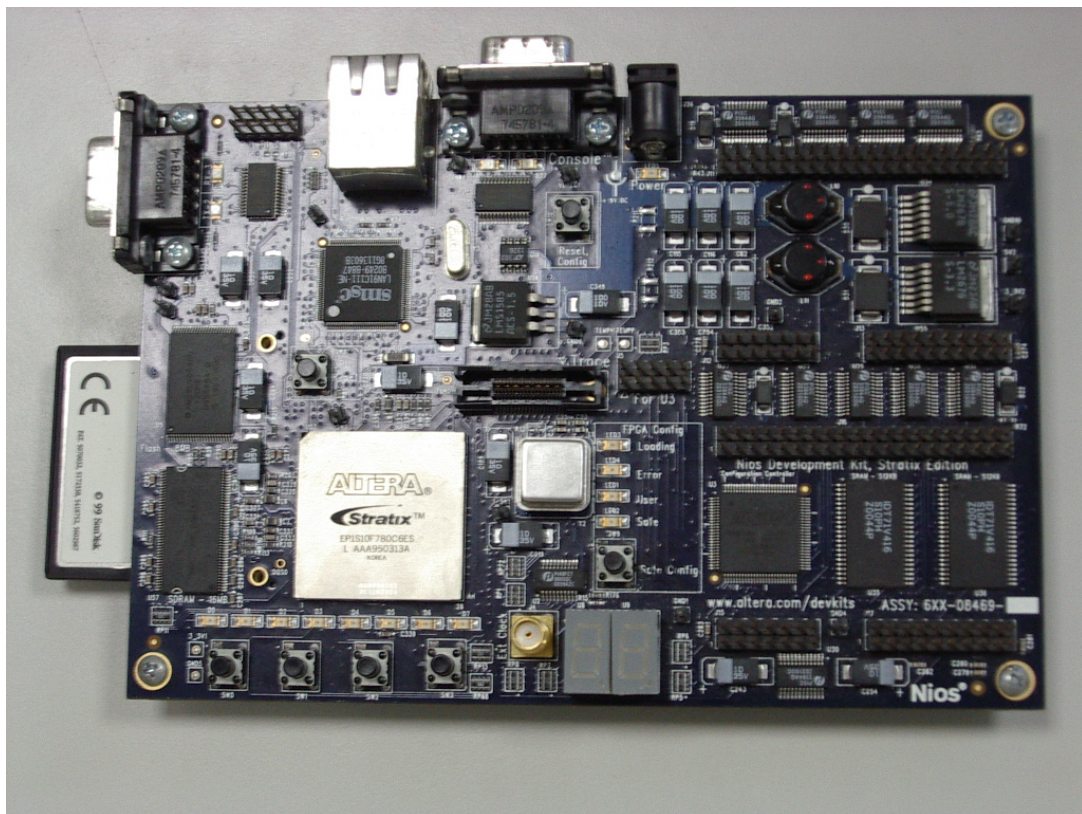


圖 12 Nios Stratix Development Board

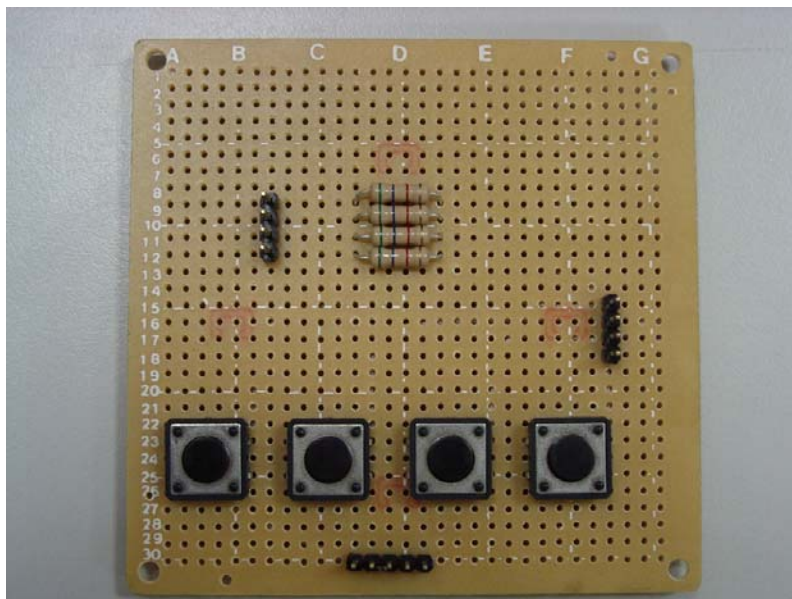


圖13 外加的4個push-button switches

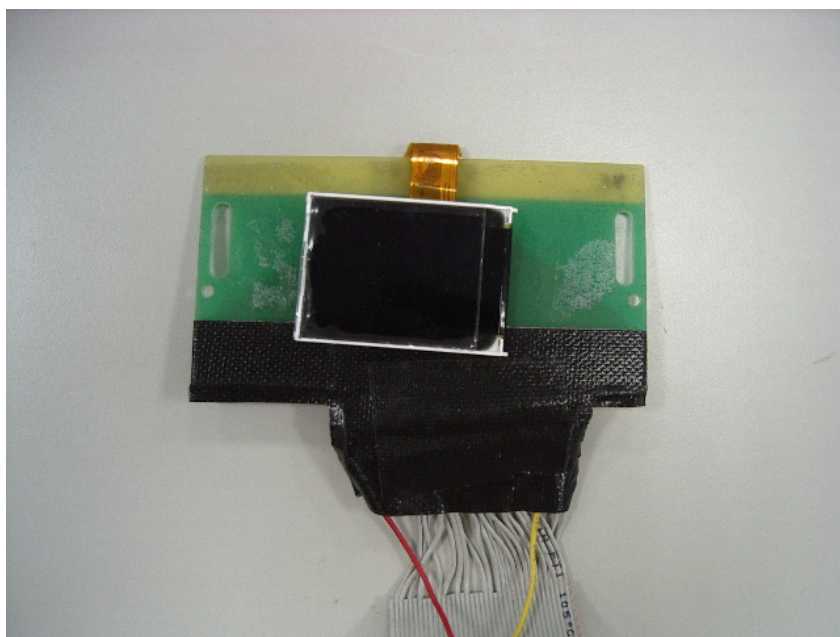


圖14 120×160 16bits的LCD

## 3.2 軟體發展工具

- Nios 3.1 32bit CPU & SDK
- QuartusII 3.0 & SOPC Builder 3.0

## 第四章 硬體配置

### 4.1 按鍵

如圖15為Stratix電路板上push-button的電路圖，在按鍵沒有按下去的情況下，use\_PB由於與VCC3\_3相接的關係，所以是高電壓，也就是訊號為1，若是按鍵按下去的話，會與GND接通，這時use\_PB訊號為0，所以在程式的設計上，剛開始所有的按鍵上的訊號都是1，只要某一個按鍵被按下去的話，那個按鍵的訊號就會變成0，然而在Stratix電路板上有4個按鍵，所以在這裡我們給每個按鍵都有一個功能，判斷哪一個按鍵變成0就去做我們對這個按鍵觸發後所要做的功能。

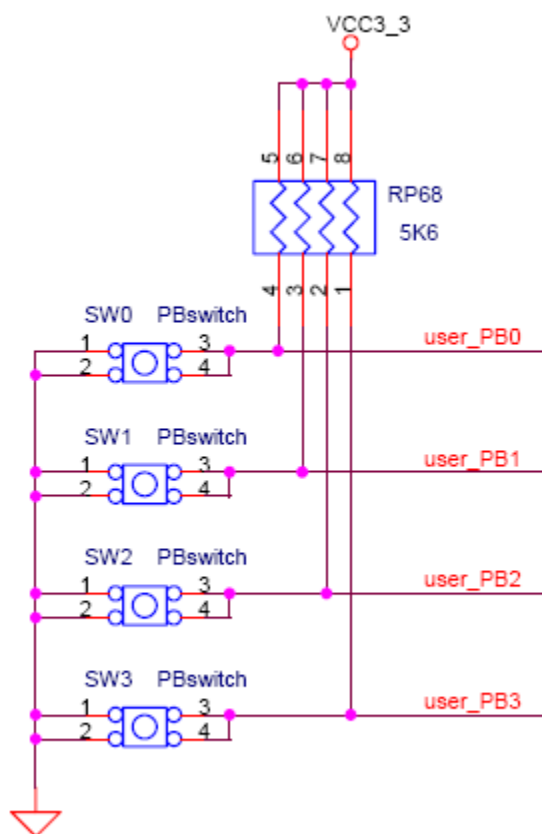


圖15 button 電路圖

由於按鍵只有四個，所以只能做4個功能，對於這次的專題似乎不夠，



於是仿照著圖15的電路圖，額外多加了4個按鍵，但由於原本Standard32（標準的硬體範例）在按鍵上只有4個bits，只能做4個功能，所以我們透過SOPC Builder來改變成8個bits，以便做出8個功能的按鍵，如圖16，點選PIO(parallel I/O)，可以出現中間的框架，將Width改成8bits即可。

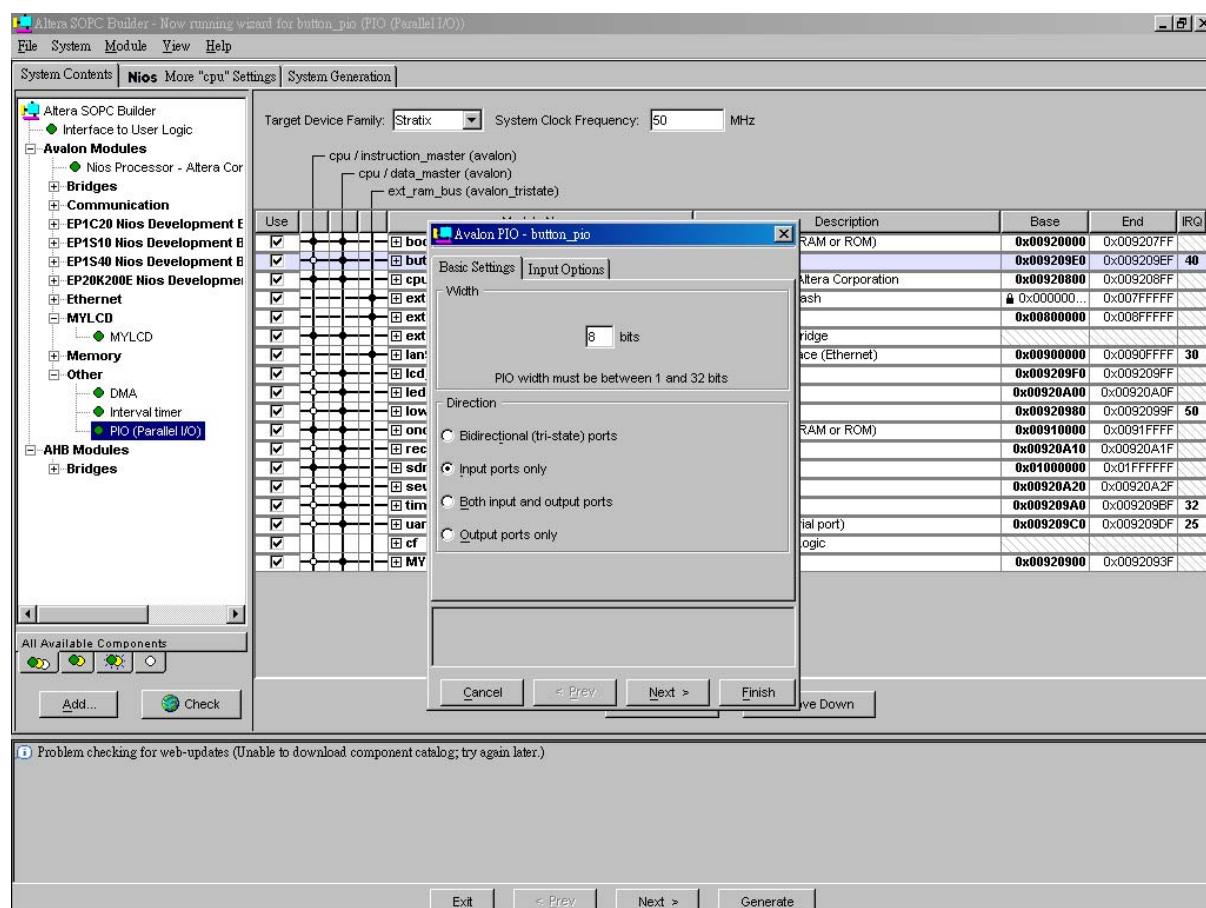


圖16 SOPC Builder PIO設定圖

另外還有就額外那個4個按鍵，我們必須作Pin assignment的動作，而圖17是Stratix電路板裡按鍵的pin，而圖18，圖中左下角的紅色框框裡的那四個pin，在實作中，我是將額外的4個按鍵，接到這4個pin上，接下來是用Quaruts II中的Assignment Editor做assign pin的動作，見圖19，可以發現將Width由4bits改成8bits後，USER\_PB[0]到USER\_PB[3]會變成USER\_PB[0]到USER\_PB[7]，多了4個讓我們設定，所以將圖18的紅色框框中的4個Pin填入即完成assign pin的動作。

Button	SW0	SW1	SW2	SW3
Stratix Pin	W5	W6	AB2	AB1

圖17 Stratix內建按鈕pin

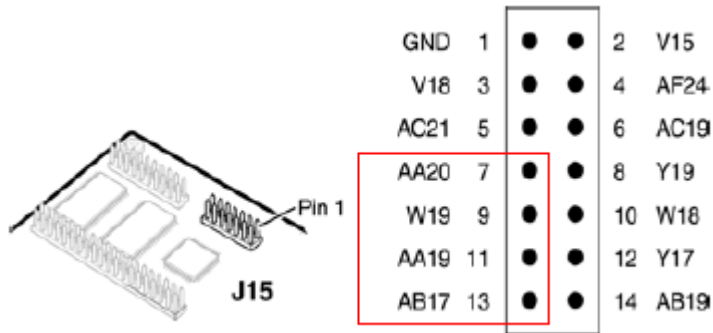


圖18 Expansion Prototype Connector - J15

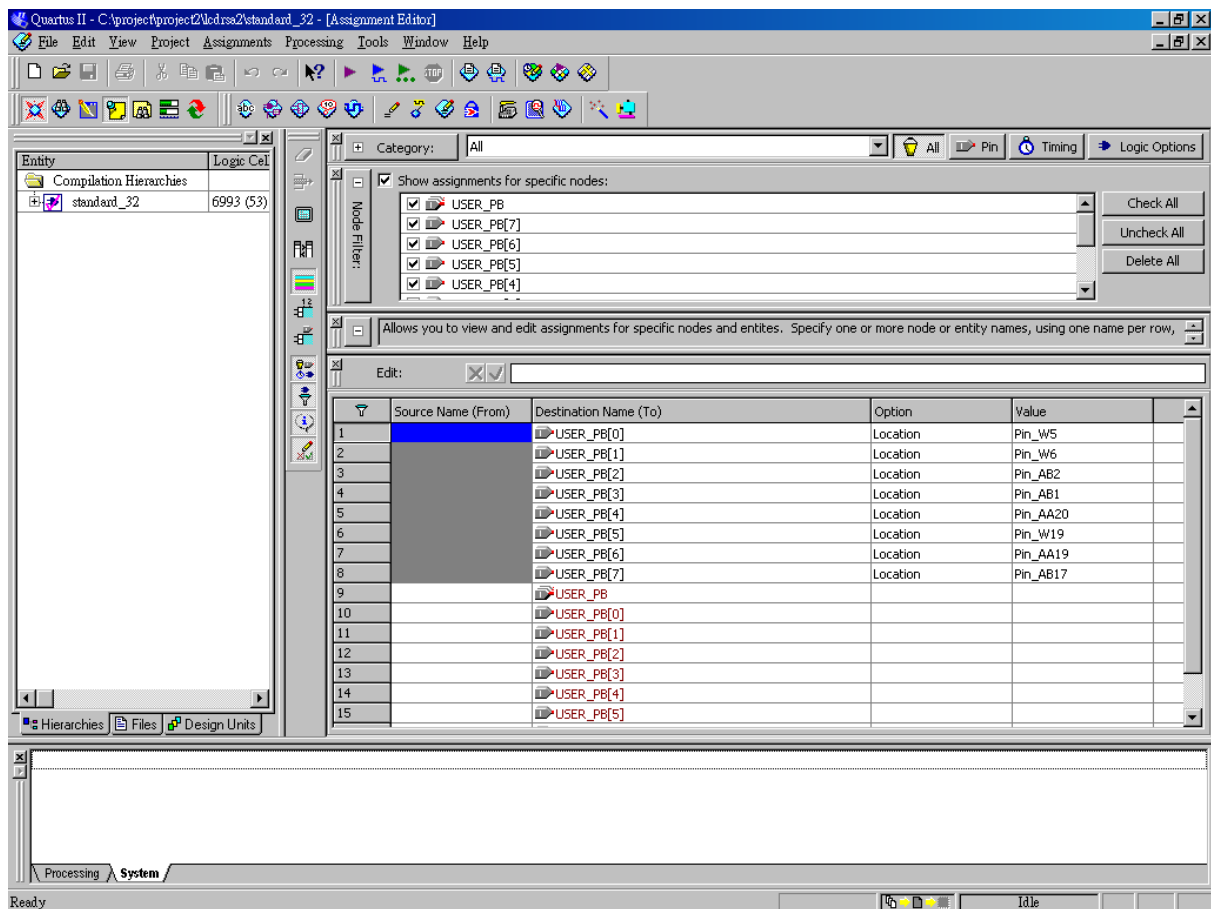


圖19 Assignment Editor

## 4.2 LCD

120×160的LCD，他的controller已經內建在lcd上了，在加上學長之前已經用hdl把溝通介面做好了，還有LCD的driver 的c code也寫好了，所以有用到就是學長寫的函式，讓我丟資料給LCD，顯示在LCD上面。用到的函式有lcd\_init()，主要是對LCD最初始化的動作，還有就是LCD\_WRITE16()，它有兩參數，第一個參數是塞入一個像素前半段的資料，第二個參數是塞入一個像素後半段的資料，共16bits。

## 4.3 System on a Programmable Chip

由於學長的程式，如果需要放入大的圖片進去，如1024×768這樣大的圖的話，在程式中需要宣告這樣大的陣列，然而這個程式若在只有1Mbyte的sram是絕對不夠用的，所以我將program memory由sram改成在有16Mbytes的sdram上執行，如圖20的紅色框框，如此就可以解決空間不夠用的問題。

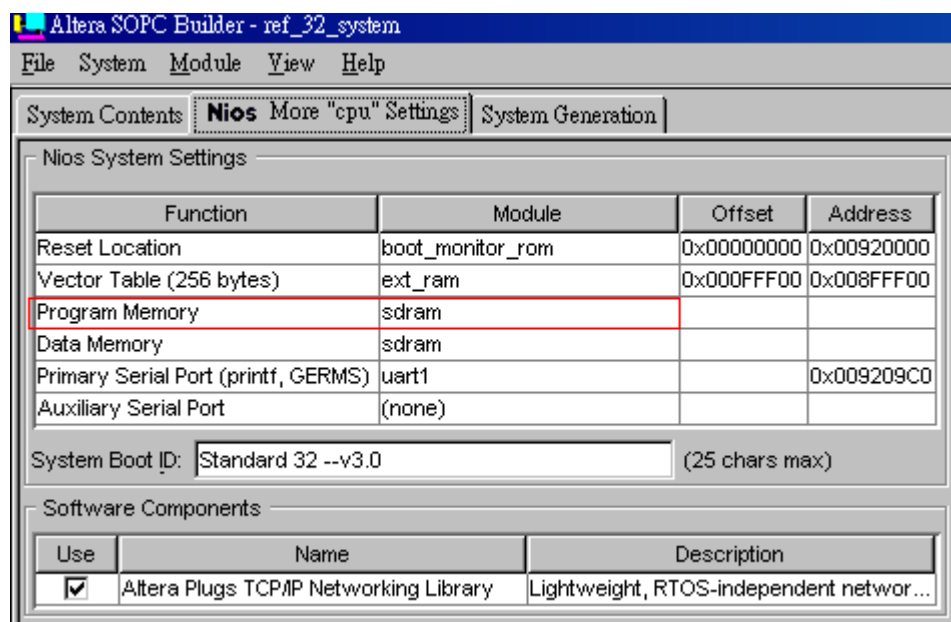
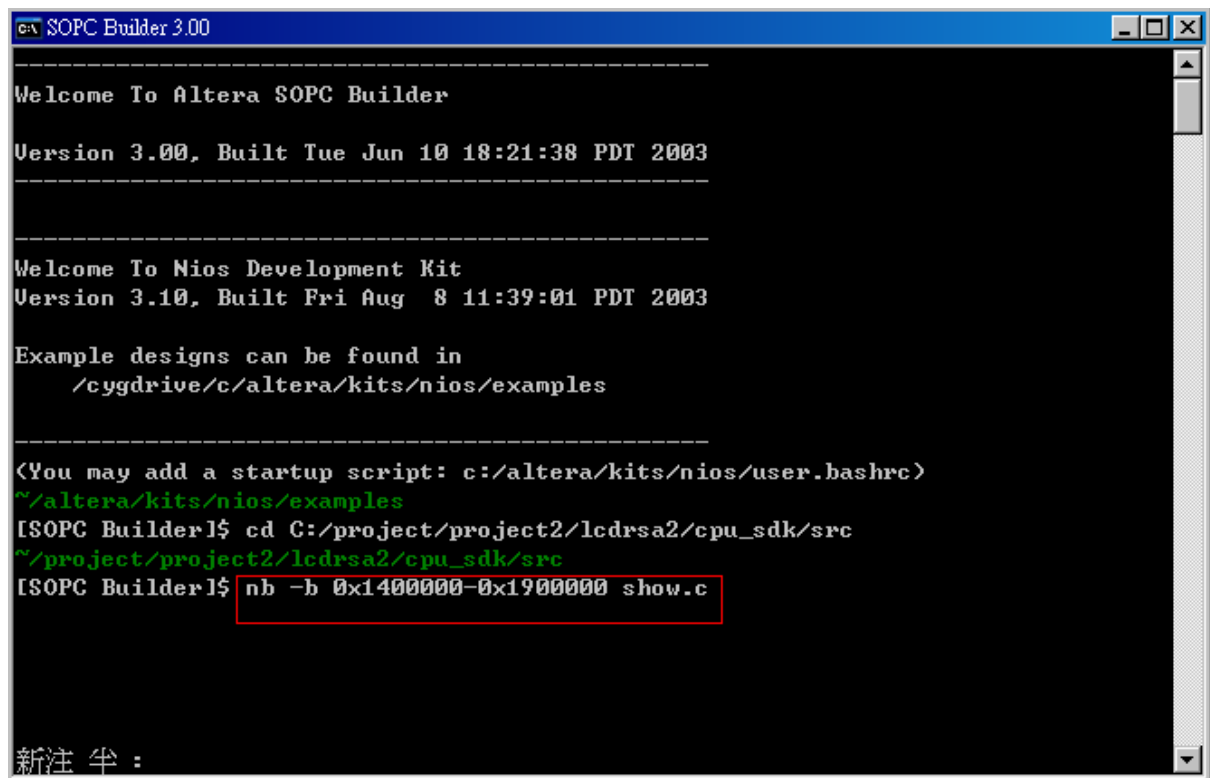


圖20 Nios system settings

設定program memory 為sdram，還有要注意的是，程式編譯的時候，要如圖21這樣設定，其中0x1400000-0x1900000為設定程式可用的記憶體範圍，如此程式才可以在sdram正常運作。(sdram的可用範圍為0x1000000-0x2000000)



```
-----
Welcome To Altera SOPC Builder
Version 3.00, Built Tue Jun 10 18:21:38 PDT 2003
-----

Welcome To Nios Development Kit
Version 3.10, Built Fri Aug 8 11:39:01 PDT 2003

Example designs can be found in
  /cygdrive/c/altera/kits/nios/examples
-----

<You may add a startup script: c:/altera/kits/nios/user.bashrc>
~/altera/kits/nios/examples
[SOPC Builder] $ cd C:/project/project2/lcdrsa2/cpu_sdk/src
~/project/project2/lcdrsa2/cpu_sdk/src
[SOPC Builder] $ nb -b 0x1400000-0x1900000 show.c

新注 半 :
```

圖21 Nios SDK Shell



## 第五章 軟體部分

### 5.1 程式簡介

主要是將JPEG中的資訊截取出來，如JPEG header的部分，還有就是JPEG解壓縮成BMP的資料，再將BMP的資料以不同方式寫入LCD，使LCD顯示出不同的效果。

還有要注意的是， $120 \times 160$ 的LCD是由圖22中左上角往右邊放資料，每一列放120像素的資料，這個跟我們程式如何放資料的方式有密切的關係。

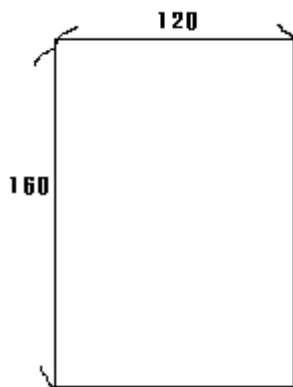


圖22 LCD示意圖

## 5.2 上下左右移動

設定一個基準座標，假設在這裡我們設定 $(X, Y) = (0, 0)$ ，並以 $(0, 0)$ 為基準，隨著按鍵的控制，我們給他做X加10，X減10，Y加10或Y減10，來算出新的基準座標，接著再和我們程式中截取出的長和寬的資訊做Mod的運算，長寬相對位置出來後，再和實際位置做運算，並將實際位置的資料內容寫入LCD，即可做圖片的上下左右運動。

所以經由Mod運算後，不管怎麼做加減，LCD上面一定都會有 $120 \times 160$ 像素的資訊，不會說一直往上移動時，圖片會不見。

## 5.3 放大縮小

我的放大演算法很簡單，如放大一倍就是將一個像素複製3份出來，如下圖23，至於兩倍，三倍等也是同樣的道理。

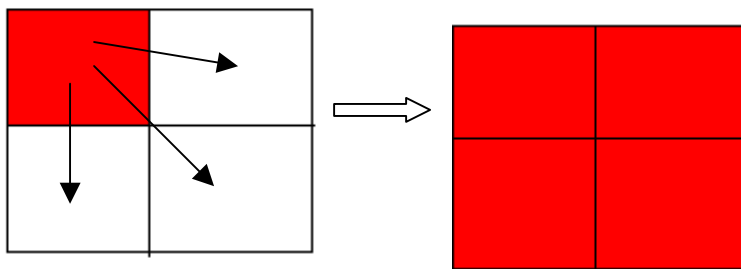


圖23 像素擴展

在程式上，我的想法是這樣的，以圖24為例，若要放大一倍，將下列四個座標除以2，這樣這四個座標就會都是 $(0, 0)$ 座標，所以當顯示這四個位置的像素，就全部會都是 $(0, 0)$ 像素的資訊，達到我們要擴展圖片的效果，所以要放大兩倍、還是三倍，都是同樣的道理，只要修改我們做運算的除數就可以了。

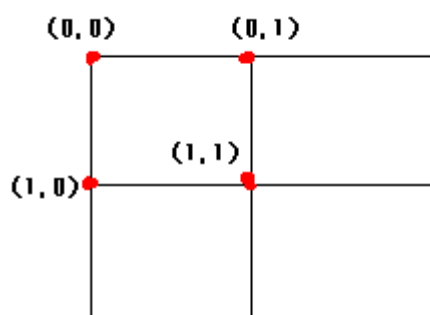


圖24 像素相對座標

縮小的方法，也是用很簡單的方法，將四個區塊的像素，縮減成一塊，若以圖24來說的話，就是保留(0,0)座標的資訊，(0,1)、(1,0)、(1,1)座標資訊一律捨去。

在程式的作法上，跟放大的方法類似，只是這時放大1倍時，將圖22的四個座標乘以2，變成(0,0)、(0,2)、(2,0)、(2,2)，這告訴我們，當LCD顯示圖片的時候，會略過(0,1)、(1,0)、(1,1)的資訊，而去顯示(0,2)、(2,0)、(2,2)的資訊，同理，放大兩倍、三倍等，就將座標乘以2、3等。

當然，在執行程式時，都是透過按按鍵來改變要座標要乘的數或要除的數，以達到用按按鍵時可以使得LCD上面有放大縮小圖片的效果。

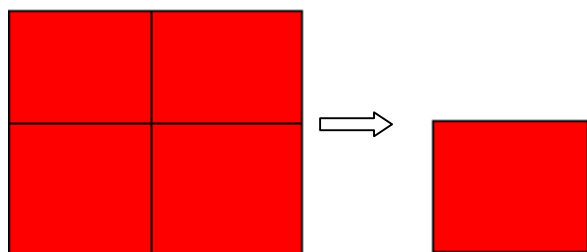


圖25 像素縮減

## 5.4 其它功能

剩下的功能是將彩色圖片轉成灰階以及換下一張圖片。

彩色圖片轉灰階：

取出由JPEG解出來BMP的資訊，然後再從每個像素中，將R、G、B的三種資訊分別取出來，做 $0.114B + 0.587G + 0.299R$ 運算後，所得的結果，即是我們要的灰階值，然後寫入LCD，即可以產生將圖片轉灰階的效果，但有個問題是做浮點運算時，速度會相當的慢，所以像乘以0.114，我在程式中，以 $116 \gg 10$ ，湊出接近0.114，使得運算速度上大為提升。

換下一張圖片：

塞入兩張圖片至記憶體，程式一開始執行只有解第一張圖片，當此功能按鍵壓下後，馬上呼叫解JPEG的函式，並將下一張圖片的起始位置當參數丟入，即可看到下一張的圖片。

## 第六章 心得感想

### 6.1 心得

當初看到學長JPEG的程式，完全搞不清楚狀況，後來跟學長借了有關JPEG的書去看，才漸漸進入狀況，對JPEG也有一定的了解，可以開始改程式，另外在硬體上，雖然改的不是很多，不過在過程中，也是滿有趣，雖然過程中，也鬧了很多笑話，如某根Pin腳沒接上，compiler半天都不會過，但總算也是吸收了一些經驗，不過說實話，我花在寫C的時間比較久，總是測來測去自己的C程式，如放大縮小的演算法，原本不是用這次專題寫的方法，之前想的方法寫的又長又笨，跑出來某些地方又有Bug，後來和同學討論，終於找到最好的方法了，有時寫程式換個角度去想還真的順了好多好多阿。

在這裡也很感謝張家維、楊勝吉學長，有問題時都可以提出不錯的意見，或幫忙解決，真的很感謝。另外也很感謝一起在實驗室的同學，有問題大家一起討論，常常獲益良多，謝謝大家。

### 6.2 未來展望

以下幾點是對於這次專題所做的延伸。

1. 由於這次專題是針對JPEG的基本壓縮模式，尚有漸進式壓縮模式、無損壓縮模式、階層壓縮模式這三種壓縮模式，可以再進一步了解這三種壓縮模式，或者除了JPEG以外的格式，如：TIF、PNG格式等，並解出來顯示在LCD上。
2. 用更大塊的LCD去顯示
3. 瞭解更多不同的演算法套用進去，如：圖片柔化、圖片亮度調整等。

## 附錄：成品照片



圖26 原圖

圖27 縮小



圖28 放大

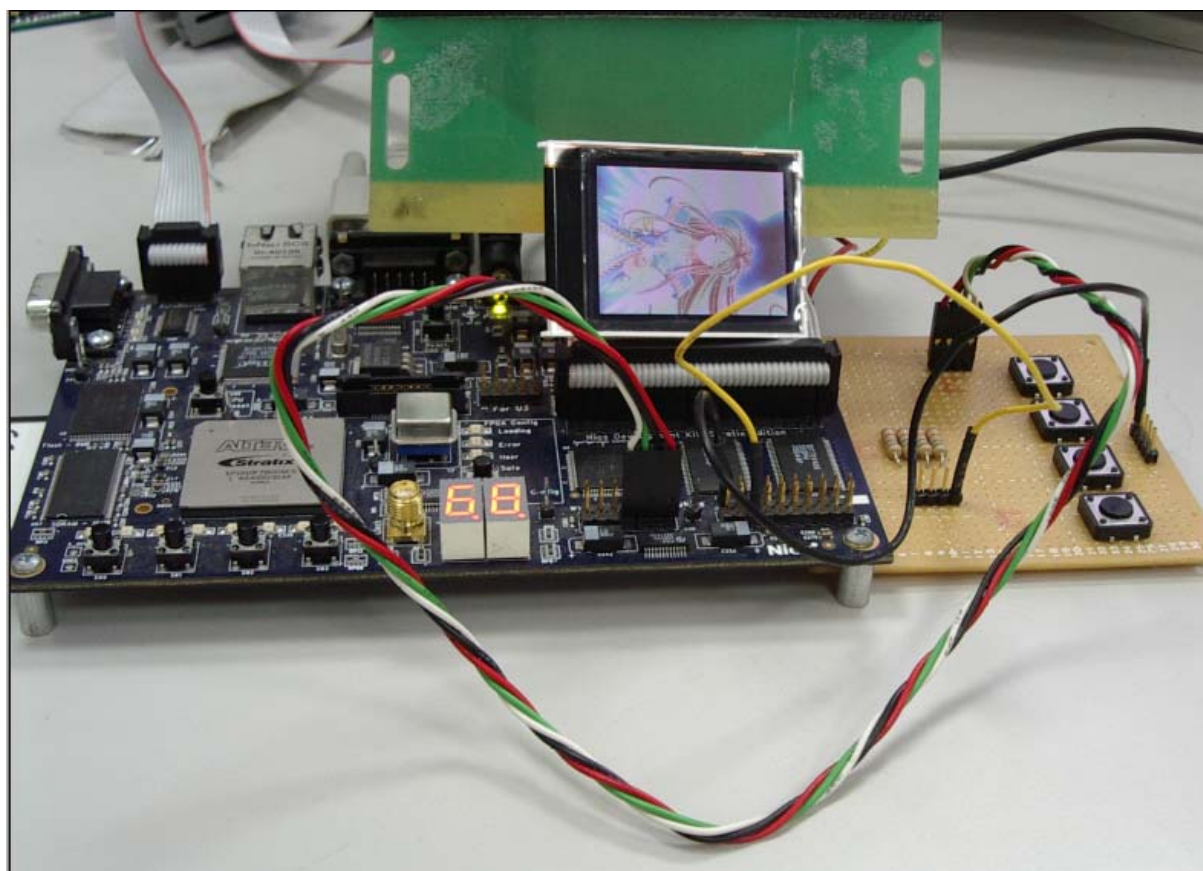


圖29 成品照片

## 參考資料：

- [1]數位影像處理 陳同孝 張真誠 黃國峰 編著
- [2] Altera Nios Tutorial & Nios Documentation
- [3]資料壓縮的原理與應用 鐘國亮 編著
- [4]PC影像處理技術(二)圖檔壓縮續篇 施威銘研究室 編著