國 立 交 通 大 學

電機與控制工程研究所

碩 士 論 文

以雙處理器核心之系統晶片

實現動態影像即時壓縮與解壓縮

Real-time Video Codec Implementation

Using a Dual-Core Processor

研 究 生： 周 春 成

指導教授： 胡 竹 生 博士

中 華 民 國 九 十 三 年 七 月

以雙處理器核心之系統晶片實現動態影像即時壓縮與解壓縮

Real-time Video Codec Implementation Using a Dual-Core Processor

研 究 生：周 春 成　　Student：Chun-Cheng, Chou

指導教授：胡 竹 生 博士　　Advisor：Prof. Jwu-Shen, Hu

國立交通大學
電機與控制工程學系
碩 士 論 文

A Thesis
Submitted to Institute of Electrical and Control Engineering
College of Electrical Engineering and Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of Master
in

Electrical and Control Engineering

July 2004

Hsinchu, Taiwan, Republic of China

中 華 民 國 九 十 三 年 七 月

博碩士論文授權書

(國科會科學技術資料中心版本，93.6.17)

本授權書所授權之論文為本人在 國立交通 大學 電機與控制工程 系所

＿＿＿＿＿＿ 組 九十二 學年度第 二 學期取得 碩士 學位之論文。

論文名稱： 以雙處理器核心之系統晶片實現動態影像即時壓縮與解壓縮

Real-time Video Codec Implementation Using a Dual-Core Processor

V同意 □不同意 (政府機關重製上網)

本人具有著作財產權之論文全文資料，授予行政院國家科學委員會科學技術資料中心、國家圖書館及本人畢業學校圖書館，得不限地域、時間與次數以微縮、光碟或數位化等各種方式重製後散布發行或上載網路。

本論文為本人向經濟部智慧財產局申請專利（未申請者本條款請不予理會）的附件之一，申請文號為：＿＿＿＿＿＿，註明文號者請將全文資料延後半年後再公開。

V同意 □不同意 (圖書館影印)

本人具有著作財產權之論文全文資料，授予教育部指定送繳之圖書館及本人畢業學校圖書館，為學術研究之目的以各種方法重製，或為上述目的再授權他人以各種方法重製，不限地域與時間，惟每人以一份為限。

上述授權內容均無須訂立讓與及授權契約書。依本授權之發行權為非專屬性發行權利。依本授權所為之收錄、重製、發行及學術研發利用均為無償。上述同意與不同意之欄位若未鈎選，本人同意視同授權。

指導教授姓名： 胡竹生 教授

研究生簽名： 周春成 學號：9112509

（親筆正楷） （務必填寫）

日期：民國 93 年 7 月 日

---

1. 本授權書 (得自 http://sticnet.stic.gov.tw/sticweb/html/theses/authorize.html 下載或至 http://www.stic.gov.tw/首頁右下方下載) 請以黑筆撰寫並影印裝訂於書名頁之次頁。
2. 授權第一項者，請確認學校是否代收，若無者，請個別再寄論文一本至台北市(106-36)和平東路二段 106 號 1702 室 國科會科學技術資料中心 黃善平小姐。
(電話:02-27377606 傳真：02-27377689)
3. 本授權書已民國 85 年 4 月 10 日送請內政部著作權委員會(現為經濟部智慧財產局)修正定稿，89.11.21 部份修正。
4. 本案依據教育部國家圖書館 85.4.19 台(85)圖編字第 712 號函辦理。

國 立 交 通 大 學

論 文 口 試 委 員 會 審 定 書

本　　校 電機與控制工程學系碩士班 周春成君

所 提 論 文 以雙處理器核心之系統晶片實現動態影像即時壓縮與解壓縮

Real-time Video Codec Implementation Using a Dual-Core Processor

合於碩士論文資格水準、業經本委員會評審認可。

口 試 委 員 ：

指 導 教 授 ：

系 主 任 ：　　教授

中華民國　九十三 年 七 月　 日

以雙處理器核心之系統晶片
實現動態影像即時壓縮與解壓縮

研究生：周 春 成　　指導教授：胡 竹 生 博士

國立交通大學電機與控制工程研究所碩士班

摘 要

在本篇論文以 OMAP 為實驗平台，並於其上研發一套適合用於雙核心架構處理器 - OMAP 的動態影像編解碼器。其中包含的 I-frame、P-frame、Intra-prediction、UMV 等壓縮演算法。透過雙核心架構處理器上的 RISC 處理器與 DSP 處理器的分工運算，讓整個系統效能得以提升。在 DSP 最佳化部分，透過 DSP 的影像加速器讓處理速度提升；在 RISC 方面，則以 Linux 作業系統為基礎，在其上發展應用程式，並透過 DSP Gateway 的達成處理器間的通訊。最後本論文採用 Software pipeline 的概念，在等待 DSP 完成的時間插入 RISC 程式片斷，使整個系統的效能得以提升。本論文並實際量測所實作的動態影像編解碼器的效能表現，透過各種最佳化方法，可達每秒 7.6 張的效能。

Real-time Video Codec Implementation Using a Dual-Core Processor

Student： Chun-Cheng, Chou Advisor： Prof. Jwu-Shen, Hu

Institute of Electrical and Control Engineering
National Chiao-Tung Univeristy

ABSTRACT

The objective of this thesis is to implement a video codec using the Dual-Core processor – OMAP. The coding system includes I-frame, P-frame, Intra-prediction, UMV and other coding elements. These tasks are distributed to the two processors for efficiency in implementation. On DSP, we use the image hardware extension in it to improve computational speed. On RISC, we use Linux as the embedded operation system and development environment. And we use DSP Gateway to make Inter-processor communication possible. To improve the overall system efficiency, the software pipelining concept is implemented among processors. Finally, the system performance of 7.6 frames/sec can be achieved.

目錄

摘　要 i
ABSTRACT ii
目錄 iii
圖列 v
表列 vii
第一章 緒論 1
1.1. 研究動機 1
1.2. 研究內容與目標 1
1.3. 論文內容概述 2
第二章 動態影像壓縮 3
2.1 動態影像壓縮導論 3
2.2 色彩空間 4
2.3 動態影像壓縮解壓縮演算法 6
2.3.1 DPCM/DCT 7
2.3.1.1 動態壓縮影像格式 8
2.3.1.2 離散餘弦轉換（Discrete Cosine Transform ,DCT） 8
2.3.1.3 Quantization & De-Quantization 9
2.3.1.4 Zigzag scan order 10
2.3.1.5 可變長度編碼 (Variable Length Coding, VLC) 11
2.3.1.6 移動預估與移動補償 11
2.3.2 無限制移動向量(Unrestricted Motion Vector, UMV) 14
2.3.3 Intra-prediction 15
2.3.4 Four Motion Vector(4MV) 16
第三章 軟硬體平台簡介 18
3.1 硬體環境 18
3.1.1 OMAP 處理器 18
3.1.2 PSI OMAP Innovator 20
3.2 軟體環境 21
3.2.1 VMware 21
3.2.2 Linux 作業系統 23
3.2.3 ARM Processor Compiler 23
3.2.4 DSP Gateway 24
3.2.5 Code Composer Studio (CCS) 27
3.3 開發環境與程式開發流程 28
3.4 實驗與測試環境 29

第四章 實作與最佳化 31
4.1 系統架構 31
4.2 編碼器實作 32
4.2.1 編碼器架構 32
4.2.2 影像擷取與 LCD 顯示 33
4.2.3 ARM 與 DSP 之間的資料傳輸 35
4.2.4 DSP Encoder Task 36
4.2.4.1 量化(Quantization) 36
4.2.4.2 VLC 37
4.2.4.3 Intra-prediction 38
4.2.4.4 UMV 39
4.2.5 DSP Monitor Task 41
4.3 解碼器實作 42
4.3.1 解碼器架構 43
4.3.1.1 Inverse Intra-prediction 43
4.3.1.2 VLD 43
4.3.2 DSP Display Task 44
4.3.3 DSP 解碼器優先權設定 45
4.4 最佳化 46
4.4.1 DSP 程式最佳化 46
4.4.2 ARM 程式最佳化 48
第五章 結論與未來展望 52
5.1 結論 52
5.2 未來展望 52
參考資料 53

圖目錄

圖 2-1 YCbCr 取樣格式 5
圖 2-2 Marco-block 影像做 YCbCr420 取樣 6
圖 2-3 DPCM/DCT 編碼器架構 7
圖 2-4 DPCM/DCT 解碼器架構 7
圖 2-5 zigzag scan order 11
圖 2-6 3-step Search 12
圖 2-7 Nearest search Algorithm 13
圖 2-8 2D autocorrelation of Image 14
圖 2-9 2D autocorrelation of Residual 14
圖 2-10 UMV 概念圖 15
圖 2-11 Intra-prediction 15
圖 2-12 DCT 係數(亮度) 16
圖 2-13 4MV 17
圖 3-1 OMAP1510 Internal 18
圖 3-2 PSI Innovator 發展平台 21
圖 3-3 VMware 執行 Red Hat Linux 22
圖 3-4 DSP Gateway Mailbox 25
圖 3-5 DSP gateway 25
圖 3-6 DSP Memroy Space 27
圖 3-7 CCS 整合發展環境介面 28
圖 3-8 發展平台架構 29
圖 3-9 測試與實驗平台 30
圖 4-1 系統架構 31
圖 4-2 編碼器程式架構 32
圖 4-3 V4L2 使用流程 34
圖 4-4 Frame buffer 裝置使用流程 35
圖 4-5 Encoder 記憶體配置 36
圖 4-6 VLC 流程圖 38
圖 4-7 VLC ESC CODE 38
圖 4-8 intra-prediction 計算順序 39
圖 4-9 預估影像(QCIF - predict frame) 40
圖 4-10 剩餘影像(QCIF - residual frame) 40
圖 4-11 原始影像的自相關矩陣係數 40
圖 4-12 剩餘影像的自相關矩陣係數 41

圖 4-13 Monitor Task 與 Encoder/Decoder Task 的關係 42
圖 4-14 解碼器程式架構 43
圖 4-15 VLD 流程圖 44
圖 4-16 DSP Display Task 45
圖 4-17 Priority for DSP Tasks 45
圖 4-18 原始程式執行架構 49
圖 4-19 相鄰的迴圈展開 49
圖 4-20 Software pipeline - 1 50
圖 4-21 Software Pipeline – 2 50

表目錄

表 2-1 I-frame 量化表 10
表 2-2 P-frame 量化表 10
表 3-1 PSI Innovator Hardware List 21
表 4-1 未最佳化執行時間 46
表 4-2 Benchmark of IMGLIB 47
表 4-3 局部最佳化後的結果 47
表 4-4 Software pipeline 後的執行時間 51

第一章 緒論

1.1. 研究動機

多媒體通訊在日常生活中佔的角色越來越重要，但是由於影音資料，尤其是影像，資料量都相當的龐大。尤其是在手持式裝置上，想要在有限的通訊頻寬的限制上，達到即時影音的需求是相當的具有難度的。不過由訊號壓縮理論的進步、一些壓縮標準的制定與通訊頻寬的進步，讓即時多媒體通訊不再難以實現。

動態影像壓縮之所以可以大量減低資料量，其原因在於它移除了影像與影像之間的相關性，但也因此造成龐大的資料運算。這樣龐大的資料運算，若使用一般的手持式裝置會由於運算能力的限制而難以達成。

OMAP 處理器包含 RISC(ARM)與 DSP 雙核心處理器，透過這兩種類型的處理器的合作，讓一些耗時間的演算法得以在手持式裝置上快速執行。在 OMAP 的 ARM 上所執行的作業系統讓更多類型的程式得以執行，手持式裝置不再單調；DSP 則可以輔助 ARM 讓耗時間的演算法得以實現。

1.2. 研究內容與目標

在架設實驗平台方面，首先將針對 OMAP 處理器修補後 Linux 核心安裝到平台上，並建立檔案系統，其次透過 DSP Gateway 來達成 ARM 與 DSP 間的通訊。在壓縮演算法方面，研究常用的動態影像壓縮演算法，並在實驗平台上實現這些演算法。最後，將實現的動態影像壓縮透過一些最佳化與雙核心處理器的分工，提昇影像壓縮時系統的效能表現。

本論文主要目的是利用 OMAP 上 ARM 與 DSP 的分工方式，時做資料量大、耗費時間的動態影像壓縮演算法實現，並透過一些最佳化的方法將系統效能提升。

1.3. 論文內容概述

本論文的組織架構如下：

第二章將簡介動態影像壓縮與演算法。

第三章將簡介硬體平台 OMAP Innovator 與 OMAP 處理器；在軟體開發環境會簡介 OMAP 上的作業系統、開發工具與個人電腦上的開發環境，並簡述開發流程與軟硬體平台之間的關係，與測試環境。

第四章將描述如何在 OMAP 平台上開發出編碼器與解碼器，並且讓編解碼器可以透過 OMAP 上的 RISC 與 DSP 的合作，將編碼器與解碼器的速度提升。

第五章對本系統做出一個總結，並提出對本系統的未來展望。

第二章 動態影像壓縮

2.1 動態影像壓縮導論

隨著網路的頻寬逐漸增加，與寬頻網路的普及，視訊影音內容也愈來愈流行，相對地，帶動影像壓縮技術的需求。其中以動態影像壓縮為主流技術。動態影像壓縮的最大的優點有兩個，第一，在無法使用原始資料的情況下，讓數位影像的傳輸或儲存成為可能。舉例而言，當網路的速度無法即時的處理原始影像資訊時，若不採用壓縮技術，即使 DVD 作為儲存媒介，也只能儲存不到幾秒的原始影像。第二個優點是動態影像壓縮可以有效的利用頻寬與儲存的空間；假設有一個高頻寬的通道可以用，傳送高解晰度的壓縮視訊或是多個頻道壓縮視訊比起傳送一個低解析度的原始視訊還要具有吸引力。因此雖然網路的頻寬和儲存媒介的大小持續的增加，壓縮還是多媒體服務不可或缺的部份。

一般而言，壓縮藉由移除訊號的重複性高的部份，可得到壓縮的效果。在無失真壓縮系統中，原始訊號中的多餘度被移除，並在接收端被完美還原。不幸的是，無失真壓縮對影像與聲音訊號造成的效果有限。大部分的動態影像壓縮是以失真壓縮為基礎，使用失真壓縮可以達到高壓縮比，但其缺點就是還原後的資訊和原來的不完全相同。動態影像壓縮演算法的目的就是在於如何同時達成有效壓縮並降低在壓縮過程中所造成的失真。

動態影像壓縮演算法是以移除時間、空間或是頻域的多餘度。某些影像，例如白色的牆壁，影像的變化很小，此即為空間的多餘度。若是將影像經過一個低頻濾波器，將高頻的部份濾除。由於人類的視覺對低頻比較敏感，對高頻較不敏感，所以雖然濾除了很多高頻資訊，但是人類的視覺系挺還是可以辨識出影像的內容。連續兩張影像，時間間隔只有 1/25 秒，比較兩張影像後，如果影像變化差異小，即可稱此情況為時間的多餘度。換句話說，在連續的影像之間，大部分的影像內容是沒有改變的。在移除

這些多餘度後，可以讓壓縮比更高，但是也造成影像失真愈嚴重。如果進一步使用 Huffman 或是 Arithmetic 編碼，則可達成更高的壓縮比。

2.2 色彩空間

由於目前數位影像的應用大多是彩色的，所以需要一套用來表示色彩資訊的機制。在灰階影像方面，只需要使用亮度來表示空間上的每一像素。但是彩色影像每個像素卻必須用三個值來表示 RGB。所以用來表示亮度與色彩的方法稱為「Color Space」。

- RGB

在 RGB 的 Color Space 中，用來表示一個像素的三個值是 R(紅色)、G(綠色)與 B(藍色)。任何的色彩都可以使用不同比例的紅綠藍組成。RGB Color Space 是非常適合用於影像擷取與顯示。在擷取彩色影像時，個別使用濾光鏡將三種不同的組成色，用不同的感應器陣列擷取。目前 CRT(陰極射線管螢幕)與 LCD （液晶螢幕）都是使用 RGB Color space 來顯示彩色資訊。

- YCbCr

人類的視覺系統對亮度比對顏色敏感。在 RGB 中的三種顏色是同等重要，所以使用相同的大小儲存資訊，但是更有效的方法是將亮度和色差分離，使用較大的解析度來儲存亮度，而使用較小的解析度來儲存色差。

YCbCr(也較 YUV)是很普遍而有效率的表示彩色影像的方法，Y 代表的是亮度，可以用 RGB 的加權平均計算出來。顏色的資訊被表示為色差，這些色差的資訊由 RGB 和 Y 相減而得。RGB 與 YCbCr 可以使用以下的公式互相轉換。

$$Y = k_r + (1 - k_b - k_r)G + k_b B$$

$$Cb = \frac{0.5}{1 - k_b}(B - Y) \qquad Cr = \frac{0.5}{1 - k_r}(R - Y) \tag{2.1}$$

$$R = Y + \frac{1-k_r}{0.5}Cr$$
$$G = Y - \frac{2k_b(1-k_b)}{1-k_b-k_r}Cb - \frac{2k_r(1-k_r)}{1-k_b-k_r}Cr \qquad (2.2)$$
$$B = Y + \frac{1-k_b}{0.5}Cb$$

ITU-R BT.601[08]的建議值為 $k_b = 0.114$、$k_r = 0.299$。

- YCbCr 取樣格式

最常見的 YCbCr 格式有三種，為 YCbCr444、YCbCr422、YCbCr420。這些格式的共同點是亮度全部都保留下來，只改變色差。由於前面說過，人類視覺對亮度較敏感，犧牲色差的精確度對人類的視覺系統而言，對影像品質並不會產太大的影響。下圖表示的就是三種不同的取樣格式。

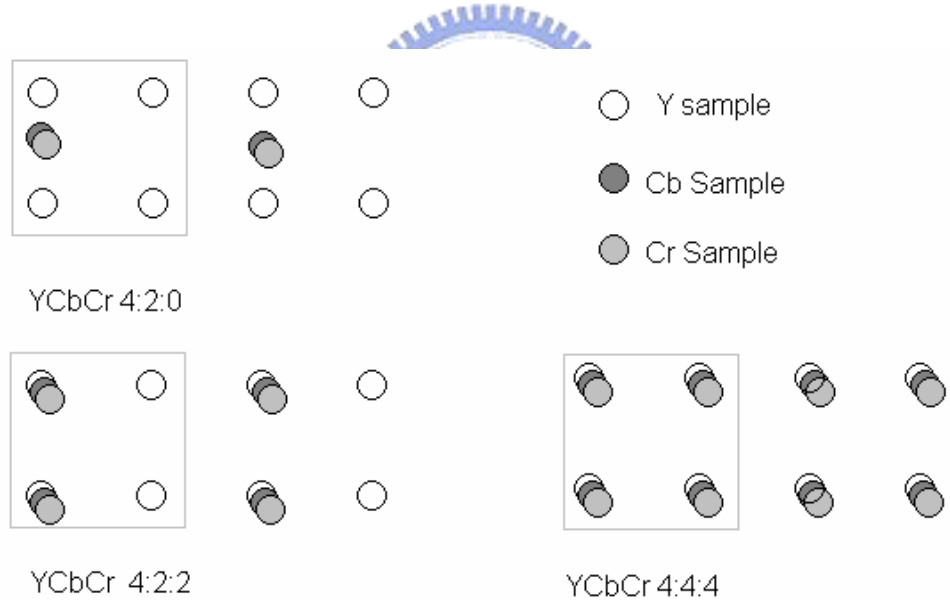

圖 二-1　YCbCr 取樣格式

YCbCr444 是指 Y、Cb、Cr 三者的解析度相同，並沒有在取樣中有資料耗損的情形；YCbCr422 如上圖所示，在選取的四個像素中對只第一行的兩個像素取色差，若是原始影像 RGB 各佔 8bit，經過 YCbCr422 取樣後 Y 為 8bit，但是 Cb 與 Cr 平均各只剩 4bit，平均總長度由 24bit 變成 16bit，減少了 33%的資訊量。若是經過 YCbCr 取樣，

則 Cb 與 Cr 平均各只剩 2bit，平均總長度由 24bit 變成 12bit，減少了 50%的資訊量，若一個 marco-block 的影像處理下來，會產生一張相同大小的亮度影像，和兩張 1/4 大小的色差影像。

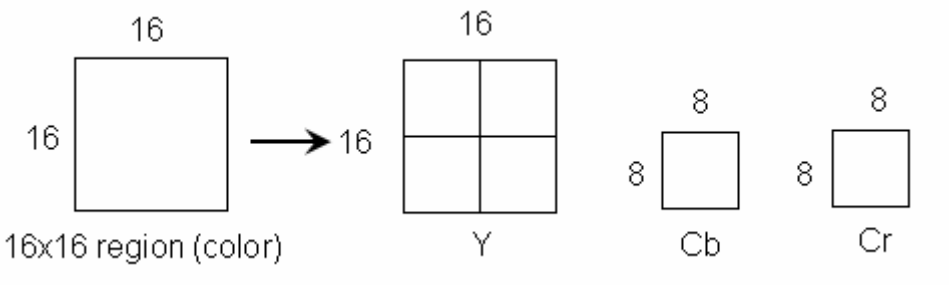

圖 二-2　Marco-block 影像做 YCbCr420 取樣

2.3 動態影像壓縮解壓縮演算法

動態影像壓縮通常利用以下三種移除多餘度(redundancy)的概念來達成壓縮的目的。

- 時間模型(Temporal Model)

  在時域上，通常在相近的時間所擷取的影像有非常高的相關性，尤其是當取樣頻域很高時更是明顯。所以這裡的目標在於降低時域上相鄰的影像的多餘度，用的方法是先產生預估影像後再與目前的影像相減。越精確的預估，相減後剩餘的影像能量就越小。預估的影像將由一到多張的以前或以後的影像所產生。傳送到解碼器的影像將會是編碼後的剩餘影像。

- 空間模型(Spatial Model)

  在空間域上，像素之間的相關性也是相當高的，也就是相鄰的像素值會很接近。經過轉換函數後，相關性高的，也就是像素值接近的，將會集中在低頻；經過適當的量化後將會使大多數的高頻訊號量化為 0，由於人類視覺對低頻的敏感度較

高，被捨去的細節會被人類視覺系統忽略，從而達成降低儲存空間或是傳輸頻寬的目的。

- 統計模型(Statistical Model)

利用統計的方法將出現機率高的符號(symbol)使用較少的位元表示，而較不常出現的使用較多的位元表示，如此達到降低資訊量的效果，而在資料儲存或傳輸頻寬上獲得好處。

2.3.1 DPCM/DCT

DPCM/DCT 是目前最普遍採用的影像壓縮架構，重 JPEG 到 MPEG4 或是 H.264 都可以看到它的蹤影，它使用前面所提到的三種概念來實現壓縮的目的，Motion Estimation 與 Motion Compensation 是屬於時間模型，DCT 與 Quantization 是屬於空間模型，而 Entropy encoder 是屬於統計模型。DPCM/DCT 的架構如下圖(圖 2-1/2)

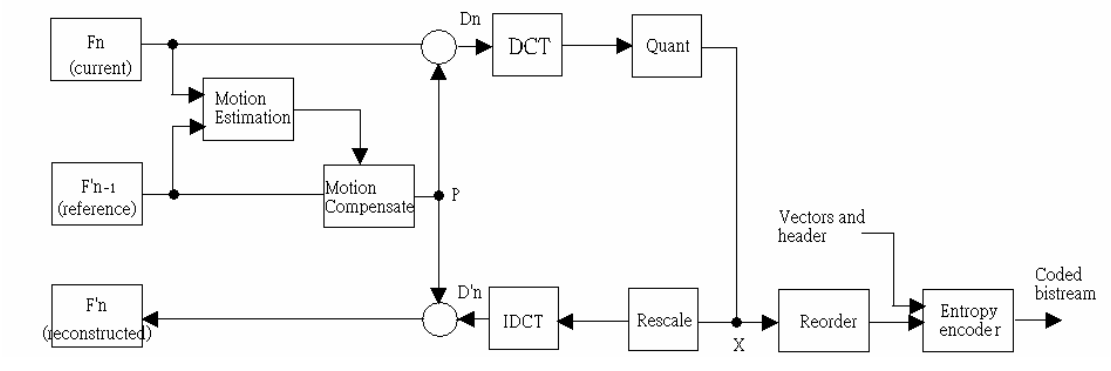

圖 二-3 DPCM/DCT 編碼器架構

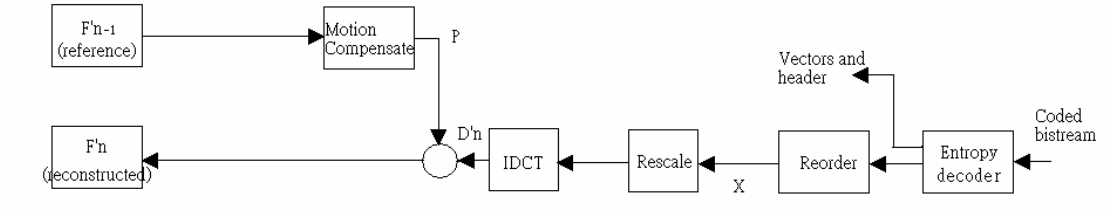

圖 二-4 DPCM/DCT 解碼器架構

2.3.1.1 動態壓縮影像格式

在 MPEG 的規格中包含三種不同的影像格式，分別為 intra-frame、inter-frame 與 bidirectional-frame。

Intra-frame 又稱 I frame，它是由整張影不經過預估編碼直接經過 DCT 與量化後的資訊，它解碼後會被當成第一張參考影像。

Inter-frame 又稱 P frame，P 的意思是代表 Predictive，也就是說它必須經過預估編碼的流程，預估的影像來自於前一張被解開的 I frame 或是 P frame。

Bidirectional-frame 又稱 B frame，它和 P frame 不同的地方在於預估的影像來可能來自於時域上前一張與後一張的影像，由於是雙向預估，比較不適合做在及時的編碼系統。

2.3.1.2 離散餘弦轉換（Discrete Cosine Transform ,DCT）

DCT 是一種轉換編碼，它可以將一組強度資料轉換成時域資料，它是一種無失真的轉換，以下是它的正轉換(Forward DCT)與反轉換(Inverse DCT)。

- Forward DCT

正轉換(Forward DCT)的轉換公式如下

$$Y_{xy} = C_x C_y \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} X_{ij} \cos\frac{(2j+1)y\pi}{2N} \cos\frac{(2i+1)x\pi}{2N}$$
$$C_i = \sqrt{\frac{1}{N}}(i=0), \qquad C_i = \sqrt{\frac{1}{N}}(i>0) \tag{2.3}$$

- Inverse DCT

反轉換(Inverse DCT)的轉換公式如下

$$X_{ij} = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} C_x C_y Y_{xy} \cos\frac{(2j+1)y\pi}{2N} \cos\frac{(2i+1)x\pi}{2N}$$
$$C_i = \sqrt{\frac{1}{N}}(i=0), \qquad C_i = \sqrt{\frac{1}{N}}(i>0) \tag{2.4}$$

2.3.1.3 Quantization & De-Quantization

用 Quantization 可以減少儲存空間的需求、減少所需的頻寬與達成較佳的壓縮比。Quantization 後的資料使用較少的位元數來儲存資料以達成壓縮的目的，被捨去的細節會被人類視覺所忽略，所以 Quantization 是壓縮流程裡一個相當重要的步驟。

- Quantization

當輸入資料為 I-frame 的 AC 值或是 P-frame 時，使用下式來量化輸入的資料

$$QDCT(i,j) = \frac{\left[\frac{32 \times DCT(i,j)}{Qmatrix(i,j) \times Quantizer\ Scale} + k\right]}{2}$$

$$\text{Where } k = 0 \text{ for intra blocks}$$

$$k = \text{sign of DCT coefficient for non-intra blocks}$$

$$\begin{cases} 1 & \text{if } DCT(i,j) > 0 \\ 0 & \text{if } DCT(i,j) == 0 \\ -1 & \text{if } DCT(i,j) < 0 \end{cases} \qquad (2.5)$$

當輸入資料為 I-frame 的 DC 值時，使用下式來量化輸入資料

$$Quantized\ Coefficient(0,0) = \frac{DCT(0,0)}{k} \qquad (2.6)$$

$$k = 8 \ for\ 8bit\ precision$$

Qmatrix 的定義如下，第一個量化矩陣(表 二-1)是給 I-frame AC 係數使用的 Qmatrix，第二個量化矩陣(表 二-2) 是給 P-frame DCT 係數使用的 Qmatrix，P-frame 使用的 Qmatrix 是屬於 Uniform Quantizer

表 二-1 I-frame 量化表

$$\begin{bmatrix} 8 & 16 & 19 & 22 & 26 & 27 & 29 & 34 \\ 16 & 16 & 22 & 24 & 27 & 29 & 34 & 37 \\ 19 & 22 & 26 & 27 & 29 & 34 & 34 & 38 \\ 22 & 22 & 26 & 27 & 29 & 34 & 37 & 40 \\ 22 & 26 & 27 & 29 & 32 & 35 & 40 & 48 \\ 26 & 27 & 29 & 32 & 35 & 40 & 48 & 58 \\ 26 & 27 & 29 & 34 & 38 & 46 & 56 & 69 \\ 27 & 29 & 35 & 38 & 46 & 56 & 69 & 83 \end{bmatrix}$$

表 二-2 P-frame 量化表

$$\begin{bmatrix} 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \end{bmatrix}$$

- De-Quantization

當作反向量化時，依下式將被量化的 DCT 係數回復

$$DCT = \frac{(2 \times QDCT(i,j) + k) \times Qmatrix(i,j) \times quantiser\ scale}{32}$$

Where k = 0 for intra blocks

k = sign of quantized coefficient for non - intra blocks (2 . 7)

(1 if quantized(i, j) > 0, 0 if quantized(i, j) == 0,

-1 if quantized(i, j) < 0)

2.3.1.4 Zigzag scan order

由於影像經過 DCT 後，DCT 係數的能量大部分集中在低頻，也就是左上角的部

份，且量化後的高頻係數，也就是右下角的部份大多為 0，因此若是使用一般掃描線由左而右、由上而下的順序，這些等於 0 的 DCT 係數將無法連續，使 Run-Length Coding 的長度增加。若是使用 zigzag 掃描順序(如圖 2-5)，由於 DCT 係數的特性，可以減少 Run-Length Coding 的長度。

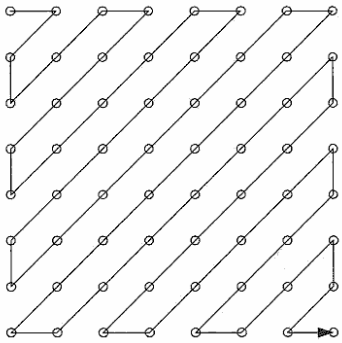

圖 二-5 zigzag scan order

2.3.1.5 可變長度編碼 (Variable Length Coding, VLC)

雖然前述的方法可以減少影像的多餘度，但是真正減少資訊量的就是 Entropy Coding。VLC 是一種 Entropy Coder，較常用的 VLC 有兩種 - Huffman Coding 與 Arithmetic coding。

2.3.1.6 移動預估與移動補償

- 移動預估(Motion Estimation, ME)

  移動預估已經被證明可以有效解決連續影像序列時間上的重複性，因此 ME 已經整合在很多現存的動態影像壓縮的標準如 MPEG-1，MPEG-2，MPEG-4，H.261 與 H.263。

- 區塊比對演算法(Block-Matching Algorithm, BMA)

區塊比對演算法最常被使用在 block-base 的壓縮演算法上，它在前一個影像的一個區間內來尋找和目前這個影像其中一個區塊最相似的區塊(稱為 best match)。Full Search(FS)演算法是搜尋區間內每一個區塊，以找出最佳的相似的區塊。但是 Full Search 相當的耗費電腦的運算量。為了解決這個問題，有相當多的快速 BMA 發展出來，例如 2-D logarithmic search(LOGS)，three-step search(TSS)，new three-step search(NTSS)，four-step search(FSS)，new diamond search(DS)，block-based gradient descent search(BBGDS)和其他演算法。這些快速演算法使用不同的搜尋樣本與搜尋策略去尋找移動向量(Motion vector)。它們和 FS 相較起來使用較少的搜尋點。

✧ N-step search Algorithm

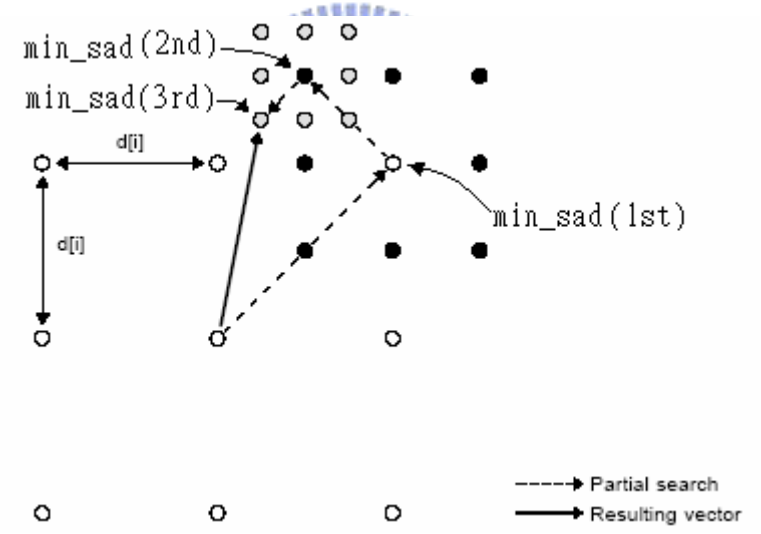

圖 二-6 3-step Search

N-step search 是一種使用有限次搜尋，找出移動向量的方法。以 3-step search 為例，這個演算法會先搜尋 9 個等距離的點(如上圖空心圓的部份)中最小的 SAD(Sum Absolute Difference)，再以此點為中心，將距離縮短為原來的一半，再重覆一次流程，最後將會找到在參考影像中與目前最像的區塊，以此求得移動向量。這個方法的好處是架構簡單，但是缺點是會落入局部最佳值中，無法找到整體最佳值。

✧ Nearest search Algorithm

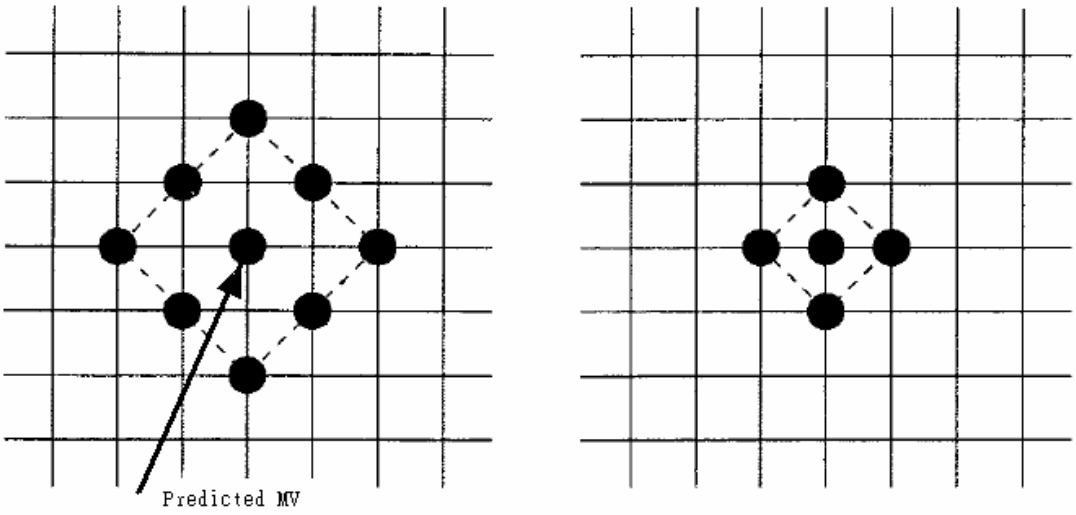

圖 二-7 Nearest search Algorithm

Nearest search algorithm 又稱 Diamond Search，首先必須先將使用鄰近或是前一張相對位置鄰近區塊的移動向量來產生預估向量，再依上圖的方法搜尋最佳的移動向量。上圖的方法分為兩種，第一種為大區域搜尋(LDSP 如上圖(a))，以移動向量所指到的像素為中心，比較附近的九個點，假設最佳的位置在中間，就轉為使用小區域搜尋(SDSP 如上圖(b))：若最佳的位置在外側，則以最佳位置為中心，再使用大區域搜尋，直到最佳位置落入中間。這樣的演算法有助於由局部最佳值跳脫出，但是缺點是架構比較複雜。

➢ 移動補償(Motion Compensation)

在使用參考影像與目前輸入影像利用 BMA 算出移動向量之後，再利用這些移動向量加上參考影像產生預估影像，再利用產生的預估影像與目前輸入的影像相減，產生剩餘影像(residual frame)的過程稱為移動補償。

移動補償的目的在於縮時域上的多餘度移除，讓影像的自相關性與能量變小(圖 2-8/9)，以達成減低資訊量的目的。

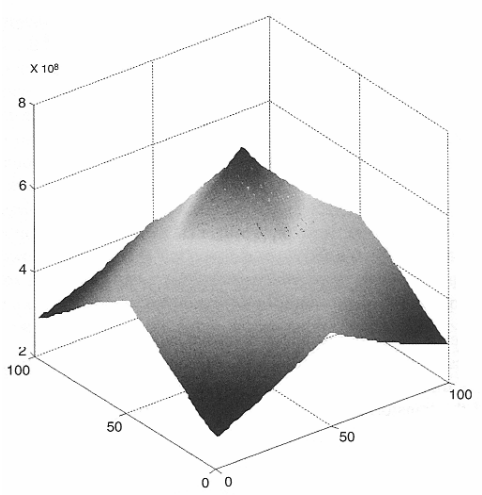

圖 二-8　2D autocorrelation of Image

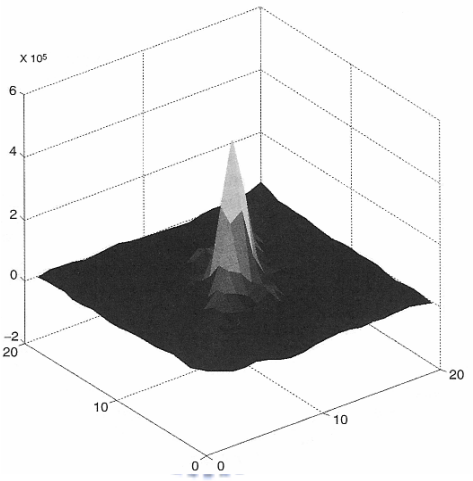

圖 二-9　2D autocorrelation of Residual

2.3.2　無限制移動向量(Unrestricted Motion Vector, UMV)

在某些情況下，16x16 的最佳預估區塊可能會超出參考影像的邊界，如下圖(圖 2-10)左下角的預估區塊超出的參考影像的邊界，若只有在目前的參考影像內尋找，將無法得到較好的預估區塊，所以這個方法如下圖最右邊所示，將邊界延伸出去讓最佳的預估區塊可以超出原來的參考影像邊界，以得到較佳的移動向量。簡單的說，UMV 就是允許移動向量可以超出參考影像的邊界。使用 UMV 可以提升移動補償的效率，特別是針對影像中有物件要移出影像外的情形。

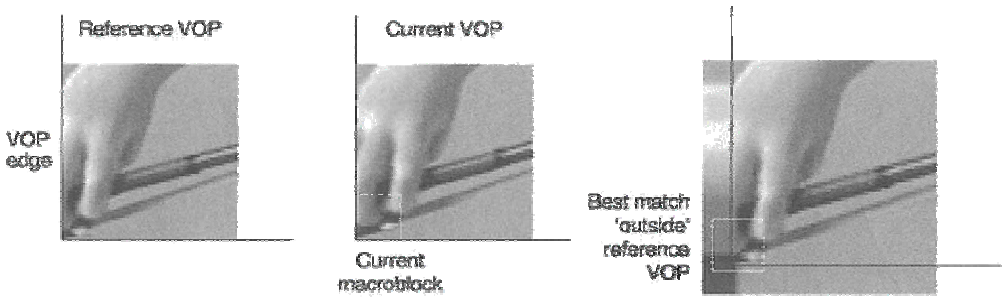

圖 二-10　UMV 概念圖

2.3.3　Intra-prediction

Intra-prediction 使用的原理是由於作完頻域轉換(DCT)後，通常相鄰的區塊低頻的部份轉換係數相關性很高，若是可以使用相鄰區塊來預估目前區塊的低頻轉換係數，將可以有效減少資料量。

Intra-prediction 使用的演算法如下

$$\begin{aligned} &if\ |DC_A - DC_B| < |DC_B - DC_C| \\ &\qquad predict\ \ from\ block\ C \\ &else \\ &\qquad predict\ \ from\ block\ A \end{aligned} \tag{2.8}$$

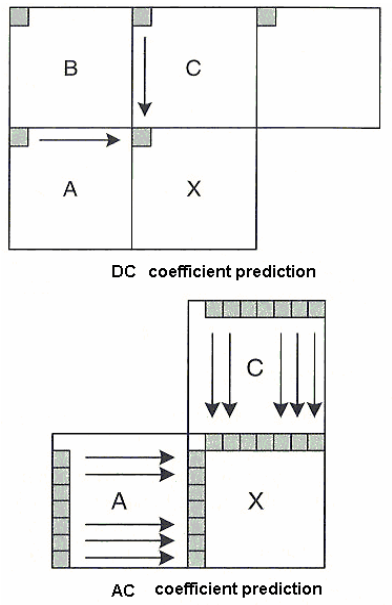

圖 二-11　Intra-prediction

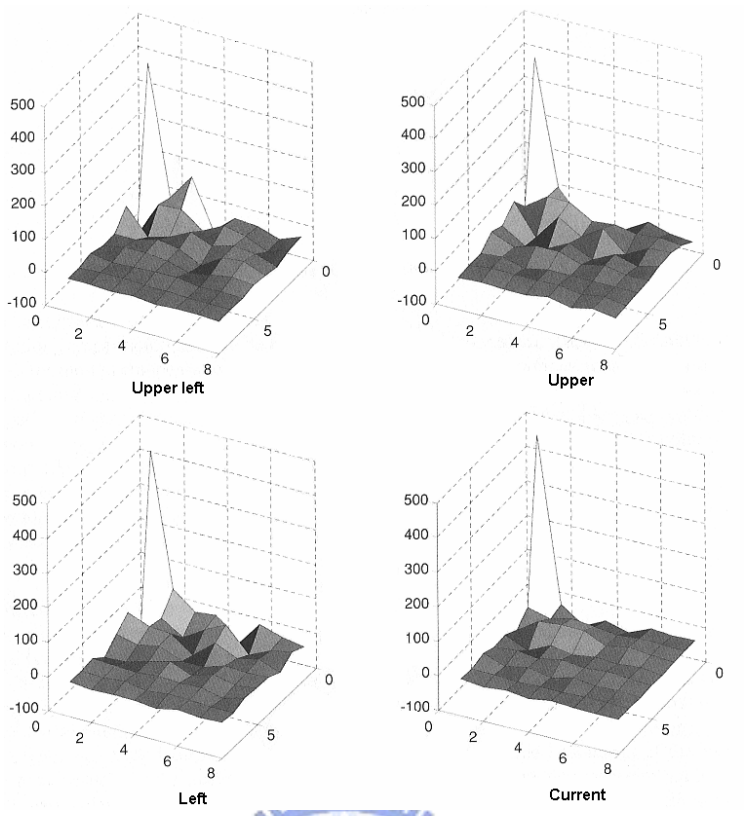

圖 二-12 DCT 係數(亮度)

上圖顯示相鄰的 DCT 係數的值，以 DC 值為例，若要預估目前的 DC 係數，使用左側的 DC 係數來預估比使用上側的 DC 係數還要好，如此剛好是$|DC_{upper\text{-}left} - DC_{left}|>|DC_{upper\text{-}left} - DC_{upper}|$；換句話說，intra-prediction 是使用讓 DC 值變化量最大的方向的 DCT 係數來預估目前的 DCT 係數，因為預估的區塊的方向和最大變化量正交，因此選擇的預估區塊的 DCT 係數會比較接近目前區塊。

在預估 AC 值時也適用一樣的法則，但是必須注意的是 AC 係數只有預估靠近預估區塊那一側的 AC 值(如圖 2-11)。

2.3.4 Four Motion Vector(4MV)

移動補償的效率會隨的區塊變小而提升效能。若使用一 Marco-block 一個移動向量，那麼區塊大小亮度為 16x16，色差為 8x8。4MV 即為 4 Motion Vector，就是在一 Marco-block 使用四個移動向量來表示移動量，下圖(圖 2-13)即為示意圖。使用 4MV

的好處在於可以將 residual frame 的能量降到最小，也就是說預估出來的影像會很接近目前輸入的影像，這樣有助於提升移動補償的效率，增加壓縮比，進而將所需的頻寬降低。

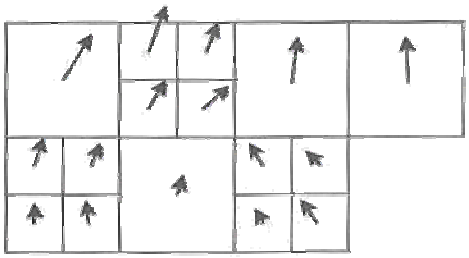

圖 二-13　4MV

第三章 軟硬體平台簡介

本章將介紹開發動態影像編解碼器的硬體環境與軟體環境，以及程式設計流程。在硬體環境將介紹使用的平台 PSI Innovator 與 OMAP 處理器；軟體方面將介紹實驗平台上的系統與程式寫作平台。

3.1 硬體環境

本節將簡單介紹實驗平台的硬體規格。

3.1.1 OMAP 處理器

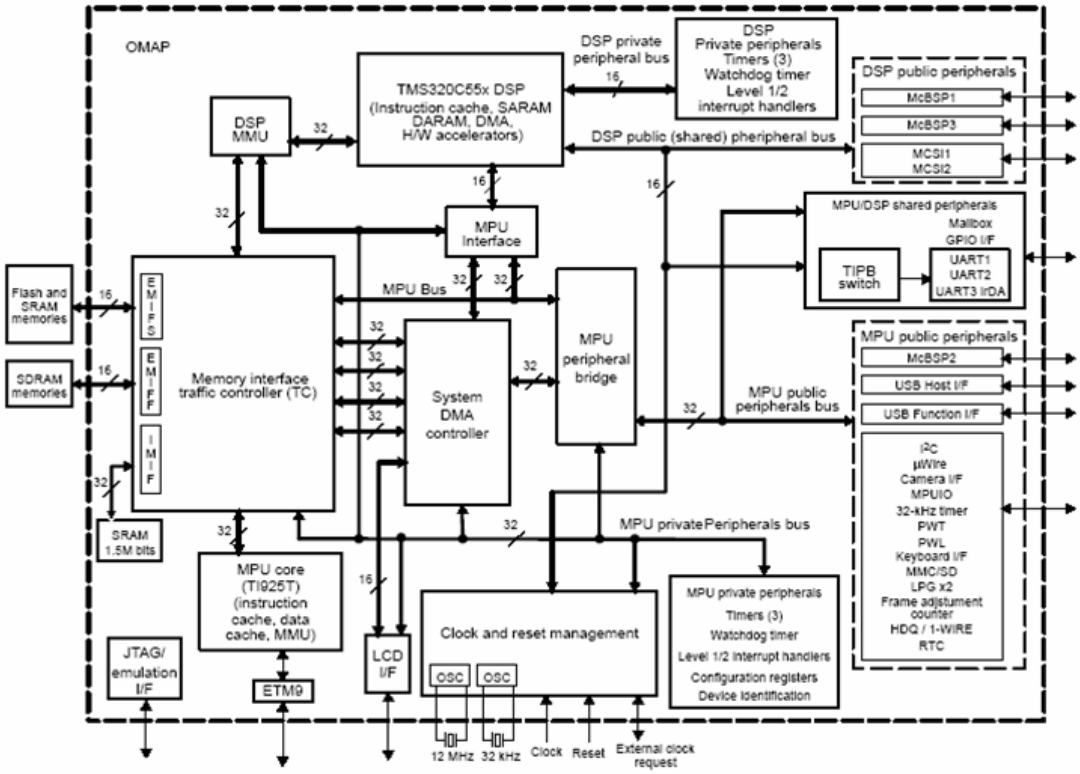

圖 三-1　OMAP1510 Internal

OMAP1510[12]是一個高度整合硬體與軟體的發展平台，是爲了符合新一代嵌入式裝置的應用所設計的。使用 OMAP 處理器可以加速產品開發到市場販售的速度，並擁有許多介面與週邊。因為 OMAP 高度整合兩顆不同核心的處理器，由於可以彈性的讓兩個處理器分工，所以它具有高處理效能與較低的耗電量。

雙核心架構提供了整合 DSP 和 RISC 這兩項不同的技術 - TI TMS320C55x DSP 核心與高效能的 TI925T ARM 核心。

- Inter-processor communication

  在 OMAP 處理器中的 ARM 與 DSP 資料交互傳遞的方法有下列三種

  - Mailbox

    ARM 與 DSP 處理器可以透過 Mailbox 中斷的機制做資料交換，這樣的機制可以提供軟體自定通訊協定讓處理器之間的通訊更有彈性。當 ARM 把資料寫進 DSP 時，DSP 會收到一個 Mailbox 中斷，藉由中斷 DSP 可以在 ARM 寫進資料後，就可以取出使用。DSP 也可以用類似的方法與 ARM 通訊。

    Mailbox 不適合拿來作大量資訊的傳輸，比較適合傳送命令、參數或是指標。

  - MPU Interface(MPUI)

    MPUI 可以讓 ARM 和系統的 DMA 控制器與 DSP 和 DSP 的週邊互相通訊。MPUI 可以存取整個 DSP 的記憶體空間與 DSP 的週邊匯流排。因此 ARM 與系統的 DMA 可以有 DSP 輸入輸出空間的完整存取權限。

    MPUI 適合用來做下列工作

    - 由 ARM 將 DSP 的程式載入到 DSP 的程式記憶體中。
    - ARM 與 DSP 之間的資料分享
    - 透過 Share Memory 達成行程間通訊的通訊協定
    - 讓 ARM 可以使用與控制 DSP 的週邊

  - Share Memory

這樣的架構是依靠 OMAP 上的 Traffic Controller 達成，藉由 ARM 與 DSP 存取相同的記憶體空間(含 SRAM 與其它記憶媒介)。ARM 可以控制 DSP 的 MMU 讓 DSP 可以存取這些記憶媒介。使用 Share Memory 的機制需要 Mailbox 與 MPUI 的幫助。

- TMS320C55x DSP 處理器簡介

C55x DSP 是一顆高效能、低耗電量的處理器。處理器中含有兩個乘累加器(MAC)，每一個 MAC 可以在一個週期內將 17-bit 的乘法完成。算術邏輯單元(ALU)有兩組，一為 40-bit 另一為 16-bit。

OMAP 的 C55x 內建三種影像處理硬體加速器

- ✧ DCT 硬體加速器
- ✧ Motion Estimation 硬體加速器
- ✧ Pixel Interpolation 硬體加速器

- TI925T RISC 處理器簡介

TI925T 是一顆使用精簡指令集(RISC)的 32-bit 處理器，它的核心使用管線(pipeline)的機制讓指令可以連續執行，處理器內部有自己的記憶體管理單元。

3.1.2 PSI OMAP Innovator

Innovator 平台是使用 TI OMAP 處理器的手持式可展開的 OMAP 開發平台，下圖(圖 3-2)為展開後的樣子。Innovator 平台增加了許多週邊設備如網路介面、音效輸入與輸出、手寫板還有與 OMAP 連結的一些週邊(如表 3-1)。

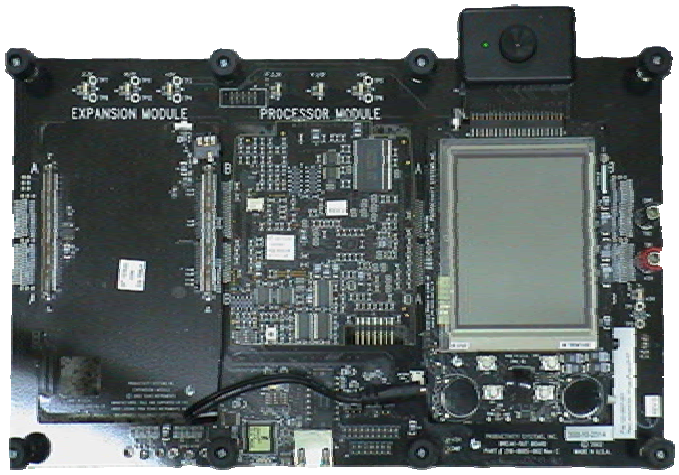

圖 三-2　PSI Innovator 發展平台

表 三-1　PSI Innovator Hardware List

PSI Innovator Hardware List
32MB SDRAM
32MB Flash
LCD with Touch screen
SD/MMC expansion slot
10Mb Ethernet
Dual RS-232 port
AIC23 Audio codec

3.2 軟體環境

本節將簡介開發程式的環境與 Innovator 平台上的軟體環境。

3.2.1 VMware

VMware[21]是一套由 VMware 公司所開發的一套虛擬機器軟體，它所模擬的系統

是屬於 x86 架構的個人電腦系統，它本身除了執行 x86 指令外，還模擬其他週邊設備，例如網路卡、序列埠、串列埠、USB 等其他一些基本的輸入輸出裝置，可以說軟體本身就是一個完整的個人電腦。

使用 VMware 的好處就在於可以在不同的作業系統環境中開發另一不同作業系統的應用程式，並可以不用經過作業系統的轉換直接測試程式的正確性。

使用 VMware 對於 OMAP 程式的開發（必須同時在 windows 作業系統上的開發 DSP 軟體與在 linux 作業系統上的 ARM 程式開發）是相當具有便利性，可以在單一台個人電腦上開發不同環境的程式。下圖(圖 3-3)是在 VMware 上執行 Linux 畫面。

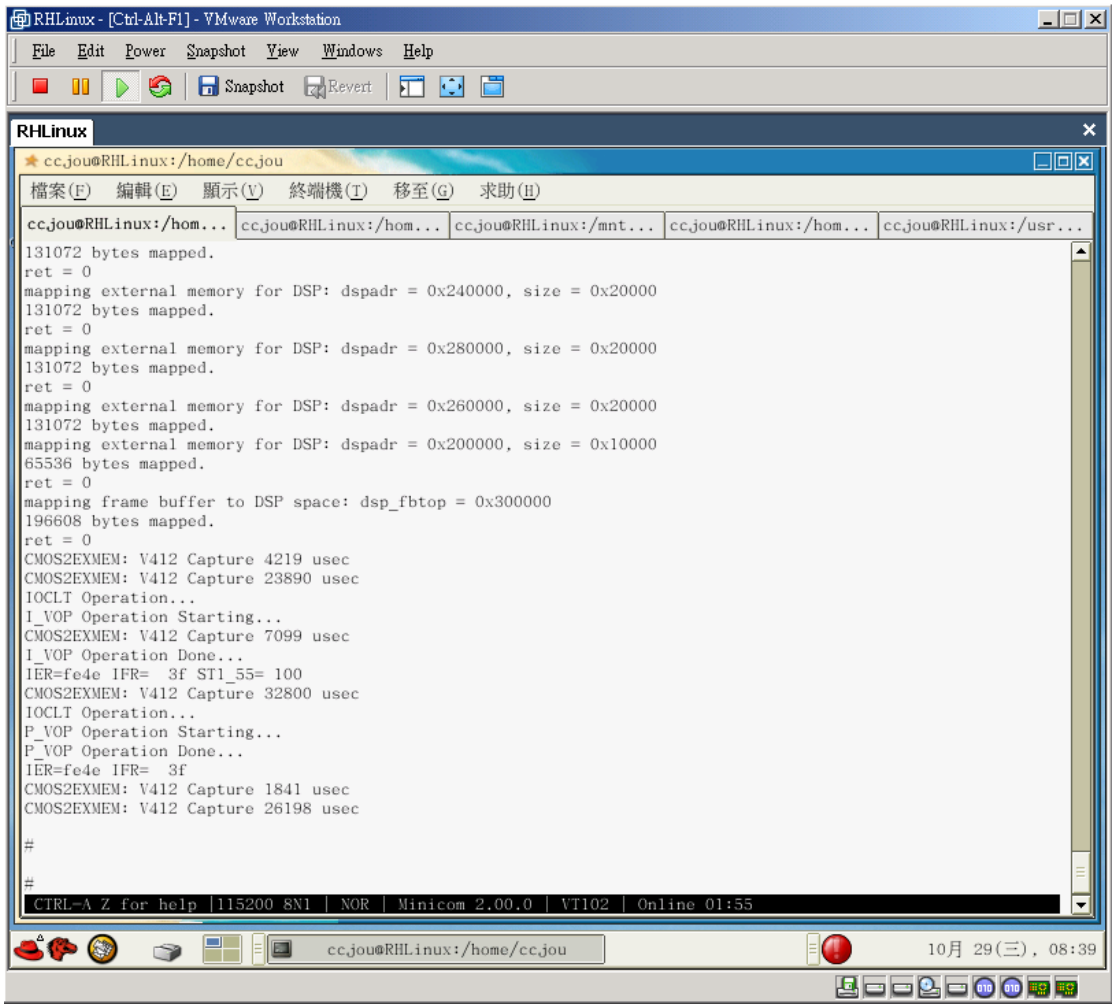

圖 三-3 VMware 執行 Red Hat Linux

3.2.2 Linux 作業系統

自從 1991 年 Linux 由 Linux Torvalds 將第一版 Linux 公開在網路上後，這套系統已經受到世界可地普遍的歡迎。現在的 Linux 已經支援相當多的處理器核心，所以現在 Linux 的應用相當的廣泛，由高階的伺服器、工作站，到常見的個電腦中都有它的蹤跡，甚至是一些手持式裝置例如手機、PDA 等一些現代人常用的嵌入式系統上都看得到 Linux 的存在。

所謂的嵌入式系統指的就是為了提高產品的穩定性、可攜性，或者是降低產品成本、增加特定功能等不同的目的，對一般的標準系統進行修改以符合特殊設計上的需求。在硬體方面靠的主要就是被稱為系統單晶片（SOC - System On a Chip）的半導體整合技術，而嵌入式作業系統則是軟體方面的技術核心。Linux 的可靠、高效能、伸縮性、低成本，以及最重要的握有控制權等幾項特點，使得 Linux 在嵌入式作業系統上站在非常有利的位置。

爲了讓 Linux 可以在 OMAP Innovator 上執行，必須使用以下修正檔，讓 OMAP Innovator 的處理器與週邊可以順利使用。

- ✧ ARM linux patch[23]
- ✧ Montavista OMAP Processor and Innovator Peripherals patch[24]
- ✧ DSP Gateway OMAP DSP patch[25]

3.2.3 ARM Processor Compiler

- ➢ GCC

GCC[22] 可以說是 Richard Stallman 所創立的 GNU 計畫中最重要的作品之一，它提供了自由軟體世界高品質的編譯器 (compiler)，實現了我們在自由軟體平台

上開發程式的夢想。如果沒有 GCC，恐怕今日我們也不會有這麼多形形色色的自由軟體可用。

GCC 一個很大的特色是高度可移植性，目前已知有超過三十種硬體平台與作業系統可以執行 GCC，其中硬體平台包括了: x86, arm,ia64, alpha, hppa, m68k, Power PC, mips, IBM rs6000, sparc/sparc64, ... 等，而作業系統則從 Microsoft 平台 (DOS/Win32) 到 IBM OS/2 到各家的 UNIX 都有。此高度可移植性正是 GCC 廣為流傳散佈的主要原因。在許多商業版的 UNIX 系統中，如果沒有特別買其專屬的編譯器的話 (其價格往往不便宜)，人們通常就選擇安裝 GCC 來使用。而且，儘管 GCC 是自由軟體計畫開發出來的，但其所編譯出來的程式品質並不輸給商業版的編譯器，甚至在某些平台上所編譯出來程式有更好的執行效能。

- Cross-compiler

通常在嵌入式平台上不會有多餘的記憶體與儲存空間來直接編譯程式，所以需要由其他機器產生嵌入式系統的執行檔，然後放到嵌入式系統上執行。Cross-compiler 是要在發展平台上產生目的平台的程式碼，在這裡發展平台是 x86 架構的個人電腦，目的平台則是 OMAP Innovator 上的 ARM 處理器。

在開發平台上使用的是 GCC 2.9.5 Cross compiler 工具組，這些軟體工具將放在 VMware 上的 Linux 上執行。

3.2.4 DSP Gateway

如前面 OMAP 處理器簡介中所提，OMAP 處理器包含 ARM925T 處理器與 TMS320C55x 數位訊號處理器。DSP Gateway 則是讓載 ARM 處理器上執行的 Linux 作業系統可以與 DSP 通訊的一套軟體。

DSP Gateway 包含兩部份，ARM 上所執行的 Linux 裝置驅動程式與 DSP 上的函式庫，靠著這兩著合作達到通訊的功能。Linux 裝置驅動程式提供介面讓 ARM 上的程

式可以 read()與 write()系統呼叫使用 DSP。DSP 的函式庫提供函式讓 DSP task 可以被 ARM 透過驅動程式使用。

DSP Gateway 中 ARM 與 DSP 處理器之間的通訊是靠著 OMAP1510 中 mailbox 的機制鎖完成，OMAP1510 中有三組 Mailbox 暫存器，一組是給 ARM 用，當 ARM 寫資料進 Mailbox 時，會送出中斷(INT5)給 DSP；其他兩組當 DSP 把資料寫進時，會送出中斷(IRQ10/11)給 ARM。在 DSP Gateway 中只使用一組 DSP 用的 Mailbox。每一組 Mailbox 是包含兩個 16-bit 的暫存器與一個 1-bit 的旗標暫存器。當中斷發生時，中斷的處理器會用兩個 16-bit 的暫存器各別將命令與資料傳到被中斷的處理器。

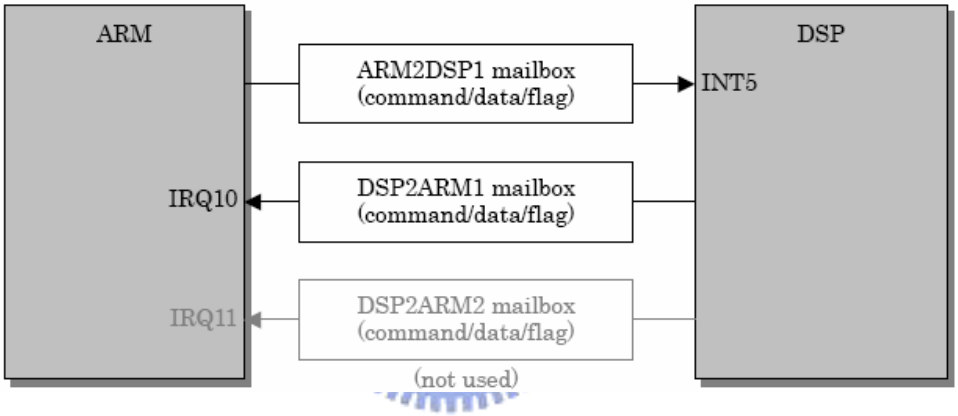

圖 三-4 DSP Gateway Mailbox

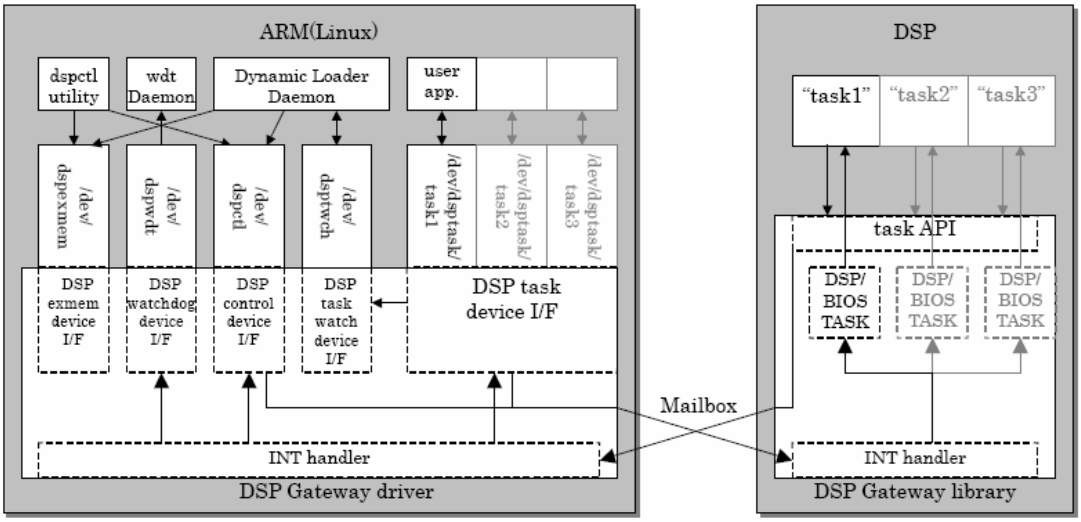

圖 三-5 DSP gateway

DSP Gateway支援五種裝置驅動程式 - DSP task devices， DSP task watch device，DSP control device，DSP watchdog device與DSP exmem device，上圖(圖 3-5)所表示的是這些裝置與應用程式之間的關係。 以下簡介這五種裝置驅動程式的用途。

- **DSP Task device**

  DSP Task device提供Linux應用程式使用DSP task的介面，它會再/dev/dsptask下自動產生device file。讀寫這些裝置意味著DSP task將會接收到資料或是傳送資料到Linux應用程式。

  DSP task分為ststic task與on-demand task。Static task與DSP Gateway的DSP函式庫做靜態鏈結，而on-demand task則不與DSP函式庫作靜態鏈結，它在載入DSP時才會和函式庫鏈結。

- DSP control device

  DSP control device提供DSP控制介面給Linux使用，Linux可以控制DSP重置、讀取DSP設定、爲on-demand task產生可以在DSP中執行的task、映射與取消映射SDRAM到DSP記憶體空間與其他功能。

- DSP exmem device

  DSP exmem device提供Linux可以存取DSP外部記憶體的介面。他在Linux上的裝置名稱為/dev/dspexmem。所以透過這個裝置可以讓Linux應用程式去存取由OMAPDSP_IOCTL_EXMAP命令所映射到DSP記憶體空間的SDRAM。

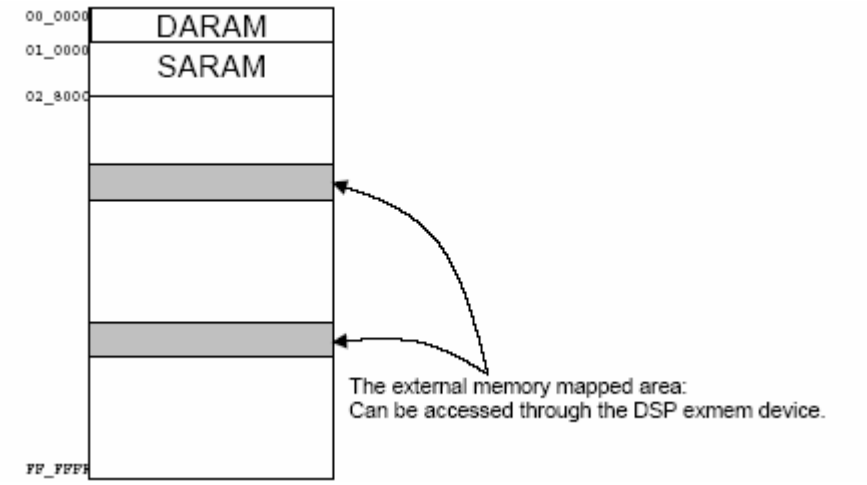

圖 三-6　DSP Memroy Space

如上圖(圖 3-6)所示，在映射 SDRAM 到 DSP 記憶體空間時必須映射到 0x028000 與 0xFF9000 的範圍內，且上圖的位址是使用 word Address，所以在使用 ARM 上的 Linux 應用程式映射時，必須改為 byte address。

- DSP Task Watch device

  DSP Task Watch device提供on-demand task要載入時的一些必要資訊，且提供哪些 DSP task沒在使用，哪些正在載入中，還有哪些已經執行的資訊。

- DSP watchdog device

  DSP watchdog device是用來偵測DSP watchdog timer是否逾時的裝置，它只提供read() 系統呼叫，一般的情況下使用read()時，是不會返回的。若是read()呼叫返回，那是代表DSP進入不可預期的狀態。

3.2.5　Code Composer Studio (CCS)

CCS 是一套整合式開發環境的發展軟體，它提供 C 語言的編譯器、組合語言的組譯器與其他工具，使用者也可以快速簡單的在軟體開發時除錯。在一般情況下它可以直接將 DSP 程式下載並在 DSP 上執行程式，但是在搭配 DSP Gateway 使用時無法直接將程式下載到 DSP 中，但是在開發 DSP 程式時還是要搭配 CCS 來編譯出 DSP 的執行檔，並藉由 DSP Gateway 讓 Linux 應用程式可以控制 DSP 的工作。

在 DSP 執行時可以藉由 Load symbol 的方式來對 DSP 程式除錯，但是必須 CCS 必須先啟動，讓 DSP 與 ARM 處理器由 Halt 的狀態改變成 Running 狀態，讓作業系統先執行，然後將 DSP 的程式藉由 DSP Gateway 讀進 DSP 後，才去 Load symbol。下圖(圖 3-7)是 CCS 的整合開發介面(IDE)。

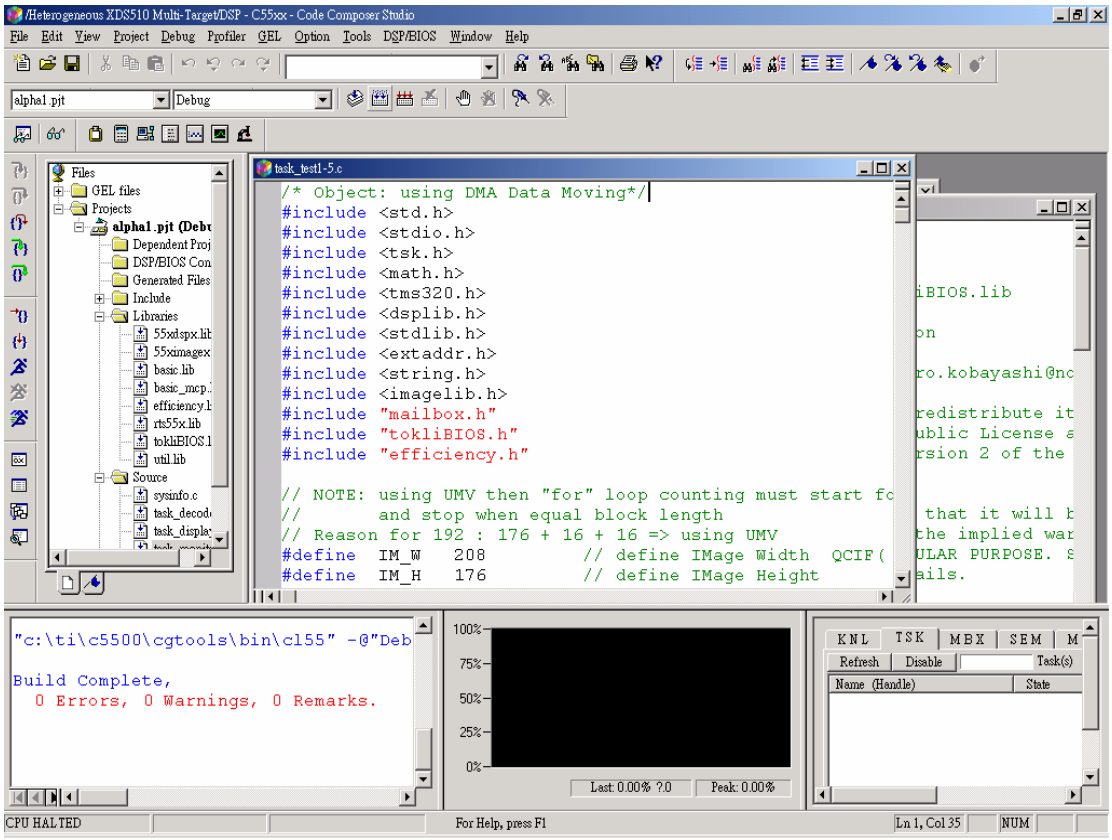

圖 三-7　CCS 整合發展環境介面

3.3　開發環境與程式開發流程

程式的開發環境是以個人電腦上的 windows 作業系統加上 VMware 上所執行的 Linux 為基礎，以 windows 作業系統上的 CCS 開發 DSP 程式，並使用在 VMware 上的 Linux 系統透過 Cross-compiler 來開發 ARM 程式。DSP 程式在編譯完後經由在 ARM 上面所執行的 Embedded Linux 來將 DSP 的可執行程式讀進 DSP 的記憶體中，並讓 DSP 程式在上面執行。

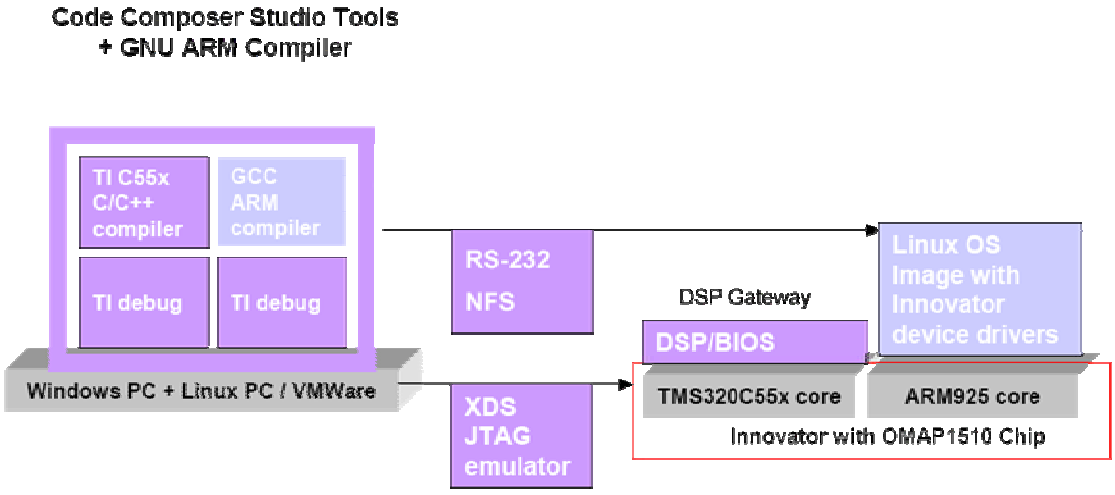

圖 三-8　發展平台架構

上圖(圖 3-8)就是整個開發平台的架構，當在個人電腦上的 ARM 程式與 DSP 程式開發完成後，不需要下載到硬體平台上，只要經過 NFS 網路檔案系統，使用 RS-232 介面來控制平台的運作，即可直接在平台上使用寫好的程式。

3.4　實驗與測試環境

實驗與測試環境是由兩組 OMAP Innovator、一個 Hub 與一台上面安裝 CCS 與 VMware 的個人電腦所構成，它們之間使用乙太網路相聯接，它們的示意圖(圖 3-9)如下

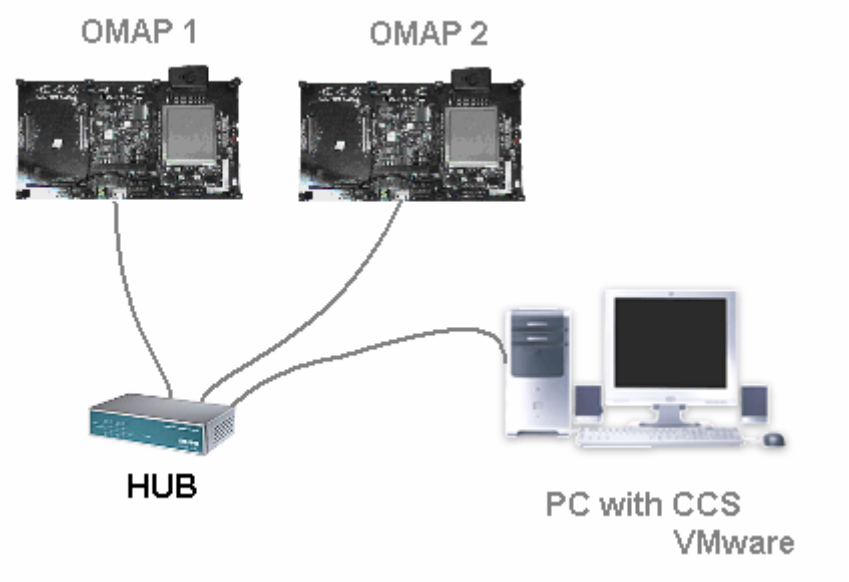

圖 三-9　測試與實驗平台

第四章 實作與最佳化

本章將介紹編碼器與解碼器實作上的一些細節與實驗結果，最後還有如何最佳化編解碼器與效能。

4.1 系統架構

編解碼器的系統架構如下圖(圖 4-1)

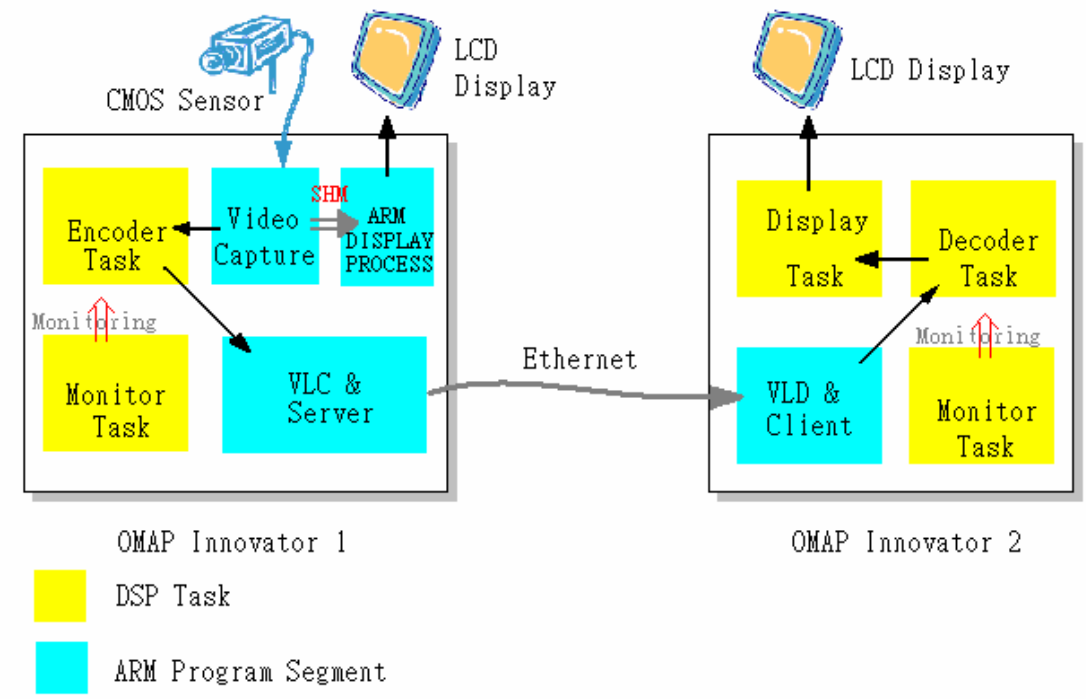

圖 四-1 系統架構

整個系統以兩台 OMAP Innovator 為主軸，在其上個別執行編碼器與解碼器，之間透過網路傳遞壓縮的資訊，壓縮檔頭為自訂並未支援任何已存在的標準。

4.2 編碼器實作

編碼器使用的是被普遍使用的 DPCM/DCT 架構，這個架構在第二章就說明過了，現在爲了要讓 DPCM/DCT 的架構在 OMAP 上可以使用，做了一些架構上的修改，讓 DSP 和 ARM 可以分工進行，架構如下圖(圖 4-2)所示。

4.2.1 編碼器架構

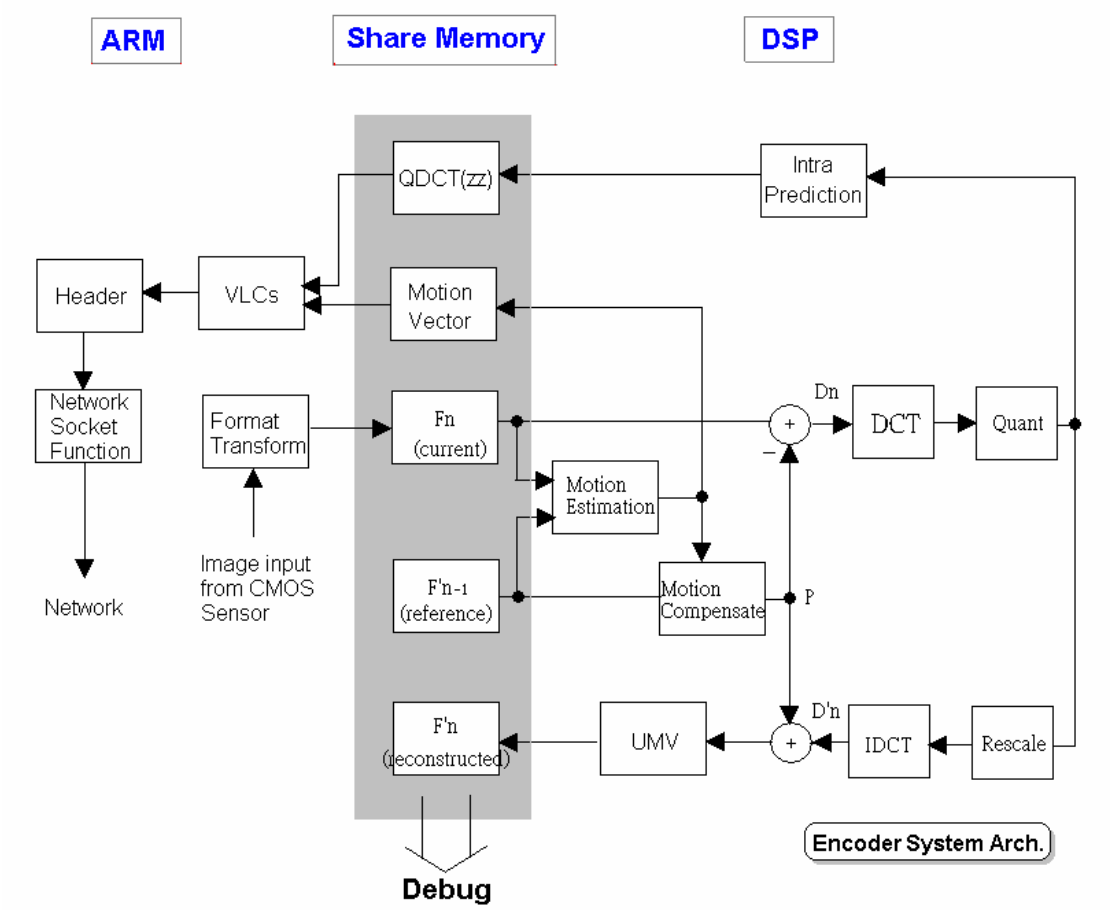

圖 四-2 編碼器程式架構

這個架構將原始 DPCM/DCT 架構中的可變長度編碼(VLCs)的部份獨立出來，交由 ARM 來處理，其他的部分還是由 DSP 來進行，ARM 除了要作 VLC 以外還要負責

將影像資料由 CMOS Sensor 擷取進來，進行格式上的轉換，再傳遞到 DSP 的外部記憶體中；當 DSP 做完運算後，ARM 還要負責將處理後的資料由 DSP 外部記憶體取出，且將做完可變長度編碼後的資料由網路輸出。使用如此的架構是希望 ARM 和 DSP 的處理時間可以相近，以利達成日後分工。

4.2.2 影像擷取與 LCD 顯示

- CMOS Sensor 影像擷取

在執行 Encoder 之前，必須要有輸入的影像作為來源，在 Innovator 發展平台上有一 CMOS Sensor，要由 CMOS Sensor 取的影像，必須透過 CMOS Sensor 的驅動程式，取得影像資訊。OMAP 上的 CMOS Sensor 與它的控制晶片的驅動程式是使用 Video for Linux Two(V4L2)。V4L2 是一組函式介面與控制 Linux 影像裝置的標準，它並不與 Video for Linux 的介面相容。

在 Innovator 發展平台上 Linux 中，V4l2 的裝置檔案位於/dev/v4l2/capture0，要使用 V4l2 擷取影像要依以下步驟(如下圖)。

使用 V4L2 擷取影像時，首先在使用時必須先設定要擷取的格式與相關資訊，接下來選擇是否要串流(Streaming)，當選擇使用串流時，接下來要使用 mmap()系統呼叫將驅動程式中的緩衝區映射到應用程式的記憶體空間，如此一來可以得到較大的資料通透量(throughput)，所以使用串流的速度會較不使用串流、直接向系統取得儲存空間，再由驅動程式的緩衝區中搬移過來的速度還快。

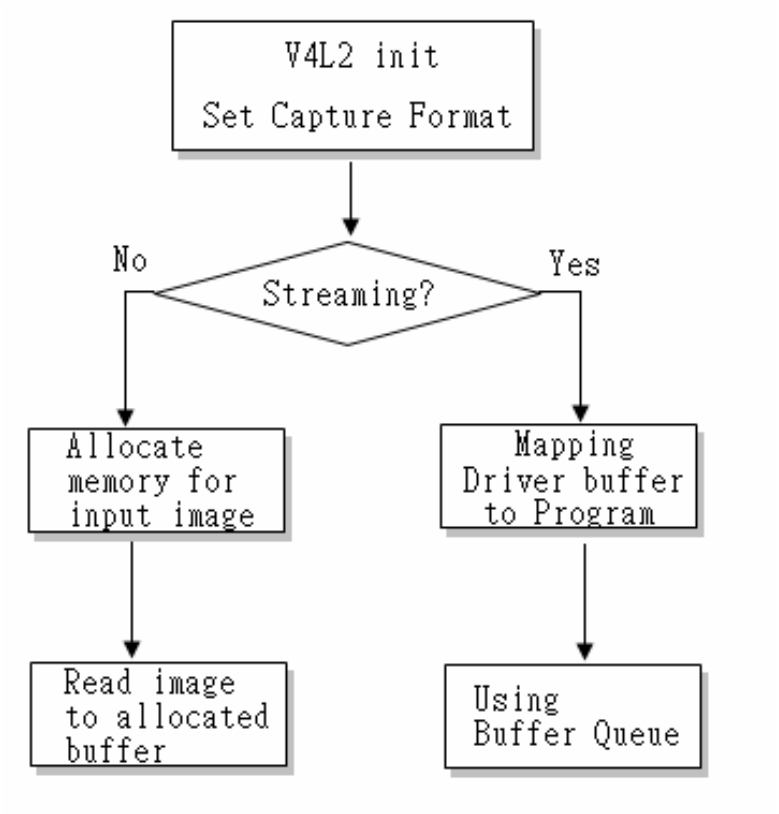

圖 四-3 V4L2 使用流程

- LCD 顯示

✧ Frame buffer

Frame buffer 裝置是一個繪圖裝置的抽象概念,它用來表示繪圖裝置的圖形緩衝記憶體，並且讓應用程式可以透過直接存取 Frame buffer 裝置，使得圖形可以顯示在繪圖裝置上，且並不需要知道任何低階的繪圖介面控制。所以 Frame buffer 裝置可以說是提供一個相當簡單的方法，使影像結果得使顯示在繪圖裝置上。

✧ 影像顯示流程

開啟 Frame buffer 裝置後，將驅動程式內的緩衝區映射到應用程式記憶體空間，就可以直接改變顯示在 LCD 上的結果，流程如下圖。

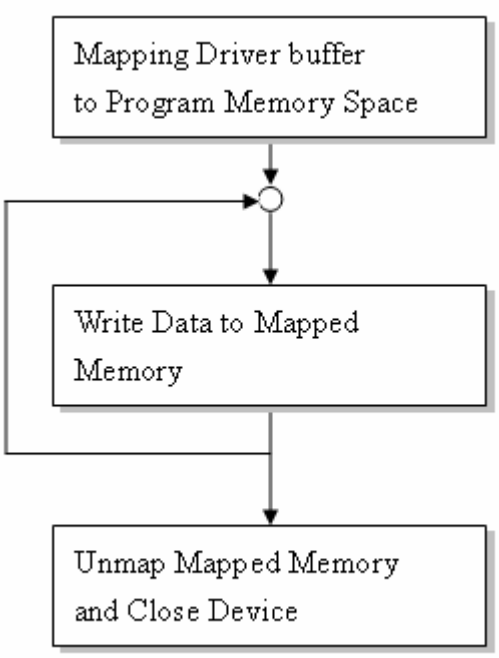

圖 四-4　Frame buffer 裝置使用流程

4.2.3 ARM 與 DSP 之間的資料傳輸

- 記憶體配置

由於影像資料相當的龐大，由於 Task 裝置檔案的緩衝區有限，無法快速的將大筆的資料傳送到 DSP，所以必須尋求直接使用 read()與 write() Task 裝置外的方法。OMAP 中有提供另一種處理器之間的通訊方法—Share Memory，這種方法適合大量的資訊傳輸。

使用 DSP Gateway 可以透過將 SDRAM 映射到 DSP 記憶體空間，再透過 /dev/dspexmem 裝置來存取映射過去的記憶體，這個方法透過實測，可以到 20MB/s，頻寬上已經足夠快了。

DSP 上的記憶體配置如下圖(圖 4-5)

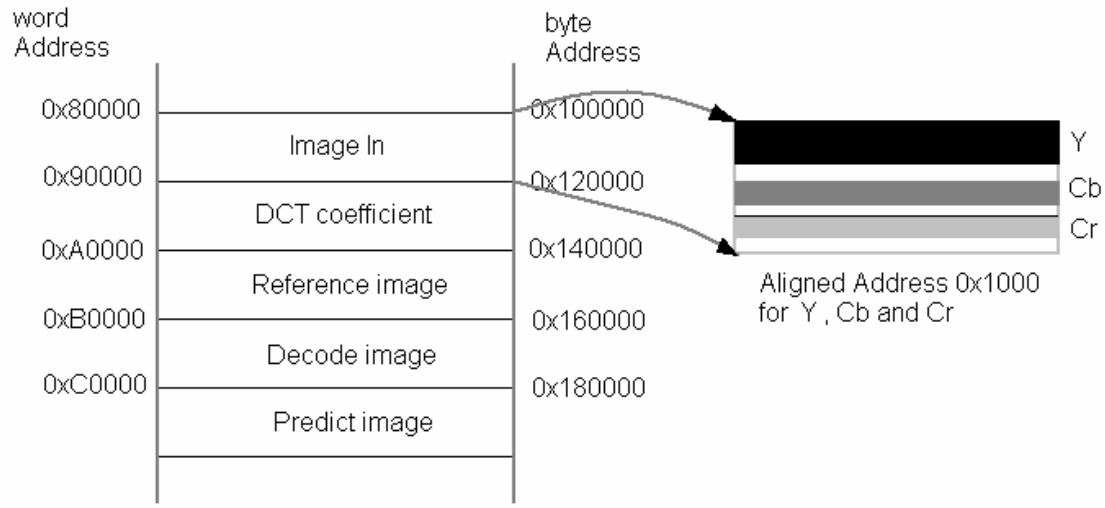

圖 四-5　Encoder 記憶體配置

編碼器的外部記憶體配置有五個大區域，每個區域長度是 0x20000，內部再細分為 Y、Cb 與 Cr，每一個細分的區域位址對齊 0x1000。對 ARM 而言，它只能掌握的資訊只有大區域的起始位址，若是要取得 Cb 或是 Cr 的資料，就必須利用大區域的起始位址做位移，讓 Cb 與 Cr 對齊 0x1000 可以方便 ARM 存取，也有利於再 DSP 上的資料搬移。

4.2.4 DSP Encoder Task

DSP Encoder Task 主要的任務是產生 zigzag 掃描後的量化 DCT 係數，並將整個 DPCM/DCT(除了 VLC)了流程完成，最後產生重建的參考影像。它包含了需多部份如量化、UMV、VLC 等。

4.2.4.1 量化(Quantization)

量化運算時多使用除法，使用除法會讓處理器的執行時間拉長，為了減少量化對執行時間的影響，必須對量化的方法加以改進。由於量化的部份是由 DSP 所處理，由

於 DSP(TMS320C55x)只支援整數運算且除法運算時間太長,所以把除法用乘法加移位運算來處理。轉換的公式如下

$$
\begin{aligned}
QDCT(i,j) &= \frac{\left\lfloor \dfrac{32\times DCT(i,j)}{Qmatrix(i,j)\times Quantizer\ Scale} + k \right\rfloor}{2} \\
&= \frac{\left\lfloor \dfrac{32\times DCT(i,j)\times 2^{16}}{Qmatrix(i,j)\times Quantizer\ Scale}\times 2^{-16} + k \right\rfloor}{2} \\
&\approx \frac{\left\lfloor \dfrac{2^{16}}{Qmatrix(i,j)}\times DCT(i,j)\times \dfrac{2^{-11}}{Quantizer\ Scale} \right\rfloor}{2} \\
&= \left\lfloor \mathrm{Reciprocal\_}Matrix(i,j)\times DCT(i,j)\times \frac{2^{-12}}{Quantizer\ Scale} \right\rfloor
\end{aligned}
\qquad (4.1)
$$

爲了計算上的方便，設定 Quantizer Scale 為 2 的冪次方，選擇 Quantizer 為 8，則上式變為

$$
\begin{aligned}
QDCT(i,j) &= \left\lfloor \mathrm{Reciprocal\_}Matrix(i,j)\times DCT(i,j)\times 2^{-15} \right\rfloor \\
&= \mathrm{Reciprocal\_}Matrix(i,j)\times DCT(i,j) >> 15
\end{aligned}
\qquad (4.2)
$$

使用這樣的方法可以讓除法運算變成乘法與移位運算，在執行速度上比起除法會快上很多。

4.2.4.2 VLC

實現 VLC 的方式是使用查表的方式，使用的 VLC 表是 MPEG-4 Standard Part2 Visual[02]附錄中的 VLC 表，它是屬於 Huffman Coding。在實現時必須將使用 zigzag 掃描後的量化 DCT 係數先使用 Run Length Coding 轉成(last,run,level)的格式後，再去查 VLC 表以產生 bitstream。

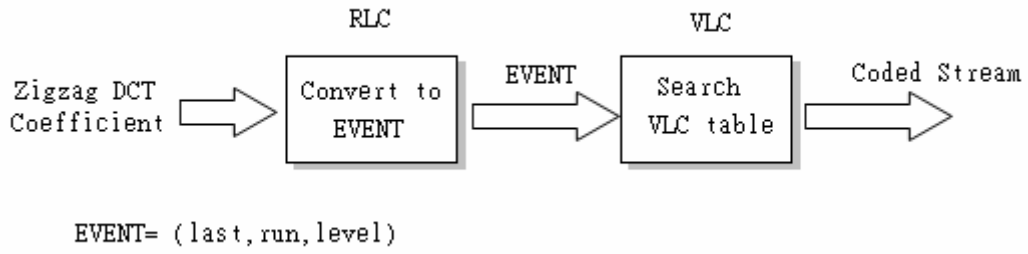

圖 四-6　VLC 流程圖

查表的好處在於可以快速的將相對應的 bitstream 快速找出，並且不需要把 Code book 傳遞到解碼端；但是缺點就是無法得到最佳的 Entropy，也會發生一些例外情形，這些例外情形都是發生機率比較低，所以沒有 VLC 表並沒有紀錄。當例外情形發生時必須加入一個機制，稱為 ESC code。

這裡所使用的是 MPEG-4 standard 中 ESC code 的第一種，它是使用(last,run,level)中的 last 與 run 去尋找對應的 level 的最大值(LMAX)，將目前的 level 減去 LMAX 後，得到新的 level_new，再用(last,run,level_new)去查表找出對應的碼。必須注意的是(last,run,level_new)還是有可能找不到對應的碼，這時候就必須再使用一次 ESC code 的機制，直到(last,run,level_new)落入表的範圍內，最後產生的編碼就會類似下圖。

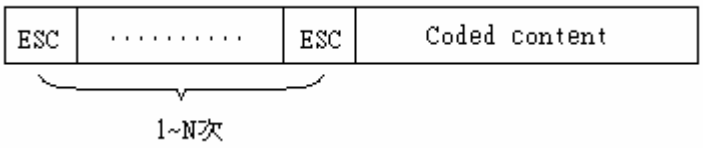

圖 四-7　VLC ESC CODE

4.2.4.3 Intra-prediction

Inter-frame 並不作 DC and AC prediction，因為 inter-frame 拿來作 DCT 的是它的 residual frame，residual frame 本身值就很接近 0，若用來預估的區塊 DC 不為 0 但是目前的區塊 DC 為 0 將會造成做完 VLC 之後的輸出內容大小比原來的大，所以 ac and dc

prediction 並不用在 inter-frame 上。在 Cb 與 Cr 之上也有類似的情形。

在做 intra-prediction 必須由最後一個 block reverse 運算回來(先算 Group2)，若是由前到後，由上到下計算(先算 Group 1)，將會造成下一次預估時的 DC 值並非原來的值，無法達成 intra-prediction 原先的目的。

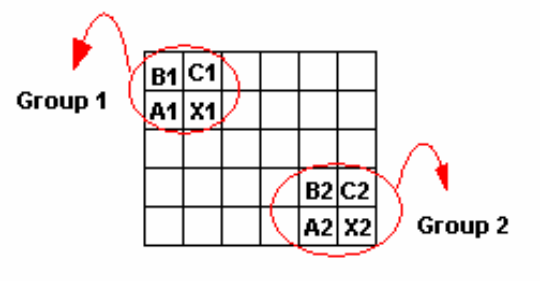

圖 四-8　intra-prediction 計算順序

加上 intra-prediction 後，I-frame 的編碼長度由 2335.8 變成 2001.7，編碼長度減少約 14.3%。

4.2.4.4 UMV

使用 UMV 可以讓影像中的移動物件由影像內往外移時的 Motion Estimation 更有效率，且使用 UMV 在程式架構上由於不用考慮邊界條件，所以程式複雜度也會降低。

使用以上演算法後，P-frame 的壓縮內容的長度(不含移動向量)為 837.5bytes。影像移動劇烈時會留下殘影，原因是因為最佳的預估區塊超出搜索範圍，所以無法產生最佳的預估影像，以至產生殘影，且 P-frame 的壓縮內容長度會增加為原來的 1.71 倍。在影像移動緩慢的情況下，壓縮比大約為 20 倍。下圖為剩餘影像、預估影像與剩餘影像與輸入的自相關性。

圖 四-9　預估影像(QCIF - predict frame)

圖 四-10　剩餘影像(QCIF - residual frame)

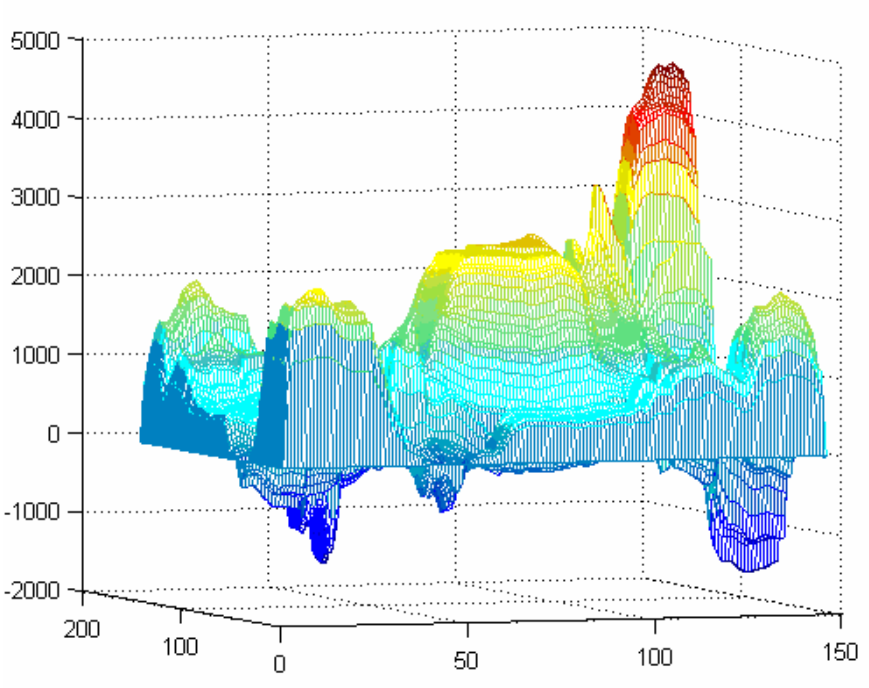

圖 四-11　原始影像的自相關矩陣係數

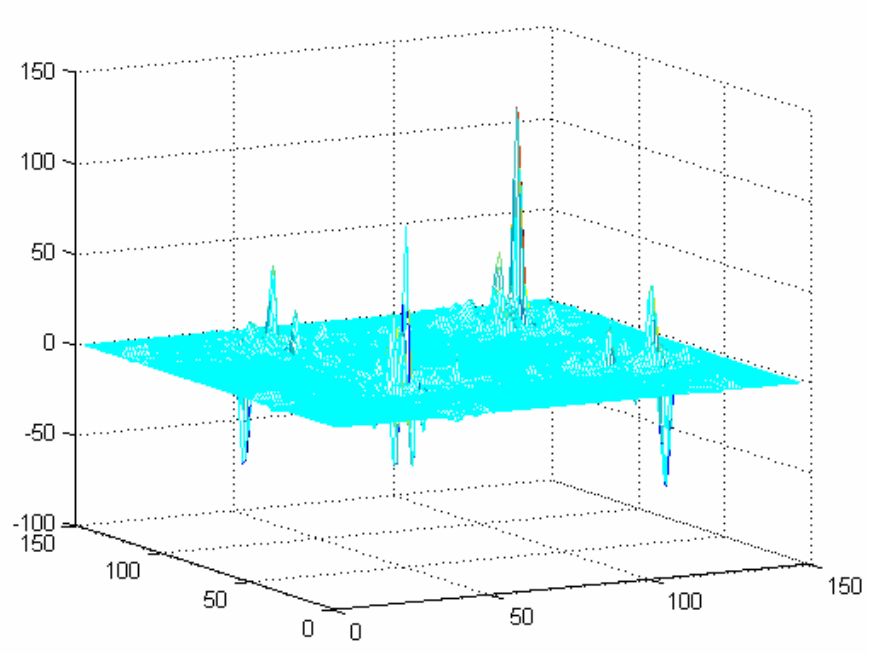

圖 四-12　剩餘影像的自相關矩陣係數

由圖 4-12 與圖 4-11 相比，移動預估與補償的確達到讓影像的相關性與能量減少的目的，以達到大量資料壓縮的目的。

4.2.5　DSP Monitor Task

DSP Monitor Task 主要的任務在於讀取 DSP 中 Encoder 與 Decoder 的狀態旗標，並回傳到 ARM 中。這個任務是 ARM 用來和 DSP 同步的任務，它使用較低的優先權，當 DSP Encoder Task 或是 DSP Decoder Task 在執行時，它會被停住，所以當 DSP 在執行 Encoder 或 Decoder 的同時，ARM 使用 read 函式要來讀取這個 Task 的值時，會因為 read 本身使用 block I/O 的關係而停止，等到 DSP 執行完 Encoder 或 Decoder 時，Monitor Task 程式執行，read 函式會返回，並傳回 Encoder 與 Decoder 的狀態旗標。簡單的說，當 read 被 block 時，就是 Encoder Task 或 Decoder Task 未完成；當 read 回傳值時，就是 Encoder Task 或 Decoder Task 已完成。

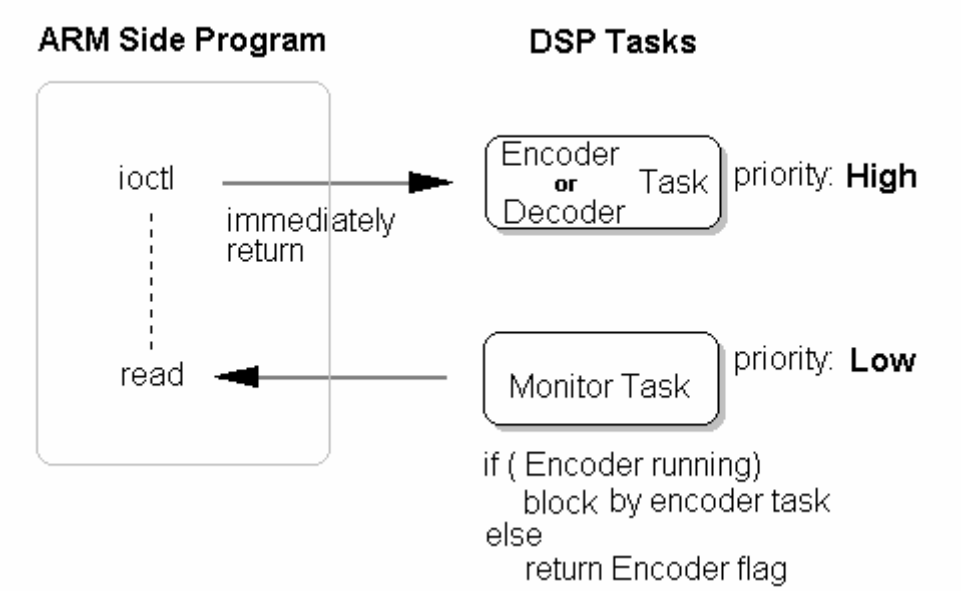

圖 四-13　Monitor Task 與 Encoder/Decoder Task 的關係

因此，使用如此的方法可以讓 ARM 程式知道什麼時候該可以繼續往下執行，以達成 ARM 與 DSP 之間工作的同步。

4.3 解碼器實作

由於編碼器必須解回壓縮過的影像來當作參考影像，讓解碼器的在解出影像可以更接近真實影像，所以在編碼器之中就已經包含解碼的流程。所以時作解碼器時只要將解碼的部分由編碼器抽出，再加上一些逆運算如 VLD 與 inverse intra-prediction。

4.3.1 解碼器架構

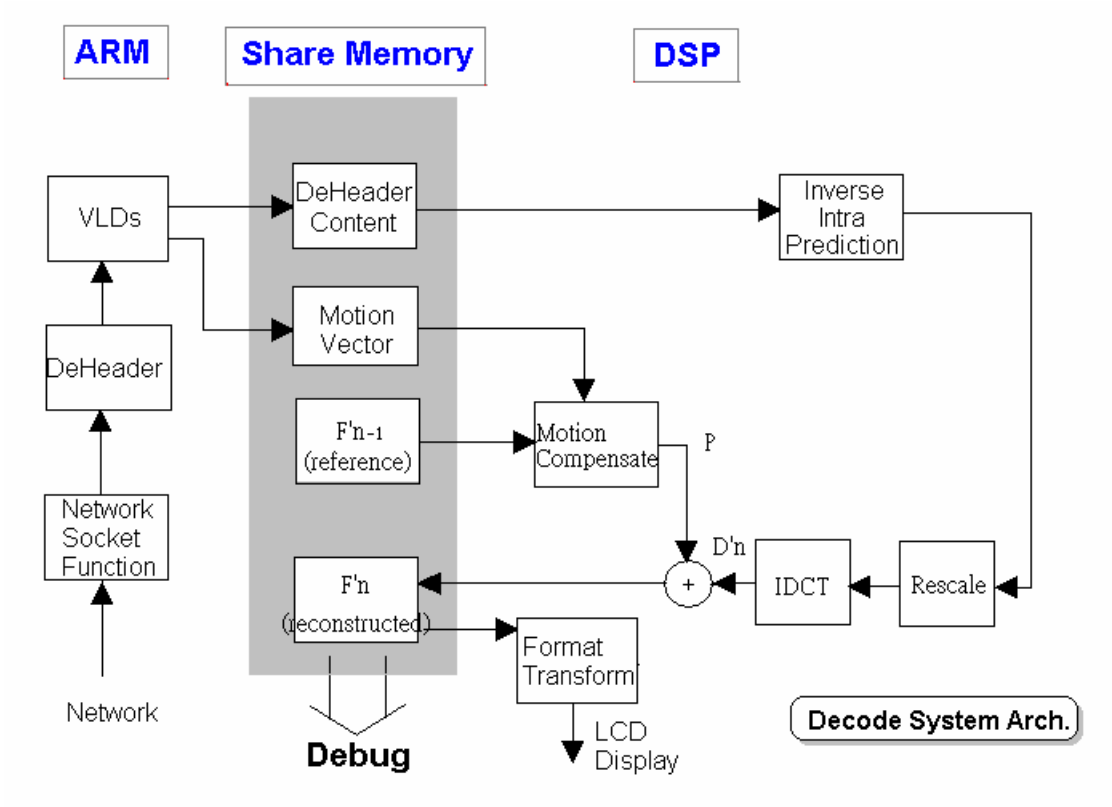

圖 四-14 解碼器程式架構

解碼器的實作方面包含 Decoder Task、之前的 Monitor Task 與 Display Task，這些 Task 的內容在後面會有清楚的解說。

4.3.1.1 Inverse Intra-prediction

在做 inverse intra-prediction 時，執行的順序剛好和 intra-prediction 相反，必須由上而下，由左而右執行。在執行時還是要執行 intra-prediction 的判斷式，找出預估的區塊，再將目前區塊的 DC 與 AC 值與預估區塊的相加，解回原來的 AC 與 DC 值。所以在還原 intra-prediction 的影像時並不需要另外讓 Encoder 傳送預估方向的資訊。

4.3.1.2 VLD

做 Huffman 解碼時，在這裡使用的是先重建 Huffman tree 的方法。使用重建 Huffman tree 來尋找對應的(last,run,level)會比使用比對 Huffman 表的速度還要快，唯一的缺點就是要多花一些記憶體空間儲存二元分類樹的資料結構。

解碼時依以下流程(圖 4-15)

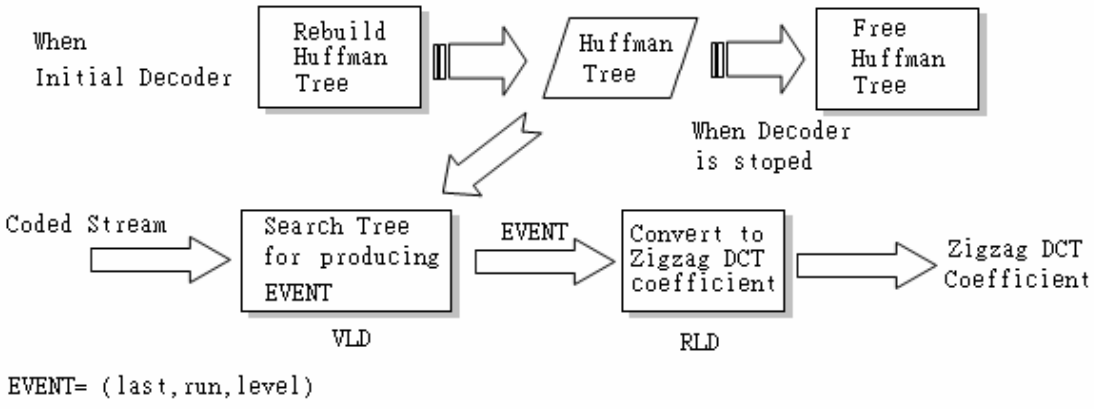

圖 四-15 VLD 流程圖

如上圖，重建 Huffman tree 只有在解碼器初始化的時候，所以他所花費的時間對整個解碼的流程並無太大的影響。

4.3.2 DSP Display Task

在 DSP 解碼完後，還原的影像如果還要傳回 ARM，再由 ARM 程式輸出到 Frame buffer，將會造成一次多餘的資料搬移，若是能由 DSP 直接輸出到 Frame buffer 就可以省下一次搬移的時間，如下圖所示。

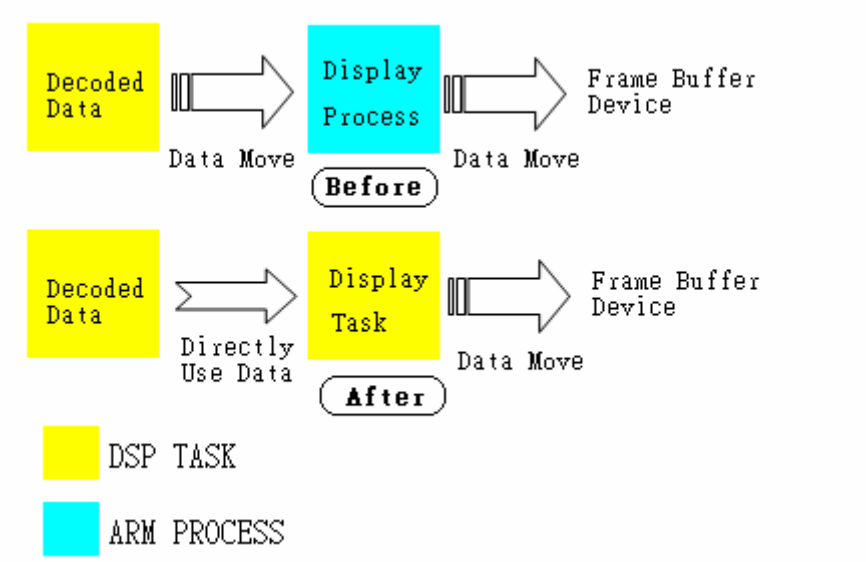

圖 四-16 DSP Display Task

在這裡，藉由 DSP Gateway 將 Frame buffer 映射到 DSP 的記憶體空間，讓 DSP 的 Task 也可以直接使用 LCD 螢幕。

4.3.3 DSP 解碼器優先權設定

由於解碼器端有三個 DSP Task 同時運作，所以在優先權的設定上就要相當的小心，以避免錯誤發生。以系統需求而言，解碼器與 Monitor Task 必須比 Display Task 要有更高的執行優先權，以保證可以每次顯示時，都可以使用最新的資料。而且若是 Display 的執行權限高，因為 Display Task 內是執行無窮迴圈，將會造成解碼器(Decoder Task)無法執行。下圖(圖 4-17)為優先權的示意圖。

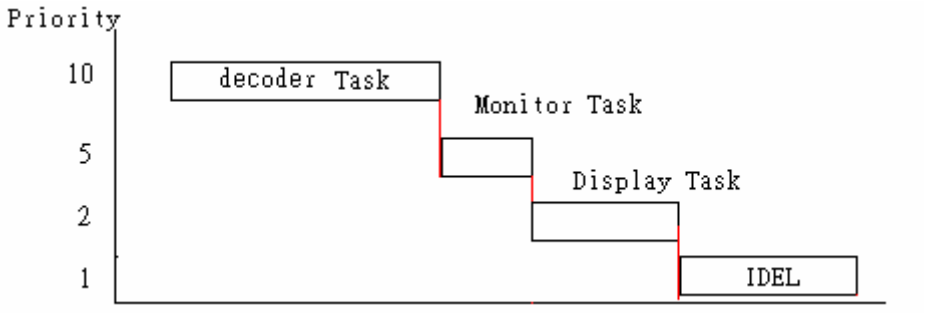

圖 四-17 Priority for DSP Tasks

上圖的 Display Task 執行的時機通常會是在解碼器等待編碼器將資料送過來的閒置時間。

4.4 最佳化

下表為未最佳化的測試結果，測試結果單位為毫秒。

表 四-1　未最佳化執行時間

<table>
<tr><td></td><td colspan="3">Encoder</td><td colspan="2">Decoder</td></tr>
<tr><td rowspan="5">ARM</td><td rowspan="3">CMOS Sensor Capture</td><td>total</td><td>41.510</td><td>VLD and Network</td><td>220.61(depend on Encoder)</td></tr>
<tr><td>Capture</td><td>2.1</td><td rowspan="2">Write to EXMEM</td><td rowspan="2">2.816</td></tr>
<tr><td>RGB to YCbCr</td><td>31.65</td></tr>
<tr><td colspan="2">Read from EXMEM</td><td>4.062</td><td colspan="2" rowspan="3"></td></tr>
<tr><td colspan="2">Write to EXMEM</td><td>3.715</td></tr>
<tr><td colspan="2">VLC and Network</td><td>47.24</td></tr>
<tr><td>DSP</td><td colspan="2">DSP VOP Encoder</td><td>206.773</td><td>DSP VOP Decoder</td><td>59.685</td></tr>
</table>

4.4.1 DSP 程式最佳化

- 使用 IMGLIB

OMAP 的 DSP 處理器 C55x 上內建影像處理加速器，IMGLIB[11]是可以發揮影像加速器的函式庫。這裡使用 IMGLIB 的三個函式，這三個函式在壓縮上佔了很重

的比例，分別為 FDCT、IDCT 與 Motion estimation(ME)這些函式執行一次所需的時間

表 四-2　Benchmark of IMGLIB

Function	Benchmark (cycle)
FDCT	240
IDCT	179
4-step ME	2225

以 4-step ME 為例，若是一個尚未最佳化的 4-Step ME 演算法與硬體加速的演算法比較，使用硬體加速的演算法比未最佳化的演算法慢約 30ms。

- 局部最佳化

用 DSP Gateway 時並無法直接將寫好的 DSP 程式做最佳化，因為最佳化後執行會造成執行上的錯誤，所以在做最佳化時必須將要最佳化的程式區段獨立出來，將獨立的區段編譯為函式庫。在編譯函式庫時，就可以開啟最佳化選項，這樣雖然不能最佳化整體程式，但也能達到最佳化的目的。

下表為將程式抽出編譯成函式庫後，整體 DSP 編碼程式所耗費的時間

表 四-3　局部最佳化後的結果

最佳化項目	DSP 執行時間 (msec)
UMV 與 intra-prediction	205,69
I-frame 量化與 DCT 部分	186.769
P-frame 減少 Pixel merge 資料搬移	184.301
P-frame 將 MCP 最佳化 (除了 Pixel merge)	178.364
將 pixel merge 部份最佳化	116.417

- 使用 Dual-MAC

OMAP 上的 DSP C55x 支援 Dual-MAC，也就是它有兩個乘累加器，如果可以善用兩個乘累加器來輔助運算，那麼就可以達到較高的編碼效率。

使用 C 語言的情況下，要使用 Dual-MAC 來加速運算，必須符合一些規定，才能讓編譯器編出 Dual-MAC 的程式。

✧ 必須有一相同的輸入

✧ 必須在內部記憶體中

由於演算法中乘法運算的次數不多，而且還有以上的限制，造成使用 Dual-MAC 的助益並沒有很大。

- 使用 DSP DMA

DSP 中有自己的 DMA，DSP 中的 DMA 總共有 6 個通道，個別連接在 SARAM、DARAM、EMIF 等記憶體與週邊上，讓 DSP 可以通過 DMA 控制器將資料內容不必透過處理器，搬移到目的位址。

由於透過 CPU 搬移記憶體，尤其是搬移外部記憶體的內容更是耗費時間，透過 DMA 的搬運將可以將資料搬移的時間降低。

4.4.2 ARM 程式最佳化

ARM 程式最佳化不只是靠編譯器的最佳化選項，還要必須靠一些程式上的編排，讓 OMAP 的架構得以發揮。原來的程式架構如下圖

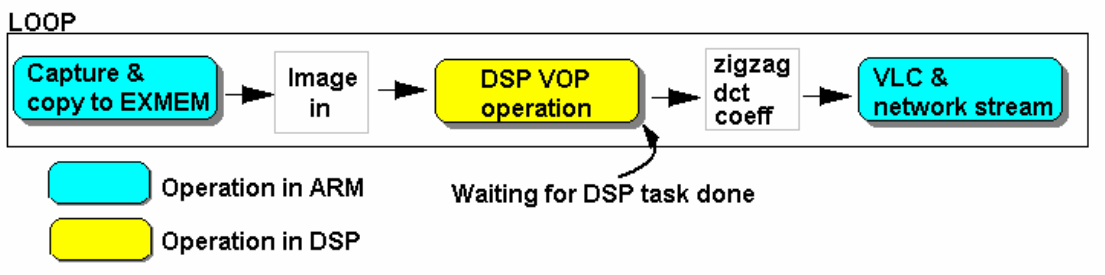

圖 四-18 原始程式執行架構

這樣的架構必須等待 DSP 執行完才能繼續向下執行，在等待時 ARM 是處於閒置狀態並不工作，如果可以讓 ARM 的其他工作可以在 DSP 處理時同時運作，那將可以省掉相當多的時間。所以在這裡引進 Software pipeline[27]的概念。

Software pipeline 的概念是減少迴圈無法執行在最高速的時間，將迴圈內每一個元件的錯開，讓彼此之間的相關性降低，讓迴圈可以一直保持在高速執行。在這裡迴圈內無法高速執行的最大關鍵是等待 DSP 執行結束，現在將相近的迴圈內的元件展開如下圖

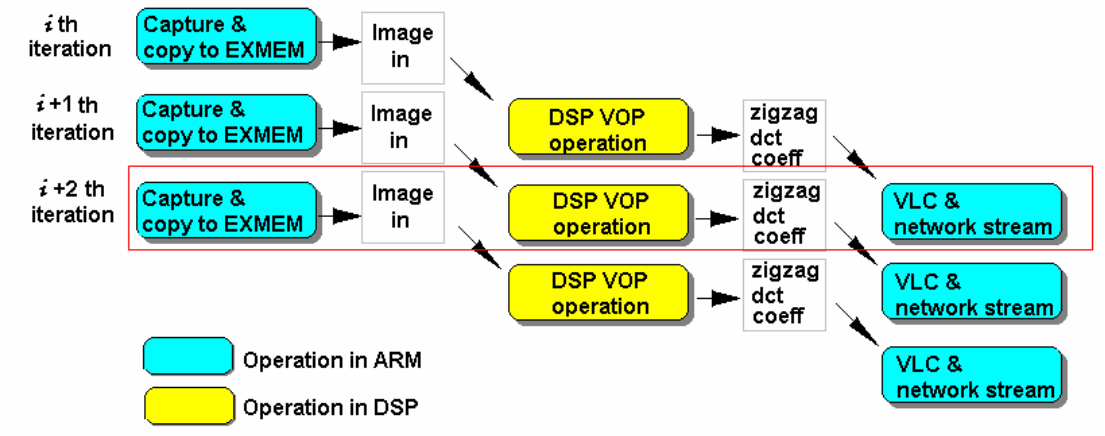

圖 四-19 相鄰的迴圈展開

將彼此無資料相關性的元件放進同一迴圈如下圖

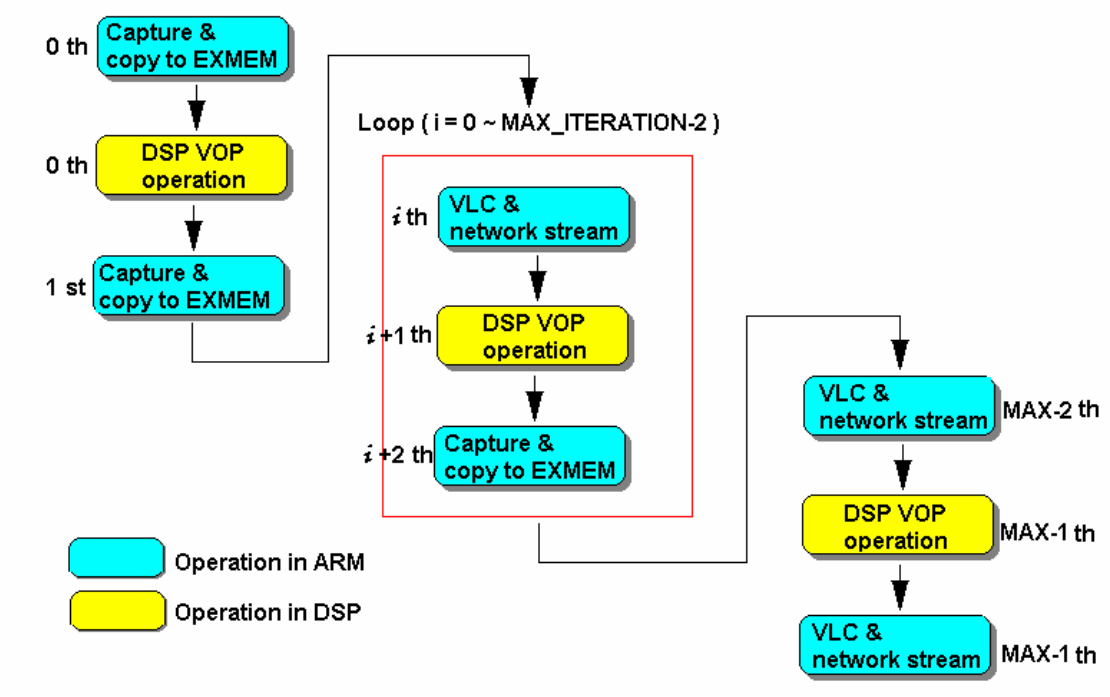

圖 四-20 Software pipeline - 1

但是這樣只是將迴圈內的資料相關性降低，ARM 卻無法利用 DSP 執行時的時間來工作，所以將上面的架構稍微修改如下圖

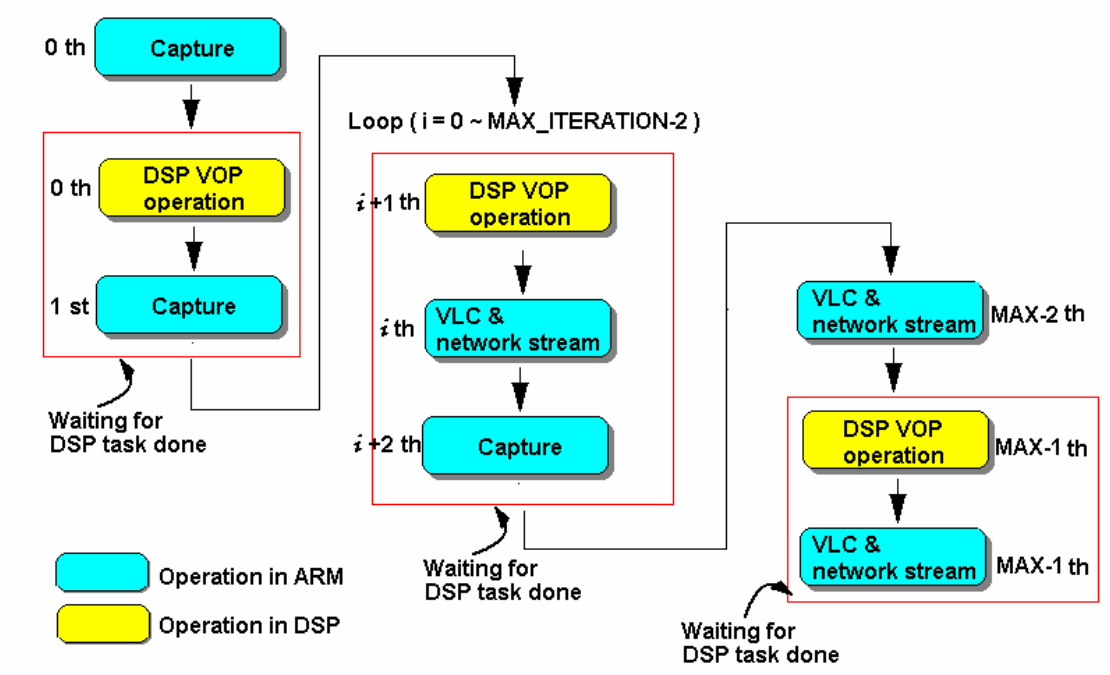

圖 四-21 Software Pipeline – 2

若是要在 DSP 執行時，讓 ARM 執行工作，則 DSP 的工作一定要在迴圈的最前端執行，否則會向第一次的 Software pipeline 架構，無法從中取得好處。所以修改後將 copy to EXMEM 的部份加進 DSP VOP operation 中，並且將 DCT 係數取出的任務也給 DSP VOP operation，這樣可以確保後面先執行的元件不會去覆蓋到還未處理的結果。

且這樣的架構有另外一項好處，不管 DSP 的 Task 或是 ARM 的程式元件誰先完成，這個架構都不會執行錯誤。假設 ARM 程式元件先執行完，那麼 ARM 程式將會停在等待 DSP 完成的判斷式；若是 DSP 工作先完成，也就是說在 ARM 程式元件執行之中，DSP 完成工作，那麼由於資料上並無相關性，所以也不會執行錯誤，最後的判斷式也會馬上返回，不會被 Monitor task 所阻塞。

修改後的整個程式的執行時間如下表(表 4-4)

	Before	After
Total ARM execution time(msec/frame)	211.3	131.6

表 四-4　Software pipeline 後的執行時間

第五章 結論與未來展望

5.1 結論

本論文成功的將 OMAP Innovator 的開發環境與其上的作業系統架構完成，且在其上成功的將動態影像的編解碼器實現，且此動態影像編解碼器可以將 OMAP Innovator 上的 CMOS Sensor 擷取的影像透過網路媒介，將編碼器所編碼的影像送到解碼器端解碼，並顯示在 OMAP Innovator 的 LCD 螢幕上。另外成功的在平台上透過 OMAP 上 RISC(ARM)與 DSP 的合作，使用論文所提的方法讓處理的速度有所提升。

整個編碼器可以每秒產生 7.66 筆壓縮內容，整個系統的效能被 DSP 的執行時間所限制，如果能將 DSP 的執行時間再縮短，那就可以將系統的效能再做進一步的提升。

5.2 未來展望

未來可以將 RVLC、Data Partition、Short Header、Video packet 等 MPEG-4 Simple profile 中的 Tools 實作出來，讓本編碼器與解碼器可以符合 MPEG-4 的標準。在 DSP 程式方面可以把較耗時間的運算使用組合語言實作，並充分利用 C55x Dual-MAC 的優勢，將 DSP 演算法的時間縮短；且全面將資料搬移使用 DSP DMA 實現，讓 DSP 程式的效能得以有效提升。在 ARM 程式方面可以改寫驅動程式，直接將應用程式的影像資訊直接映射到 DSP 的記憶體空間，以節省資料搬移時間。

參考資料

[01] Iain E. G. Richardson, H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia, John Wiley & Sons, 2003

[02] ISO/IEC 14496-2, Coding of Audio-Visual Objects – Part2:Visual, 2001.

[03] K. Sayood, Introduction to DATA compression, 2ed edition, Morgan Kanfmann, 2000

[04] L.M. Po and W. C. Ma, “A novel four-step search algorithm for fast block motion estimation,”*IEEE Trans. Circuits Syst. Video Technol.*, Vol.6, pp.313-317, June 1996.

[05] S. Zhu and K. K. Ma, “A new three-step search algorithm for block motion estimation,” *IEEE Trans. Circuits Syst. Video Techno.*, Vol.4, pp.438-442, Aug. 1994.

[06] S.Zhu and K. K. Ma, “A new diamond search algorithm for fast block-matching motion estimation,” *IEEE Trans. Image Processing*, vol.9, pp.287-290, Feb. 2000.

[07] A. V. Oppenheim, R. W. Schafer and J. R. Buck, Discrete-time signal processing, 2nd edition, Prentice Hall International, 1999

[08] Recommendation ITU-R BT.601-5, Studio encoding parameter of digital television for standard 4:3 and wide-screen 16:9 aspect ratios, ITU-T, 1995

[09]OMAP5910 Dual-Core Processor Data. Texas Instruments. Dallas, Texas. [Online]. Abailable: http://www.ti.com.

[10]T. Kobayashi, Linux DSP Gateway Specification, 2nd edition, Nokia cooperation, 2002

[11]TMS320C55x Image/Video Processing Library Programmer's Reference Preliminary. Texas Instruments. Dallas, Texas. [Online]. Abailable: http://www.ti.com.

[12]OMAP5910 Dual-Core Processor DSP Subsystems Reference Guide. Texas Instruments. Dallas, Texas. [Online]. Abailable: http://www.ti.com.

[13] TMS320VC5503/5507/5509/5510 Direct Memory Access(DMA) Controller Reference Guide (Rev. B). Texas Instruments. Dallas, Texas. [Online]. Abailable:

http://www.ti.com.

[14]TMS320C5000 DSP/BIOS Application Programming Interface (API) Ref Guide (Rev. F). Texas Instruments. Dallas, Texas. [Online]. Abailable: http://www.ti.com.

[15]TMS320C55x Optimizing C/C++ Compiler User's Guide (Rev. E). Texas Instruments. Dallas, Texas. [Online]. Abailable: http://www.ti.com.

[16]TMS320 DSP/BIOS User's Guide (Rev. B). Texas Instruments. Dallas, Texas. [Online]. Abailable: http://www.ti.com.

[17]TMS320C55x DSP Programmer's Guide . Texas Instruments. Dallas, Texas. [Online]. Abailable: http://www.ti.com.

[18]TMS320C55x Hardware Extensions for Image/Video Applications. Texas Instruments. Dallas, Texas. [Online]. Abailable: http://www.ti.com Programmer's Reference

[19] C. Hollabaugh, Embedded Linux-Hardware ,Software, and Interfacing, 1st edition, Pearson Education, 2002

[20] 中華民國軟體自由協會 (2001, Aug.). 自由軟體總藍圖. (1st ed.) [Online]. Available: http://www.slat.org/project/software-map/v1.01/map.html

[21] VMware, http://www.vmware.com.

[22] GNU Compiler Collection, http://gcc.gnu.org.

[23] The ARM Linux Project, http://www.arm.linux.org.uk/.

[24] Montavista OMAP Innovator Linux Patch, http://www.mvista.com/.

[25] DSP Gateway for Linux, http://dspgateway.sourceforge.net/.

[26] Bill Dirks (2003, June 26). Video for Linux Two. [Online] http://www.thedirks.org/v4l2/.

[27]H. Patterson, Computer Architecture: A Quantitative Approach, 3rd edition, Morgan Kanfmann, 2003

[28] R. Stones and N. Matthew, Beginning Linux Programming, 2nd edition, Wrox Press, 1999

[29]D. E. Commer and D. L. Stevens, Internetworking with TCP/IP volume III: Client-Server programming and Applications BSD socket version, 2nd edition, Prentice Hall, 1996

[30] A. Rubini, Linux Device Drivers,1st edition, O'Reilly & Associations, 1998