國立交通大學

電機與控制工程學系

碩士論文

JPEG XR 編碼器之架構設計與實現

Architecture Design and Implementation for
JPEG XR Encoder

研究生：簡清彥

指導教授：黃聖傑　博士

中華民國九十七年六月

JPEG XR 編碼器之架構設計與實現

Architecture Design and Implementation for JPEG XR Encoder

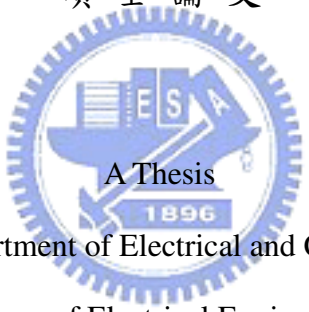
研 究 生：簡清彥　　Student：Ching-Yen Chien

指導教授：黃聖傑 博士　　Advisor：Sheng-Chieh Huang

國 立 交 通 大 學

電 機 與 控 制 工 程 學 系

碩 士 論 文



A Thesis

Submitted to Department of Electrical and Control Engineering

College of Electrical Engineering

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Electrical and Control Engineering

June 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年六月

JPEG XR 編碼器之架構設計與實現

研究生：簡清彥　　指導教授：黃聖傑 博士

國立交通大學電機與控制工程學系碩士班

摘要

快速發展的感測器、顯示裝置、運算引擎與有效的演算法與架構，使得影像無所不在，過去幾年，JPEG 已經融入了我們的生活，像是數位相機、部落格等等，但 JPEG 所能提供的影像品質有限，在 2006 年，新的 JPEG XR 影像壓縮標準已經提出並且在此討論與使用 VLSI 架構去實現，JPEG XR 擁有相當高的編碼效率與豐富的編碼功能，在相同壓縮倍率下，更能提供與 JPEG2000 相似的影像品質。

本論文中，會對 JPEG XR 編碼器進行演算法分析與提出新的硬體架構，除此之外，4：4：4 的 HD 影像可以順利編碼，在 JPEG XR 編碼器中，熵編碼是整個編碼器的核心，這部份是數學運算最複雜的地方，所以我們第一個提出的技巧就是在熵編碼時，使用管線排程的方式來增加執行速度與產出量，另外再提出的架構就是在前置濾波器與 PCT 部份，為了最佳化這部份與充分的利用矽晶圓的單位面積，我們提出資料重複技術來解決此問題，此技術可以節省對外部記憶體存取的頻寬達 33%，其它還有很多的架構來節省面積與增加編碼效率。

我們藉由元件庫設計方式實做一顆 JPEG XR 編碼器，實際模擬結果，我們提出的架構可以執行每秒 34.1 百萬樣本的編碼器，這顆 IC 可以廣泛的應用在數位影像上，並且是低運算複雜度、低儲存空間與高動態範圍的一顆影像壓縮晶片。

Architecture Design and Implementation for JPEG XR Encoder

Student：Chien-Yen Ching　　Advisor：Dr. Sheng-Chieh Huang

Institute of Electrical and Control Engineering
National Chiao-Tung University

Abstract

With rapid progress of sensors, display devices, computing engines, and efficient algorithm/architecture, image application exists everywhere. In the past years, JPEG is well-known image compression standard. It has been merged together with our life such as digital still camera, blog and others. But JPEG can't satisfy the rapid progress of technology. For satification of the high quality image compression, the new JPEG XR compression algorithm is discussed and implemented with the VLSI architecture. JPEG XR has high encoding efficiency and versatile functions. The image quality of JPEG XR is nearly equal to JPEG 2000 with the same bit-rate.

The analysis and architecture design of JEPG XR encoder are also proposed in this thesis. Besides, the 4:4:4 high definition photo can be encoded in smooth. In JPEG XR encoder, Entropy coding is the heart of encoder. Therefore, we first proposed a timing schedule of pipeline architecture to speed up the entropy encoding, which is the most computationally intensive part in JPEG XR encoder. Another improved architecture in this work is the optimization of Pre-filter and PCT. To optimize this problem and maximize the silicon area efficiency, we also proposed a data reuse skill to solve this problem. The data reuse skill can reduce 33% memory bandwidth form external memory. There are many techniques to reduce the hardware cost under the same throughput.

A baseline JPEG XR encoder has been implemented by cell-based IC design flow. According to the simulation results, the throughput of the proposed design can encode 34.1 M samples/sec. This design can be used for the digital photography applications to achieve the low complexity of computation, low storage, and high dynamic range.

誌謝

首先感謝指導教授黃聖傑博士的指導，在這二年期間，給了我很大的空間可以盡情發揮創意，一路上提供寶貴的經驗與建議來幫助我走對的方向，並且強調以人為出發點，這讓我的受益匪淺。

接著感謝實驗室的士翔、煜傑、奕澄、雷峻，在平常討論與 Meeting 所給的建議，也一起度過在實驗室努力研究的日子，當然也有歡樂的日子，在學習路途上，感謝陳良基教授與簡韶逸教授的指導，讓我功力大增，特別感謝從一開始的維尼到 Peter 學長，在 Verilog Coding、硬體設計架構與實做上的指導，不辭辛勞的幫我解答，在 IC 設計領域上給了我一條明確的方向，並從零開始到完成一顆 IC，再來就是潘佳河學長，提供寶貴的實做經驗與研究遇到挫折時，幫我打氣，研究所生涯的 2 年期間，多謝有永洲學長、昭明、宗明、毅泓、暉鈞同學、茂樵、東欣學弟，平常在實驗室的砥礪與製造歡樂的氣氛，使得研究所生活變的多彩多姿，也感謝口試委員蘇朝琴、簡韶逸、連崇志、黃毓文在口試時的建議，並提供寶貴的經驗，讓本論文可以針對初稿做出校正並更加完整。

最後感謝父母親在背後的支持，讓我可以全力的衝刺，再一次的感謝所有幫助過我的人，進步的動力來自你們的幫助與建議。

簡清彥

2008/08/05

目錄

圖目錄

表目錄

第一章

緒論

1.1 動機

隨著科技的進步，可攜式裝置與無線網路的普及，人們對於影音娛樂方面的需求可是與日俱增，例如影音播放器與迅速竄起的網路部落格(Blog)，無時無刻都希望享受影像所帶來的樂趣或者將照片分享給朋友一起同樂，不知不覺中，影像已經成為另一種溝通語言，可以藉由影像的內容帶給人們喜怒哀樂，相對的，影像品質與檔案容量的要求也越來越高，無非希望能以最小的儲存空間存放大量高品質的相片，節省傳輸時間與儲存容量。

以目前感測器技術，如：液晶電視與相機的 CMOS 感測器對單色的解析度都已達 10 位元以上甚至更高，現在傳統的單色 8 位元表示法與影像壓縮技術已不合所需，如：JPEG，無法提供更豐富的色彩，而 JPEG XR 可以支援到單色 16、32 位元與浮點數壓縮，以滿足感測器所需，提供較高的動態範圍(High Dynamic Range, HDR)，從影像擷取端至壓縮演算法都不會失去任何資訊，即使放大影像倍率也無損畫質細膩的程度，消費者只要選擇適當的顯示器與色調轉換公式(Tone Mapping)即可享受高畫質的影像，且動態範圍的數值越高，對影像的長期保存也非常重要，微軟新興影像與視訊技術計畫經理 Robert Rossi 先生指出『再過五，六年後，印表機和顯示器會遠遠超過現行 JPEG 技術的處理能力』，所以這新世代的 JPEG XR 影像壓縮演算法會是未來的趨勢。

JPEG 組織(Joint Photographic Experts Group)在 1991 年推出了 JPEG 影像壓縮演算法，這讓我們的生活發生了重大的改變，影像在網際網路的快速傳輸，數位相機的儲存媒介到現在很熱門的數位像框都大量採用 JPEG，在 2000 年 JPEG 組織定了 JPEG2000 影像壓縮演算法，所提供的影像品質更好且包含許多編碼技巧，例如：Quality Scalability 與 Resolution Scalability 等等，都可以增加影像的壓縮率，但也大幅度的增加演算法複雜度，使得 JPEG2000 一直在市場上不受到青睞，所以新世代 JPEG XR 影像壓縮演算法將被提出來改善現有的缺點。

實現 JPEG XR 演算法的平台當然有很多種，例如：使用中央處理器(Central Processing Unit, CPU)、DSP(Digital Signal Processor)、ARM(Advanced RISC Machine)或積體電路 (Integrated Circuit, IC)完成，每種實現方式各有優缺點，考慮到市場上的數位相機、掃瞄器與手機等等，不外乎都是以輕巧、小、便宜為主要考量，所以我們選擇單晶片為平台來實現 JPEG XR 影像壓縮演算法，過去幾年，隨著電晶體製程的快速發展，相同單位面積下可以擁有更多的電晶體，這項特性能幫助我們達到低功率、低成本與高效能的晶片，實現成晶片的另一個好處在於可包裝為矽智財(Silicon IP)，掛在系統(System)上，這是為了考量晶片的重覆使用性，對於影像壓縮演算法當然可以使用在各種不同地方，例如：相機、掃瞄器與監視系統等，都可利用矽智財來達成，進一步降低系統成本。

我們將會提出多種新的架構與設計技巧，分析演算法發現，此壓縮技術以區塊(Block)為基礎，使用大量的濾波器、適應性編碼等技巧來提昇影像的品質與壓縮率，造成編碼器的複雜度大大增加，所以首要目標就是減少演算法的複雜度，由硬體電路來實現，並適度的管線化與整體最佳化，提升效能，再者就考慮晶片使用的資源，加入資料重複使用技術(Data Reuse)，減少記憶體存取次數與頻寬使用量，使用較少的記憶體完成編碼，壓縮過程中完全不使用浮點數運算，減少硬體資源與功率，對可攜式裝置可是一大福音，實際考慮市面上電子產品的需求。

1.2 JPEG XR

JPEG XR 又稱 HD-Photo 與 Windows Media Photo(WMP)，在 2006 年 11 月由微軟(Microsoft)提出的影像壓縮演算法[1][2][3][4]，希望取代 JPEG[5]與 JPEG2000[6][7]，並且於 2007 年 10 月送交聯合影像專家組織，正式稱作 JPEG XR，XR 的意思為『延伸範圍』(Extended Range)，希望成為新一代的影像標準，此算法的特性在於支援多種影像格式，如：RGB、CMYK、Grayscale、n-Channel 等，並使用 scRGB 色域(Color gamut)空間，其所包含之顏色比 JPEG 所使用的 sRGB 色域空間豐富，所以影像經過編解碼之後，依然保持最真實的色彩，符合高品質影像要求，圖 1.1 是個例子，說明廣色域的好處，在黃色氣球與人臉膚色部份就可以明顯看出之間的差別，圖 1.2 展示出 scRGB 的色域空間已經大於人眼所能感測的範圍，也比 JPEG 所使用的 sRGB 還大。

圖 1.1：JPEG 與 JPEG XR 的比較

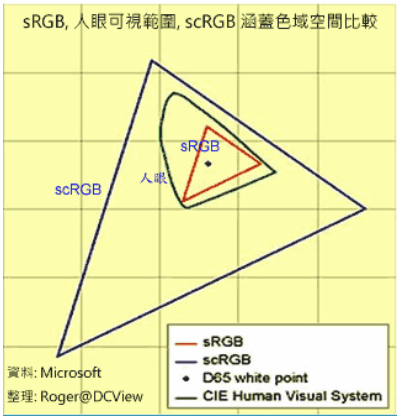


圖 1.2：scRGB 與 sRGB 比較

在 1.1 節的地方有提到，在此之前已經有 JPEG 與 JPEG 2000 兩套影像壓縮演算法，而 JPEG 2000 所提供的影像品質明顯優於 JPEG，但卻不受到市場青睞，原因在於演算法複雜度太高，實做成晶片後的成本太高，專利方面也被獨佔，科技公司生產 JPEG 2000 晶片的花費可觀，且在高壓縮率時，人眼才會覺得 JPEG 2000 所提供的影像品質明顯優於 JPEG，種種原因使得 JPEG2000 不受市場歡迎，

所以新世代 JPEG XR 當然要突破此困境，根據目前消息是免費授權的，在失真(Lossy)與無失真(Lossless)壓縮演算法都是採用同一套演算法，這對可攜性電子產品可是一大福音，不需要花費二倍的硬體成本，即可提供二種演算法，壓縮率與躁訊比(Peak Signal-to-Noise Ratio, PSNR)方面，JPEG XR 可以在相同的影像品質下提供比 JPEG 高一倍的壓縮率，有效降低檔案傳輸時間，如圖 1.3 的全彩影像與圖 1.4、圖 1.5 與圖 1.6 的灰階影像，JPEG XR 能提供與 JPEG 2000 近似的影像品質，PSNR 的差距約在 0.5dB 以內，演算法複雜度卻大幅減少，非常適合在低耗電或可攜式的電子產品上，我們根據 JASPER 與微軟所提供的標準軟體進行演算法複雜度分析，測試的項目為壓縮一張 Lena 512x512 的全彩影像，比較單位為中央處理器所花費的機械週期(Machine Cycle) ， 測試的結果如圖 1.7 所示，JPEG XR 所花費的機械週期明顯小於 JPEG 2000，大約為 3.5 倍，此結果更驗證了 JPEG XR 將會是以後影像壓縮的霸主。

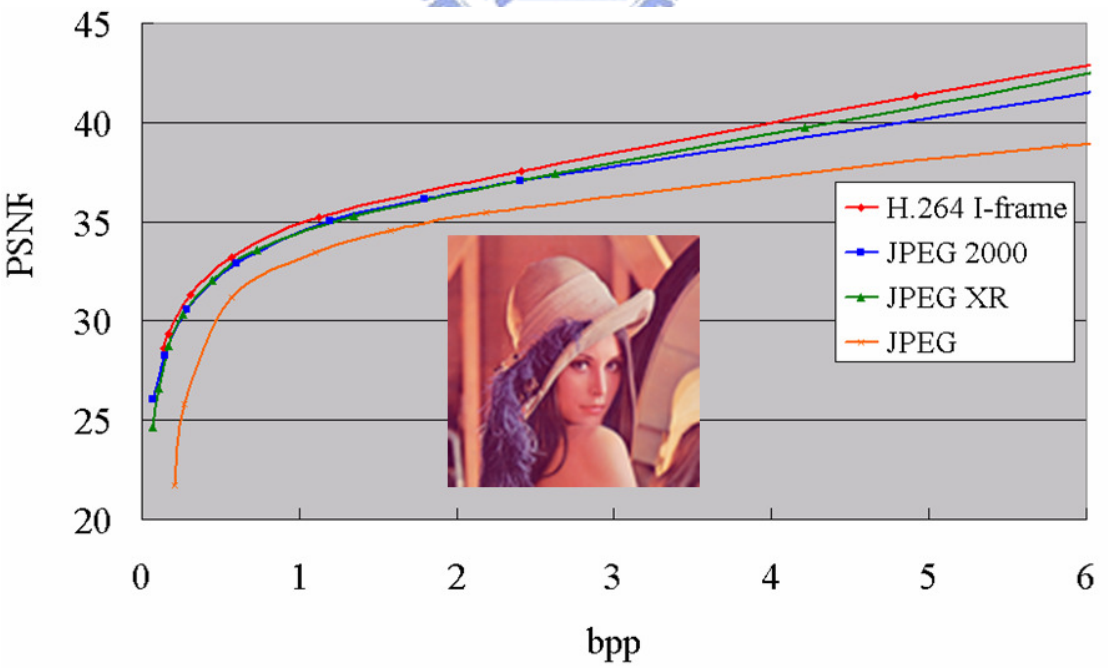
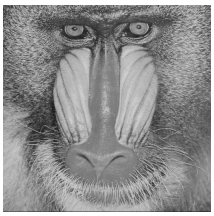


圖 1.3：PSNR 比較



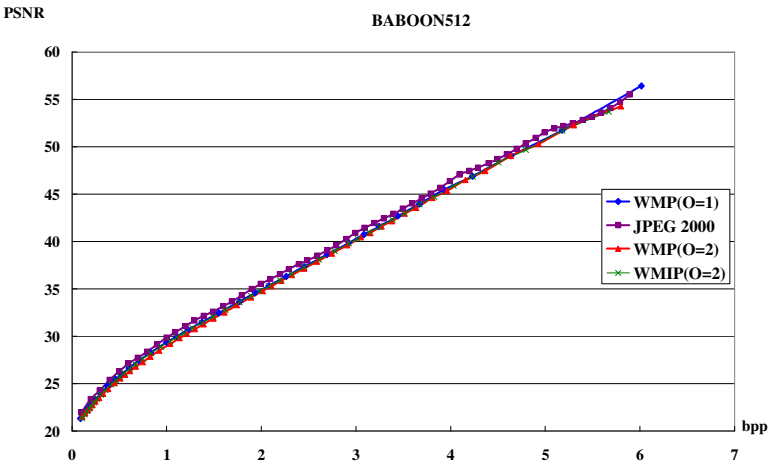


圖 1.4：PSNR 比較

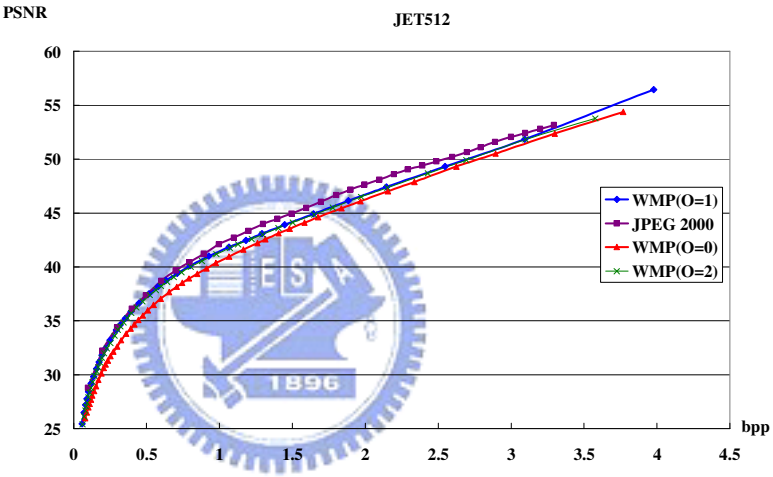


圖 1.5：PSNR 比較

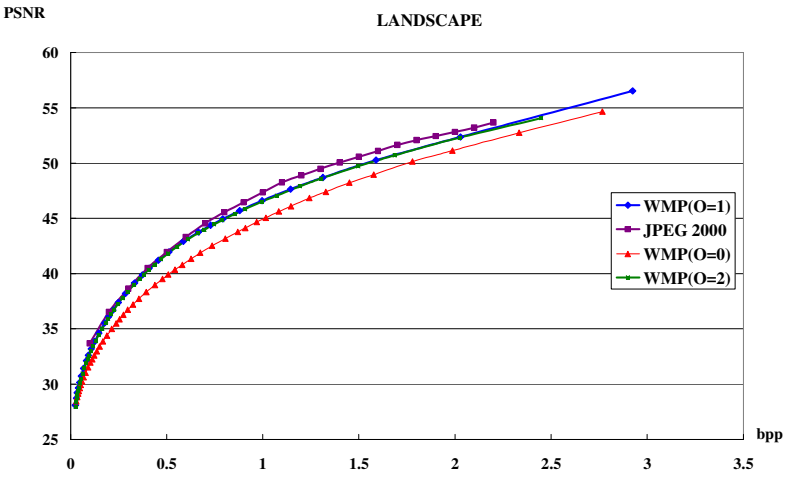


圖 1.6：PSNR 比較

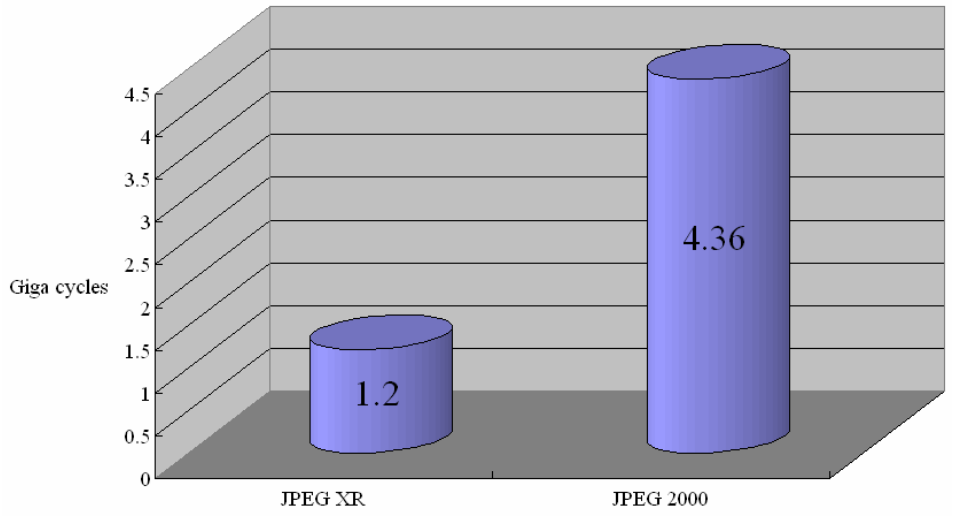


圖 1.7：演算法複雜度比較

圖 1.8、圖 1.9 與圖 1.10 的視覺直觀比較就可以清楚分辨好與壞，此外，JPEG XR 在無失真壓縮的表現也是同樣亮眼，能高達 2.5 倍的壓縮率，其它的功能還包括影像能隨意的旋轉、任意的調整解析度。

(a) (b)

圖 1.8：在相同壓縮率 60 倍的比較結果 (a)JPEG XR (b)JPEG 表現結果

(a) (b)

(c) (d)

圖 1.9：60 倍壓縮率的比較結果 (a)原始影像 (b)JPEG

(c)JPEG XR (d)JPEG2000

(a) (b)

圖 1.10：在相同壓縮率 80 倍的比較結果 (a)JPEG XR (b)JPEG 表現結果

目前微軟在 Vista 作業系統上有內建 JPEG XR 的影像格式，Windows XP 只需下載 Framewok 3.0 即可使用，還有 Adobe 公司的 Photoshop 都已經支援此格式，在工業界方面，微軟也積極與廠商合作開發相關規格之產品，如：高階數位相機製造商 Hasselblad 與 Canon、相機影像處理 IC 設計公司 NOVATEK、SUNPLUS、ARM 等等大廠都已經和微軟共組聯盟，共同推動新世代的影像格式。

JPEG XR 挾帶了強大的技術實力，並受到各種系統規格的積極採用，系統廠也馬上推出了適用於各種應用場合的解決方案，當然於對 IC 設計公司來說，這也代表了全球性的市場、與全球性的機會。尤其在解碼器方面，因為多媒體撥放器的市場，往往會比多媒體錄製器的需求大的很多，所以解碼晶片，更是各大廠兵家必爭之地。

對於以晶片設計起家的台灣，本論文希望提出一項新穎有效率的架構，推出台灣的第一項高解析度 JPEG XR 編碼器的硬體解決方案，幫助台灣的廠商切入這塊市場，在全世界的多媒體產品中占有一席之地。

1.3 論文架構

本論文共分為五個章節，在本章節說明 JPEG XR 的實做動機與簡介，下個章節，更深入的演算法介紹與分析將會一一說明，第三章即是本論文的核心部份 - 硬體架構設計，我們將會分析硬體設計上遭遇的困難與解決方法，並提出多個硬體架構來實現，在第四章會呈現實做與模擬結果，最後一章進行結論。

第二章

JPEG XR 影像壓縮演算法

2.1 演算法概要

JPEG XR 壓縮演算法採用整數轉換(Integer Transform)的方式完成，沒有複雜的數學運算，不使用小數點演算法，在硬體設計上也就不需要考慮到小數點與誤差問題，在熵編碼的部份不使用算數編碼(Arithmetic Entropy Coding)，如此即可增加編碼速度，硬體方面也可以減少資源，非常適合可攜性電子產品的特性，還有其它如：編碼過程中，以區塊(Block)為基礎的處理方式，只需要少量記憶體即可完成，編碼器與解碼器的架構是對稱的，使得硬體資源可共享，進而達到高效能、低功率的演算法。

2.1.1 串流編碼架構(Bitstream Architecture)

首先在編碼影像時，先將影像如圖 2.1 所示可由使用者決定切分為多個 Tile，每一個 Tile 都可以視為一張獨立影像進行編碼，好處就在當傳輸時發生位元錯誤時，不至於影響到整張影像，只影響受到損害的那個 Tile，根據研究與實驗結果如表 2.1，使用的測試影像大小為 512x512，不同的 Tile 大小會影響到壓縮率，在相同的量化值(Q)下發現不使用 Tile 的壓縮率最高，相對的，Tile 切的越多則壓縮率越小，但在傳輸上所要承受的錯誤風險也越小，所以在 Tile 的切割上、傳輸環境與壓縮倍率間必須做權衡(Trade Off)。

表 2.1：Tile 與壓縮率之間的比較

Q = 5	No tiles	8x8	4x4	1x32	1x1
Lena	344 K	346 K	350 K	345 K	433 K
Baboon	484 K	490 K	504 K	497 K	704 K
Airplane	274 K	276 K	279 K	279 K	332 K
Pepper	387 K	390 K	397 K	392 K	501 K
Q = 70	**No tiles**	**8x8**	**4x4**	**1x32**	**1x1**
Lena	8.34 K	9.55 K	12.6 K	10.9 K	38.9 K
Baboon	31.8 K	32.8 K	35.7 K	34.3 K	60.7 K
Airplane	9.27 K	10.5 K	13.8 K	12.3 K	39.6 K
Pepper	10.4 K	11.6 K	14.6 K	12.8 K	39.7 K

Tile 裡再切分多個大區塊(Macroblock)，最後每個大區塊再切分為 16 塊區塊，而編碼器在編碼時，就是以大區塊為單位，由左而右，由上而下按照順序進行編碼，JPEG XR 另一個特別的地方就是在串流編碼(Bitstream Encode)時，能夠使用空間模式(Spatial Mode)或頻域模式(Frequency Mode)，如圖 2.2，空間模式是將每一個 Tile 內的大區塊依照順序排列輸出即可，頻域模式則是將 Tile 內所有大區塊經過頻域轉換後的係數(Coefficient)，依照頻率的高低排列輸出，好處就是在有限的頻寬下，接收端可以先解出直流係數(DC Coefficient)，代表接收端可以顯示出影像的大約輪廓，使用者可以約略知道圖像的內容，進而選擇這張圖片是否為自己想觀看的，是的話就繼續傳輸，隨著資料接收的愈多，圖片的品質也越好，類似 JPEG 的漸進式模式(Progressive Mode)。

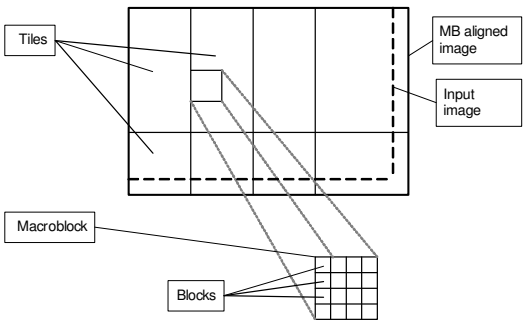


圖 2.1：影像切割示意圖

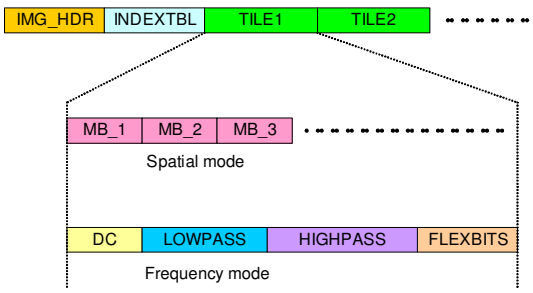


圖 2.2：串流編碼架構

演算法流程如下圖 2.3 所示，由於壓縮過程中使用取樣(Sampling)演算法會造成部份資料的流失，Pre-Scaling 的作用即為調整資料大小，增加表示範圍，在經過後面演算法的處理時，才不至於損失更多的原始資訊，在色域上的分析，傳統的紅色、藍色、綠色中，不容易看出之間的關連，經過研究後發現，其實眼睛對亮度的敏感度比彩度還要強，所以色相轉換(Color Conversion)的目的即是為了符合人眼特性，JPEG XR 自行設計一套轉換公式，目的都是為了減少 RGB 顏色在人類視覺上的關連性，由於 JPEG XR 在色相轉換之前有動態調整過資料的表示範圍，所以在此會再將亮度調整一次表示範圍，以利後續處理，轉換後得到的資訊就可以進行下一部的分析。

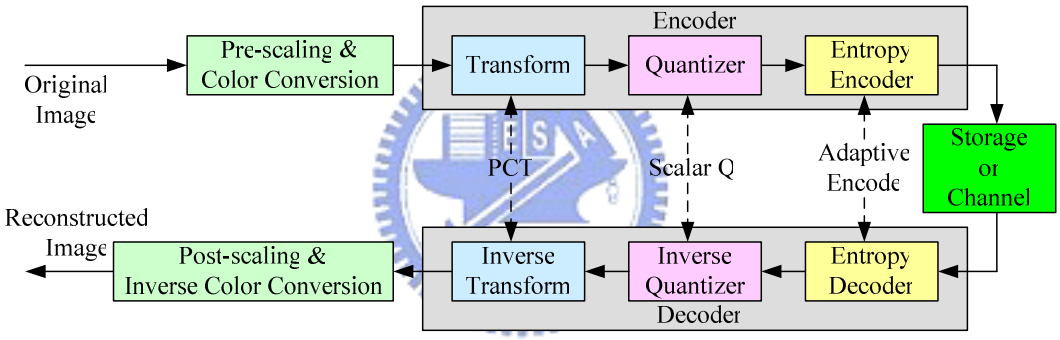


圖 2.3：編解碼流程圖

如之前所敘述的，人眼對彩度的敏感度比較低，所以這部份也是影像壓縮的目標之一，主要取樣格式分為 4：4：4、4：2：2 與 4：2：0，第一種 4：4：4 格式是不取樣，完全保留資料進行後續編碼，所能提供的影像品質最高，但本身的資料量也最大，後面二者分別對彩度進行二分之一與四分之一取樣，目的都是減少處理彩度的資料量，達到高效能低耗電的演算法，並且維持一定的影像品質，以符合人類視覺系統(Human Visual System, HVS)。

整個空間上的資訊的關連可以藉由頻率域的分析來判斷，例如常用的傅立業轉換(Fourier Transform)即是很好的分析工具，可以得到頻譜的分佈來選擇需要的資訊，另一項研究發現人眼對高頻信號的敏感度沒低頻信號強，此一特性就可以再利用以減少空間資訊上的累贅，進一步減少資料量，但缺點就是這類演算法的運算量太大，造成運算時間與硬體資源消耗資源過多，為了改善這缺點，JPEG XR就採用整數運算，並且有相同效果。

最後即是使用熵編碼(Entropy Coding)，也稱亂度編碼，利用霍夫曼(Huffman)的機率統計方式提供無損式的影像編碼，減少資料的儲存空間，利用出現機率較頻繁者，在編碼時則給予較短的唯一碼字(Codeword)，相反的，出現機率低的係數，則給予較長的碼字，達到資料壓縮的目的。

我們將上面所論述的演算法串起來，做個硬體方塊流程圖的整理，如圖2.4，演算法的執行順序與 JPEG、JPEG2000 差不多，深入的演算法探討就在下章節論述。

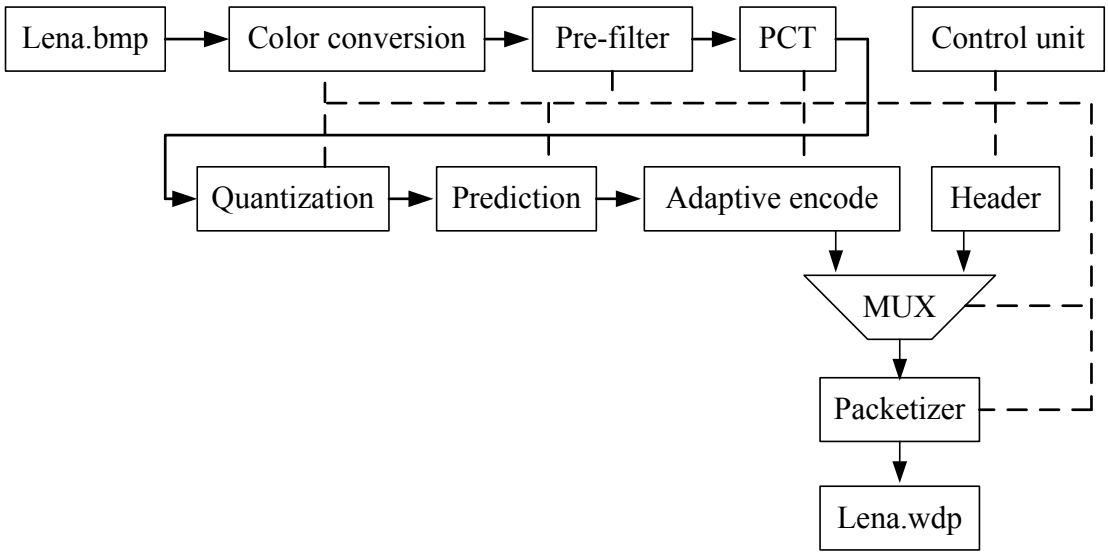


圖 2.4：詳細的編碼流程

2.1.2 檔頭(Header)

JPEG XR 的檔頭採用 TIFF 格式來實現，TIFF 是以標籤為結構的圖檔格式，最早是由 Aldus Corporation 的 Aldus Developer's Desk 公佈，目前版本為 6.0，好處是可以靈活運用內部的空間儲存資料，且可相容於各種平台與作業系統，TIFF 可分為三大部份：IFH(Image File Header)、IFD(Image File Directory)與資料本體，底下就繼續詳述。

IFH

IFH 共有 8 個位元組組成，由圖 2.5 可知在位子 0～1 是紀錄 0x4949，這表示此架構採用 Little-Endian 的表示式，位子 2～3 紀錄目前 JPEG XR 的版本，最後 4 個位元組紀錄第一個 IFD 相對於 IFH 的位子。

IFD

IFD 是由多個 IFD Entry 組成並可以紀錄有關輔助影像的資料，架構的最前面 2 個位元組紀錄總共有多少個 Entry，經由 IFD Entry 的 Tag 即可知道影像的高度、寬度…等，接著 Type 與 Count 分別紀錄所用的資料型態與資料的大小，最後才是存放資料或者是大量的資料時，就必須存放記憶體位子以供系統去找尋相對應的資料，由於 Tag 可以不斷擴充，所以 TIFF 格式的彈性可是潛力無窮。

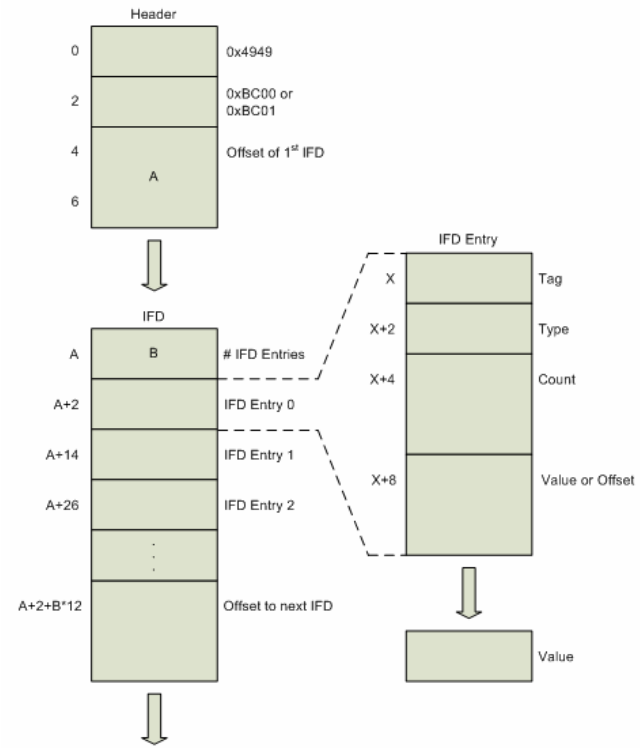


圖 2.5：檔頭架構

2.2 JPEG XR 演算法

在此章節，我們將詳細的分析 JPEG XR 演算法與其它影像壓縮標準比較。

2.2.1 色相轉換(Color Conversion)

色相轉換的目的即是反映人眼對於訊號的接收方式，從眼睛外之物體來的光在視網膜上成像，形成了圖案視覺，人眼的光受體有 2 類，錐狀體(Cones)和桿狀體(Rods)，每隻眼睛的錐狀體數量大約有 600 萬到 700 萬間，主要分佈於視網膜中央凹附近，並且對色彩很敏感，但必需要在光線足夠的地方才有功用，錐狀細胞又稱白晝視覺或亮光視覺。桿狀細胞的數目就比錐狀細胞多很多了，大約有 7500 萬到 15000 萬，分佈在視網膜表面，功能適用來獲得視野中大體上的影像，對亮度相當敏感，例如白天看起來很鮮豔的花朵，在月光下看起來卻沒有色彩，這是因為只有桿狀體受到了刺激，此現象稱為夜視覺或昏暗視覺。

經過上述討論，我們可以使用人眼對亮度比較敏感的特性來設計多種色相轉換公式，JPEG XR 採用的是由紅、綠、藍色組成的 RGB 色彩空間轉換成 YUV，Y 即是亮度，存放著大量對人眼相當敏感的資訊，U、V 負責存放彩度的資訊，比較特別的是 JPEG XR 不採用傳統的 RGB 轉 YUV 公式，原因在於傳統的轉換公式是將 RGB 係數乘上有小數點的係數，這在硬體實做上會花費較多的成本且會有精準度的問題，所以 JPEG XR 就設計了一套整數運算的轉換公式，硬體設計上只需要加法器與移位器即可完成且此轉換過程是可逆的，可同時支援失真壓縮與無失真壓縮。

$$V = B - R$$

$$U = -\left[R - G + \left\lfloor \frac{V}{2} \right\rfloor \right]$$

$$Y = G + \left\lfloor \frac{U}{2} \right\rfloor + offest$$

底下，我們引用[7]書上的一個例子來實際說明色相轉換的功用，影像在 RGB 色域上的關連性相當低，或者說有相當高的累贅，經過轉換後即可看出影像在 Y 有大量的能量集中。

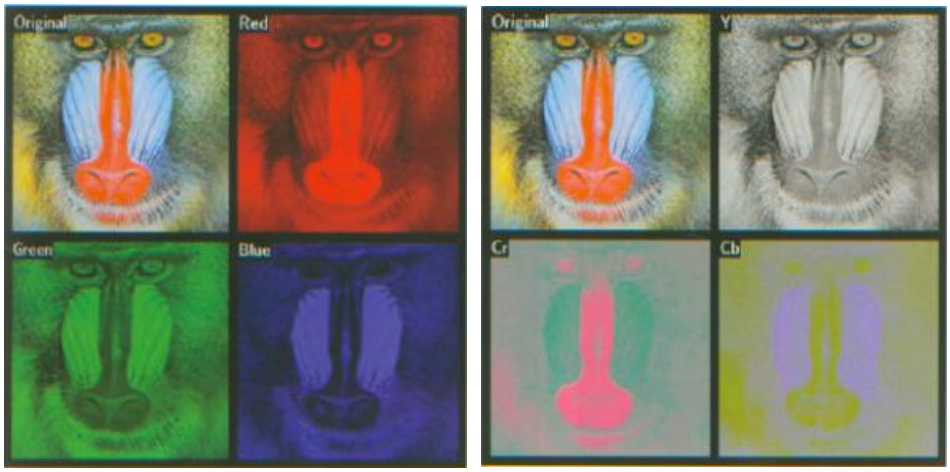
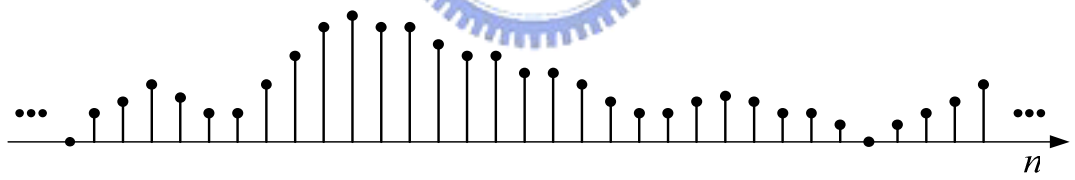


圖 2.6：RGB 色域空間轉換為 YCrCb 色域空間

2.2.2 取樣(Sampling)

取樣最大的好處在於可以減少資料量，那影像的資料如何減少？多虧色相轉換，將原本毫無關係的 RGB 資料轉換為 YUV 資料，由上述例子很明顯的看出重要的資訊與能量都集中在 Y 上，而 Cr 與 Cb 就顯得不是那麼重要，這正是我們要處理的對象，JPEG XR 採用濾波器的方式來進行取樣的動作，可以分為 4：2：2 取樣，意思是說因為 Y 的部份因為聚集了大量有關影像的資訊與能量，所以不進行取樣，而 Cr 與 Cb 在空間域上的資料只取一半，達到資料壓縮的目的，取法如圖 2.7 所示，將影像的某一列(Row)以直方圖畫出， n 代表每一個像素且具有不同的能量大小，進行取樣演算法時，採用寬度為 5 像素的濾波器來運算，中間位子的權重(Weight)最大，左右兩邊次之，最外面的係數權重最低，之所以這樣設計的原因在於取樣時，除了參考本身位子上的係數值，還去考慮左右兩邊的係數變化來取得一個適當的運算結果，另外在硬體實做上可以發現，整體取樣演算法只需要加法器與移位器即可完成，可以大幅降低硬體複雜度。



$$y[n]=\left\{\left[\left(x[n-1]+x[n]+[n+1]\right)\times 4\right]+\left(x[n]\times 2\right)+x[n-2]+[n+2]+8\right\}\div 16$$

圖 2.7：取樣演算法

整張影像處理完後，依使用者需求可再進行一次取樣，稱為 4：2：0，U 與 V 的資料量再取一半，為原來的四分之一，整體資料量更小但也會犧牲影像的品質，取樣法則是接續 4：2：2 後的處理結果，將每一行(Column)再次進行上圖的公式即可完成取樣演算法。

接著我們比較一下經過取樣後的模擬結果，圖 2.8 可以清楚的發現不取樣的 PSNR 值是最高的，但在非常高的壓縮率時，大部分的係數都被量化成零，所以有沒有進行取樣是不會有差別的。

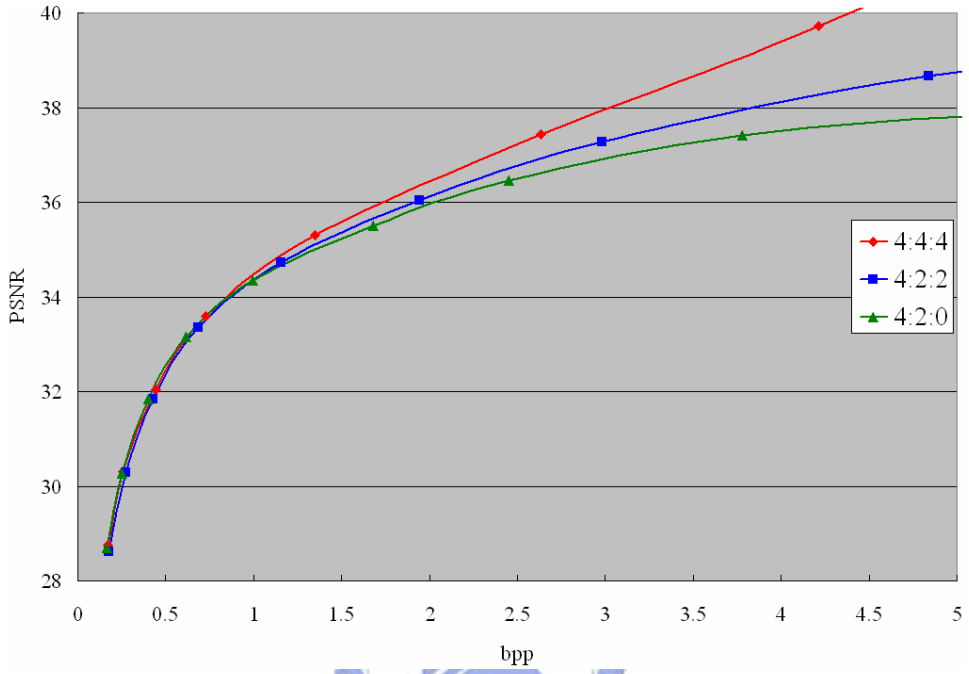
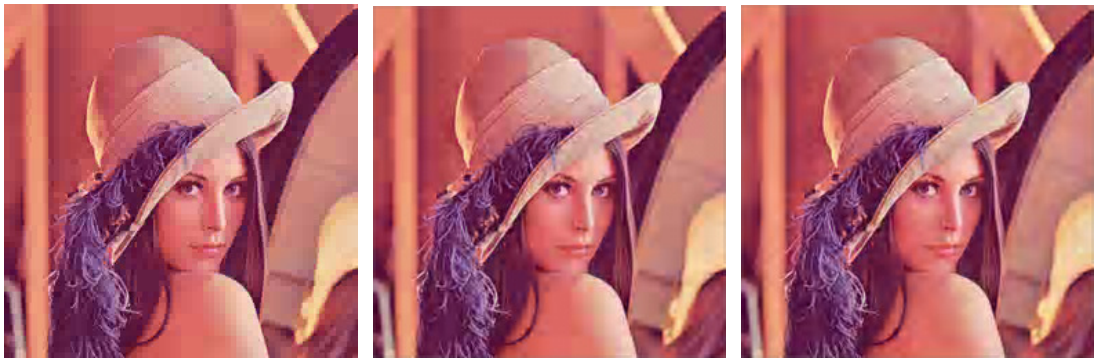


圖 2.8：取樣演算法的 PSNR 比較

2.2.3 前置濾波器(Pre-filter)

影像處理上都是以區塊為基礎的處理方式，此方法雖然在編碼上有一定的效率，但是也會產生不自然的區塊效應(Blocking-Effect)，在影像邊界處會造成鋸齒狀的結果，尤其在壓縮率越高時會越明顯，根據人類視覺系統發現，圖像的邊受到了損毀或嚴重失真，則人眼所看到的影像品質會大幅降低，所以為了解決這問題，前置濾波器也就孕育而生，目的即是為了解決圖像邊緣因以區塊為準之轉換量化（Block-Based Transform & Quantization）所造成之不連續現象，同時保存原有物體邊緣之銳利度，濾波器也是以區塊為單位進行處理，在影像邊界上的資料則另外處理，整個 JPEG XR 有二層前置濾波器，其用意在低壓縮率時，影像本身的品質就不錯，可以選擇不進行濾波器的處理，節省運算時間，中等壓縮率時

就採用一層濾波器，高壓縮率則進行二層濾波器處理，犧牲運算時間來獲取較好的影像品質，此外，濾波器還可以增加壓縮率，根據模擬結果，在低壓縮比時可增加壓縮率達 30%，圖 2.9 與圖 2.10 分別提供了視覺上的比較與 PSNR 的比較。



(a) (b) (c)

圖 2.9：前置濾波器在高壓縮比時的視覺比較；(a)不使用濾波器(L=0)， (b)使用一層濾波器(L=1)， (c)使用二層濾波器(L=2)

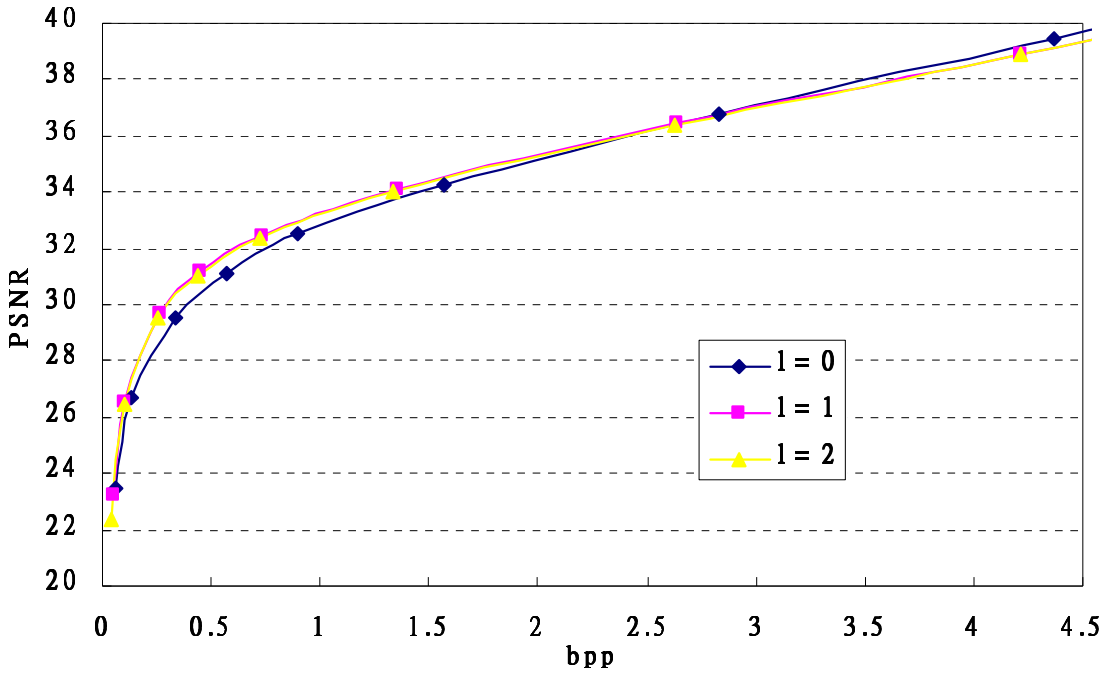


圖 2.10：前置濾波器的 PSNR 比較圖

前置濾波器在處理的地方剛好是區塊的邊界，這部份也是人眼最容易看出區塊間的不連續的地方，會產生的原因當然是量化所造成的，如圖 2.11，每個區塊很明顯的不連續，以頻率域的觀點來看，原因出在直流係數的位準差太多，所以前置濾波器在處理時就將所有係數轉換成頻率域，分成低頻、中頻與高頻，再去針對不同頻段進行處理，例如對區塊邊界的低頻與中頻使用濾波器進行調整，使得邊界係數的落差不至於太大，最後再轉換回空間域的資料，即可完成前置濾波器的工作。

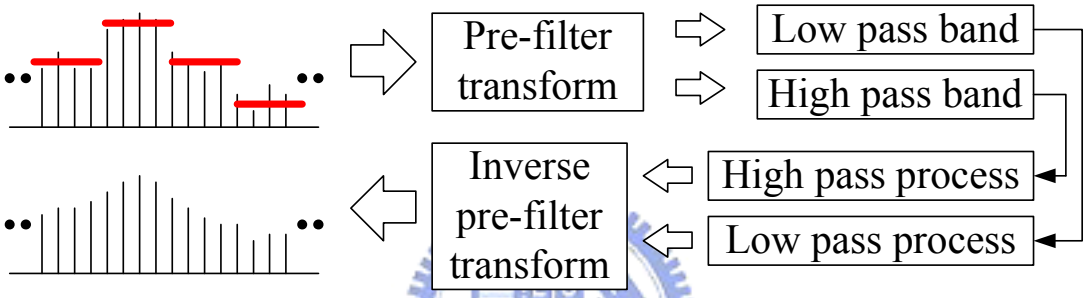


圖 2.11：前置濾波器的設計構想

2.2.4 Photo Core Transform(PCT)

PCT 即是整體 JPEG XR 壓縮法的核心，有別於 JPEG 的離散餘弦轉換(Discrete Cosine Transform, DCT)，而是採用類似 JPEG 2000 的離散小波轉換(Discrete Wavelet Transform, DWT)架構，進行多層次的運算，過濾出高頻、低頻與直流係數，以利後續的處理，將大區塊切成 16 個 4x4 的區塊分別進行第一階段(First Part)的轉換，每個區塊都產生一個直流係數與 15 個高頻係數，當 16 個區塊 都處理完成後，在將 16 個區塊的直流係數分別擷取出來再做一次相同的演算法，如圖 2.12，藍色方塊即代表每個區塊的直流係數，再做一次 PCT 運算，最後每個大區塊會得到 1 個直流(DC)係數、15 個低頻(Low Pass)係數與 240 個高頻(High Pass)係數，為了減化 PCT 轉換的複雜度，所以全部的轉換演算法都採用整數運算的方式處理，避免浮點數對硬體帶來的困擾。

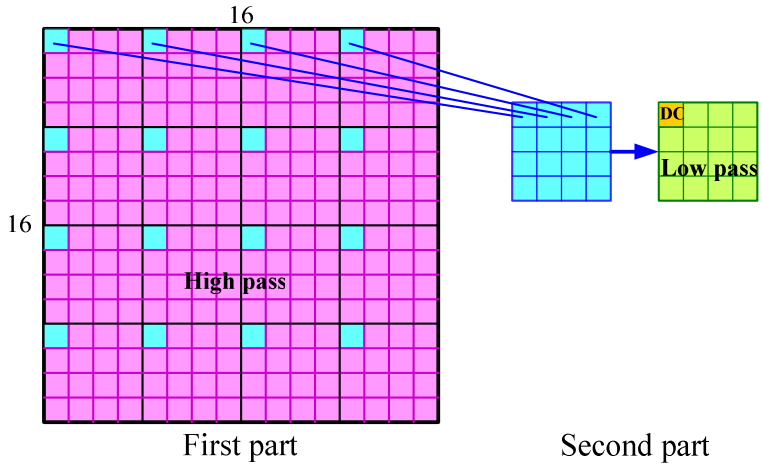


圖 2.12：PCT 轉換架構

圖 2.13 顯示出 PCT 是由多個 Hardmard Transform 演算法組成，區塊在處理時分為四次進行處理，使用 T_h 演算法將區塊分成 LL、LH、HL 和 HH 頻段並且將低頻係數集中於左上角，接著再使用 T_h、T_{odd} 與 T_{odd_odd} 演算法處理即可完成 PCT 轉換，表 2.2 以 C code 說明 PCT 的子演算法，可以發現三段程式都使用整數運算且沒有除法計算過程，實際在硬體實做上只需要加法器與移位器即可完成，相對於 JPEG 與 JPEG 2000 都必須考慮到小數、乘法與誤差等問題，這部份可是 JPEG XR 的一大優勢，圖 2.14 說明 PCT 轉換的運算過程。

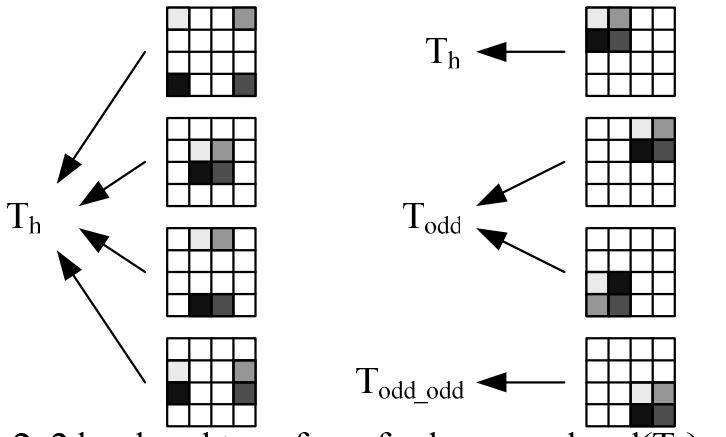


圖 2.13：PCT 轉換方式

表 2.2：PCT 內部演算法

```
T_h(int &a, int &b, int &c, int &d, int &R) {
  a += d;
  b -= c;
  int t1 = (a  -  b + R) >> 1;
  int t2 = c;

  c = t1 - d;
  d = t1  -  t2;
  a -= d;
  b += c;
}
```

```
T_odd(int &a, int &b, int &c, int &d) {
  b -= c;
  a += d;
  c += (b + 1) >> 1;
  d = ((a + 1) >> 1)  -  d;

  b -= (3*a + 4) >> 3;
  a += (3*b + 4) >> 3;
  d -= (3*c + 4) >> 3;
  c += (3*d + 4) >>3;

  d += b >> 1;
  c -= (a + 1) >> 1;
  b -= d;
  a += c;
}
```

```
T_odd_odd(int &a, int &b, int &c, int &d) {
  int t1, t2;
  b = -b;
  c = -c;

  d += a;
  c -= b;
  a -= (t1 = d >> 1);
  b += (t2 = c >> 1);

  a += (b*3 + 4) >> 3;
  b += (b*3 + 3) >> 2;
  a += (b*3 + 3) >> 3;

  b -= t2;
  a += t1;
  c += b;
  d -= a;
}
```

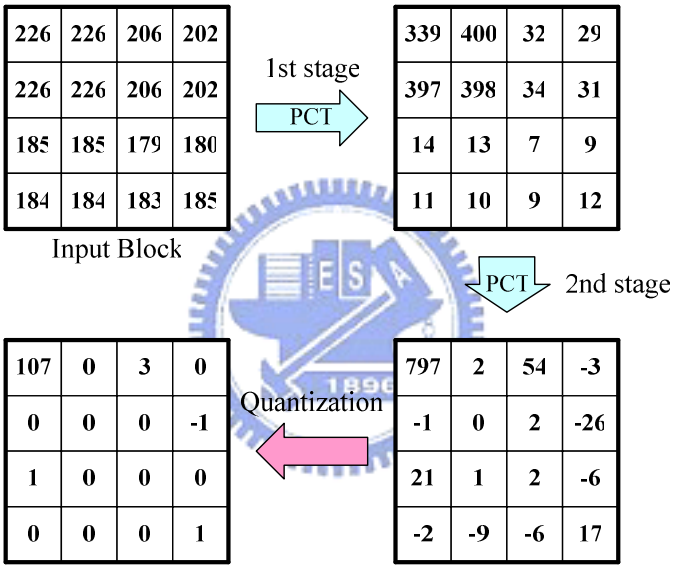


圖 2.14：PCT 轉換的運算過程

2.2.5 量化(Quantization)

根據人類視覺系統研究發現眼睛在一個相對大範圍區域，辨別亮度上細微差異是相當的好，但是在一個高頻率亮度變動之確切強度的分辨上，卻不是如此地好。這個事實能在高頻率成份上極佳地降低資訊的數量。簡單地把頻率領域上每個成份，除以一個對於該成份的常數就可完成，且接著捨位取最接近的整數。這是整個過程中的主要失真運算。以這個結果而言，經常會把很多更高頻率的成份捨位成為接近零，且剩下很多會變成小的正或負數。

JPEG XR 採用以分頻的方式來處理量化過程，主要分成直流、低頻與高頻段，每個頻段都除相同的係數即可完成，而非 JPEG 的量化方式，以減少演算法與硬體設計的複雜度。

2.2.6 資料預測(Prediction)

為了減少係數在前後筆之間的關係，所以就必須做個資料預測，類似 JPEG 的誤差脈衝編碼調變(Differential Pulse Code Modulation, DPCM)，JPEG XR 不採用每個係數間的運算，而是以區塊為單位進行資料預測並且分成直流、低頻與高頻係數進行運算，被預測的資料則視本身資料與周圍資料間的關係由特殊演算法決定要參考誰，以確保經過預測後的資料是最精簡的，達到影像壓縮的目的。

直流係數的預測如下列演算法所示，依照目前處理的大區塊所在位子，會有不同的預測方式，例如圖 2.15：大區塊的位子在粉紅色的區域上，即是處理整張影像的第一個大區塊，無任何資料可以預測，所以整個大區塊的資料將不進行預測，當分別處理到藍色與綠色區域的大區塊時，所能參考的資料只有左方與上方的大區塊，所以進行特定方向的預測，黃色區域代表目前的大區塊有左方與上方的資料，則由演算法來決定往哪邊進行預測。

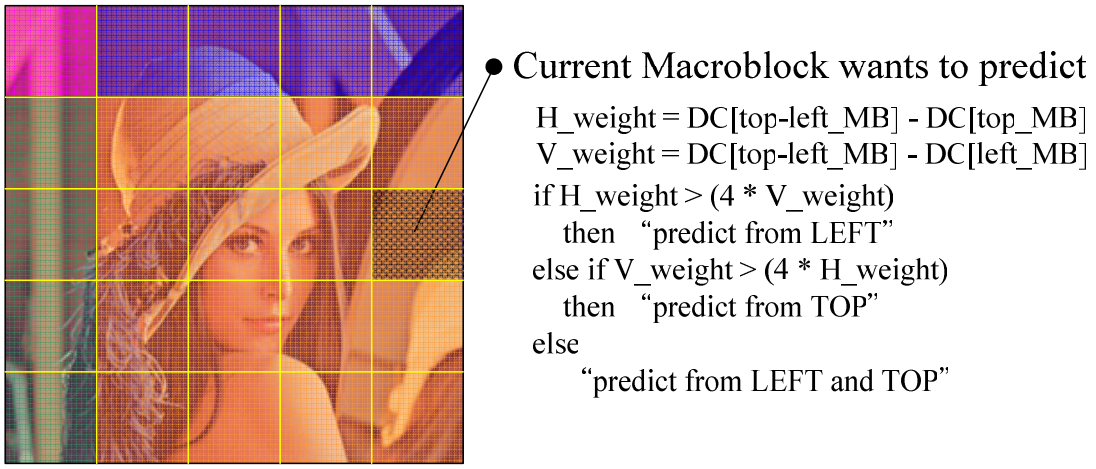


圖 2.15：DC 值的預測演算法

圖 2.16 說明假如 DC 係數的預測方向是本身大區塊來判斷與使用上述演算法來判斷的差別，最主要的差別在於上述演算法在串流編碼時，不需要記錄預測方向，節省二位元的儲存空間，比較影像在 33db 時，可增加 3.9%的編碼效率。

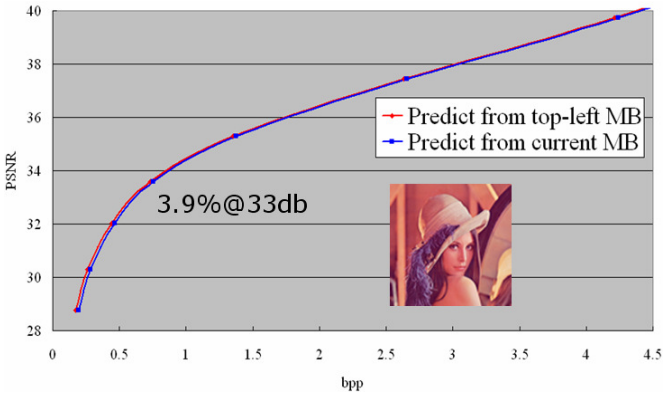


圖 2.16：DC 係數的預測演算法

低頻係數也稱為 AD 係數，它的預測方向是跟隨直流係數的，圖 2.17 說明了只有往左或往上參考，每次進行運算時，都只有三個係數會去進行運算，這原因是 PCT 在轉換時就經過設計，認為一個小區塊只需要三個係數進行運算即可，以減少系統的運算量與複雜度，在硬體實做上，由於目前的 AD 係數要往上方預測時，就必須將上方的所有 AD 係數存起來，但 JPEG XR 只有四個係數要運算而已，所以需要儲存的資料只有原來的四分之一，降低硬體成本。

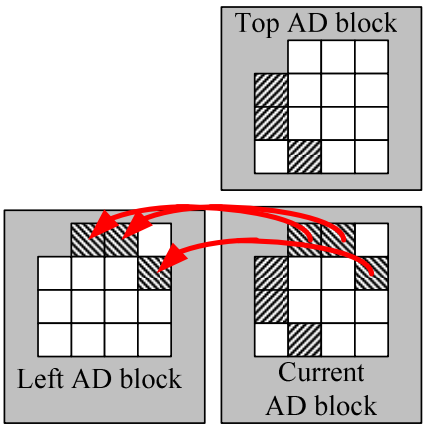


圖 2.17：AD 係數的預測演算法

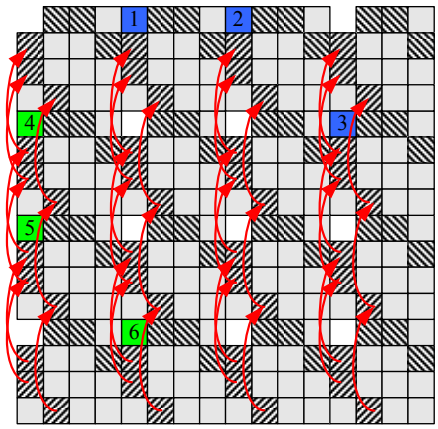


圖 2.18：AC 係數的預測演算法

高頻係數稱為 AC 係數，圖 2.18 是個例子用來說明預測方式，空格代表示低頻係數的所在位子，預測演算法執行的方式跟 AD 係數的方式一樣，每個小區塊都只有三個係數會去計算，往左邊或上方的區塊來參考進行預測，目的還是希望能減少運算量與執行時間，AC 係數決定預測方向的演算法可見表 2.3。

表 2.3：AC 決定預測方向的演算法

```
PSEUDO CODE:
  Hozi_weight = abs(lowpass_Y(1)) + abs(lowpass_Y(2) + abs(lowpass_Y(3))
                abs(lowpass_U(2)) + abs(lowpass_V(2));
  Verti_weight = abs(lowpass_Y(4)) + abs(lowpass_Y(5) + abs(lowpass_Y(6))
                abs(lowpass_U(5)) + abs(lowpass_V(5));
  if (4 * Hozi_weight < Verti_weight)
    then  "predict from LEFT"
  else if (Hozi_weight < 4 * Verti_weight)
    then  "predict from TOP"
  else
    then  "NULL predict"
```

2.2.7 熵編碼(Entropy Coding)

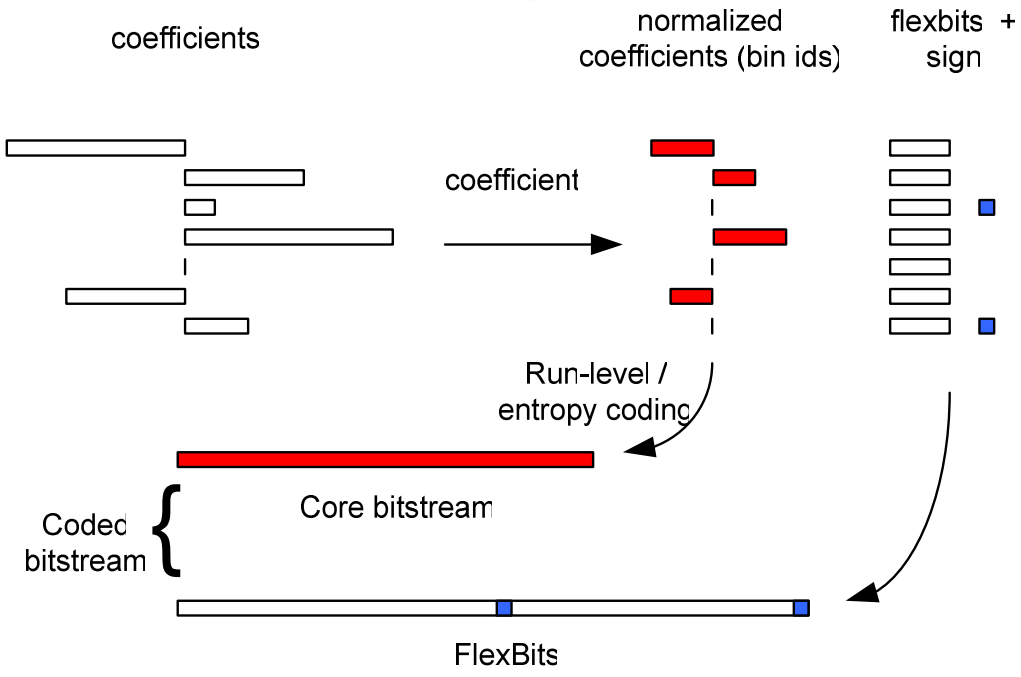


圖 2.19：串流編碼架構

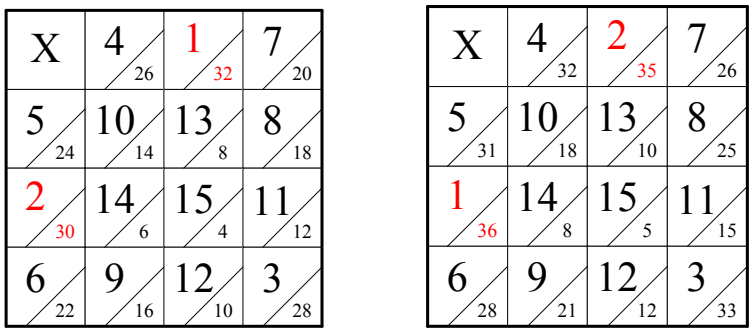
如上圖 2.19 所示，目前 CMOS 鏡頭的解析度都可以達到 10、12 位元以上，意思即為每個係數都可能會是以 12 位元來表示，直接送去霍夫曼編碼所造成的系統負擔太大，在硬體或軟體的實做上，過大的霍夫曼碼表會消耗大量的搜尋時間與功率，所以 JPEG XR 的熵編碼是採用適應性的編碼方式，編碼的概念是認為每個係數，在送進編碼前，先將係數做正規化(Normalized)，舉個例子說明，圖 2.19 是所有係數以白色長條柱表示，希望經過演算法後能找到所有係數的趨勢(紅色長條柱)，縮減係數的表示範圍，將每個係數最具代表性的部份擷取出來進行編碼，即可降低霍夫曼編碼表的大小與系統負擔。

圖 2.19 左上方的白色長條柱即代表每個區塊裡的係數，熵編碼器會先將這些係數依照適應性掃描順序一一抓取出來處理，首先熵編碼器給予每一個區塊適當的正規化係數，例如給予 16 來正規化低頻係數，當然，區塊內的係數不一定能使用 16 來正規化，係數有可能正規化之後會為 0，至於可被正規化的部份係數就額外處理成圖 2.19 上的正規化係數(Normalized Coefficient)，Flexbits 就是每個係數將有意義的部份接取之後剩下來的資訊，藍色點代表該係數的符號，最後再將資料按照順序排列即可完成熵編碼。

2.2.8 適應性掃描(Adaptive Scan)

JPEG XR 的掃描採用適應性掃描的方式，而非大家熟悉的固定掃描法，例如：JPEG 的 Z 型掃描(Zig-zag Scan)。首先由系統先給一系列的掃描順序(Scan order)與權重(Weight)作為預設值，進行掃描時，就記錄該係數是否非零，是的話就在該係數的權重上累加，處理完一個區塊就進行更新掃描順序的動作，將掃描順序依照權重的大小重新排列，過程完全動態調整，如圖 2.20 所示，區塊內的數字即為預設的掃描順序，每個順序都會有相對應的權重，依據圖 2.20(a)上的掃描順序將係數擷取出來，此時，演算法也會判斷係數是否為 0，否的話則在該

掃瞄順序的權重加一，例如：當掃描順序為第二的係數發現是非零的次數(即權重)大於掃描順序為第一個係數時(36>35)，就交換彼此的順序，目的就是減少資料間的亂度，使得非零係數能集中並先處理。



(a) (b)

圖 2.20：適應性掃描 (a)之前的掃描 (b)更新後的掃描順序

2.2.9 Run-Level Encode(RLE)

Run-Level Encode 是 JPEG XR 的特殊編碼方式，概念類似 JPEG 的串長編碼(Run-Length Encoding)，目的都是減少資料間的累贅，JPEG XR 採用不同程度的正規化方式完成編碼，即之前談論的正規劃概念，實際演算法根據不同頻段的係數給予不同的正規化程度，例如將經過預測演算法後的係數依不同的頻道(Channel)如：Y 與 UV；不同的頻段(Band)，如：直流、低頻與高頻係數；進行編碼，我們知道經過預測演算法後的高頻係數，大部分都集中在 0 附近，就固定使用 2 來做正規化，或者低頻係數由於所包含的能量較大，就使用 16 來正規化。當然，每個低頻係數都不一定能被正規化，係數大於 16 時能表示的範圍時就需要另外處理，圖 2.18 可以說明這處理流程，當一個區塊進行適應性掃描將係數擷取出來並且選擇 16 來作為正規化參數來編碼，發現係數 17 經過正規化之後不為 0，系統會將多餘的值(紅色底線部份)標示為 Level，而前面 3 個係數經過正規化後都為 0，所以把 Run 標示為 3，最後將被正規化後的部份(藍色底線)與 Run、Level 係數查霍夫曼表進行輸出即可完成 RLE 編碼。

其中每個係數經過正規化後剩餘的部份(藍色底線的部份)稱為 Flexbits，這部份可以參考圖 2.21，這部份是直接輸出至串流編碼而不進行霍夫曼編碼，原因在於編碼的效率不佳，一個 2 進制的係數將最高位元 (Most Significant Bit, MSB) 附近的的部份截去，剩下的部份(即藍色底線部份)很難找出關聯性。

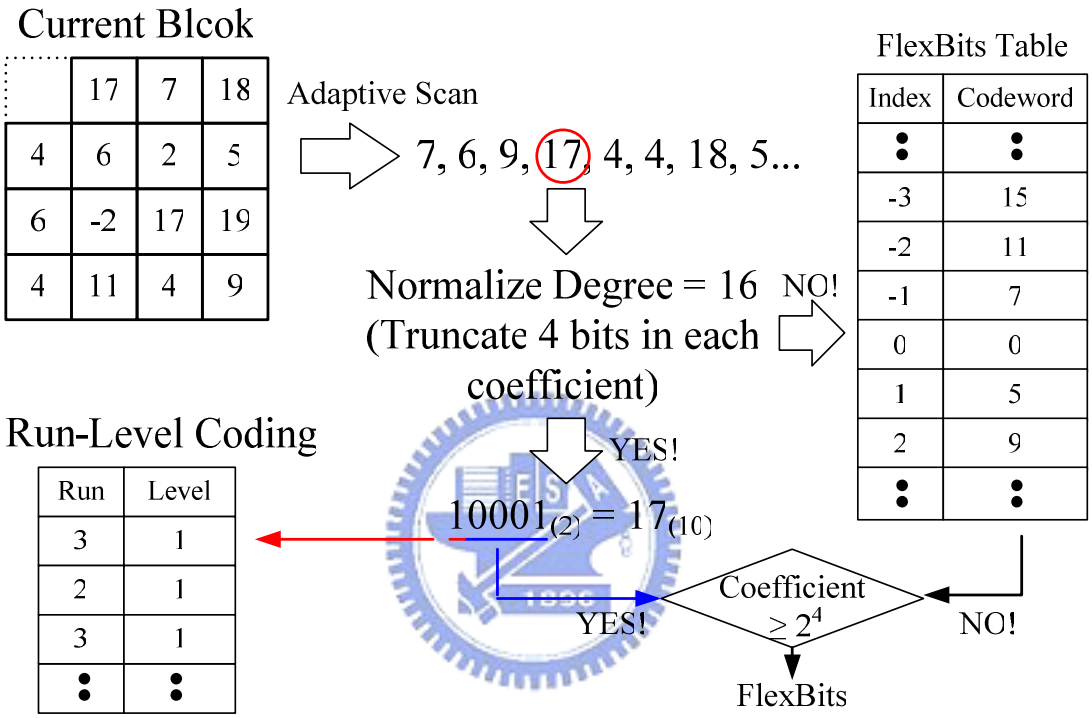


圖 2.21：RLE 演算法的一個例子

2.2.10 霍夫曼編碼(Huffman Coding)

霍夫曼編碼是電腦資訊最常見的編碼方式，屬於無失真壓縮演算法，JPEG XR 也採用此演算法來改良加以應用，而不採用算術編碼來增加硬體複雜度，傳統式的霍夫曼碼表是一大塊，任何係數都使用同一個表，這就會造成無法最佳化的問題，例如上述 RLE 的例子，Run 與 Level 兩種參數的分狀況不一樣，Run 分佈在 2～3 之間，Level 則在 1，這會導致 1、2、3 這三個數在編霍夫曼碼表的長度一樣，編碼效率不佳且在硬體實做上，一次搜尋整塊霍夫曼碼表的速度較慢也較消耗功率，所以 JPEG XR 採用較小的霍夫曼碼表，針對不同參數設計不同

的表，每個參數又有多個表可使用，如表 2.4 所示，由演算法根據目前的使用 Index，即目前要用霍夫編碼的值來選擇相對應的 Delta 值，JPEX XR 會利累積這數值，我們可以看見如果使用 Series 9-1 的表在 Index 5-8 的 Codesize 會比 Index 0-3 的 Codesize 來的短，Series 9-2 剛好相反，二個表都有不同的特性，依據 Delta 累積的結果為正或負來選擇霍夫曼碼表，達到最佳的壓縮效率。

表 2.4：JPEG XR 所使用的霍夫曼碼表

Huffman Table, Series 9-1		
Index	*Codeword*	*Codesize*
0	2	3
1	0	5
2	2	4
3	1	5
4	2	5
5	1	1
6	3	3
7	3	5
8	3	4

Huffman Table, Series 9-2		
Index	*Codeword*	*Codesize*
0	1	1
1	1	3
2	2	3
3	1	4
4	1	6
5	3	3
6	1	5
7	0	7
8	1	7

Delta Table, Series 9	
Index	*Delta*
0	2
1	2
2	1
3	1
4	-1
5	-2
6	-2
7	-2
8	-3

第三章

架構設計

3.1 效能分析(Profiling)

在設計 JPEG XR 之前，我們先對微軟所提供的原始碼進行效能分析，結果如圖 3.1 所示，測試的項目為執行時間，將整個編碼流程主要分為三段，分別是頻域轉換(Transform)、資料預測(Prediction)與熵編碼(Entropy Coding)，分析發現整個熵編碼所佔的比例為 65%，相當驚人，其中又以編碼高頻係數最花時間，低頻係數次之，直流係數最少，原因在於每個大區塊有 240 個高頻係數、15 個低頻係數與 1 個直流係數，在進行運算時，熵編碼有多個回授演算法來偵測目前係數的狀況來調整 JPEG XR 的編碼參數，例如調整正規化的參數與適應性掃描順序等等，來達到最佳化的編碼狀態，增加壓縮效率，但所付出的成本就是運算複雜度增加，與頻域轉換的運算量不成比例，且在熵編碼階段，每筆係數又有前後順序的關係在，實做上相當棘手，所以在設計 JPEG XR 的編碼晶片時，必須先加速高頻係數的熵編碼，否則這部份將會是硬體實做上最大的瓶頸，另外一個值得注意的地方在於資料預測的部份，所佔的運算比例非常低，原因在於每個大區塊不一定要去執行資料預測演算法，需要進行運算時，每個大區塊也只有部分係數需要運算，所以整體需要運算的時間非常低，這裡又造成另一個問題就是熵編碼的複雜度很高，資料預測的複雜度最很低，而頻域轉換的部分運算量卻在中間，如何在硬體架構上使用管線技巧去平衡與加速這三部分，我們將在之後的章節詳細論述。

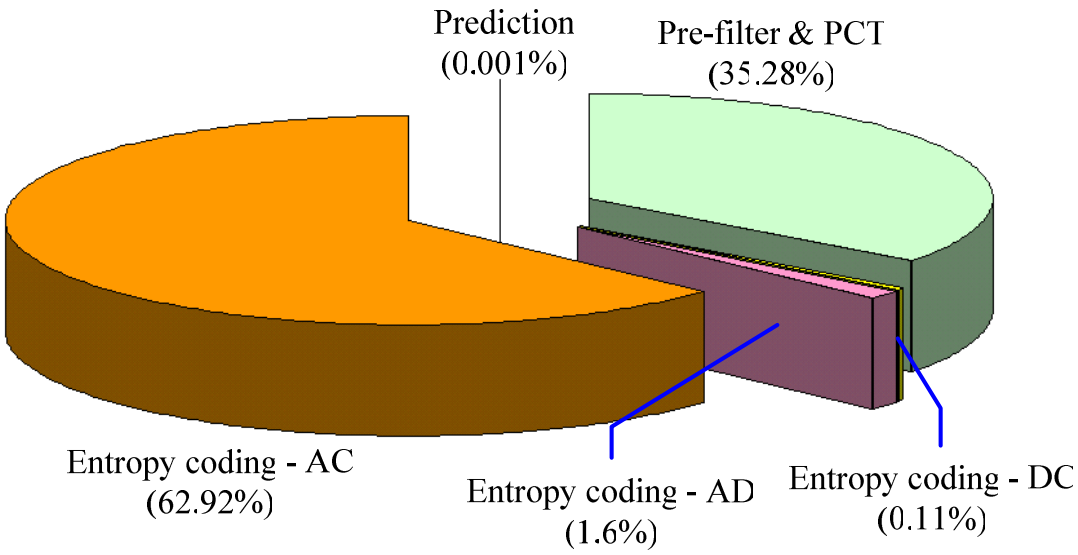


圖 3.1：Profiling

3.2 設計挑戰(Design Challenges)

JPEG XR 在硬體實做上會遇到的困難點，在此列舉。

1. 資料排程的不相同(Data Schedule Mismatch)

   在前置濾波器與 PCT、資料預測、熵編碼的區塊執行順序都不一樣，這意味著 PCT 處理完後的區塊不能馬上給資料預測演算法運算，而資料預測運算完之後也不能馬上給熵編碼，這勢必須要一些暫存器來暫存資料，增加硬體成本。

2. 區塊順序的不相同(Block Order Mismatch)

   跟 JPEG 演算法流程比較，JPEG XR 多了一個前置濾波器，好處是可以增加壓縮率與在高壓縮比時，減低方塊效應的產生，但前置濾波器在硬體實做上卻會造成嚴重的問題，它處理的位子剛好是區塊的邊界，意思是經過色相轉換後的區塊不能直接給前置濾波器，必需集滿四個區塊後再抓取部份係數給前置濾波器處理，這部分又要花費一些暫存器來暫存資料。

3. Code Block Pattern (CBP)

在熵編碼上，有一個單元為 CBP，它的功用在於標記每一個區塊是否有係數被正規化，每個大區塊共有直流、低頻與高頻頻段需要標記，舉個例子，大區塊在輸出為串流編碼時，必須先偵測每個區塊是否有被正規化來決定 CBP 值，在進行編碼輸出，以時間上的觀點來看，即每一個區塊判斷是否有正規化之後，此區塊內經過熵編碼後的的係數不能馬上輸出為串流編碼，必須等待所有區塊的 CBP 都判斷完，這意味著又必須花費多餘的暫存器去暫存每個區塊的資料。

4. 熵編碼器的設計

根據 Profiling 的分析可以得知，熵編碼器是最複雜的單元，原因在於 JPEG XR 有多種的回授機制來偵測目前的係數狀態，給予適當的編碼參數來達到最有效率的壓縮，所以複雜度相當高，硬體設計上就必須想辦法加速熵編碼的運算，否則整個晶片的瓶頸將會卡在這，但其它部份的運算量最遠遠小於熵編碼器，如何在硬體實做上將每個單元做最有效率的規劃是一大難題。

3.3 系統概觀(System Overview)

JPEG XR 採用以大區塊為單位的處理方式來進行編解碼，根據之前原始碼的效能分析發現，整體 JPEG XR 在歸劃上的困難點在於熵編碼器的設計，使用多個適應性演算法來達成最佳化的壓縮，也就是說適應性演算法會一直回報目前係數狀況給系統，所以這部份在時序上的控制會是最大的挑戰，在管線的切割上，我們規劃為三級管線架構，管線階段 1(Stage 1)包含色相轉換、前置濾波器，PCT 與量化，經過 SRAM 將資料傳給管線階段 2(Stage 2)來進行資料預測的動作，最後由管線階段 3(Stage 3)收尾，做熵編碼與可變長度編碼(Variable Length Coding, VLC)，每個管線階段裡還有細部的管線架構，為超管線架構(Hyper

pipeline)，例如管線階段 1 裡的前置濾波器與 PCT 轉換都以管線來執行，增加硬體效能。

之所以將 JPEG XR 演算法切為三個管線階段的原因在於資料預測演算法與熵編碼都有自己執行區塊的順序，這順序必須由本身大區塊的內容決定，例如預測方向可能往上或往下，所以在這中間必須有管線暫存器(SRAM)來存放資料，而熵編碼的過程中，區塊的執行順序又不一樣且係數與係數間的關係很密切，根據上面討論，以三階管線執行是最有效率的方式，如圖 3.2 所示，在控制單元上，我們採用有限狀態機器來實現，設計多種狀態來供所有處理單元使用。

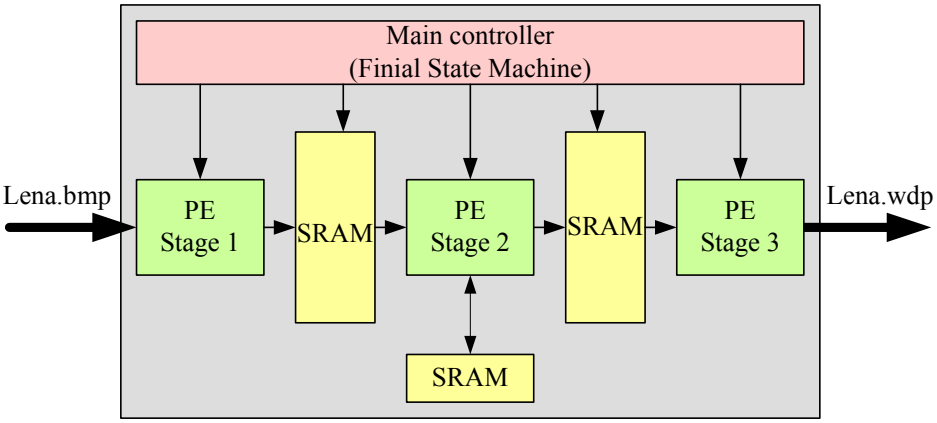


圖 3.2：JPEG XR 的管線切分式意圖

3.4 管線架構(Pipeline Architecture)

由圖 3.3 說明一個例子來表示 JPEG XR 以三階管線執行的方式，每個方塊代表一個大區塊，以列為單位進行管線運算，由於前置濾波器與 PCT 之間溝通的緣故，所有大區塊執行前都有前處理(Pre-Processing)演算法將資料預先處理置暫存器陣列裡，JPEG XR 演算法上比較特別的地方就是在每個列的資料執行完後，會將部分參數重置，如熵編碼的正規化參數等，無法使用很密集的管線處理，所以管線的效率會稍微打折。

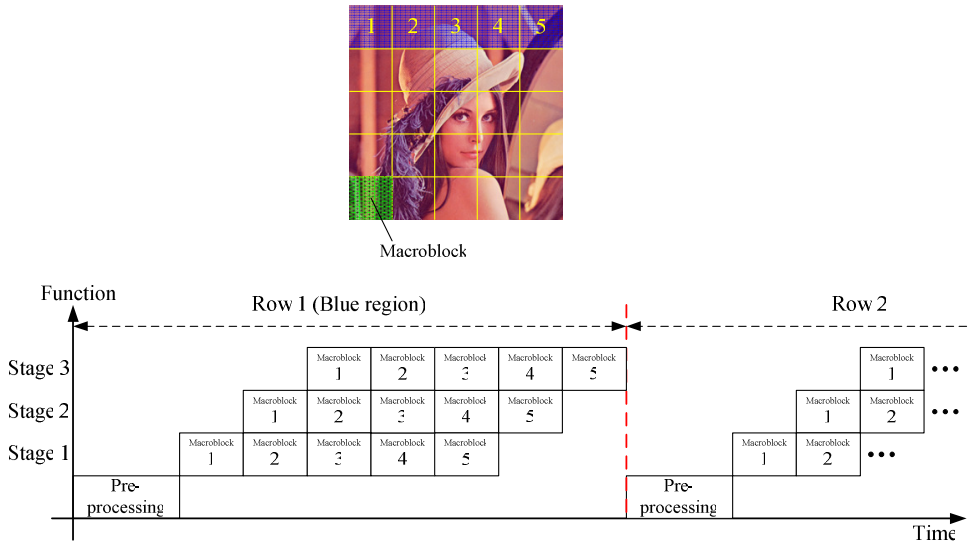


圖 3.3：以時間軸說明管線執行方式

3.5 有限狀態機器(Finite State Machine, FSM)

首先分析管線階段 1 內部元件的特性，發現都是以區塊為單元進行處理，一級傳給一級，這非常適合以管線方式進行處理，而管線階段 1 在硬體實做上有一個特別的東西就是前置濾波器，這會造成在處理區塊時的順序不相同，原因是前置濾波器與 PCT 所抓取的區塊資料剛好不一樣，所以在中間必須建立暫存器陣列來存放部份資料，加入資料重複使用技術，以便 PCT 能夠順利抓取一個完整的區塊資料，有限狀態機器的規劃上就會多一個前處理的狀態，目的就是為了暫存器陣列。

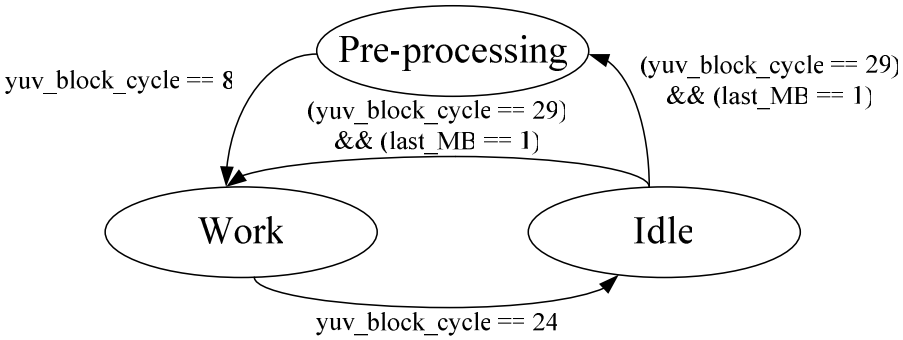


圖 3.4：管線階段 1 的 FSM

3.6 資料重複技術(Data Reuse)

JPEG XR 在硬體設計上有一個困難點就是前置濾波器與 PCT 轉換所處理的區塊並不是同一個，以前置濾波器所處理的區塊剛好落在 PCT 要處理的區塊的邊界上，如圖 3.5 所示，虛線框框代表前置濾波器要處理的區塊，藍色方塊則代表 PCT 所要處理的區塊，總共有 16 個，發現 PCT 要處理一個區塊則必須由四個經過前置濾波器處理後的區塊來組成，這是相當麻煩的地方，尤其是在邊界的地方，所以處理一個大區塊，就必須連同周圍的區塊一起處理，因此我們設計了一套資料重複使用技術來解決重複處理的問題，在色相轉換、前置濾波器與 PCT 中間加入暫存器陣列來暫存處理後的資料，我們比較加入資料重複技術在對外讀取的頻寬如圖 3.6，可減少了 33.3%的頻寬所需。

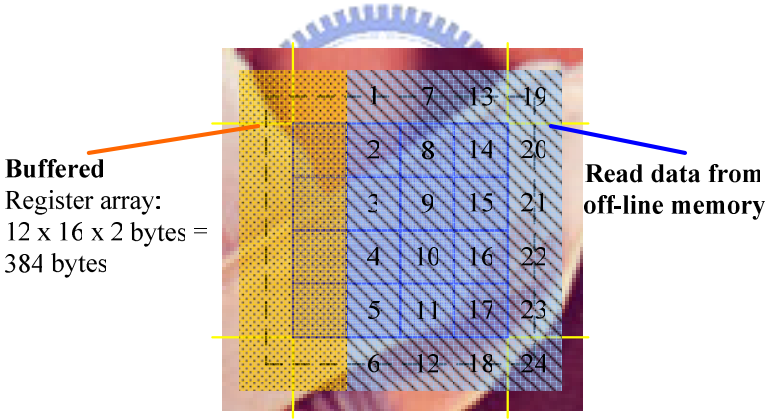


圖 3.5：區塊的執行方式

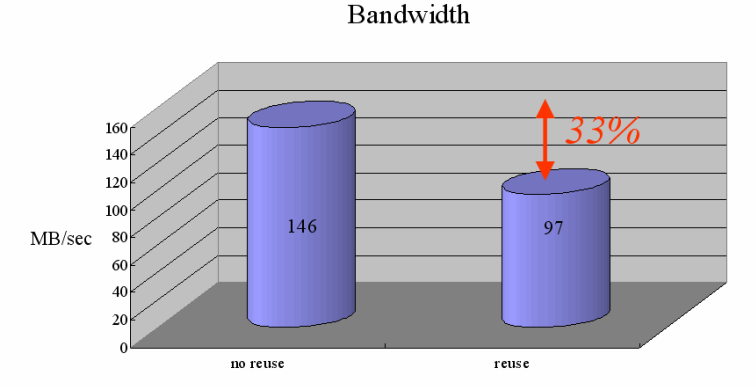


圖 3.6：所需的頻寬比較

在我們的資料重複技術上，每個大區塊在處理時，只需要讀取 24 個區塊即可，如圖 3.5 所示，藍色斜線部份即是要從外部記憶體讀的，剛好 24 個區塊，而左邊橘色的代表在處理上一個大區塊已經從外部記憶體讀取過了，我們將它暫存起來，以便目前的大區塊使用，對於下一個大區塊而言，就必須將目前編號 13～24 的區塊暫存起來供之後使用。

以圖 3.7 管線執行方法來描述資料重複技術，管線階段 1 在每個大區塊只需要處理 24 個區塊，剩餘的時間可以使用 Clock Gating 來節省功率消耗，主要元因是受限於後端熵編碼的緣故，而管線階段 2 與管線階段 3 在區塊的處理上就沒有之前敘述的問題，所以只有管線階段 1 需要使用資料重複使用技術。

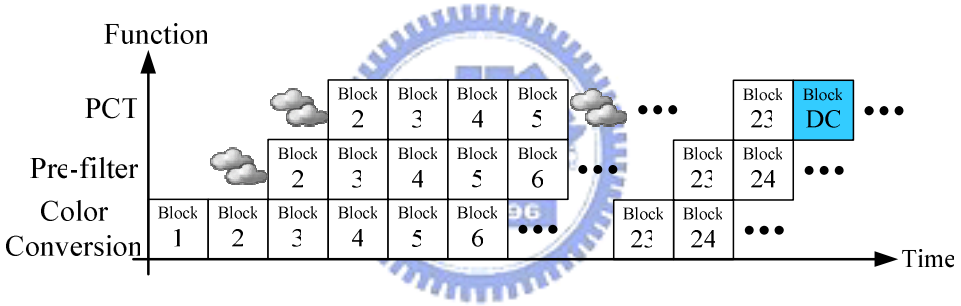


圖 3.7：管線階段 1 的內部管線圖

3.7 架構實現

3.7.1 管線階段 1

管線階段 1 包含有色相轉換、前置濾波器與 PCT 轉換，圖 3.8 所示，所有單元都是以區塊為處理單位，所以放在同一個管線階段，此架構內部以三階管線的方式執行來增加效能，為了前置濾波器，我們加入了二條暫存器陣列來暫存色相轉換與前置濾波器處理後的結果，原因在 3.2 節有討論過，並在 3.4 節提出了資料重複技術來解決此問題，經過管線階段 1 的處理，將運算結果放置 SRAM 供管線階段 2 存取使用。

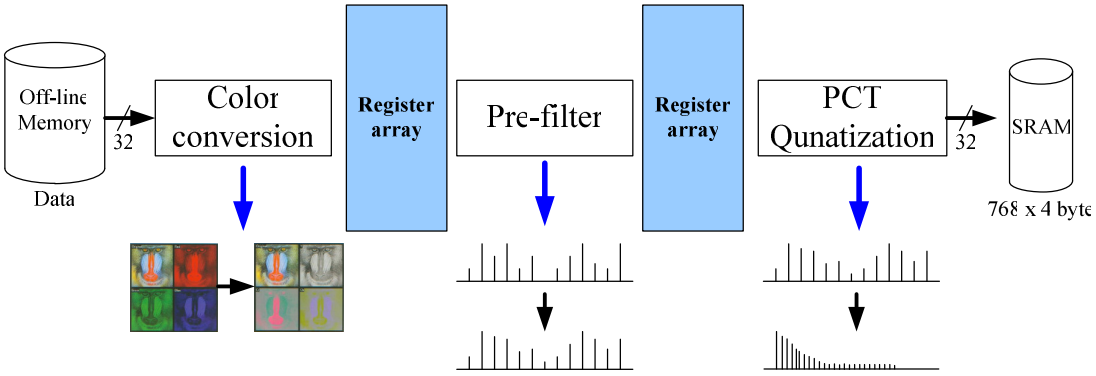


圖 3.8：管線階段 1 的內部資料流

色相轉換、前置濾波器與 PCT 架構

色相轉換在硬體實做上是最簡單的單元，由加法、減法等即可完成運算，我們可以見圖 3.8 與圖 3.9，資料從外部離線式記憶單元獲得，將資料依照順序存放置色相轉換單元裡的暫存器，等到紅色、綠色與藍色的資料都到齊之後就進行運算，一次轉換四個像素，也就是說以區塊為單位，一次轉換一列，執行四次即可完成一個區塊的轉換。

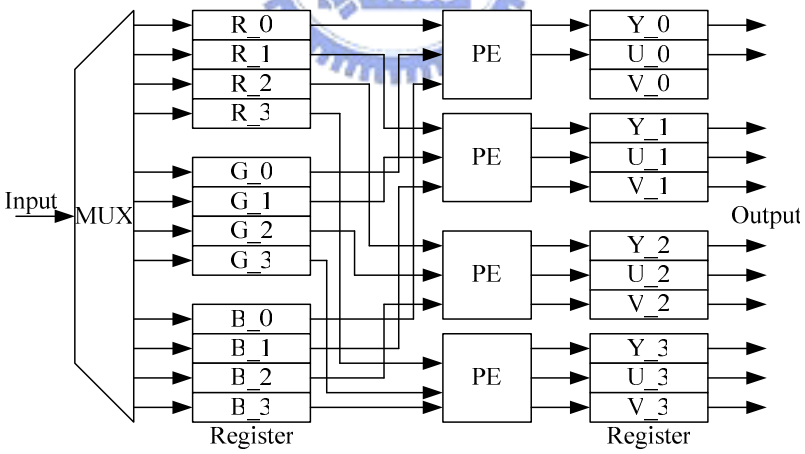


圖 3.9：色相轉換架構

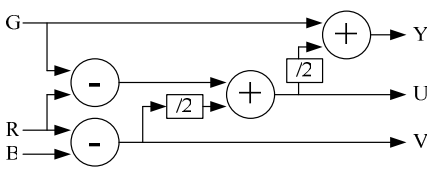


圖 3.10：色相轉換的處理單元

管線階段 1 複雜的地方就是前置濾波器與 PCT，前置濾波器演算法即是一堆數學，經過大量的加法、減法與移位，由多個子程式組成的轉換，運算量相當大，如果為了節省面積與功率消耗，可以將這單元捨棄，權衡方式如 2.2.3 章節所述。整體架構採用管線式設計，提升輸出量(Throughput)和工作頻率，每個區塊可以連續一直處理，並分為邊界處理與非邊界處理單元，由控制單元來做適當的選擇。

PCT 的架構類似前置濾波器，如圖 3.11，都有複雜的數學運算，不同的地方在於 PCT 必須將轉換後的區塊，把每個區塊的直流係數回存至暫存器陣列，準備做第二階段的處理，過濾出大區塊的直流係數與低頻係數，以利量化器的處理，PCT 也是採用管線設計方式，可以大量的處理資料，輸出由 SRAM 負責儲存運算後的結果，演算法由多個加法、減法、移位器組成，對硬體實做而言，是相當友善的演算法。

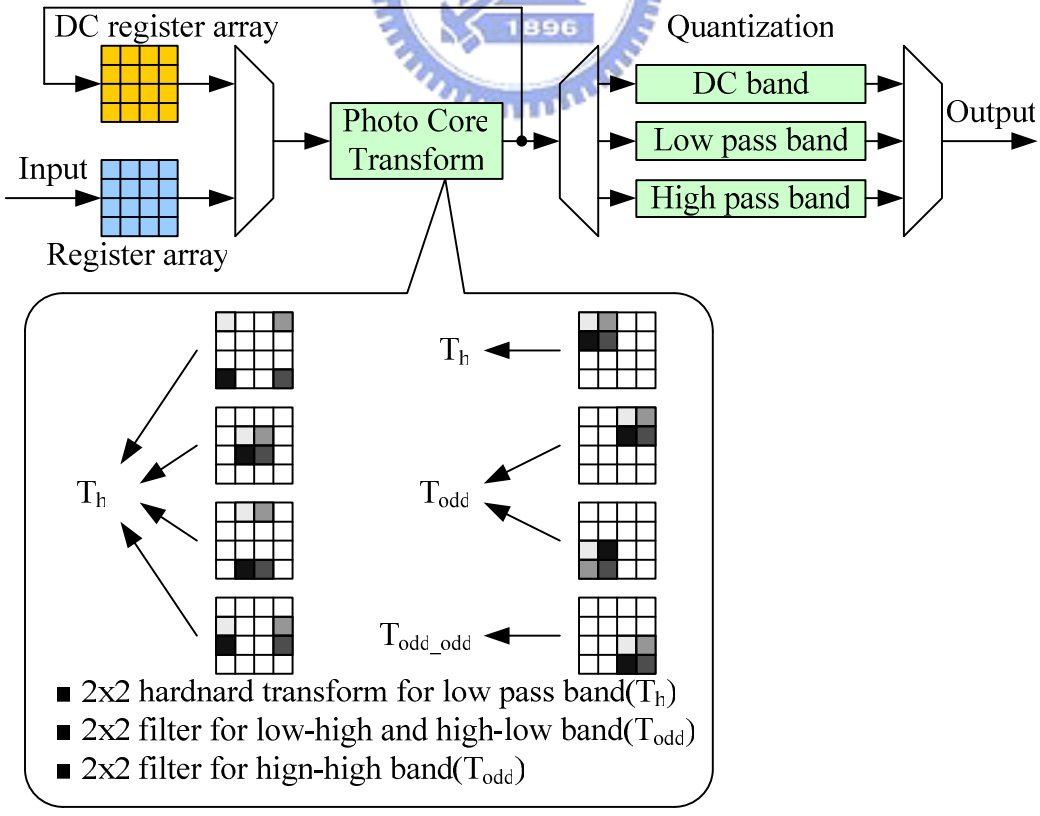


圖 3.11：PCT 的硬體架構

3.7.2 管線階段 2

資料預測架構

資料預測在硬體實現上是最簡單的，如圖 3.12，只需控制好有限狀態機器，讀取適當的係數進行減法運算，SRAM 1 與 SRAM 2 即為管線暫存器，容量為 768 Bytes 的原因是必須存放一個大區塊內的所有係數，比較特別的地方就是 SRAM 3 的存在，直流係數要決定從哪個方向進行預測時，必須由左上方、上方與左方的大區塊的直流係數來決定,所以 SRAM 3 的功用即為存放上方那列所有大區塊的直流係數與部份低頻係數，另一點更特別的地方在於此架構還包含了 CBP 預測器(CBP Predictor) ，由於 CBP 會造成硬體設計上的麻煩，必須多一級管線來處理，花費多餘的硬體成本，但該管線階段的運算量又很低，只有判斷每個區塊有沒有係數被正規化，至於發生的原因在設計挑戰的地方已經詳述過。

我們在此加入 CBP 預測器來預先判斷每個區塊的係數是否有被正規化，將結果一並存放至管線暫存器中，隨著管線的傳遞，熵編碼器就能同時抓起大區塊的係數與 CBP 資料，即可馬上編碼，不需多花費一級管線的時間即可完成編碼，節省硬體資源。

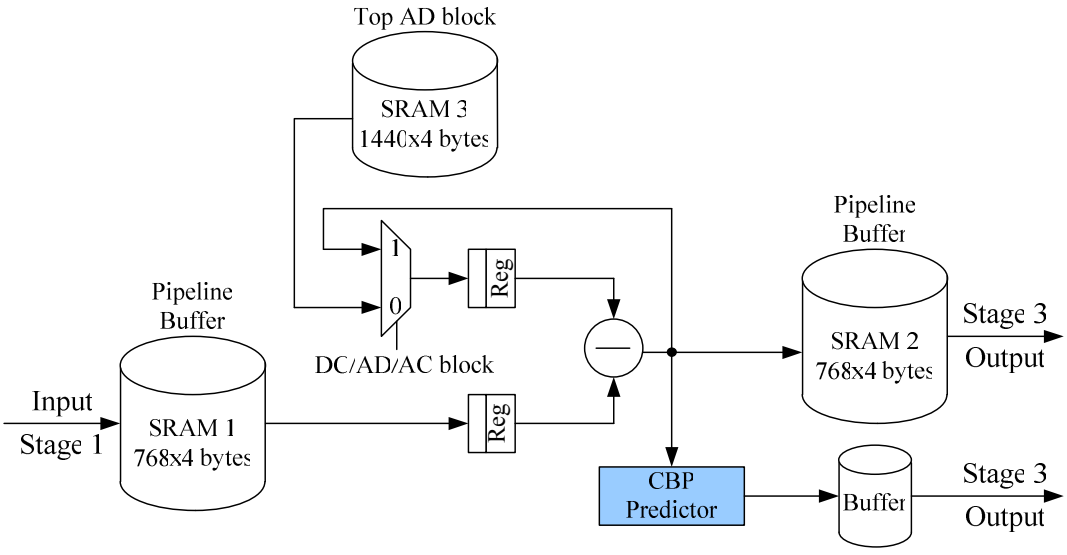


圖 3.12：資料預測的硬體架構

3.7.3 管線階段 3

經過之前演算法的討論，可得知此部份是 JPEG XR 最複雜的地方，我們使用圖 3.13 來說明熵編碼，在第二章演算法時，已經有說明過熵編碼首先會進行適應性掃描來選擇係數進行正規化與 RLE 編碼，最後再進行適應性的霍夫曼編碼，硬體實做上就必須要將上述演算法實現並最佳化，除此之外，還有 CBP 必須完成，由於我們在管線階段 2 的地方有加入 CBP 偵測器，可以預先執行，所以在管線階段 3 的設計上，CBP 只要將偵測的結果進行霍夫曼編碼即可，不會是設計上的主要困難點，剩下的目標就是將適應性掃描、正規化與適應性霍夫曼編碼實現。

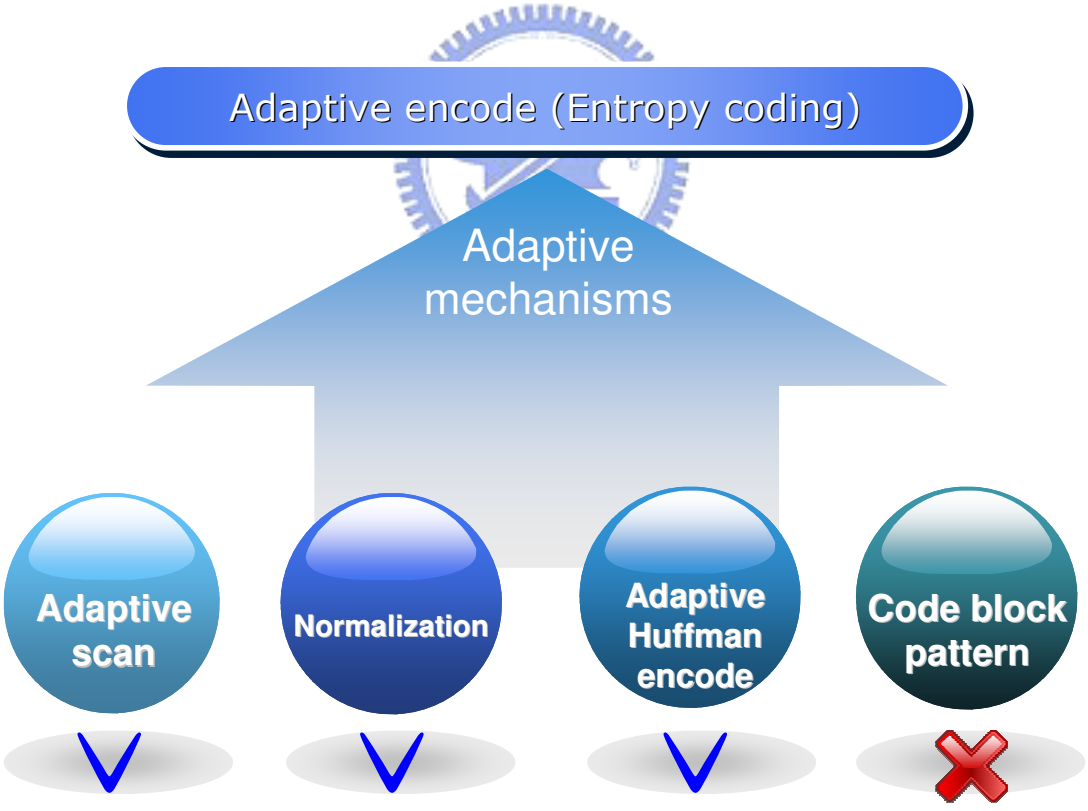


圖 3.13：適應性機制

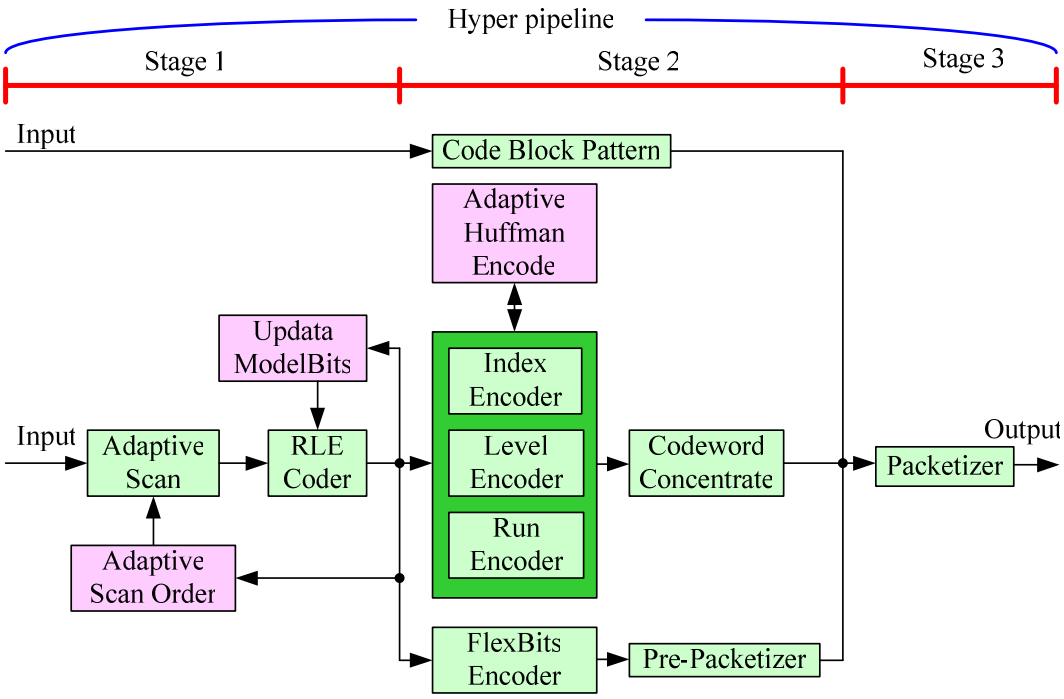


圖 3.14：提出的架構圖

整理成硬體方塊圖在圖 3.14，由多個子電路組成，綠色元件為一般的處理單元(Processing Element, PE)，粉紅色即為回授電路(Feedback Circuit)，有適應性掃描(Adaptive Scan Order)、適應性霍夫曼編碼(Adaptive Huffman Encode)與正規化(Update ModelBits)，專門用來適應目前串流編碼的狀態，達最佳的編碼效率，資料路徑分為二條，一條用來處理 Flexbits 的編碼，由 Flexbits Encode 單元負責，另一條處理直流、低頻與高頻的係數，經過適應性掃描後，將資料以串列方式供 Run-Level Encode 的進行，將資料整理後，由 Code Block Patten 編碼目前區塊被正規化的情形，最後進行適應性霍夫曼編碼，經過霍夫曼查表所得到的碼字(Codeword)依照順序給 Packetizer 單元進行排列。根據上述討論我們提出圖 3.15 的管線架構，為了增加硬體使用率與產出效能，我們將管線階段 3 的處理單元以內部 3 級管線執行，即可以增速 3 倍，管線的切割在 RLE Coder、霍夫曼編碼與 Packetizer 之間，之所以這樣切分的原因在於每個管線階段都是獨立的回授電路，彼此不互相影響。

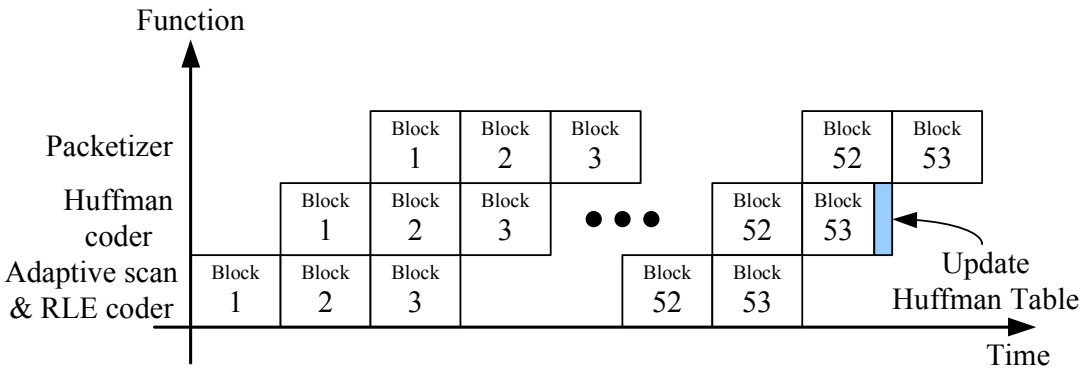


圖 3.15：管線階段 3 的內部管線執行圖

Run-Level Encode(RLE) 硬體架構

圖 3.16 是 RLE 的硬體架構，演算法如之前敘述，依據適應性掃描將資料擷取出來並判斷是否有大於 ModelBits(即正規化的程度)所能表示的範圍，所以資料路徑分為多條，將多工器依照適當的判斷即可輸出運算後的結果。

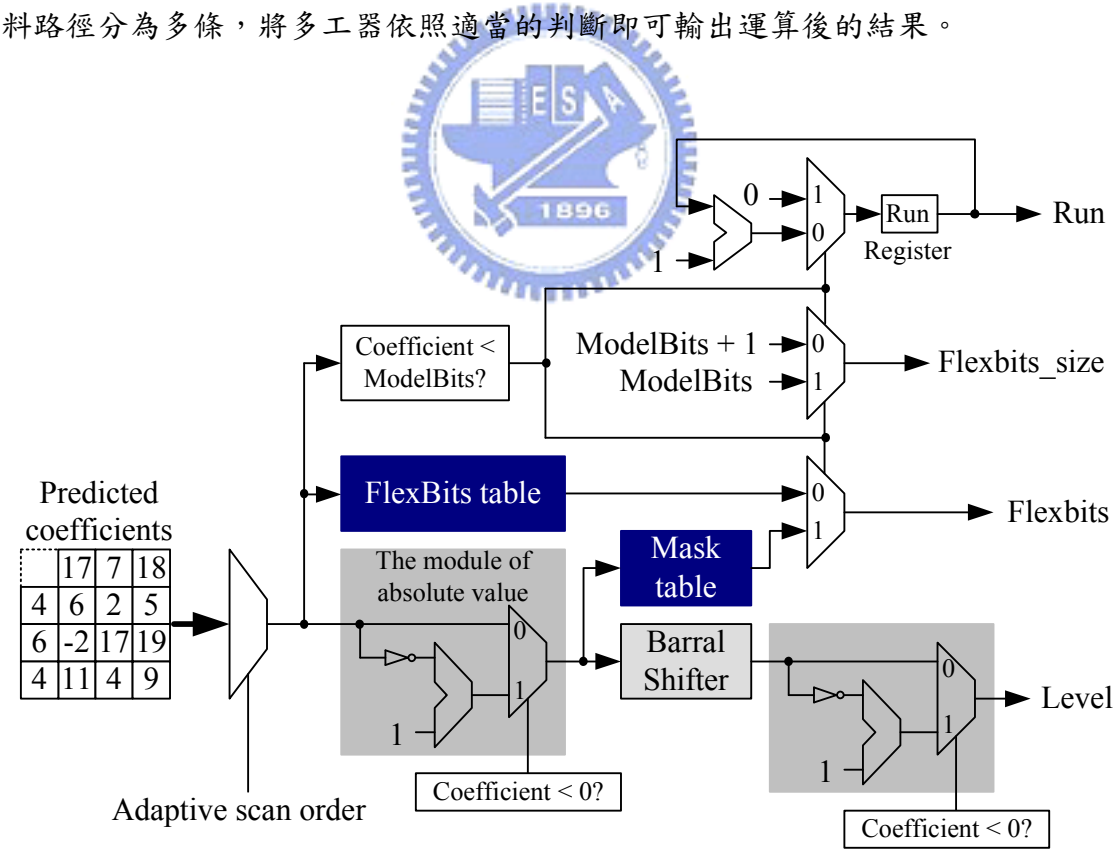


圖 3.16：RLE 的硬體架構

JPEG XR 在硬體實做上的難點之一在於圖 3.17 所示的硬體架構，每個係數經過霍夫曼編碼之後都會產生多個碼字，如果直接送去 Packetizer 處理，則會大幅降低輸出量，進而拖垮整個系統的效率，所以我們在這中間加入了圖 3.18 的電路，先將 Index、Level 與 Run 產生的碼字與其它運算結果先做合併，再送入 Packetizer。

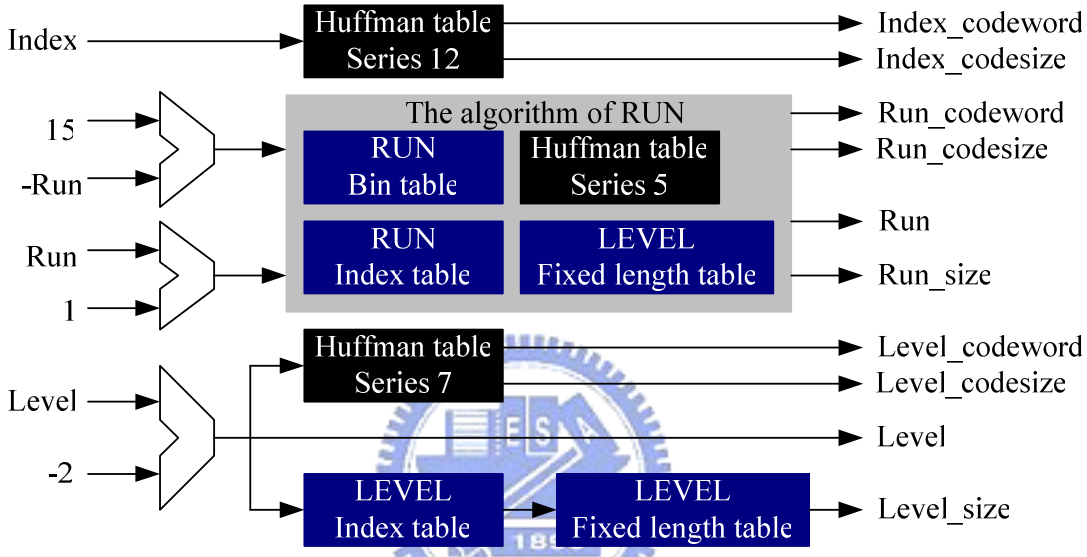


圖 3.17：RLE Huffman encode 的硬體架構

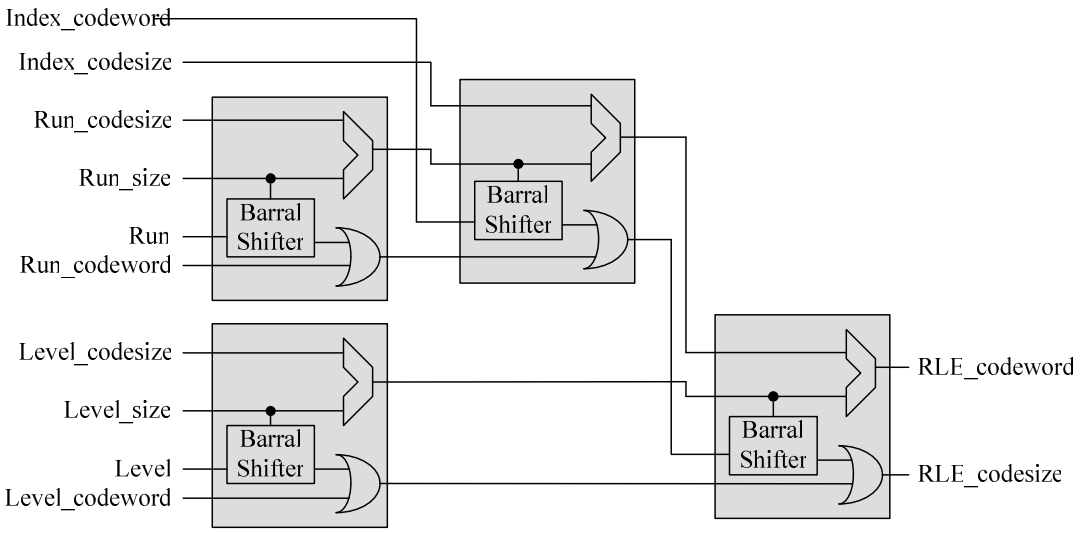


圖 3.18：Codeword Concentrate 的硬體架構

Adaptive Huffman Encode 架構

接著，我們提出適應性霍夫曼編碼架構，如圖 3.19 所示，從輸入端獲得數值並進行霍夫曼查表，在此同時，Delta Table 也會同時動作，根據輸入得到一個 Delta 值累積在 Discriminant 暫存器裡，在適當的時機裡，Discriminant 暫存器會去比較是否有大於 Upper Bond 或小於 Lower Bound，有的話則在 Table Index 暫存器進行運算，最後的輸出就是依照 Table Index 的值來進行選擇。

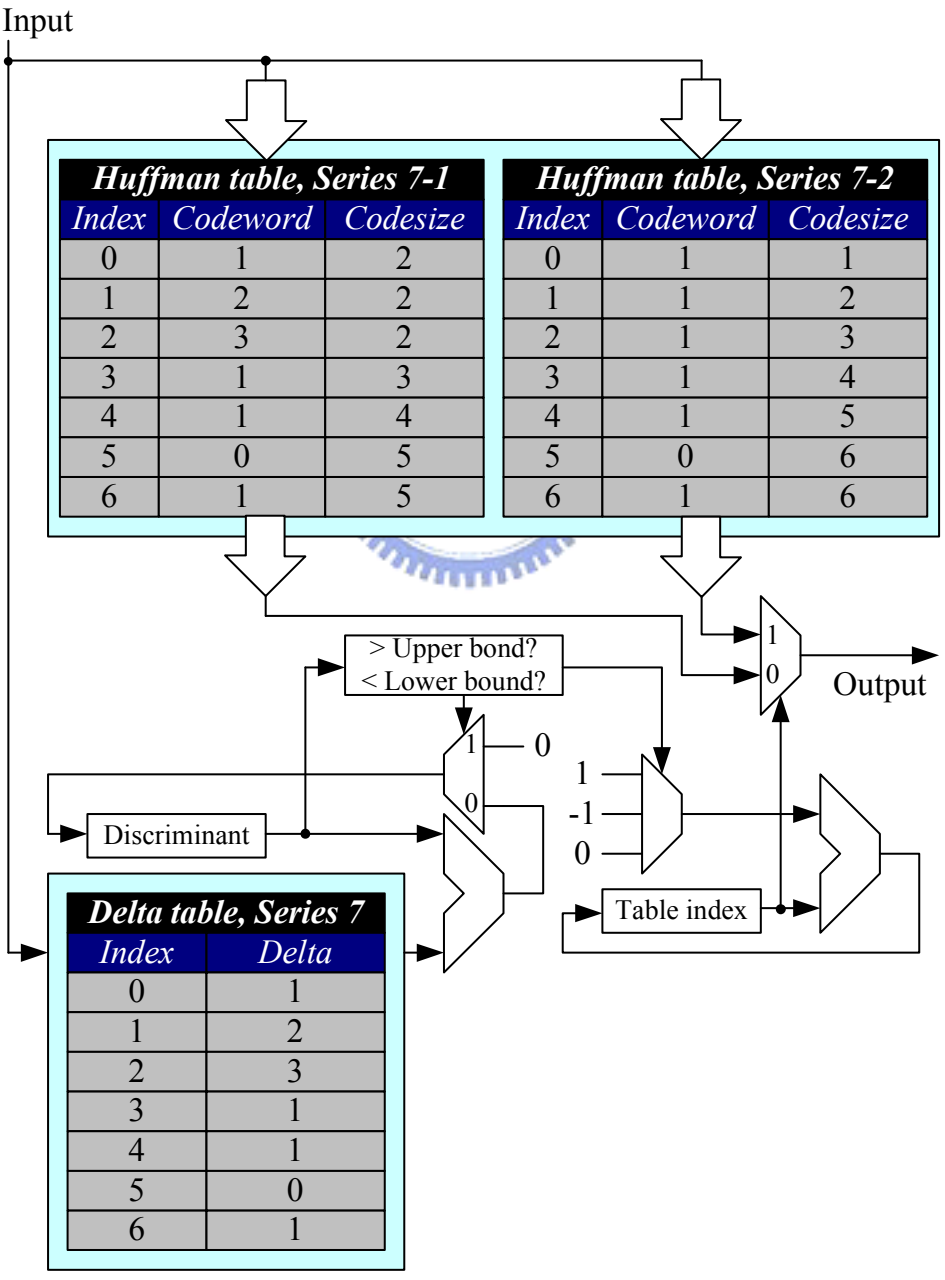


圖 3.19：Adaptive Huffman Encode 的硬體架構

Packetizer 架構

Packetizer 架構我們採用[9] Agostini, L.V.的架構如圖 3.20，簡單明瞭，並且可以調整輸出匯流排的寬度，有相當的彈性，我們設定的輸出匯流排為 32 bit，以符合市場所需，首先 Codeword 與 Codesize 一起輸入，Acc 暫存器會累積 Codesize 的和，Barrel Shifter 由 Acc 暫存器控制 Codeword 的位移量，當 Acc 暫存器累積超過 32 時，OK 信號發出，告訴後端目前的 Word 輸出是有效的。

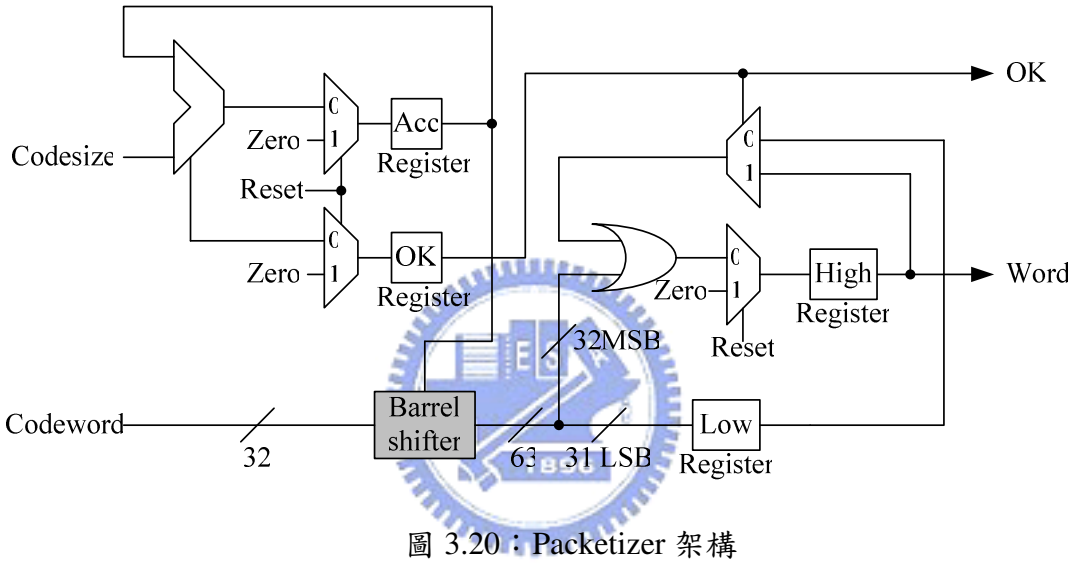


圖 3.20：Packetizer 架構

3.8 總結

在這章節中，我們進行了 JPEG XR 的效能分析與討論實做上會遇到的困難，接著就著手規劃與設計硬體架構，並提出多種設計技巧與電路來解決問題與增加效能，我們就由 3.2 小節所遇到的困難在此進行結論。

1. 資料排程的不相同(Data Schedule Mismatch)

   我們提出以三階管線執行的架構來解決此問題，管線階段 1 包含色相轉換、前置濾波器與 PCT 轉換，原因在於都是以區塊為處理單位，管線階段 2 是資料預測，減少資料的相關性，管線階段 3 則是熵編碼。

2. 區塊順序的不相同(Block Order Mismatch)

   由於加入了前置濾波器來增加壓縮率與減少方塊效應，但這會造成硬體實做上的困難，所以提出了資料重複技術來解決此問題，可節省頻寬達 33%。

3. Code Block Pattern (CBP)

   CBP 的功用及在標記哪一個區塊是有被正規化的，在串流編碼時為了增加壓縮率，對 CBP 進行編碼是不可避免的，但這在硬體實做上必需多切一級管線來執行，我們提出在管線階段 2 的地方多加入 CBP 偵測器來偵測每個區塊是否有被正規化，偵測的結果再給熵編碼統一進行運算。

4. 熵編碼器的設計

   根據 Profiling 的分析，熵編碼是整個 JPEG XR 編碼器最複雜的單元，會造成硬體設計上的瓶頸，所以我們將熵編碼設計成以三階內部管線執行的架構來增加產出量。

第四章

實現

4.1 設計流程

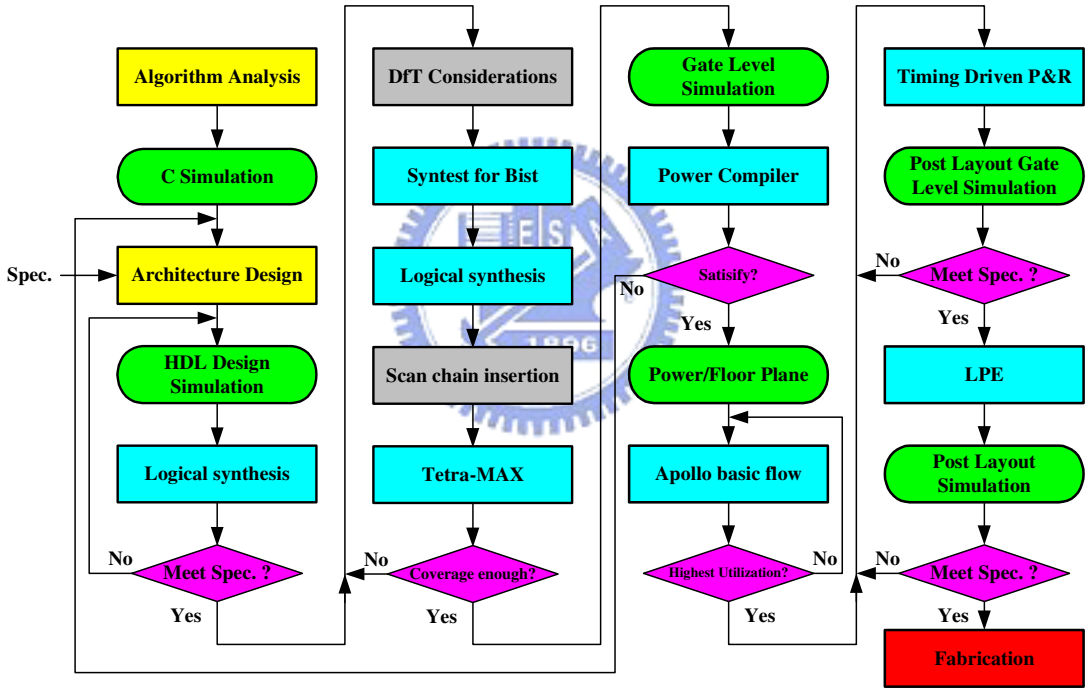


圖 4.1：設計流程圖

1. 演算法分析（Algorithm Analysis），C 語言模擬（C Simulation）

分析 JPEG XR 演算法的複雜度以及硬體實作的可行性、所需的運算量等等，發展適合硬體的演算法，以 C 語言實現之，並確定與原本的原始碼編碼結果完全相同。

2. 架構設計（Architecture Design），Verilog 模擬

根據上一步得到的資訊，進行整體架構的設計。將前述的 C 程式模擬硬體的情況，並分析 System Bus 與 External Memory 之頻寬，以及產生驗證硬體的 Test Pattern。同時我們採用 Verilog 來進行 RTL 的撰寫，使用 ncverilog 與 verdi 來模擬與驗證架構功能的正確性。

3. 設計上測試考量（DfT Consideration）

由於使用相當多的內嵌記憶體，所以採用 March CM 演算法來實現記憶體自我測試（SRAM BIST）模組。

4. 邏輯合成與最佳化（Logic Synthesis and Optimize），掃描鏈（Scan Chain Insertion）

加入測試考量後，完整的 Verilog 在 Synopsys Design-Compiler 環境下進行邏輯合成與最佳化。閘級時脈也在這時候合成。並且加入掃描鏈（Scan Chain Insertion）作測試考量。

5. 閘級時脈驗證（Gate Level Timing Verification）

由於 Design-Compiler 的 STA(Static Timing Analysis)並非完全準確，我們在做完 Synthesis 後，額外使用 Prime Time 軟體驗證 Timing。

6. 閘級模擬（Gate Level Simulation），功率消耗估計（Power Estimation）

將 Synopsys 合成的 Gate Level 的電路，和記錄時間的 SDF 檔案，再使用 Verilog Simulator 進行閘級的模擬，其結果必須與合成前之模擬結果要一樣，而且沒有 Timing Violation 發生。同時利用 Prime Power 收集閘級模擬時的 Switching Activity，作閘級功率消耗估計。閘級功率消耗為 368.7mW @ 62.5MHz，1.8V。

7. Place & Route

我們在晶片繞線方面，使用 Synopsys Astro。由於有閘級時脈限制，為了確保每一點的時脈都是正確無誤的，使用了 SYNOPSYS Astro Timing-driven 的擺放與繞線技巧，使得閘控時脈能夠正常無誤的工作。

8. 佈局驗證（Layout Verification）

產生出來的佈局擋在 Calibre 下作 DRC 以及 LVS 的驗證。

9. 後閘級時脈驗證及模擬（Post Gate Level Timing Verification & Simulation）

為了確保每由於 Timing-driven 的擺放與繞線技巧會改變邏輯閘，所以再做一次用 Primetime 進行時脈上的驗證，並且也使用從佈局中粹取出的電阻電容延遲資訊(RC Delay Information)，再進行一次功能上的驗證。

4.2 晶片規格

此晶片之詳細規格如表 4.1 所示

表 4.1：晶片完整規格內容

Technology	TSMC 0.18um CMOS 1P6M
Core Size	3.05 mm x 3.05mm (9.3025 mm^2)
Die Size	4.525 mm x 4.525 mm (20.47 mm^2)
Gate Count	651.7K (2 Input NAND Gate)
Work Clock Rate	62.5 MHz
Power Consumption	368.7 mW@62.5MHz，1.8V
On-Chip Memory	256x32 SRAM x 6 (Single Port) 768x32 SRAM x 2 (Single Port) 480x32 SRAM x 3 (Single Port) Total: 144,384 Bits = 18,047 Bytes
Processing Capability	34.1 Mega pixels within one second 5.45 fps for 4:4:4 HDTV(1920x1080) @62.5MHz 53.75 fps for 4:4:4 VGA(640x480) @62.5MHz 183.3 fps for 4:4:4 CIF(352x288) @62.5MHz
Input Pad	37
Output Pad	34

4.3 Layout

CKT name : JPEG XR Encoder

Technology : TSMC 0.18um 1P6M CMOS (使用製程)

Max. Frequency : 62.5MHz (最高工作頻率)

Power Dissipation : 368.7mW (功率消耗)

Gate Count : 651.3K (2 Input NAND Gate)

Core Size : 3.05 mm x 3.05mm

Die Size : 4.525 mm x 4.525 mm (晶片面積)

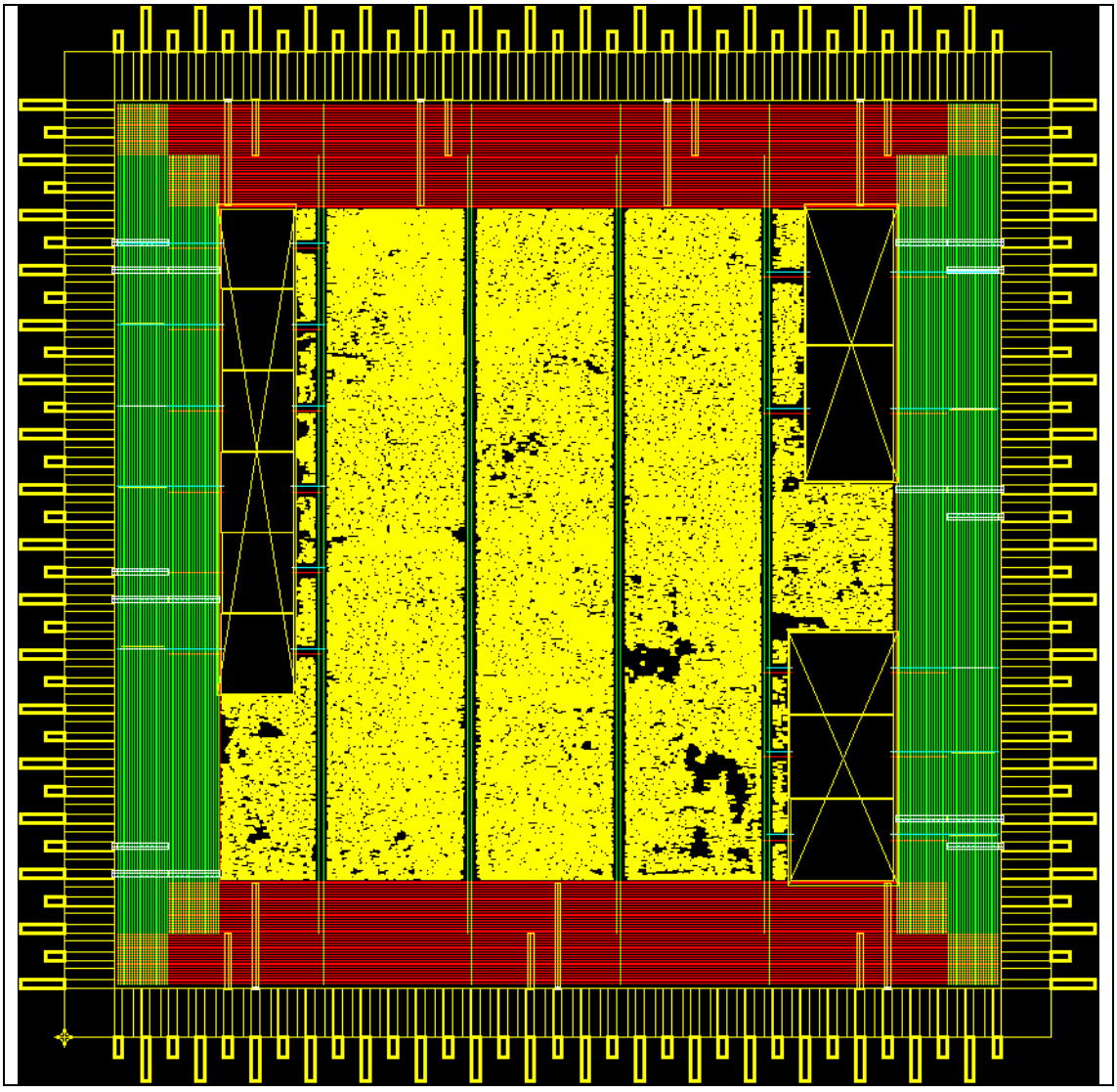


圖 4.2：晶片佈局圖

4.4 佈局分部

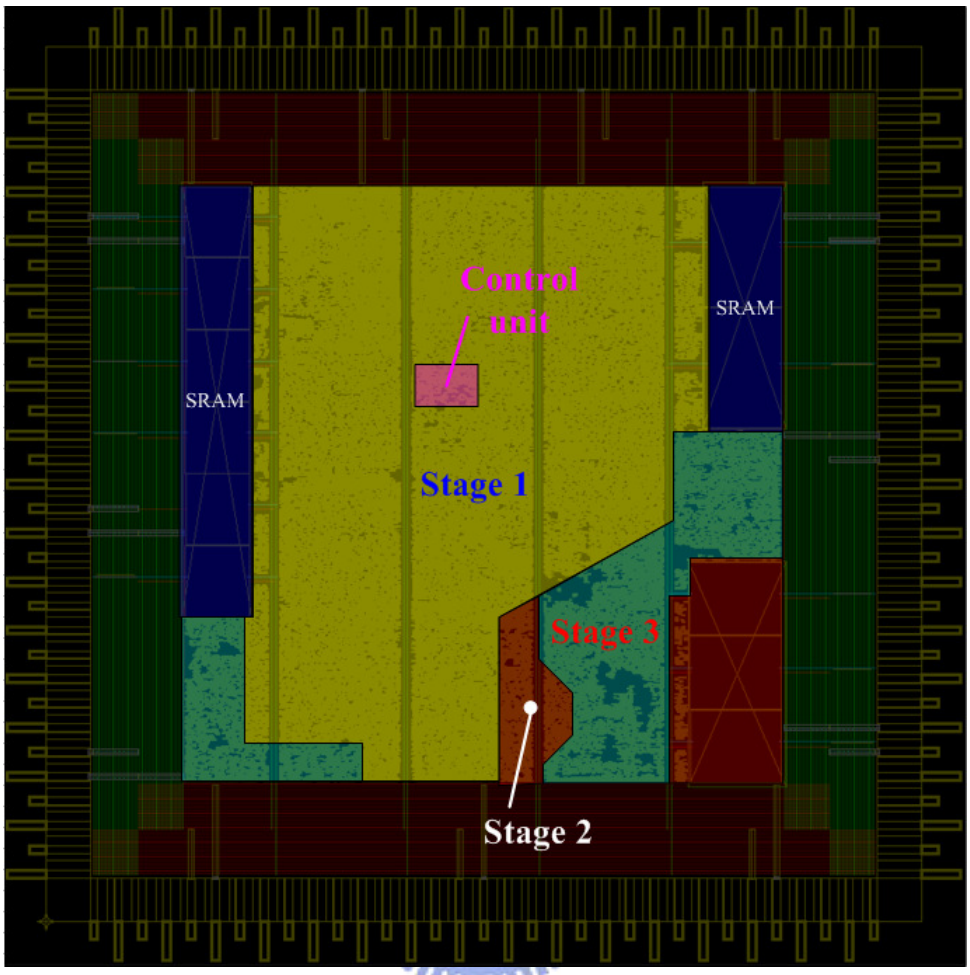


圖 4.3：佈局資源分配圖

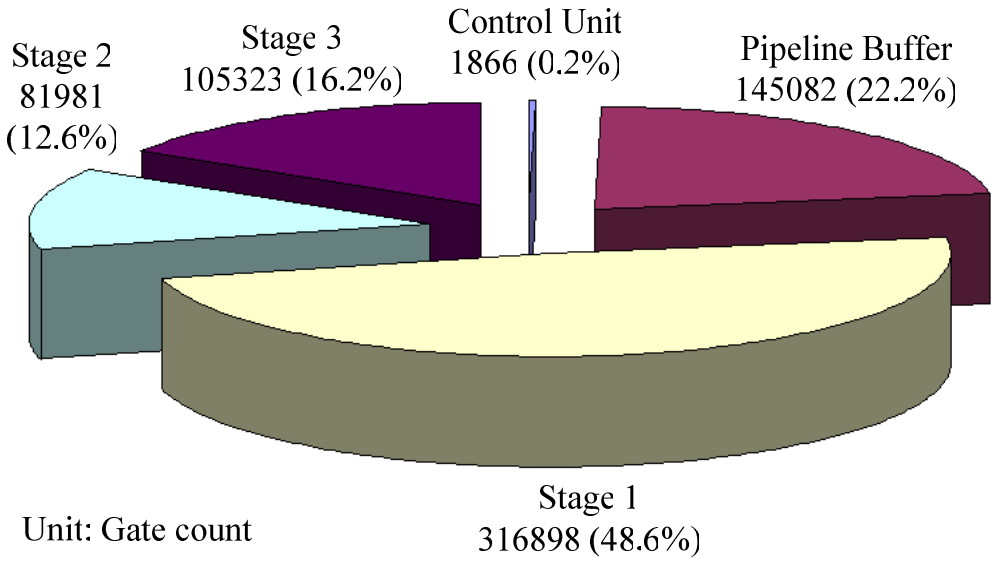


圖 4.4：合成資源分配圖

由圖 4.3 將管線階段 1 再切分出更細的分佈圖如圖 4.5，內部色相轉換、前置濾波器與 PCT 大約佔管線階段 1 的一半面積左右，大部分的面積都為暫存器陣列(黃色區域)，這是我們使用資料重複技術來解省對外存取頻寬所付出的代價，管線階段 2 的比例與 Profiling 的分析結果不一樣的地方在於 3.7.2 節所示的架構，包含有 SRAM 與 CBP 預測器，所以面積會偏大，藍色區域則是管線暫存器。

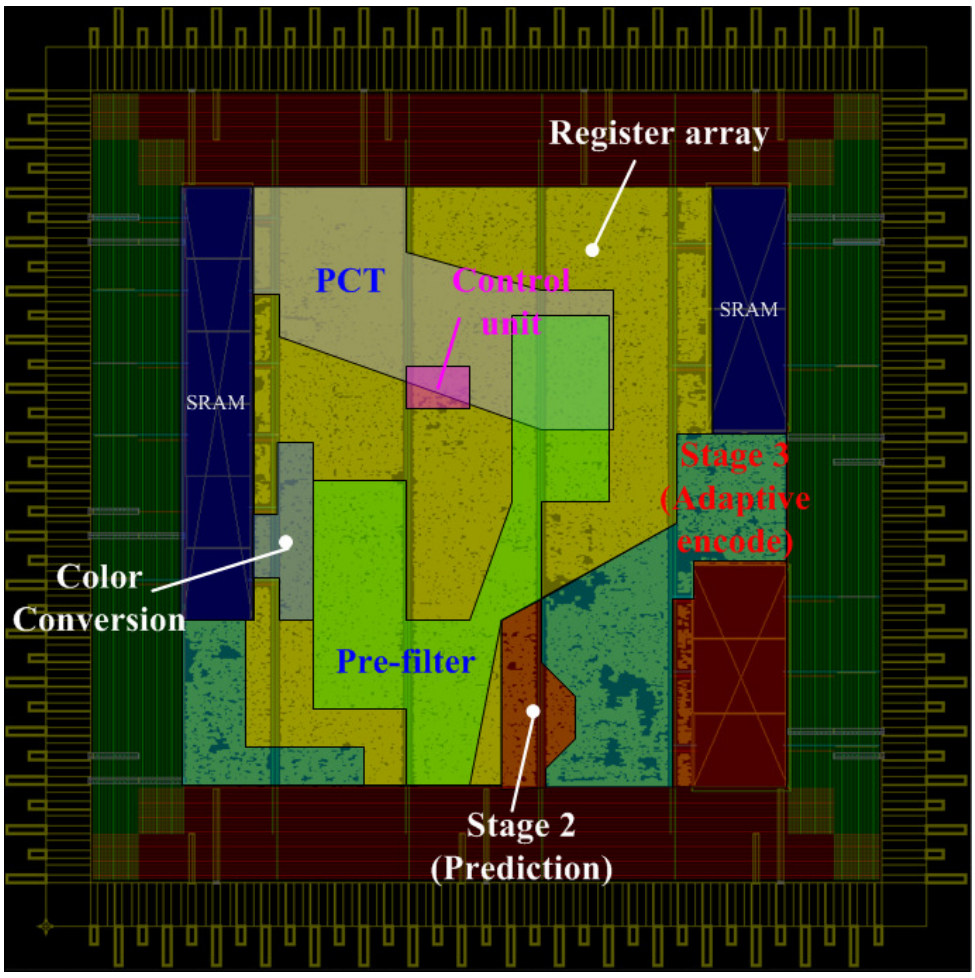


圖 4.5：更細部的合成資源分配圖

4.5 功率分部

在功率方面，總共消耗 368.7mW，每個單元的功率分部如圖 4.6，在管線階段 2 的地方消耗的比 Profiling 所分析的結果還要大很多的原因跟 4.4 節所論述的一樣，但運算量還是所有演算法中最低的，所以消耗的功率也最低，管線階段 1 的部份做更詳細的分析，為了資料重複技術所建立的暫存器陣列消耗了 80.7mW，算是比較大的比例，在圖 4.7 表示在不同頻率下所消耗的功率，最低點頻率及為每秒壓縮一張 HD 影像所消耗的功率。

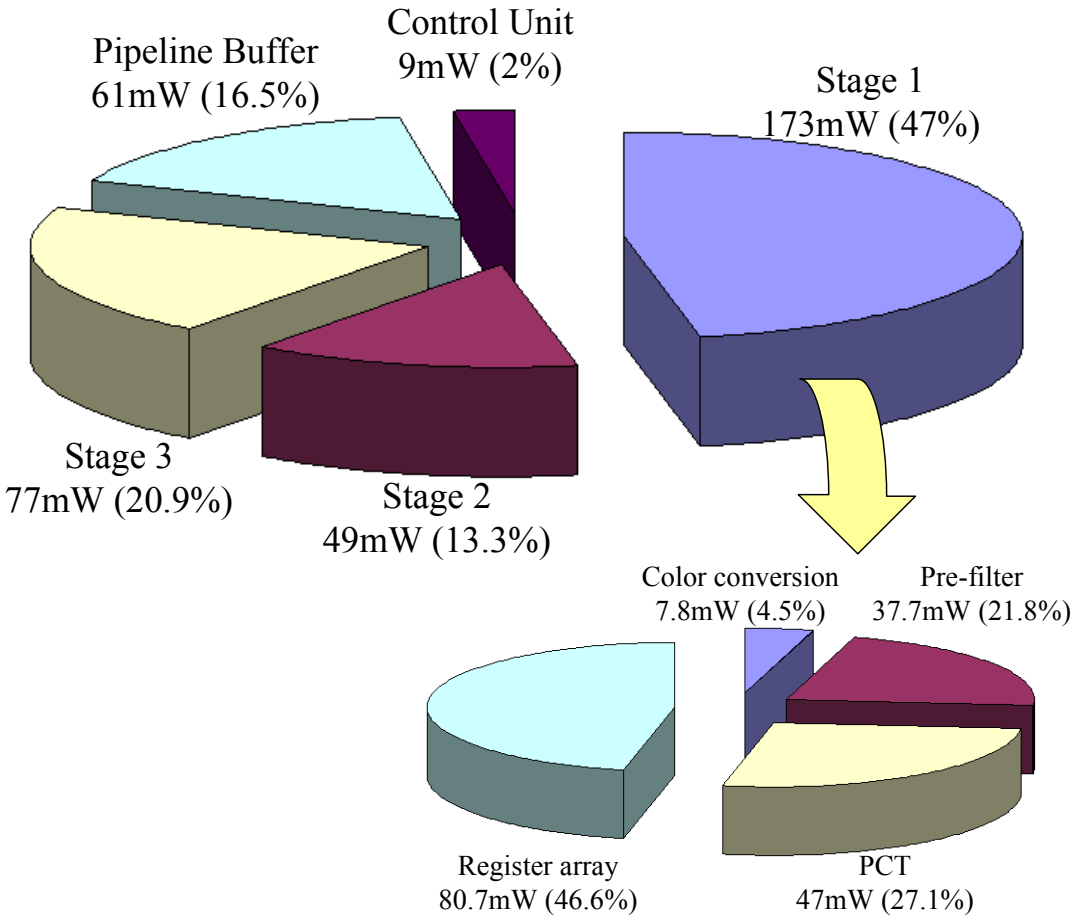


圖 4.6：功率分配圖

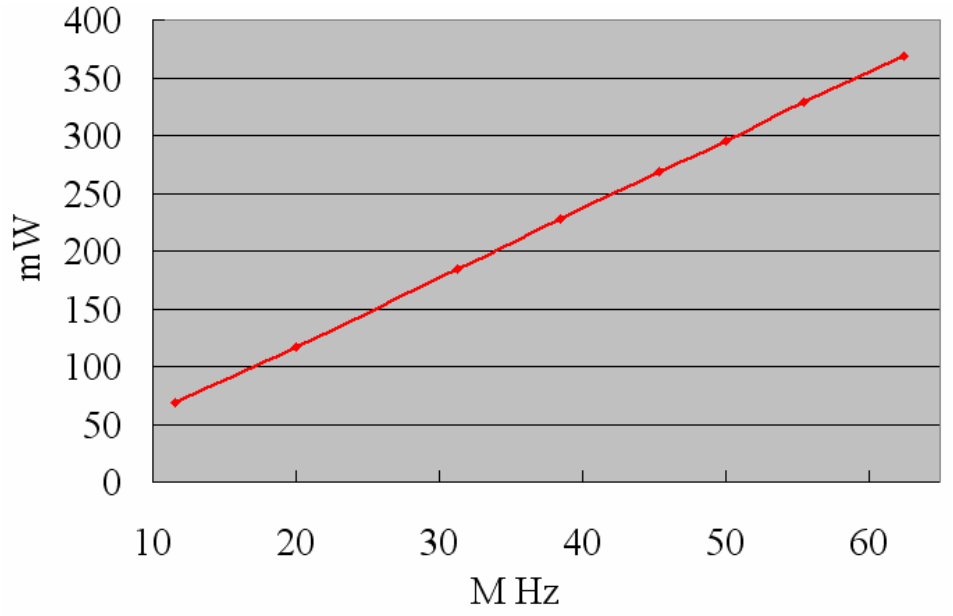


圖 4.7：功率分配圖

4.6 Bandwidth Profile

我們根據實做結果分析 Bandwidth profile 如圖 4.8。

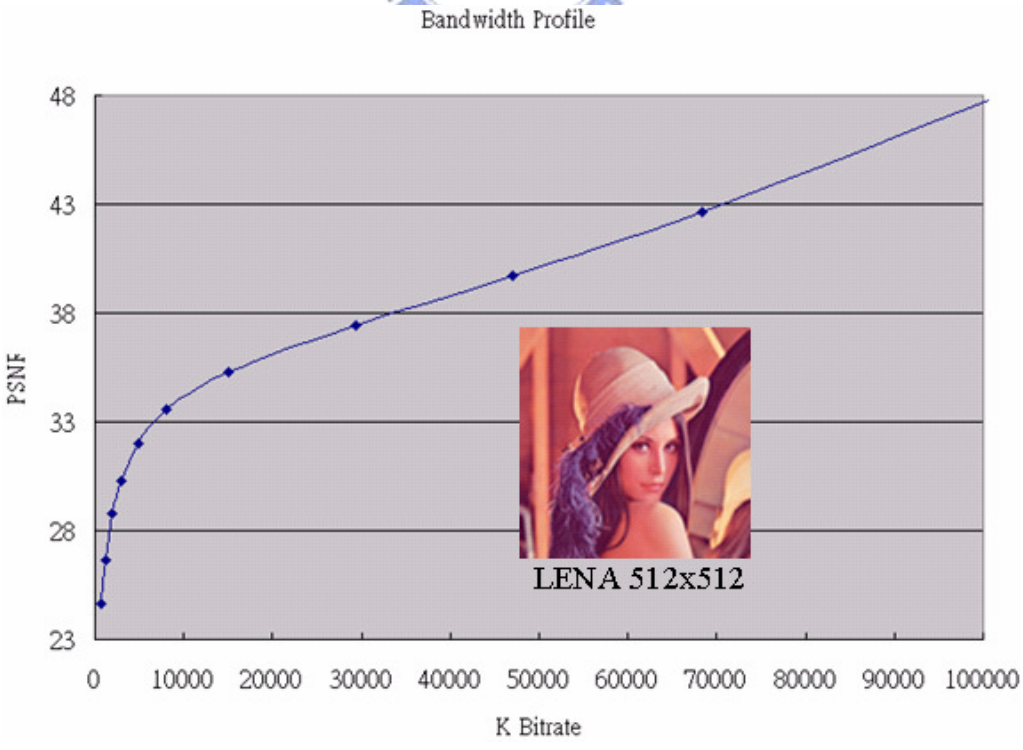


圖 4.8：Bandwidth Profile

4.7 晶片比較

表 4.2：晶片比較結果

	Liu[12] – JSSCC 2004	Hayashi[13] – ISCAS 2003	Proposed
Technology	DONGBU 0.18 2P5M	Hitachi 0.18 2P5M	TSMC 0.18 1P6M
Standard	JPEG2000	JPEG2000	JPEG XR
Area	$20mm^2$	$34.81mm^2$	$20.47mm^2$
Frequency	100M Hz	55M Hz	62.5M Hz
Power	450mW	N/A	368.7mW

我們提出的 JPEG XR 晶片與[12][13]比較的結果不分上下，跟據目前所知道的文獻記載，尚未有人對 JPEG XR 發表過相關的硬體架構設計文章，所以我們在此提出 JPEG XR 的硬體架構並與 JPEG2000 做比較，由之前實做的結果可以發現，因為前置濾波器所使用的暫存器陣列在面積上佔了相當大的比例(黃色區域)，在功率消耗上，也是花費相當多的比例，所以未來我們不排除將暫存器陣列使用 SRAM 來代替甚至使用外部 DRAM 來存放，如此即可省略三分之一以上的面積與四分之一的功率消耗，以符合一開始的軟體複雜度的模擬結果，未來我們將繼續改善現有架構，朝更低功率、更小面積的目標前進。

第五章

結論

本論文中，我們探討了 JPEG XR 與其它影像壓縮技術的比較，例如：JPEG XR、JPEG、JPEG2000 與 H.264 I-frame 的 PSNR 與相關特性，獲得不錯的結果，而 JPEG 在影像壓縮市場上已稱霸 15 餘年，取而代之的就會是以支援 HDR 影像與提供更好影像品質的 JPEG XR，根據目前的文獻記載，尚未有人發表過相關的硬體架構或相關的技術，所以在這提出全新的硬體架構以三階管線執行，由於前置濾波器的關係，處理區塊上會造成區塊順序的不相同並使用資料重複使用技術，可節省對 Off-chip memory 的存取頻寬，節省功率消耗與頻寬達 33.3%，熵編碼還包括 CBP 的處理，提出在進行資料預測時，就偵測目前係數的值，使得 CBP 能在管線階段 3 順利編碼，根據 Profiling 的分析，在適應性編碼部份所花費的執行時間達 65%，在實做上我們考慮到晶片面積，所以在執行適應性編碼時，提出一簡單有效率的架構以三階超管線執行，增加硬體使用率與增加整體效能，最後將 JPEG XR 編碼器以元件庫設計方式實現，實做結果與數據也在上章節呈現。

參考文獻

[1] http: /www.microsoft.com/whdc/xps/wmphoto.mspx

[2] Bill Crow, “Windows Media Photo: A new format for end-to-end digital imaging”, Windows Hardware Engineering Conference, Seattle, 2006

[3] Daniele D. Giusto and Tatiana Onali, “Data Compression for Digital Photography: Performance comparison between proprietary solutions and standards” International Conference on Consumer Electronics, pp. 1-2, Jan. 2007

[4] Dmitriy Vatolin, Alexey Moskvin and Oleg Petrov, “Windows Media Photo and JPEG 2000 Codecs Comparison”, CS MSU Graphics & Media Lab Video Group, Aug. 2006

[5] CCIT T.81, “Information Technology – Digital Compression and Coding of Continuous-Tone Still Images-Requirements and Guidelines”

[6] Michael D. Adams, “The JPEG-2000 Still Image Compression Standard”, ISO/IEC JTC 1/SC 29/WG 1N 2412, Dec. 2002

[7] Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi, “The JPEG 2000 Still Image Compression Standard” IEEE Signal Processing Magazine, Vol. 18, pp. 36-58, Sep. 2001

[8] 吳炳飛, 胡益強, 瞿忠正, 蘇崇彥, JPEG 2000 影像壓縮技術, 初版, 全華科技圖書公司, 台北市

[9] Yu-Wen Huang, Bing-Yu Hsieh, Tung-Chien Chen and Liang-Gee Chen, “Analysis, Fast Algorithm, and VLSI Architecture Design for H.264/AVC Intra Frame Coder”, IEEE Transactions on Circuits and Systems for Video Technology, vol. 15, no. 3, pp. 378-401, Mar. 2005.

[10] Agostini, L.V.; Silva, I.S.; Bampi, S.; “Pipelined entropy coders for JPEG compression” Integrated Circuits and Systems Design, 2002. Proceedings. 15th Symposium pp. 203-208, Sep. 2002

[11] Ching-Yen Chien, Sheng-Chieh Huang, Shih-Hsiang Lin, Yu-Chieh Huang, Yi-Cheng Chen, Lei-Chun Chou, Tzu-Der Chuang, Yu-Wei Chang, Chia-Ho Pan, and Liang-Gee Chen; “A 100 MHZ 1920x1080 HD-PHOTO 20 FRAMES/SEC JPEG XR ENCODER DESIGN”, IEEE International Conference on Image Processing(ICIP), Oct. 2008

[12] Leibo Liu, Ning Chen, Hongying Meng, Li Zhang, Zhihua Wang and Hongyi Chen; **“**An efficient VLSI architecture for JPEG2000 encoder”, IEEE Journal of Solid-State Circuit, Vol. 39, pp. 2032-2040. Nov. 2004

[13] Hayashi Y., Tsutsui H., Masuzaki T., Humi T., Onoye T. and Nakamura Y., “Design framework for JPEG2000 encoding system architecture”, International Symposium on Circuits and Systems (ISCAS), Vol. 2, pp. 25-28, May. 2003