國立交通大學

資訊工程系

碩 士 論 文

在雙核心平台上進行 H.264 視訊壓縮的最佳化

H.264 Video Encoding Optimization on Dual-Core Platform

研 究 生：邱正男

指導教授：蔡淳仁　博士

中 華 民 國 九 十 四 年 六 月

在雙核心平台上進行 H.264 視訊壓縮的最佳化
H.264 Video Encoding Optimization on Dual-Core Platform

研 究 生：邱正男　　Student：Cheng-Nan Chiu

指導教授：蔡淳仁　　Advisor：Chun-Jen Tsai

國 立 交 通 大 學

資 訊 工 程 系

碩 士 論 文

A Thesis

Submitted to Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science and Information Engineering

June 2005

Hsinchu, Taiwan, Republic of China

中華民國九十四年六月

國　立　交　通　大　學

博碩士論文全文電子檔著作權授權書

（提供授權人裝訂於紙本論文書名頁之次頁用）

本授權書所授權之學位論文，為本人於國立交通大學 資訊工程 系所 ＿＿＿＿＿＿組， 93 學年度第_2_學期取得碩士學位之論文。

論文題目：在雙核心平台上進行 H.264 視訊壓縮的最佳化

指導教授：蔡淳仁 博士

■ 同意　□不同意

本人茲將本著作，以非專屬、無償授權國立交通大學與台灣聯合大學系統圖書館：基於推動讀者間「資源共享、互惠合作」之理念，與回饋社會與學術研究之目的，國立交通大學及台灣聯合大學系統圖書館得不限地域、時間與次數，以紙本、光碟或數位化等各種方法收錄、重製與利用；於著作權法合理使用範圍內，讀者得進行線上檢索、閱覽、下載或列印。

論文全文上載網路公開之範圍及時間：	
本校及台灣聯合大學系統區域網路	■ 中華民國 94 年　8 月 30 日公開
校外網際網路	■ 中華民國 94 年　8 月 30 日公開

授 權 人：邱正男

親筆簽名：＿＿＿＿＿＿＿＿＿＿＿＿

中華民國　94 年　8 月　30　日

國家圖書館博碩士論文電子檔案上網授權書

ID:GT009117561

本授權書所授權之論文為授權人在國立交通大學 電機資訊 學院 資訊工程 系所 _____ 組 _93_學年度第_2_學期取得碩士學位之論文。

論文題目：在雙核心平台上進行 H.264 視訊壓縮的最佳化

指導教授：蔡淳仁 博士

茲同意將授權人擁有著作權之上列論文全文（含摘要），非專屬、無償授權國家圖書館，不限地域、時間與次數，以微縮、光碟或其他各種數位化方式將上列論文重製，並得將數位化之上列論文及論文電子檔以上載網路方式，提供讀者基於個人非營利性質之線上檢索、閱覽、下載或列印。

※ 讀者基於非營利性質之線上檢索、閱覽、下載或列印上列論文，應依著作權法相關規定辦理。

授權人：邱正男

親筆簽名：_____

民國　　94 年 8 月　30　日

國立交通大學

論文口試委員會審定書

本校 資訊工程系 碩士班 邱正男 君

所提論文:

H.264 Video Encoding Optimization on Dual-Core Platform

在雙核心平台上進行 H.264 視訊壓縮的最佳化

合於碩士資格水準、業經本委員會評審認可。

口試委員：

指導教授：

系主任：

中 華 民 國 九十四 年 六 月 二十二 日

在雙核心平台上進行H.264視訊壓縮的最佳化

學生：邱正男　　指導教授：蔡淳仁 博士

國立交通大學資訊工程學系 (研究所) 碩士班

摘 要

本論文的研究重點是在於利用雙核心平台的特性來對視訊壓縮器的架構做適當的調整。雙核心的晶片通常在手機的基頻處理器(Cellular Baseband Processor)中被廣泛利用，這個基頻處理器通常是由 RISC 核心和 DSP 核心所組成，利用彼此間不同的硬體架構特性相互合作來完成所需要處理的工作。影像壓縮處理演算法是一向非常耗費計算量的技術，當我們考慮想要在類似的雙核心平台上實現，我們思考的方面應該不光只是演算法本身的計算特性，而是必須把處理器面臨的狀況也考慮進來。現有的視訊壓縮器的軟體架構運算重心是以 DSP 核心爲主，但是隨著 RISC 核心的強化以及 DSP 核心處理通訊協定的負擔加重，RISC 也足夠的能力處理類似的視訊壓縮演算法。本論文提出一個獨特的視訊壓縮器架構，利用這個新的架構，可以充分利用 ARM 與 DSP 的實際運算能力來達到加速影像壓縮處理的效能。

H.264 Video Encoding Optimization on Dual-Core Platform

Student: Cheng-Nan Chiu Advisor: Dr. Chun-Jen Tsai

Institute of Computer Science and Information Engineering
National Chiao-Tung University

Abstract

The research in this thesis focuses on appropriately adjusting the architecture of video encoder according to the characteristic of dual-core platform. Cellular Baseband Processor of mobile phones, which consists of RISC core and DSP core, usually adopts dual-core chips to process jobs via cooperating of the different hardware characteristics between two cores. The algorithms for video encoding are always a kind of extremely computing-consuming technology. When we consider implementing those algorithms on dual-core platforms, we must take not only the computing characteristics of the algorithms themselves but also the processing conditions on processors into account. The mainstream to build up software computing architecture in video encoder is to use DSP core but because of the strengthening of RISC core and the DSP core's heavy loading of processing communication protocol, RISC also affords to process the algorithms of video encoding. A specific and unique architecture for video encoder is proposed in this thesis. Adopting this new architecture, the good computing abilities of ARM and DSP would be fully used to speed video encoding processing.

致謝

能完成這篇論文，我必須感謝很多人。首先要感謝的是：我的指導教授 蔡淳仁 博士，在我碩士的求學生涯中的細心教導，讓我不僅在課業上有所突破，老師在研究上的嚴謹態度，更是我所要效法的。另外還必須感謝我的家人，給予我經濟上的資助還有精神上的鼓勵，讓我沒有後顧之憂的完成學業。最後要感謝交大資工「嵌入式多媒體實驗室」的學長和學弟妹，由於他們的幫助，讓我能快速的解決研究上所遇到的瓶頸。

目錄索引

圖表索引

表格索引

1. 簡介 (Introduction)

1.1. 論文題目和研究重點

本論文的研究題目是”在雙核心平台上進行 H.264 視訊壓縮的最佳化 (H.264 Video Encoding Optimization on Dual-Core Platform)”。主要的內容是研究 H.264 這個視訊壓縮器在含有 RISC 與 DSP 的環境下，如何能充分利用這兩種不同的運算單元來達到即時影像壓縮的目的。H.264 是新一代的視訊壓縮標準，擁有很好的壓縮效率，足以在現今的網路技術下達到比較好的影像要求。H.264 雖然擁有很好的壓縮率，但是它也需要大量的資料運算，一般的嵌入式平台受限於其運算能力，所以很難達到即時壓縮的效果。

本論文實驗所用的是 TI (德州儀器)的 OSK5912 發展平台，OMAP5912 採用的是 ARM 和 DSP 的雙核心架構。透過兩種不同類型的處理器的合作，來讓需要大量運算的 H.264 壓縮器能在此平台上實現。

1.2. 研究方法和預期的目標

H.264 是一個複雜度相當高的視訊壓縮標準，它需要大規模的計算和大容量的記憶體使用空間。可是現有的 H.264 參考軟體 JM9.6，並不適合在嵌入式的環境下執行，爲了達到即時壓縮的效果，決定重寫一個比較精簡的 H.264 視訊壓縮器，並經由程式流程的控制、記憶體的管理和有效的簡化演算法的複雜度，來達到時間上的需求。所以我們的實驗步驟分爲：

1. 重新設計 H.264 的資料結構。
2. 移植到 OMAP 平台上。
3. 參考 OMAP 平台的特性，適當的修改程式架構。

經過這樣的步驟，除了希望 H.264 在 OMAP 上有良好的效能外，並能達到以下的幾個目標：

1. RISC 與 DSP 核心會以緊密結合(tightly-coupled)的方式，分配計算工作。
2. 降低兩個核心間因爲溝通協調所浪費的效能。

1.3. 論文各章節介紹

在第二章中，我們會簡單介紹一些雙核心的平台，並定義出什麼是 Loosely-Coupled 與 Tightly-Coupled 的工作分配機制，並引出本篇論文研究的動機。第三章說明一些影像壓縮常用的演算法，以及簡單介紹 H.264 的緣由以及壓縮演算法的特性。第四章是實驗所用的軟硬體平台的介紹和設定，將詳細說明擁有緊密結合特性的視訊壓縮器在 OMAP 上的軟體架構及實驗環境的設定。第五章列出實驗的數據，並針對數據及 Tightly-Coupled 的架構作深入的分析。第六章則是研究結果的討論和未來的展望。

2. 緒論 (Previous work)

2.1. 研究背景

隨著無線網路與手機的快速發展，未來的頻寬與使用著的需求越來越高，除了滿足人們對傳統語音通訊之外，包括圖形、影像、動畫等多媒體的應用顯的格外重要。這些多媒體的應用往往需外大量的運算，因此傳統上單一處理器的系統已經無法有效負擔這些需求，特別是一些具有即時性的應用。爲了應付這些日趨重要的多媒體任務，需要多個處理核心平行工作來提供使用者所期望的服務品質。

通常這類平台的處理核心都是非同質性(heterogeneous)的，包括一般目的處理器(GPP，General-Purpose Processor)，如 ARM、MIPS 等，及一些專門用來處理多媒體任務龐大運算的處理器。這些多媒體處理核心的選擇包括：數位訊號處理器(DSP，Digital Signal Processor)、可程式化的硬體加速器(如 FPGA)或者是特定對象的硬體加速器(如 Video Accelerator)。

這些多媒體處理核心都有他們自己的特色，例如 DSP 和 FPGA 都是可程式化的，適用於一般的多媒體處理；硬體加速器(ASIC)所針對的對象往往都是固定的，具有比較高的效率但是缺乏設計的彈性。

2.2. 相關研究

在這一小節中，我們會介紹一些雙核心的平台，這裡所指的雙核心是指 GPP 和 DSP。目前產業界推出許多雙核心的平台架構，這種雙核心的架構具有較高的處理效率及設計彈性。但是核心與核心之前的溝通合作需要程式設計人員或作業系統來在調配，不但增加作業系統的複雜度，也考驗程式設計人員的專業知識。

2.2.1. TI - OMAP

開放式多媒體應用平台(OMAP，Open Multimedia Application Platform)是 TI 所推出在多媒體應用上的整合晶片，OMAP 採用一種獨特的雙核心架構，把控制性能較強的 ARM 核心和高效能低功耗的 DSP 核心結合。主要目標是滿足 2.5G 和 3G 的手持式

電話、PDA 上的語音和多媒體需求。OMAP 硬體平台主要是由 DSP 核心、ARM 核心和記憶體控制單元(Traffic Controller)所組成，這三個部分可以獨立地進行工作頻率的調整，可以有效的控制耗電量。雙核心的硬體架構技術可以提升作業系統的效率和多媒體程式的最佳化。

ARM9 的 RISC 核心適合處理一般性的控制工作，包括使用者介面、作業系統…等；而 C55x 的數位訊號處理核心具有強大的數學運算能力和搭配在核心內的快速記憶體，適合用來處理一些具有即時性的大量多媒體資料處理，利用 DSP 核心來解決大部分的多媒體運算，所以具有高度的程式設計彈性。它也提供一些影像處理的硬體延伸指令 (如：DCT/iDCT，Interpolation，Motion Estimation)，幫助處理龐大的影像壓縮/解壓縮的運算。這樣分別利用了 DSP 的低功耗的較強的數據處理能力和 ARM 的較強控制能力的優勢，與傳統只利用 ARM 或 DSP 單晶片的系統相比，OMAP 成功地解決性能與功耗的最佳組合問題。下圖是 OMAP1510 的系統架構圖：

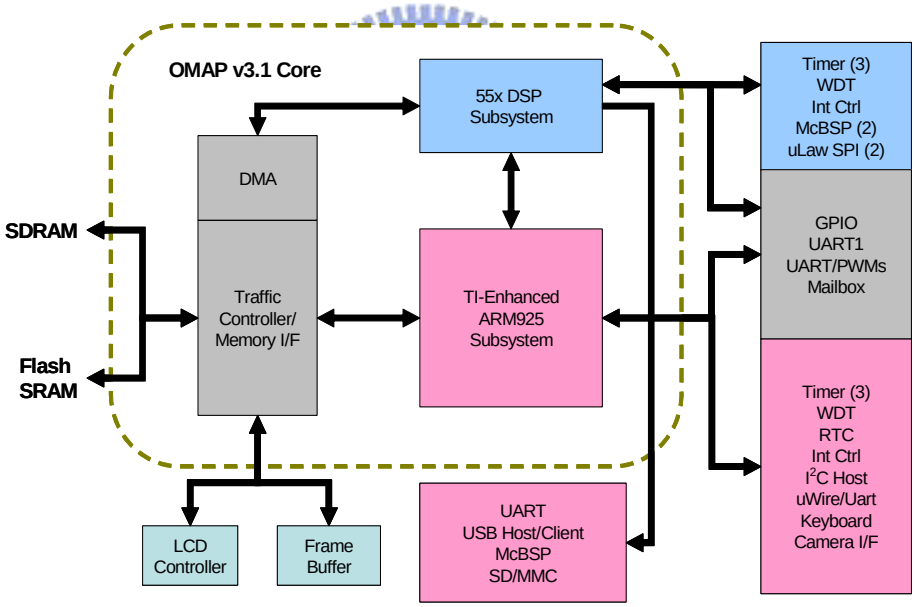


Fig. 1 OMAP1510 hardware architecture

2.2.2. Intel - PCA

爲了應付行動內容及無線網路的資料的急速發展，Intel 訂定了 PCA (Personal Internet Clint Architecture)架構來迎合市場的需求，特別是針對目前通訊和多媒體應用整合的裝置所設計。例如爲了無線網路市場所推出的 Intel PXA800F 處理器[2]， PCA

架構把通訊和應用分開，分別利用不同的處理器來處理。這個高效能低功耗的處理器架構整合包括了 Intel XScale 核心，晶片內的記憶體系統及 Intel 的 MSA (Micro Signal Architecture)核心，適合現在的 GSM/GPRS 手機系統，並具備有快速方便的軟體開發環境。晶片中的 Intel XScale 處理核心最高能運行到 312MHz，具備有資料快取及指令快取，並整合了解析度 120x240 的 LCD 控制器。 MSA[2]是一個由 Intel 和 ADI (Analog Devices) 合作開發的 DSP 處理核心，負責 GSM/GPRS 的基頻資料處理，具備哈佛快取架構、Dual-MAC 的深管線(deep-pipeline)技術及 104MHz 的處理速度。

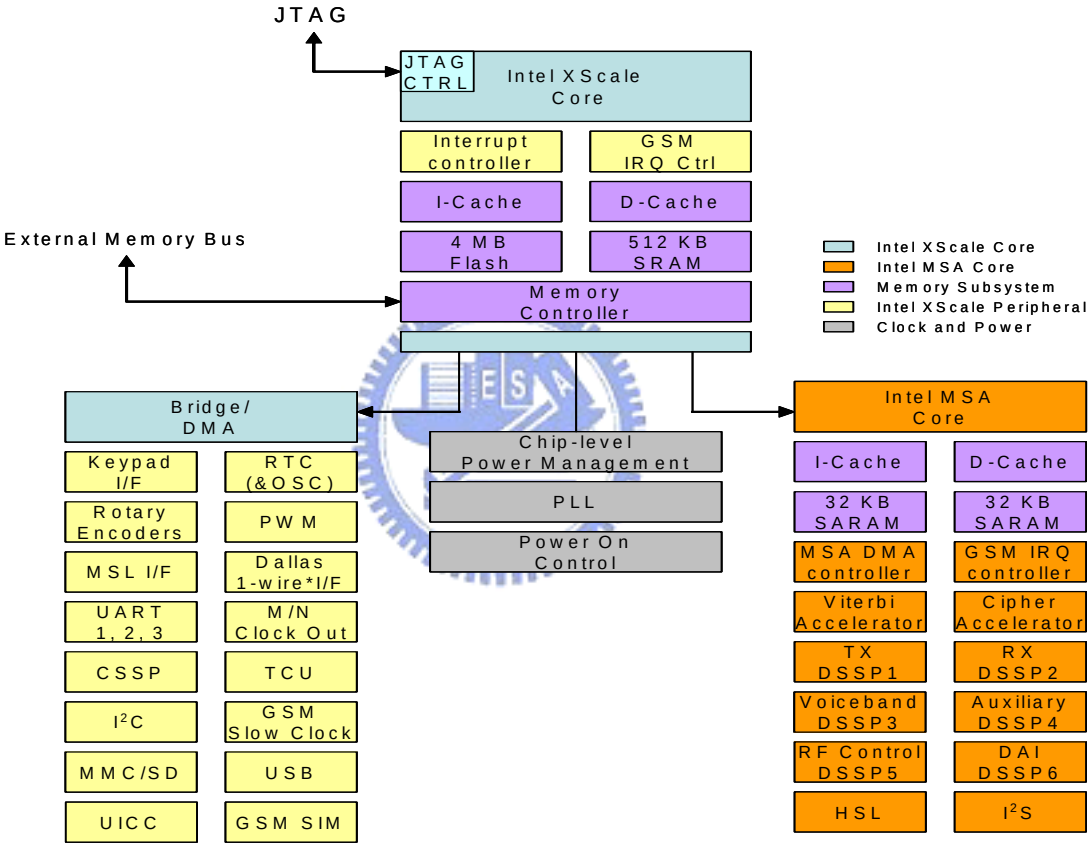


Fig. 2 Intel PXA800F hardware architecture

2.2.3. Freescale - MXC

Freescale (前身是 Motorola 的半導體部門)也推出了雙核心的平台架構，稱爲 MXC (mobile extreme convergence)架構，是爲了整合向 2D、3D 的圖形加速，行動多媒體，行動 IP 和個人化的娛樂環境等高階特性而設計。MXC 強調高整合性，可將應用和通訊整合在同一晶片內，並由不同的處理核心處理不同的工作，特別是針對手機平台的

設計，MXC 架構能減少百分支 50 的設計複雜度[4]。StarCore DSP 時脈頻率能達到 208MHz，並搭配四組 MAC 和兩個位址產生單元(address generation unit)，每個時脈週期能執行六組指令，這種搭配所提供高效能超越了基頻的處理需求底限。MXC 的專屬應用處理器是 ARM1136 核心，晶片內建 128KB 的 L2 快取和周邊元件，支援 400MHz 的運作頻率。該核心也是第一款搭載 L2 快取的 ARM 核心，雖然 ARM 核心和 DSP 核心共用相同的記憶體頻寬，但 L2 快取使 ARM 核心效能獲的有效的提升。

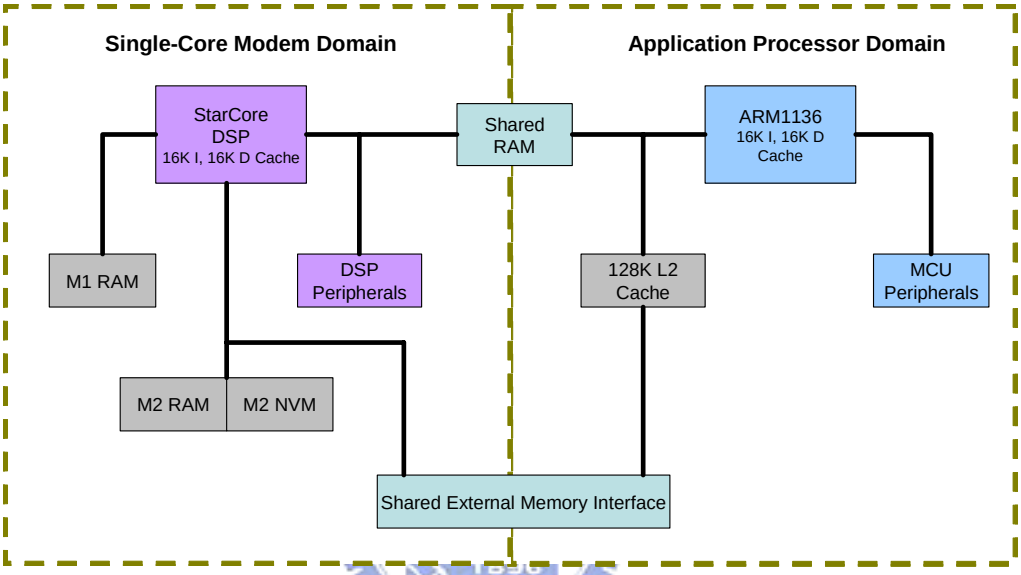


Fig. 3 Simplified Block Diagram of the MXC Host Processor

2.2.4. Philips – Viper

飛利浦(Philips)公司推出使用該公司 Nexperia PNX-8525 晶片的 Viper 機上盒平台[5]。包括一個 TriMedia TM32 的 DSP 核心(200MHz)和一個 150MHz MIPS 3940 RISC 核心。

TM32 是一個擁有超長指令集架構 VLIW(very long instruction word)的核心，透過這種特殊的指令集架構(ISA)，可以即時地處理關於多媒體方面的任務。TM32 最高可以將五個動作組合在一個 VLIW 的指令中，使的這五個動作能同時執行。TM32 擁有 32KB 的指令快取和 16KB 的資料快取。

RISC 的核心是採用 150MHz 的 PR3940，負責處理作業系統和控制的工作。他擁有很多的作業系統支援，包括 WindRiver、VxWork 和 Linux，與可以在 Windows CE

環境下發展程式。它支援 6 個階段(stage)的管線化作業(pipeline)，也擁有 16KB 的指令快取(I-Cache)和 32KB 的資料快取(D-Cache)。

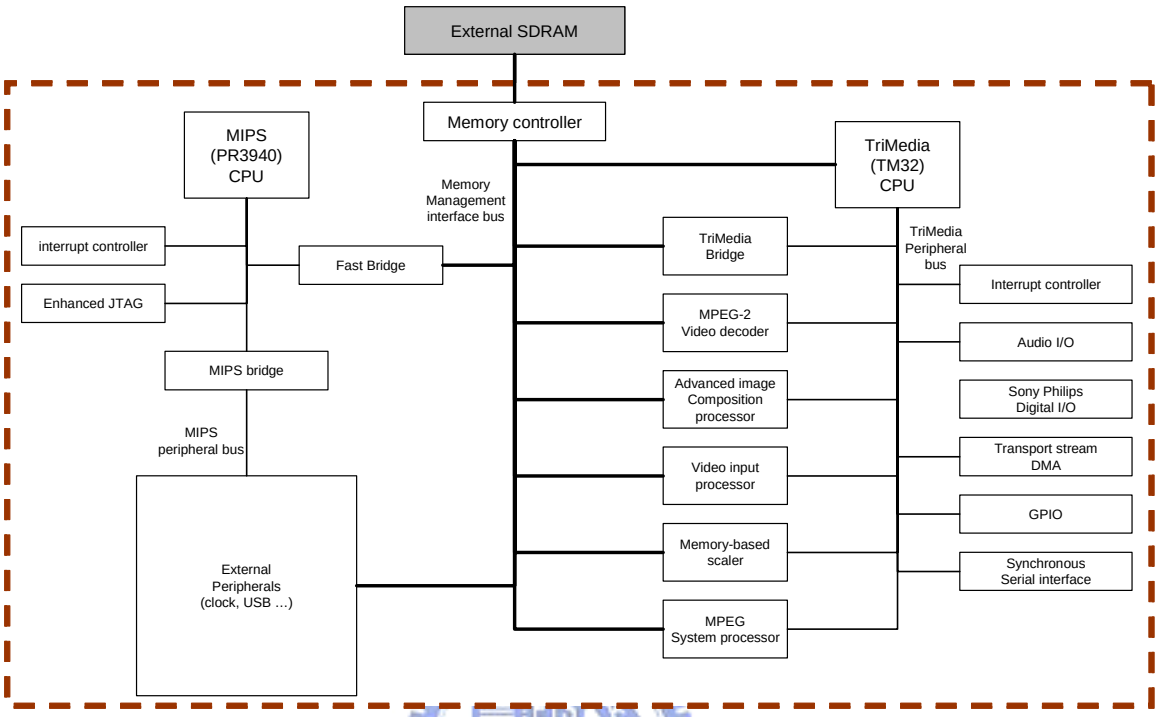


Fig. 4 Simplified block diagram of Viper

2.3. 論文研究目標

要實現兩個或以上平行的處理器分工，妥善的軟體整合十分重要，如何在現有的硬體架構下，利用軟體的分工來達到高效能的表現是一個很重要的課題。在目前的軟體設計環境趨勢中，大多根據資料處理的特性來進行工作的分配，而且這種工作的分配是離線的(off-time)，在編譯的階段就需要被決定了，決定哪些工作要在 GPP 上執行，哪些工作要在 DSP 上執行。但是每個使用者的使用習慣和使用環境都不盡相同，這樣事先分配的機制，反而會降低處理器的效率。例如有人在使用 3G 手機時會想要一邊錄影一邊聽 MP3，甚至當有視訊電話進來時，她會想要接起這個電話而不放棄之前的兩項工作。以一般的認知，錄影、聽 MP3 和看視訊，這些應該屬於 DSP 的工作，但是當使用者在做前兩項工作時，DSP 已經處於一個很忙碌的狀態，而我們又將視訊的工作再交給 DSP 做，DSP 很可能已經負擔不過來，但是此時的 GPP 甚至處於閒置的狀態。本論文即是在探討這個問題，我們希望我們的工作分配是在執行時根據兩個

核心間的狀況來分配的，當 DSP 忙碌時，GPP 也可以幫忙做些多媒體的工作，使得系統的設計變的更有彈性。以下我們定義了兩個新的名詞：*Loosely-Coupled* 和 *Tightly-Coupled*，Loosely-Coupled 即是在編譯階段(compile time)就決定工作分配的方式；Tightly-Coupled 強調的是在程式執行時(run time)根據各個核心間的狀態來分配工作的機制。

2.3.1. Loosely-Coupled Dual-Core System

以 Freescale 針對 MXC 架構所建議的智慧型手機的軟體架構爲例，利用 ARM1136 來執行作業系統以及使用者介面的控制；StarCore DSP 來處理一些具有即時性及需要大量運算的工作，如音訊的解壓縮、通訊協定(Layer1、2、3)的處理[32]。

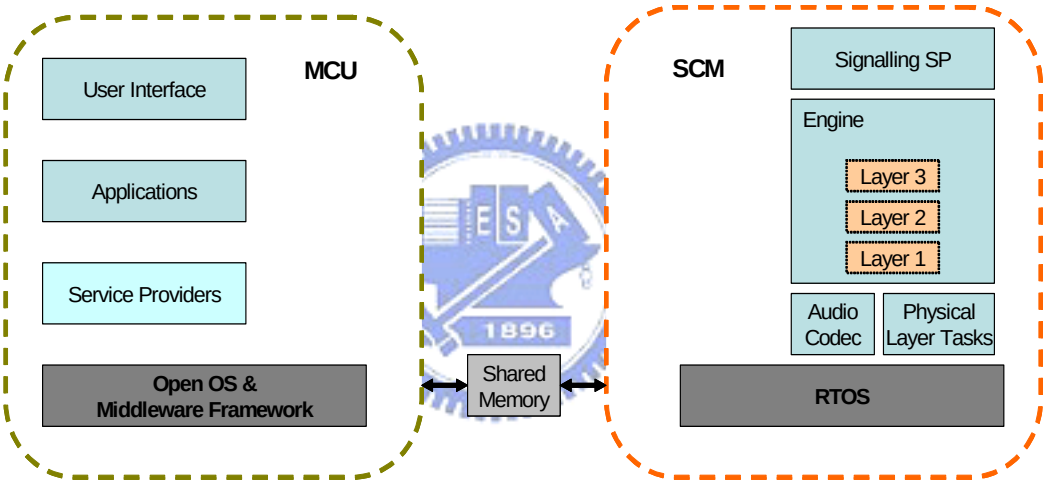


Fig. 5 Mobile Extreme Convergence Smartphone Software Architecture

這樣的設計具有一定的優點，因爲軟體切割的完整性，使的在設計上具有非常強烈的模組性。GPP 和 DSP 的設計流程是獨立的，有助於軟體的開發和除錯，而且具有高度的產品移植性，對於快速進入市場有相當程度的優勢。在雙核心平台上，利用資料處理的特性來決定該工作該由哪個核心來執行的排程，我們定義爲 **[Loosely-Coupled dual-core system]**。但是這種靜態排程潛藏著一些值得探討的問題，當 DSP 已經很忙碌時，我們是否適合把認爲該是 DSP 處理的工作再往 DSP 丟，而加重 DSP 的負擔，而且可能這時 GPP 正處於閒置的狀態，如圖 5 所示。

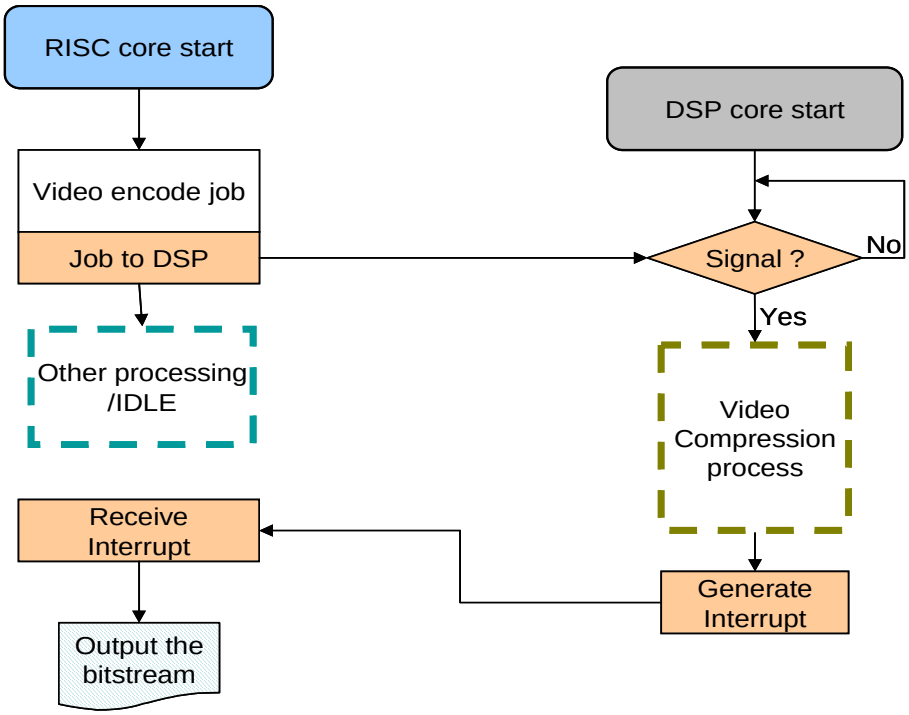


Fig. 6 video encode for loosely-coupled dual-core system

這種機制還有一個缺點，就是程式設計人員必須對多媒體任務的特性有相當程度的了解，他必須先經過相當多的資料分析和測試，才能決定哪些工作是適合 GPP 做的，哪些工作是適合 DSP 做的，加重了程式設計人員的負擔。

2.3.2. Tightly-Coupled Dual-Core System

在 Freescale 所建議的軟體架構中(圖 5)，我們可以發現 DSP 所需要負責的工作越來越多，包含當初是 MCU 所負責的 layer2、layer3[32]。如果在這個平台上我們想要增加一項多媒體播放或錄影的功能，按照多媒體特性的直覺，我們可能會將這一項任務分配給 DSP 來處理。但是這真的是洽當的嗎？

當我們考慮利用雙核心平台來進行需要大量的運算的視訊壓縮時，龐大的資料處理量(如 HDTV)可能不是 DSP 處理核心還有辦法負擔的，在不改變現有的硬體架構下，我們希望藉由運算能力日益強大的 GPP 的幫忙來達到使用者的要求。本論文針對多媒體視訊壓縮器提出 **[Tightly-Coupled dual-core system]** 的架構 (圖 6)，希望在既有的雙核心硬體平台環境下，利用 GPP 和 DSP 來平行處理視訊壓縮工作，來達到即時性的要求。

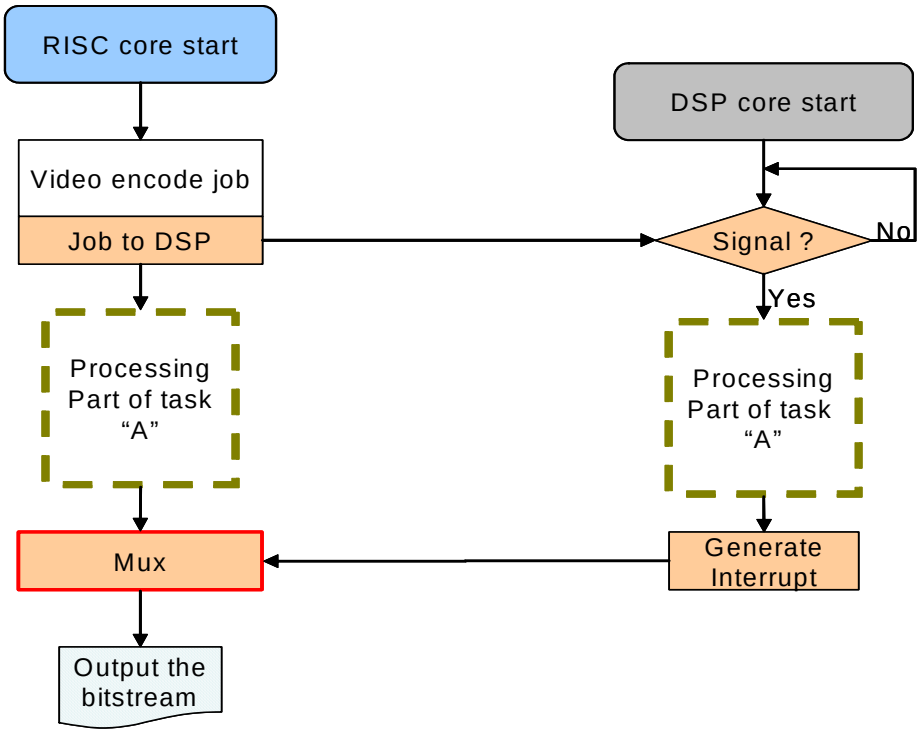


Fig. 7 Video Encoder for tightly-coupled dual-core system

3. 數位影像壓縮系統

3.1. 概論

一般而言，多媒體的資料都是很龐大的，所以要紀錄或是傳遞都需要很大的容量和頻寬，但是現有的資料儲存媒介(CD、DVD)和網路的頻寬都是有限的，所以資料壓縮是一種必要的技術。在日常生活中，我們也可以看到類似的例子：當我們想從甲地運送兩公噸的橘子汁到乙地，常用的手法便是先將橘子汁的水份分離出來，變成乾燥的橘子汁粉加上一張兩公噸的字條，然後將這些東西運送到乙地，乙地的人只要根據這張兩公噸的字條和橘子粉便能變回類似原來風味的橘子汁。這種手法是大多數人都可以接受的，而且也達到經濟的目的。在視訊影像方面，我們也是想透過類似的手法來達到減少資料量和頻寬的需求，以一個解析度爲 720x480、每秒 30 張的視訊影像(RGB)而言，在未經過壓縮之前每秒約 31MB 的資料量，換言之一張 700MB 的 CD 光碟片大概只能儲存 22 秒的畫面。所以我們對視訊的壓縮有強烈的需求，但是也要盡量達到無失真的程度以滿足使用著的需求。一般的視訊系統大概如下圖所示：

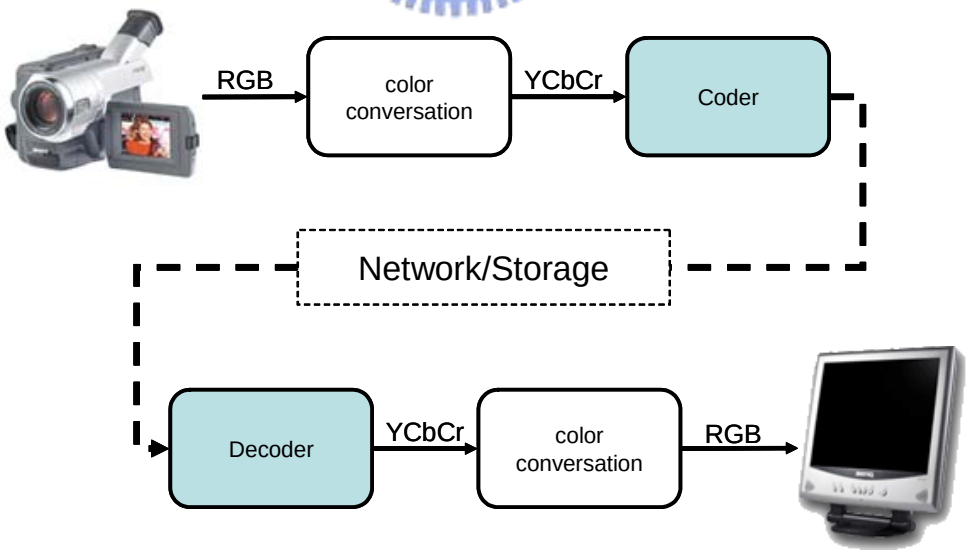


Fig. 8 Video System

當我們從機器擷取下來的影像，通常一個像素(Pixel)是由三個分量所組成的，分

別代表：紅色(Red)、綠色(Green)和藍色(Blue)三個原色。但是由於人眼對各個顏色的敏銳度不同，所以在影像處理上又定義了明度像素(Luminance Pixel，以 Y 表示)和色度像素(Chrominance Pixel，以 Cb 和 Cr 表示)[6]。藉由人眼對明度像素的高敏感度和色度像素比較不敏感的特性，我們將影像資料從 RGB 轉到 YCbCr 以符合人眼的感光特性。因爲人眼對色度像素比較不敏感的特性，所以我們會將色度像素作取樣 (sub - sampling)的動作，藉以達到資料壓縮的目的。

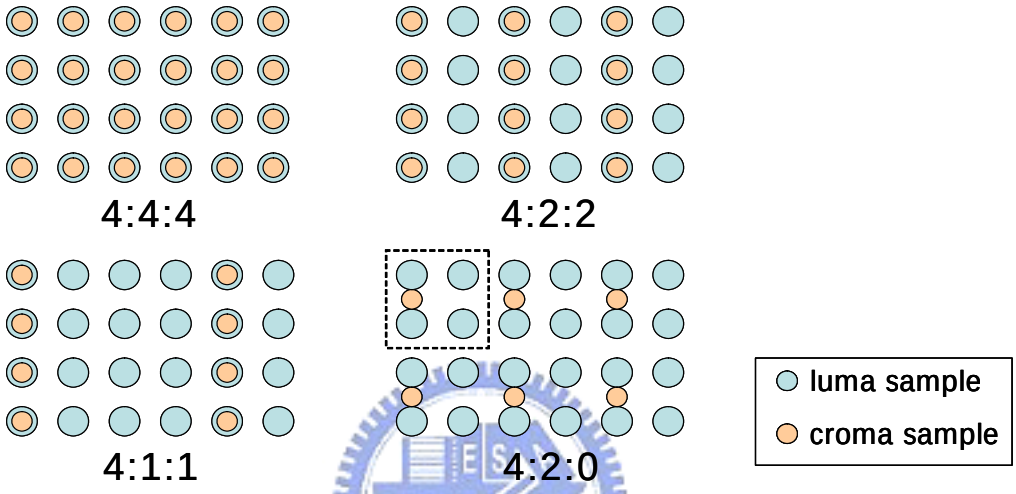


Fig. 9 Chrominance Pixel Sub-Smapling

這樣的動作(RGB 轉到 YCbCr420)可以達到壓縮一半資料量的效果，這是影像壓縮的第一步，也是影像失真的第一步，不過運用人眼的特性，這種失真是可以被接受的。本篇論文探討的方向都是在於影像格式爲 YCbCr420。

壓縮端的模組我們稱爲 Coder 或 Encoder，主要的工作便是去除影像資料間的空間上的資料多餘量(Spatial Redundancy)和時間上的資料多餘量(Temporal Redundancy)，把影像資料壓縮到最精簡的程度(橘子粉)，再透過網路或儲存媒介傳給解壓端。解壓縮端的模組我們稱爲 Decoder，主要的工作便是透過傳過來的影像壓縮資料(橘子粉)和訊息(字條)，把經過壓縮的影像資料一步一步的還原(橘子汁)。

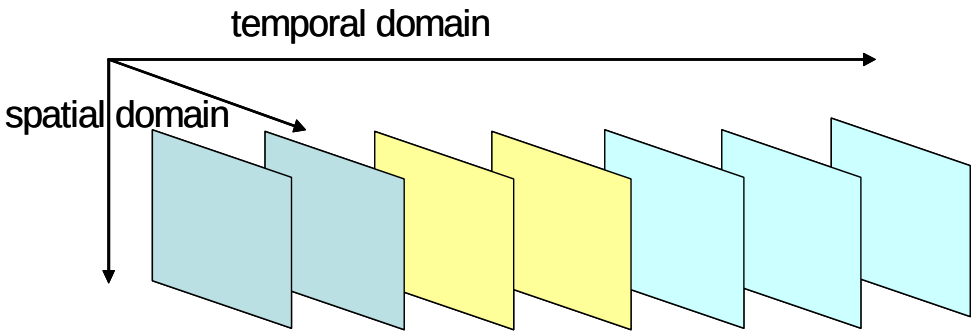


Fig. 10 Characteristics of video sequence

一般的視訊壓縮系統都是階層式的壓縮系統(Layer coding)，以一個畫面(frame)爲中心，連續的畫面組成序列(sequence)。當針對畫面作壓縮時，因爲每種畫面的解析度不盡相同，爲了壓縮效率和標準化作業，所以會把畫面切割成固定大小的巨區塊(macroblock)來做影像壓縮處理的基本單位。以 YCbCr420 的一般切分法是：明度(luminance)畫面的巨區塊是 16x16，兩個色度(chrominance)畫面相對應的便是 8x8 的區塊。

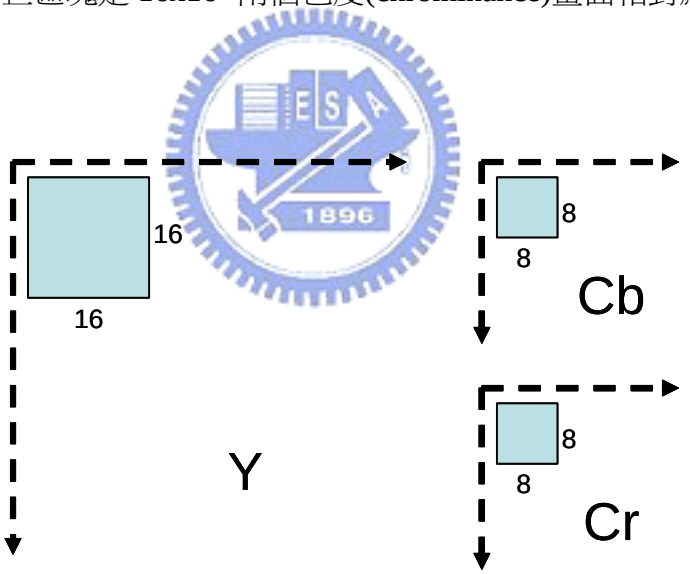


Fig. 11 Macroblock Layer

一般會把這編碼器和解碼器兩個模組合稱爲 Codec (Coder/Decoder)，在一般的編碼器中往往是由四個大的模組所組成，包括：預測編碼(prediction)、轉換處理(transform)、量化處理(quantization)和熵編碼(entropy coding)。這四個模組的介紹如下：

3.1.1. 預測編碼 (prediction)

假設有一連串的數列要壓縮:【0, 1, 2, 3, 4, 5, …】，如果我們要直接編碼的話，每

一個符號(symbol)至少要用 3 個位元來表示。但是換個方向思考，我們只需要告訴解碼端我們傳遞過去的是差值(residual)，接下來我們要處理的便是像這樣的數列：【0, 1, 1, 1, 1, 1, …】，每一個符號我們只需要用 2 個位元就可以表示了，每個符號就可以節省一個位元數，達到壓縮的效果。

相對於視訊序列中也包含許多這樣的特性，我們稱這種特性爲資料的多餘性(redundancy)，預測編碼的目的就是要去除這種資料的多餘性，讓視訊序列的資料能用最小的位元數來表示。視訊序列中的資料多餘性分爲兩大類，包括畫面內的空間多餘性(spatial redundancy)和畫面間的時間多餘性(temporal redundancy)。

想像我們所要壓縮的畫面是一扇藍色的大門，在這個大門內的每個像素值都是相似的，當我們要壓縮目前的像素值，可以考慮用這個像素周圍的像素來做預測，我們所需要做處理便是一些值域範圍較小差值(residual)。這種方法便是利用去除畫面內的資料多餘性來達到壓縮的目的。

在空間多餘性方面，考慮我們所擷取的畫面間，很多時候畫面與畫面之間會有許多重複的地方，例如我們在拍攝演講節目，畫面中除了演講人之外，其餘的背景幾乎都是不動的。一般當我們壓縮目前的畫面時，我們會考慮用巨區塊(macroblock)或是更小的區塊去前幾張的畫面作比對，找到一個最符合的區塊，我們所要壓縮的便是目前區塊與最符合區塊的差值。我們必須要傳遞過去給解碼端的資料包括：區塊差值(residual)和最符合區塊的位址(motion vector)。這種作區塊比對、尋找最佳區塊的動作，稱爲 Motion Estimation (ME)；而作區塊差值的運算稱爲 Motion Compensation (MC)。

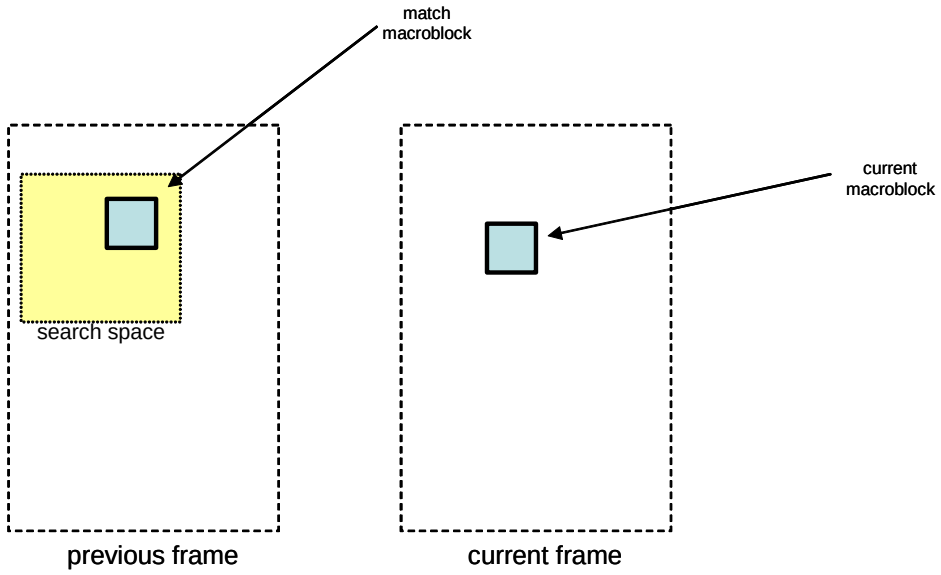


Fig. 12 block motion estimation

預測編碼模組的優點是能有效的提高壓縮率，但是在做資料比對、找出最佳的預測區塊的複雜度相當高，這是它的缺點之一。預測編碼基本上是建構在做爲預測的像素或是區塊是正確的條件上，但是如果是在網路的傳輸情況下，很有可能會有封包丟失(packet lose)或位元錯誤(bit error)的情況產生，這時就會有錯誤延續(error propagation)的情況出現。

3.1.2. 轉換處理 (transform)

轉換編碼是將區塊內的像素大小從時間域(time domain)轉到頻率域(frequency domain)，因爲人眼對低頻率的訊號比較敏感，對高頻的訊號比較不敏感。透過轉換基底的特性，可以把能量集中在我們感興趣的低頻上，忽略比較不重要的高頻信號來達到壓縮的效果。

一般在影像處理方面常見的轉換方式有兩種：DCT (discrete cosine transform)和 DWT (discrete wavelet transform) [7]。在視訊序列的壓縮標準上，一般還是以 DCT 爲主，透過 DCT 的矩陣轉換，會使能量集中在左上角(即低頻的區域)，我們稱頻率最低的 DCT 係數(即左上角的值)爲 DC Term，其他的稱爲 AC Term。DC Term 是最重要的一個係數，可以代表整個畫面的平均；其他 AC Term 的值通常會趨近於 0，這種現象在愈高頻的地方愈明顯。所以轉換處理後，在經過量化處理，會使高頻的區域出現一堆 0，透過特殊的排列方式(如 Zig-zag Scan)可以將這些 0 集中在一起，便很適合熵編

碼(entropy coding)的處理。

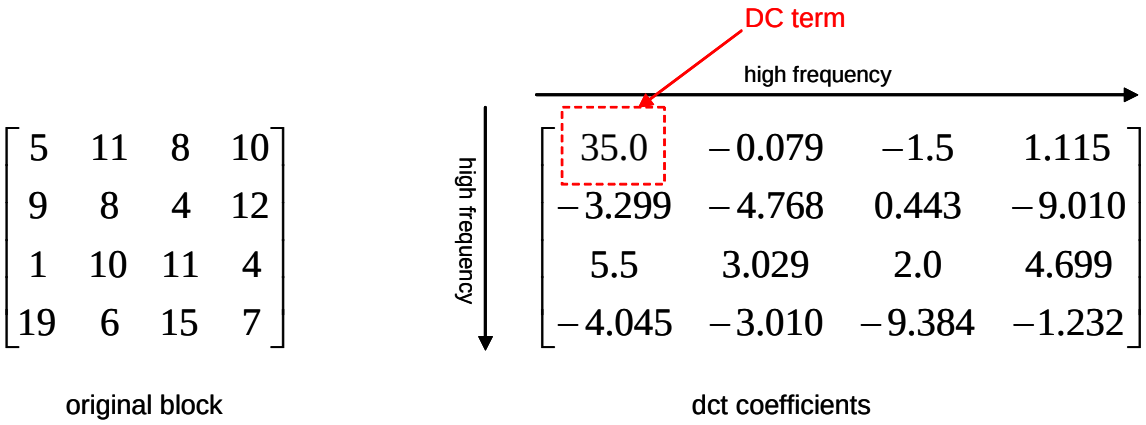


Fig. 13 DCT Transform

透過轉換處理，可以讓我們清楚地判斷哪些訊號是重要的，哪些是比較不重要的，之後利用不同的處理手法(quantization)，使的同樣的位元率，會表現出不同的影像品質。通常這類的轉換處理理論上是可逆的，即影像區塊經過轉換和逆轉換後是不會失真的，但是受限於電腦浮點數的精確度有限，尤其是一些嵌入式系統甚至是不支援運算較昂貴的浮點數運算，所以常常在逆轉換(inverse transform)後會出現失真的情況，稱爲【iDCT mismatch】。

3.1.3. 量化處理 (quantization)

在經過轉換處理後的值域範圍通常都會放大，放大的結果我們必須用更多的位元數去代表一個符號(symbol，coefficient)，反而會造成壓縮率降低。但是人眼對像素值的微小差異並沒有這麼敏感，舉例來說很少人能分辨出 254 跟 255 所代表的顏色像素。所以我們會加入量化處理來降低值域的範圍，例如原先是 [0, 1, 2, 3, 4, 5, 6, 7, …]的數列，經過量化單位(quantization step size)是 3 的量化處理，結果會變爲 [0, 0 ,0, 1, 1, 1, 2, 2, …]，經過還原後(inverse quantization)，會變爲 [0, 0, 0, 3, 3, 3, 6, 6, …]會有失真的情況產生，但是在不大量影響人眼視覺品質的前提下，又可以降低所需要的位元數增加壓縮率，大多數的壓縮標準都會採用這個處理模組來增加壓縮的效果。

經過轉換處理的係數(coefficient)值會落在一個很大的範圍中，量化處理可以把這些數值限制在某些數值中，如圖 13，使的後續的熵編碼更好處理。也可以利用量化處

理來處理經過轉換處理後較高頻的係數，因爲這些係數是比較不會影響解壓縮後的影像品質，所以利用量化處理把這些高頻數值盡量量化成 0，來提升壓縮率。

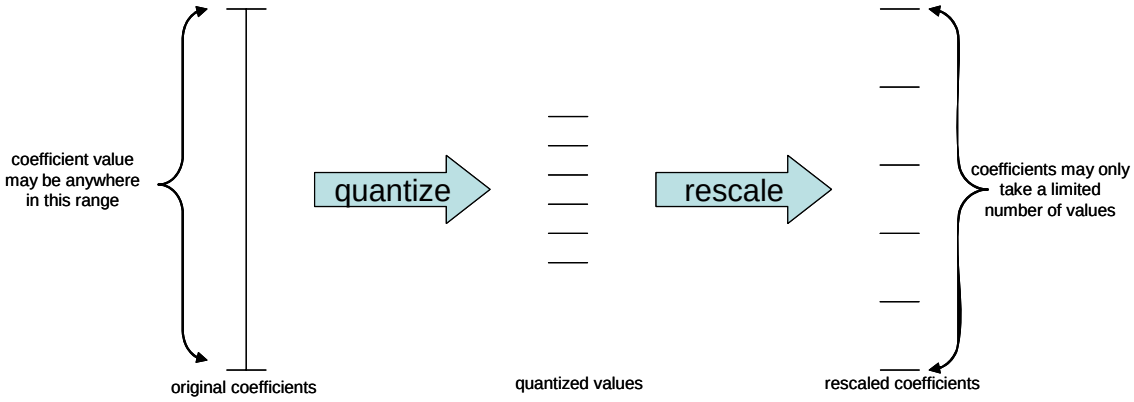


Fig. 14 Quantization and rescaling [8]

3.1.4. 熵編碼 (entropy coding)

編碼法可以分爲兩類：一種是固定長度的編碼法(FLC, fixed length coding)，另一種則是不固定長度的編碼法(variable length coding)。固定長度的編碼法使指每個符號用固定長度的位元數來代表；不固定長度的編碼法是利用每個符號出現的機率不盡相同，機率高的符號用比較少的位元數來代表，機率比較低的用較多的位元數來代表，使的壓完一連串的符號後能使用較少的位元數。最有名的不定長度編碼法就是霍夫曼編碼法(Huffman coding)，利用符號出現的機率來建立霍夫曼樹(Huffman tree)來編碼。但是在影像壓縮時，當我們還沒壓縮完整個的影像序列時，我們根本不清楚每個符號出現的機率，所以影像壓縮標準(video coding standardization)會先根據一般的狀況統計、分析來設計符號表，使的壓縮端和解壓端都能根據這些表來做熵編碼和解碼。

3.2. 視訊壓縮標準&H.264

3.2.1. 簡介

從 1990 年代開始，標準化的視訊壓縮標準開始被制定，目的是使工業界能遵循一定的規範，設計出能互相溝通的視訊壓器和解壓器。主要是由兩個國際組織所策劃，包括 ITU-T 的 VCEG(Video Coding Experts Group)和 ISO/IEC 的 MPEG(Moving Picture

Experts Group)。現有的國際標準視訊壓縮標準依時間先後順序有：H.261[9]、MPEG-1[10]、MPEG-2/H.262[11]、H.263[12]和 MPEG-4(Part 2)[13]…等。

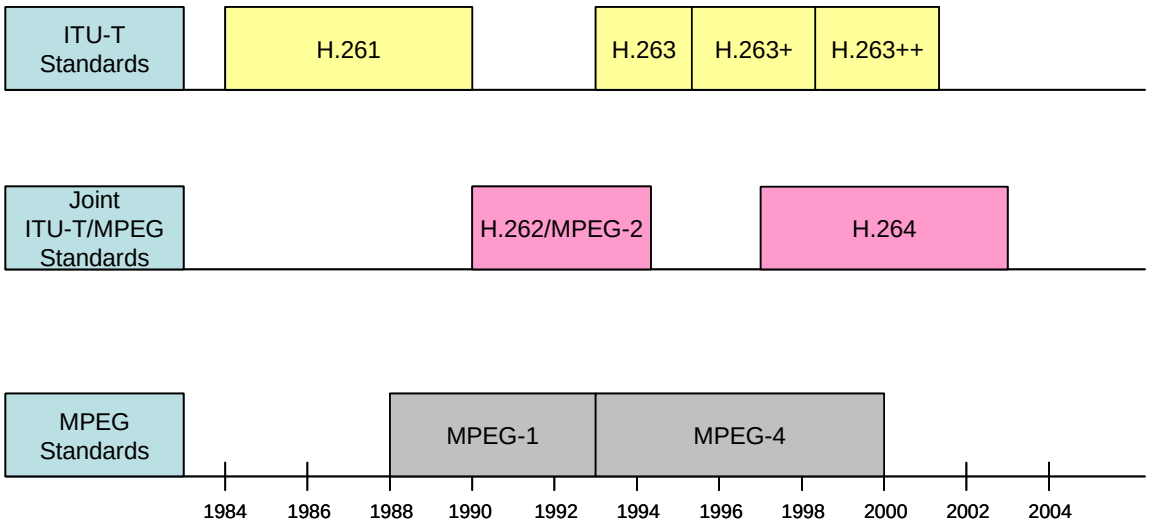


Fig. 15 Progression of the ITU-T and MPEG standards [18]

這些國際視訊壓縮標準所訂定的範圍大概都圍繞在解壓器(Decoder)的範疇內[14]，包括壓縮爲後位元流(bitstream)的語法(syntax)和解碼器中解壓演算法運作的先後順序，如下圖所示。

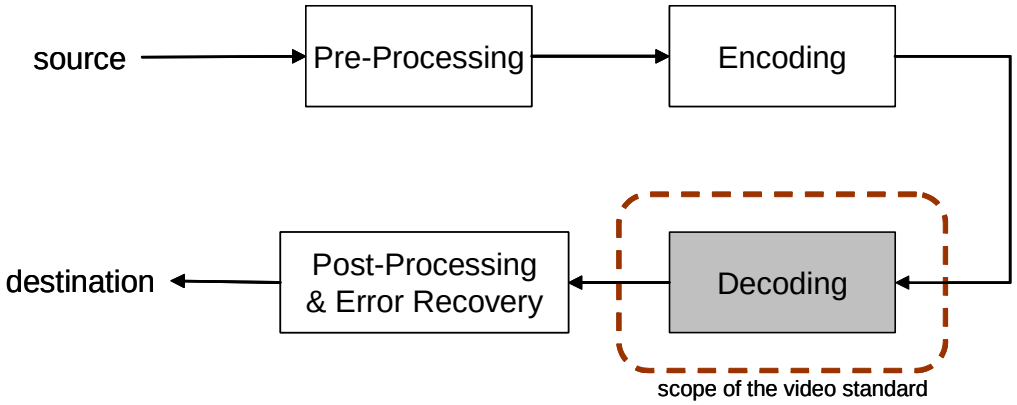


Fig. 16 Scope of video standardization

相對於解碼的過程，這些標準組織是不制定編碼演算法的，目的是爲了保持彈性。讓各家遵循這些標準的組織能自由發揮設計出自己的編碼器，但是又能讓各家的

位元流(bitstream)能互相解壓縮，讓這些各家設計的視訊壓縮器會有不同的影像品質，使的彼此會有良好的商業競爭基礎。

3.2.2. H.264

在 1998 年，VCEG 提出了企劃書的邀請(Call for Proposal)想要訂定一個名爲 H.26L 新的視訊標準，隔年完成了初步的版本。2001 年時，MPEG 完成了 MPEG-4 Part2[13] 的專案，也想要在訂定一各擁有更好的壓縮率的視訊壓縮標準，所以他們也向各界提出計畫書的邀請。VCEG 決定把他們的 H.26L 初步版本提到 MPEG 這個邀請中，並希望能一起合作訂定這一個新的標準。MPEG 比較過各個計畫書之後，決定選擇 H.26L 做爲基礎，並與 VCEG 合組一個團隊(Joint Video Term，JVT)共同開發下一世代新的影像壓縮標準。這個團隊在 2001 年 12 月成立並計畫在 2003 年完成標準的制定，這個標準的正式名稱在 VCEG 稱爲 H.264[15]，在 MPEG 中稱爲 MPEG-4 Part 10 Advanced Video Coding (AVC)[15]。

這個標準根據不同的應用環境設計出不同的 Profile. Baseline Profile 的設計包括具有較低的複雜度(low complexity)和較高的抗錯性(high robustness)，適合應用於網路視訊的環境；Main Profile 的設計強調擁有很高的壓縮率(high coding efficiency)；Extended Profile 則是將 Baseline Profile 的高抗錯性和 Main Profile 的高壓縮特性做一個結合。這個標準在 2004 八、九月時，又根據新的應用在做一次延伸(fidelity range extensions, FRExt)，增加了四個新的 Profile[16]。

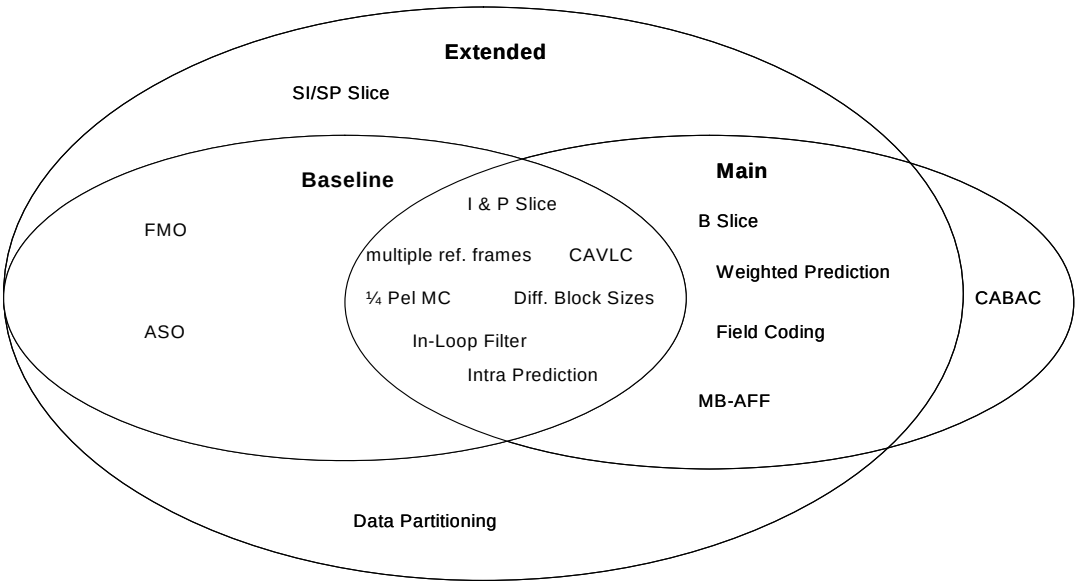


Fig. 17 H.264/AVC profiles and corresponding tools[17]

H.264 的在位元流(bitstream)的設計上，主要分爲兩個部分：NAL(Network Abstraction Layer)和 VCL(Video Coding Layer)，VLC 是儲存視訊畫面壓縮完後的資料，NAL 是用來包裝 VCL 和一些解壓縮的重要參數。一個 NAL 可以視爲在網路傳輸上的一個封包(Packet)，除了包裝 VCL 外，他也包括了 SPS(Sequence Parameter Sets)和 PPS(Picture Parameter Sets)這兩個解壓縮所需要的重要資訊。SPS 中包含視訊的應用範圍(Profile 和 Level)、畫面長寬等資訊。PPS 會指出熵編碼的格式(CABAC 或 CAVLC)、QP 的大小，及一需 VCL 內壓縮工具的設定(in-loop filtering、weighted prediction 等)。這些資訊(SPS 和 PPS)是重要的而且變動性是很小的，讓這些資料獨立出來不但可以增高壓縮率，在網路傳輸方面也可以利用可靠的機制(如：reliable channel)來保護這些重要資訊，增加系統的強韌性(robustness)。

H.264 和 MPEG 或 ITU-T 之前所設計的視訊標準一樣，都是採用一種稱爲混合型的區塊視訊壓縮法(block-based hybrid video coding approach)[14]，主要的壓縮架構如下圖 17 所示。在 VCL 的設計上，還是仿照舊有的視訊標準的設計，每張畫面切格成固定大小的巨區塊(macroblock)，這些巨區塊的大小在明度上(luminance，”luma”)是 16x16，在色度方面(chrominance, “chroma”)是 8x8 的大小(以 YCbCr420 爲例，本論文皆是以此格式爲討論範圍)。多個巨區塊會集合成 slice，一張畫面(picture)會切割成一個以上的 slice，slice 的大小由應用的環境所決定，在網路封包傳輸上，通常一個封包(Packet)即是一個 NAL unit，這個 NAL 封包會包含一個 slice 的資料，所以 slice 的大小大約等於一個封包的大小再減掉 VCL 前的一些標頭(header)資訊。但是如果是在儲存應用上，則一張畫面劃分爲同一個 slice，會擁有較好的壓縮效率。

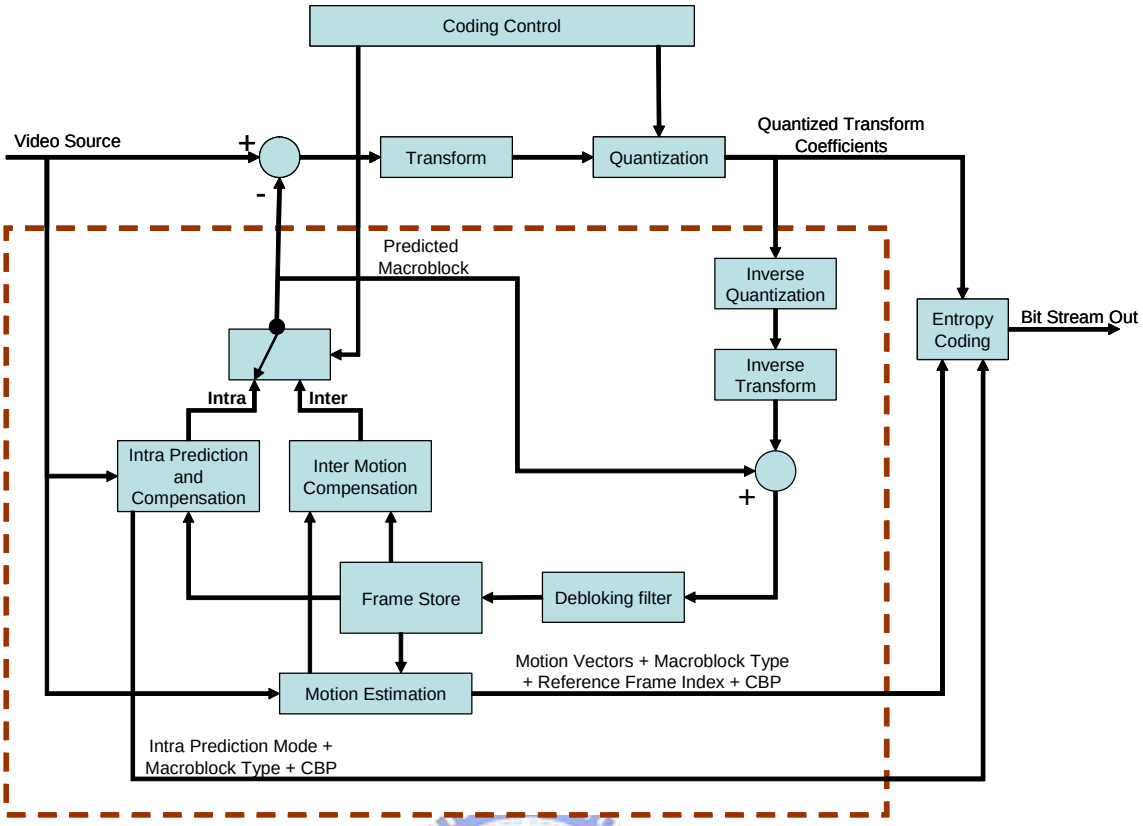


Fig. 18 Block diagram of the H.264 encoder [18]

H.264 這個新一代的影像視訊壓縮標準在 VCL 的設計上有下列的特點和壓縮的模組，包括：

- **ASO**

因爲每個 slice 都是可以獨立解壓縮的，所以 H.264 設計出同一張畫面的 slice 可以不依照一定的順序解壓縮，稱爲 ASO(Arbitrary Slice Order)，以適應可能會造成封包亂序(out of order)的現今網路協定(如 UDP)[19]。ASO 可以增加封包遺失的強韌性(lose robustness)和減少因爲網路傳輸造成的延遲(reduction delay)。

- **FMO**

傳統的壓縮標準(MPEG-4 Simple Profile)每次丟失一個封包(或 slice)，所造成的是畫面上連續區塊的失真，會影響錯誤回復(error resilience)的能力。所以 H.264 支援 FMO(Flexible Macroblock Ordering)，在 slice 中的巨區塊不一定是連續的，是可以均勻分布在畫面中的區塊，如果真的發生封包遺失的話，可以把失真分散開來[20]。

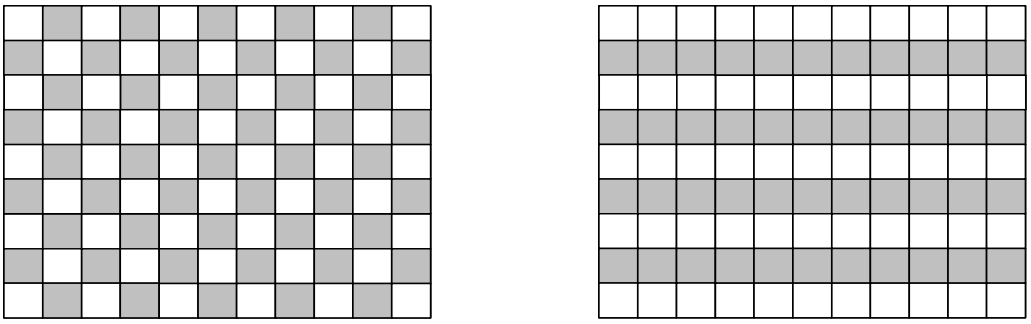


Fig. 19 FMO checker board and interleaving mode

- **Slice Types**

H.264 內的 slice 分爲五種型態，除了與先前標準相似的 I Slice、P Slice 和 B Slice，更增加了 SP 和 SI 的特殊型態。SP 和 SI 的型態在網路傳輸時非常有用，可以在同樣視訊序列但是不同壓縮率的位元流做快速切換，以適應頻寬會隨時變動的真實網路狀況。它還有另一項功用，可以當隨機撥放的撥放點使用。關於 SP 與 SI Slice 的原理與應用環境可以參考[21]。

- **Intra Prediction**

H.264 爲了減少空間上的多餘性(spatial redundancy)，所以採用 Intra Prediction 來減少 intra 巨區塊的資料量。在明度區塊上，巨區塊可以以 16x16 或 4x4 的大小去做預測，16x16 支援四種預測的模式：Vertical、Horizontal、DC 和 Plane；在 4x4 方面支援 9 種模式，如圖 20 所示，除了包括八種方向外還有 DC mode。在色度 8x8 區塊中，支援類似明度 16x16 區塊的四種預測模式(Vertical、Horizontal、DC 和 Plane)。明度區塊和色度區塊中預測模式的選擇是獨立的，不會互相影響限制。

與先前標準不同的是，MPEG-4(Simple Profile)的 Intra Prediction 是在經過轉換處理(Transform)後，在頻率域上(frequency domain)運作(AC/DC Prediction)；H.264 是在轉換處理前做 Intra Prediction，即是在時間域上(time domain)處理。

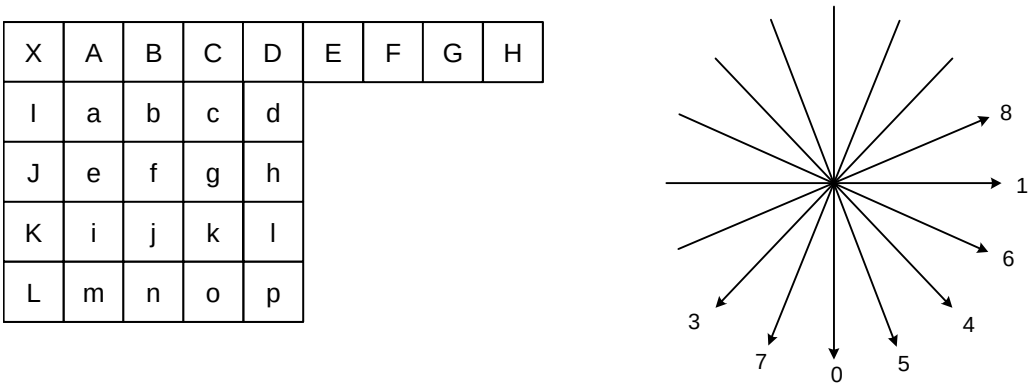


Fig. 20 4x4 Intra Prediction mode: 8 directions + DC

■ **Inter Prediction**

H.264 採用的是以區塊爲基礎的位移補償法(block-based motion compensation)來去除時間上的資料多餘性(temporal redundancy)，與先前的視訊標準不同的是它的區塊最小可以是 4x4(MPEG-4 Simple Profile 最小是 8x8)，區塊越小理論上可以在先前的畫面(reference frame)找到約相似的區塊來增加壓縮率，但是要考慮的是區塊越小所增加的標頭(header)資訊越多，所以必須壓縮時必須仔細考慮區塊大小及標頭資訊來取得比較好的壓縮率[22]。

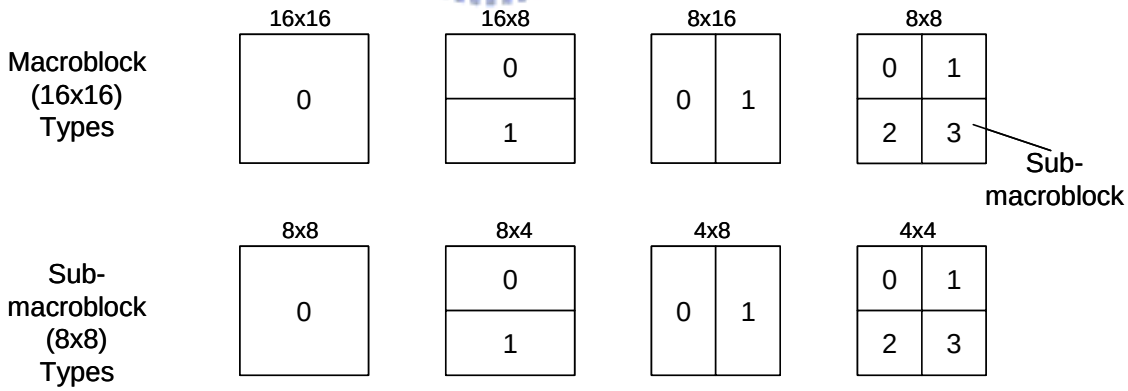


Fig. 21 Variable block-size motion compensation with small block size

另一個不同的是：H.264 可以在前幾張已經解壓縮完的畫面尋找目前要所要壓縮的區塊的最符合的區塊，而不是只能在最近的一張找(MPEG-4 Simple Profile)，因爲參考的畫面增加，可以增加壓縮率，但是要比對的區塊變多也會使複雜度大增。

還有一個不同的是，一般視訊壓縮標準的位移向量(motion vector)的精準度(Accuracy)是到 1/2 像素(half-pixel)，如 MPEG-4 Simple Profile，而 H.264 和 MPEG-4 ASP(Advanced Simple Profile)都是採用精準到 1/4 像素(quarter-pixel)來提高壓縮率，但是 H.264 採用的是 6-Tap 濾波器來產生 1/4 像素，比 MPEG-4 ASP 的 8-Tap 濾波器複雜度低。

- **Weighted Prediction**

H.264 針對一些影片種常見的特殊效果，如淡出或抹去等，可以在做完位移補償完後，在做加權的預測，針對這些效果的到更好的壓縮率。

- **Transformation and Quantization**

以往的視訊壓縮轉換處理大都採用的是 8x8 的浮點數轉換矩陣(MPEG-4 SP & H.263)，H.264 採用的是整數的轉換矩陣(integer transform)，並使用 3 個轉換矩陣。

$$H_1 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \qquad H_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \qquad H_3 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Fig. 22 H.264 Transform Matrices

第一個矩陣(H_1)是針對巨區塊內的每個 4x4 區塊做處理，包括明度區塊和色度區塊，稱爲"Core Transform"。如果預測模式是採用INTRA_16x16 的話，則會將明度區塊中 4x4 經過Core Transform後的DC值集合成一個 4x4 的區塊，在利用矩陣H_2再做一次轉換，稱爲Hadamard Transform. 第三個矩陣(H_3)是利用來針對色度區塊的DC值再做一次轉換[23]。

H.264 支援了 0 到 51 的 QP 值，壓縮器會根據指定的 QP 值算出相對應的量化單位(quantization step size)來作量化處理。當 QP 每增加一各單位，量化單位大約會增加 6%；QP 每增加 6 單位，量化單位會加倍。

QP	0	1	2	3	4	5
QStep	0.625	0.6875	0.8125	0.875	1	1.125
QP	6	7	8	9	10	11
QStep	1.25	1.375	1.625	1.75	2	2.25
QP	12	13	14	15	16	17
QStep	2.5	2.75	3.25	3.5	4	4.5
QP	18	19	20	21	22	23
QStep	5	5.5	6.5	7	8	9
QP	24	25	26	27	28	29
QStep	10	11	13	14	16	18
QP	30	31	32	33	34	35
QStep	20	22	26	28	32	36
QP	36	37	38	39	40	41
QStep	40	44	52	56	64	72
QP	42	43	44	45	46	47
QStep	80	88	104	112	128	144
QP	48	49	50	51		
QStep	160	176	208	224		

Fig. 23 Quantization Parameter and Quantize Step Size

- **Entropy Coding**

H.264 支援的熵編碼模式有兩種：CABAC(context-adaptive binary arithmetic coding)[24]和 CAVLC(context-adaptive variable length coding)。使用 CAVLC 時，雖然它跟之前的標準一樣都是經過查表(VLC Table)來決定最後的壓縮碼(codeword)，但是不同的是它擁有許多 VLC 表，可以根據周圍的資訊找出最適當的表來進行轉換以增加壓縮率。CABAC 採用的是 arithmetic coding，可以避免類似霍夫曼編碼(Huffman coding)中每個符號都需要整數值的壓縮碼中不可避免的位元浪費，理論上 Arithmetic Coding 可以用非整數的壓縮碼來代表一個符號(Symbol)。所以 CABAC 擁有比較好的壓縮率，根據統計大約比 CAVLC 節省 10%~15%的位元數[14]。

- **In-Loop Deblocking Filter**

通常這種混合型的區塊補償視訊壓縮法(block-based hybrid video coding)在經過區塊轉換和量化處理後，很容易看到畫面會有明顯區塊狀的失真(blocking artifact)。Deblocking Filter 主要的功能便偵測這些非真正邊界(boundary)的地方，然後做一些"平滑"的動作，藉以消除區塊狀的失真。之前的視訊壓縮標準，這個工具並不是必要的，通常在撥放前的後處理(post processing)會包括做這個動作(Out-Loop Filter)。但是 H.264

將這個工具模組放進標準的壓縮迴圈中，所以又稱 In-Loop Deblocking Filter。

4. 系統簡介及視訊壓縮器的設計

本論文利用德州儀器(Texas Instruments)所開發的雙核心處理器 OMAP(Open Multimedia Architecture Platform)來驗證我們所提出的[Tightly-Coupled Dual-Core System]。所以本章分成三個小節，第一小節介紹採用的 OMAP 晶片的開發平台，並針對硬體架構進行規格說明。第二小節則是簡介 TI 爲了配合 OMAP 處理器縮發展程式的開發軟體，並闡述雙核心間的程式開發流程。第三小節談論爲了配合 Tightly-Coupled 的軟體架構，對視訊壓縮器所做的設計及修改，藉此可以更了解 Tightly-Coupled 的概念。

4.1. 硬體平台架構

4.1.1. OSK5912

OSK5912[27]是採用 TI OMAP5912[28]所開發出來的嵌入式平台，擁有 32MB 的 SDRAM 和 32MB 的 Flash，以及 10MB 的網路介面和串列埠(serial port)，如圖 25 所示。

Fig. 24 OMAP5912 OSK board

OMAP5912 的晶片採用的是 OMAP3.2 的架構[29]，內部的硬體架構組要分爲三大模組：RISC 核心、DSP 核心和 Traffic Controller。

- RISC 核心是採用工作頻率最快能支援到 192MHz 的 ARM926EJS，擁有 16KB 的指令快取和 8KB 的資料快取，和 MMU(memory management unit)來管理記憶體。
- DSP 核心是最快能跑到 192MHz 的 TMS320C55x 的核心，擁有自己的 24KB 指令快取和 64KB 的 DARAM 和 96KB 的 SARAM，而且用有 6 通道的 DSP DMA 和自己的中斷控制器。
- Traffic Controller 則是管理晶片上的記憶體系統和資料匯流排，可以控制 ARM 核心或 DSP 核心對於記憶體的存取權限。

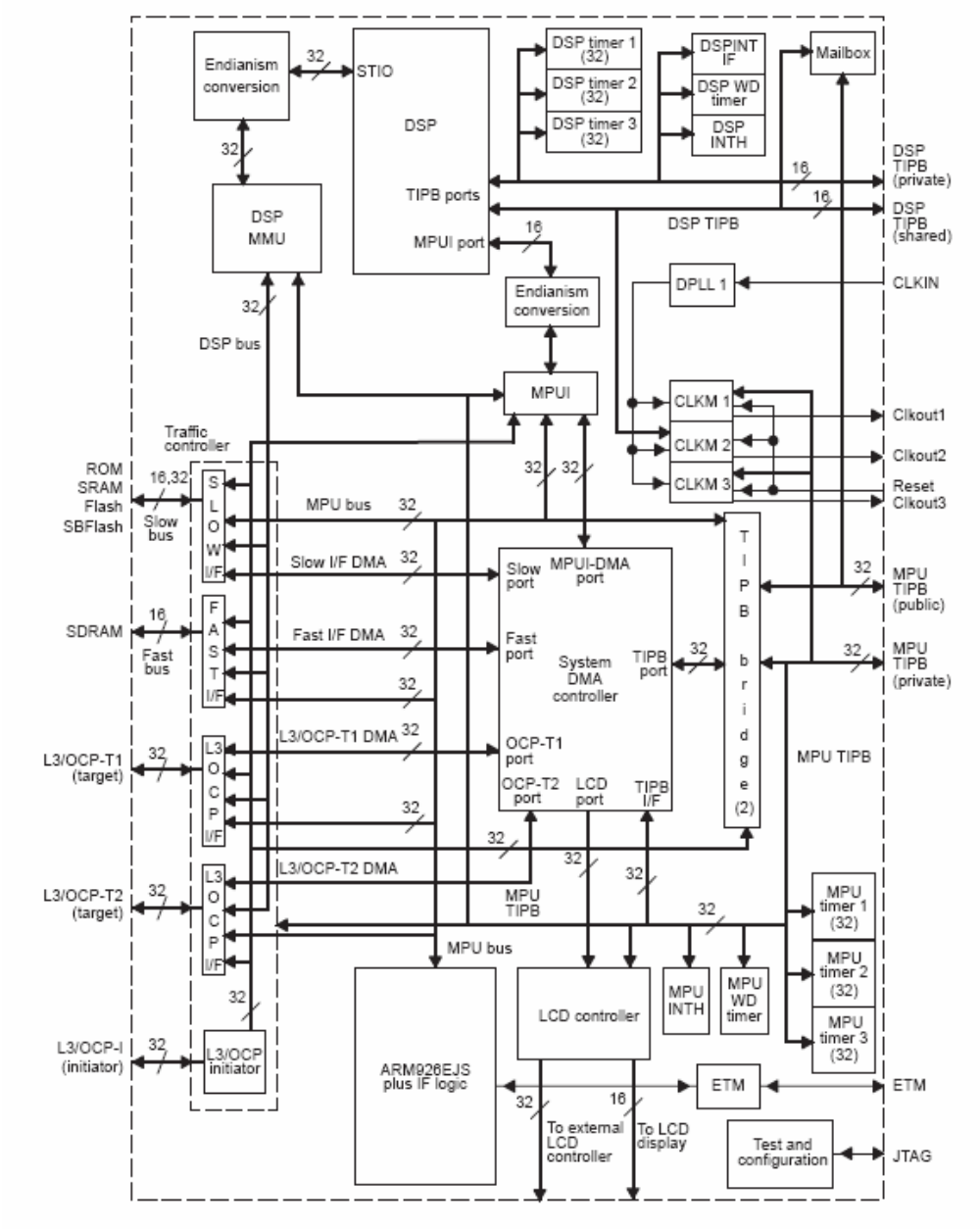


Fig. 25 OMAP5912 Processor

4.1.2. OMAP 的溝通機制

在 OMAP 處理器中，ARM 核心和 DSP 核心溝通的機制有下列兩種：

- Mailbox

在 OMAP 的晶片上，提供了 mailbox 的硬體溝通機制，一組 mailbox 包括命令暫存器，資料暫存器和相對應的旗標(flag)。這種傳遞訊息的方式是單向的，所以 OMAPA 提供兩類的 mailbox，一類是 ARM 到 DSP 的，另一類的就是 DSP 到 ARM

的。當 ARM 到 DSP 的 mailbox 被 ARM 設定後，該旗標會被設爲 1，而且會在 DSP 的核心上觸發中斷，DSP 處理完中斷後必須把旗標清除爲 0，代表訊息已經被處理了。本論文採用的便是這種的溝通方式，如圖：

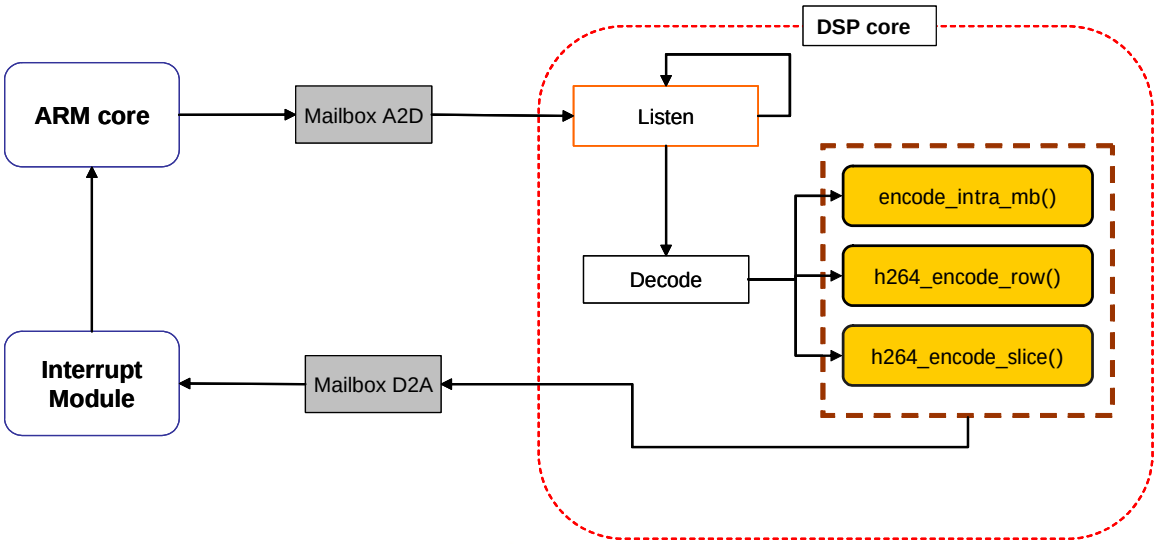


Fig. 26 Inter-processor communication by mailbox

- Share memory(DSP MMU 或 MPUI)

  兩個核心間的溝通也可以透過 share memory 的方式，然後利用 polling 的方式來做確認。至於 share memory 的方式有兩種，一個是設定 MPUI，ARM 透過 MPUI 可以存取 DSP 的周邊和記憶體(DARAM 和 SARAM)。另外一種則是啓動 DSP MMU，透過 DSP MMU 的映射(mapping)，DSP 可以看到 ARM 的某些記憶體，在透過這些記憶體來完成 ARM 與 DSP 之間的資料分享。

4.2. 軟體平台架構

4.2.1. 作業系統

因爲現有的作業系統都沒支援類似 Tightly-coupled 的軟體架構，所以在本論文的實驗中，都是在沒有作業系統的環境下完成數據的量測。

4.2.2. CCS (Code Composer Studio)

CCS 是一套 TI 所發展的整合發展環境，它提供了包括 ARM 和 DSP 的 C/C++語

言編譯器(compiler)、組合語言的組譯器(assembler)、連結器(linker)和其他工具。在使用 CCS 必須先針對 ARM 或 DSP 做好記憶體的規劃，經過 CCS 的程式碼最佳化，可以分別產生 ARM 和 DSP 的二元檔(binary code)，在透過 JTAG 將程式碼下載到事先規劃好的記憶體，然後可以透過 CCS IDE 來控制 CPU 的狀態(Run 或 Halt)，也可以對程式編碼做即時的除錯和分析。

4.3. Proposed Tightly-Coupled Video Encoder

在本論文中我們決定用新一代的視訊壓縮標準 H.264 來驗證 Tightly-Coupled 架構，但是現有的參考軟體(JM9.6 [30])的軟體架構並不符合我們需求，所以我們花了一段時間來重新設計我們的壓縮器的資料結構，並盡量在記憶體的使用量和壓縮器的效能兩者間取得平衡。

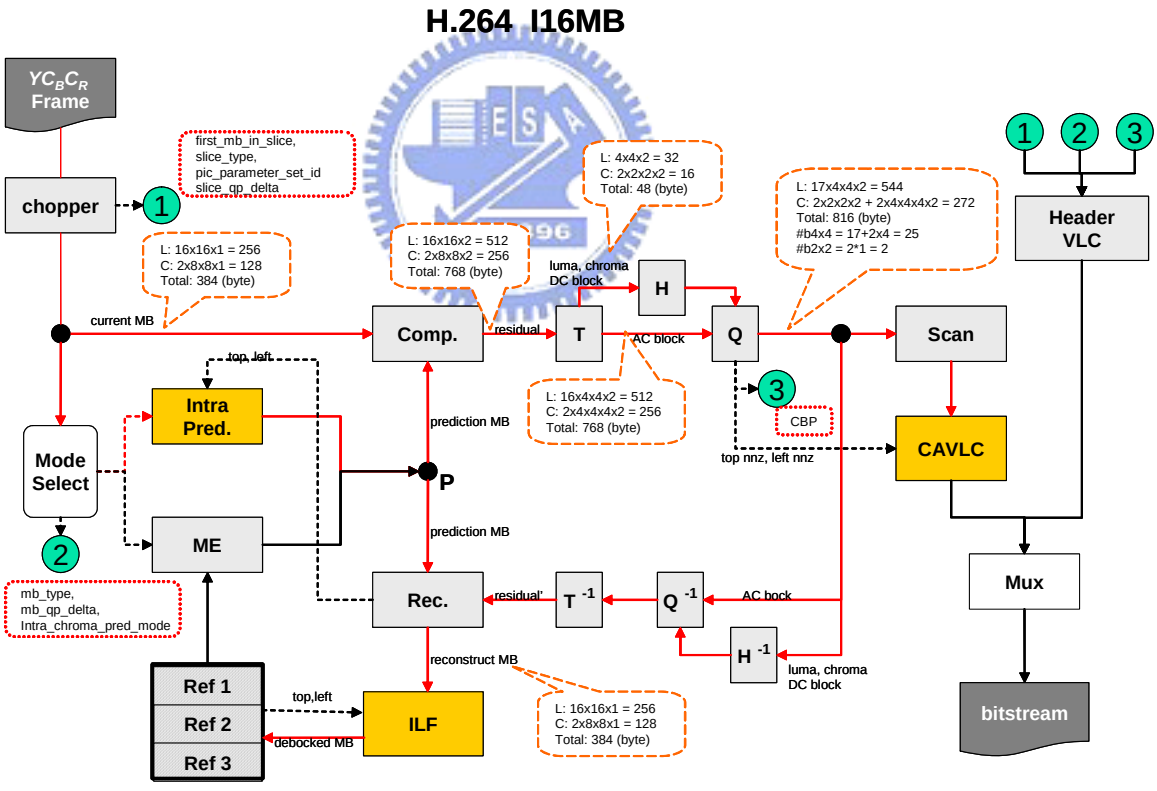


Fig. 27 H.264 I16MB block diagram

考量在 Intra frame 的壓縮演算法時，每個畫面(frame, slice)會切割成若干個巨區塊(macroblock)來壓縮，但是因爲 Intra prediction、CAVLC 和 In-Loop Filter 這些壓縮模

組的關係，所以巨區塊與巨區塊間會有一定的相依關係。當上方的區塊或左邊的區塊未完成之前，現有的區塊是無法做壓縮的，這對我們的架構設計會有相當大的影響。我們的設計是以巨區塊做爲工作分配的基本單位，參考過去實驗室的經驗[31]，有下列幾點考量：

- ARM 與 DSP 的執行速度

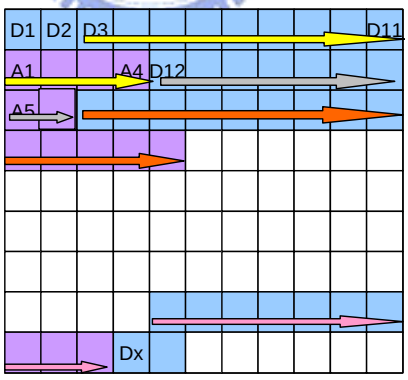
  依照 ARM 與 DSP 資料處理速度的特性，我們知道 DSP 執行的速度會比 ARM 快，所以我們不會將 ARM 與 DSP 做一比一的分配，而是根據核心現有的執行能力自動分配工作區塊的個數。

- 兩個核心之間溝通時間的花費

  因爲兩個核心是合作完成同一件工作，所以兩個核心了解彼此的資訊，比起用單一核心來完成工作，溝通的花費是額外的。我們並不想要核心每做完一個巨區塊就溝通一次，因爲這樣的花費(overhead)會成爲整個壓縮器系統瓶頸。

- 巨區塊間的相依性

  考量巨區塊間的相依性和 ARM 及 DSP 的處理速度，如果依照原本影像壓縮的區塊掃描方式(raster scan)做分配，DSP 會因爲在等待 ARM 巨區塊的完成而處於沒有工作可做的狀態，這樣會浪費許多 DSP 的效能，也會拖累整個系統的效能。



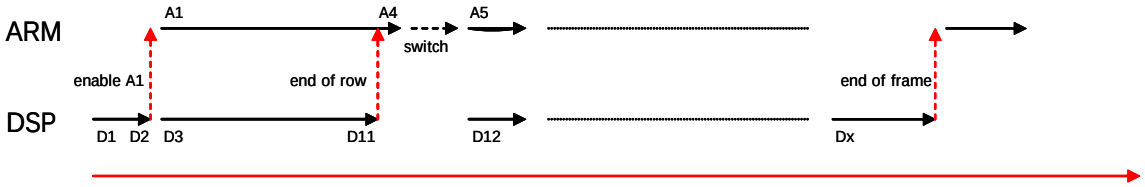


Fig. 28 Proposed tightly-coupled video encoding process

我們論文實驗的區塊分配方法如上圖所示，因爲 DSP 的執行速度比較快，所以一開始時我們先將上面一列的巨區塊(D1~D11)都分配給 DSP 處理，當 DSP 處理完後第二個區塊(D2)後，第二列的第一個區塊(A1)所需要的資訊也都具備了，所以 ARM 可以開始處理 A1，這個時候 DSP 也同時在處理 D3，也就達到平行處理的效果。當 DSP 處理完第一列後，DSP 會利用 mailbox 通知 ARM，這個時候 ARM 可能在處理 A4 這個區塊，當 ARM 處理完 A4 時會做一個交換(switch)的動作，即 ARM 會把這一列剩下的工作交給 DSP，ARM 在繼續處理下一列的第一個巨區塊(A5)，利用這樣的方法來達到平行處理及動態分配的效果。

5. 實驗數據與分析

在這一個章節中，我們會簡單介紹我們實驗環境的設定並列出資料，並根據這些現有的資料來做進一步的分析。

5.1. 實驗環境與數據

在H.264的INTRA_16x16的壓縮過程中，我們切割成幾個模組並分別編譯成ARM與DSP的程式碼，如下圖所示。在prediction & compensation模組中，我們現在只有將DC prediction的程式碼加上去，其他包括Horizontal、Vertical、Plane的程式碼我們都還沒整合到這個架構中，所以我們現在壓縮的模式都是用DC prediction。

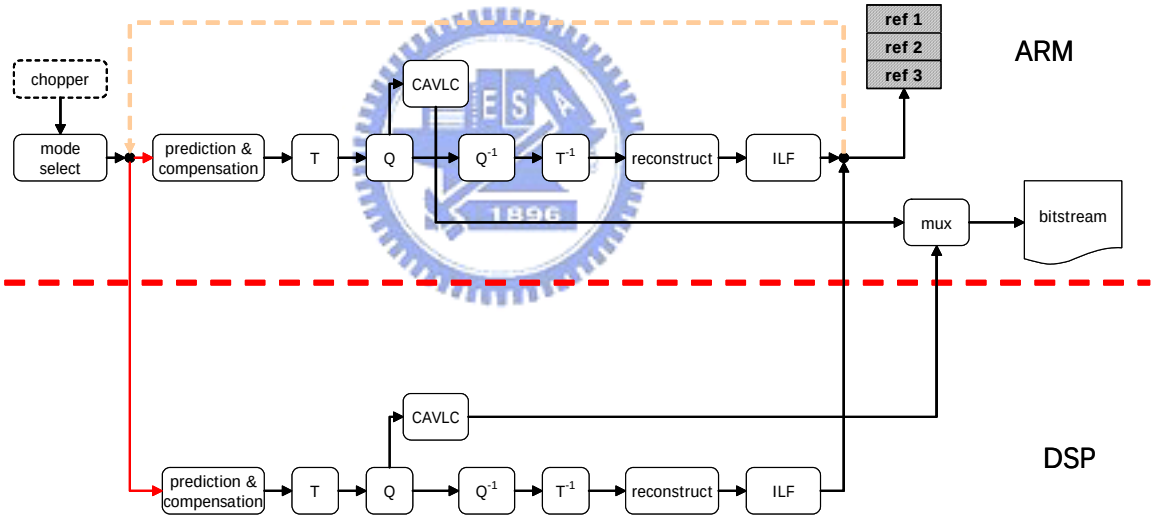


Fig. 29 INTRA_16x16 encoding process

爲了兩個核心的溝通方便，所以我們利用 DSP MMU 的設定來映射(mapping)SRAM到DSP的記憶體空間，做爲兩個核心的共享記憶體(share memory)，這個共享記憶體主要是存放現有要壓縮的畫面和解壓完後的畫面。ARM 的程式碼主要是放在SDRAM上，DSP的程式碼是放在SARAM上。因爲我們在ARM的快取記憶體的設定上一直出現問題，所以目前無法使用 ARM 上的快取記憶體，爲了公平起見，我們也將DSP上所需要用到的記憶體空間設定在SRAM(share memory)上，而不是比較快的DARAM上。

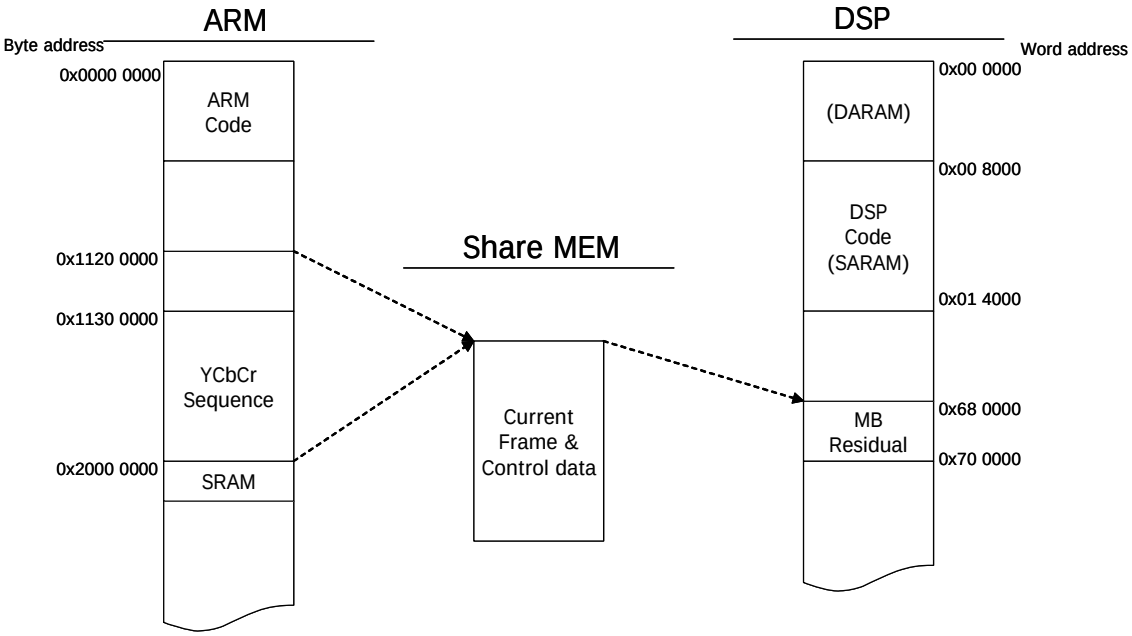


Fig. 30 memory map

我們實驗用的影像序列是 stefan 的前 30 張，解析度是 QCIF(176x144)，QP 固定爲 26。至於發展板上的設定，雖然 ARM 跟 DSP 都能達到 192MHz 的工作頻率，但是我們是將他設定在 96MHz，以模擬現有的嵌入式系統的環境，Traffic Controller 我們將它設定在它的最高頻率 96MHz。所量得數據如下：

- Loosely-coupled

  我們分別只利用單純的 ARM 或 DSP 來做整個的壓縮流程，我們利用每個巨區塊所必須耗費的週期數(cycles per macroblock)來做統計。由下表我們可以很清楚的發現 ARM 壓縮一個巨區塊耗費的時間是 DSP 處理的 7.62 倍，換句話說 ARM 處理一個巨區塊的時間可以讓 DSP 處理 7.62 個巨區塊。

	ARM		DSP	
(Y) DC pred. & comp.	44224	1.87%	8465	2.72%
(Y) Transform	285453	7.84%	23178	7.46%
(Y) Quant	248798	10.51%	39521	12.72%
(Y) Inv. Quant	223279	9.43%	30002	9.65%
(Y) Inv. Transform	196585	8.31%	27863	8.97%
(Y) Reconstruct	105923	4.48%	17835	5.74%
(UV) DC pred. & comp.	50307	2.13%	10142	3.26%
(UV) Transform	87297	3.69%	11379	3.66%
(UV) Quant	131223	5.54%	20814	6.70%
(UV) Inv. Quant	124746	5.27%	15410	4.96%
(UV) Inv. Transform	98465	4.16%	13907	4.47%
(UV) Reconstruct	53941	2.28%	8861	2.85%
cavlc	431071	18.21%	46204	14.87%
ILF	385643	16.29%	37209	11.97%
Total	**2366961**	**7.62**	**310796**	**1**

Table 1. Loose-coupled ARM and DSP performance

- Tightly-coupled

利用 Tightly-coupled 的分工模式所得到的數據如表一所示，實驗的結果反而比單純用一個 DSP 來跑的數據慢了大概 27.25%。ARM 所處理的巨區塊是 16 個，DSP 是 83 個，分布的情形如圖 31 所表示，ARM 所完成的都是給一列的最左邊兩個巨區塊(第一列除外)。

	ARM	DSP	Tightly-couple
Encode count per frame	244293630	30574135	42025593
ratio	5.81	0.7275	1
			#ARM = 16 #DSP = 83

Table 2. Tightly-coupled dual-core performance

D	D	D	D	D	D	D	D	D	D	D
A	A	D	D	D	D	D	D	D	D	D
A	A	D	D	D	D	D	D	D	D	D
A	A	D	D	D	D	D	D	D	D	D
A	A	D	D	D	D	D	D	D	D	D
A	A	D	D	D	D	D	D	D	D	D
A	A	D	D	D	D	D	D	D	D	D
A	A	D	D	D	D	D	D	D	D	D
A	A	D	D	D	D	D	D	D	D	D

Fig. 31 tightly-coupled macroblock distributuion

5.2. 結果分析

從上一小節的數據中發現，Tightly-Coupled 的執行速度竟然還比用單一 DSP 還慢，所以我們分析到底是爲什麼？從圖 31 中我們大概可以猜的到 DSP 處理完一列後，浪費許多時間在等 ARM 把第二個巨區塊執行完，而 ARM 幫忙做的區塊又比不上 DSP 在等待的這段時間中所能處理區塊數。

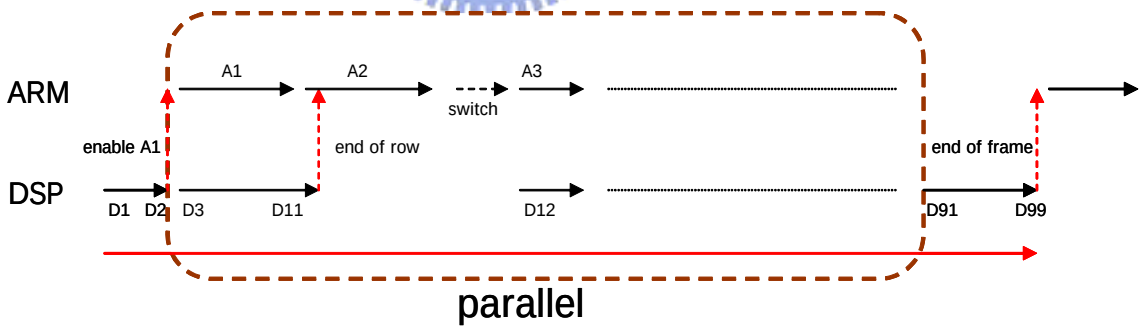


Fig. 32 Tightly-Coupled execute time

Tightly-couple 每張畫面壓縮所花的時間：42025593 (clock cycles)，真正壓縮的時間爲：T(DSP)*2+T(ARM)*18+T(DSP)*9 = 41290132 (clock cycle)，所以在兩個核心在溝通上與分配區塊所花費的週期數爲 42025593 – 41290132 = 735461，所以會使系統的效能降低 1.75%。

在平行化處理 88 個巨區塊的這段時間，總共花的時間爲：T(ARM)*18 = 37871376

(clock cycles)，如果這 88 個巨區塊都分配給 DSP 執行所花的時間爲：T(DSP)*88 = 275350048(clock cycles)，所以浪費了 3781376 – 27350048 = 10521328 (clock cycles)，約佔了全部的 25.035%。

6. 研究成果討論及展望

6.1. 研究結果討論

在第五章的實驗中，我們分別列出 ARM、DSP 與 Tightly-coupled 視訊壓縮器的效能，得知 Tightly-Coupled 的架構並沒有帶來加速的效果，難道 Tightly-Coupled Video Encoder 並不適當嗎？其實也不盡然，我們當初的出發點在於 DSP 上可能會執行許多個任務，而不是單純的只負責視訊壓縮的工作。

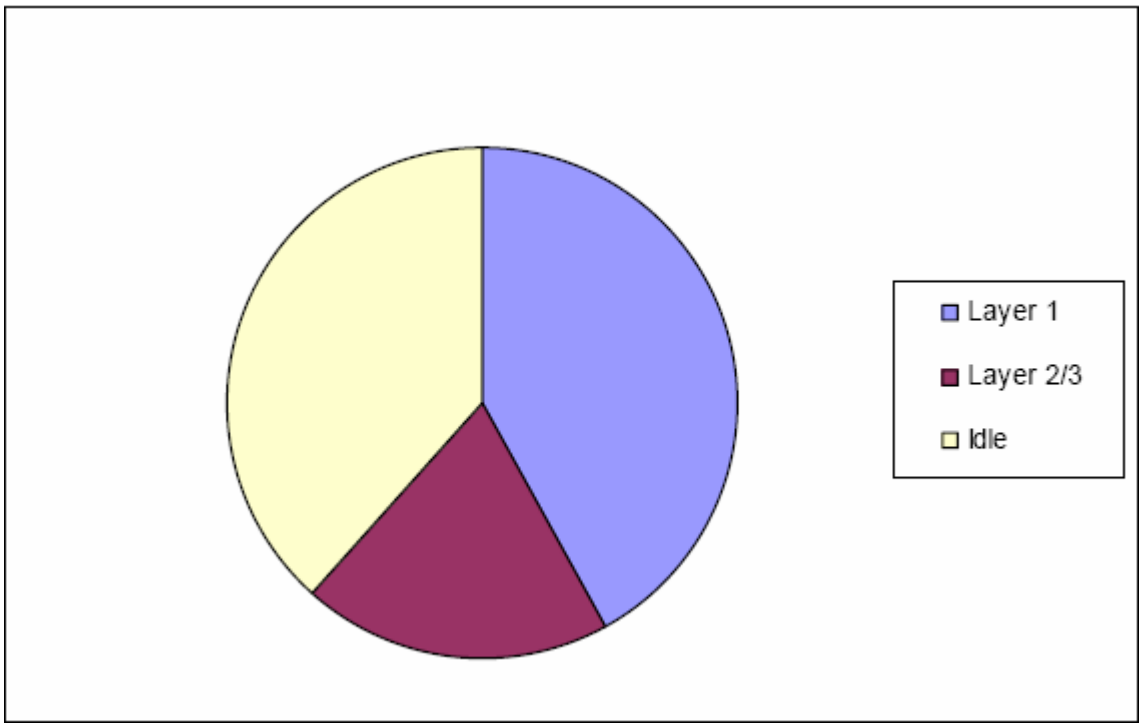


Fig. 33 Estimated StarCore DSP Task loading for EDGE Class 12[32]

例如在 Freescale 的 MXC 架構白皮書中提到，它的 DSP 需要負責 EDGE Class 12 的 Layer 1、Layer 2 和 Layer 3 的工作，需要用掉 60%的運算資源[32]。所以在這種前提下，DSP 處理巨區塊的能力就會剩下原來的 40%，與 ARM 之間的處理能力就會拉近，這種狀況下 Tightly-Coupled 的視訊壓縮器的優點就會顯現出來。

6.2. 未來工作及展望

展望未來的工作，有兩點是我們所需要繼續努力的，包括：

- 完整的 H.264 視訊壓縮器的實現

  目前我們擁有重新設計資料結構的 H.264 Intra coding 的程式，必須還要把 Inter Coding 的程式早點實現，這樣才能考量完整的記憶體使用規劃。另外我們也可以利用系統 DMA(System DMA)和 DSP DMA 來加速我們的效能，因爲記憶體的搬移是很耗費資源的，所以我們是事先做了一些 DMA 的測試，以便日後可以幫忙做一些系統的整體考量，測試的數據如下面兩個表格所表示。最右邊的欄位代表的 TI OMAP5910 文件所列出的數據可作爲參考用[32][33]，DSP DMA 的效能明顯比 System DMA 好很多。

	ARM&DSP [192MHz] TC [96 MHz]	ARM&DSP [96MHz] TC [96MHz]	Spec. (TC cycles per 16-bit word)
SDRAM → SRAM	6.813 MB	6.813 MB	31.47 MB
SDRAM → SARAM	6.767 MB	6.731 MB	14.77 MB
SRAM → SARAM	14.093 MB	9.270 MB	14.77 MB

Table 3. System DMA throughput

	ARM&DSP [192MHz] TC [96 MHz]	ARM&DSP [96MHz] TC [96MHz]	Spec. (TC cycles per 16-bit word)
SDRAM → SARAM	15.842 MB	15.858 MB	16.69 MB
SDRAM → DARAM	15.842 MB	15.855 MB	16.69 MB
SRAM → SARAM	23.879 MB	21.254 MB	38.4 MB
SRAM → DARAM	23.879 MB	21.256 MB	38.4 MB

Table 4. DSP DMA throughput

- 模擬 Tightly-Coupled 視訊壓縮器處於 DSP 多工環境下

  我們接下來所必須的實驗是模擬真實世界會發生的情況，並量得一些新數據證明我們的 Tightly-Coupled 架構是有它存在的必要性。當然這也必須要經過程式的最佳化，包括 ARM 程式中快取的應用和 DSP 中 DARAM 和 SARAM 的充分利用，使我們的影像視訊壓縮器能達到即時性的需求。

7. 參考文獻

[1] Jamil Chaoui, etc. Open multimedia application platform: enabling multimedia applications in third generation wireless terminals through a combined RISC/DSP architecture, Proceeding of ICASSP2001, Pages:1009~1012 vol.2, May 2001

[2] Intel PXA800F Cellular Processor for GSM/GPRS Mobile Solutions, [Online] http://www.intel.com/design/pca/prodbref/252336.htm

[3] Intel Micro Signal Architecture, [Online] http://www.intel.com/design/msa/

[4] Sheila Rader, etc., "Mobile Extreme Convergence: A Streamlined Architecture to Deliver Mass-market Converged Mobile Devices," freescale (Motorola) white paper

[5] Santanu Dutta, etc., "Viper: A Multiprocessor SOC for Advanced Set-Top Box and Digital TV System," Design & Test of Computers, IEEE Volume 18, Issue 5, Sept.-Oct. 2001 Page(s):21 – 31

[6] K.R.Rao, J.J.Hwang, "Techniques and Standards for Images, Video, and Audio Coding, Chapter 2 Color Formats" Prentice Hall, July 1996.

[7] Iain E. G. Richardson, "Video Codec Design: Developing Image and Video Compression System Chapter 7 Transform Coding" John Wiley & Sons Ltd. 2002.

[8] Iain E. G. Richardson, "Video Codec Design: Developing Image and Video Compression System Chapter 7.6 Quantisation" John Wiley & Sons Ltd. 2002.

[9] ITU-T, "Video codec for audio/visual services at px64 kbits/s," ITU-T Rec. H.261 v1: Nov 1990, v2: Mar. 1993

[10] ISO/IEC JTC 1, "Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s-Part 2: Video," ISO/IEC 11172(MPEG-1), Nov. 1993.

[11] ITU-T and ISO/IEC JTC 1, "Generic coding of moving pictures and associated audio information – Part 2: Video," ITU-T Rec. H.262 and ISO/IEC 13818-2 (MPEG-2), Nov. 1994

[12] ITU-T, "Video coding for low bit rate communication," ITU-T Rec. H.263; v1: Nov. 1995, v2: Jan. 1998, v3: Nov. 2000.

[13] ISO/IEC JTC 1, "Coding of audio-visual objects – Part 2: Visual," ISO/IEC 14496-2 (MPEG-4 Part 2), Jan. 1999.

[14] Gary J. Sullivan, Thomas Weigand, "Video Compression – From Concepts to the H.264/AVC Standard" Invited Paper, Proceeding of the IEEE, vol. 93, issue 1, pp. 18~31, Dec. 2004.

[15] Joint Video Term of ITU-T and ISO/IEC JTC 1, "Drift ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 144496-10 AVC)," May 2003.

[16] Gary J. Sullivan, Pankaj Topiwala, and Ajay Luthra, “The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions,” SPIE Conference on Application of Digital Image Processing, Aug. 2004.

[17] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer and T. Wedi, “Video coding with H.264/AVC: Tools, Performance, and Complexity,” IEEE Circuits and Systems Magazine, vol. 4(1), pp. 7-18, 2004.

[18] UB Video, “Emerging H.264 Standard: Overview and TM320DM642 – Based Solution for Real-Time Video Application,” white paper, 2002.

[19] T. Weigand, G. J. Sullivan, G. Bjontegarrd, and A. Luthra, “Overview of the H.264/AVC Video Coding Standard,” IEEE Transactions on Circuits and Systems for Video Technology, vol.13, no. 7, pp. 560-576, July 2003.

[20] S. Kumar, L. Xu, M. K Mandal, S. Panchanathan “Overview of Error Resiliency Schemes in H.264/AVC Standard,” To Appear in Elsevier Journal of Visual Communication and Image Representation, 2005

[21] M. Karczewicz and R. Kurceren, “The SP- and SI-frames design for H.264/AVC,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, pp. 637-644, July 2003.

[22] G. J. Sullivan and R. L. Baker, “Rate-distortion optimized motion compensation for video compression using fixed or variable size blocks,” IEEE Global Telecommunication Conference (GLOBECOM), pp.85-90, Dec. 1991.

[23] H. Malvar, A. Hallapuro, M. Karcrewicz, and L. Kerofsky, “Low-complexity transform and quantization in H.264/AVC,” IEEE Trans. on Circuits and Syst. for Video Technology, vol. 13, no. 7, pp. 598-603, July 2003.

[24] D. Marpe, H. Schwarz, and T. Weigand, “Context-adaptive binary arithmetic coding for H.264/AVC,“ IEEE Trans. on Circuits and Syst. for Video Technology, vol. 13, no.7, pp. 620-636, July 2003.

[25] P. List, A. Joch, J. Lainema, G. Bjontegaard, M. Karczewicz, “Adaptive deblocking filter,” IEEE Trans. on Circuits and Syst. for Video Technology, vol. 13, no. 7, pp. 614-619, July 2003.

[26] OMAP5912 Dual-Core Processor Data, Texas Instruments, Dallas Texas, [Online] http://www.ti.com/

[27] OSK5912 Support Home. [Online] http://omap.spectrumdigital.com/osk5912/

[28] OMAP5912 Application Processor, [Online] http://www.ti.com/

[29] OMAP5912 Multimedia Processor OMAP3.2 Subsystem Reference Guide, [Online] http://www.ti.com

[30] H.264/AVC reference software. [Online] http://iphome.hhi.de/suehring/tml/

[31] 曾建堂, “Video Codec Optimization for Dual-core Architecture,” 國立交通大學資訊工程系碩士論文, 2004.

[32] S. Rader, J. Mena, M. Pandya, A. Bansal, F. Shearer and N. Marshall, “Mobile Extreme Convergence: A Streamlined Architecture to Deliver Mass-Market Converged Mobile Devices,” Freescale White Paper.

[33] OMAP System DMA Throughput Analysis (spra883.pdf), [Online] http://www.ti.com

[34] OMAP5910 DSP DMA Throughput Analysis (spra889.pdf). [Online] http://www.ti.com