國立交通大學

資訊科學與工程研究所

碩 士 論 文

在多核心嵌入式平台使用平行處理加速 H.264 視訊解碼

Parallelization of H.264 video decoder for Embedded Multicore Processor

研 究 生：陳彥廷　　Student：Yan-Ting Chen

指導教授：蔡淳仁　　Advisor：Chun-Jen Tsai

中 華 民 國 103 年 7 月

在多核心嵌入式平台使用平行處理加速 H.264 視訊解碼

Parallelization of H.264 video decoder for Embedded Multicore processor

研 究 生：陳彥廷　　Student：Yan-Ting Chen

指導教授：蔡淳仁　　Advisor：Chun-Jen Tasi

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

July 2014

Hsinchu, Taiwan, Republic of China

中華民國 103 年 7 月

中文摘要

本論文主旨在於在多核心平台上對 Baseline H.264/AVC decoder 用不同的平行化方式達到加速的效果，並分析 data parallelism 與 task parallelism 的平行技術，如 static scheduling 與 dynamic scheduling、static partition 與 dynamic partition、task buffer monitoring 等等，並探討平行化過程中所產生的負擔與問題，如資料的搬移、記憶體使用率、synchronization、load balancing 議題等等，以上的平行視訊解碼設計及分析是在一個新的應用處理器架構上進行[14]。這個新的多核心處理器架構可以在不增加額外 system bus 負擔下，採用工作切割精細的 software-pipeline 平行化方式，來增加 pipeline-based video decoder 的效能，根據我們實驗的結果，採用 dynamic pipeline partition 在三顆核心下能相對於單核心 H.264/AVC 解碼器有接近三倍的加速。

Abstract

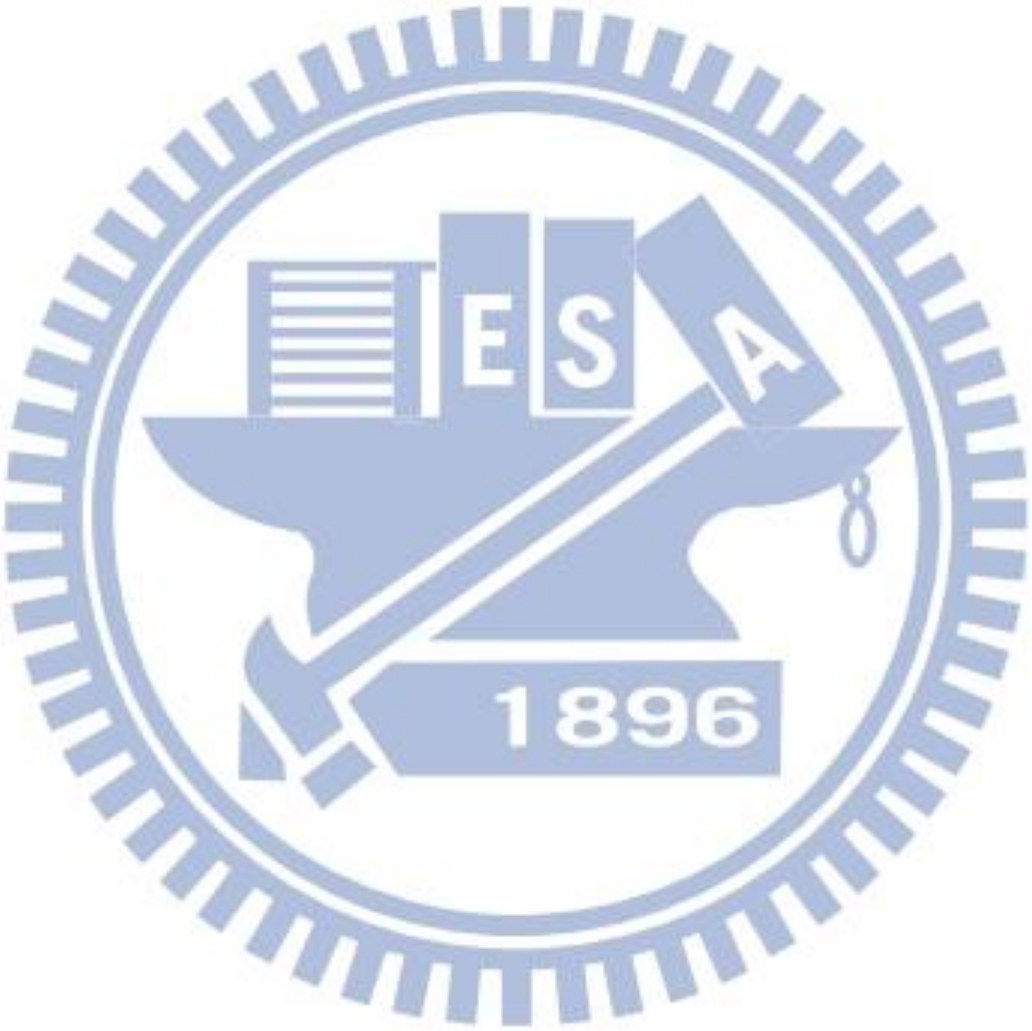
In this thesis, we present two parallel decoding approaches to enhance the performance of H.264/AVC decoders on multiprocessor platform. We analyze data-level parallelism and task-level parallelism for various parallel decoding techniques, such as static and dynamic scheduling, static and dynamic partitioning or FIFO task buffers monitoring etc. We discuss some overheads and problems resulted from the parallelization process, such as data transfer, memory usage, synchronization, or load balancing issue etc. The investigation of parallel video decoding is conducted on a new multicore application processor architecture that facilitates the adoption of the fine-granularity software-pipeline parallelism without causing extra burden on the system bus and it will increase pipeline-based video decoder performance. Experimental results show that the adoption of the dynamic pipeline partition approach could nearly be three times faster than a single-core H.264/AVC decoder does.

誌謝

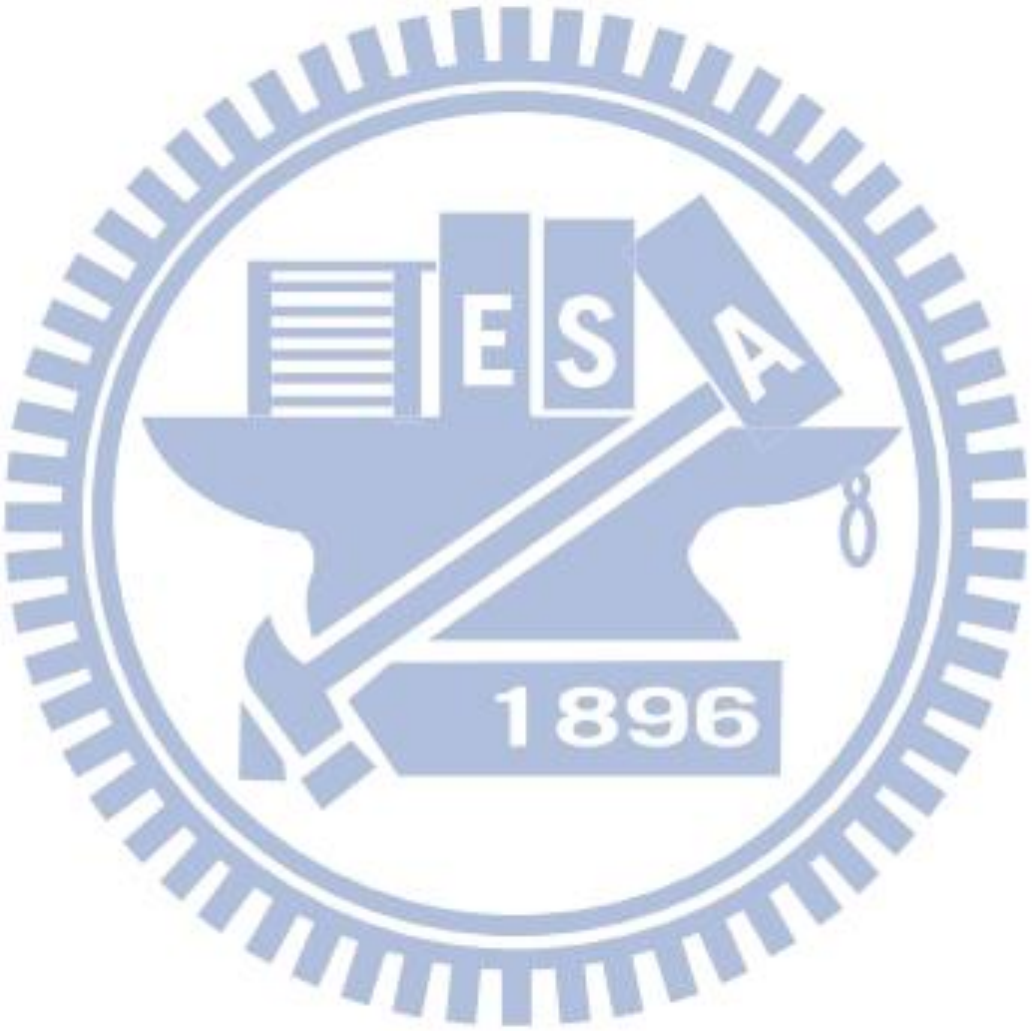
這篇論文能夠順利完成，最需要感謝的就是蔡淳仁老師，每次研究遇到瓶頸時，老師總是耐心的提點，而每次在與老師討論的過程中，總是能獲益良多，而且也讓我見識到何謂是嚴謹的推演、以及對數據的敏感，也謝謝老師總是能分享在業界的經驗以及小技巧。在這裡也感謝實驗室的各個同學，謝謝他們總是能分享各種經驗，讓許多事情能順利的完成，也消磨了不少研究所單調的日子。接下來要感謝父母支持我完成碩士學位，而因為有大家的支持，才能成就今日的我，我也才能順利的完成論文、並且畢業。

章節目錄

List of Figures



List of Tables



一、緒論

1.1 簡介

隨著 3C 技術的整合與網際網路的蓬勃發展，嵌入式系統被視為是未來科技應用的主軸，而與我們生活息息相關的是以嵌入式系統為主的資訊家電，如智慧型手機、數位隨身聽、數位相機、平板電腦、遊戲機等等，其中以智慧型手持式裝置滲透率最高，未來成長最具有潛力，而手持式裝置的功能也已經從單純收發簡訊及電話通訊轉為現今的視訊通話、高清影片播放和進行高特效遊戲等等的強大功能，智慧型手持裝置可以被視為一台縮小版的桌上型電腦。近年來網路興盛，3G、4G 技術興起，網路速度飛躍性的成長，這意味著手持式裝置能在極短的時間內接收大量的資料，因此嵌入式系統將需要提供更強大的運算能力來因應使用者需求。

而在嵌入式系統各種應用中，多媒體應用往往是嵌入式系統設計中的一大重點，因為多媒體應用是最直接與使用者進行互動，且多媒體程式其優劣能容易被使用者在視覺或聽覺上感受到，要能提供使用者感受到平順的影音同步播放，就必須在短時間內處理大量多媒體資料，而又因為網路速度的提升，支援 FULL HD 以上規格的多媒體應用已經是往後必須面對的課題[1]，不僅影音呈現上要達到平順且能源的消耗也要被嚴格的限制，因此單核心架構已經不足以負擔，故提出了多核心架構(multiprocessor architecture)。

多核心架構比傳統的單核心架構有著許多的優勢，單核心處理器在高頻率執行作業時其功率大幅驟升，而多核心處理器可用較低的頻率運作，這通常會耗用較少的功率，也可以透過平行運算來處理，因此可以比單核心處理器更快完成運算工作。而且在能源的使用上，多核心架構可以依照工作量而動態的開啟或是關閉某些核心，達到節能的效果。在多媒體處理方面，但在較高的解析度、位元率(bit rate)和帧率(frame rate)的情況下，利用多核心平行解碼是不可避免的。

[1] 手持式裝置通常可支援高於其螢幕解析度的輸出

過去我們提出一個新的 SMP 多核心架構[14]，在傳統的 SMP 應用處理器中增加一些由硬體實作的 IPC(Inter-processor communication) controller，能使 processor cores 透過 IPC channels 及 local scratchpad memory 來共享資料，用來減輕 system bus 頻寬且減少傳送或接收資料的負擔，是專門為 SMP processors 設計的 hardwired pipeline datapath。另外，我們也為此 IPC controller 設計易於使用的 APIs，以減輕程式設計師實作 pipeline program 的負擔。

而本篇論文在這個系統架構下，以 DLP(data-level parallelism)和 TLP(task-level parallelism)[2]的平行化方式實作 H.264/AVC baseline decoder，在第四章節會對相關論文做討論，並對兩種平行化方式各別進行分析探討，如 TLP 動態切割與靜態切割的 load balancing 議題[3]、或是 DLP 同步化機制的探討等等，並分析本研究提出的 IPC 對於 TLP 效能上的幫助。

1.2 論文架構

本篇論文主要分為六個章節，第一章緒論與論文架構，第二章簡介 H.264/AVC 各個模組的基本架構與演算法，第三章則是介紹 H.264 平行化的技術與 data-level parallel video decoder 實作，第四章則是探討在特殊架構下的 task-level parallel video decoder，第五章則是介紹硬體平台與系統架構以及實驗結果和分析，最後，第六章則是結論與未來的展望。另外論文後面附加附錄，附錄為介紹在 PAC DUA 平台上實作 Software pipelined decoder 與分析。

[2] Task-level parallelism 又可稱 function-level parallelism 或是 pipeline

[3] 不只 TLP 有動態與靜態切割的議題，本研究也會探討 DLP 的動態與靜態切割

二、H.264/AVC

本章節是對 H.264/AVC 主要的模組進行介紹，並對每個模組可能平行化的部分進行討論。

2.1 H.264/AVC 壓縮標準概論

ISO MPEG 與 ITU-T VCEG 從 1990 年代開始制定一連串的影音訊號編碼標準，ITU 制定 H.261/262/263/264 標準，ISO 則制定 MPEG-1/2/4 標準，而兩者之間互有合作，Figure 1.所示。在 2001 年 12 月，ITU-T VCEG 與 ISO MPEG 共同組成聯合視訊小組(Joint Video Team，JVT)來研訂新的視訊壓縮標準，此新的標準在 ITU-T 組織中稱為 H.264，在 ISO 組織中則被納入 MPEG-4 Part-10(ISO/IEC 14496-10)並命名為 Advanced Video Coding(AVC)，故通常合併稱為 H.264/AVC[4]。且相關研究顯示 H.264/AVC 能用一半左右的位元率達到與 MPEG-1/2 相同的視訊品質

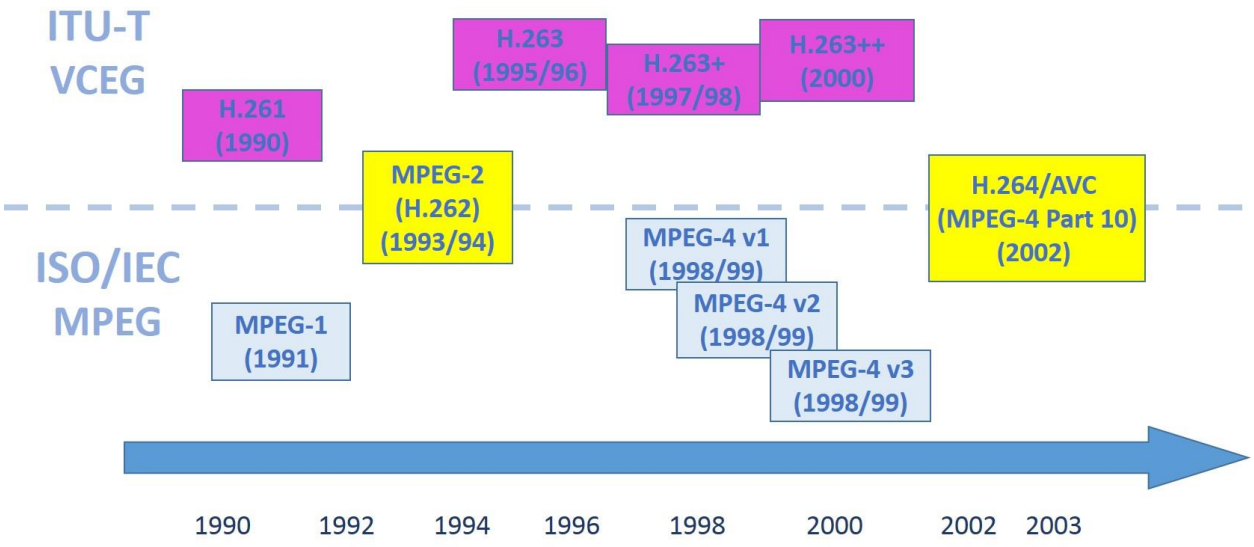


Figure 1. Chronological Table of Video Coding Standards

[4] 文獻中 H.264、MPEG-4 AVC、AVC、MPEG-4 part 10 或 H.26L 皆代表同一個視訊編碼標準

H.264/AVC 標準在架構上包含了 VCL(Video Coding Layer)與 NAL(Natwork Abstraction Layer)兩層，VCL 為視訊壓縮的部分，其技術核心包含動作估計(Motion Estimation)、線性轉換編碼(Linear transform coding)、預測編碼(Prediction coding)、去區塊效應濾波器(In-loop filter)、及熵編碼(Entropy)等技術，詳細內容會在後續小節說明，NAL 提供 VCL 編碼資料與實際網路之間的介面，進行編碼資料的格式化，加入必要的檔頭資訊(NAL header)，在封裝成適當的傳輸單元，而 H.264/AVC 的 NAL 提供多種封裝方式，以方便適用於各種的通訊傳輸協定，如 Figure 2.所示，而 Transport Protocol 不屬於 H.264/AVC 標準的規範。

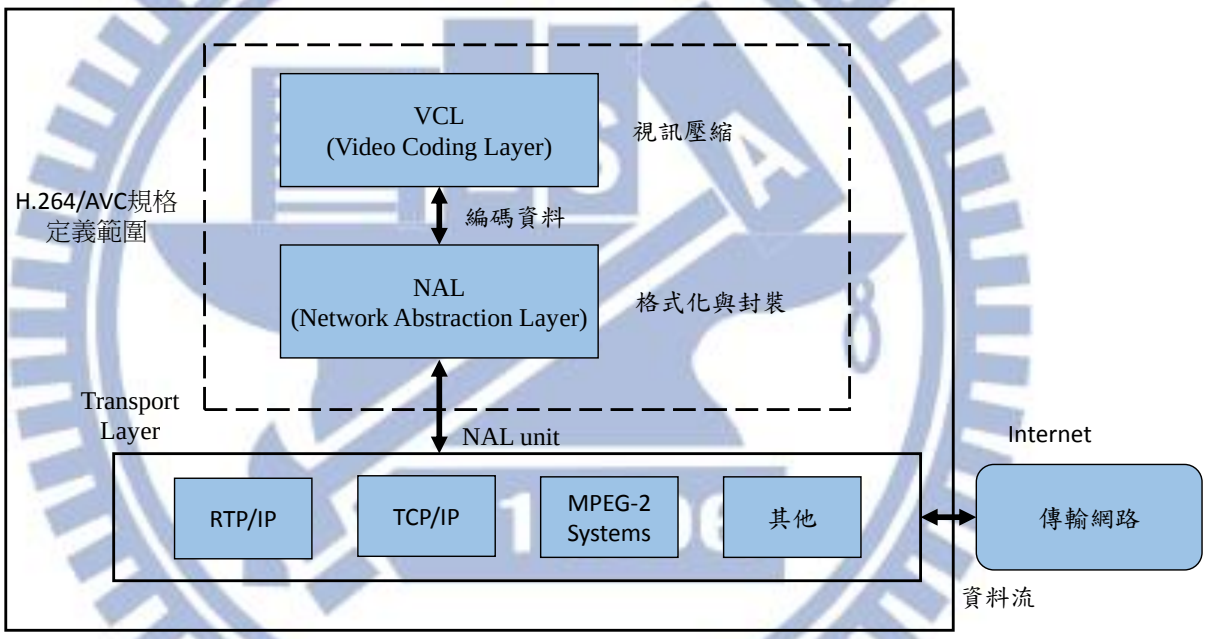


Figure 2. H.264/AVC 分層圖

2.1.1 Profile

H.264/AVC 根據使用的編碼工作種類來提供十幾種 Profiles， 其中較常使用在消費性電子產品的幾種 profiles 包括了 Baseline profile、Main profile、extension profile、High profile，不同 profile 規範了適合不同應用的編碼工具，前三種 profiles 的工具如 Figure 3 所示，至於 High profile 則是在 Main profiles 之上增加了 8×8 transform 等工具。而相對應的影片解析度與位元率等資訊由不同的 Level 所規範。Baseline profile

主要是應用在低複雜度，低延時的應用，所以適合應用在手持式裝置的多媒體支援，本篇論文則是使用 H.264/AVC Baseline profile decoder 作為 reference design，Main profileadd 則增加了 coding tools(如 B slice 、CABAC)，故具有較好的壓縮效率，適合應用於 HDTV 數位電視廣播，Extensio profile 則有高抗錯性的編碼工具(error resilient tools)，以及增加 SP 和 SI slice，故適合於 streaming media application。

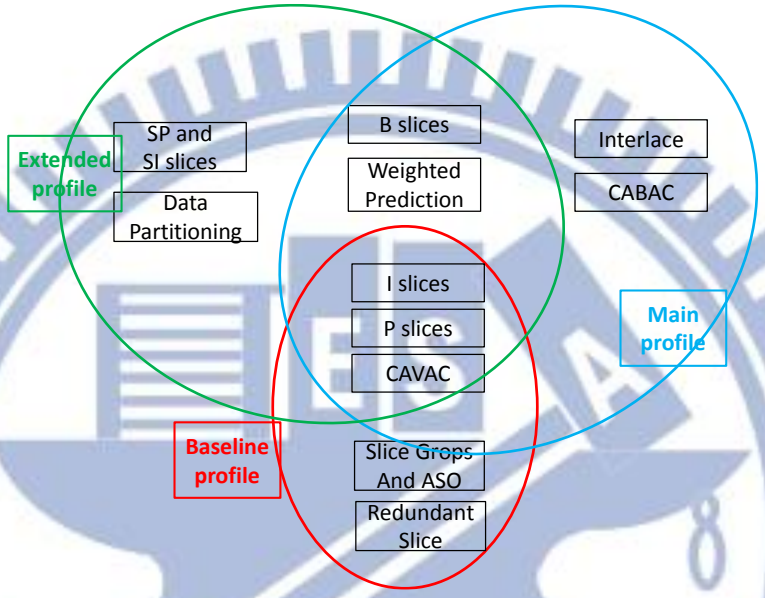


Figure 3. H.264 Profile

2.1.2 H.264/AVC 影像格式階層架構

H.264/AVC 的階層架構由小到大依序是 Sub-block、block、Macroblock(MB)、slice、slice group、frame/field-picture、sequence，以 Baseline profile 常用的 420 取樣的 MB 而言，是由 16x16 點的 Luma(Y)與相對應的 2 個 8x8 點 Chroma(Cb 和 Cr)所組成，MB 可在分割成多個 16x8、8x16、8x8、4x8、8x4、4x4 格式的 sub-blocks，MB 為 H.264/AVC 解碼的最小基本單位，則所謂的 slice 是許多連續的 MB 的集合，如圖 Figure 4.所示，slice 為 H.264/AVC 格式中的最小可解碼單位(self-docedable unit)，即一個 slice 單靠本身的壓縮資料就能做解碼，不需要仰賴其他的 slice，而 slice 主要可以分成三種，第一種為 I-slice， slice 的全部 MB 都採用 intra prediction 的方式編碼；第二種為 P-slice，slice 中的 MB 使用 intra prediction 和 inter prediction 的方式

來編碼，但每一個 inter prediction MB 最多只能使用一個移動向量(Motion vector)；第三種為 B-slice[5]，與 P-slice 類似，但是可以使用兩個移動向量，而本論文研究是使用 baseline profile，所以不討論 B-slice 的情況。而 H.264/AVC 另外有增加兩種特殊的 slice 類型(使用在 extension profie)，第一種為 SP-slice(Switching P slice)，為 P-slice 的一種特殊類型，可以用來串接兩個不同 bitrate 的 bitstream；第二種為 SI-slice(Switching I slice)，為 I-slice 的一種特殊類型，除了用來串接兩個不同內容的 bitstream 外，也可用來執行隨機存取(random access)。Slice group 由一個以上的 slice 所組成，在本論文沒有使用這個格式。Figure 5.為 H.264 Syntax。

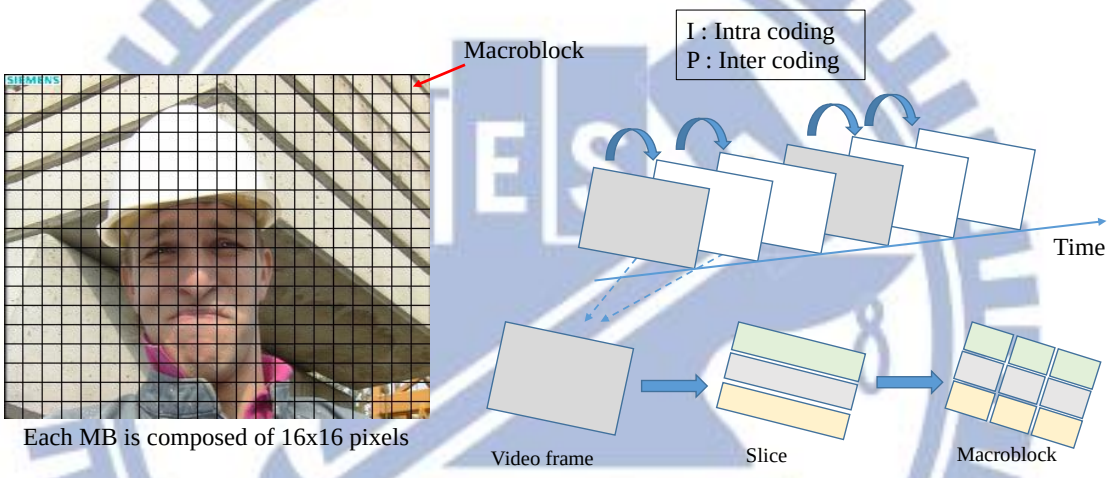


Figure 4. H.264/AVC Slice and MB

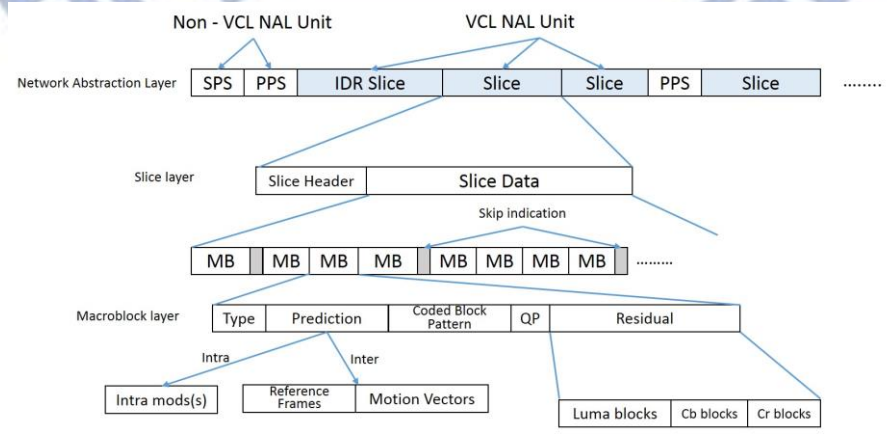


Figure 5. H.264 syntax

[5] B 是指 Bi-prediction，與 MPEG-2/4 B-frame 的 Bi-directional 概念不同，H.264/AVC 不拘限於指定要前後各一張才可以。

2.2 H.264/AVC 核心技術

從 H.261 至今的主要的視訊壓縮標準都是遵循 Block-Based Hybrid Video Coding 的架構，如 Figure 6.所示為 H.264/AVC Baseline Deocder，而圖中(虛線內)在各個模組執行的基本單位為 Macroblock(MB)，接下來小節將對主要的模組做介紹[6]，Control unit 是用來控制使用 I-MB 或 P-MB 解碼、In-loop filter 強度等等的操作，最後 output 為 YCbCr 的 frame。。

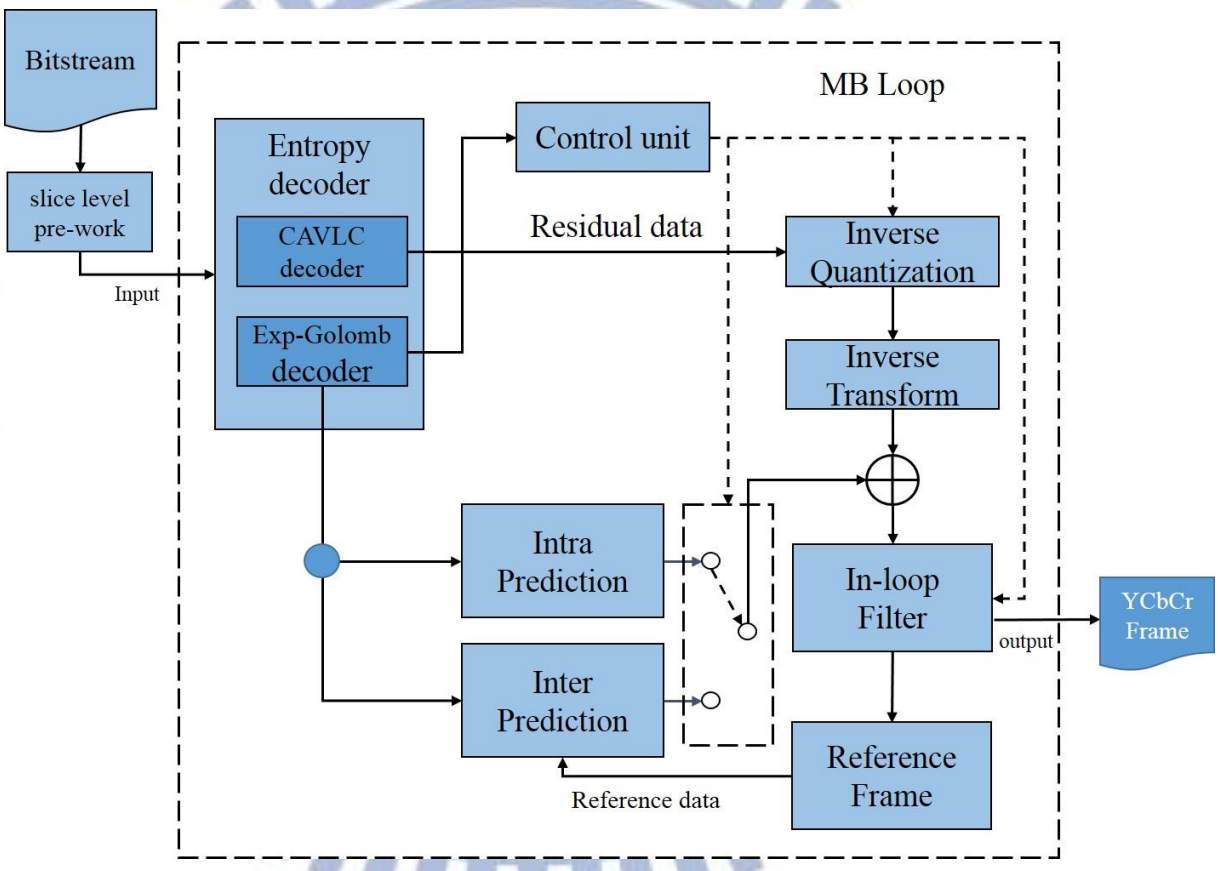


Figure 6. H.264/AVC Baseline Deocder

[6] Slice level pre-work 指的是對 SPS、PPS NAL 解碼、對 Refernece list 初始化等等的動作

2.3 熵編碼(Entropy)

此模組是針對量化後的轉換系數和其餘的 H.264/AVC syntax 進行編碼，其目的是去除編碼冗餘(coding redundancy)來提高壓縮比，而編碼冗餘是指量化後符號機率並非完全相等，且符號之間也存在著相關性。

(1) Exp-Golomb

在 H.264/AVC syntax 的部分指的是 Header information 或是 Motion Vector Difference、Prediction mode 等等的有關影像資訊，而這些資訊是使用 universal VLC(UVLC)中的名為 Exp-Golomb 編解碼技術，該技術是以採表的方式來完成，如圖 Figure 7.，因此需要額外的記憶體來儲存編碼表。

位元內容	數值	套用公式
1	0	2^0-1+0
010	1	2^1-1+0
011	2	2^1-1+1
00100	3	2^2-1+0
00101	4	2^2-1+1
00110	5	2^2-1+2
00111	6	2^2-1+3
0001000	7	2^3-1+0
0001001	8	2^3-1+1
0001010	9	2^3-1+2
0001011	10	2^3-1+3
0001100	11	2^3-1+4

Figure 7. Exp-Golomb code table

(2) Contex Adaptive Variable Length Coding (CAVLC)

在轉換系數編碼的部分，H.264/AVC 提供兩套編碼技術，Contex-adaptive binary arithmetic coding(CABAC)和 Contex-adaptive variable length coding(CAVLC)，CABAC 提供了較佳的壓縮效率，但相對的複雜度是高於 CAVLC，而在 baseline profile 是採用 CAVLC。

由於每個影像區塊在經過 Linear Transform 後會將大部分的能量集中在低頻的部分，而在經由量化後會產生大量的為零係數，而末端的非零係數通常都是為正負一，所以 CAVLC 是針對此特色來進行處理，先用 zig-zag oreder 的掃描方式[7]，如 Figure 8.所示，然後在將 4x4 區塊係數整理成一連串的係數，針對非零係數與為零係數間位置的關係採用 run-level 的技術來做編碼。解碼則是利用 coeff_token 得知非零數值與 TrailingOne 的個數，根據 TrailingOne 的個數可得知最後幾個的值為+1 或-1，其實不是 TrailingOne 的 Residual 則是按照其他方向由後到前依序解碼，根據 TotalZero 與 RunBefore，計算出每個非零數值與其前一個非零數值之間其零的個數，並將這些零放入。

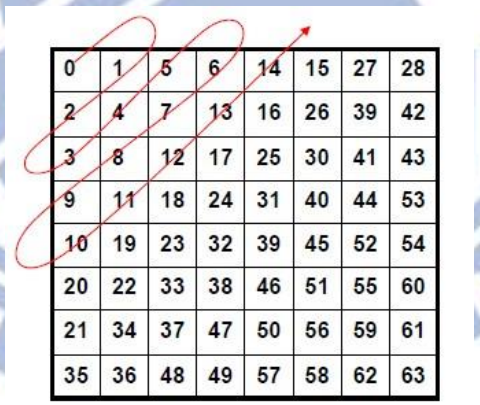


Figure 8. Zig-Zag order scan

[7] 把資料由 2D 轉向 1D 或反之

2.4 Inverse Transform and Inverse Quantization

H.264/AVC 轉換(Transform)的部分是採用類似 DCT(DCT-like)進行訊號能量集中，與傳統 DCT，即 Figure 10.，差異是採的是純整數空間的轉換，在解碼端得到的結果和編碼端相同，即不會產生 Mis-match 的情況，而且 H.264/AVC 在轉換時採較小的 4x4 區塊運算[8]，可使用較短的運算元長度(16-bit)並只需要使用加法與乘法，故降低在轉換上的複雜度，而小區塊的運算也能降低區塊效應(Blocking effect)的影響，其 4x4 inverse transform 的轉換公式如 Figure 9.。

$$Y = \begin{bmatrix} 1 & 1 & 1 & \frac{1}{2} \\ 1 & \frac{1}{2} & -1 & -1 \\ 1 & -\frac{1}{2} & -1 & 1 \\ 1 & -1 & 1 & -\frac{1}{2} \end{bmatrix} \left([X] \otimes \begin{bmatrix} a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \\ a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \end{bmatrix} \right) \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \frac{1}{2} & -\frac{1}{2} & -1 \\ 1 & -1 & -1 & 1 \\ \frac{1}{2} & -1 & 1 & -\frac{1}{2} \end{bmatrix}$$

Figure 9. Transform formulas

■ Forward transform (for encoder) :

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

■ Backward transform (for decoder) :

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

Figure 10. DCT formulas

[8] 彩度(chroma)是採用 2x2 的矩陣轉換

H.264/AVC 提供量化器(Quantizer)來縮減轉換參數和降低編碼資訊，其原因是人類視覺對於高頻影像部分敏感度較低，故可以做到降低編碼資訊而人眼感受不大。而 H.264/AVC 提供 52 個量化步階(step sizes)可供選擇，其對應的量化步階(step sizes)依指數非線性成長，相鄰的量化步階約差 12%，每隔六個量化步階其值變成原來的兩倍，這樣的設計讓 H.264/AVC 可以適用於不同壓縮率的環境，其完整流程如 Figure 11.所示。

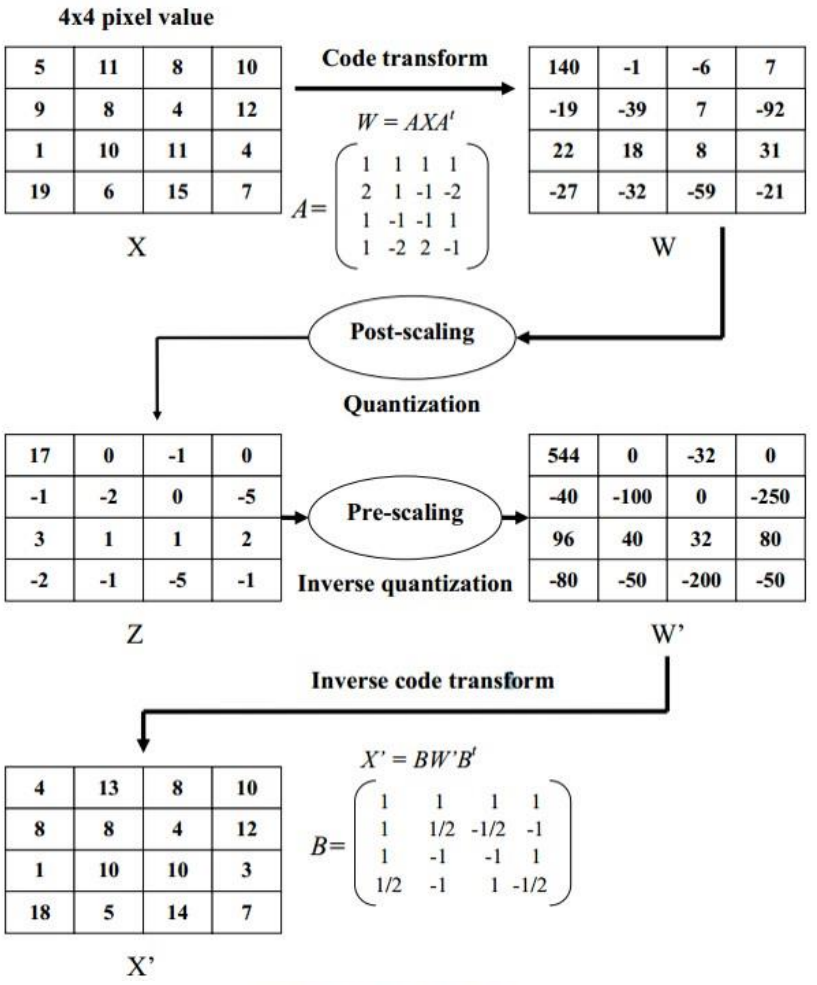


Figure 11. Transformation and Quantization

2.5 Intra Prediction

H.264/AVC 利用一張畫面內相鄰的 MB 的關聯性來減少原本大量的資料量，即拿相鄰資料來預測(Prediction)，而稱做 Intra prediction，而 Intra prediction 提供兩種不同 block size 選擇，即 Intra 4x4 mode 和 Intra 16x16 mode。

(1) Intra 4x4 mode

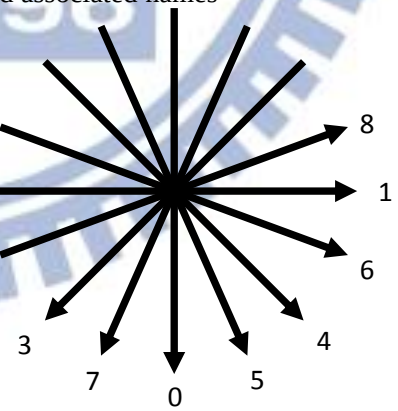
H.264/AVC 對於此方法提供了九種不同預測模式，Figure 12.所示，包含了 DC 模式[9]和其他八種不同方向的預測模式。舉例來說，我們選擇的是模式 0(Vertical mode)，則像素 a-p 則依照，a、é、i、m 是由 A 來產生，b、f、i、n 是由 B 產生，c、a、k、o 由 C 產生，d、h、l、p 由 D 產生。

Intra 4x4 mode	Name of Intra 4x4 Prediction Mode
0	Vertical
1	Horizontal
2	DC
3	Diagonal Down-Left
4	Diagonal Down-Right
5	Vertical Right
6	Horizontal Down
7	Vertical Left
8	Horizontal Up

(a) Specification of intra 4x4 mode and associated names

M	A	B	C	D	E	F	G	H
I	a	b	c	d				
J	e	f	g	h				
K	i	j	k	l				
L	m	n	o	p				

(b) Reference pixel



(c) Directions

Figure 12. Intra 4x4 modes and their directions

[9] DC mode 使用 mean(A-L)作為 a-p 之預測值

(2) Intra 16x16 mode

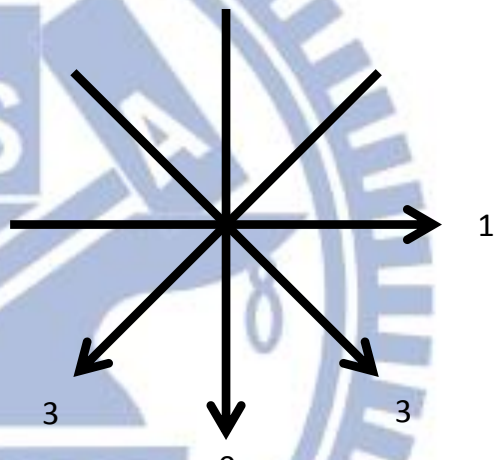
H.264/AVC 則對於 16x16 亮度(luma)區塊提供四種不同的預測模式[10]，如 Figure 13.所示，在處理方式上 Intra 4x4 與 Intra16x16 是差不多的，由相鄰的 A-P 像素來當作 a-p 之預測值，而如果有鄰近的參考區塊無法取得時(如畫面邊緣)，則某些模式則無法採用。

Intra16x16 mode	Name of Intra16x16 Pred. Mode
0	Vertical
1	Horizontal
2	DC
3	plane

(a) Specification of intra 16x16 mode and associated names

	A	B	C	D	E	F	G	H
I	a	b	c	d	a	b	c	d
J	e	f	g	h	e	f	g	h
K	i	j	k	l	i	j	k	l
L	m	n	o	p	m	n	o	p
M	a	b	c	d	a	b	c	d
N	e	f	g	h	e	f	g	h
O	i	j	k	l	i	j	k	l
P	m	n	o	p	m	n	o	p

(b) Reference pixel



(c) Directions

Figure 13. Intra 16x16 modes and their directions

[10] Plane mode prediction 適用於較平滑的畫面變動

2.6 Inter Prediction

動態影像(video sequence)是由連續的畫面(frame)所構成，每張畫面是景物(Object Scene)加上背景(Back Scene)所構成，在連續的畫面之間移動量較小時，背景通常不變或相似度極高，景物通常則是以有規律性的方向移動，綜合以上以上景物與背景的現象，在連續的畫面中畫面與畫面之間的關聯性很高，如 Figure 14 所示，因此在畫面與畫面間進行預測編碼，我們稱之為 Inter Prediction[11]。

在 H.264/AVC 視訊壓縮標準中提供了七種不同的 Macroblock partition 模式，有 P16x16、P16x8、P8x16、P8x8、P4x8、P8x4、P4x4，且每個 Macrblock 內部又可以切成多個 sub-macroblock，有 8x8、8x4、4x8、4x4 形式，Figure 15.所示

Figure 14. Sample sequence

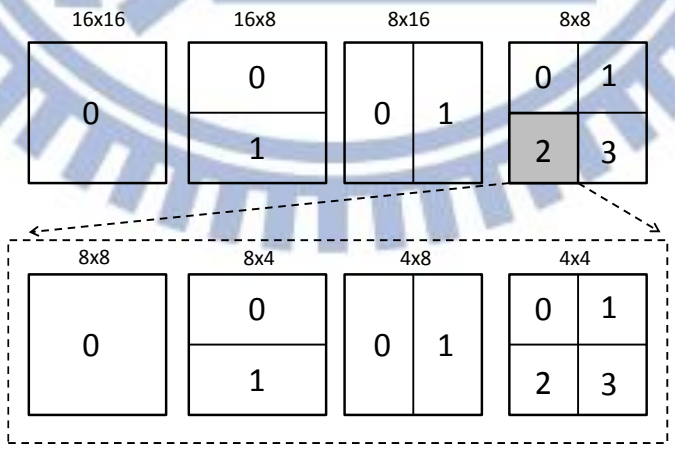


Figure 15. Partition of macroblock and sub-block

[11] Inter prediction 的編碼效能通常比 Intra prediction 高，因此其所佔之比重通常較高

Inter Prediction 解碼的過程主要分成兩個部分，第一個部分為取得 Reference Index 和 Motion Vector(MV)資料，Reference Index 代表目前我們要解碼的 MB 參考到哪張 frame，這資料能從 Bitstream 中取得(由 Entropy 解碼)，另外 MV 是代表目前解碼的 MB 和參考的 MB 之間的移動向量，而 H.264/AVC 為了減少 bitstream 大小，所以沒有把 MV 直接紀錄在 Bitstream 中，只會紀錄 Motion vector differences(MBD)，而我們要算出 Motion Vector Predictor(MVP)在加上 MVD 才能取得 MV[12]，而我們把第一個步驟稱為 Motion Vector Reconstruction(MVR)。

第二部分，當我們知道 Reference Index 和 Motion Vector 兩項資料後，先從 Reference Index 找出參考的 Frame，再利用 Motion Vector 取得 Inter Prediction Block，如 Figure 16.所示，接下來把經過 IQIT 後的 Residual Block 和 Inter Prediction Block 做重建(Reconstruct)動作，這動作我們稱為 MB Video Reconstruction(VR)，而以上兩個步驟就是 Inter Prediction 解碼的流程。

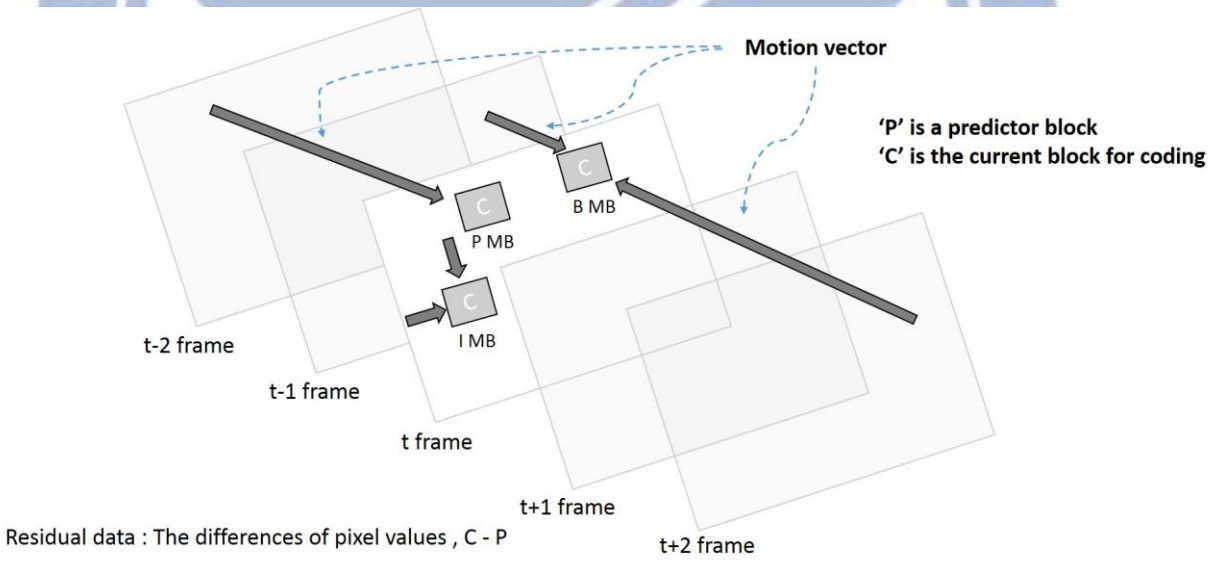


Figure 16. Inter Prediction[13]

[12] 計算方式為 MV = MVP+MVD

[13] 在本研究是採用 Baseline profile，故不討論 B slice

2.6.1 Sub-pixel interpolation

Sub-pixel positions sample 是不存在在 reference picture，所以需要使用 interpolation filter 使用附近的 image pixels 來建立 Sub-pixel，如 Figure 17 所示，灰色區塊為 full-pixel，白色區塊則是 sub-pixel，sub-pixel sample 需要使用 6 tap Finite Impulse Response(FIR) filter 計算，此 filter 的權重為(1/32，-5/32，20/32，20/32，-5/32，1/32)，計算方法為:

b = min (255, max (0, ((E-5*F+20*G+20*H-1*I+J+16)>>5)))

h = min (255, max (0, ((A-5*C+20*G+20*M-5*R+T+16)>>5)))

j = min (255, max (0, ((cc-5*dd+20*h+20*m-5*ee+ff+16)>>5)))

Table 1 所表示的是當 MB 長寬為 M 和 N 時，refernece pixel 的範圍大小，如要取得 e 時，需要 Mx(N+5)的 refernece data。

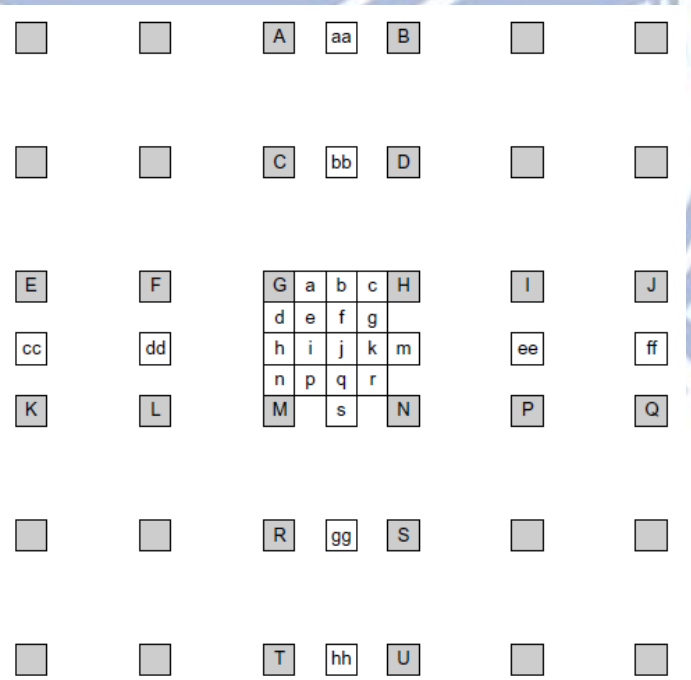


Figure 17. Full-pixel samples (shaded blocks with upper-case letters) and sub-pixel sample positions (un-shaded blocks with lower-case letters) fro quarter-pixel sample luma interpolation

Table 1. The interplation filters and reference data size of one MxN luma partition of single direction

Interpolation position	Interpolation filters	Minimized reference data size
G	No	MxN
a,b,c	6-tap horizontal	Mx(N+5)
d,h,n	6-tap vertical	(M+5)xN
e,f,g,i,j,k,p,q,r	6x6 tap filter	(M+5)x(N+5)

當我們要對 MxN chroma partition 進行 interpolate 時，需要(M+1)x(N+1)的 reference block，如 Figure 18 所示。

而計算方式為:

a = min (255, max (0, (((8-dx) * (8-dy) * A+dx * (8-dy) * B + (8-dx) * dy * C + dx * dy * D + 32) >> 6)))

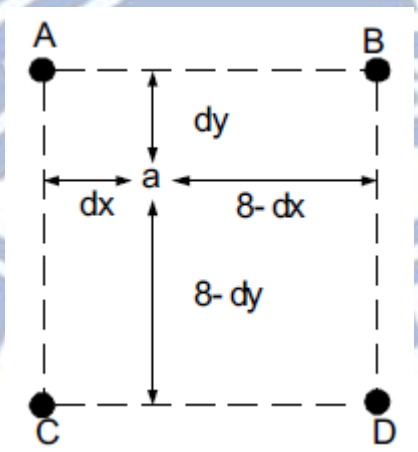


Figure 18. Sub-pixel sample position a in chroma interpolation and surrounding full-pixel position samples A, B, C, and D

2.7 In-loop Filter

H.264/AVC 是以區塊為單位進行處理，所以會有區塊效應(Blocking effect)，因此我們需要 In-loop filter 來消除區塊效應，來改善影像品質。

在 In-loop Filter 的流程中，由兩個主要的參數來控制 Filter，第一個為 Boundary Strength(BS)，是用來決定該使用何種強度的濾波器(Filter)來執行濾波的動作，而 BS 由 Figure 19.公式來決定。另一個重要參數為 Threshold(α、β、indexA)，而此參數是用來判斷是否為 true edge[14]，而 In-loop filter 流程如 Figure 20.所示。

Situation	Boundary Strength
p or q is intra coded **and** boundary is a Macroblock boundary	Bs = 4 , strongest filter
p or q is intra coded and boundary is **not** a Macroblock boundary	Bs = 3
neither p or q is intra coded ; p or q contain coded coefficients	Bs = 2
neither p or q is intra coded ; neither p or q contain coded coefficients; p and q have different reference frames **or** a different number of reference frames **or** different motion vector values	Bs = 1
neither p or q is intra coded ; neither p or q contain coded coefficients; p and q have same reference frame and identical motion vectors	Bs = 0 , no filter

Figure 19. Determining the boundary strength

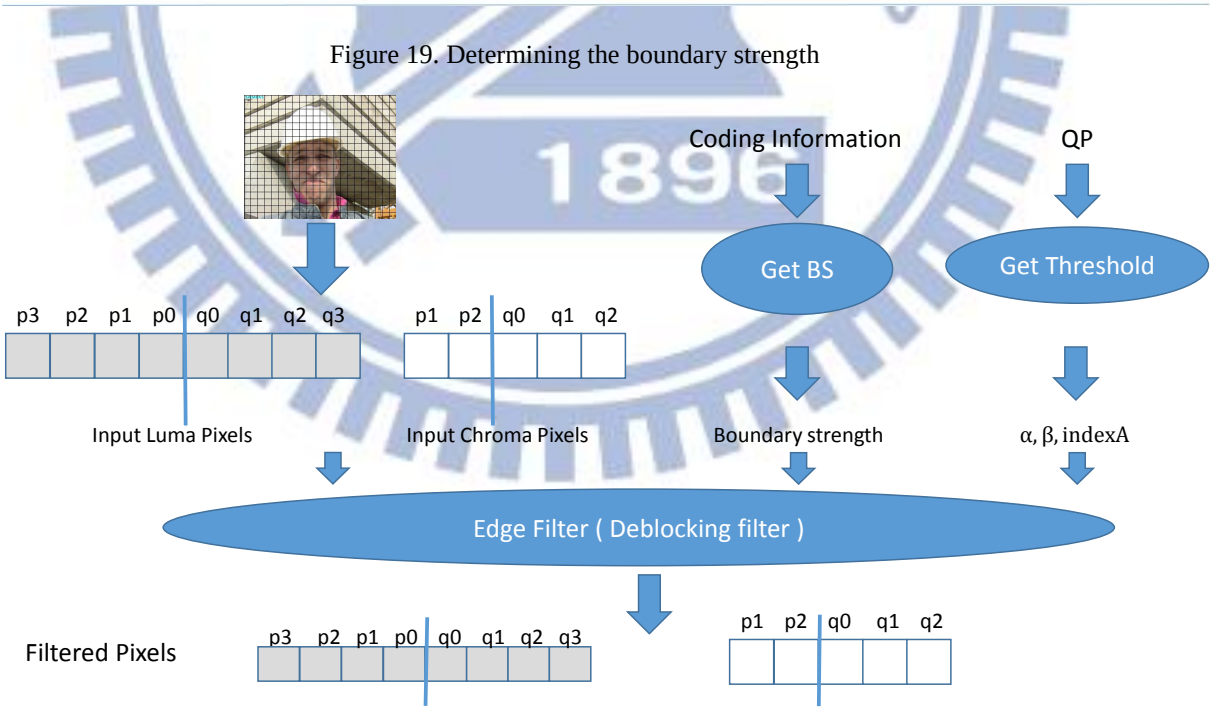


Figure 20. In-loop filter 流程圖

[14] True edge 代表此 edge 確實有受 block effect 影響

2.8 Google Android's H.264/AVC baseline decoder

在我們的研究中，single-core H.264/AVC decoder 是從 Google Android project[20] 取出來的 baseline reference decoder，此程式碼位於 Figure 21 所示 Libraries 中的 Media Framework 部分，Figure 22.為 OpenCore 的架構，H.264/AVC 為 Video Codecs 其中之一。

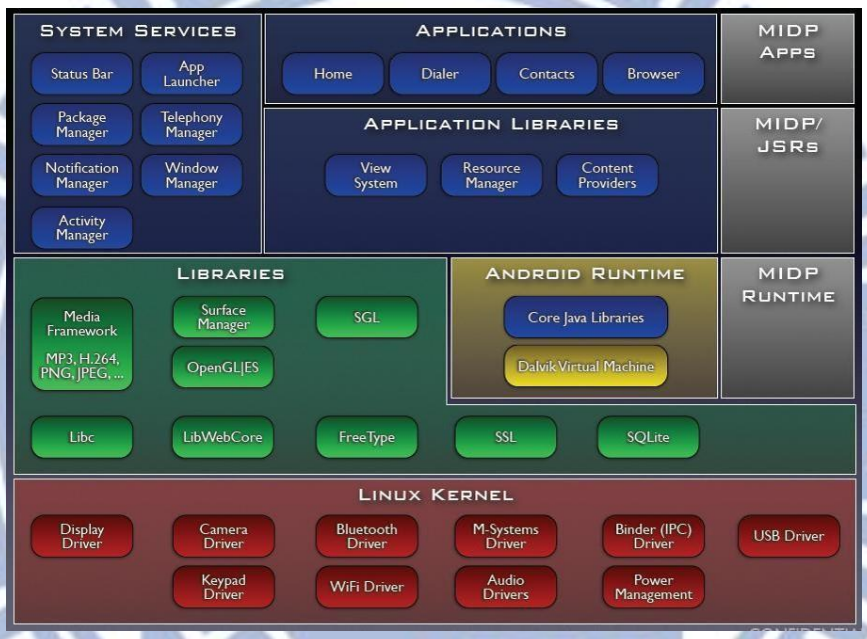


Figure 21. Android System Architecture

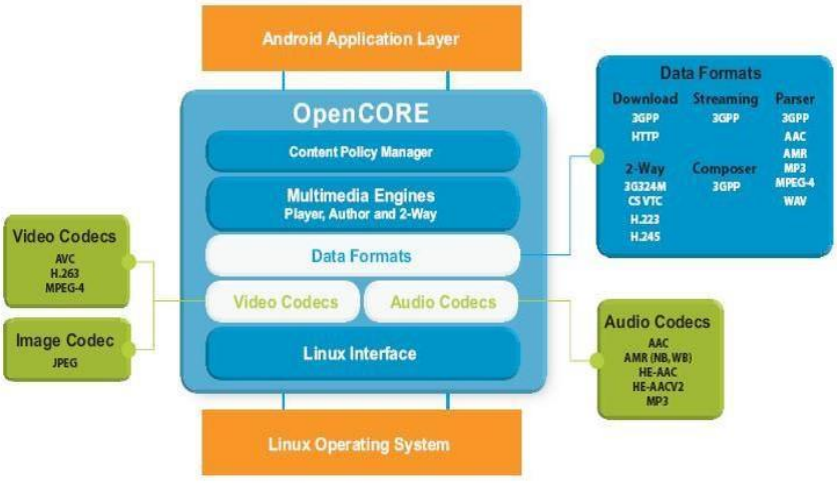


Figure 22. Android Media Framwork

三、Introduction to Parallel H.264 Decoding Scheme

此章節是分析 H.264 Baseline Decoder 平行化的方式，首先 3.1 節為 Related work 來探討 Parallel H.264 Decoder 的發展與技術，第 3.2 節就 H.264/AVC 架構上與資料結構上的相依性做討論。因為本論文是探討軟體平行解碼，而目前在傳統 Symmetric Multi-Processor(SMP)架構下的軟體平行解碼大多採用 Data-Level Parallelism(DLP)的技術，因此接下來在 3.3 節針對 DLP Video decoder 與其實作進行詳細討論。在下一章，我們會探討針對特殊架構處理器設計 Task-Level Parallelism(TLP)的視訊平行解碼器進行討論。

3.1 Related Work

現今主要的視訊壓縮標準大多只有支援 slice-level coarse-granularity 的 DLP 平行計算[15]，如果想採用 MB-level 的 fine granularity DLP 運算，在視訊格式中存在著資料相依性(Data dependencies)，而相關的平行演算法使用不同級別的劃分方式來確保沒有資料相依性的問題。在 frame level partitioning 是不適用於嵌入式系統中，因為其 inter prediction [1] 和 buffer size 的限制，尤其是在高解析度的視訊(如 FHD 或 HD)，其記憶體需求會是一個極大的負擔。而在 slice level partitioning，除了 MPEG-2 標準能充分的發揮平行化，其他主流的標準，如 MPEG-4、H.264/AVC 皆無法在此級別發揮良好的平行化，因為 MPEG-2 標準定義每個 MB row 為一個獨立的 slice，而在 H.264/AVC 可以允許一個 frame 只有一個 slice，而大多視訊壓縮也皆採用此壓縮方式[16]，因此最適合的級別將會是 MB level partitioning，但 MB level 仍有資料相依性的問題需要解決，將會在之後章節介紹。雖然 slice level partition 在實務上平行度有限，但在實作上，因為每個 slice 為獨立解碼的基本單位，所以在平行化的實

[15] 最新的 HEVC 標準提供 fine-granularity DLP 平行化計算的支援，如 Wavefront 與 Titles 技術但會犧牲 coding efficiency。

[16] 即一個畫面(frame)，只具有一個 slice

作上負擔不大，而 FFmpeg H.264 decoder[21]就有提供 slice-level parallelism 的支援。

使用 MB level 的平行化方式主要分成兩類，Data level parallelism(DLP)和 Task level parallelism(TLP)兩種，DLP 是讓不同的 threads 或 cores 同時去執行相同的 function 但不同範圍的 input data，Van Der Tol et al.在 [2] 提出 Stairway-shaped data partition，用來減少 inter-partition 的相依性。Seitner et al.在 [3] 中，分析了多種不同 partition 方式，如 single row(即每個 thread 或 process 對 single row 作解碼動作)、Multi-column partition 等等，探討其 cache performace、buffer size、data transfer for reference data 在不同 partition 的情況。而在 [4] [5] [6] 中，在探討 DLP 之 scalability，並提出 2D-Wave 方式，且更進一步的加入 frame level 的平行化，稱之 3D-Wave，而此種方法可以大幅提升 MB 平行化的程度，但存在了一些限制，如需限制 encoder 端的 MV range 和 reference list number，且在核心數和記憶體有限的嵌入式系統中，效益並不顯著。而也有相關研究在 Cell Broadband Engine[17]上做分析 [7] [8]，利用其多個 Synergistic Processing Units(SPUs)，來同時執行不同的 MB，但其平台在撰寫 Parallel programming 其 Local storage 和 Shared memory 資料之間的管理，對於 Programmer 是一個負擔。Jike Chong et al. 在 [9] 針對 data scheduling 的挑戰 (即平行度與複雜度等等) 與技術進行討論。

TLP 為分別讓指定的 thread 或 core 去執行不同的 function(或稱 task)，或以 pipeline 的方式稱之。Schöffmann et al.在 [10] 提出了 static partition pipelining，分成五個

[17] 為 IBM 和 SONY 共同開發之平台

stage，而此方法會存在 load balance 的問題，且在 thread 個數大於核心數的情況下，平行化效能會因為 thread scheduling 的影響，而造成加速幅度有限。Minsoo Kim et al. 在 [11] 提出 Dynamic load balancing method 在其雙核心 DSP 平台，系統架構包含了雙核心 DSP、VLD 硬體、MC 硬體、Deblocking 硬體和 SDRAM，因此在整個解碼流程中，DSP 只負責少許的部分，而 Minsoo Kim et al 在 [12] 擴充其系統架構至四核心的 DSP，且 Debloking 和 MC 也由 DSP 處理，另外提出了減輕 system bus 頻寬的平行化技巧。Chen, Ding-Yun, et al 在 [13] 在雙核心平台提出藉由監控兩核心之間 buffer 來動態更改其 partition 來解決 load balance 問題。Chun-Jen Tsai et al. 在 [14] 提出 Application Processor Architecture for Multi-Core Software Video Decoding，並利用提出的 Inter-preocssor communication (IPC) controller 能使得 TLP deocder 減輕 system bus 頻寬，以及減少資料傳輸所造成的開銷。

3.2 Macroblock dependencies of H.264/AVC

H.264/AVC 在 Macroblock level 的資料相依性可以分成三類，如 Figure 23.和 Figure 24.所示。Figure 23 (a) Intra prediction 是利用目前解碼的 Macroblock 其左邊、左上、上方、右上之 pixels 來作為 prediction 之依據，而此部分的解碼能利用放在同一顆核心來執行，可以減少用來解決相依性的 inter-preocessor communication。Figure 23 (b) Deblocking filter 部分則是需要上面 MB 的最後四個 rows 和左邊 MB 的最後四個 columns 的 pixels 值，用來解決 blocking effect，另外還需要目前 MB 其上面與左邊 MB 的 Motion Vector 和 Reference Index 來計算 Boundary Strength(BS)。最後 Figure

23 (c) Inter-prediction 則是需要確保目前 frame 所要參考的 frame 其位置(MB or Sub-MB)已經解碼完畢，Meenderinck et al.在 [4] 中，有利用在 H.264/AVC encoder 端限制 SearchRange 和 NumberReferenceFrames[18]來增加 Scalability。

另外 Entropy 解碼部分因為現今視訊標準採用的壓縮編碼法(如 VLC，AC)再兩個 synchronization points 之間必須進行完全循序的資料處理。而且不少 video bitstreams 每張影像只插入一個 synchronization point，因此此模組不會加入至平行化的實作中，意思是先執行完 Entropy 在進行之後模組的平行化，因此不討論 Entropy 模組的資料相依性。

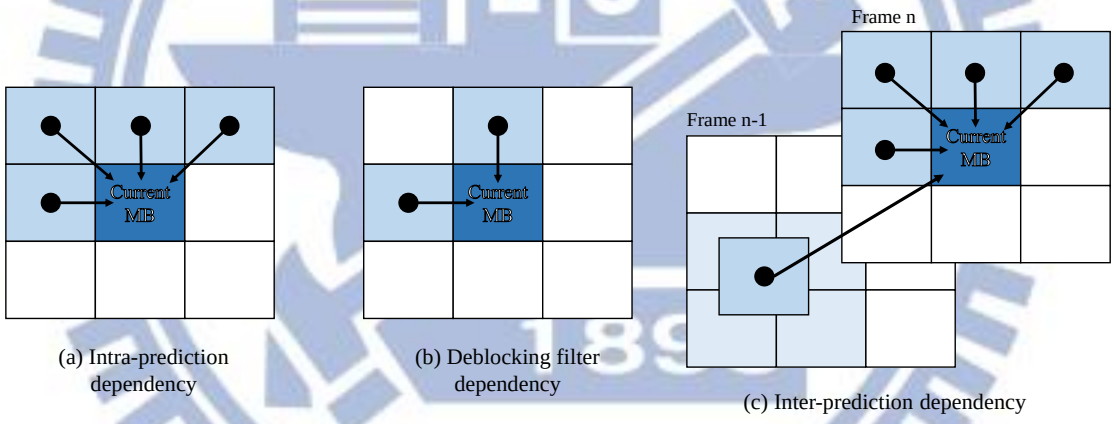
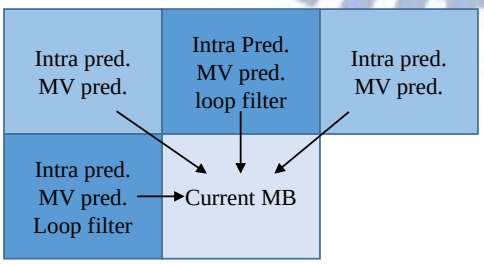


Figure 23. Dependencies of H.264/AVC



Intra pred. : pixel prediction for the current MB requires data from this block.
MV pred. : motion vector prediction for the current MB requires data from this block.
Loop filter : de-blocking of the current MB requires data from this block

Figure 24. Data dependency neighborhood of the MB decoding process

[18] 本研究其使用之參數為 SearchRange : 32 和 NumberReferenceFrames : 6

3.3 Data-level Parallelism

在 3.3.1 節會簡單的介紹 DLP 並介紹使用何種 scheduling，第 3.3.2 節會介紹本論文實作的 static scheduling，而第 3.3.3 則是對 DLP 的總結。

3.3.1 DLP Overview

Data-level Parallelism(DLP) decoder 又可以稱做 Wavefront-based decoder，DLP 是讓不同的 threads 或 cores 同時去執行相同的 function 但不同範圍的 input data，如 Figure 25.所示。根據 4.2 節可以知道 H.264/AVC 在結構上存在的相依性，而依照其相依性在 MB level 的平行化，我們可以很容易的推出 Figure 26.[19]，此圖為一個 5x5 的 frame，其代表的是當 T1 時間點時只存在一個 MB(0,0)被解碼，而當 T7 時間點時，則可以三個 MB(4,1)、(2,2)、(0,3)同時被解碼，以此類推，當解析度放大至 Full High Definition(FHD，1920x1080)時，在整個解碼過程中平均約有三十個 MB 能同時進行解碼，甚至最多能達到同時有六十個 MB 能並行解碼，如 Table 2.所示(Slots 代表達到最高 MBs 時 times slots 個數)。

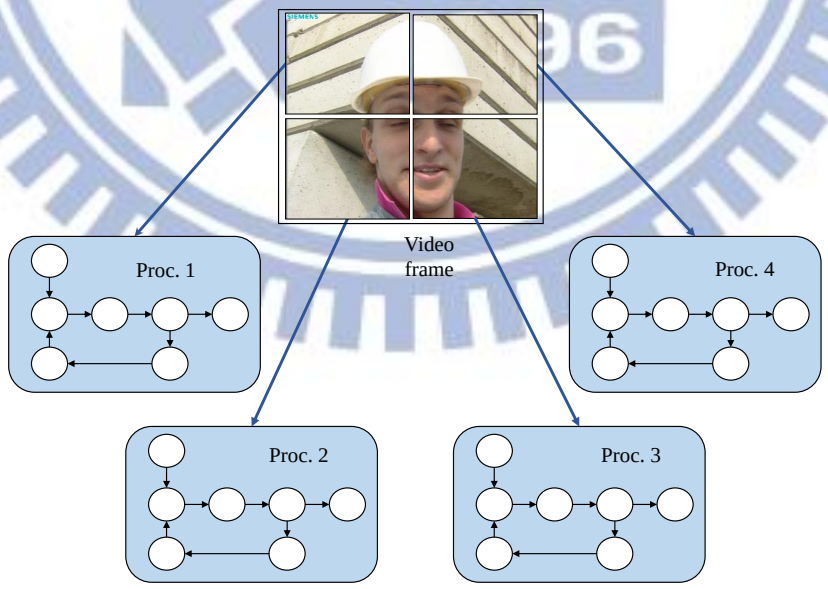


Figure 25. Data-level Parallelism

[19] 假設每一個 MB 解碼時間都是相等的，但事實則是每個 MB 解碼時間為 Variable length

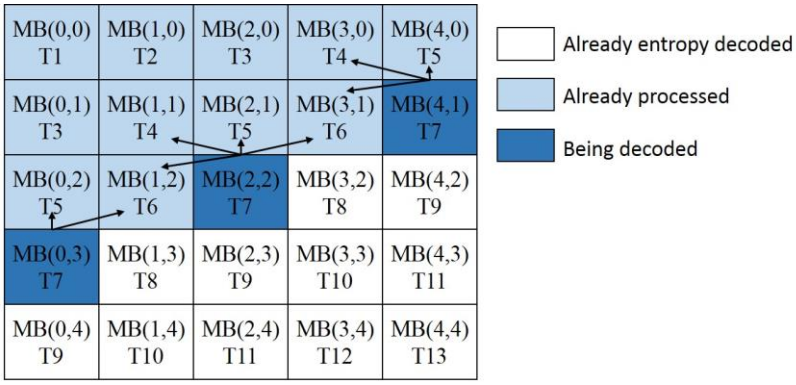


Figure 26. Exploiting MB parallelism in the spatial domain.

DLP 在 MB-level 擁有許多優點，最明顯的就是增加 scalaility(參考 Figure 26.)。而 Á lvarez Mesa 在[5] 計算 Maximum speedup 和最高處理器的使用數都能證實 DLP 的高擴展性。另外就是在 dynamic scheduling 中能做到較好的 load balance，因為對每一個 MB 解碼時間不固定，而是要取決於 MB 的形態(I 4x4 or I 16x16 or P 8x8 etc.)與資料量(如 residual data)。但本論文是採用 Static scheduling，即指定某範圍的資料(MB)給指定的 Thread 或 Core 解碼，其原因是本研究是以嵌入式手持式平台(如 smart phone)為目標，使用 dynamic scheduling 需要維護 FIFO queue(用來讓 worker thread 取資料來解碼)，且 dynamic scheduling 複雜度會相對於 static scheduling 高，而在手持式裝置上處理器核心數有限的情況，無法從 scalability 得到好處，因此本論文採用是 static scheduling，可以減少額外的記憶體使用，也能減輕 scheduler 的負擔。

Table 2. Maximum parallel MBs for several resolutions using DLP

Resolution		MBs	Slots
QCIF	176x144	6	4
CIF	352x288	11	8
SD	720x576	23	14
HD	1280x720	40	6
FHD	1920x1080	60	9

3.3.2 Static scheduling 實作與技術

首先先把 H.264/AVC decoder 分成兩部分，第一部份為 Main thread，是用來執行 Entropy 跟 Scheduling，第二部分則是 worker threads，是執行 IQIT、MC、Intra 或 Inter prediction 和 Loop filter，如 Figure 27.所示。由 Main thread 執行 Entropy 的原因是在 Entropy 模組中需要以 Raster scan 的順序進行解碼，且往後模組需要的資訊皆要在 Entropy 模組中取得，所以 Entropy 這部分無法加入到平行化的部分且要先被完成，而這也是 DLP 缺點之一。

論文中 DLP 的 Data partition(splitting strategy)是採用 Single-row approach，如 Figure 28.(圖上之數字代表 Processors 編號)、Figure 29.所示，即每個單一行上的 MBs 由一個 Processor 解碼，更正式的說法為，我們假設 N 為 Processors 個數，當 $y \bmod N = i$ 時，Processor $i \in \{0,\ldots,N-1\}$ 其解碼資料對映到 y th 行 MBs 上。

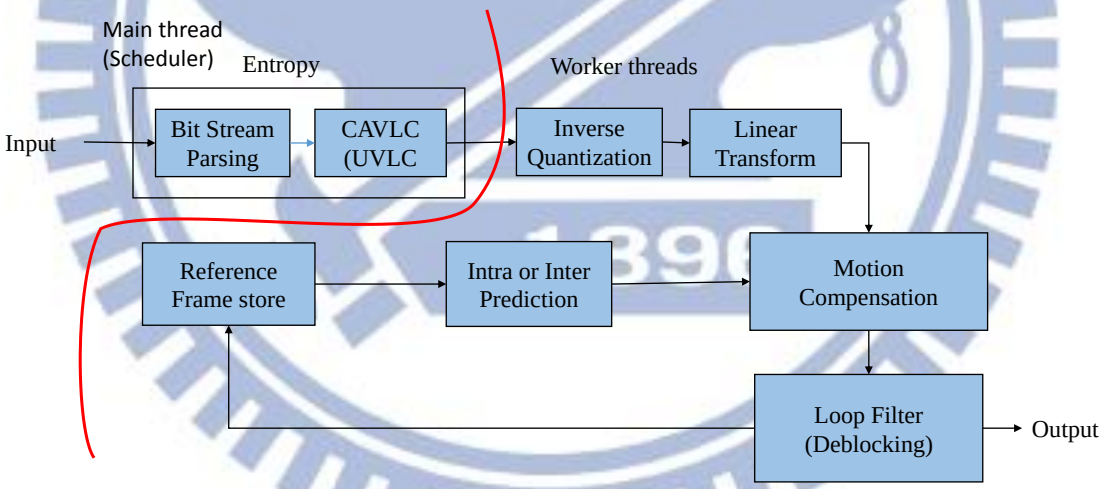


Figure 27. DLP- Flow chart

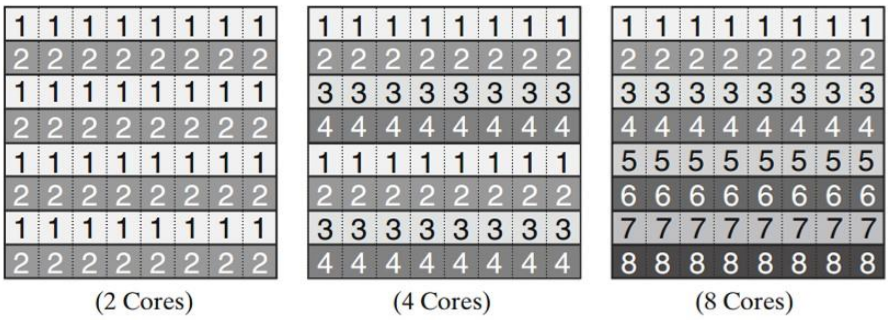
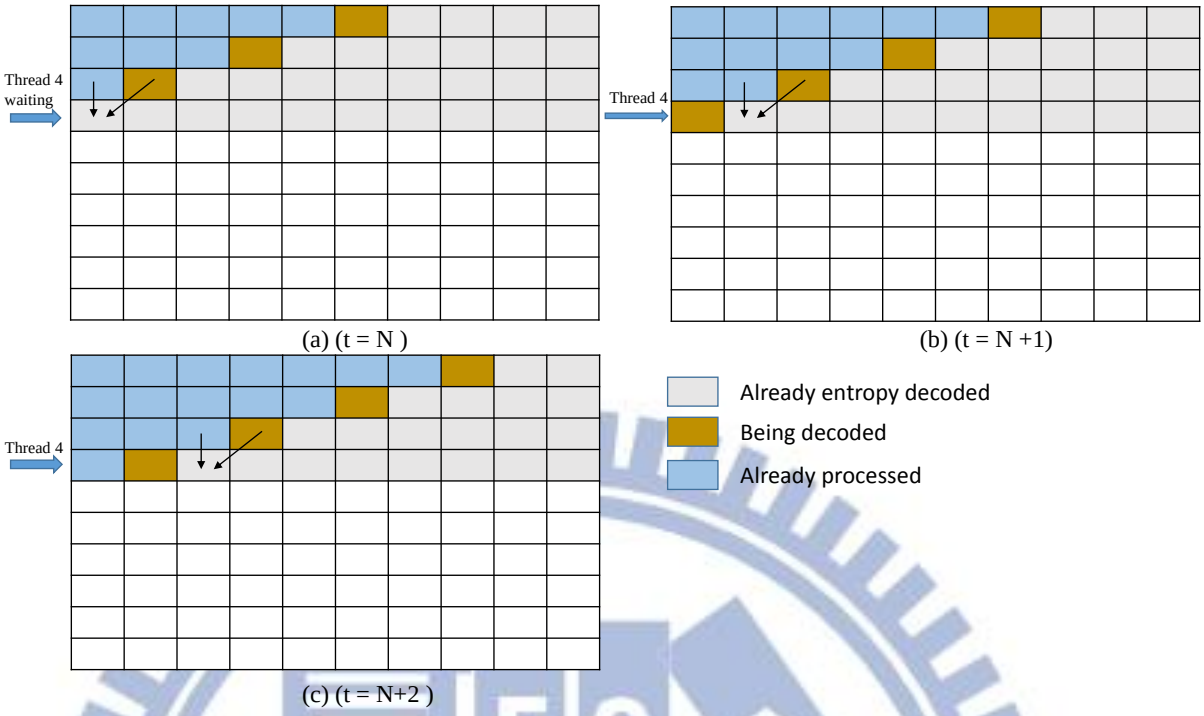


Figure 28. The Single-row splitting approach

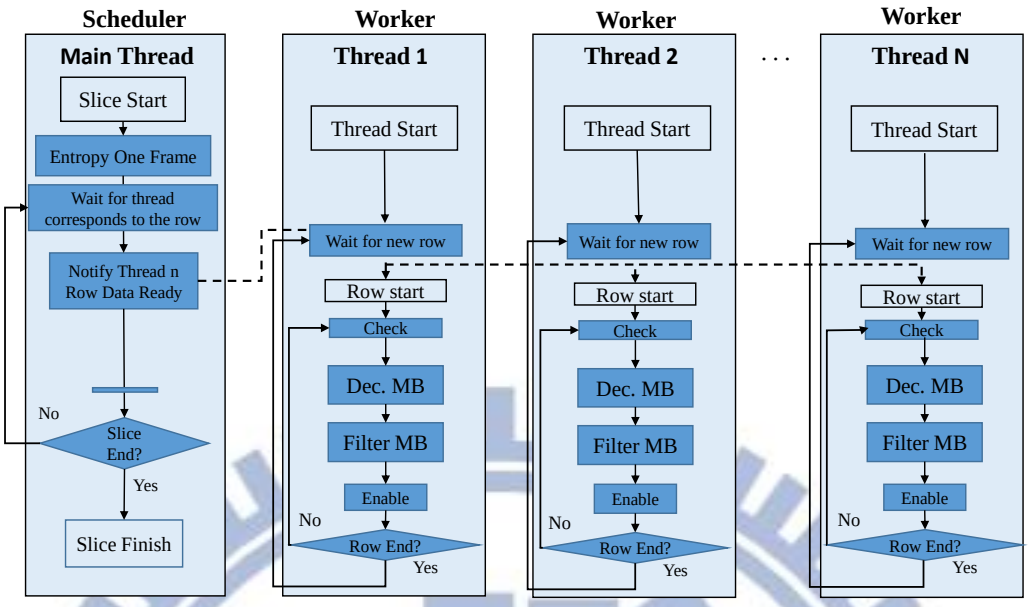


(a) (t = N)　(b) (t = N +1)

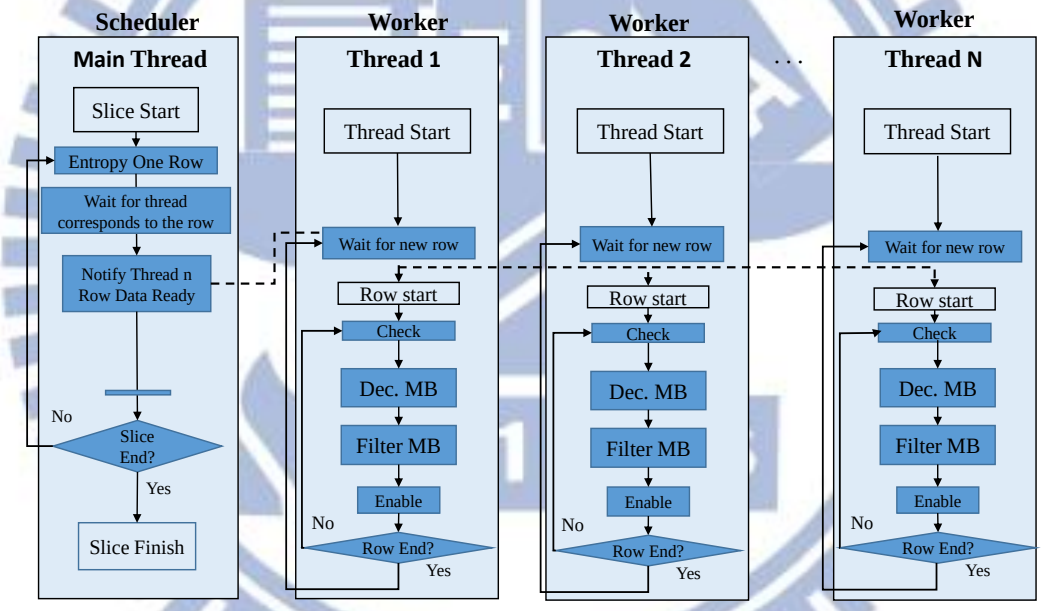
(c) (t = N+2)

Figure 29. Example of the Single-row splitting approach

在這 Single-row approach 下，我們採用了兩種 Entropy decoding schemes，我們預設有三顆核心可以由 parallel wavefront video decoder 所使用，第一種機制為讓一顆核心為 Main thread 去對整張 frame 執行 Entropy-decode，然後分配 rows 給三顆核心同時進行解碼動作，而我們把這個方法在本論文稱作 Non-interleaving wavefront decoder。第二種機制則是讓其中一顆核心(為 Main thread 或稱 Scheduler)去 row-by-row 的執行 Entropy-decode，而每當 Entropy-decoder 完一個 row 時，則馬上分配給其他 worker thread(core)執行解碼，而我們把此方法稱作 Interleaving wavefront decoder，此方法可以解決 Static scheduling 所造成的 Main(或稱 Master) thread 大部分時間都是在等待其他 worker threads 完成解碼，因此可以讓整體效能提升。



(a) Non-interleaving wavefront decoder



(b) Interleaving wavefront decoder

Figure 30. Two different entropy decoding schemes

流程如 Figure 30.所示，而圖中 Wait for new row 階段是利用 Hardware Mutex 來維護，建立 row 個數的 Mutex[20]並使其狀況為 lock，每當一個 row 執行完 Entropy-decode 時，Scheduler 呼叫 unlockMutex(row_num)動作，這時候 worker thread 就能順利對該 row 進行解碼。而 Check 階段則是進行 checkNeighborMB 的動作，當 MB 要進

[20] Hardware mutex 提供 64 組 Mutex 使用

行解碼時，要確保其附近的 MBs 是已經解碼完(原因可看 4.2 節)，否則會進入 busy waiting 直到條件觸發，而這部分就是 DLP 的瓶頸所在，假如把 flags 放置在 DDR2，則需要不斷的呼叫 cache invalidation function[21]，造成效能的下降，因此在 DLP 設計中的 flags 都放置在 On-chip shared SRAM(uncached memory)，但當多顆處理器頻繁的讀取 flags，會導致整體效能受影響。

另外 DLP 在資料結構上的設計，主要分為三種 slice-level data structure 與 MB-level data structure 與 Intra prediction 所需要的資料，slice-level 主要是儲存一張 slice 中共用之資料(如 slice_id、slice type、PicWidthInMbs)，所佔之空間為 552 bytes，而 MB-level(如 mvL0、ref_idx_L0、mbMode、CBP)則總共為 2KB，假設影像解析度為 CIF(352x288)，則 Intra prediction 所需要的資料量為 1056 bytes[22]，因此當對一張 slice 進行解碼時，需要 794 KB[23]資料儲存在 DDR2(shared memory)，假如在進一步把解析度放大至 VGA(640x480)，約需要 2.4MB 的資料量，有此可知，DLP 對於記憶體的需求其實是不容忽視的，尤其是在高解析度的視訊上。

[21] 本論文中使用的 cache 為 Non-coherent cache

[22] 2 x (PicWidthInMbs<<4) x 3/2 = 1056 bytes

[23] 552 bytes + 2 KB x 396 + 1056 bytes = 794 KB , 396 為一張 slice 之 MB 個數

四、Proposed Parallel H.264 Decoder Scheme

本論文所採用的是在特殊架構下的 task-level parallel video decoder，系統架構如 Figure 31 所示，架構的設計細節請參考[14]。在第 4.1 節會先簡介 Task-Level Parallelism(TLP)的 H.264 解碼器。第 4.2 節則是會介紹本論文中的三種 TLP 實作與其技術，即 static pipeline partitioning、Dynamic pipeline partitioning using Macroblock mode 和 Dynamic pipeline partitioning using monitoring buffer，而第 4.3 節則是對 TLP 平行解碼時記憶體的使用做一個討論。

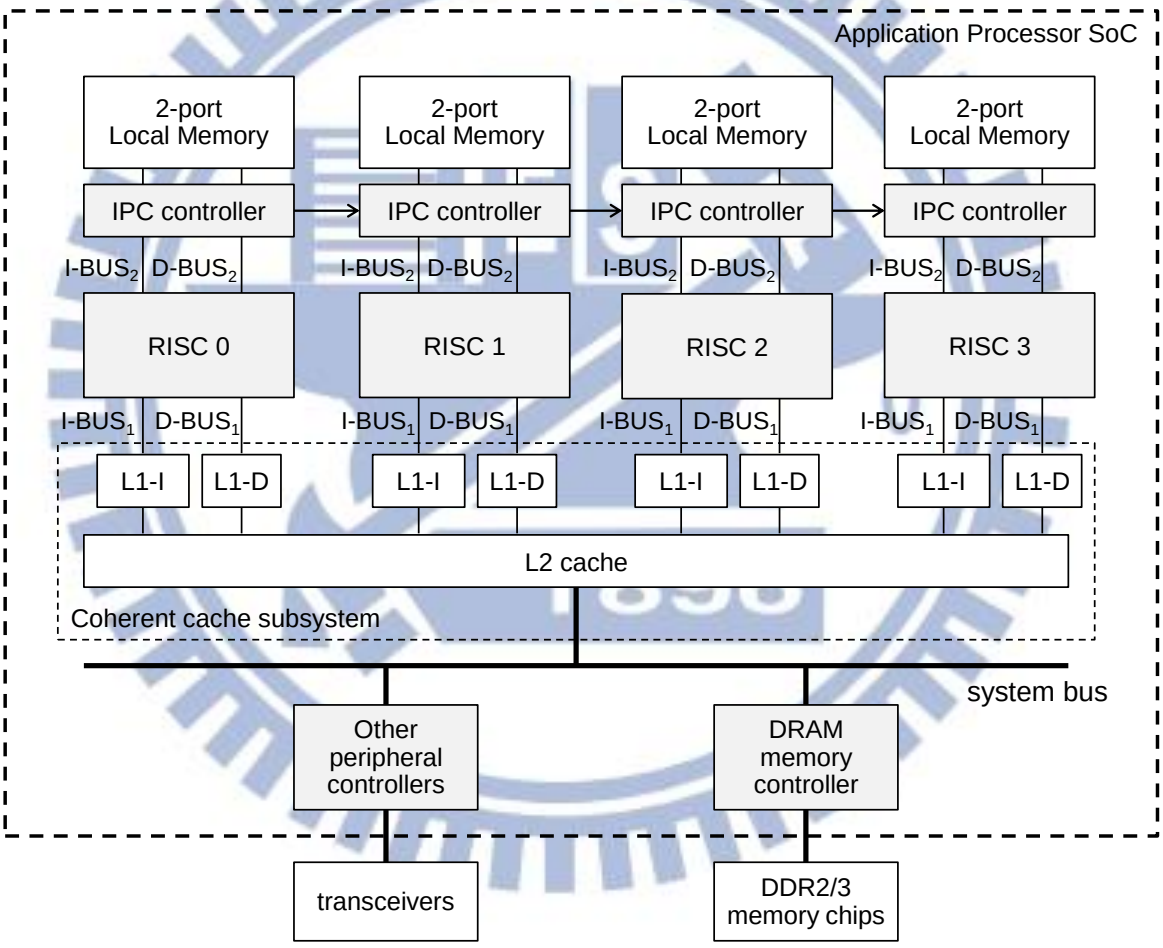


Figure 31. Application processor 架構圖

4.1 TLP Overview

Task-level Parallelism Video Decoder 可以稱作 Pipeline-based Video Decoder，TLP 是讓每個 processor core 執行一個 stage，或稱執行 task，如 Figure 32.所示，而進行

解碼的基本單位為 16x16 Macroblock。TLP 設計上主要有兩個議題，即 scalability problem 和 load balancing problem，在 pipeline 的平行化中 scalability problem 主要是受限於 pipeline stages 的個數，如 Figure 33.所示，H.264/AVC 在 pipeline 解碼時，對單一 Marcoblock 的解碼程序分成七個步驟，即 Entropy Decoding(ED)、Intra Mode Reconstruction(IMR) 或 Motion Vector Reconstruction(MVR) 、 Inverse Quantization(IQ) 、 Inverse Transform(IT) 、 Intra Prediction(IP) 或 Motion Prediction(MP)、MB Video Reconstruction(VR)、In-loop Filter(ILF)，但因為為了減少 inter-stage communication 負擔與達到較好的 load balancing，大部分的實作會把解碼流程合併成三或四個 pipeline stages，但假如 inter-stage communication 負擔可以忽略的情況，事實上可以在對目前七個步驟在做細分，切成為 fine-granularity stages，使得更輕易達到 load balancing，例如 H.264/AVC 之中運算成本較高的 In-loop filter，Wang 在[15]提出在 Deblocking filter 上使用 wavefront-based 的平行化方式，而其切割非相依性資料的方式可以利用在 MB-level pipelined 上，在 In-loop filter 實作上，會先對 4x4 垂直邊(vertical edges)進行處理，再來對 4x4 平行邊(horizontal edges)做處理，因此簡單的切割方式是垂直為一個 stage，處理平行邊也獨立一個 stage(利用[14]中討論 independent pixel)，所以至少[24]能再獨立出兩個 stages，而在現今大部分的手持式應用處理器不超過八顆 RISC 核心的情況下，scalability 不會是我們研究上對 TLP 設計上的主要關注議題。

[24] 在實作上 MB boundary edge 會另外處理，也是先執行垂直邊再處理平行邊，這部分也可以獨立

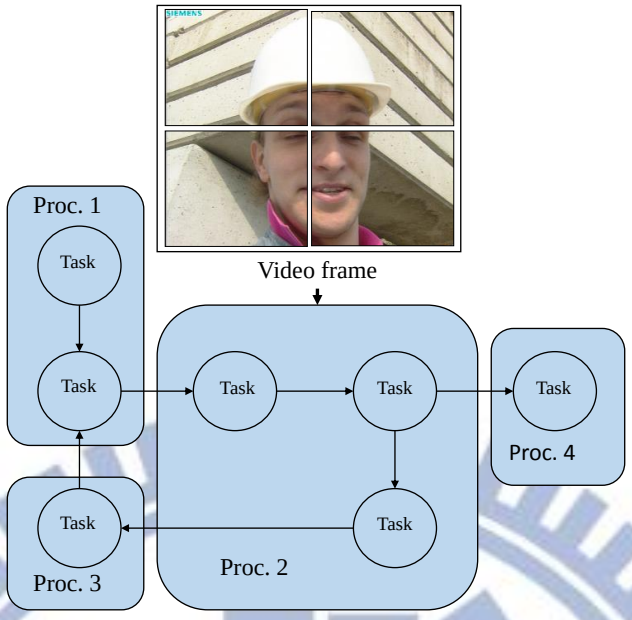


Figure 32. Task-level Parallelism

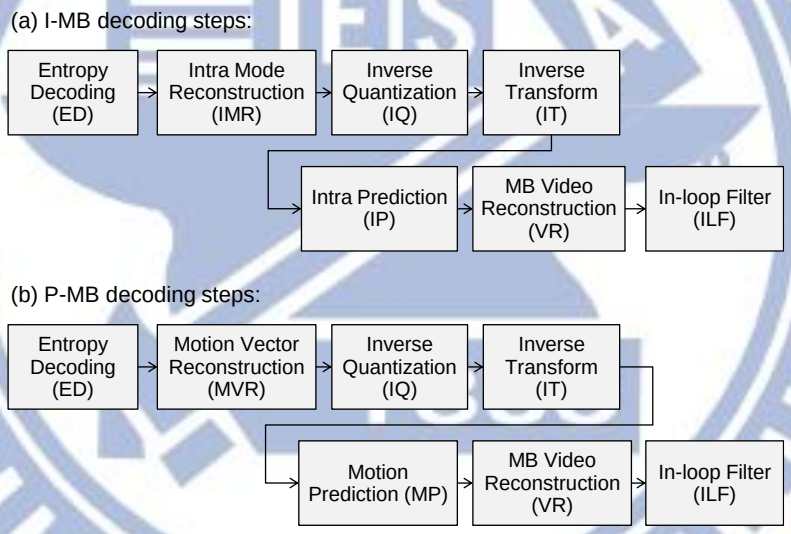


Figure 33. The decoding steps of an I-MB and a P-MB of H.264/AVC

Load balancing 主要是因為對每個 Macroblock 解碼是 variable complexity，造成有些解碼時間是消耗在 synchronization 上面，而我們這部分使用每兩個 processor 之間設置一個 FIFO circular buffer，用來吸收因為 variable complexity 所造成的時間消耗，如 Figure 35.所示，我們使用了四顆核心，所以我們使用了三個 FIFO circular buffer，Input buffer 用來放置壓縮後視訊的 bitstream data，buffer #1 與 buffer #2 則是放置 MB 解碼所需要的資訊，在我們的實作中，為了避免需要 pack 與 unpack 處理，因此兩個 buffer 的資料結構是相同的，而每個 buffer node 資料量為 2KB，另外

除了 Input buffer 放置在 DDR2，其餘兩個 buffer 皆是放置在處理器的 local memory (為 on-chip memory)，放置在 local memory 原因為軟體實作的 pipeline 不像硬體 pipeline 般，能將暫存資料(或 intermediate data)儲存在 on-chip pipeline registers，通常是儲存在 memory bank(不管是 on-chip SRAM 或 off-chip DRAM)，因此讀寫須透過 system bus，而不幸的事是 system bus 可能會被其他 tasks 或 device controllers 使用，而造成效能上的影響，因此把資料放置在 local memory 則可以減少對 system bus bandwidth 的使用。

4.2 TLP 實作與技術

Pipeline-based video decoder 的效能是非常仰賴於每個 pipeline stages 的平衡，因為每個 Macroblock 解碼複雜度變化是非常明顯的，所以很難利用固定的 stage partition 來達到接近完美的 load balancing，因此我們才在每兩個 pipeline stages 之間加入 FIFO circular buffer，來吸收因為 MBs 擁有不同解碼複雜度所造成的負擔，但這樣還是無法接近完美，因此我們在提出了兩種 Dynamic pipeline partitioning 的方法，所以接下來章節會就(1)Static pipeline partitioning、(2)Dynamic pipeline partitioning using Macroblock mode 和(3)Dynamic pipeline partitioning using monitoring buffer 三種實作來介紹。

1) Static pipeline partitioning

如 Figure 34.所示，每個核心執行固定的 tasks，Core 1 執行 ED、MVR 或 IMR[25]，會這樣分配是因為 MVR 與 ED 會共享資料，特別是目前 MB 與鄰近 MBs 的 MV 和 refIdx，MV 與 refIdx 儲存在 DDR2，因此當這兩個 task 分配在不同的核心時，這兩個核心會在 DDR2 讀取同樣的資料，故放在同核心就只要在 cache 中讀取，可以減少 system bus 的使用。Core 2 則是執行 IQIT、MP 或 IP、VR，這部分是三個核心中負載最重的一顆(因為受限於核心數)，而 Core 3 則是單純執行 ILF。

我們知道 MB 的解碼時間與 coding modes 有相當大的關係，舉例來說，當目前解碼的單位為 Skipped Macroblock 時，則在 ED 時會消耗很少的時間，而相對的，當解碼單位為 Intra Macroblock 時，ED 需要花費較多的時間解碼，其原因為 Intra Macroblock 通常擁有較多的 Residual bits。Table 3.所示為在五個 MPEG 測試視訊，為各種 Macroblock types 的分布與加速比，我們可以很輕易的注意到在 News 影像中當 skip MB 比重增加時，則效能會下降，而當 inter MB 比重上升，效能則相對上升。

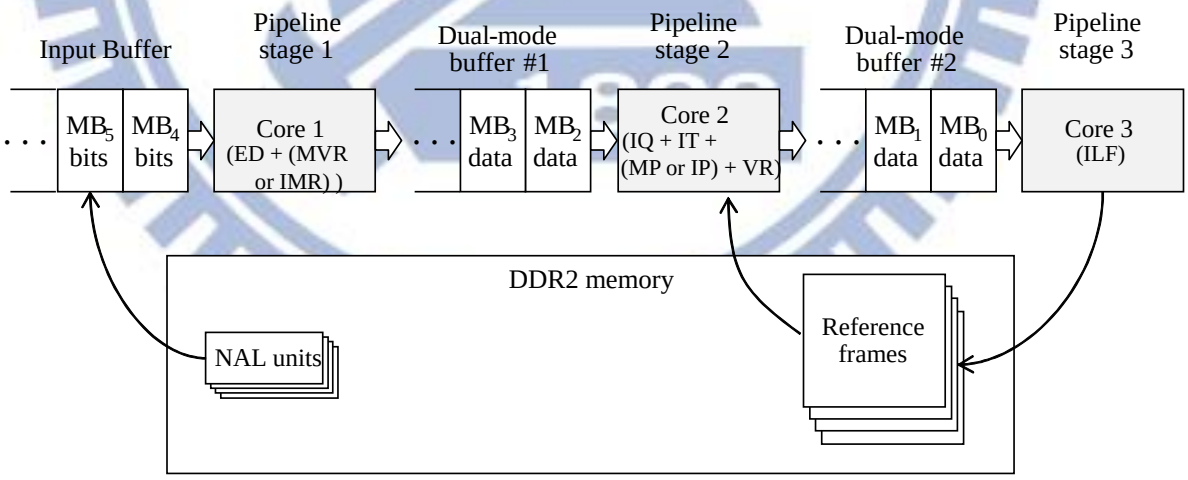


Figure 34. Static pipeline partitioning for the P-MBs and I-MBs of H.264/AVC

[25] 當 MB type 為 P-MB，則執行 MVR，反之執行 IMR

Table 3. Macroblock mode distributions and performance (use static pipeline partitioning)

Sequence	Bitrate	Intra MB (%)	Inter MB (%)	Skip MB (%)	Speedup ratio
Crew	512 k	14.4	62.8	22.8	2.54
	1.5 m	15.4	76.5	8.1	2.63
Foreman	512 k	4.1	66.2	29.7	2.56
	1.5 m	4.2	80.4	15.4	2.74
Mobile	512 k	0.5	73.1	26.4	2.47
	1.5 m	0.5	86.4	13.1	2.72
News	512 k	1.4	29.2	69.4	2.31
	1.5 m	1.8	46.4	51.8	2.74
Stefan	512 k	3.0	59.4	37.6	2.29
	1.5 m	3.8	72.9	23.3	2.54

The percentages of the macroblock modes are computed for the whole sequence of 300 frames.

Skip MBs 影響效能的原因是很簡單的，當在視訊中一連串的 skip MBs 會造成 ED 解碼時花費很少的時間，而當 FIFO circular buffer #1 深度不足時，在短時間內，第一顆核心就會因為 buffer 滿載而閒置，理想上，FIFO circular buffer 其大小是要越大越好，但現實中會受限於 local scratchpad memory 的容量限制，而為了解決這個問題，我們提出了 Dynamic pipeline partitioning。

2) Dynamic pipeline partitioning using Macroblock mode

Dynamic partition 是在 Execution time 來決定目前的 partition 方式，能依照目前解碼情況來改變，這樣可以避免和 Static partition 般，因為一連串的 Skip MB 造成第一顆核心與第二顆核心的負載量不均，而我們用來判斷動態切割的依據是利用 macroblock type，當 macroblock type 為 skip MB 時，使用 Figure 35(a)方法，圖中 SNIP 為執行 SaveNeighborForIntraPred，其功能是儲存目前 MB 部分 pixels，作為 Intra prediction 的參考資料，而 SNIP 必須固定在同一顆核心處理，否則需要把其 SNIP 資料儲存在 DDR2，會影響解碼效能。

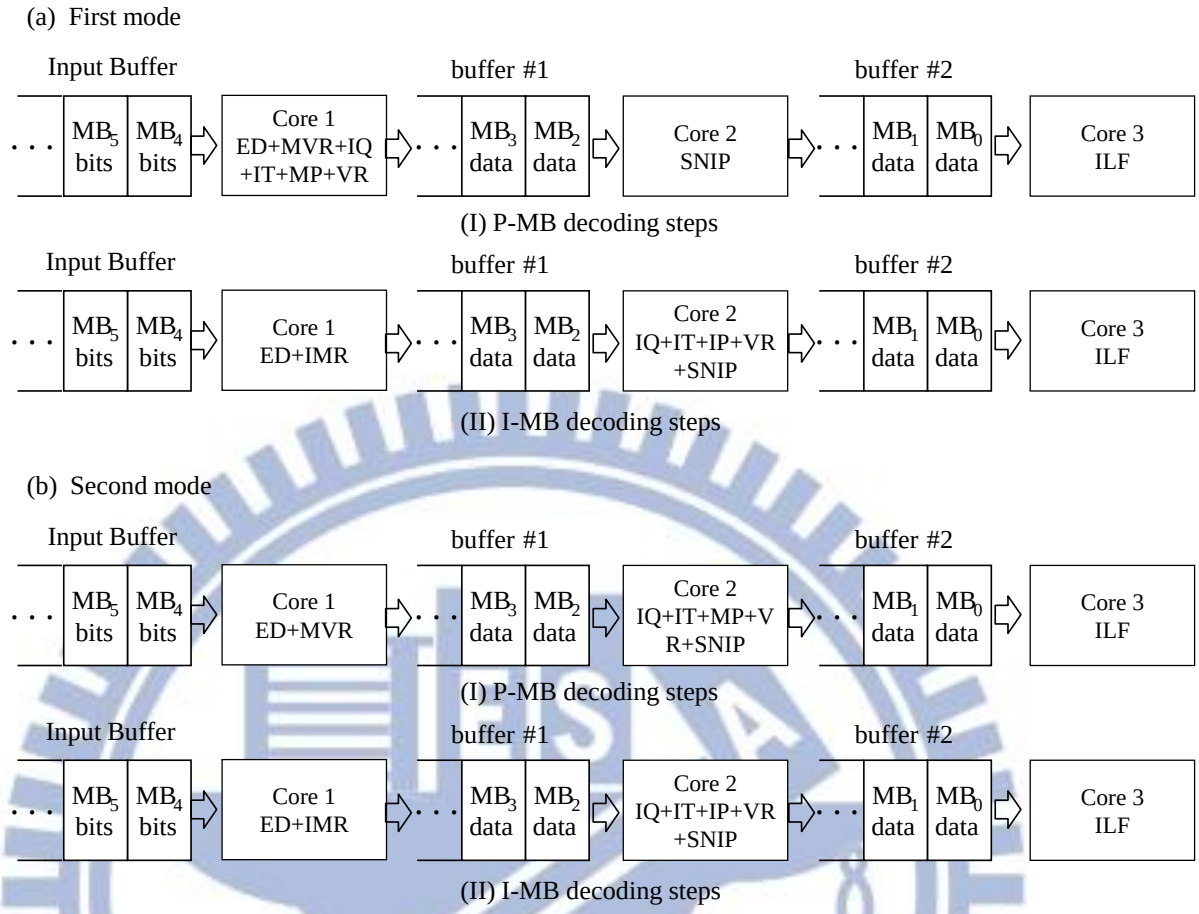


Figure 35. (a) ~ (b) Two kinds of different partition for dynamic pipeline partitioning using MB mode

為了讓 MB types 影響平行化效能能更容易被理解，如 Figure 36、Figure 37、Figure 38 和 Figure 39 所示，為兩種 software pipeline decoder(static 和 dynamic pipeline partitioning schemes)的加速比，而測試視訊為 Crew 1.5mbps 和 Stefan 512kbps，可以很明顯的發現 bits 數下降的 frame 與 speedup ratio 是有對應的，bits per sample 與 skip MB 有很大的關係，更進一步的觀察 Figure 39，就能發現當 Skip MB 數上升(尖峰)，會對應到較低的 speedup ratio，而在 Crew1.5mbps 的部分，能觀察出 speedup ratio 的尖峰是對應在 bits per sample 的尖峰也是對應在 number of the intra 4x4 的尖峰。而 Figure 37 所示，Stefan 512kbps 中 skip MB 比重較重，更精確的說法是，出現連續的 skip MB 比重較高，因此能看到當我們採用 dynamic partition 時，效能有顯著的改進，而在 skip MB 比重較低的 Crew 1.5mbps 影響就有限。

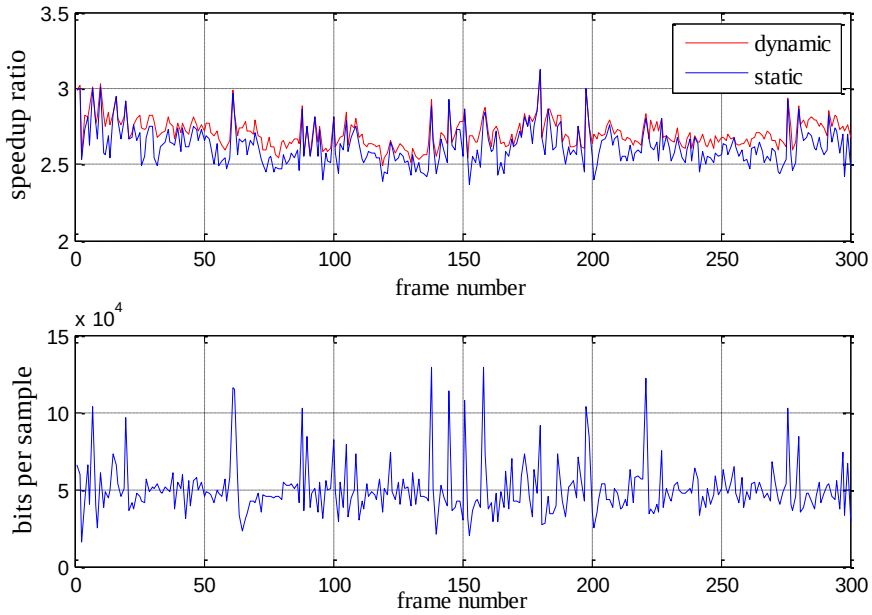


Figure 36.Top: the speedup ratio of the dynamic (use MB type) [26]and static pipeline partition decoder for Crew@1.5mbps. Bottom:bits per frame of the video sequence

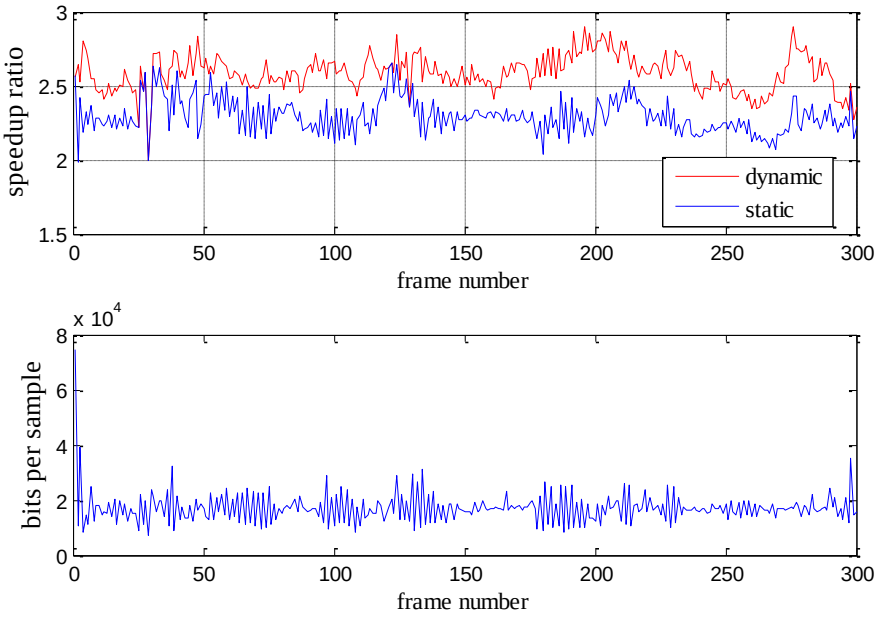


Figure 37. Top: the speedup ratios of the dynamic (use MB type) and static pipeline partition decoder for Stefan@512kbps.Bottom:bits per frame of the video sequence

[26] Figure 36 會發現 speedup ratio 會超過 3，其原因是 single core 的版本資料一律放置在 DDR2。

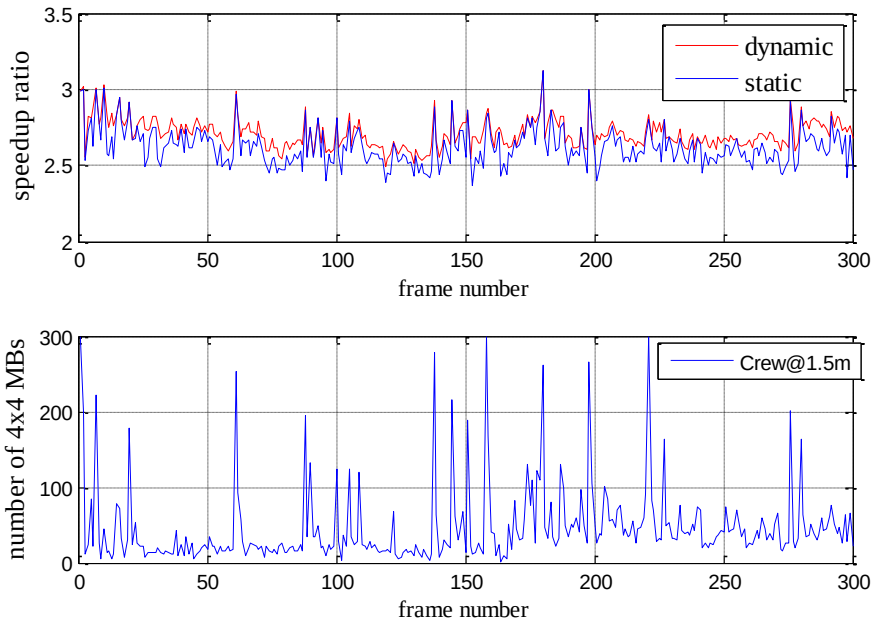


Figure 38. Top: the speedup ratio of the dynamic and static pipeline partition decoder for Crew@1.5mbps.

Bottom: the number of the I4x4 MBs

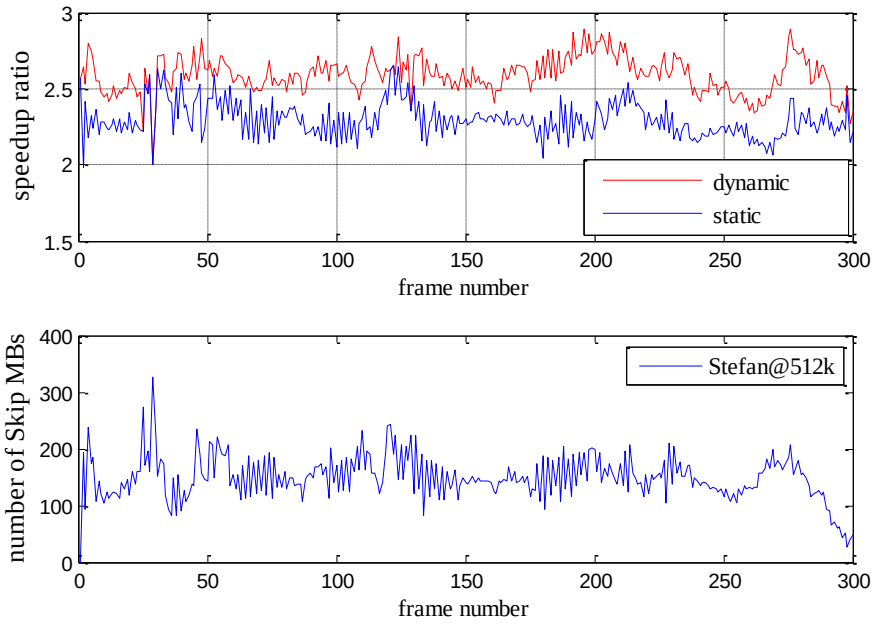


Figure 39. Top: the speedup ratio of the dynamic and static pipeline partition decoder for Stefan@512kbps.

Bottom: the number of the Skip MBs

在我們的 static pipeline partition decoder 設計中，總共有三個 statges，第一個處理 Entropy decoder 和 Intra mode 或 Motion Vector Reconstruction，第二個處理 IQ、IT、MB prediction 和 Reconstruction，而第三個則是執行 In-loop filter，而在 decoder 中有兩個 FIFO circular buffer，如 Figure 34 所示，每個 buffer node 具有相同的資料結構，而 speedup ratio 是直接被 buffer 是否 underflow 或 overflow 影響，假如在解碼的過程中 buffer 不存在 underflow 或 overflow，則平均 speedup ratio 將會接近於三倍，但現實中每個 MB 為 variable complexity，因此可能會造成 underflow 或 overflow，而效能因此被受影響。Figure 40 為 Stefan@512kbps frame #1 在 runtime 時 buffer depth 增長情況，frame #1 為此 sampley 在 static partition pipeline decoder 中最低 speedup ratio(1.99)的 frame，可以發現第一個 FIFO 幾乎都處於 overflow 狀態，第二個則是 underflow，而我們在使用 dynamic partition，就能發現第一個 buffer overflow 的情況舒緩了許多，效能也提升到 2.65 倍。Figure 41 為 Crew@1.5mbps frame #152，是該 sample 的 speedup ratio 最低的 frame，一樣存在的同樣問題，雖然 dynamic partition(skip mode)有改善，但 buffer ovderflow 情況依然明顯，因此換個角度，從 FIFO buffer 這邊出

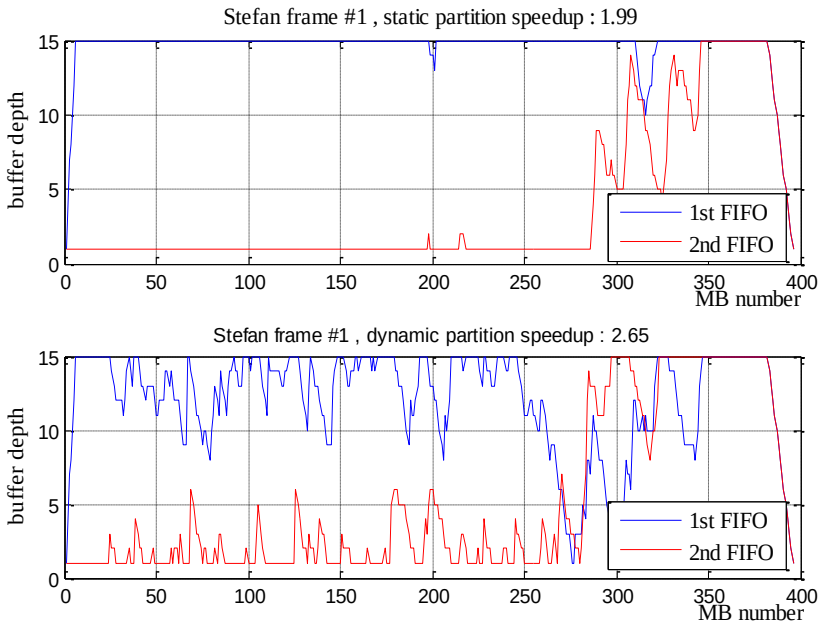


Figure 40. The runtime buffer growth at frame #1 of the Stefan@512kbps for the static and dynamic (skip)

partition pipeline decoder

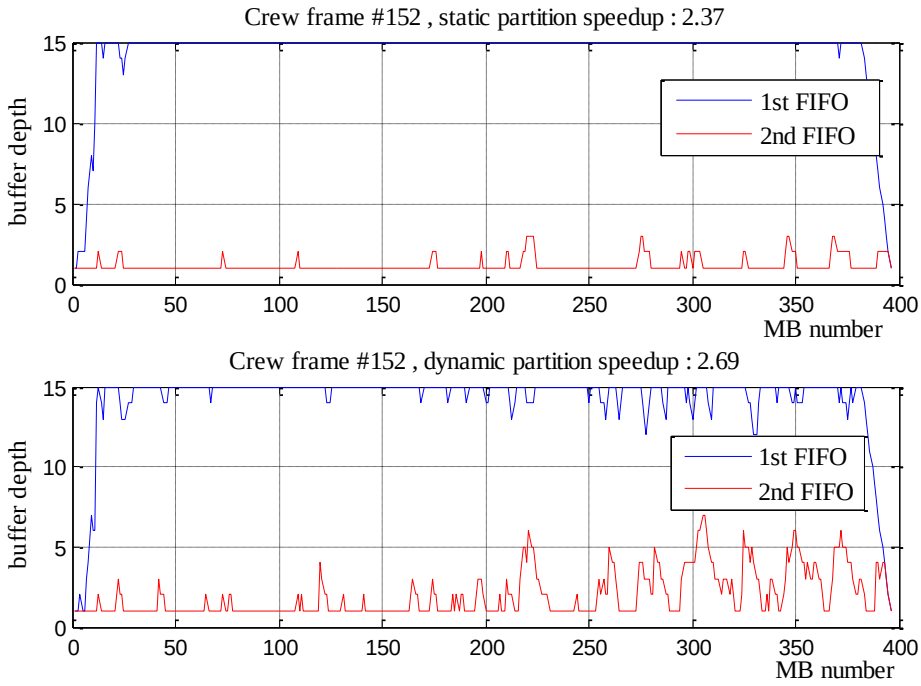


Figure 41. The runtime buffer growth at frame #152 of the Crew@1.5mbps for the static and dynamic (skip) partition pipeline decoder

發，因此我們提出了第二個 dynamic partition 的機制。

3) Dynamic pipeline partitioning using monitoring circular buffer

上一小節提到了，MB type 與效能有直接的關係，因此我們使用最容易影響效能的 skip MB 作為 dynamic partition 的依據，但不可否認的，其他如 Intra 4x4、Inter 8x8 或一些 MB 具有較高的 residual bits，這都會直接我間接的影響效能，但把這些因素全部都考慮，將會造成 dynamic partition 的規則過於複雜，也需要做大量的測試與會受限於輸入的視訊影像而有不同的結果，因此我們直接使用兩個 FIFO circular buffer 的情況作為改變 partition 的機制，其規則與切割模式如 Figure 42 與 Figure 43 所示，當 buffer#1 深度超過 threshlod α 時，就會使用 mode 1，即把第二顆處理 P-MB 部分拉到第一顆處理器執行，而當 buffer#2 深度超過 threshlod β 且 buffer #1 沒有過載的情況時(避免第二顆處理器本身負載也過重)，使用 mode 2，即把 In-loop filter 拉到第

二顆處理器執行，用來減輕 buffer#2 之負擔，而沒有發生以上兩種情況則使用預設的 mod 0，α 與 β 是依照模組比例與 fine-tune 後決定的，為總深度*0.8 與總深度*0.9。

```
if ( buffer_1 > threshold α)          // α is TolalBufferDepth*0.8
        dynamicMode = 1;
else if ( buffer _2 > threshold β && buffer_1 < α ) // β is TolalBufferDepth*0.9
        dynamicMode = 2;
else
        dynamicMode = 0;
```

Figure 42.The rule for dynamic partition using monitoring buffer

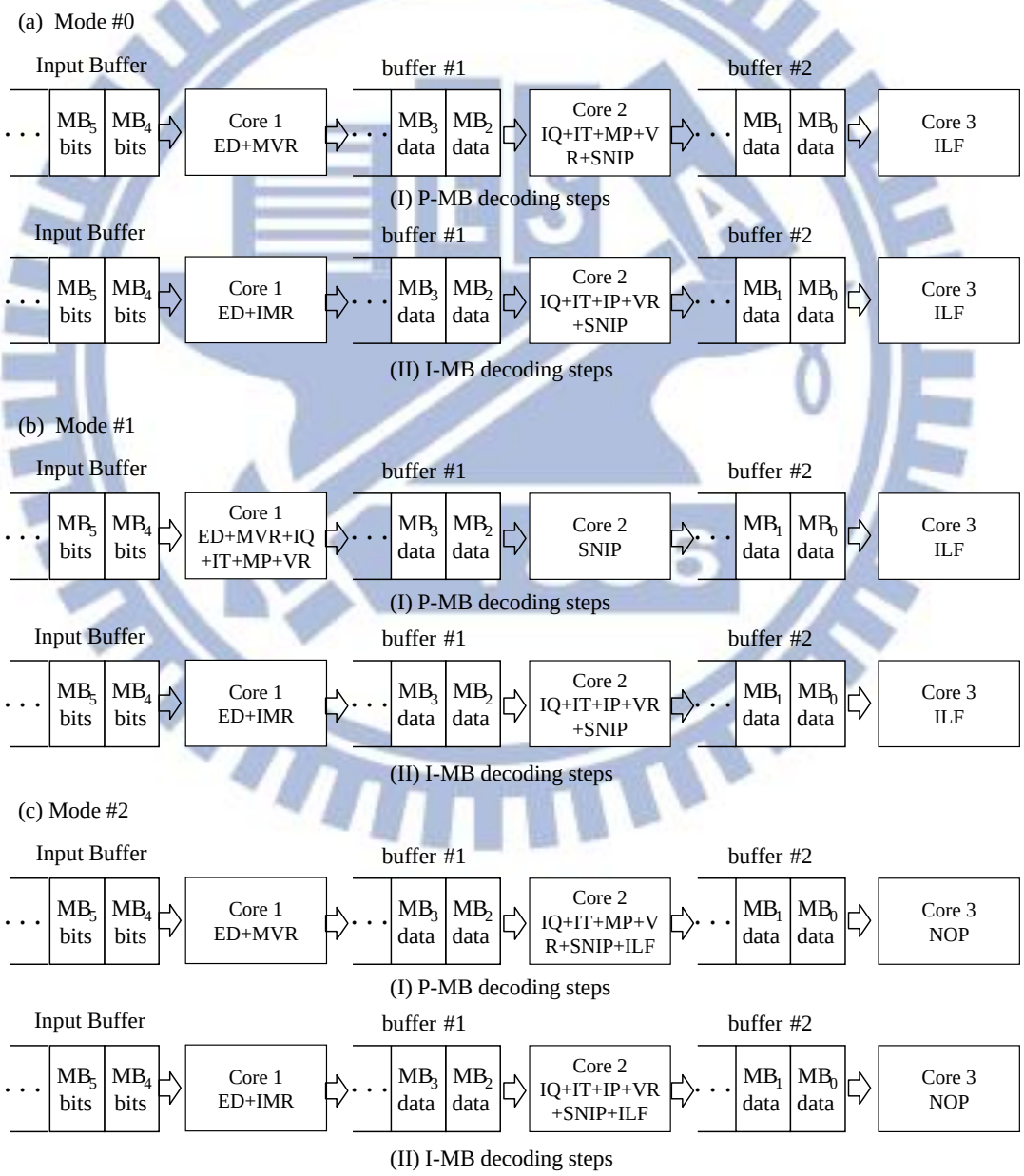


Figure 43. The decoding steps for dynamic partition pipeline decoder using monitoring buffer

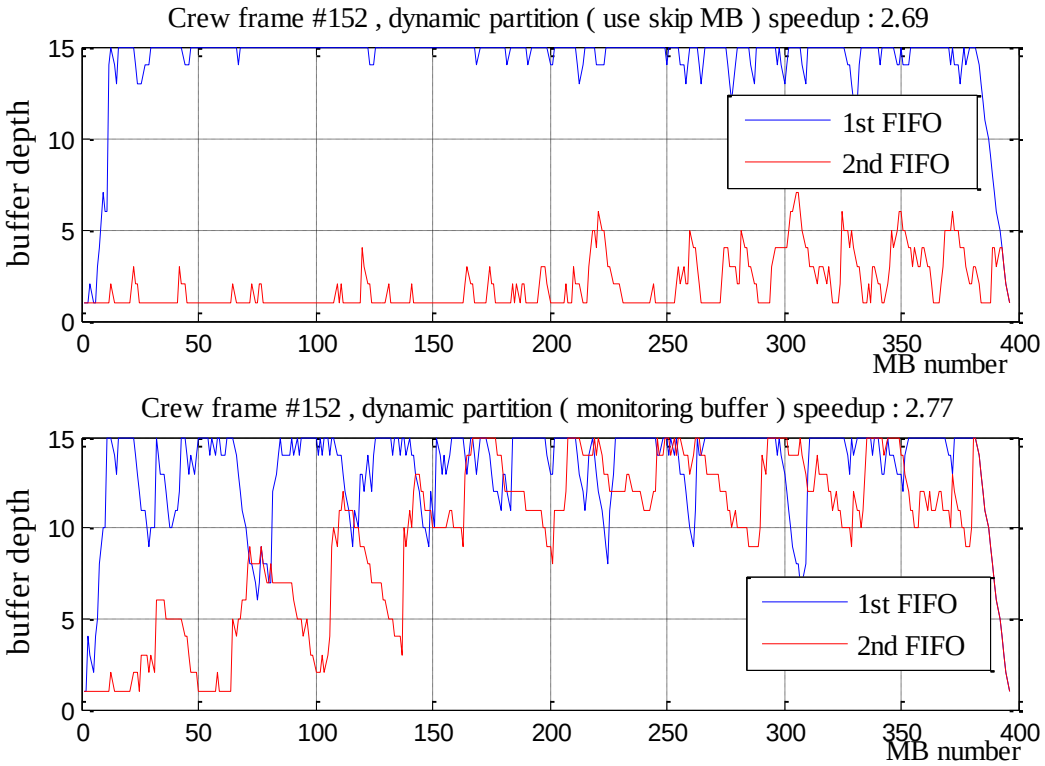


Figure 44. The runtime buffer growth at frame #152 of the Crew@1.5mbps for the dynamic#1 (use skip) and dynamic#2 (monitor) partition pipeline decoder

Figure 44 所示，利用 monitoring buffer 來控制 partition 方式，效能從 2.69 提升到 2.77 倍，圖中也能觀察出 1st FIFO 的 overflow 情況明顯變少， 2nd FIFO 的使用率也有提高，因此能觀察的出來 monitoring buffer 可以減少核心閒置的情況。

4.3 Memory usage of parallel H.264/AVC video decoding

H.264/AVC 相較於以往的視訊壓縮標準(如:MPEG-2、H.263)擁有較低的 bit rates，其原因是利用 variable block size motion estimation、sub-pixel motion estimation、multiple reference frames、In-loop filter 等等的新技術，其中 variable block size motion estimation 相對於 fixed 16x16 blocksize motion estimation 節省了約 4%到 20%的 bit rate，而使用 quarter-pixel motion vector 相較於 half-pixel motion vector 提升了 30%編碼效率，但是這兩項技術也增加了解碼器的計算與 memory access 的複雜度，sub-pixel interpolation 大約占了解碼器約 25%的運算量[18]且 variable block size 和 sub-pixel motion compensation 會大幅提升 memory access 頻率，而在我們討論的手持式嵌入式裝置上，將會是很重要的議題，Ronggand Wang 在[16]有提出三種機制來減少 motion compensation 在 off-chip memory access，Chen 在[17]提出利用 on-chip SRAM 或 register file 來 cache 高重複率的 reference data 來減少 DRAM acceess。我們測試了 crew@1.5mbps 和 news@512mbps，兩組視訊所有 pixel 重複被參考的次數，如 Figure 45 和 Figure 48 所示，從 crew 可觀察出，約有七成的 pixel 將會重複被參考，甚至有四成的 pixel 是會被重複參考超過兩次，但把 sub-pixel motion compensation 關掉，則可以發現約有 35%的 pixels 不會被重複參考，85%的 pixels 只會被參考一次內，這也應證了先前提到的，sub-pixel interpolation 會明顯提高 off-chip memory access 次數，另外從 Figure 45、Figure 46、Table 4 也可以觀察出，當 bitrate 越高，sub-pixel 比重越重，repeated reference pixeld 次數越高。而 news 也可以觀察出同樣結論，但因為 news 相對於 crew 較為靜態，因此重複參考比率相對較低，我們也可以從這邊了解，cache performance 對於 motion compensation 模組有一定的影響。

在 H.264/AVC baseline video decoder 中，有四個主要模組需要 off-chip memory access(DRAM)，如 Table 5 所示，我們可以發現 motion compensation 為解碼過程中

主要 access off-chip memory 的模組，約占 75%(為最差情況)。

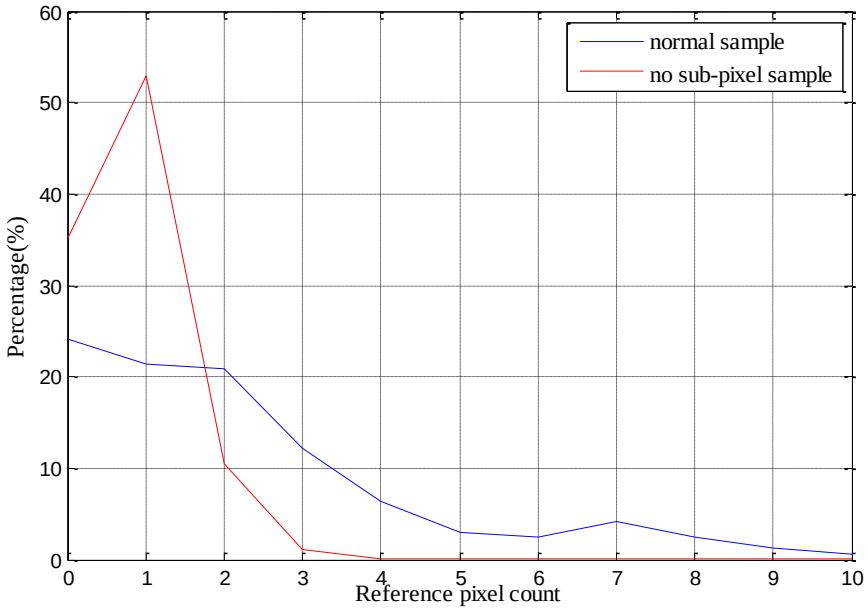


Figure 45. The count of repeated reference pixel for crew@1.5mbps

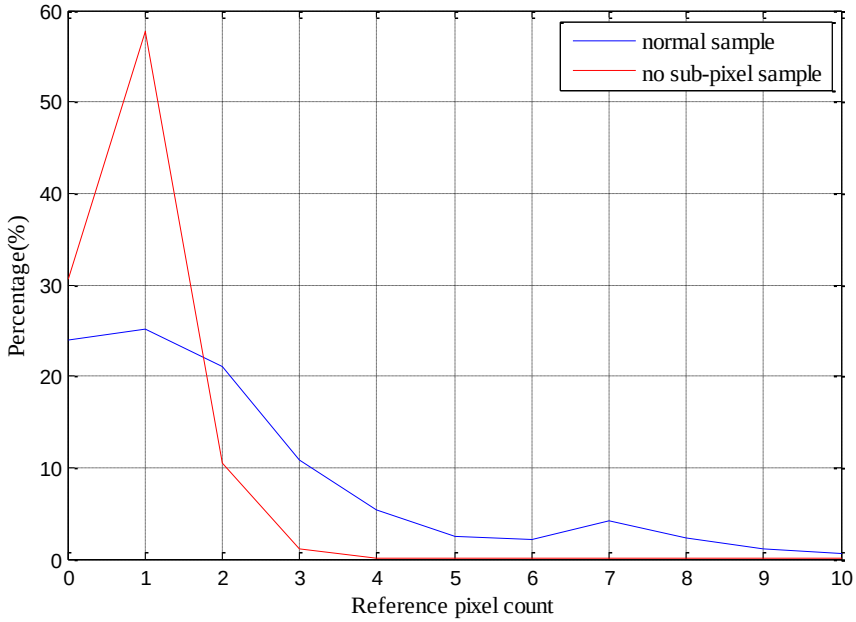


Figure 46. The count of repeated reference pixel for crew@512kbps

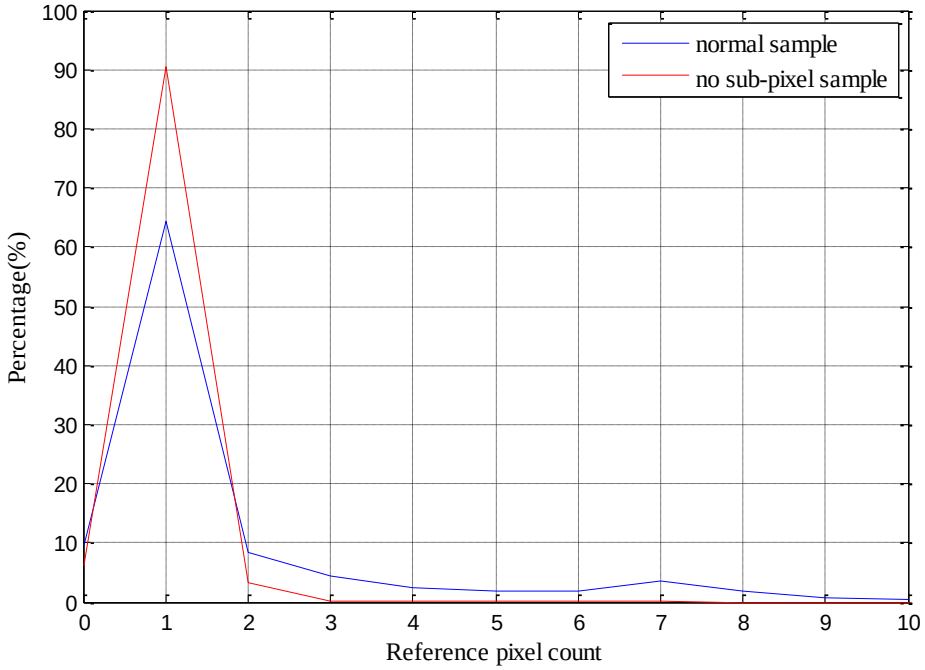


Figure 47. The count of repeated reference pixel for new@15mbps

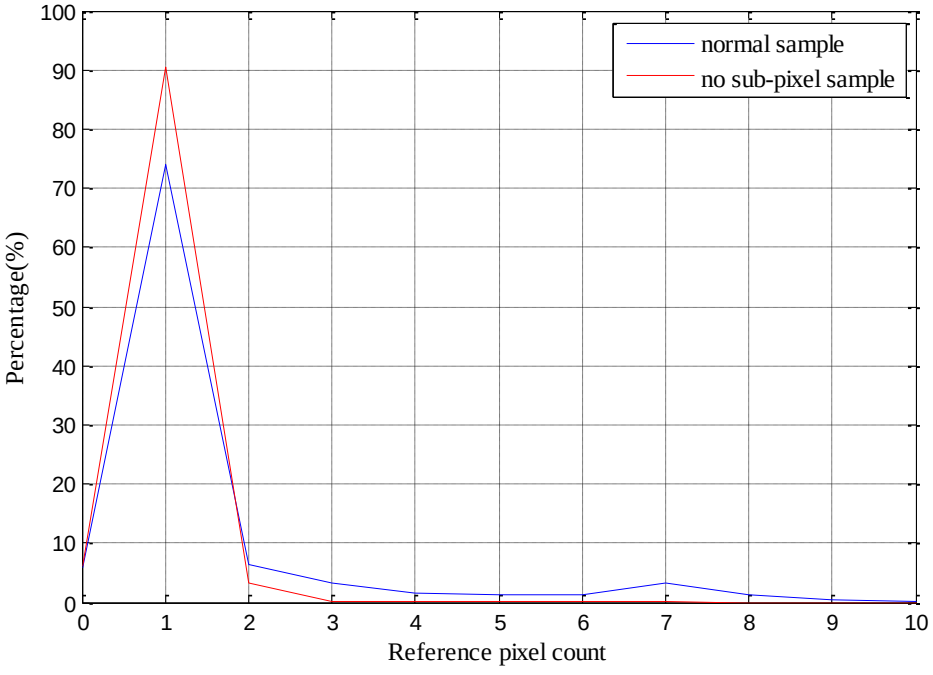


Figure 48. The count of repeated reference pixel for news@512kbps

Table 4. The count of Integer MV and Non-Integer MV

Sequence	Integer MV (count, %)	Non-Integer MV (count, %)
Crew 15mbps	(22011, 8.17%)	(247421, 91.83%)
Crew 512kbps	(20066, 11.85%)	(149316, 88.15%)
News 15mbps	(132181, 49.73%)	(133634, 50.27%)
News 512kbps	(108556, 58.72%)	(76311, 41.28 %)

Table 5. Memory access ratio of each H.264/AVC decoder module (W: frame width, H: frame height)

Module name	Max memory access bytes	Ratio(≒)
Ref. picture store	W*H+2*(W/2)*(H/2)	10%
Motion compensation	(W/16)*(H/16)*16*(9*9+2*3*3)*2	75%
In-loop filter	(W/16)*(H/16-1)*(16*4+4*4*2)*2	5%
Display feeder	W*H+2*(W/2)*(H/2)	10%

在 TLP video decoder 與 DLP video decoder 的設計中，motion compensation 依然是最頻繁讀取 off-chip memory 的模組，但 TLP 在 Static pipeline partitioning 中 motion compensation 固定在第二顆處理器執行，使得能從 spatial locality 獲得好處，即 reference data 能從 cache 中獲得，而不需要透過 system bus 來 access off-chip memory，dynamic partition 相較於 DLP 仍然是有較少的 access off-chip memory 次數。

至於 DLP 與 TLP 所使用的 off-chip memory 空間，主要差異是在於 DLP 在 intra prediction data，需要(W+2*(W/2))* (H/16) bytes 來儲存資料(W: frame width, H: frame height)，而 TLP 則只需要 W+2*(W/2) bytes，且可以放置在 local memory 中(因為不需要多顆核心共享)，因此 TLP video decoder 在設計上可以更節省記憶體的使用與減少 system bus 的使用頻率。

五、實驗結果

在這個章節一開始我們會介紹所使用的硬體平台，包括 Xilinx XUPV5-LX110T development board 以及 Nexus 7 2013。在 5.1 節會介紹我們在 Xilinx vertex-5 FPGA 上所使用的系統架構，在 5.3 節則介紹在此架構下的 software kernel 和 synchronization schemes，而在 5.2 節則會介紹我們採用的另外一個測試平台 NEXUS 7 2013 版本。之後，我們會使用我們在 Xilinx 平台上所實作出的 DLP video decoder、TLP video decoder 與未平行化的 H.264 解碼器版本進行效能比較，我們使用五段測試影片(CIF)與不同的 bitreate 進行效能比較，且在測試較高解析度影片(SD)對於平行化效能的影響。

5.1 Xilinx XUPV5-LX110T development board

Xilinx XUPV5-LX110T[2]是 FPGA 的開發板，並提供以下特色

- Xilinx Virtex-5 XC5VLX110T FPGA
- Two Xilinx XCF32P Platform Flash PROMs (32 Mbyte each) for storing large device configurations
- Xilinx SystemACE Compact Flash configuration controller
- 64-bit wide 256Mbyte DDR2 small outline DIMM (SODIMM) module compatible with EDK supported IP and software drivers
- On-board 32-bit ZBT synchronous SRAM and Intel P30 StrataFlash
- USB host and peripheral controllers
- Programmable system clock generator
- Stereo AC97 codec with line in, line out, headphone, microphone, and SPDIF digital audio jacks
- RS-232 port, 16x2 character LCD, and many other I/O devices and ports

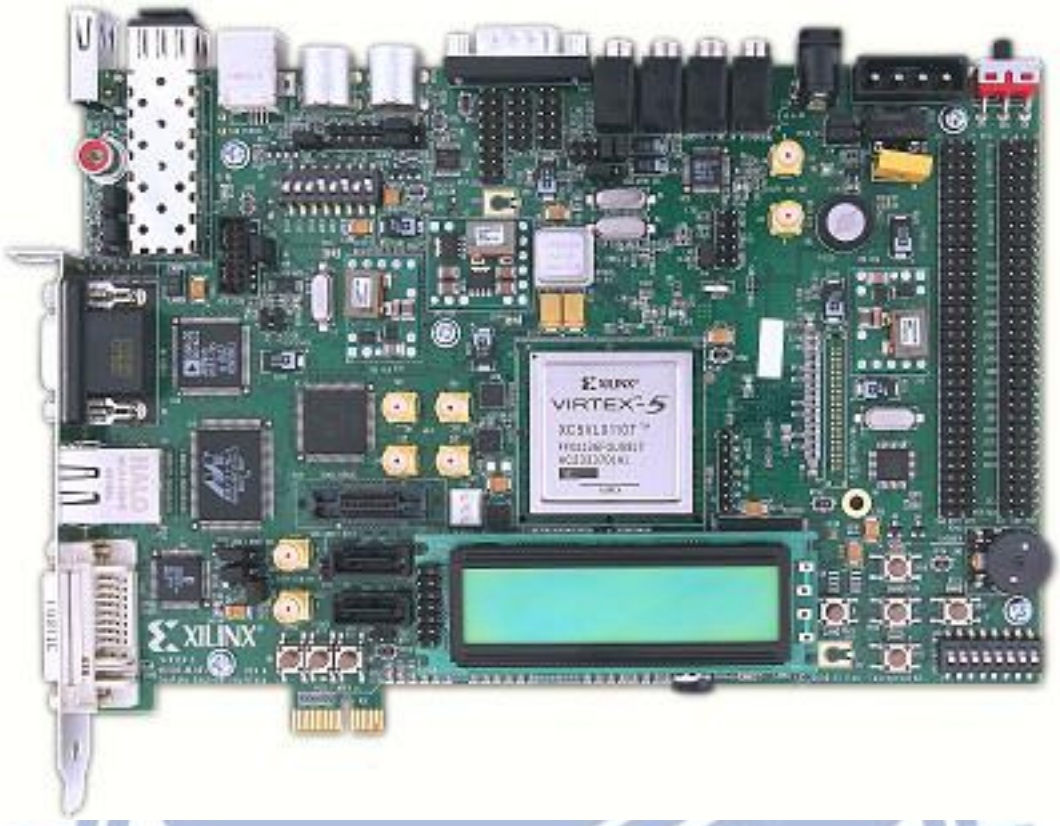


Figure 49.Xilinx XUPV5-LX110T Evaluation Platform

開發系統包括以下:

- XUPV5-LX110T board
- 1GB Compact Flash card
- 256 MB SODIMM module
- SATA cable
- XUP USB-JTAG Programming Cable
- DVI to VGA adapter
- 6A power supply

Xilinx XUPV5-LX110T Virtex-5 FPGA 提供足夠的邏輯單元給整個 Soc 設計。而本論文使用 Compact Flash card 作為程式中輸入與輸出資料的儲存裝置。

5.1.2 Application Processor Soc 架構

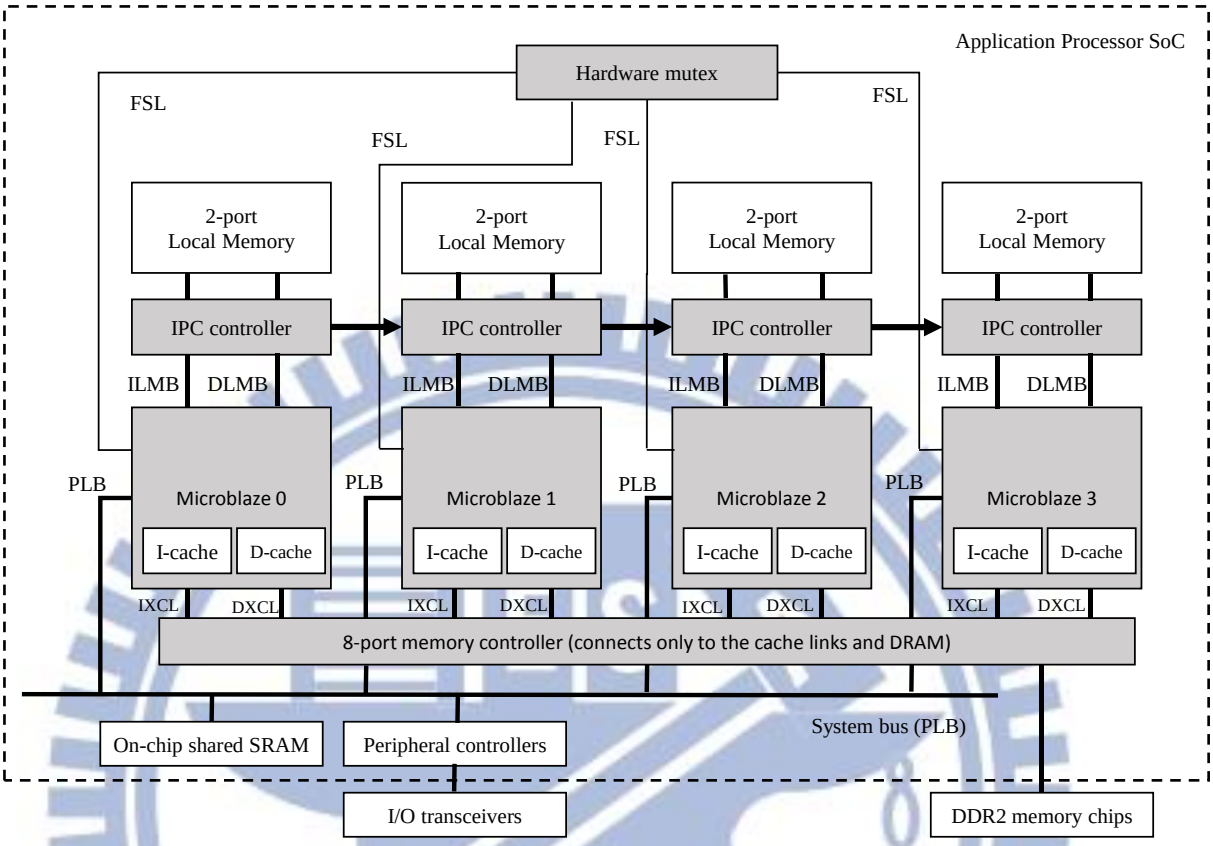


Figure 50. Proposed architecture on a Xilinx Virtex-5 FPGA

本論文所採用的 SoC 架構是發表在[14]的多核心平台。這個平台(Figure 49)除了能支援傳統的 SMP(Symmetric multiprocessor) multi-threading 系統之外，也能使得 Software-pipeline 的執行更有效率，且 pipeline program 不會占用額外的 system bus 頻寬，而撰寫上也不會造成程式設計師有多餘的負擔，並且能輕易的移植到現存的 SMP 作業系統上。

這個擁有四顆RISC核心的架構是利用 Xilinx Virtex-5 FPGA 實驗板進行系統實作驗證，Xilinx Microblaze processor IP[27]是用來作為 RISC 核心，且每顆核心有三組 instruction/data ports，第一組利用 Local Memory Bus(LMB)與一塊 on-chip memory

[27] IP 是一種事先定義、曾經驗證、可以重複使用的功能組塊，IP 亦可解釋為功能元件組塊。

bank 做連結，第二組則是與 system bus(PLB)做連結，第三組透過 Xilinx Cache Link(XLC)與 DRAM contorller 連結。而由於 Microblaze IP 的限制[28]，使得在這 SMP 的架構下 Microblaze processors 在跨核心之間沒有支援 coherent data cache，因此程式設計師必須自行呼叫 cache validation function 來確保共享的資料沒有 cache coherence 問題，此行為會造成 processor core 的效能下降，因此用來同步機制使用的標籤(flag)並不會放置在會被 cache 所攔截的共享記憶體中(本論文中指的都是 DDR2 memory chips)，而是放置在**錯誤! 找不到參照來源。**.中所示的 On-chip shared SRAM[29]，用來降低沒有 coherent data cache 的影響。

另外 Hardware mutex 部分，Xilinx 本身有提供 mutex IP，但是該 IP 是透過 PLB bus 來進行存取，而 mutex lock 的機制為 spin lock 的方式實作，所以當多顆處理器都要對 mutex 進行存取時，會造成系統因為 shared PLB 負擔過高而整體系統效能下降，因此自行設計硬體實作的 mutex 裝置來處理 thread synchronization，mutex 裝置包含 64 組 32-bit 專門的暫存器，每一個是用來當作 mutex 變數提供給程式中所有的 threads 作為識別。而當 processor 需要 mutex lock/unlock 時是透過 xilinx 提供的 FSL(Fast Simplex Links)[30] bus，故只需要幾個 clock cycles 即可完成動作，且當 processor core 被 mutex lock 給鎖住(spin-lock loop)時，只會占用自己的 FSL 頻寬，

[28] Xilinx 沒有提供 Microblaze 完整的 RTL code

[29] On-chip shared SRAM 為 uncached memory。

[30] FSL 為 32-bits 寬在 Microblaze 上的介面，FSL channels 為單向點對點的數據流介面。

而以上兩項設計是為了減輕系統架構中 non-coherent data caches[31]對程式效能的影響和降低 shared PLB 負擔，但不可否認的是依然會造成效能上的下降。

另外在系統架構設計上，為了減輕可能數個 threads 密集對共享記憶體(DDR2)作讀寫動作所產生的負擔，因此在每個 processor core 掛上一塊屬於自己的 scratchpad memory bank[32]，而此記憶體裝置為 on-chip memory，故讀寫速度只需幾個 cycles 就可完成，因此適合放置暫存的資料。

Table 6. FPGA LOGIC USAGE OF THE PROPOSED ARCHITECTURE

IP (single instance)	LUT6	Flip-Flops	RAM
Microblaze[33]	2164	2391	8+16KB
IPC controller	567	297	
Local scratchpad memory			64KB
Hardware mutex	1103	568	
8-port memory controller	6580	7836	2KB

The system clock rate is 83.3MHz.

Table 6.所示的是在此本論文採用的系統架構中，主要的模組所使用的 FPGA logic 情況，而前述所提到的 Hardware mutex 裝置以及 on-chip shared SRAM 兩塊裝置並非是這個系統架構的 pipeline datapath 的必要元件，未來架構中有自己的 coherent data cache 時，將可以拿掉這兩個元件。

[31] 論文中採用的是 write through 機制。

[32] Pipeline 的實作上，scratchpad memory bank size 為 64KB

[33] 為 Xilinx 所提供的 32bit RISC soft processor，採用 Harvard 架構

5.1.3 The IPC controller

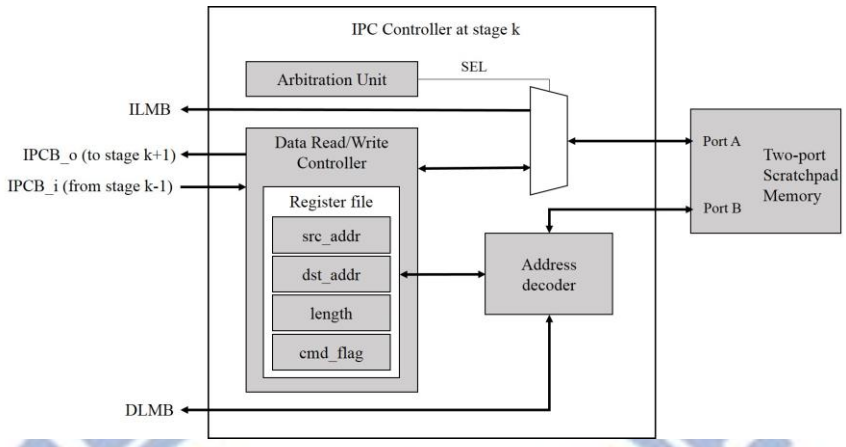


Figure 51. Stage k 之 IPC controller 架構圖

在[14]的多核心硬體平台上，包含了一組特殊的 IPC controller(每個核心須配置一個 controller)，是用來幫助 pipeline-based 程式能更有效率的運作，利用 IPC datapath 減輕 system bus 頻寬(本論文指的是 PLB bus)，且提供 API 能使程式設計師在設計 pipeline program 上更便利，API 部分下一小節會詳細介紹。而在 pipeline 每個 stage 裡包含一個 processor core、一個 IPC controller 和一個 local scratchpad memory，IPC controller 的架構為 Figure 51.所示，IPC controller 負責處理兩顆處理器之間 scratchpad memory 的隨機存取(random memory access)，在兩個連續 pipepline statges 使用 burst data transfer。而 ILMB 擁有 memory port 的最高優先權，但如果 ILMB 不使用 memory port 時，且兩個 IPCB's 進行爭搶時，將會輪流使用共享的 memory port。而比較值得注意的是當資料在 pipeline stage buffers 傳輸時，處理器不需要任何關注，這意味著當資料在 memory port 進行傳輸時，處理器能進行其他工作的運算處理。

IPC controllers 有四組 32-bit controller registers，這些寄存器是用來觸發指定範圍的資料從目前 scratchpad 到下一個 scratchpad，而這些 controller register 被對應在 local memory 位址的 0xFFF0 到 0xFFFC。

5.1.4 Software kernel and Synchronization Schemes

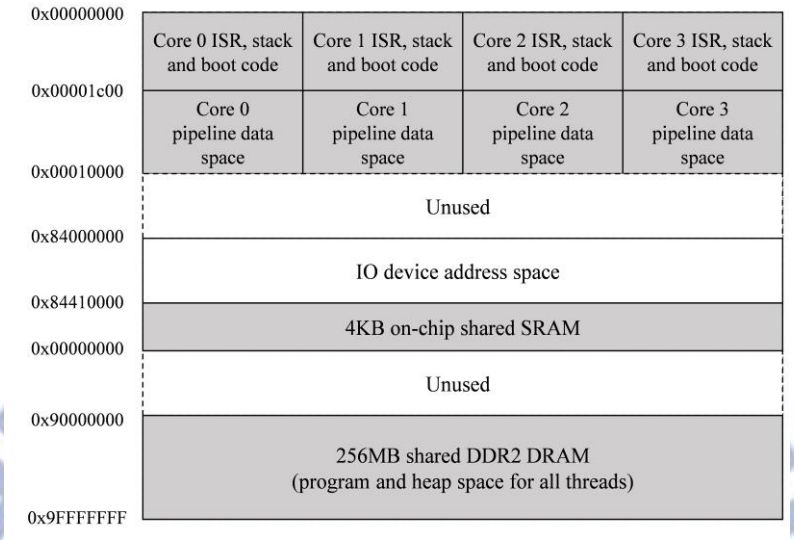


Figure 52. The memory map of the system on an FPGA development boad

在[14]提出的架構中，包含了一個 non-preemptive multi-threading operating system kernel 讓四顆應用處理器能夠在 SMP 架構中合作，而因為在每顆處理器上的 system kernel 不支援 preemptive multi-threading，所以每顆處理器只能執行單一線程。如圖 Figure 52.所示，程式碼以及 heap space 放置在 DDR2 memory 且被每個處理器對稱共享，而 stacks 以及當系統被喚醒時每個處理器所執行的 boot code 則放置在各自處理器的 local memory。

在這個系統架構中，如果要對 threads 進行同步化，主要有兩種方法，第一種為利用 hardware mutex device 所擁有的 64 組 mutexs，每一組 mutex 可以在幾個 clock cycles 對單一線程做 lock 或 unlock 的動作[34]，假如有 thread 想 lock 已經被其他 thread 鎖住的 mutex 時，則會進入一個 spin-lock loop，而在被 mutex 擁有的 thread 解鎖之前該 thread 並不會消耗任何 system bus 頻寬或是 data cache 頻寬。第二種同步化機制則是使用 on-chip shared memory 放置 flags 或共享資料，而 system bus 仲裁方式採用 round-robin 的機制，所以不會有任何 threads 在要取得共享資料或讀取標籤時

[34] 使用 int mutex_lock(int mutex_num)、以及 int mutex_unlock(int mutex_num) API

產生 starving 的情況。

另外此架構每個核心都擁有屬於自己的 local scratchpad memory，為了讓此塊儲存裝置能有效率的被使用，且不會增加程式設計師多餘的負擔，故設計了 lm_alloc(size_t xWantedSize)、vPortFree(void *pv)功能，參數 xWantedSize 是用來指定所需要的 local memory size，利用此 api 來配置或釋放所需要的 local memory space，而實作方式是使用 link list 資料結構，而為了設計 thin software kernel 所以在 local memory management 上不處理外部碎裂(external fragmentation)的問題。

而在為了減輕 pipeline program 設計上的負擔，設計了能簡單利用 IPC controller 的介面，提供兩個簡單操作的功能 void lm_transfer(void* dst_addr, void* src_addr, int len)以及 int transfer_status(void)，而參數 dst_addr，src_addr，和 len 是用來對應到指定的寄存器來控制目標位置，來源位置，以及傳輸資料的大小，而當呼叫 lm_transfer 時，會寫入非零的值在 cmd_flag，用來觸發 burst data transfer。所以在 pipeline stage 的實作上會像 Figure 53.所示，而圖上所示的 buf 為一個 circular FIFO buffer。

```
void stage_k_thread(void *param)
{
    extern int *buf[];     // Shared stage buffer pointers.
    buf[k] = lm_alloc(64); // Each stage allocate its buffer
                           //   from its local memory.

    while (1) // never-ending pipeline operations
    {
        wait_for_new_data(buf[k]); // Wait for stage k-1's input.
        stage_k_operation(buf[k]); // Perform operations on buf[].
        wait_for_trans_ready(buf[k+1]); // Wait for stage k+1 ready.
        lm_transfer(buf[k+1], buf[k], 64); // Output to stage k+1.
    }
}
```

Figure 53. Pseudo code that performs stage k operations in pipeline datapath

5.2 NEXUS 7 (2013 版)平台

Figure 54. NEXUS 7 (2013) platform

在本論文研究中是對 H.264/AVD decoder Data-level parallelism(DLP)與 Task-level parallelism(TLP)進行分析討論，而在 TLP 的 scalbility 是相當依賴 memory subsystem，在論文中採用的架構，其 data cache subsystem 目前尚無法做到跟現今的手機應用處理器一樣的效率，雖然對平行視訊解碼軟體而言，只要程式設計得當，並不需要一個大型的高效能 cache，不過為了證明這個說法，因此我們提供在最新的 Nexus 7 平板上所測試的 TLP video deocder 版本數據，且能驗證 TLP 在完善的 cache subsystem 上之效能。

Figure 54 所示，平板為 Qualcomm Snapdragon S4 Pro APQ8064-1AA 應用處理器，為四核心頻率為 1.5GHz，而處理器有三層的 data cache(L0:4KB，L1:16KB，L2:2MB)。而平板上執行的作業系統為 Android 4.3，搭載的 Linux kernel 版本為 3.4.0，而使用 Android NDK r9b 編譯 TLP 程式，且利用 Android Debug Bridge (ADB)[35]工具做為與平板溝通的橋梁。

[35] ADB 詳細操作請參閱 http://developer.android.com/tools/help/adb.html

5.3 測試環境

我們使用的五段測試影片是 Moving Picture Experts Group(MPEG)國際標準組織所採用的測試影片，測試影片解析度分別是 352x288 與 720x480。測試影片是利用 H.264/AVC JM reference software version 18.4[19]來壓縮，這五段影片(352x288)分別為 crew、foreman、mobile、news、stefan，其中 crew 有 720x480 的解析度，關於這五段影片的壓縮資訊如下 Table 7。

Table 7. The information contained in video sequences

Frame	300
Frame rate	30
Resolution	352x288(CIF) and 720x480(SD)
Format	H.264/Baseline profile
Number of reference frames	5
Quant. param for I Slices (0-51)	28
Quant. param for P Slices (0-51)	28
Max search range	32
Bitrate	512kbps、1.5mbps and 1.5、3、6Mbps

使用兩組測試的平台，如下

(1) Xilinx Vertex-5 FPGA XUPV5-LX110T 開發板，其規格如下:

- Microblaze IPs : 83.3MHz , Microblaze core is about 1.2DMIPS/MHz
- Each Microblaze processor has 8KB instruction cache and 16KB data cache
- The local scratchpadmemory for each core is 64KB
- The platform has 256MB of 64-bit DDR2 DRAM
- The synthesizable RTL model of the IPC controller and the hardware mutex
- A hardware timer , a hardware debug agent(Xilinx MDM IP), a RS232 UART controller
- Xilinx SystemACE IC used for dumping the decoding video frame to the FAT32 file system on a Compact Flash(CF) card (only used to verify the correctness)
- The soft-core IPs provided by the Xilinx EDA tools
- The complete hardware-software system is integrated using the Xilinx XPS 13.4 and SDK 13.4

(2) Nexux 7 2013 tablet computer，其規格如下

- A Qualcomm Snapdragon S4 Pro APQ8064-1AA application processor , running at 1.5GHz
- The processor has three levels of data caches (L0:4KB, L1:16KB, and L2:2MB)
- RAM : 2GB
- It runs Android 4.3 with the Linux kernel 3.4.0.
- The compiler used to build the wave front decoder is from the Android NDK r9b

5.4 實驗數據與探討

Table 8 為 baseline single-core decoder 在目標平台上的效能，而所有數據皆是執行三次再取平均值，由數據上可以知道在此平台上，single-core decoder 對 CIF 影片解碼時無法達到 real- time decoding(30Hz)，但我們關注的是在此平台上實做的各種平行化 H.264 解碼器其加速比。

Table 9.~Table 12.為利用我們提出的 IPC datapath 的四種 Three-core Pipeline Decoder，雖然我們有四顆 Microblaze 核心，但因為第一顆是用來從 SDRAM 中的 bitstream 抽取 NAL unit，且把 NAL unit 傳給 pipeline 解碼，並透過 RS232 裝置在 console 顯示資訊等等的動作，所以 pipeline decoder 作實際解碼動作的只有三顆 Microblaze 核心。從 Table 11 可以發現 pipeline 架構相對於 single-core decoder 可以接近三倍的加速，而為了瞭解有多少效能提升是因為我們提出的 datapath，因此提供出 DRAM-based pipeline decoder，數據 Table 11 所示，DRAM-based 的設計是仿照 Static Pipeline Partition Decoder，差異是在 shared FIFO buffer 是放置在 DRAM，而不是 local scratchpad memory，而我們觀察出其效能相對於 single-core decoder 只有提升 1.83 倍，跟 static partition 的 2.54 倍或是 dynamic partition 的 2.82 倍都有明顯的差距，這意味著，使用我們的 IPC datapath 能更有效的發揮 pipeline-based video decoder 相較於傳統沒有提供 IPC 機制的 SMP 架構。Table 13 為 Wavefront Three-Core Decoder，可以看出其效能雖高於 DRAM-based pipeline decoder，但卻低於我們提出的 pipeline decoder。

Table 8. Baseline Single-Core Video Decoder Performance

Video Sequence	512 kbps	1.5 mbps
Crew	97.57	136.55
Foreman	86.33	118.97
Mobile	85.61	116.87
News	57.15	81.27
Stefan	86.39	119.3

All sequence are in CIF resolution and have 300 frames. The numbers are decoding time in seconds.The test platform has 16KB of local scratchpad(not used here) and 64KB of data cache per processor core. Only one core is used to run the single-core video decoder.

Table 9. Proposed Static Pipeline Partition Decoder Performance

Video Sequence	512 kbps	1.5 mbps
Crew	38.46	51.84
Foreman	33.73	43.35
Mobile	34.7	42.93
News	24.76	29.65
Stefan	37.66	46.94

All numbers are decoding time in seconds. The decoder uses the proposed pipeline datapath for decoding. Static pipeline-stage partitioning is adopted. The test platform has 64KB of local scratchpad and 16KB of data cache per processor core.

Table 10. Proposed Dynamic Pipeline Partition Decoder (use MB mode) Performance

Video Sequence	512 kbps	1.5 mbps
Crew	35.63	50.31
Foreman	30.91	41.17
Mobile	31.83	41.58
News	24.58	30.05
Stefan	33.5	45.35

All numbers are decoding time in seconds. The decoder uses the proposed pipeline datapath for decoding. Dynamic pipeline-stage partitioning is adopted (use MB mode). The test platform has 64KB of local scratchpad and 16KB of data cache per processor core.

Table 11. Proposed Dynamic Pipeline Partition Decoder (monitoring buffer) Performance

Video Sequence	512 kbps	1.5 mbps
Crew	34.60	43.96
Foreman	30.51	38.32
Mobile	29.56	38.43
News	24.53	29.91
Stefan	31.31	42.05

All numbers are decoding time in seconds. The decoder uses the proposed pipeline datapath for decoding. Dynamic pipeline-stage partitioning is adopted (monitoring buffer). The test platform has 64KB of local scratchpad and 16KB of data cache per processor core.

Table 12. DRAM-BASED Three-Core Pipeline Decoder Performance

Video Sequence	512 kbps	1.5 mbps
Crew	53.19	70.92
Foreman	47.81	61.47
Mobile	49.29	61.67
News	40.65	51.50
Stefan	50.68	64.63

All numbers are decoding time in seconds. The decoder uses the shared main memory (DDR2-DRAM) to store the pipeline FIFO buffers. The test platform has 16KB of local scratchpad (not used here) and 64KB of data cache per processor core.

Table 13. Wavefront Three-Core Decoder Performance

Video Sequence	512 kbps	1.5 mbps
Crew	42.86	63.00
Foreman	38.86	56.24
Mobile	38.82	56.14
News	29.59	44.80
Stefan	38.81	56.87

All numbers are decoding time in seconds. The test platform has 16KB of local scratchpad (not used here) and 64KB of data cache per processor core

Table 14和 Table 15 為我們提出使用 monitoring buffer 機制的 dynamic pipeline partition decoder、DRAM-based pipeline decoder 與 wavefront decoder 的加速比，加速比為 single-core decoder 的解碼時間除於各個平行化解碼器的解碼時間所得到的，理論上，當平行化解碼演算法成效夠好，其加速比會接近三，但現實中，因為 system bus 或 memory bandwidth 議題與 IPC overheads，會造成加速比通常是低於三以下的，但值得討論的是，可以從表中觀察出，當 bitrate 從 512kbps 提升到 1.5mbps 時，我們提出的 dynamic pipeline partition decoder 其效能是明顯提升的，較高的 bitrate 通常會改善其 load balance，因為 motion compensation 是整個解碼過程中，負載最重的模組，當 bitrate 提升時，Entropy-decode 模組的負載會相對提高，這時就會接近較好的 load balance 情況。

但另外一方面，wavefront decoder 在 bitrate 提升時，效能會下降，其原因是 Entropy-decode 模組在 H.264/AVC 是必須要連貫的，其意味著，需要 Entropy-decode 完一整張 frame 才能進行分配 MBs 給 worker threads 解碼，當 bitrate 提高時，Entropy-decoder 的負載會升高，因此整體效能會下降，而這也是為什麼我們提出了 interleave entropy-decode，這會減緩 entropy-decode 是連貫的影響，但是分配 entropy-decode task 與所有的 MB-decode tasks 給所有的核心並不是容易的事，因此我們實作了兩個版本的 interleaving wavefront video decoder，第一種為 three-core scheduling policy，即第一顆核心 row-by-row 的執行 entropy-decode，而當一個行的 MBs 執行完 entropy-decode 後，則指派給第二顆或第三顆核心執行剩下的解碼動作，假如其他兩顆皆在忙碌於執行 MB-decode tasks 的情況，第一顆則是會主動執行 MB decode 的動作。第二種為 four-core scheduling policy，即第一顆核心只會執行 entropy-decode，每當一個行的 MBs 執行完 entropy-decode，則會指派給其他三顆核心去執行，假如第一顆核心執行完所有 MBs entropy-decode 且其他核心還沒執行完所有 MBs-decode，則第一顆核心會進入 busy waiting，直到所有 tasks 結束為止。從 Table 16 可以觀察出來，當 bitrate 不管是 512 kbps 或 1.5 mbps 時，three-core

interleaving wavefront decoder 效能不會比 non-interleaving 版本還好，其可能原因是事實上視訊解碼的過程是高度的可變複雜度(variable complexity)，因此造成在 entropy-decode task 與 MB decoding tasks 之間交錯的成效並不好。但當採用 4-core interleaving wavefront decoder 則效能明顯比其他平行化解碼器好(除了 monitoring buffer 版本)，而 4-core policy 版本加速比則可以視為當 3-core policy 版本達到 good load balance 時加速比的 upper –bound。

我們能從實作上知道 wavefront-based decoder 在影片解析度提高時，效能可能會增加，Table 17 所示，使用 720x480 版本的 crew 且有三種不同的 bitrate:1.5、3.0、6.0mbps，測試我們提出的 three-core pipeline decoder 與 three-core wavefront decoder，能發現雖然 wavefront decoder 效能確實有所提升，但提出的 pipeline decoder 不管在哪種 bitrate 的情況下都會優於 wavefront decoder，而更不用提 4.5 節所討論的，wavefront decoder 會因為影片解析度增加，會需要額外的 off-chip memory 空間來儲存資料，但 pipeline-based decoder 對於解析度的改變與資料量的關係不高。

從以上的觀察可以看出，wavefront parallel video decoder 的 scalability 對於 memory subsystem 有高度的依賴性，雖然我們把 stack 放置在高速的 on-chip scrachpad memory，用來減少在 FPGA 平台上過於簡單的 cache 影響，但為了驗證 cache subsystem 對於 wavefront decoder 的影響，我們在 Nexux 7 2013 平板上執行 wavefront decoder，如 Table 18 所示，可以發現在平板上測試的加速比與在 FPGA 上是差不多的，由 Table 19 各 wavefront decoder 比較圖能知道，wavefront decoder 在三顆核心加速比大約是在 2.7 左右。

Table 14. Speedup Ratio of Different Decoders at 512 kbps

Sequence	Proposed Pipeline	DRAM Pipeline	Wavefront
Crew	2.82	1.83	2.28
Foreman	2.83	1.81	2.22
Mobile	2.90	1.74	2.21
News	2.33	1.41	1.93
Stefan	2.76	1.70	2.23

The single-core decoder is used as the reference decoder for the calculation of the speedup ratios of all the three-core decoders. Thus, the upper bound of the speedup factor is around 3

Table 15. Speedup Ratio of Different Decoders at 1.5mbps

Sequence	Proposed Pipeline	DRAM Pipeline	Wavefront
Crew	3.11	1.93	2.17
Foreman	3.10	1.94	2.12
Mobile	3.04	1.90	2.08
News	2.72	1.58	1.81
Stefan	2.84	1.85	2.10

The single-core decoder is used as the reference decoder for the calculation of the speedup ratios of all the three-core decoders. Thus, the upper bound of the speedup factor is around 3

Table 16. Speedup Ratio Wavefront with Interleaved Entropy Decoder

Sequence	512 kbps		1.5 mbps	
	3-core	4-core	3-core	4-core
Crew	2.16	2.72	2.11	2.95
Foreman	2.11	2.61	2.05	2.83
Mobile	2.09	2.63	2.01	2.82
News	1.80	2.36	1.75	2.59
Stefan	2.07	2.62	1.99	2.80

Table 17. Speedup Ratio for crew 720x480 video sequence

Bitrate	Proposed Pipeline Decoder	Wavefront Decoder
1.5 mbps	2.57	2.28
3.0 mbps	2.68	2.20
6.0 mbps	2.71	2.04

The single-core decoder is used as the reference decoder for the calculation of the speedup ratios of both three-core decoders. The decoding times of the single-core decoder are 314.7, 382.1, and 468.8 seconds, respectively. The video is the 300-frame Crew sequence.

Table 18. Speedup Ratio of Wavefront Decoder on NEXUS 7 2013

Sequence	512 kbps		1.5 mbps	
	3-core (NIE)	4-core (IE)	3-core (NIE)	4-core (IE)
Crew	2.32	2.80	1.88	2.97
Foreman	2.20	2.65	1.89	2.81
Mobile	2.11	2.60	1.78	2.72
News	1.87	2.55	1.52	2.30
Stefan	2.16	2.62	1.85	2.75

The average decoding times of the single-core decoder for different videos are 6.44, 5.68, 5.66, 3.29, and 5.62 seconds, respectively.The 3-core decoder does not interleave the entropy decoding (NIE) task with the MB decoding tasks while the 4-core decoder interleaves the entropy decoding (IE) task with the MB decoding tasks row-by-row. For the 4-core decoder, the first core is solely responsible for the entropy decoding task

Table 19. Comparisons of parallelization in various H.264/AVC Wavefront Decoder

	Mesa[5]	**Schöffmann[10]**	**Baik[8]**	**Jo[1]**
Parallelism	DLP	DLP	DLP	DLP
Codec	FFmpeg	Self-implemented	N/A	JM 13.2
Architecture	SGI Altix	x86 - xeon	CBE	x86 MP
Video Resol.	1080p	1080p	1080p	CIF、720p
Num. of core	3	4	5	4
Speedup	1.98x	2.34x	3.5x	2.9x

六、結論與未來展望

以往大家在多核心平台討論平行化 H.264/AVC 大部分採用 Data level parallelism video decoder，因為現今平行化計算追求的是 scalabilty 的極致，希望能越多核心同步處理越好，而 DLP 在 scalability 與程式設計師在撰寫上的負擔都佔有優勢，不需要如 TLP video decoder 需要仔細的對 decoding steps 作切割，且需要對解碼行為有一定的了解才有辦法達到 good load balancing 的切割方式，但經過前面章節的探討，我們能知道，現實中在嵌入式環境下的情況又是不一樣的，DLP 在 off-chip memory 使用上相對於 TLP 有很明顯的差異，在嵌入式的環境中，對於記憶體的使用是很重要的議題，DLP 在這方面將是明顯的缺點。另外 DLP 的 scalability 優勢，在現在的智慧型手機或是平板，因為散熱因素或追求輕薄等等的情況下，大多是採用四核心、或最多為八核心，似乎沾不到太多好處。而且透過我們提出的 IPC datapath，將更有效的發揮 pipeline-based video decoder，在傳輸資料上負擔降低，在 system bus 使用頻率上也降低，而且在程式設計上，透過我們提供的 thin OS kernel 將能輕鬆的開發 pipeline-based program，而我們提出的 dynamic pipeline partition video decoder 也能透過 monitoring buffer 的技術來改善 pipeline-based video decoder 效能，在三顆核心的情況下可以接近三的加速倍率。

雖然我們的實作目前看起來很完善，但不管是提出的平台架構與 pipeline-based、wavefront-based video decoder 都仍有需要做加強和改進的地方。首先系統架構上，目前在 Xilinx Vertex-5 FPGA 開發板上，沒有支援 coherence cache，這會造成程式設計師需要自行呼叫 invalidation function 處理 race condition 問題，而這部分需要從電路方面做修改，往後實驗室會關掉 Microblaze 支援的 cache，然後放上我們自行設計的 coherence cache，這可能會解決 DLP video decoder 在此平台下效能無法完全發揮的問題(需要在做相關數據分析與驗證)。

另外 video decoder 的部分，wavefront decoder 目前是採用 static scheduling，未來希望能做成 dynamic scheduling，即 scheduler(或稱 main thread)來維護 task queue，

而 worker threads 則在 queue 中取 task 來執行，這將會解決 static scheduling 較常會發生 load unbalance 情況，但這部分也需要 coherence cache 的支援，因為在設計 dynamic scheduling 需要使用 off-chip shared memory，而避免複雜設計所造成的頻繁存取 off-chip memory 而造成效能下降，就需要完善的 cache subsystem 的支援。另外 pipeline decoder 部分，目前是採用 dynamic pipeline partition 搭配 monitoring buffer 的技術，這部分希望每個核心 task 的分配能夠有彈性，因為目前設計 motion vector reconstruction 或執行 SaveNeighborPixelForIntra 等等的模組需要在指定的核心或是限制執行順序，而其原因是我們的 IPC controller 目前設計是單向的，只能由第 p 顆核心將資料傳遞給第(p+1)顆核心，此種往下傳遞資料的方式，這會造成資料不能往回送，因此降低了 dynamic partition 的彈性，而這部分實驗室之後也會將支援雙向的 IPC controller。

而最後我們希望我們提出的系統架構將能被應用在現今的多核心平台上，且我們提出的平行化視訊解碼的部分，也能夠在資源有限的嵌入式環境中，發揮其功效。

參考文獻

[1] JO, Song Hyun; JO, Seongmin; SONG, Yong Ho. Efficient coordination of parallel threads of H. 264/AVC decoder for performance improvement.*Consumer Electronics, IEEE Transactions on*, 2010, 56.3: 1963-1971.

[2] VAN DER TOL, Erik B.; JASPERS, Egbert G.; GELDERBLOM, Rob H. Mapping of H. 264 decoding on a multiprocessor architecture. In: *Electronic Imaging 2003*. International Society for Optics and Photonics, 2003. p. 707-718.

[3] SEITNER, Florian H., et al. Evaluation of data-parallel splitting approaches for H. 264 decoding. In: *Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia*. ACM, 2008. p. 40-49.

[4] MEENDERINCK, Cor, et al. Parallel scalability of video decoders. *Journal of Signal Processing Systems*, 2009, 57.2: 173-194.

[5] Á LVAREZ MESA, Mauricio, et al. Scalability of macroblock-level parallelism for H. 264 decoding. In: *Parallel and Distributed Systems (ICPADS), 2009 15th International Conference on*. IEEE, 2009. p. 236-243.

[6] AZEVEDO, Arnaldo, et al. Parallel H. 264 decoding on an embedded multicore processor. In: *High Performance Embedded Architectures and Compilers*. Springer Berlin Heidelberg, 2009. p. 404-418.

[7] BAKER, Michael A., et al. A scalable parallel H. 264 decoder on the cell broadband engine architecture. In: *Proceedings of the 7th IEEE/ACM international conference on Hardware/software codesign and system synthesis*. ACM, 2009. p. 353-362.

[8] BAIK, Hyunki, et al. Analysis and parallelization of H. 264 decoder on cell broadband engine architecture. In: *Signal Processing and Information Technology, 2007 IEEE International Symposium on*. IEEE, 2007. p. 791-795.

[9] CHONG, Jike, et al. Efficient Parallelization of H. 264 Decoding with Macro Block

Level Scheduling. In: *ICME*. 2007. p. 1874-1877.

[10] SCHÖFFMANN, Klaus, et al. An evaluation of parallelization concepts for baseline-profile compliant H. 264/AVC decoders. In: *Euro-Par 2007 Parallel Processing*. Springer Berlin Heidelberg, 2007. p. 782-791.

[11] KIM, Minsoo, et al. H. 264 decoder on embedded dual core with dynamically load-balanced functional paritioning. In: *Image Processing (ICIP), 2010 17th IEEE International Conference on*. IEEE, 2010. p. 3749-3752.

[12] KIM, Minsoo, et al. Hybrid partitioned H. 264 full high definition decoder on embedded quad-core. *Consumer Electronics, IEEE Transactions on*, 2012, 58.3: 1038-1044.

[13] CHEN, Ding-Yun, et al. A novel parallel H. 264 decoder using dynamic load balance on dual core embedded system. In: *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012. p. 2313-2316.

[14] Chun-Jen Tsai, Yan-Ting Chen, and Chien-Chih Tseng, "An Efficient Application Processor Architecture for Multi-Core Software Video Decoding," accepted by IEEE Trans. on Circuits and Systems for Video Technology on April 17, 2014, available on-line at http:/www.cs.nctu.edu.tw/~cjtsai/research/pipeline_ipc.

[15] WANG, Sung-Wen, et al. A multi-core architecture based parallel framework for H. 264/AVC deblocking filters. *Journal of Signal Processing Systems*, 2009, 57.2: 195-211.

[16] WANG, Ronggang, et al. High throughput and low memory access sub-pixel interpolation architecture for H. 264/AVC HDTV decoder. *Consumer Electronics, IEEE Transactions on*, 2005, 51.3: 1006-1013.

[17] CHEN, Xianmin, et al. Block-pipelining cache for motion compensation in high definition H. 264/AVC video decoder. In: *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*. IEEE, 2009. p. 1069-1072.

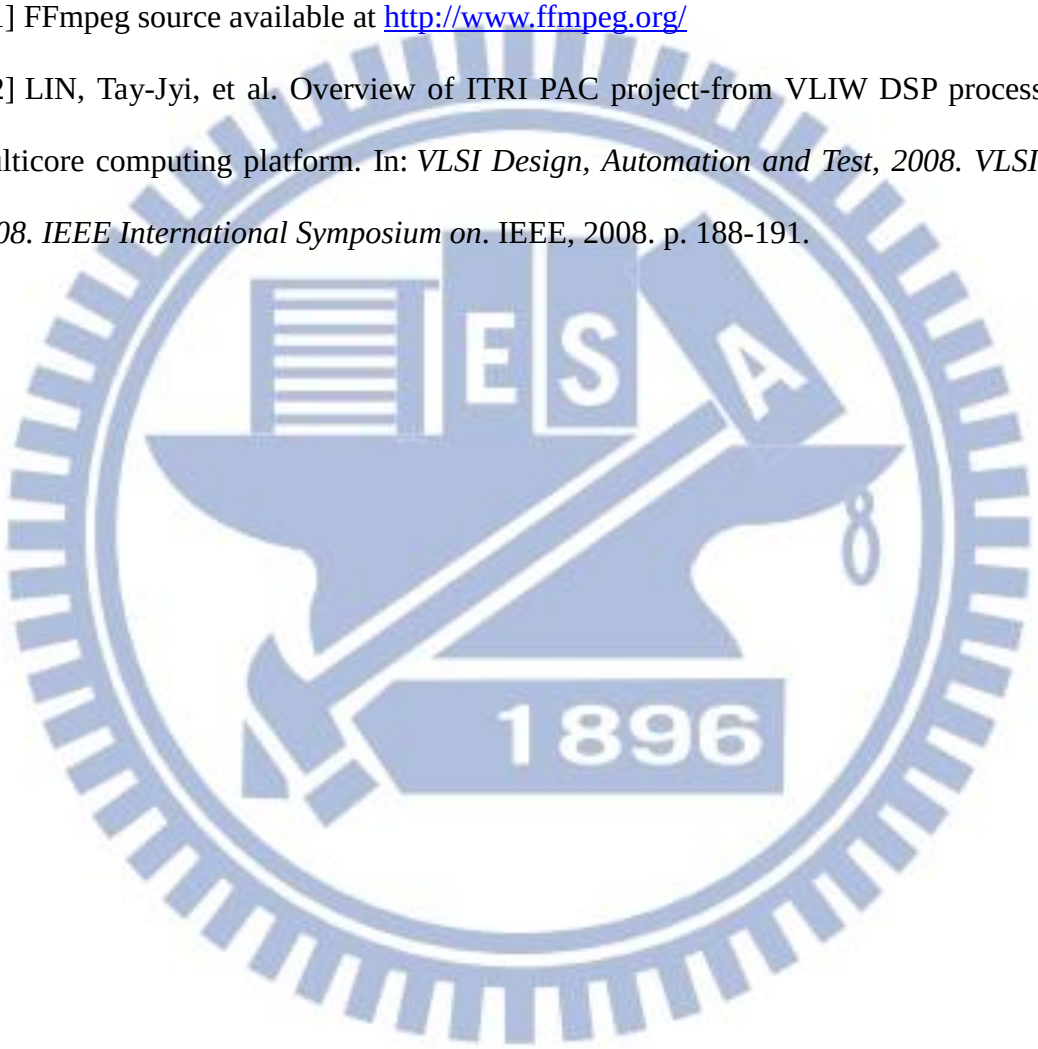
[18] HOROWITZ, Michael, et al. H. 264/AVC baseline profile decoder complexity analysis. *Circuits and Systems for Video Technology, IEEE Transactions on*, 2003, 13.7: 704-716.

[19] JM source available at http://iphome.hhi.de/suehring/tml/

[20] Android source available at http://source.android.com/

[21] FFmpeg source available at http://www.ffmpeg.org/

[22] LIN, Tay-Jyi, et al. Overview of ITRI PAC project-from VLIW DSP processor to multicore computing platform. In: *VLSI Design, Automation and Test, 2008. VLSI-DAT 2008. IEEE International Symposium on*. IEEE, 2008. p. 188-191.



附錄 A:Speedup ratio of different cache size

因為 wavefront 主要資料放置在 off-chip memory，故 cache size 會影響其效能，故附錄 A 為測試不同 cache size 對於 wavefront decoder 效能的影響，wavefront decoder 版本為 3-core non-interleaving。

Table 20. SPEEDUP RATIO OF DIFFERENCE DECODERS AT 512 KBPS

Sequences	16K cache - Wavefront	64K cache - Wavefront
Crew	2.27	2.28
Foreman	2.21	2.22
Mobile	2.19	2.21
News	1.92	1.93
Stefan	2.21	2.23

Table 21. SPEEDUP RATIO OF DIFFERENCE DEOCDERS AT 1.5 MBPS

Sequences	16K cache - Wavefront	64K cache - Wavefront
Crew	2.16	2.17
Foreman	2.09	2.12
Mobile	2.07	2.08
News	1.80	1.81
Stefan	2.09	2.10

附錄 B:Macroblock Mode Distributions and Performance (Detail version)

附錄 B 為五個 MPEG 影片(皆為 300 張 frame，CIF)所有的 Macroblcok mode 比重，與 static pipeline partition 的加速比。Crew 在 Intr4x4 與 Intra16x16 比重相對於其他組影片高，News 為較靜態的影像，其 skip MB 比重較重。

Macroblock Mode Distributions and Performance (Detail version)								
Video sequence	Bit rate	Intra 16x16	Intra 4x4	Inter 16x16	Inter 8x16 or 16x8	Inter 8x8	Skip MB	Performance gain(in times) over single-core
Crew	512K	8.52%	5.87%	32.14%	22.89%	7.8%	22.78%	2.51
	1.5M	3.63%	11.75%	26.82%	26.84%	22.83%	8.13%	2.51
Foreman	512K	2.15%	1.94%	33.43%	21.11%	11.7%	29.68%	2.53
	1.5M	1.41%	2.77%	29.58%	25.31%	25.53%	15.4%	2.58
Mobile	512K	0.11%	0.37%	37.51%	22.79%	12.81%	26.41%	2.43
	1.5M	0.1%	0.38%	33.97%	27.13%	25.27%	13.15%	2.52
News	512K	0.34%	1.11%	10.53%	9.3%	9.31%	69.4%	2.18
	1.5M	0.18%	1.64%	12.79%	12.39%	21.18%	51.82%	2.13
Stefan	512K	1.14%	1.81%	24.83%	19.94%	14.61%	37.67%	2.26
	1.5M	1.42%	2.37%	25.83%	22.38%	24.73%	23.25%	2.38

附錄 C:PAC Duo Platform

C.1、摘要

在本附錄,我們在多核心應用處理器上提出了應用在視訊解碼上的 software pipeline 架構，此設計是實作在 ITRI PAC-Duo 平台。由實驗顯示，當我們利用 PAC cores 來執行 software pipeline 機制可以得到比由 ITRI 提供的 single PAC core decoder 還快約 2.11 倍，而在多核心平台上使用 software pipeline 會存在著一些 overhead 會導致效能降低，如資料在兩核心之間的搬移與 synchronization 或是因為 MBs 為 variable complexity 所造成的 load unbalancing 問題，我們會利用 FIFO circular buffer 和優化 DMA 等方法來改進 software pipeline 效能。

C.2、系統架構

在這章節會介紹我們使用的硬體 PAC 平台，第 3.1 節會介紹 PAC 系統架構，第 3.2 節則是介紹 PAC SoC 晶片，最後 3.3 節為介紹 PAC DSP 處理器。

C.2.1 PAC 系統架構

Parallel Architecture Core(PAC)平台是工研院晶片系統技術發展中心所研發出的多媒體處理器核心技術(PAC DSP)為主的多核心平台，可廣泛應用在可攜式多媒體播放器或智慧型手機等行動多媒體嵌入式裝置中。

PAC Duoo 開發板為近年工研院極力推廣的自製開發板，外觀如 Figure 55 所示，主要原件有 I/O(UART、USB、Ethernet、Audio in/out)、LCD 觸控螢幕、DSP ICE port、ARM ICE port 還有最主要的 PAC Duo SoC 晶片，其晶片包括 ARM 處理器和 PAC DSP 處理器。

Figure 55. PAC Duo 開發板

PAC 平台的特色:

- 異質多核心晶片(PAC Duo SoC):由一個 ARM(A926EJ-S)和兩個工研院自製開發的 PAC DSP，Figure 56 所示

  ARM : A926EJ-S　204MHz

  DSP: 5-way VLIW architecture

  9-stage pipeline

  32KB instruction cache

  64KB SRAM data memory

  204MHz

- DVFS:動態調節電源電壓與時脈頻率，可用來幫助降低功耗的技術
- FPGA 擴充:PAC 平台目前可擴充 Virtex-5 FPGA 板，可自行設計硬體電路

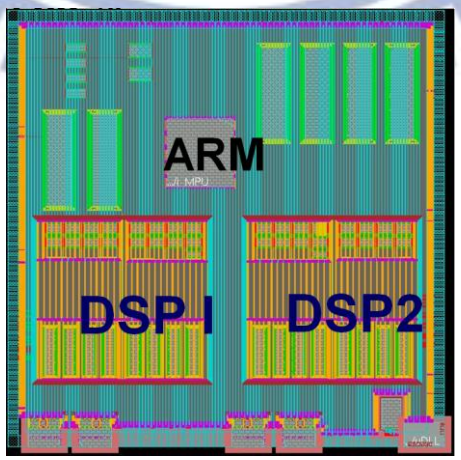


Figure 56. PAC Duo SoC 晶片

C.2.2 PAC Duo SoC 系統架構

Figure 57 為 PAC Duo SoC 系統架構圖，由 AXI、AHB、APB 三條 BUS 所組成，其整個平台的 shared memory 架構如下:

- AXI on-chip SRAM : 128 KB
- AXI off-chip DDR2-SDRAM : 128 MB
- AHB on-chip SRAM : 256 KB
- AHB off-chip SDRAM : 128 MB
- AHB off-chip Flash : 128 MB

此平台主要整合三個子系統:ARM 子系統、DSP 子系統和慢速 I/O 子系統，ARM 子系統 AHB-lite BUS 上有 ARM 926EJ-S 處理，目前 ARM 處理器可以執行 Andorid 或 linux 作業系統。DSP 子系統設計目的是在數位訊號、多媒體影音加解碼方面能展現較高效能的處理，而在 AXI BUS 上的稱作 DSP 子系統。

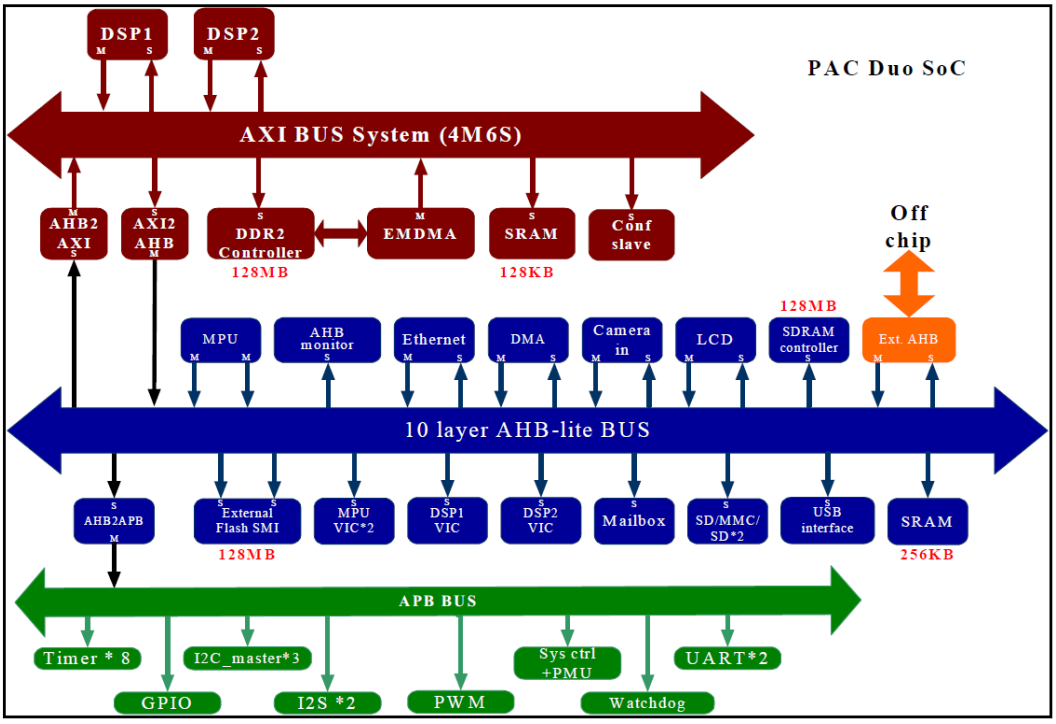


Figure 57. PAC Duo SoC 系統架構圖

C.2.3 PAC DSP 系統架構

PAC DSP 開發重點著重於低功耗、高效率的 32-bits 數位訊號處理器核心，PAC DSP 是以 5-WAY VLIW 架構和 SIMD 指令集提昇平行度的運算量，這種架構很適合媒體應用程式，PAC DSP 特點如下:

- Scalable VLIW data path : PAC DSP 由一個 Scalar Unit 和兩個 Cluster，每個 Cluster 包涵 Load/Store Unit 和 Arithmetic Unit，共有五個運算單元，可有效提昇運算能力。
- Variable instruction word/packet length : 這樣機制有助於減少 Code size。
- Heterogeneous Register Files : 每個運算單元都有自己獨立的暫存器，對於每個暫存器可直接拉線到各自運算單元，這樣可減少能量的損失，也可減少硬體的面積。
- Constant Register File : 在這些暫存器中可儲存固定不常變動的常數，可減少資料的搬運降低能源的消耗。
- Inter-cluster communication (ICC) by memory controller : 透過特殊指令集(ICC)，可使兩個 Cluster 之間互相溝通，可互相提昇兩個 Cluster 合作的效能。
- Dynamic power management : 可動態調整 PAC DSP 的頻率以減少能源的消耗。
- Customized functional unit interface : 可自訂硬體加速器透過這個界面與 PAC DSP 溝通並提昇 PAC DSP 的運算能力。

Figure 58 是 PAC DSP 架構圖，在 DSP 中最主要有幾個部份，即 Program Sequence Control Unit(PSCU)、Scalar Unit 和兩個 Cluster，PSCU 用來控制執行中程式的流程，主要功能是更新 Program Counter (PC)、Fetch address 和處理中斷程序。Scalar Unit 可以當作一個簡單的 32-bit RISC 處理器，可有效增加指令之間的平行度(Instruction-level parallelism)。每一個 Cluster 包括 Arithmetic Unit 和 Load/Store Unit，

Arithmetic Unit 和 Load/Store Unit 有基本的 Arithmetic、Comparison 和 Bit-Manipulation 指令外，Arithmetic Unit 有 Multiplication 指令，而 Load/Store Unit 則是有 Load and Store 指令。

Figure 59 為 PAC DSP 的指令封裝格式，PAC DSP 一次會讀取五到指令交給五個計算單元進行處理，依次是 Scalar Unit、Cluster 1:Load/Store Unit、Cluster 1:Arithmetic Unit、Cluster 2:Load/Store Unit 和 Cluster 2:Arithmetic Unit，而在 register 使用上會依照 cluster 不同而有所限制。

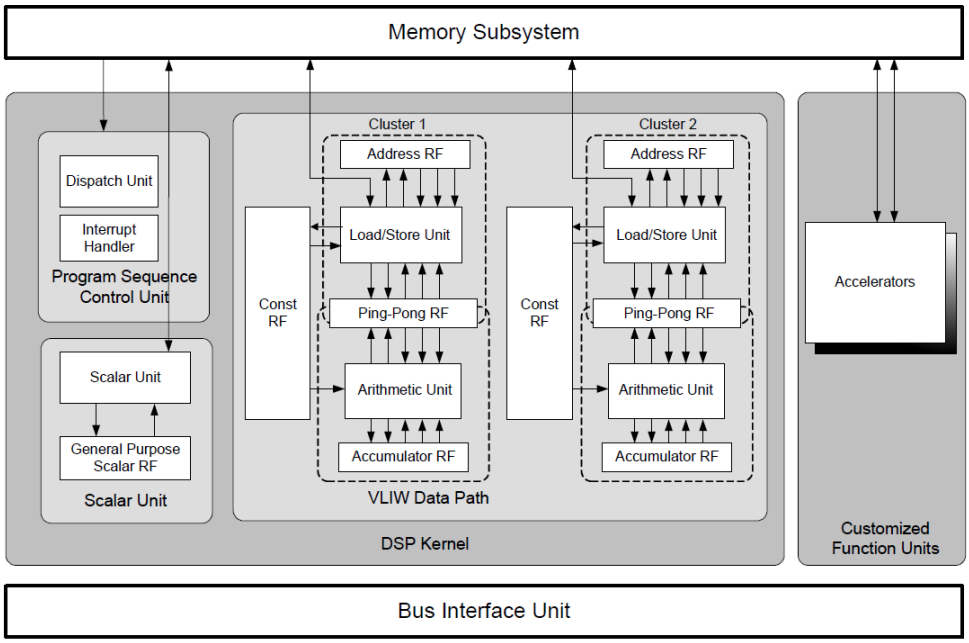


Figure 58. PAC DSP 系統架構圖

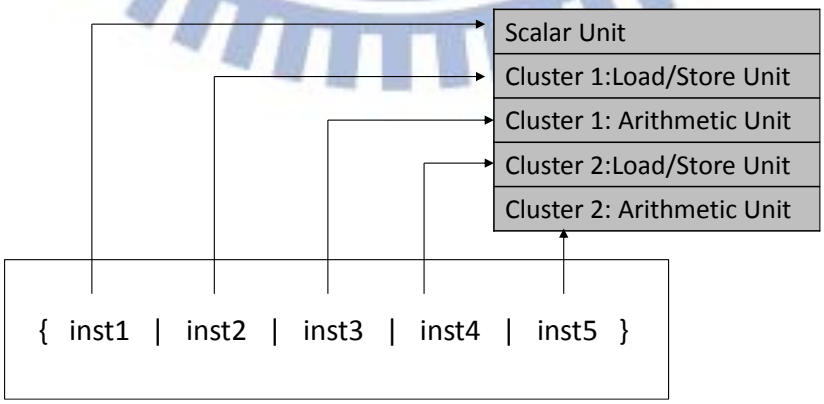


Figure 59. PAC DSP 指令封裝格式

C.3、Software Pipeline 實作

C.3.1 Issues with software pipelines for video decoding

在 software pipeline 的設計中，有兩個最主要影響效能上的問題 第一個是 pipeline stage buffer 放置在 main memory(DRAM)，而放置在 DRAM 將會造成 access latency 過高的問題，在兩顆核心要對 shared memory 作存取時，勢必需要處理 race condition 問題，而且 stage buffer 時常會被存取，因此 cache coherence overhead 將不可被忽視。

第二個議題則是因為每個 MB 在解碼過程中是 variable complexity，這會造成 load unbalancing，即核心可能時常處於飢餓的狀態(starving)。

C.3.2 The proposed software pipeline architecture

首先第一個議題，我們可以利用 HMP 平台來解決，傳統 HMP 應用處理器架構(如 PAC-Duo)，每顆核心具有一塊 local memory，而每個 VLIW PAC-DSP core 皆有一塊 64KB 的 local memory，我們利用這塊 on-chip memory 作為 stage buffer 放置的儲存裝置，這將會大幅減少讀取 stage buffer 的時間。

為了解決第二個問題，我們設計了一個 FIFO circular buffer，用來吸收因為 MB 為 variable complexity 所造成的負擔，buffer 每個 node 需要 3372 byte 來儲存資料，理論上，circular buffer depth 要越大越好，但現實中因為 local memory 有空間的限制，因此 pipeline stage 只有 13 個 nodes，我們利用 DMA 來傳遞資料，buffer node 資料結構如 Figure 60 所示，而 Figure 61 為我們提出的 software pipeline 架構圖。

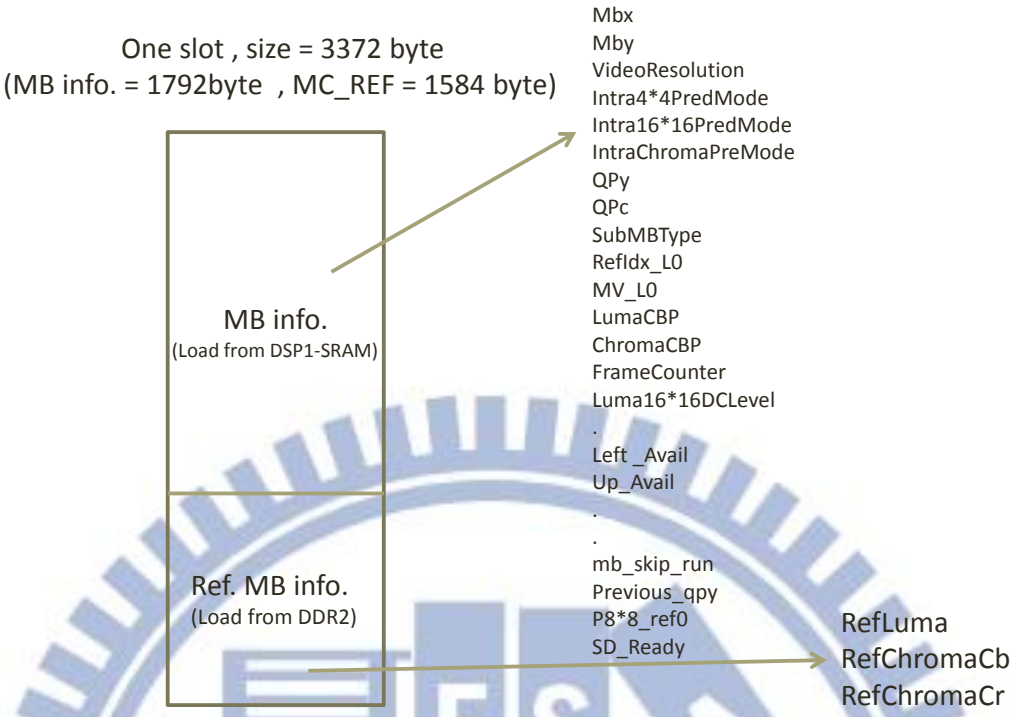


Figure 60. Circular buffer node structure of each MB

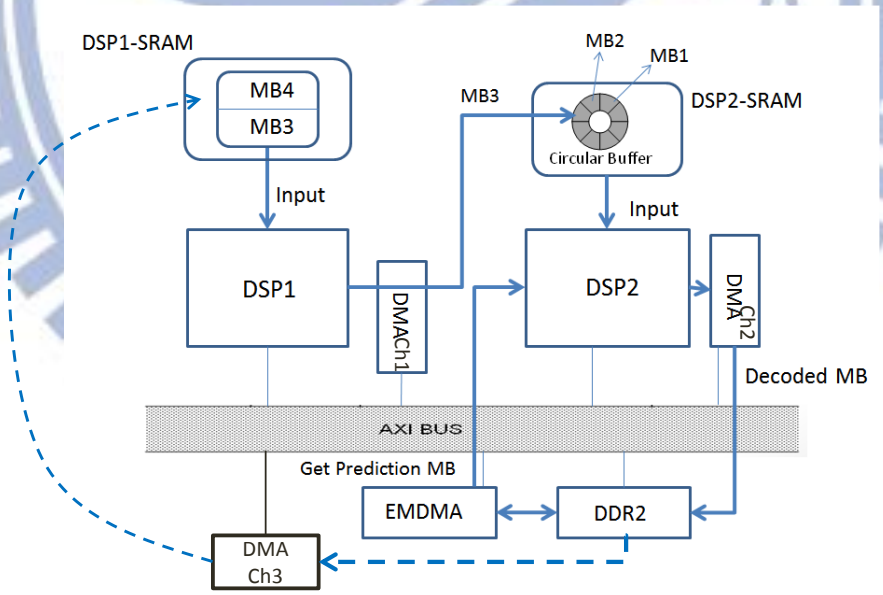


Figure 61. The proposed software pipeline architecture for the H.264/AVC decoder

C.3.3 Pipeline stage partition

在 H.264/AVC video decoder 解碼流程主要分成五部分，VLD、Intra/Motion Prediction、IT/IQ、Vídeo Reconstruction (VR)和 Deblock filter(DF)，在 motion prediction(MP)模組中，又可以分成 motion vector reconstruction(MVR)和 data interpolation(DI)兩部分。在 PAC-Duo 中執行 DF 將會花費大量的時間，應該要把 DF 獨立交給單一 DSP 執行，但因為 PAC-Duo 只有兩顆 DSP 核心，因此我們選擇把 DF 關掉，如 Figure 63 所示。Figure 62 為解碼的流程圖，在 PAC-Duo 平台中，EMDMA 會處理 DI 部分，因此將 IQ/IT 放置在第二顆核心開頭，這將會產生 overlapping，而達到提升效能。

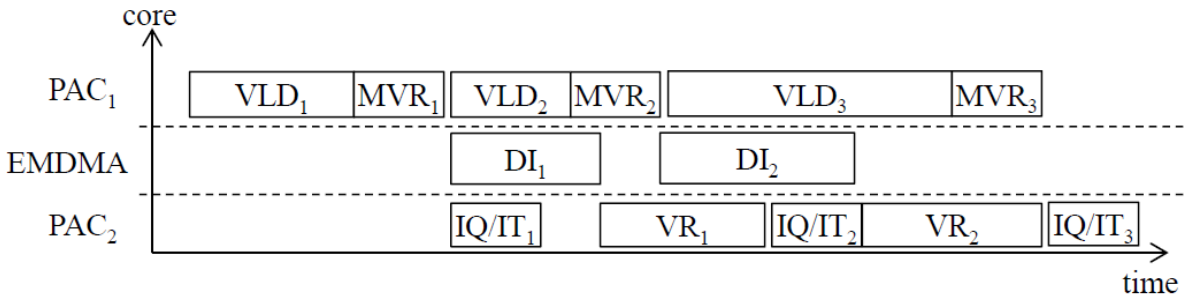


Figure 62. Overlapping of each core's operations. The subscripts are MB number

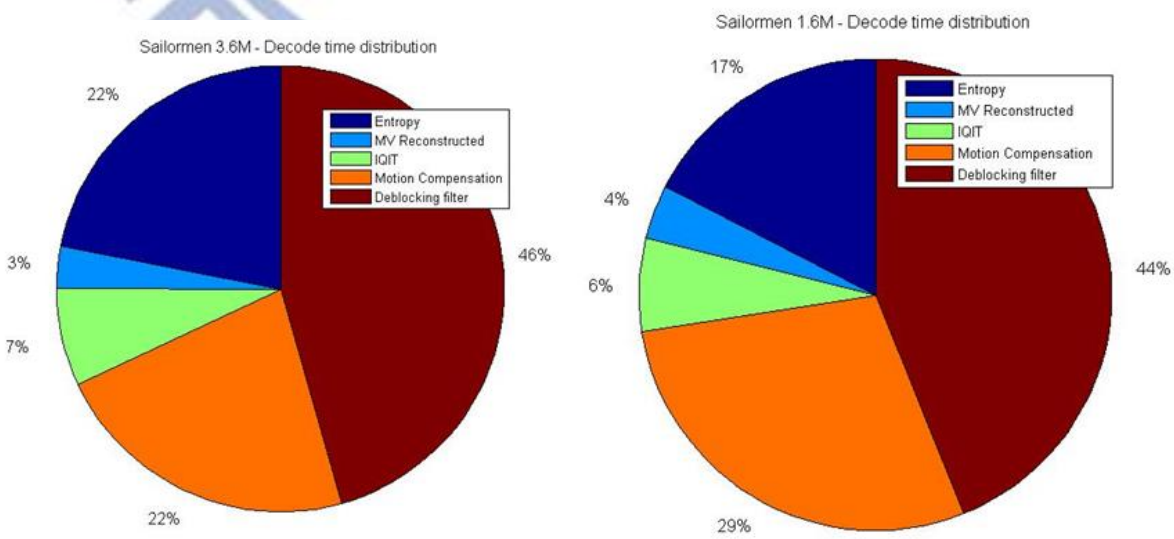


Figure 63. 各模組時間分佈

C.4、實驗結果

在此章節我們使用五個 VGA video sequences，分別是 Keiba、Flower Vase、Crew、Race Horse 和 Sailor。每個影片皆由 JM1.84 baseline tool 所編碼。

C.4.1 Impact of MB type distribution

Table 22 為每個 test sequence 的 macroblock types 的分佈，根據我們的實驗，skip MBs 對於 software pipeline decoder 效能有明顯的影響，當 skip MBs 比重越重時，解碼器的效能將會越不好，換句話說，當 non-skip MB 比重較高時，解碼效能會較好。而這原因是很簡單的，當 skip MB 出現時，entropy-decode 執行時間將會大量減少，假如出現連續的 skip MBs，會造成 circular buffer 呈現 overflow(現實中 buffer depth 會受限於記憶體大小而限制其個數)，第一顆核心則會進入 idle 狀態，造成效能下降，這意味著，non-skip MBs 比重較多的影片，其會越接近 good load balancing。

Table 22. Distribution of the MB types

MB type percentage					
Video sequence	Bit rate	Intra MB	Inter MB	Skip MB	Performance gain (in times) over single-core
Keiba	1.6M	16.8%	37.8%	45.5%	2.03x
	3.6M	17.9%	47.8%	34.3%	1.91x
Crew	1.6M	18.9%	56.6%	24.6%	2.06x
	3.6M	21.5%	65.4%	13.1%	1.98x
FlowerVase	1.6M	6.2%	49.4%	44.4%	1.96x
	3.6M	6.3%	64%	29.8%	1.91x
RaceHorse	1.6M	9.8%	55.5%	34.7%	2.11x
	3.6M	10.8%	67.8%	21.5%	2.01x
Sailormen	1.6M	2.5%	59.3%	38.2%	2.11x
	3.6M	3.9%	71.8%	24.3%	2.05x
Each sequence is 640*480 and 300 frames per video sequence					

C.4.2 Impact of circular buffer depth

Figure 64 為 buffer 在 Sailorman@1.6 mbps frame #78 的使用度(maximal depth 為 13)，這張 frame 的 pipeline 解碼速度相對於 single-core 快 2.18 倍，我們可以發現會出現 buffer overflow 的情況，這意味著，當增加深度可以使效能再向上提升。

而另外一個重點則是，可以發現當連續的 skip MBs 出現時，buffer depth 將會直線上升，甚至發生 overflow 得情況，這將能驗證 4.1 節所提到的。

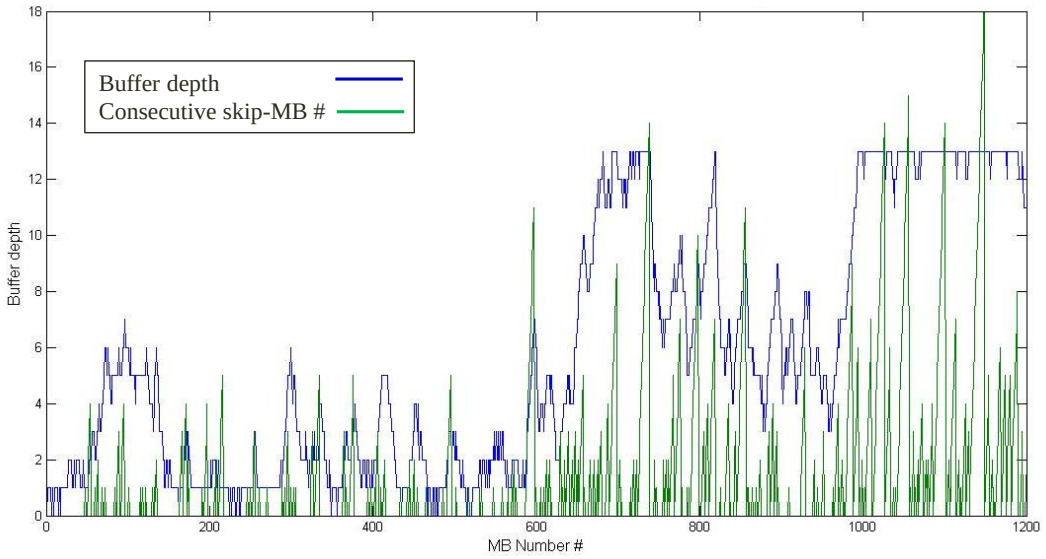


Figure 64. Impact of the buffer depth

C.4.3 Experimental results

Table 23 為總結的效能加速比，我們可以發現存在測試影片其加速比超過兩倍，在雙核心平行化加速下，超過兩倍是不合理的，但是因為 EMDMA 會執行 DI，且在傳送資料過程中，第一顆核心仍然在執行 task，因此超過兩倍為 overlapping 所造成的。

Table 23. Decoding 300 frames of VGA video at two different bitrates

	Single-core by ITRI (in secs)		Proposed Dual-core (in secs)		Speedup Ratio	
	1.6M	3.6M	1.6M	3.6M	1.6M	3.6M
Keiba	15.32	19.39	7.56	10.12	*2.03x*	*1.92x*
Crew	16.93	21.24	8.21	10.73	*2.06x*	*1.98x*
FlowerVase	15.70	20.75	8.00	10.85	*1.96x*	*1.91x*
RaceHorse	17.85	22.23	8.44	11.04	*2.11x*	*2.01x*
Sailman	18.82	22.89	8.92	11.15	*2.11x*	*2.05x*