

國立交通大學

資訊科學與工程研究所

碩士論文

即時移除不再使用到的像素來減少 real time H.264 解
碼端解碼圖片緩衝器的容量

Reducing the real time H.264 Decoder's Decoded Picture
Buffer Size by removing the pixels that are no longer
used in time

研究生：劉盈亨

指導教授：傅心家教授、鍾崇斌教授

中華民國一百零一年九月

即時移除不再使用到的像素來減少 real time H.264 解
碼端解碼圖片緩衝器的容量

Reducing the real time H.264 Decoder's Decoded Picture
Buffer Size by removing the pixels that are no longer
used in time

研究生：劉盈亨

Student：Ying-Heng Liu

指導教授：傅心家、鍾崇斌

Advisor：Hsin-Chia Fu、
Chung-Ping Chung

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
Sep. 2012

Hsinchu, Taiwan, Republic of China

中華民國一百零一年九月

即時移除不再使用到的像素來減少 real time H.264 解碼端解碼圖片緩衝器的容量

學生：劉盈亨

指導教授：傅心家博士、鍾崇斌博士

國立交通大學資訊科學與工程研究所碩士班

摘要

在標準 H.264 解碼端中，解碼圖片緩衝器(Decoded Picture Buffer(DPB)) 占了 95% 以上的記憶體空間以及 95% 以上所需的晶片面積。若能有效減少解碼圖片緩衝器所需的記憶體空間，即可降低所需的晶片面積以及電量消耗，特別是針對嵌入式系統而言較為有利。

在此篇 paper 中，我們主要貢獻在於分析圖片間像素的參考特性，發現每一張圖片在特定時間點後就有 60~80% 的像素不再被使用，進而可由解碼圖片緩衝器內即時移除。另外，根據資料串流的特性，我們提出適合的解碼工作排程，以達到穩定的圖片輸出率。為了減少解碼圖片緩衝器的儲存空間，我們針對每一張圖片選擇一個適當時機移除不再被使用的像素，並提出一個預先解碼機制事先確認哪些像素在此時機後不再被使用，另外，我們提出一個新的記憶體架構用以管理資料。在不影響解碼效能的情況下，相較於傳統 H.264 解碼端中，我們的設計在圖片品質上損失 4.1% 的情況下，能省下 62.4% 的記憶體空間。在晶片面積以及電量消耗部分，相較於傳統 H.264 解碼器，能省下 66.7% 的晶片面積以及 56.2% 的電量消耗。

Reducing the real time H.264 Decoder's Decoded Picture Buffer Size by removing the pixels that are no longer used in time

Student : Ying-Heng Liu

Advisor : Hsin-Chia Fu 、 Chung-Ping Chung

Institute of Computer Science and Engineering
National Chiao-Tung University

Abstract

In the H.264 video compression standard decoder, the Decode Picture Buffer (DPB) consumes over 95% memory space and the chip area. If we can reduce the DPB size, it can achieve the reduction in chip area and power consumption especial the embedded video decoder.

In this paper, we focus on the analysis of the reference behavior among the frames. We find that there are 60% to 80 % of pixels of each frame which are no longer used in specific time. Therefore, we can remove the unused pixels after the specific time to reduce the DPB size. Besides, according to the properties of streaming data, we proposed the suitable decoding pipeline to achieve the steady frame output rate. In order to reduce the DPB size, we propose the suitable time to remove the unused pixels for each frame, a pre-decoding mechanism to obtain the lifetime information of pixels, and new memory architecture to store the frames. Without affecting the decoding performance, our design is able to achieve a 62.4% reduction in storage, with a decrease in image quality less than 4.1%. Compared to the traditional H.264 decoder, our design can achieve a 66.7% reduction in chip area and a 56.2% reduction in power consumption.

致謝

經過三年的努力，終於完成了碩士論文，也經由完成此論文，將我的學生生涯畫下句點。這一路上，雖然經過很多挫折與困難，但由於老師與實驗室同學的陪伴與幫助使我度過重重難關，也在學術知識上與個人發展上都更加充實與成長。

首先最要感謝的就是鍾崇斌老師以及傅心家老師，在我不斷犯錯的時候，老師們總是有耐心的教導以及鼓勵我，並且適時的給予建議讓此篇論文能夠繼續下去。另外，口試時也非常感謝謝萬雲老師的寶貴意見，使得這篇論文可以更加完整。另外，還要感謝博士班學長翁綜禧，總是鼓勵我並且很熱心地與我討論，讓我在遇到每一個問題的時候，都能有更完整的思考。還要謝謝實驗室的學長姊、同學以及學弟，不論是一起做研究、一起吃飯、一起出遊總是這麼的快樂。最後要感謝我的家人，一路上的支持與鼓勵，才能有今天的我。

劉盈亨 謹誌於

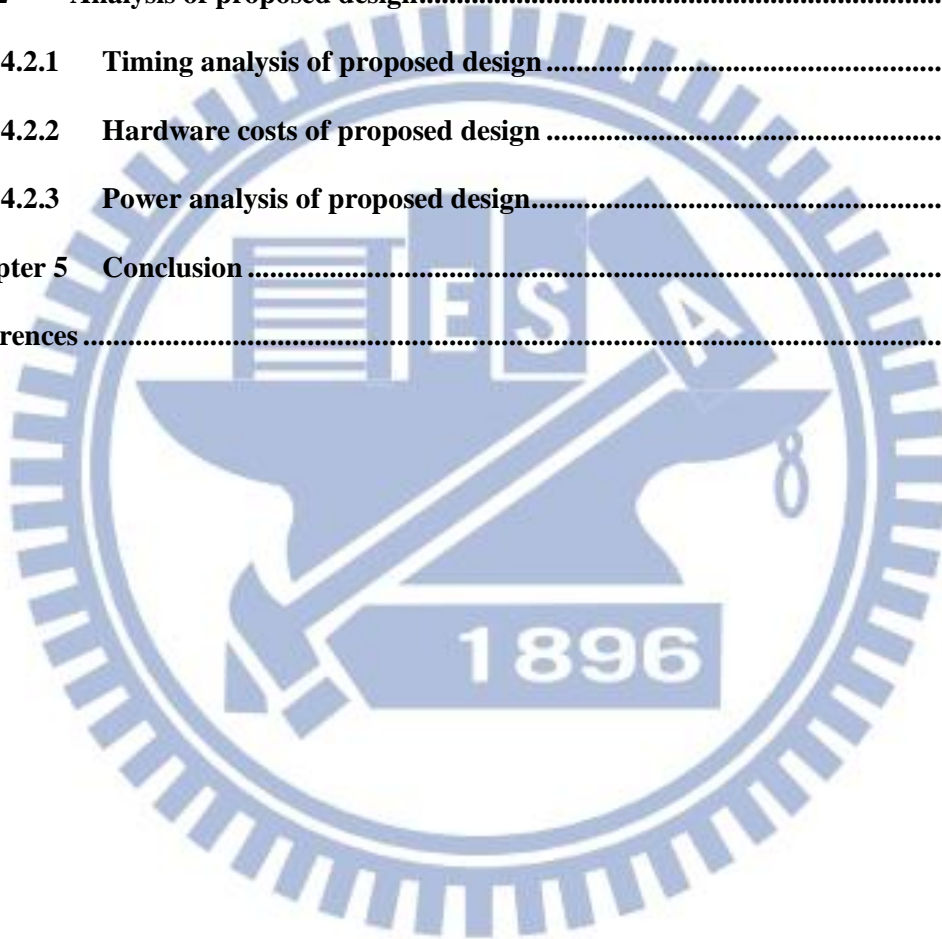
國立交通大學資訊科學與工程研究所碩士班

中華民國一百零一年九月

Content

摘要.....	i
Abstract.....	ii
致謝.....	iii
Content.....	iv
List of figure.....	vi
List of table	viii
Chapter 1 Introduction.....	1
Chapter 2 Background and Related work	3
2.1 Background.....	3
2.1.1 Reference behaviors of frames	3
2.1.2 Output time of frames for display.....	23
2.1.3 Replacement Policy of the DPB.....	23
2.2 Related work.....	23
2.2.1 Hierarchical B Pictures reference scheme.....	23
2.2.2 Related work	24
2.2.3 Summary of related work.....	26
2.3 Opportunity	27
Chapter 3 Design.....	28
3.1 Propose a new decoder scheme and decoding pipeline	29
3.2 Decision of Partial Frame Data Removed Time.....	31
3.3 Pre-decoding mechanism	32
3.3.1 Partial decoder	33
3.3.2 PRT Table.....	33
3.3.3 Input Buffer.....	35

3.4	Modified DPB	36
3.4.1	Complete Reference Frame Storage	37
3.4.2	Fragmented Reference Frame Storage.....	38
3.5	Compensated method while the conflicts misses of FRFS happened.....	43
Chapter 4	Simulation.....	48
4.1	Experiment	48
4.2	Analysis of proposed design.....	60
4.2.1	Timing analysis of proposed design	61
4.2.2	Hardware costs of proposed design	64
4.2.3	Power analysis of proposed design.....	65
Chapter 5	Conclusion	67
References	68



List of figure

Figure 2-1 An example of display and decoding orders of a H.264 coded video..... 4

Figure 2-2 An example of status of reference frame in the DPB when each frame decoded..... 7

Figure 2-3 (a) The percentage of pixels of I frame which are still referenced using the number of the subsequent decoded frames as the horizontal axis. (b) The percentage of pixels of P frame which are still referenced using the number of the subsequent decoded frames as the horizontal axis. (c) The percentage of pixels of B₁ frame which are still referenced using the number of the subsequent decoded frames as the horizontal axis. (d) The percentage of pixels of B₂ frame which are still referenced using the number of the subsequent decoded frames as the horizontal axis..... 9

Figure 2-4 The percentage of pixels of frame of different videos which are still referenced using the number of the subsequent decoded frames as the horizontal axis. 21

Figure 2-5 Referential behavior between frames in a 4-layer HBP reference hierarchy. 24

Figure 2-6 Data dependency of frames in a 4-layer HBP reference scheme. 25

Figure 2-7 Accumulated percentage of frames which can be safely removed using the number of the subsequent decoded frames as the horizontal axis. 27

Figure 3-1 Block diagram of proposed new H.264 decoder..... 30

Figure 3-2 Proposed decoding pipeline. 31

Figure 3-3 The time when each frame is no longer highly utilized and the earliest time at which the frame can be finished being output for display..... 31

Figure 3-4 Partial frame data removed time of each frame. 32

Figure 3-5 Block diagram of pre-decoding mechanism. 33

Figure 3-6 Data structure of PRT table. 34

Figure 3-7 Pre-decoding and decoding time of frames. 34

Figure 3-8 Memory architecture of the modified DPB. 36

Figure 3-9 Frames stored in the CRFS using time as the horizontal axis. 37

Figure 3-10 Block diagram of Fragmented Reference Frame Storage..... 41

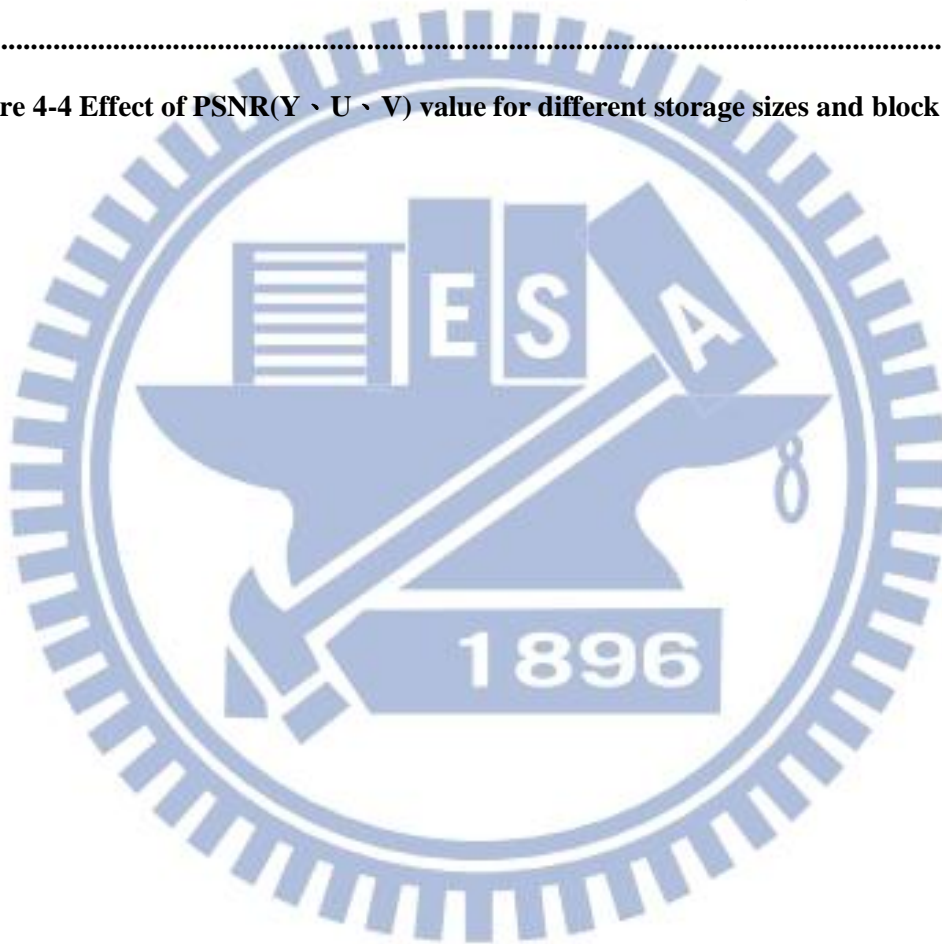
Figure 3-11 Time to remove the invalid blocks of frames in the FRFS. 42

Figure 4-1 Effect of PSNR(Luma) value for different storage sizes and block sizes. 51

Figure 4-2 Effect of PSNR(Chroma(U)) value for different storage sizes and block sizes.
..... 52

Figure 4-3 Effect of PSNR(Chroma(V)) value for different storage sizes and block sizes.
..... 52

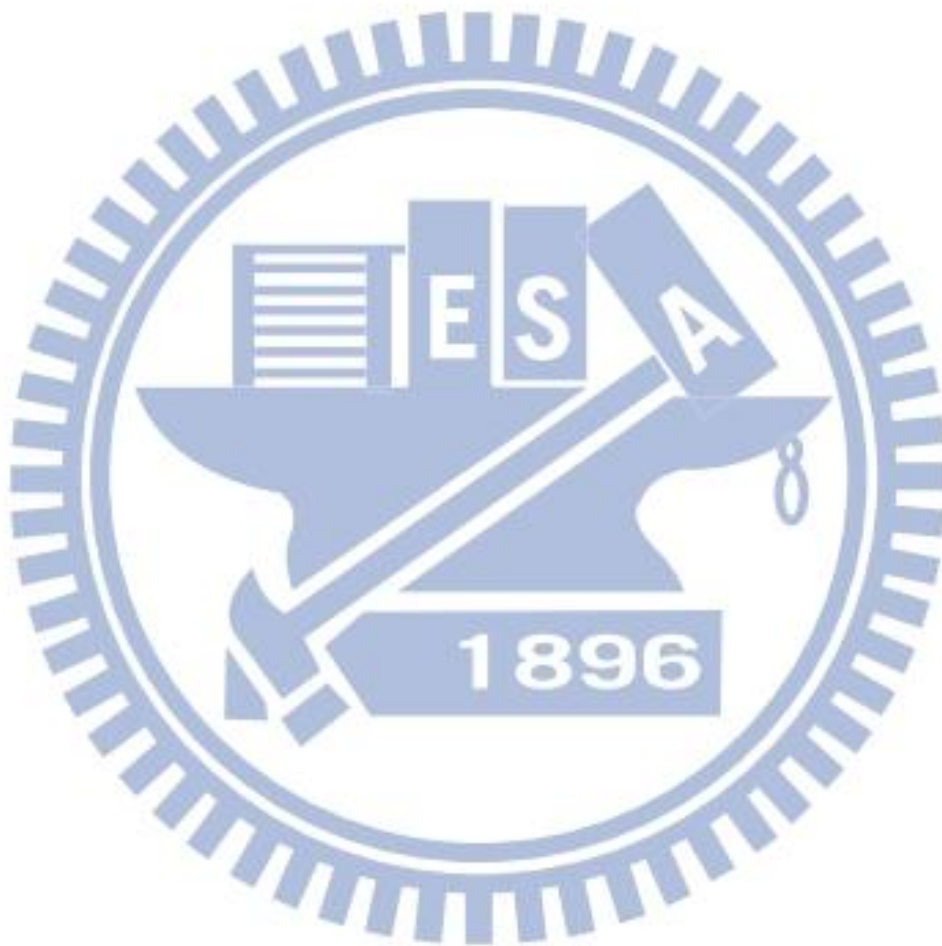
Figure 4-4 Effect of PSNR(Y、U、V) value for different storage sizes and block sizes. 59



List of table

Table 2-1 The time when the frames are no longer highly utilized for different frame types (Assume the frame sequence is “I, B1, B2, P1,...”).	7
Table 2-2 Benchmarks	8
Table 2-3 The time when the frames are no longer highly utilized for different frame types (Assume the frame is "I,B1,B2,...,Bn,P1,..."(n>0)).	22
Table 2-4 DPB status when decoding each frame in a 4-layer HBP reference scheme. .	25
Table 3-1 Required PRT table size for different block sizes in unit of storing on frame’s pixels.....	35
Table 3-2 Comparison of traditional DPB and CRFS.	37
Table 3-3 Comparison of the data structure in FRFS	40
Table 4-1 Common video sequences used in experiment.....	48
Table 4-2 Average PSNR of standard H.264 decoder	49
Table 4-3 The issues are affected by different block sizes of Modified DPB.....	49
Table 4-4 The issues are affected by different block sizes of storage overheads.....	49
Table 4-5 Storages sizes of PRT table and CRFS with luma parts for different block sizes.....	50
Table 4-6 Storages sizes of PRT table and CRFS with chroma parts for different block sizes.....	50
Table 4-7 Overall DPB sizes and storage overheads.....	53
Table 4-8 Environment parameters of CACTI	60
Table 4-9 Timing analysis of writing one encoded frame into Input Buffer.	62
Table 4-10 Access time of modified DPB and traditional DPB.	62
Table 4-11 Timing analysis of loading all pixels of the frame reaching PRT from CRFS and storing useful pixels into FRFS.	63
Table 4-12 Timing analysis of removing one invalidated frame in FRFS.	64

Table 4-13 Hardware cost of traditional H.264 decoder..... 65
Table 4-14 Hardware cost of proposed design. 65
Table 4-15 Power analysis of traditional H.264 decoder. 66
Table 4-16 Power analysis of proposed design. 66



Chapter 1 Introduction

H.264 主要目標是在提供可接受的圖片品質下，相較於之前的視訊壓縮標準能達到更好的壓縮率。除上述主要目標外，H.264 提供具有可適應性及抗錯誤能力的視訊壓縮技術，以提供廣泛的應用範圍。

Decoded Picture buffer(DPB)在 H.264 decoder 中其主要的功能是儲存已經解碼完的 frames，而這些 frames 有兩個主要功用包括：1. 須被後續解碼 frames 做參考 2. 需要被輸出到 display buffer 等待播放。而根據不同的 H.264 profiles，其所給予的 DPB sizes 亦有所不同。對於常見的 Baseline profile H.264 decoder 而言，其 DPB 總共可儲存 17 張的 frames，其中包含 16 張已經解碼完的 frames 以及 1 張目前正在解碼的 frame。另外，由[1]得知，DPB 所需的 chip area 以及記憶體空間占整個 H.264 decoder 95%以上，若可以有效減少所需的 DPB size，對於嵌入式 video decoder 而言，可大幅降低所需的 hardware costs。

[2]目前針對 Hierarchical B Picture(HBP)參考模式[3]下，提出減少 DPB size 的方法，其主要的設計概念是以 frame 為單位，移除已經不再需要被使用的 frame，以縮減所需儲存的 frames，來達到減少 DPB size 的目的。根據此方法約可省下 70.5%的儲存空間。然而[2]限制了 H.264 編碼端中 frame 間的參考方式，其所提出的 decoder 僅能接受以 HBP 參考模式下所編碼出來的 videos，才能減少所需的 DPB size。而此方法在標準 H.264 coded video 下難以獲得減少 DPB size 的好處，其主要原因是無法保證 DPB 內每一張 frame 是否可以提早移除。另外，由標準 H.264 所解碼出的 frames，其所能達到的 image quality 相較於以 HBP 參考模式下解碼出的 frames 來的更好，主要是因為 HBP 限制了每一張 frame 所能參考的對象以及數量。

相較於[2]，我們去觀察標準 H.264 中 frames 間的參考行為，發現 DPB 內的每一張 reference frame，其 life range 可被歸類成兩個時期，第一個時期是此張 frame 的 pixels 被大量參考，第二個時期為此張 frame 的 pixels 僅剩少量被參考。我們根據上述的觀察結果，對於每一張 reference frame，選擇一個適當時機，在此時機點後，僅保留住仍須被使用到的 pixels，來減少所需儲存的資料量，以達到減少 DPB size 的目標。

此篇論文剩餘的部分其組織如下。在 chapter 2 中，我們將介紹 DPB 的相關背景以及減少 DPB size 的相關研究。在 chapter 3 中，介紹我們針對減少 DPB size 所提出的相關設計。在 chapter 4 中，我們將呈現模擬結果並且分析我們所提出的設計的可行性。最後的部分為此篇論文的結論以及未來仍可能進行的工作。



Chapter 2 Background and Related work

2.1 Background

在此章節中，我們將介紹影響 DPB 內每一個 pixel 其 life range 結束的因素以及對於減少 DPB size 的相關研究。

首先，我們先定義 DPB 內每一個 pixel 的 life range 為該 pixel 自解碼完存入 DPB 開始，直到該 pixel 不再被使用為止，但仍可能儲存於 DPB 當中，此段時間稱為該 pixel 的 life range。而影響每一個 pixel 的 life range 結束的時間主要有 2 個因素，包括：1. Frames 間的參考行為、2. 每一張 frame 輸出到 display buffer 以待播放的時間。對於每一個 pixel 而言，其 life range 結束的時間取決於上述 2 點較晚發生者。而接下來的段落，將分別介紹上述 2 個因素以及 DPB 的 replacement policy。

2.1.1 Reference behaviors of frames

為了進一步分析每一個 pixel 的 life range，我們需要去了解 frame 間彼此參考的行為。而根據 H.264[4]，frame 可歸類成 3 種型態：

A. I (intra-coded)- frame:

此類型的 frame 在編解碼時並不會參考到其他的 frames，僅會參考此張 frame 內部已經解碼完的區域，故此種 frame 可獨立編解碼。另外，此類型的 frame 通常出現在 video 的第一張 frame。

B. P (predicted)-frame:

P-frame 允許參考之前已經解碼完的 frames，並且僅能參考依照播放順序之前的 frames。

C. B(Bi-directionally predicted)-frame:

B-frame 與 P-frame 皆可參考已經解碼完的 frames。然而與 P-frame 最大的不同之處在於 H.264 允許 B-frame 可依照播放順序參考前後已經解碼完的 frames。

由於 B-frame 允許前後參考其他的 frames，故 H.264 調整解碼順序來滿足 B-frame 的需求。以 Figure 2-1 為例，當播放順序為 I, B₁, B₂, ..., B_n, P₁ (n>0) 時，其解碼順序為 I, P₁, B₁, B₂, ..., B_n。P₁ 必須較 B₁, B₂, ..., B_n 先解碼完，如此一來，所有的 B-frame 才可以依照播放順序前後參考已經解碼完的 frames。

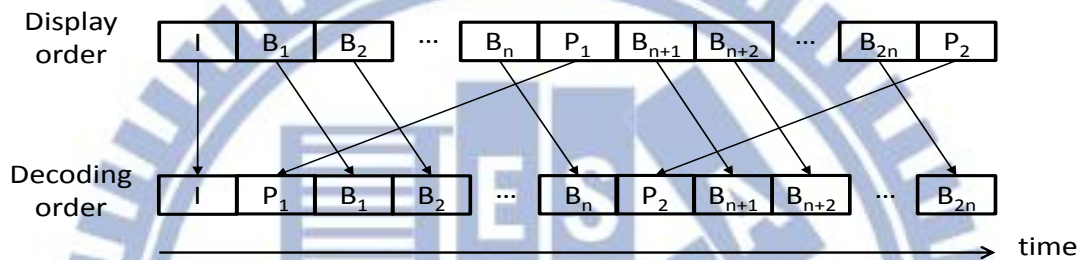


Figure 2-1 An example of display and decoding orders of a H.264 coded video.





根據標準 H.264，每一張正在編碼的 frame 允許參考 DPB 內所有的 reference frames。因此，解碼端的每一張 reference frame，最多被後續 16 張 frames 所參考。然而，基於鄰近 frame 間高度相似的性質(依照播放順序)，正在解碼的 frame 較有機會參考到最鄰近的 decoded frames。根據上述理由，我們觀察出每一張 reference frame，其 life range 可被歸類成兩個時期，第一個時期是此張 frame 中有大量的 pixels 仍須被參考(highly utilized)，第二個時期為此張 frame 中僅剩少量的 pixels 仍須被參考(lightly utilized)。在此我們先定義何謂大量 pixels 被參考的時期以及僅剩少量 pixels 被參考的時期。

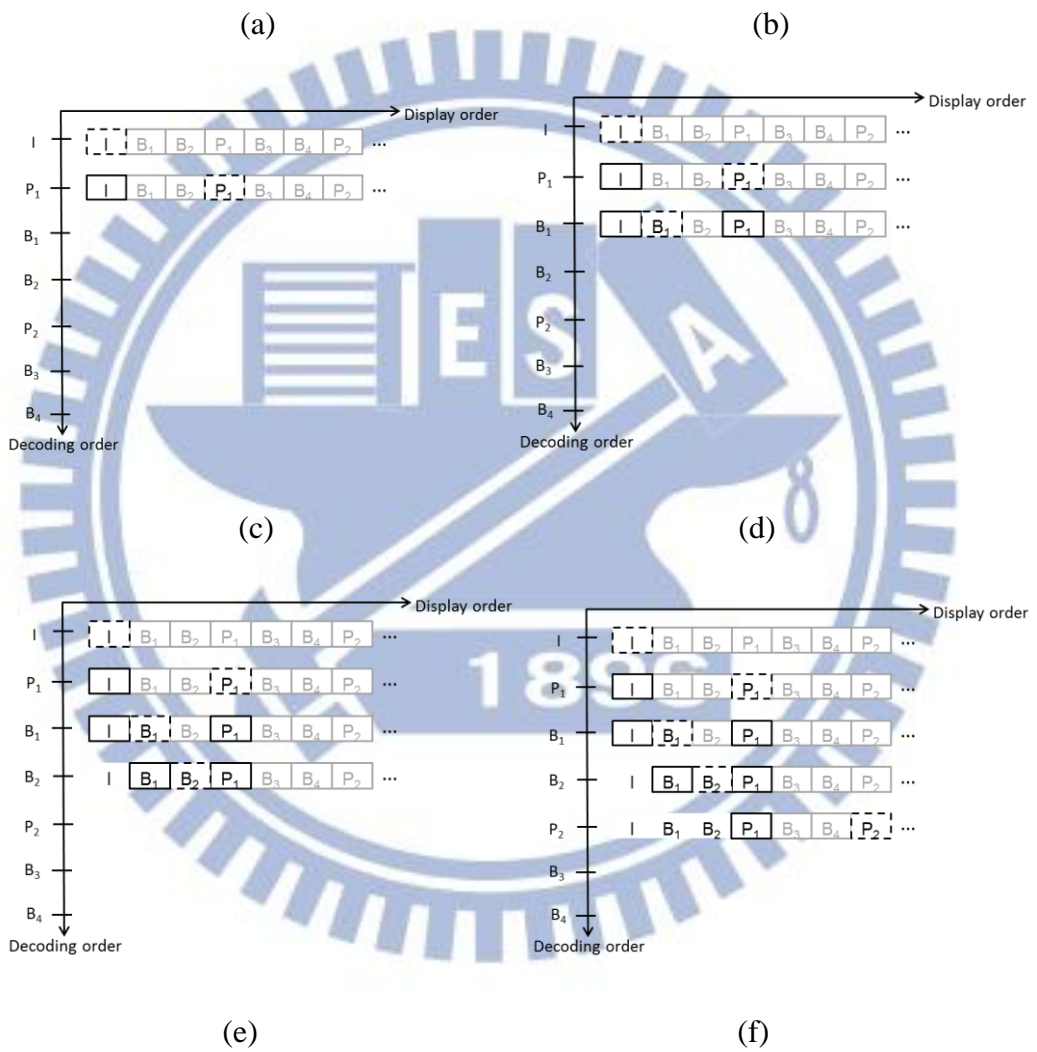
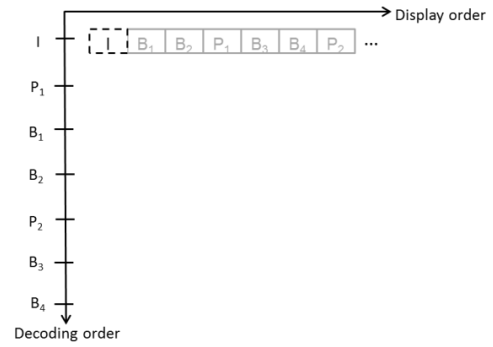
- A. 大量 pixels 被參考的時期(highly utilized): 當此張 frame 已經解碼完存入 DPB 中，且是目前正在解碼 frame 前後最鄰近的 frame(依照 display order)，我們則認定此張 frame 中的 pixels 此時期正在被大量參考。

B. 僅剩少量 pixels 被參考的時期(lightly utilized): 當此張 frame 已經解碼完存入 DPB 中，已經不是目前正在解碼 frame 前後最鄰近的 frame(依照 display order)，我們則認定此張 frame 中的 pixels 此時期僅剩少量被參考。

我們將由下兩段分別說明每一張 frame，何時 pixels 被大量參考，何時僅剩少量 pixels 仍須被參考，並且以統計結果來驗證我們的推測是合理的。

而此段我們將根據不同的 frame 型態(I、P、B)，推論每一張 frame 中，大量 pixels 仍須被參考的時間以及僅剩少量 pixels 仍須被參考的時間。以 Figure 2-2 所示，我們以 I, B₁, B₂, P₁...作為舉例，且此例子具有代表性。Figure 2-2 (a)用來表示每一張 frame 目前的狀態，分別代表著尚未解碼、目前正在解碼、大量 pixels 被參考、以及僅剩少量 pixels 被參考。Figure 2-2 (b)~ Figure 2-2 (h)是以解碼順序來說明每一張 frame 目前的狀態。我們以 I-frame 作為主要說明的對象，Figure 2-2 (b)顯示著目前正在解碼的 frame 為 I-frame，而其他 frame 尚未解碼。Figure 2-2 (c)顯示著目前正在解碼的 frame 為 P₁，而 P₁ 會大量參考 I-frame 中的 pixels，因為 I-frame 為 P₁ 最鄰近的可參考圖片。Figure 2-2 (d)顯示著目前正在解碼的 frame 為 B₁，由於 B₁ 允許前後參考，而按照撥放順序最鄰近且可參考的 frame 為 I frame 以及 P₁，故 B₁ 會大量參考此 2 張 frame 的 pixels。Figure 2-2 (e)顯示著目前正在解碼的 frame 為 B₂，由於按照撥放順序最鄰近且可參考的 frame 為 B₁ 以及 P₁，故 B₂ 會大量參考此 2 張 frame，而此時 I-frame 已經不再是目前解碼圖片最鄰近的 frame，因此 I-frame 僅剩少量 pixels 會被後續解碼圖片所參考。根據上述推論，即可發現 I-frame 中的 pixels 會被後續的 2 張解碼 frame 大量參考，而後就僅剩少量 pixels 會被參考。而其他 frame 也可按照此規則類推，得出大量 pixels 被參考的時期以及僅剩少量 pixels 仍須被參考的時期。

-  Frame : Currently decoded frame
-  Frame : The frame is highly utilized
-  Frame : The frame is not decoded yet
-  Frame : The frame is lightly utilized



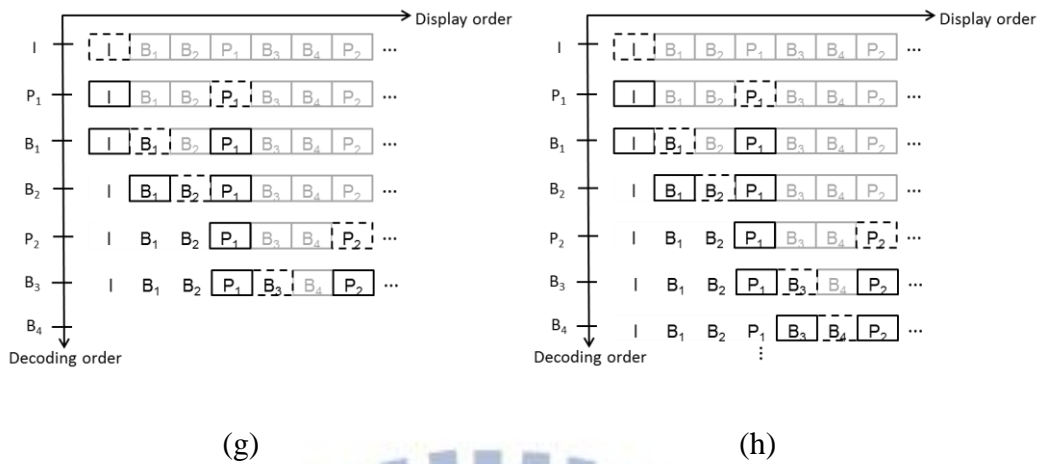


Figure 2-2 An example of status of reference frame in the DPB when each frame decoded.

根據上述推論，我們可以根據不同 frame 型態，歸納出每一張 frame 中 pixels 不再被大量參考的時機，由 Table 2-1 所示。

Table 2-1 The time when the frames are no longer highly utilized for different frame types (Assume the frame sequence is “I, B1, B2, P1,...”).

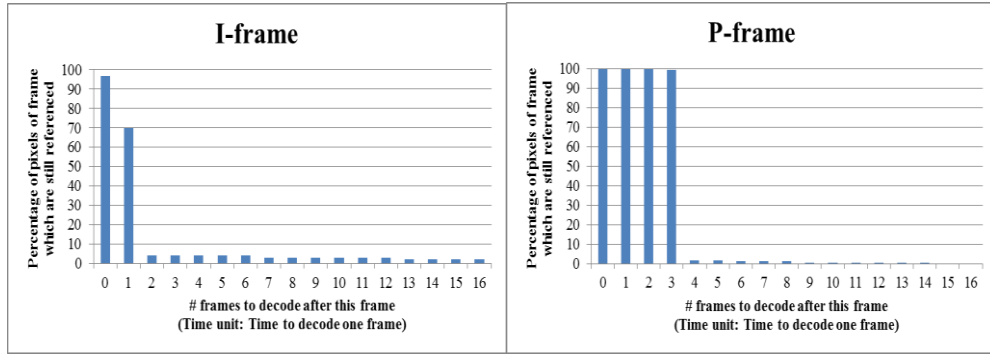
Frame 型態	此張frame的pixels不再被大量參考的時機 (在此張frame解碼後，有多少張frame被解碼)
I frame	2
P frame	4
B ₁ frame	1
B ₂ frame	0

為了驗證上述所推論每一張 frame 的 pixels 不再被大量參考的時機，我們針對常用的 benchmarks 進行了測試，其 benchmarks 如 Table 2-2 所示。主要去觀察每一張 frame，從解碼完存入 DPB 開始，直到該張 frame 而被其他新進的 frame 取代為止，統計在每一個時間點(時間單位:解碼一張 frame 的時間)後還有多少比例的 pixels 仍須被參考。Figure 2-3(a)~ Figure 2-3 (d)是以 Bridge-close 作為例子，分別根據不同的 frame 型態，統計每一個時機點後仍需被參考的 pixels 比例。我

們以 I-frame 作為主要說明的對象，可發現當 I-frame 解碼完並且在解碼後續 2 張 frame 之後，I-frame 內仍須使用 pixels 的比例就會大幅降低，與我們所推論 I-frame 的 pixels 不再被大量參考的時機吻合。而其他 frame 亦同。Figure 2-4(a)~ Figure 2-4 (x)列出其他 benchmarks 的統計結果。

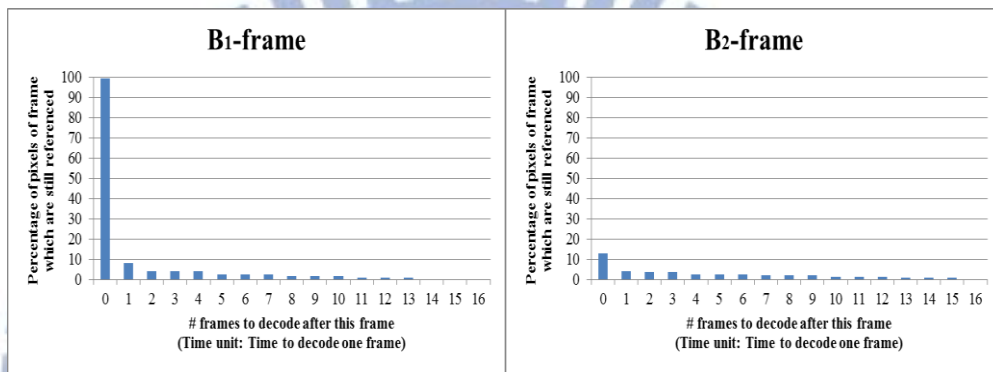
Table 2-2 Benchmarks

Benchmarks	Resolutions	Number of frames
Carphone	QCIF	100
Crew	QCIF、CIF	100
Ice	QCIF、CIF	100
Foreman	QCIF	100
Highway	QCIF	100
City	QCIF、CIF	100
Mother-daughter	QCIF	100
Walk	QCIF	100
Suzie	QCIF	100
Coastguard	QCIF	100
Bridge-close	QCIF	100
Claire	QCIF	100
Container	QCIF	100
Grandma	QCIF	100
Hall	QCIF	100
Miss-america	QCIF	100
Silent	QCIF	100
Trevor	QCIF	100
Akiyo	QCIF	100
Salesman	QCIF	100
News	QCIF	100
Bride-far	QCIF	100



(a)

(b)

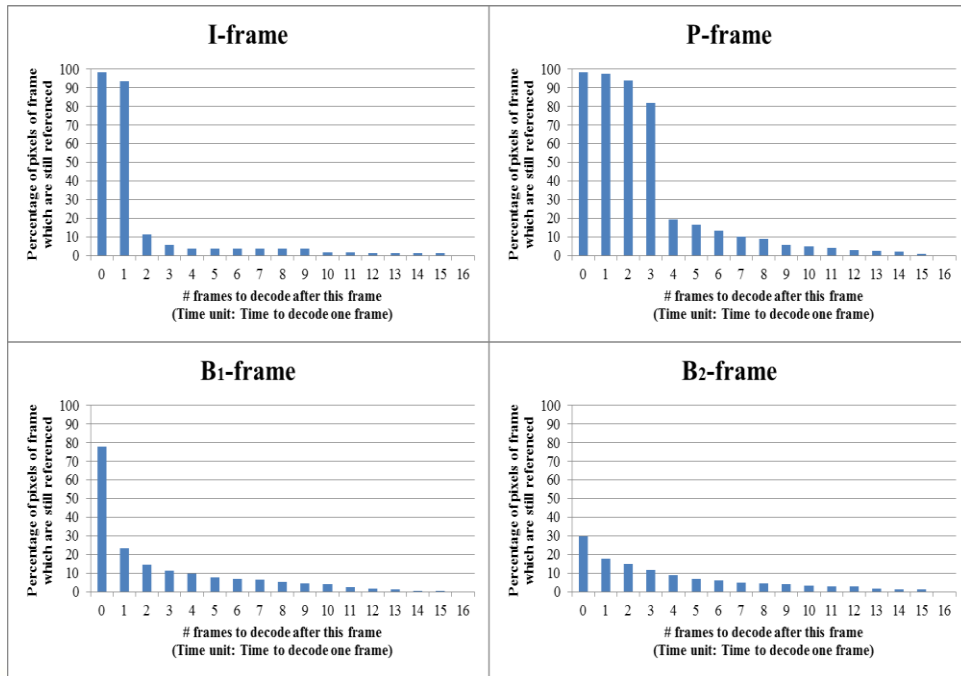


(c)

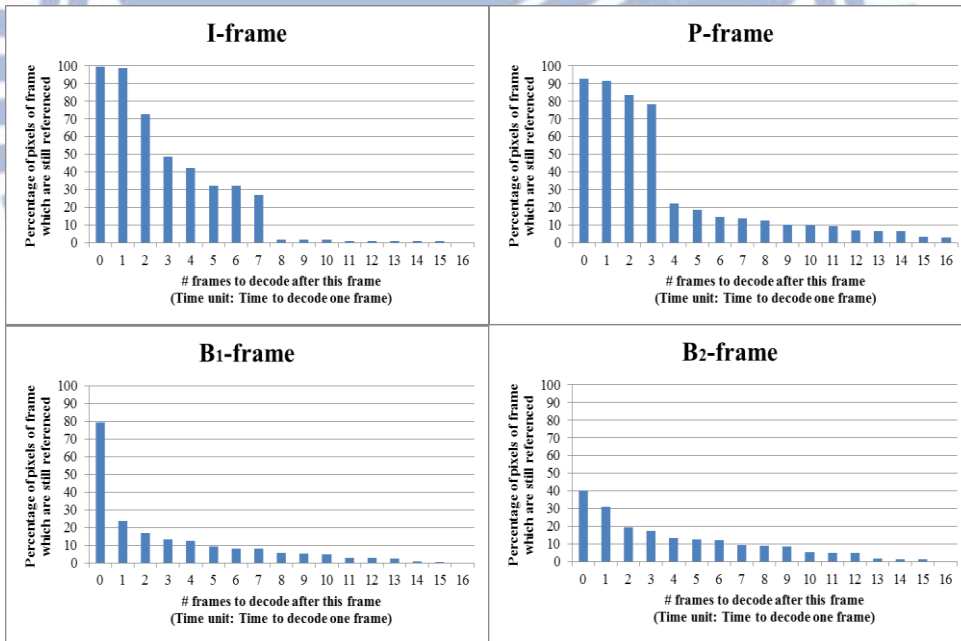
(d)

Benchmark: Bridge-close

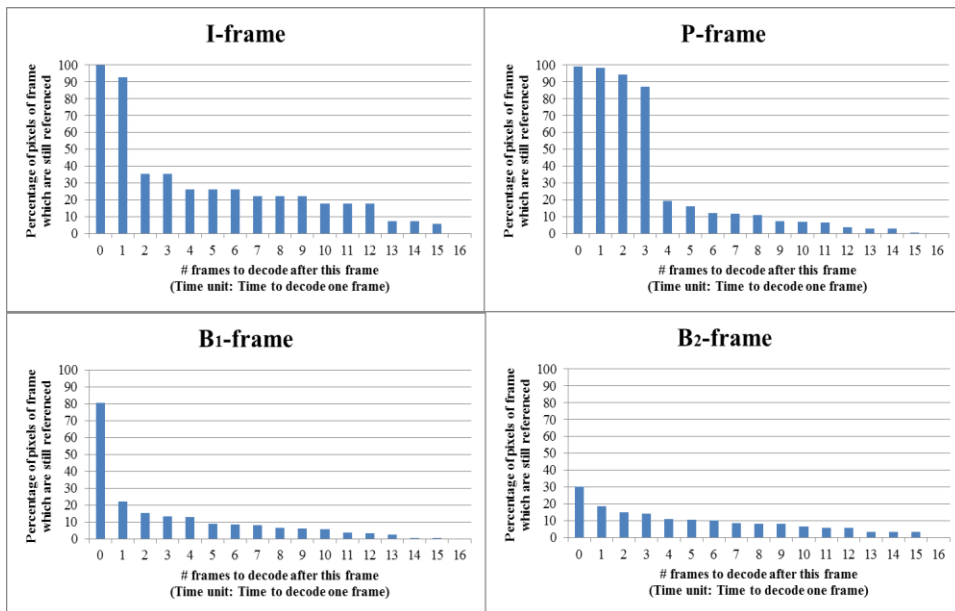
Figure 2-3 (a) The percentage of pixels of I frame which are still referenced using the number of the subsequent decoded frames as the horizontal axis. (b) The percentage of pixels of P frame which are still referenced using the number of the subsequent decoded frames as the horizontal axis. (c) The percentage of pixels of B₁ frame which are still referenced using the number of the subsequent decoded frames as the horizontal axis. (d) The percentage of pixels of B₂ frame which are still referenced using the number of the subsequent decoded frames as the horizontal axis.



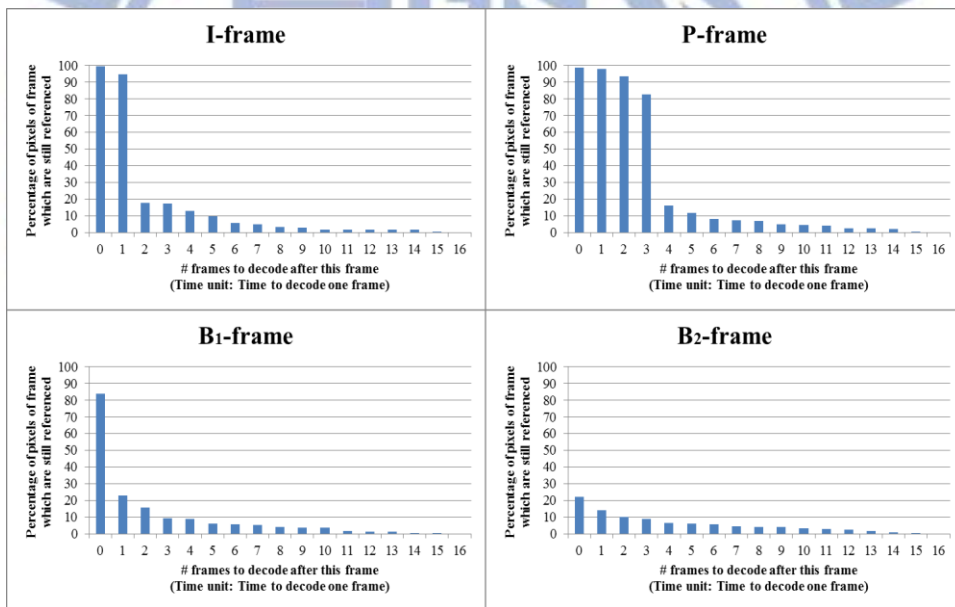
(a)Carphone



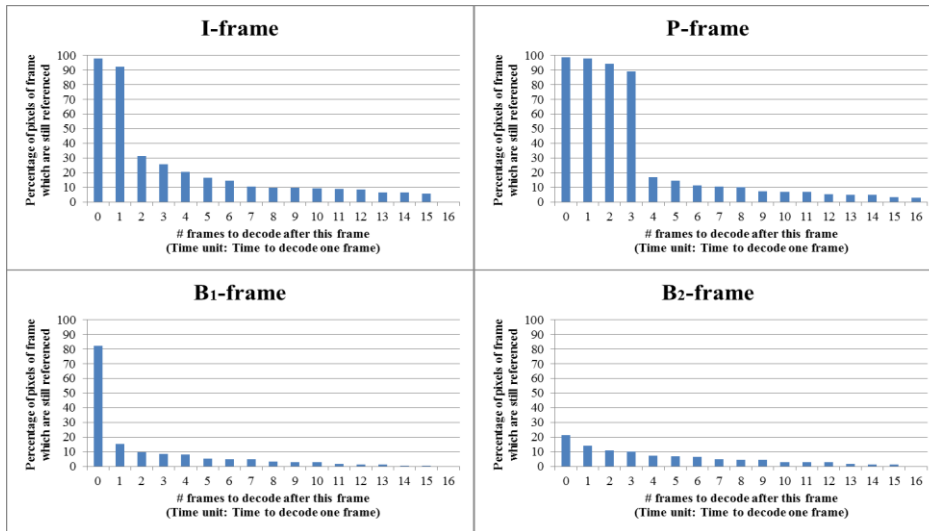
(b)Crew



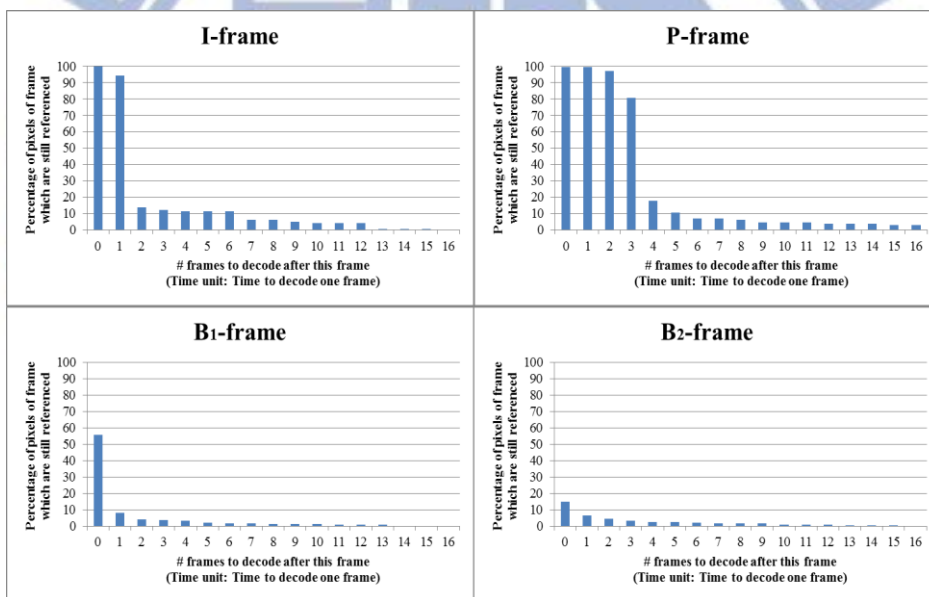
(c)Foreman



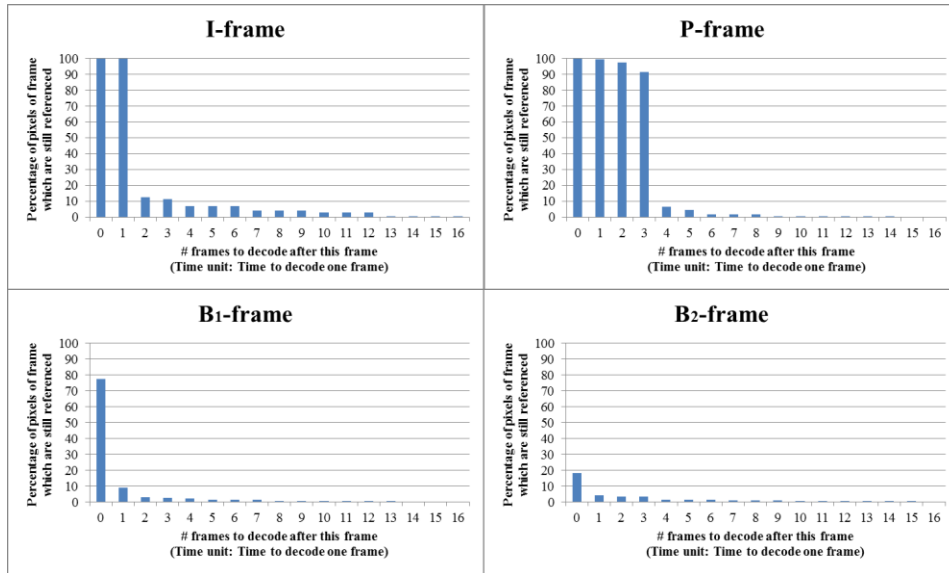
(d)Highway



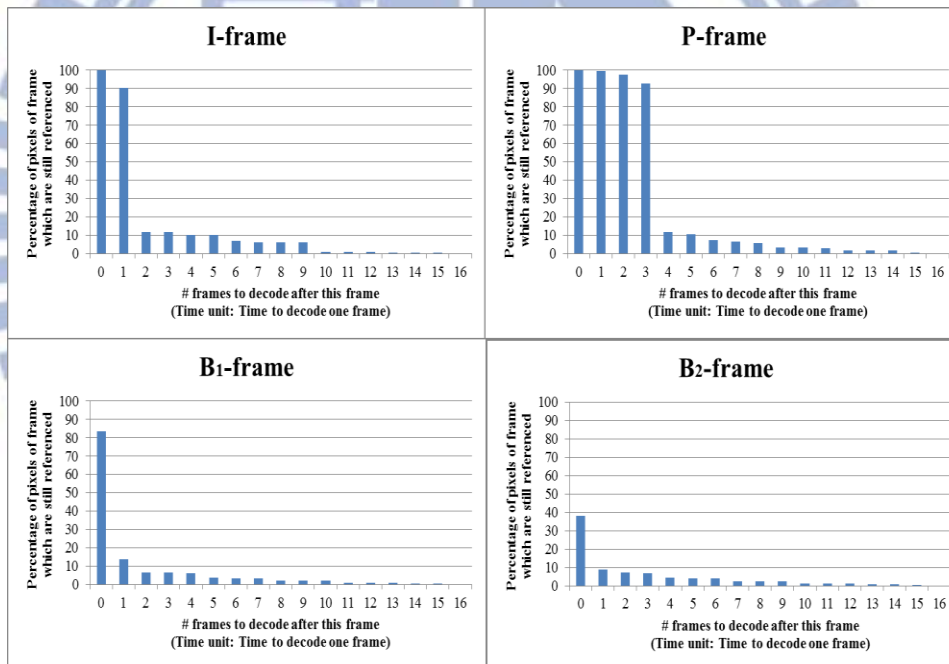
(e)Ice



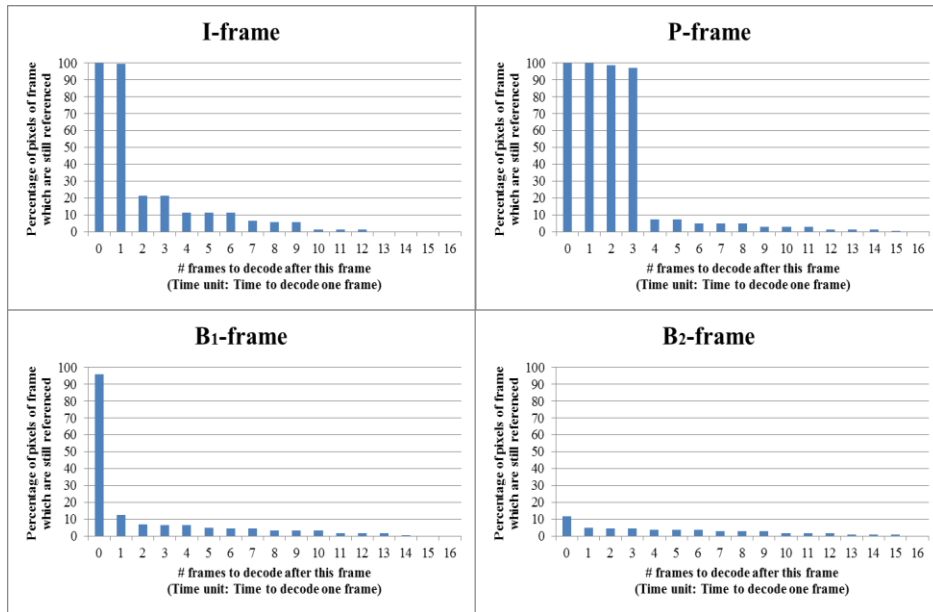
(f)City



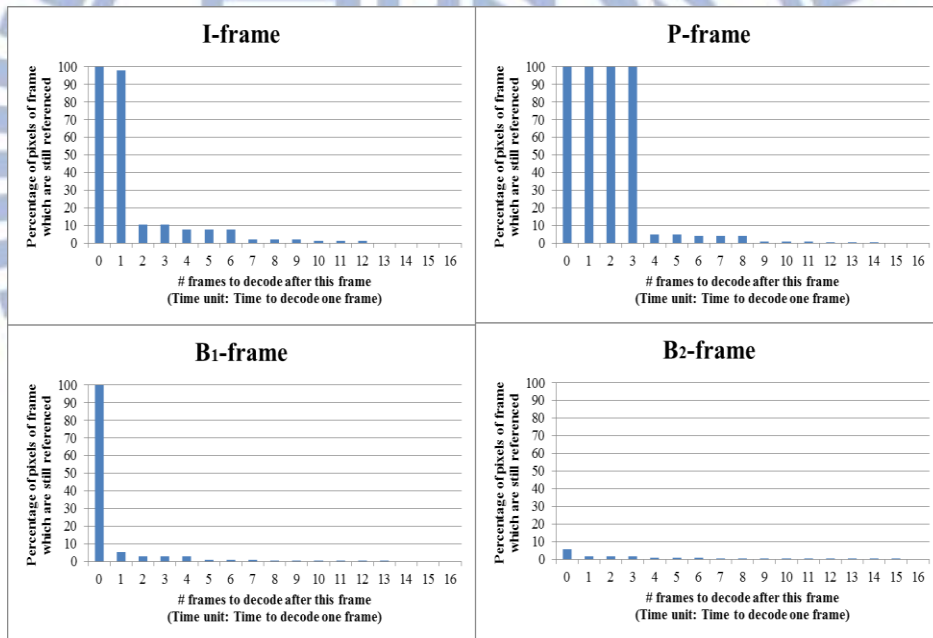
(g) Suzie



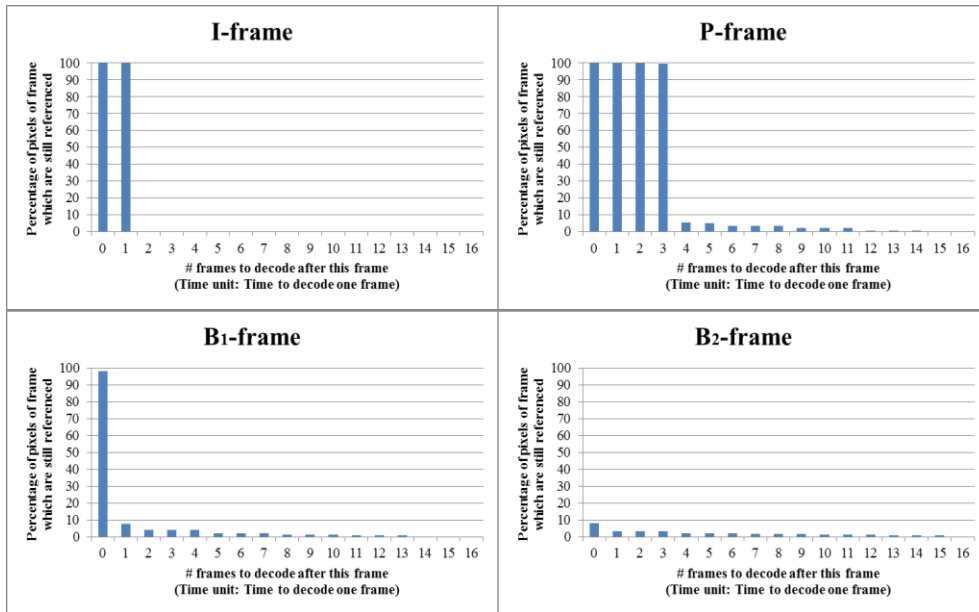
(h) Coastguard



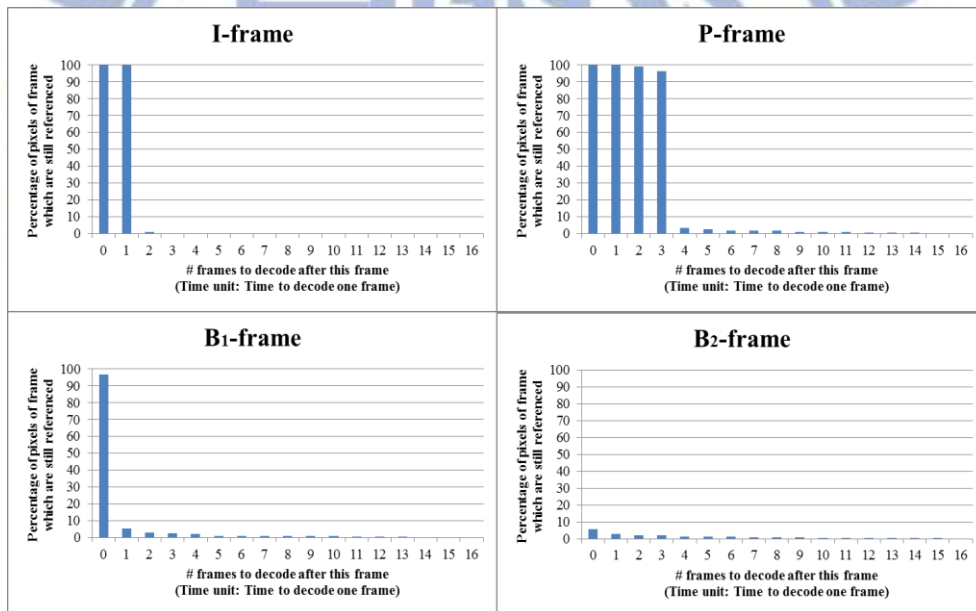
(i) Mother-daughter



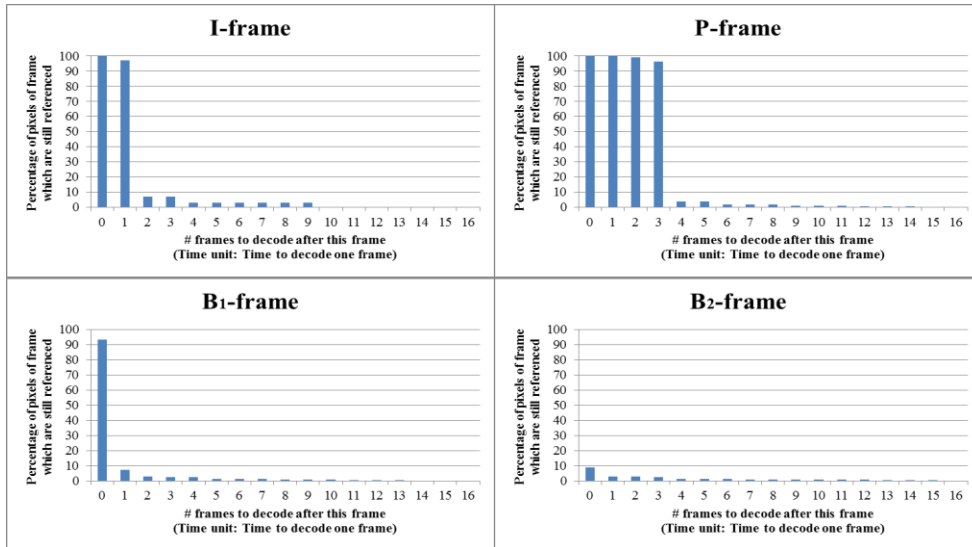
(j) Container



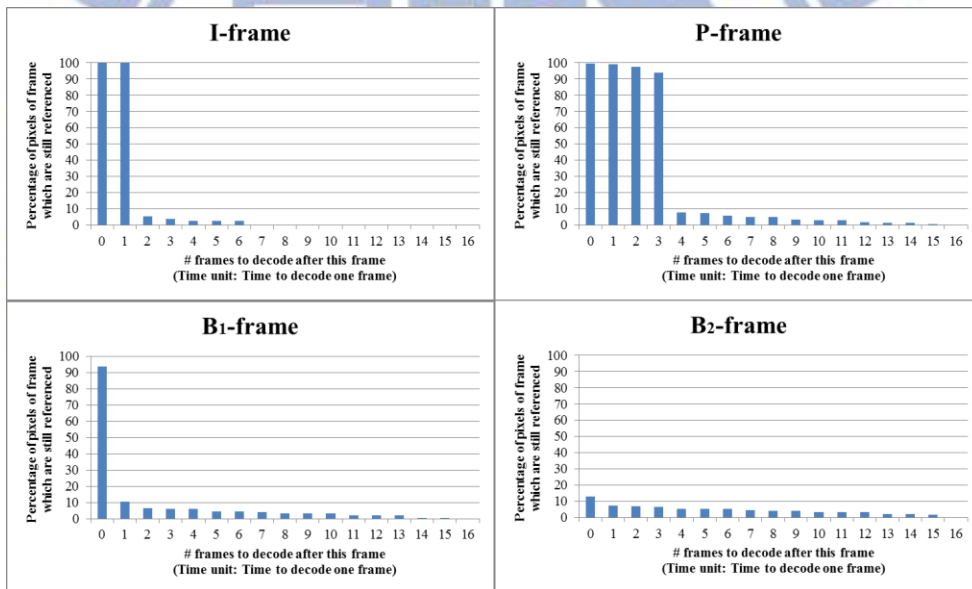
(k)Grandma



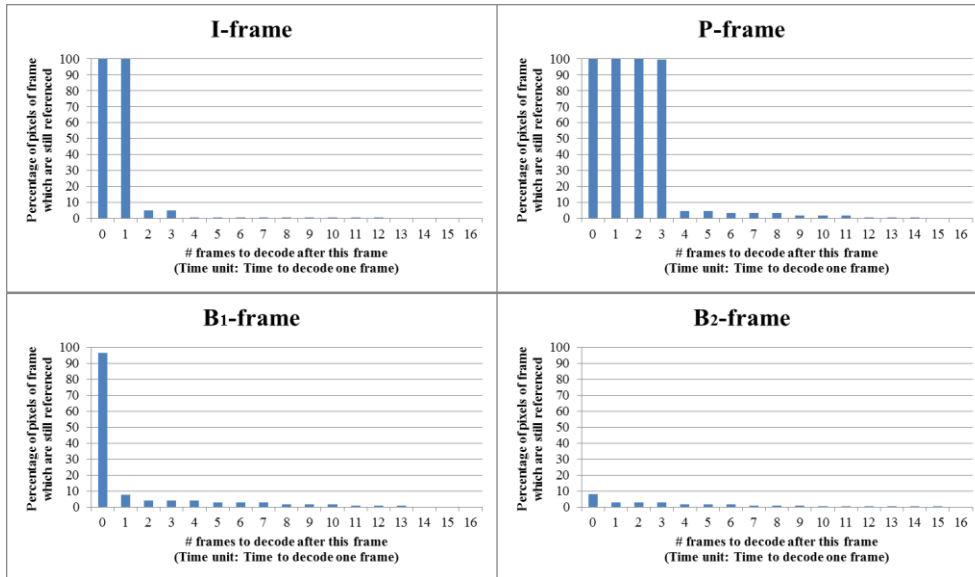
(l)Hall



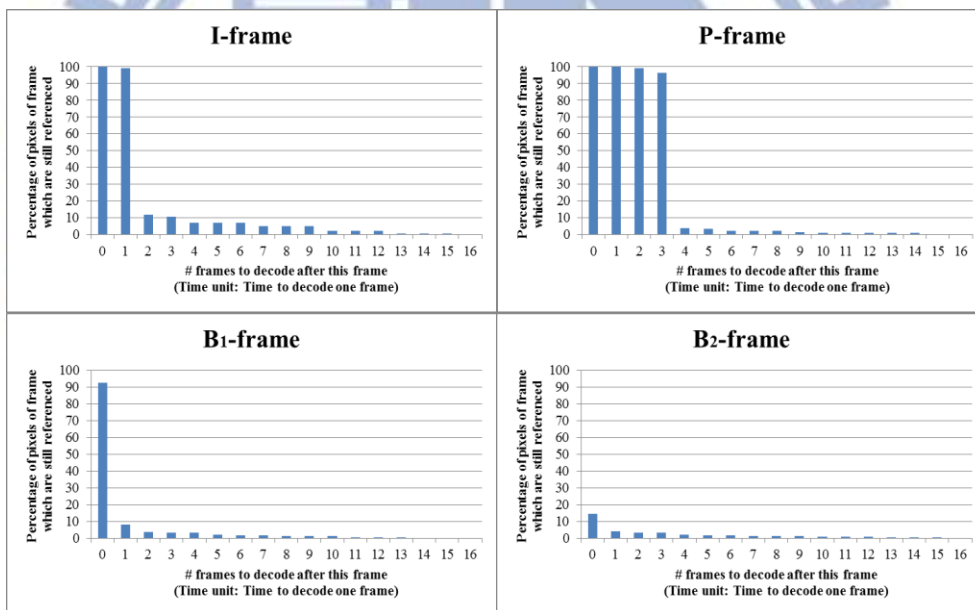
(m)Miss-america



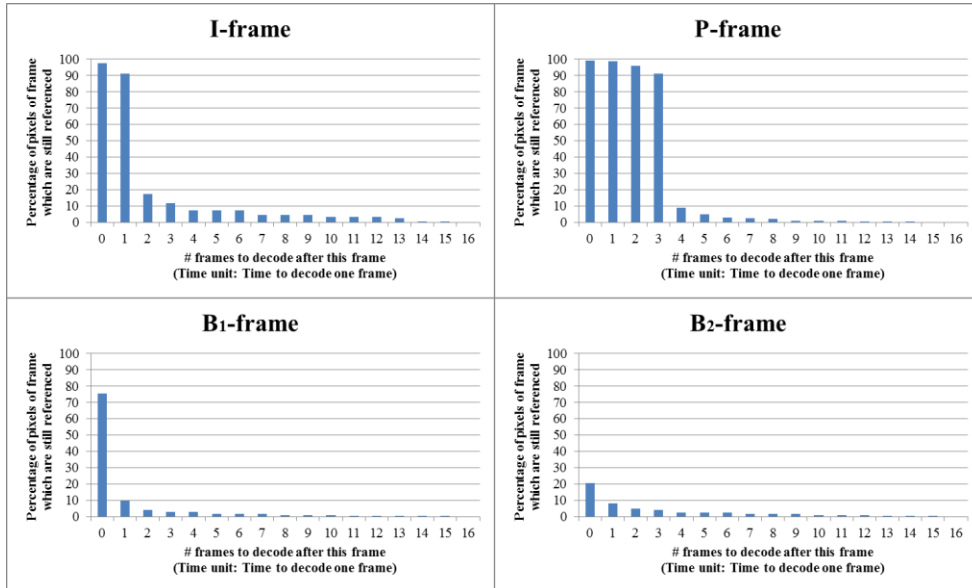
(n)Silent



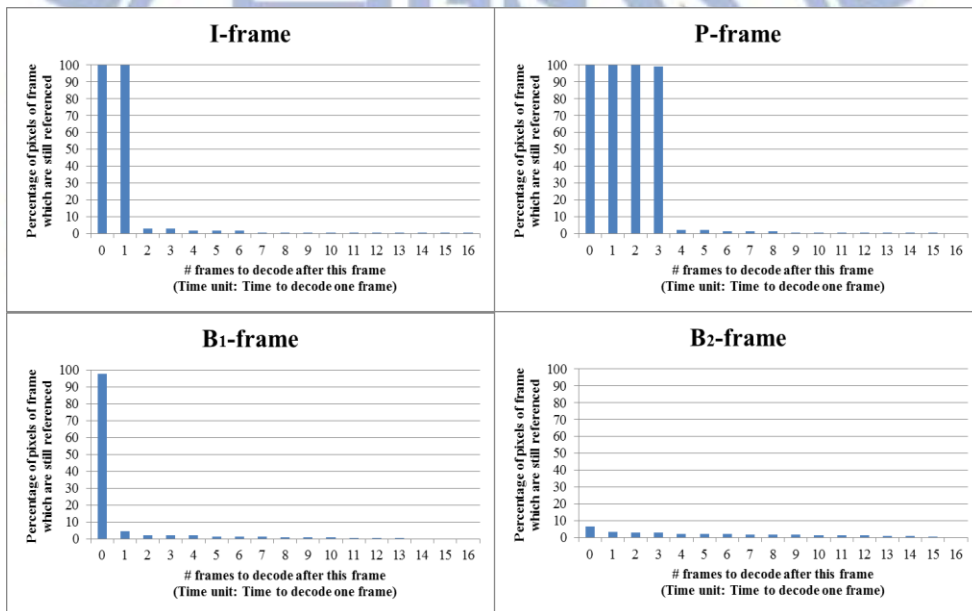
(o) Claire



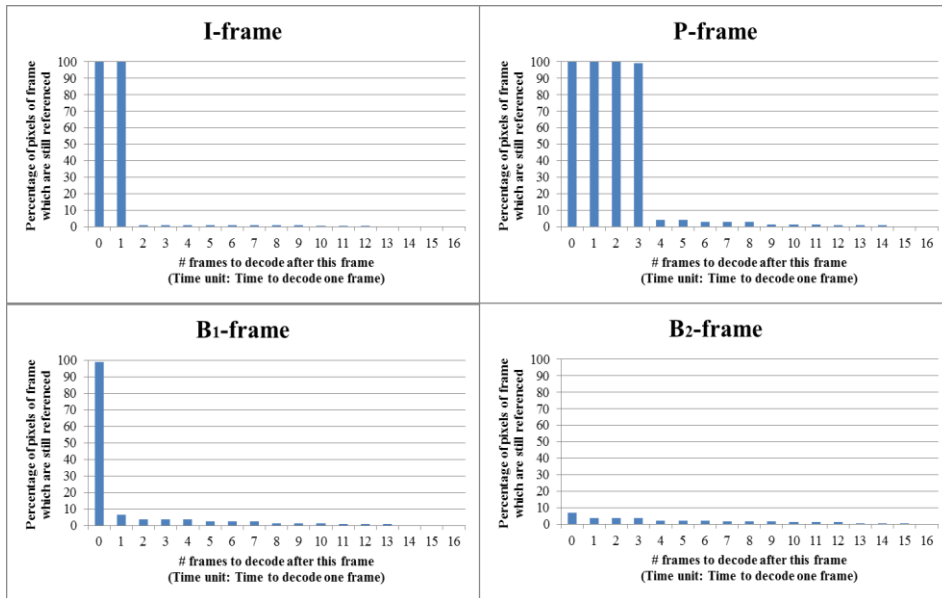
(p) Trevor



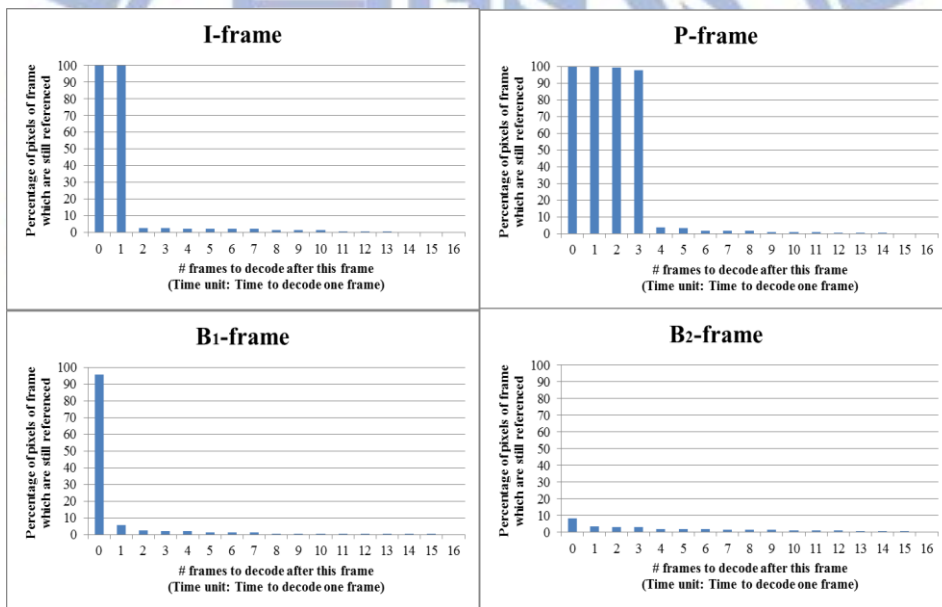
(q)Walk



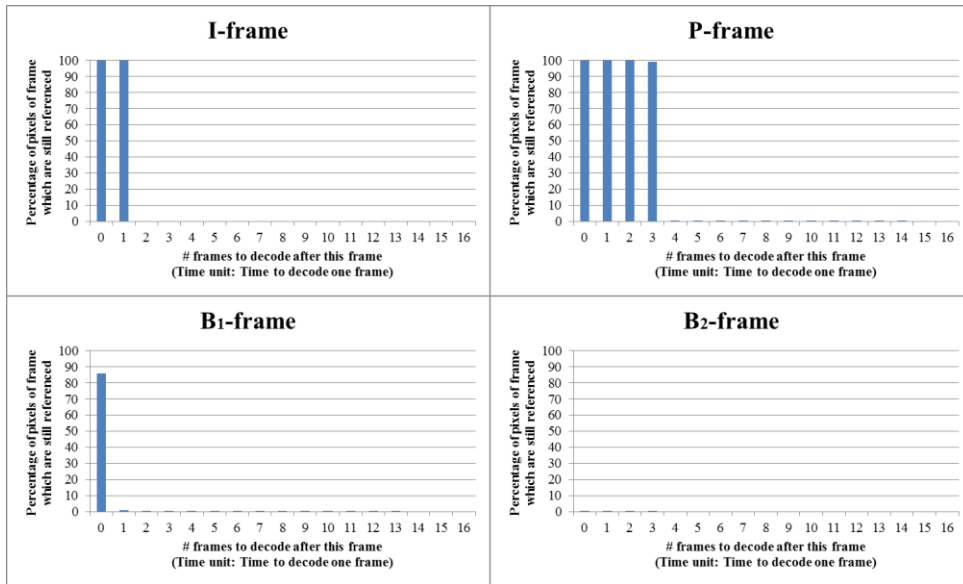
(r)Akiyo



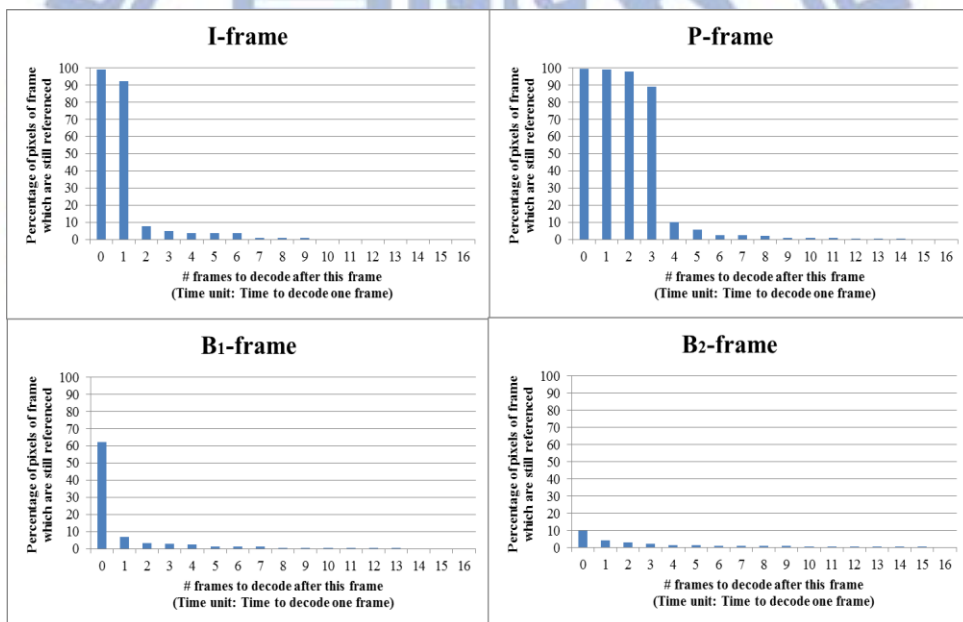
(s)Salesman



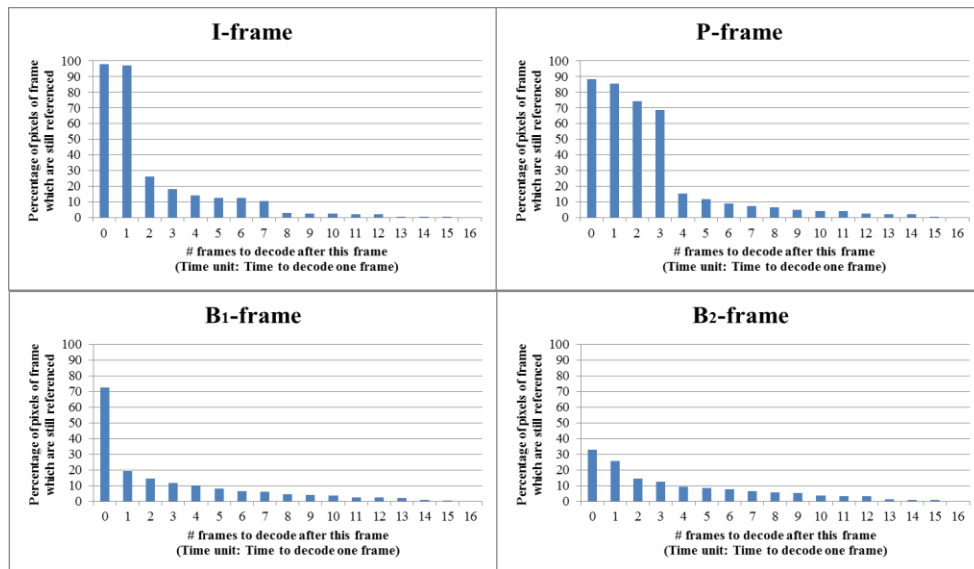
(t)News



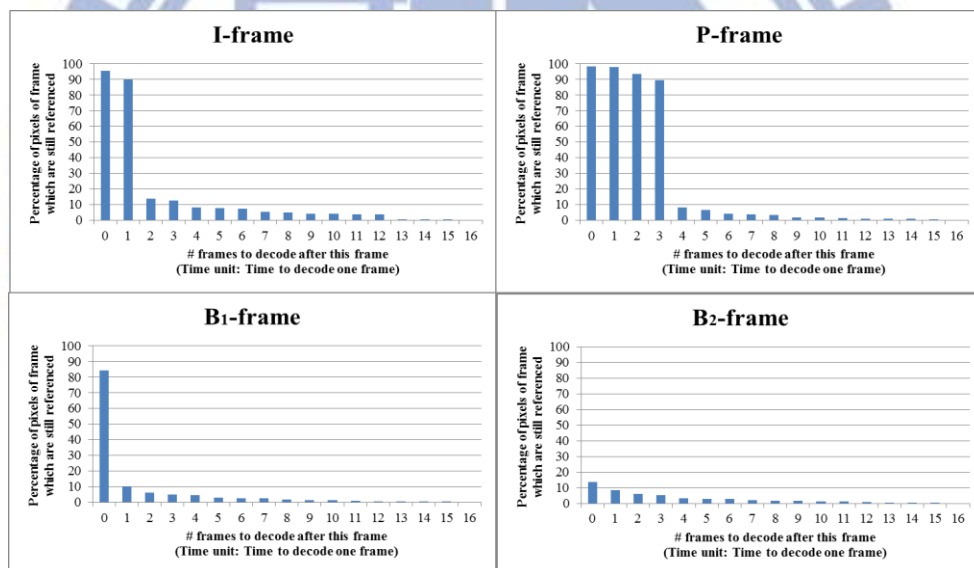
(u)Bridge-far



(v)City(CIF)



(w)Crew(CIF)



(x)Ice(CIF)

Figure 2-4 The percentage of pixels of frame of different videos which are still referenced using the number of the subsequent decoded frames as the horizontal axis.

根據上述統計結果，可發現大部分的 benchmarks 都會有固定的時間點，會由大部分 pixels 仍須被參考轉變成僅剩少量 pixels 仍須被參考。並且可以根據不同的 frame type 進行歸類。另外，可發現每個 video 中當每一張 frame 不再 highly utilized 之後，其仍須被參考的 pixels 數量不盡相同。我們根據每一個 video 中

pixels lifetime 的分布情形發現，frame 間場景移動較多或變化較快的 video(ex: Carphone、Crew、Ice、Foreman、Highway...)，當每一張 frame 不再被 highly utilized 之後，仍須被參考的 pixels 數量相較於其它 videos 多。而會發生此現象的原因就是因為變化較快的 videos，通常會因為鄰近 frame 變化太多，而參考到較遠的 frames，因此可能造成某些 frames 被較遠的 frames 做參考，而拉長了該張 frame 內 pixels 的 lifetime。而對於場景變化較少的 video(ex: Bridge-close、Akiyo、News、Bridge-far..)而言，其內 pixels 的 lifetime 較短，因此當該張 frame 不再被 highly utilized 之後，仍須被參考的 pixels 數量是較少的。

另外，針對相同 video 不同解析度的 frame，我們也進行測試(City、Crew、Ice)，可發現兩種不同解析度的 video，其內的每一張 frame 的 pixels life time 分布大致相同，因此可觀察出影響 pixels lifetime 長短的因素，與 video 的性質仍較有關連。

而當 video sequence 為 I, B₁, B₂,..., B_n, P₁...時(n>0)，根據不同 frame 型態，所推得每一張 frame 的 pixels 不再被大量參考的時機如下表 2-2 所示。

Table 2-3 The time when the frames are no longer highly utilized for different frame types (Assume the frame is "I,B₁,B₂,...,B_n,P₁,..."(n>0)).

Frame 型態	此張frame的pixels不再被大量參考的時機 (在此張frame解碼後，有多少張frame被解碼)
I frame	2
P frame	n+2
B ₁ ~B _{n-1} frame	1
B _n frame	0

對於 frame 間彼此的參考行為，我們在此做個總結。雖然 H.264 給予 DPB 內所有 frame 皆可做為參考對象的彈性，但由於鄰近 frame 間高度相似之特性，實際上 frame 在做解碼時，仍較有機會參考到最鄰近且已解碼的 frames。而根據此

性質，我們可發現每一張 frame 中 pixels 被大量參考的時期以及僅剩少量 pixels 被參考的時期。而此觀察結果可作為我們選擇何時移除每一張 frame 不再使用到 pixels 的依據。

2.1.2 Output time of frames for display

每一張 frame 的每一個 pixel 其 life range 結束的時間不僅取決於最後一次被參考的時間，亦須考慮該張 frame output for display 的時間。而 H.264 並沒有規範每一張 frame 何時須 output for display，因此只要維持 frame 輸出之順序與播放順序的一致性，則每一張 frame 其輸出的時間是可以做調整的。

2.1.3 Replacement Policy of the DPB

當 DPB 中存滿 frames 時，H.264 通常以 FIFO(先進先出)將最早進入 DPB 的 frame 給取代掉。因此，若以 DPB 可以儲存 17 張 frames 為例，每一張 frame 的 life range 最多不超過後續解碼 16 張 frames 的時間。

2.2 Related work

對於減少 DPB size 的相關研究中，[2]針對 Hierarchical B Picture(HBP)[3]參考模式提出一個方法來減少 DPB size。而此方法的主要設計概念是以 frame 為單位，當該張 frame 的 life range 結束時，就將該張 frame 從 DPB 中移除，以減少 DPB 所需儲存的資料量，以達到減少 DPB size 的目標。接下來我們將分別介紹 Hierarchical B Picture 參考模式以及[2]針對 HBP 參考模式下所提出減少 DPB size 的方法。

2.2.1 Hierarchical B Pictures reference scheme

有別於標準 H.264 提供 DPB 內所有 frames 皆可被其他 frame 參考的彈性，Hierarchical B Picture 參考模式限制了 frame 間彼此的參考行為，並且將 frames

依重要性而分層，越底層的 frames 其重要性越高。以下我們舉 4 層的 HBP 參考模式為例，如 Figure 2-5 所示。其中最底層是以 I-frame 或 P-frame 所組成，而其他層則由 B-frame 構成。對於所有 B-frame 而言，其所能參考的對象為最鄰近(依照播放順序)且階層數較低的 frames，如 B_3 僅能參考 B_2 以及 B_4 ，而 P-frame 僅能往前參考同一層最鄰近的 frame。而越底層的 frames 其重要性越高，主要原因在於被越多的 frame 所參考，而最上層的 frames 皆不會被其他 frames 所參考。

HBP 參考模式的層數可提供彈性做調整，而影響的將是最終所得到的 video quality 以及 H.264 encoder 所編出來的 bit 數。根據我們對於不同 HBP layers 的統計發現，在固定編碼端所壓縮出來的 bit 數的前提下，其 4-layer HBP 參考模式相較於其他 layer 的 HBP 所能達到較佳的 video quality，因此，在後續章節中，我們將以 4-layer HBP 參考模式作為舉例以及參考的對象。

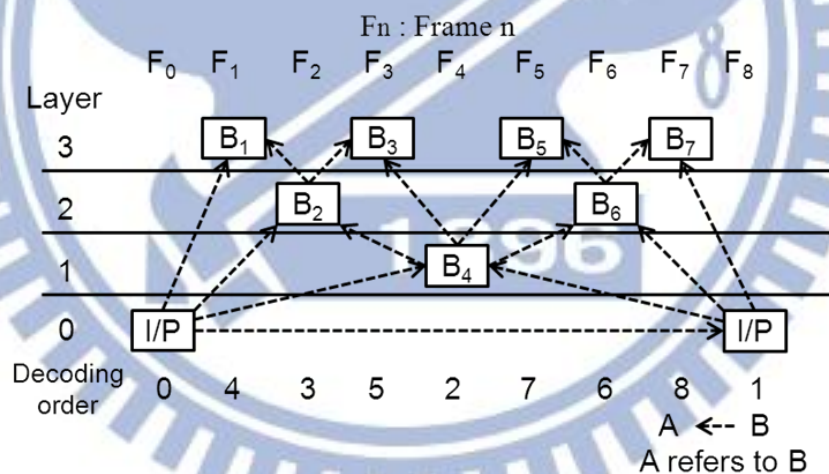


Figure 2-5 Referential behavior between frames in a 4-layer HBP reference hierarchy.

2.2.2 Related work

藉由分析 HBP 參考模式下 frame 間彼此參考的行為，可觀察出每一張 frame 的 life range，以 Figure 2-6 表示 frame 間 data dependency 的關係。可發現 I-frame

需等到 B1 解碼完之後其 life range 就會結束，而其他 frames 的 life range 結束時間亦可得知。另外，由 Figure 2-5 可發現每當最上層的 frames 解碼完成後，其按照播放順序前一張 frame 的 life range 也一併結束。

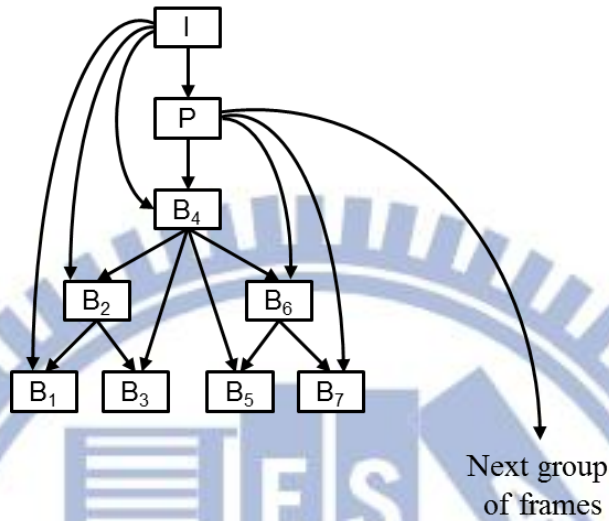


Figure 2-6 Data dependency of frames in a 4-layer HBP reference scheme.

根據每一張 frame 其 life range 結束的時間，此篇 work 提出以 frame 為單位，移除其 life range 已經結束的 frames，來達到減少 DPB size 的目標。Table 2-4 為每解碼完一張 frame 後，DPB 目前儲存那些 frames 以及那些 frame 的 life range 已經結束可以被移除。以 4-layer HBP 參考模式為例，在解碼正確性的情況下，DPB 僅需保留住 5 張 frames 的空間即可。而當 HBP 參考模式為 N 層時，其 DPB 需要保留 N+1 張 frames 的空間。

Table 2-4 DPB status when decoding each frame in a 4-layer HBP reference scheme.

Display Number	Used for reference	Frames to be removed	Stored frames in the DPB after decoding the frame				
0	Yes		0				
8	Yes		0	8			
4	Yes		0	8	4		
2	Yes		0	8	4	2	
1	No	0,1	0	8	4	2	1
3	No	2,3	8	4	2	3	
6	Yes		8	4	6		
5	No	4,5	8	4	6	5	
7	No	6,7	8	6	7		
16	Yes		8	16			
12	Yes		8	16	12		
10	Yes		8	16	12	10	
9	No	8,9	8	16	12	10	9
11	No	10,11	16	12	10	11	
14	Yes		16	12	14		
13	No	12,13	16	12	14	13	
15	No	14,15	16	14	15		
.....

Decoding order ↓

2.2.3 Summary of related work

根據此篇 work 所提出之方法，當 HBP 參考模式為 N 層時，可將 DPB 縮減至保留 N+1 張 frames 的空間即可，以 4-layer HBP 參考模式為例，可將 DPB 由原始需保留 17 張 frames 縮減至僅需保留 5 張 frames 的空間，約可省下 70.5% 的儲存空間。

然而此篇 work 必須限制 H.264 encoder 以 HBP 參考模式做編碼，其所設計的 decoder 才可以得到減少 DPB size 的好處。另外，以 HBP 參考模式所編碼出的 videos，相較於標準 H.264 encoder 而言，在相同 bit rate 的情況下，標準 H.264 所能達到最佳的 image quality，主要是因為 HBP 限制了每一張 frame 所能參考的對象以及數量。

若將此方法應用在標準 H.264 decoder 上，其所能減少 DPB size 的機會相對較小，而主要是因為此篇 work 是以 frame 作為移除的基本單位。根據我們對於標準 H.264，統計所測試的 benchmarks 內每一張 frame 其 life range 的結果，如 Figure 2-7 所示，可發現難以保證有機會提早移除不再使用到的 frames。

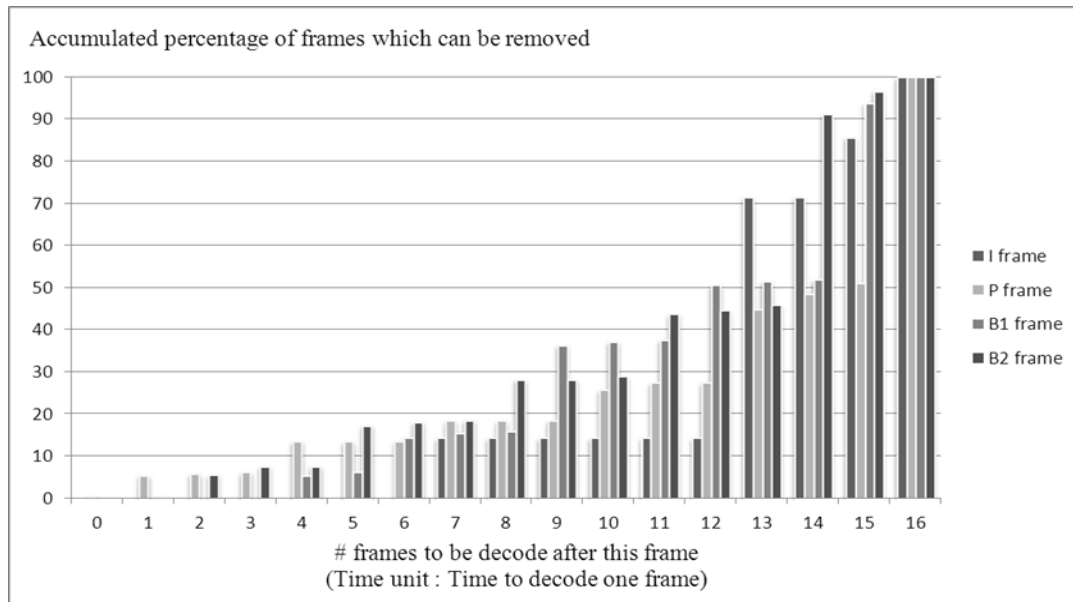


Figure 2-7 Accumulated percentage of frames which can be safely removed using the number of the subsequent decoded frames as the horizontal axis.

2.3 Opportunity

根據 Figure 2-7 顯示，在標準 H.264 decoder 下，若以 frame 作為移除的基本單位，則無法保證每一張 frame 皆可提早做移除，故所能減少的 DPB size 的機會較低。因此，我們傾向以更小的粒度作為移除的單位。根據 frame 間彼此的參考行為(2.3.1)，我們發現對於每一張 frame 的 pixels 被大量參考的時期以及僅剩少量被參考的時期。另外，根據 2.3.2 得知每一張 frame 其輸出到 display buffer 的時間是具有可調整的彈性。由上述兩項觀察結果得知，我們可以對每一張 frame 選擇一個時機點移除掉不再使用到的 pixels，僅保留仍須使用的 pixels，藉此來降低所需儲存的資料量，以達到減少 DPB size 的目標。

Chapter 3 Design

為了減少 DPB size，我們針對 DPB 內每一張 frame，分別選擇一個適合的時機點，僅保留住仍需被使用的 pixels，並將仍需使用的 pixels 轉存於所設計的儲存裝置中。如此一來，傳統 DPB size 則得以縮減。

為了達到上述目標，以下為我們將解決的問題，其中包含：

1. 對於每一張 frame，如何選擇一個適當時機(此時機我們在後續簡稱為 PRT(Partial-Frame-Data-Removed-Time))，僅保留住仍需使用的 pixels?

針對此問題，我們考慮以下兩點來決定每一張 frame 適合的 PRT

- a. 每一張 frame 不再被大量參考的時機點
- b. 每一張 frame 最早可以被 output for display 完的時間

2. 當每一張 frame 的 PRT 到達時，如何確認哪些 pixels 可以移除?

針對此問題，我們提出 Pre-decoding mechanism，預先部分解碼每一張 frame，已確認每一個 pixel 在 PRT 之後是否仍須被使用。

3. 如何管理 PRT 之前的完整 frames 以及 PRT 之後的 fragmented frames?

針對此問題，我們考量 decoding performance 以及 storage size，我們設計兩個儲存裝置分別稱為：

- a. Complete Reference Frame Storage(CRFS):儲存 PRT 之前的完整 frames (相似於傳統 DPB 的設計，皆儲存完整的 frames)

- b. Fragmented Reference Frame Storage(FRFS):儲存 PRT 之後的 fragmented frames(考量 decoding performance，選擇以 Indexed table 作為管理)(注意:此 Indexed table 可能發生 conflicts)
4. 當 FRFS 發生 conflicts misses 時，若被取代掉的資料未來仍需被使用的話，該如何做彌補?

針對此問題，我們主要考量解碼圖片的 image quality 以及 decoding performance，在不影響 decoding performance 的情況下，選擇與遺失的資料相像的 pixels 做彌補。

在此章節中，我們將一一介紹每一個問題的解決方法及考量。

3.1 Propose a new decoder scheme and decoding pipeline

Figure 3-1 為我們所設計新的 H.264 decoder scheme，此設計包含:

1. 一個傳統的 H.264 decoder(並不包含傳統 DPB 的部分)
2. 一個 Pre-decoding 的機制，在每一張 frame 尚未進入傳統 H.264 decoder 之前，確認該張 frame 在 PRT 之後有哪些 pixels 仍須被使用。
3. 一個 modified DPB，其中包含 2 個儲存裝置

a. Complete Reference Frame Storage(CRFS): 用來儲存由 decoder 所解碼出的完整 frames 以及相關的 PRT 資訊(PRT information 代表著每一個 pixel 在 PRT 之後是否須留下)，直到該張 frame 的 PRT 到達為止。當該張 frame 的 PRT 到達後，我們僅保留仍需使用的 pixels 並轉存於另外一個儲存裝置，稱為 Fragmented Reference Frame Storage。

b. Fragmented Reference Frame Storage(FRFS): 用來儲存每一張 frame 在 PRT 之後仍需使用的 pixels。

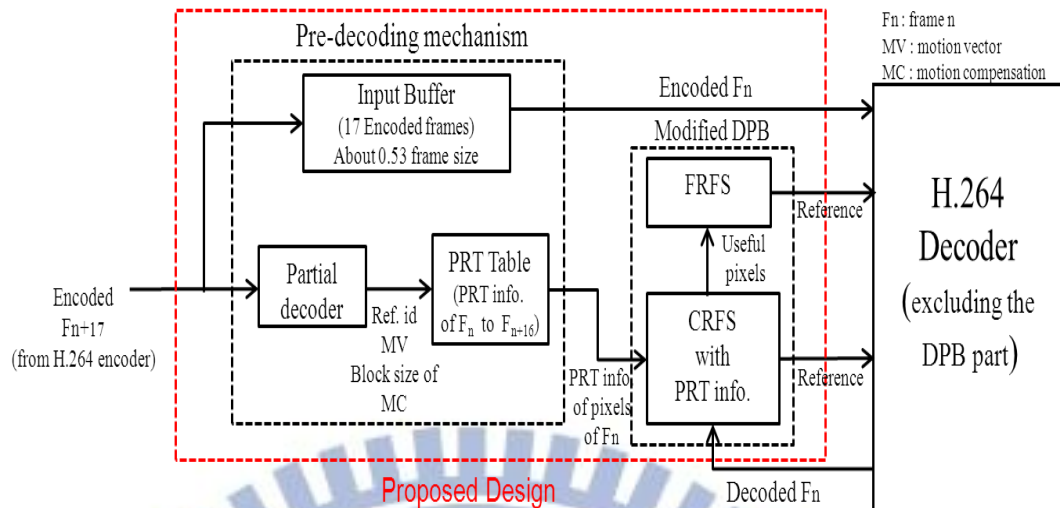


Figure 3-1 Block diagram of proposed new H.264 decoder.

Figure 3-2 為我們所提出的 decoding pipeline，針對每一張 frame 在每一個時間單位內列出其所執行的工作。橫軸為時間軸(時間單位:解碼一張 frame 所需的時間)，縱軸為 frames 的解碼順序。

對於每一張 frame 而言，需經過 1.預先部分解碼(partial pre-decoding)取得所參考的 pixels 資訊，並且 update PRT table(此 table 用來紀錄每一個 pixel 在 PRT 之後是否仍須有用)、2.送入傳統 H.264 decoder 做解碼、3.當此張 frame 的 PRT 到達時，將此張 frame 輸出到 display buffer 並僅保留仍須使用的 pixels、4.直到此張 frame 的死亡而整張移除。而每一張 frame 所需經過的步驟，都可以被 pipeline 執行。

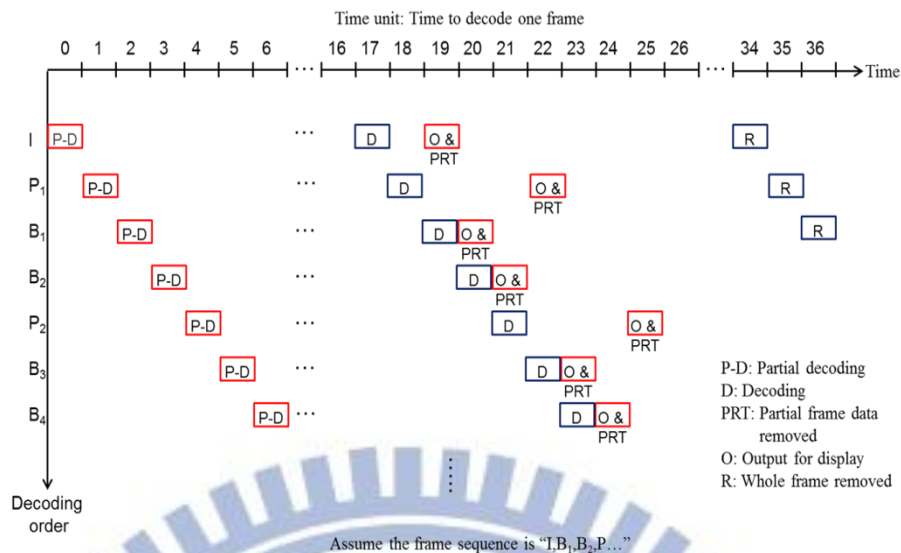


Figure 3-2 Proposed decoding pipeline.

在此章節中，我們將一一介紹 Figure 3-1 以及 Figure 3-2 中相關的設計細節。

3.2 Decision of Partial Frame Data Removed Time

對於決定每一張 frame 的 PRT，我們主要考量 2 個因素，其中包含：1. 每一張 frame 的 pixels 不再被大量參考的時機以及 2. 每一張 frame 最早可以被 output for display 完的時間。Figure 3-3 我們以 I, B₁, B₂, ..., B_n, P₁(n>0) 作為例子，列出每一張 frame 解碼的時間、不再被大量參考的時間(以 A 表示)以及最早可以被 output for display 完的時間(以 B 表示)。對於每一張 frame 來說，最早可以 output for display 的時間不會超過不再被大量參考時間。

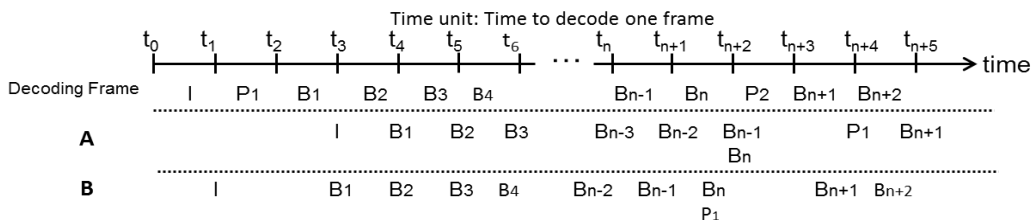


Figure 3-3 The time when each frame is no longer highly utilized and the earliest time at which the frame can be finished being output for display.

因此我們在決定每一張 frame 的 PRT 時，主要以每張 frame 不再視為主要參考對象的時間為考量，但仍有做部分調整。為了使每一個時間單位規律的移除一張 frame 不再使用到的資料，我們將 B_n -frame 的 PRT 延後一個時間單位，如 Figure 3-4 所示。另外，對於每一張 frame 輸出到 display buffer 的時間只要不超過該張 frame 的 PRT 情況下，仍有調整的空間。而基於硬體設計的考量以及追求穩定的播放速率，我們將每一張 frame 的輸出時間，延後跟 PRT 的時間相同。如此一來，在硬體的考量上，可減少所需的 Read port 數量。另外，在播放速率方面，可達到每解碼一張 frame 即輸出一張 frame 到 display buffer。

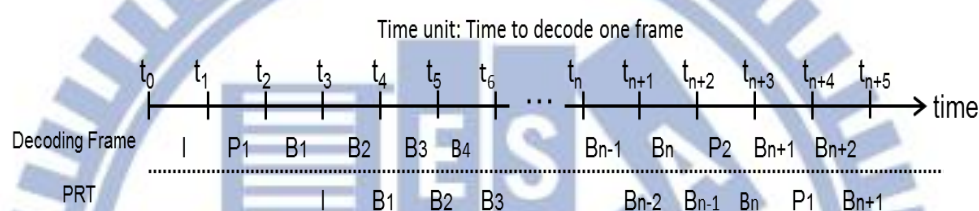


Figure 3-4 Partial frame data removed time of each frame.

3.3 Pre-decoding mechanism

由 3.2 中已經決定每一張 frame 的 PRT，但仍然無法確認每一張 frame 在 PRT 之後那些 pixels 仍須被使用。為了在每一張 frame 解碼之前，即確認該張 frame 所有 pixels 在 PRT 後是否仍須被使用，我們必須去追蹤後續 16 frames 的參考資訊，其主要原因是每一張 frame 的 pixels 最多被後續 16 frames 所參考。

為了在每一張 frame 解碼之前，即確認該張 frame 所有 pixels 在 PRT 後是否仍須被使用，我們提出 Pre-decoding mechanism。如 Figure 3-5 所示，此機制包含了：

1. 一個 Partial decoder(為傳統 H.264 decoder 的 entropy decoder 的簡化)，用來獲得每一張 frame 的參考資訊。

2. 一個 PRT Table，用來記錄每一個 pixel 在 PRT 之後是否仍須被使用的資訊。

3. 一個 Input Buffer，用來儲存由 encoder 所送來的 Encoded frames，當 Frame n 的 PRT 資訊紀錄完成後，則將 Encoded frame n 送入 decoder 做解碼。

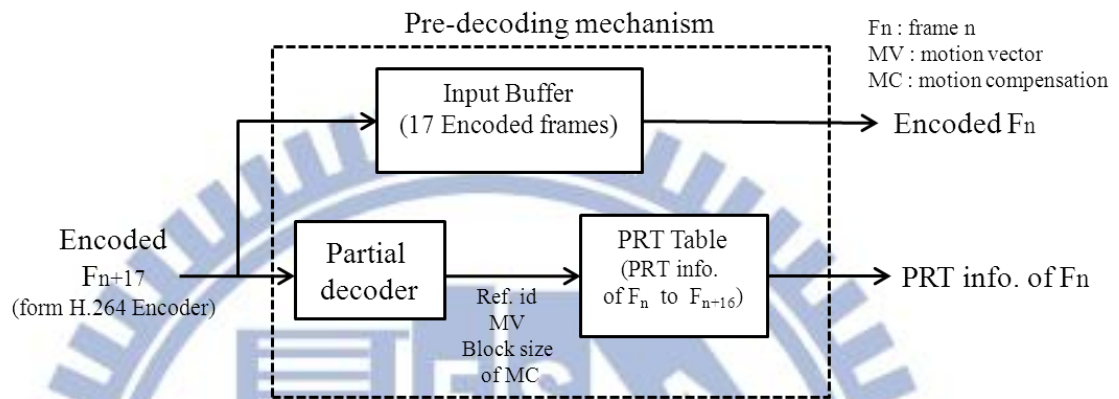


Figure 3-5 Block diagram of pre-decoding mechanism.

3.3.1 Partial decoder

此 partial decoder 為傳統 H.264 decoder 中的 entropy decoder 的簡化，相較於 entropy decoder 其功用僅負責解碼出每一張 frame 的參考資訊，其中包含所參考的 frame ID、motion vector(所參考的位置)以及 block size of motion compensation(所參考的範圍大小)並將上述資訊輸出給 PRT Table 做紀錄。

3.3.2 PRT Table

在預先解碼的過程中，我們額外設計一個儲存裝置，記錄每一個 pixel 在 PRT 之後是否仍須被使用，稱為 PRT Table，如 Figure 3-6 所示。由於每一個 pixel 是由 luma(亮度)以及 chroma(色度)所組成，且此兩部分的 PRT information 不盡相同，因此我們須分別記錄 luma 以及 chroma 的 PRT info，此後我們皆以 luma 做舉例。為了縮減 PRT 所需的儲存空間，我們僅用 1 個 bit 去紀錄每一個 pixel

在 PRT 之後是否仍須留下。當每一張 frame 的 PRT 到達時，則每一個 pixel 的 PRT bit 即可代表此 pixel 是否仍需留下。

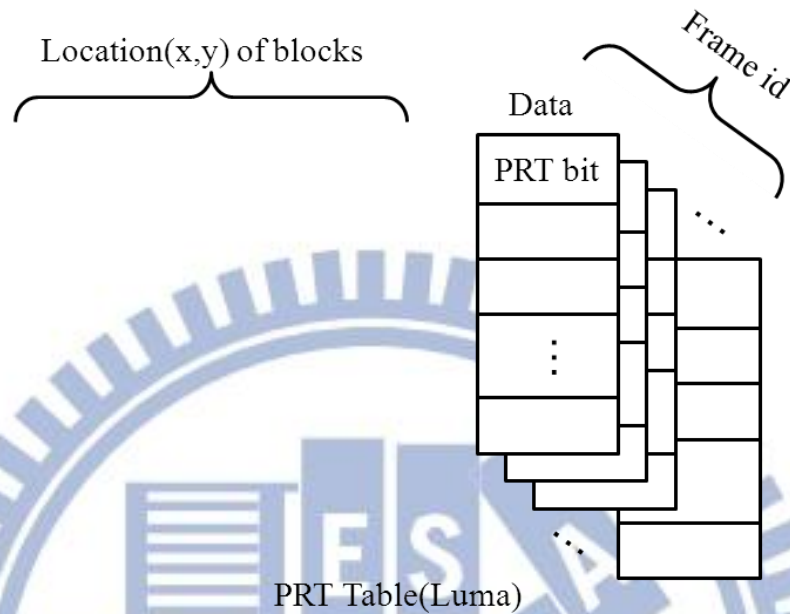


Figure 3-6 Data structure of PRT table.

PRT Table 同時須保留住 17 張 frames 相關的 PRT 資訊，我們由 Figure 3-7 來說明原因。以 frame F_0 為例， F_0 的 PRT table 從時間 t_1 開始記錄，直到時間 t_{17} 結束此 F_0 PRT table 才完成。因此，我們需要同時保留 17 張 frames 的 PRT info。

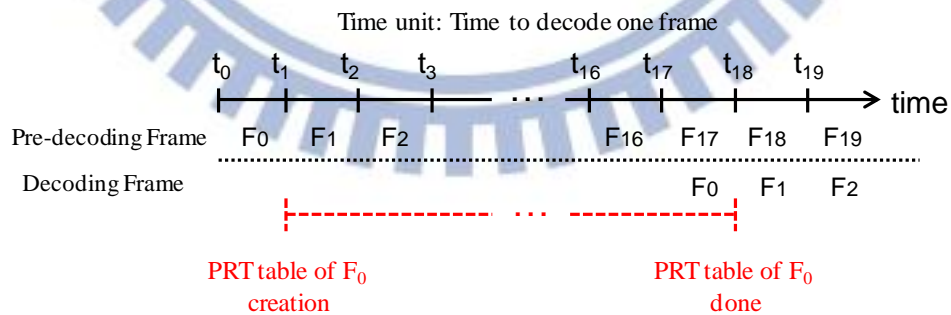



Figure 3-7 Pre-decoding and decoding time of frames.

由於同時保留 17 張 frames 的 PRT information，即使對於每一個 pixel 僅使用 1 個 bit 紀錄是否仍需留下，仍可能導致 PRT Table 儲存空間過大的問題。因此，

決定適合的 block size 做為紀錄的單位，可以有效控制 PRT Table 所需的儲存空間。(block:一張 frame 可切成多個相同大小的 block，並且彼此不重疊)

對於不同的 block sizes，Table 3-1 估算了 17 張 frames 所需的 PRT Table 儲存空間，隨著 block size 增大，其所需的 PRT table size 則越來越小。但由於以不同的 block sizes 做紀錄，會影響每一張 frame 在 PRT 後所需留下的資料量，當 block size 增加時，其所留下的資料量亦會增加。因此對於用何種 block size 作為紀錄 PRT info.的基本單位，我們將由 simulation 的結果決定。

Table 3-1 Required PRT table size for different block sizes in unit of storing on frame's pixels.

Block size	Size of PRT table of 17 frames (Unit : size of storing one frame)	The number of pixels which are needed to be kept after PRT for each frame
1 pixel	(Luma)1.42+(chroma)0.36	 <p>The number of pixels which is needed to be kept increase</p>
2*2 pixels	0.36+0.09	
4*4 pixels	0.09+0.023	
8*8 pixels	0.023+0.06	
16*16 pixels	0.006+0.0015	

3.3.3 Input Buffer

在 Pre-decoding mechanism 中，對於每一張 frame 我們為了預先部分解碼後續 16 張 frame 的參考資訊，因此延遲了每一張 frame 真正解碼的時間。由 Figure 3-7 所示，在 t17 之後才開始解碼 Frame 0，在此之前 Input Buffer 需保留住 Frame 0~Frame 16 經由編碼過後的資訊。因此，Input Buffer 同時須保留住 17 encoded frames。根據 H.264 壓縮率的統計[5]，17 encoded frames 總共約 0.53 張 frame size(frame size: 儲存一張 frame 所需的儲存空間)。

3.4 Modified DPB

由 3.1~3.3 的章節中，對於每一張 frame，我們解釋了如何決定 PRT 以及如何
在該張 frame 真正解碼前，確認每一個 pixel 在 PRT 之後是否仍需留下。在 3.4
節中，我們將解釋如何儲存以及管理每一張 frame。

Figure 3-8 為我們所提出的 modified DPB 架構，其中包含了 2 個儲存儲存裝
置，分別稱為 Complete Reference Frame Storage 以及 Fragmented Reference Frame
Storage。當每一張 frame 被解碼完且尚未到達該張 frame 的 PRT 時，我們將其
儲存於 CRFS 中，故 CRFS 所儲存的 frames 皆為完整的 frames 以及相關的 PRT
資訊。當每一張 frame 的 PRT 到達時，則將仍須使用的 pixels 轉存於 FRFS 中，
直到此張 fragmented frame 不再被使用則將其移除。

對於 CRFS 內所儲存的完整 frame，我們以類似傳統的 DPB 的設計來管理這
些 frames，而對於 FRFS 而言，為了不影響 decoding performance，我們提出以
Indexed table 來管理這些 fragmented frames。

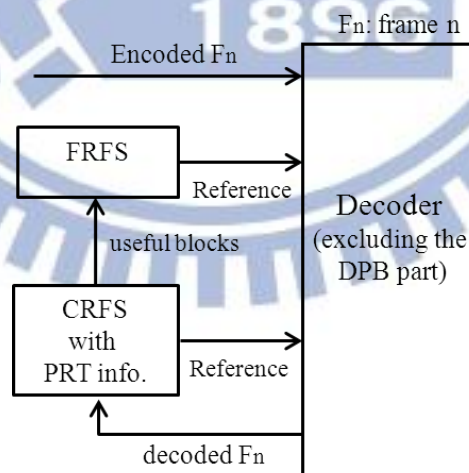


Figure 3-8 Memory architecture of the modified DPB.

3.4.1 Complete Reference Frame Storage

CRFS 負責儲存每一張解碼完的完整 frame 以及此張 frame 相關的 PRT 資訊，直到每一張 frame 的 PRT 到達。而 CRFS 對於完整 frames 的管理方式相似於傳統 DPB 的設計，我們由 Table 3-2 做比較，其中主要的不同在於傳統 DPB 需儲存 17 frames 而 CRFS 僅需儲存 3 張 frames 以及 Replacing order 的不同。

Table 3-2 Comparison of traditional DPB and CRFS.

	Traditional DPB	CRFS
Similar parts	<ul style="list-style-type: none"> Storing whole frames Indexed by frame id and location of pixels. 	
Stored data	Pixels	Pixels and PRT information
Kept frames	17 frames	3 frames
Replacing order	Decoding order (FIFO)	Display order

傳統 DPB 的 Replacing order 採取 FIFO(decoding order)，而我們可由 Figure 3-4 得知每一張 frame 其 PRT 的時間恰好是依照 display order，因此 CRFS 的 Replacing order 為 display order。另外，Figure 3-9 顯示在每一個時間點其 CRFS 內所儲存的 frames。由於每一個時間單位規律解碼一張 frame 並儲存於 DPB 內，另外從時間 t_3 開始皆有一張 frame 的 PRT 到達(將該張 frame 仍須使用的 pixels 轉存於 FRFS)，因 CRFS 僅需儲存 3 張 frames 的空間即可。

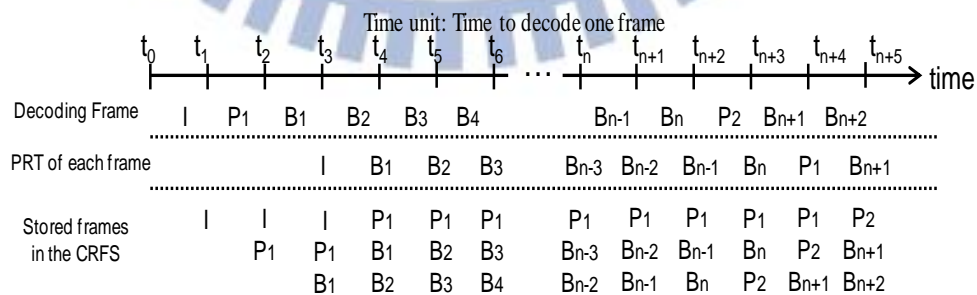


Figure 3-9 Frames stored in the CRFS using time as the horizontal axis.

3.4.2 Fragmented Reference Frame Storage

當每一張 frame 的 PRT 到達時，我們僅保留住仍然需使用的 pixels 並轉存於 FRFS 中。針對此 FRFS 的設計考量，我們主要以不影響 decoding time 以及降低整體所需的儲存空間為原則來選擇用何種資料結構較為合適。

為何在意是否影響 decoding time，主要是因為對於 real time decoder 而言，維持穩定的 output rate 是首要達到的目標，若所選擇的資料結構會影響到 decoding time，則會進而拖慢 output rate。由於傳統 DPB 的設計，皆儲存完整的解碼圖片，因此在搜尋資料（以 Pixel 為單位搜尋）、儲存資料（以 pixel 為單位依序儲存）以及移除資料（以 frame 為單位整張移除）的時間需求上，皆可在 $O(1)$ 時間內完成（註：移除整張 frame 的工作，可直接由下一張解碼圖片所覆蓋，因此實際上無需執行移除整張 frame 的動作）。因此，對於所選擇的資料結構部分，我們會特別重視其 timing performance，去衡量所需的搜尋時間、儲存資料的時間以及移除整張 frame 時所需的時間。

另外，因為此篇論文的目標在於減少整體所需要的儲存空間，因此在選擇資料結構時，也必須針對此部分作考量。由於 fragmented frames 的資料型態（零碎的 Pixels，而並非原本整齊排列的 pixels），因此我們必須要去記錄每一個 pixel 其所屬的 frame id、X 以及 Y 的位置，這樣才可辨識 FRFS 內的 pixels 是屬於哪張 frame 的哪個位置，此部分屬於我們額外增加的 storage overhead。因此，降低所需的 storage overhead 亦是要考量的重點之一。

基於上述兩點需求，我們在選擇以何種資料結構作為考量時，優先以不影響 decoding time 為主要考量，其次才為所需的 storage overhead。因為就 real time decoder 而言，較著重的仍是維持穩定的 output rate，其次才以降低所需儲存空間作考量。

在思考選擇以何種資料結構儲存的過程中，我們從常見的資料結構下手進行

分析，因為我們認為常見的資料結構簡單並且已有大量研究針對所需的時間以及空間進行改善。Table 3-3 列出一些常見的資料結構包含 Linked-list、Binary search tree、Indexed table 以及 Original DPB design 其所需的搜尋時間(以 pixel 為單位搜尋)、儲存資料的時間(以 pixel 為單位儲存)、移除時間(以 frame 為單位移除)以及所需的 storage overhead。

由此表可看出，在搜尋時間以及儲存時間的部分，Indexed table 會比 Linked-list 以及 Binary Search Tree 來的好，並且所需的時間複雜度與 Original DPB Design 相同，皆為 $O(1)$ 。而在移除一張 frame 所需的時間而言，Indexed table 表現不如其他的資料結構，但此部分實際上在 Chapter 4 的時間分析部分，可以利用 pipeline 的概念，將此工作隱藏在一個 stage(解碼一張 frame)內完成，因此不影響 Decoding time。因此在 timing performance 部分，發現實際上 Indexed table 的表現較佳。

針對空間需求的部分，在 storage overhead 的部分，由於 FRFS 的資料皆為 fragmented frames，因此除了 Original DPB design，其他三個資料結構皆需要記錄每一個 pixel 所需的 frame id 以及 Location(X,Y)，因此在 storage overhead 部分，三個資料結構的表現約略相同。但 Indexed table(類似 cache)，可能會發生 conflict misses 的情況，也就是雖然仍有空間可以儲存，但由於位置索引到相同的 set 中，而造成某些資料被取代之情況。因此對於空間的利用度而言，Indexed table 的表現比不上 Linked-list 以及 Binary Search Tree。而 Original DPB Design 由於是以完整 frames 作儲存，若只儲存 fragmented frames 的話，會造成多數儲存空間的浪費。

根據上述分析，可發現對於時間部分的效能，選擇 indexed table 儲存效果較好，我們在 Chapter 4 的分析中，證明使用 Indexed table 來管理 fragmented frames 不會影響 decoding time。而在空間利用性上，Indexed table 的表現比不上

Linked-list 以及 Binary Search Tree，因為會發生 conflict misses 的情況。基於以不影響 decoding time 為前提，降低所需儲存空間為次要考量下，我們最終選擇以 Indexed table 的資料結構來儲存 fragmented frames，雖然可能會發生 conflict misses 的情況，但由於 Indexed table 其結構類似 cache，而 cache design 已經是很成熟的技術，因此在降低 conflict misses 的技術上都有機會應用於此 Indexed table，來增加其 hit rate。另外，因為此 Indexed table 提供多種參數(indexing、associativity、line size 以及 replacement policy..等)作為選擇，可模擬出多種資料結構，因此，我們有機會可以根據需求而調整出適合的參數配對，來降低發生 conflict misses 的情況，以得到更好的效。

Table 3-3 Comparison of the data structure in FRFS

Data structure	Search time (以pixel為單位搜尋)	Insertion time (以pixel為單位儲存)	Remove time (以frame為單位移除)	Storage Overhead (不包含所需儲存的pixel值)	Other
Linked-list	$O(n)$	$O(1)$	$O(1)$	需紀錄: 1. 每一個pixel的frame id、Location (X,Y) 2. 指向下一個pixel的指標	
Binary Search Tree	$O(\log n)$	$O(\log n)$	$O(1)$	需紀錄: 1. 每一個pixel的frame id、Location (X,Y) 2. 指向兩個子樹的指標	
Indexed table (similar cache)	$O(1)$ (direct mapped)	$O(1)$	$O(n)$	需紀錄: 1. 每一個pixel的frame id、Location (X,Y)	會發生有空間可以儲存，但無法被利用的情況
Original DPB design	$O(1)$	$O(1)$	$O(1)$	N/A	會有多數空間無須被利用，而造成硬體資源的浪費

PS: 假設FRFS中有n筆資料

Figure 3-10 為我們所提出 FRFS 的方塊圖。其 Address 為 Frame id、pixel 的 (X,Y) 座標所組成。另外，對於每一個 entry 而言，由 valid bit(用來表示該 entry 的空間是否可被使用)、Tag 以及 Data(某一個 block 內所有的 pixels value)所組成，而此架構亦可調整成為 multiple-way。

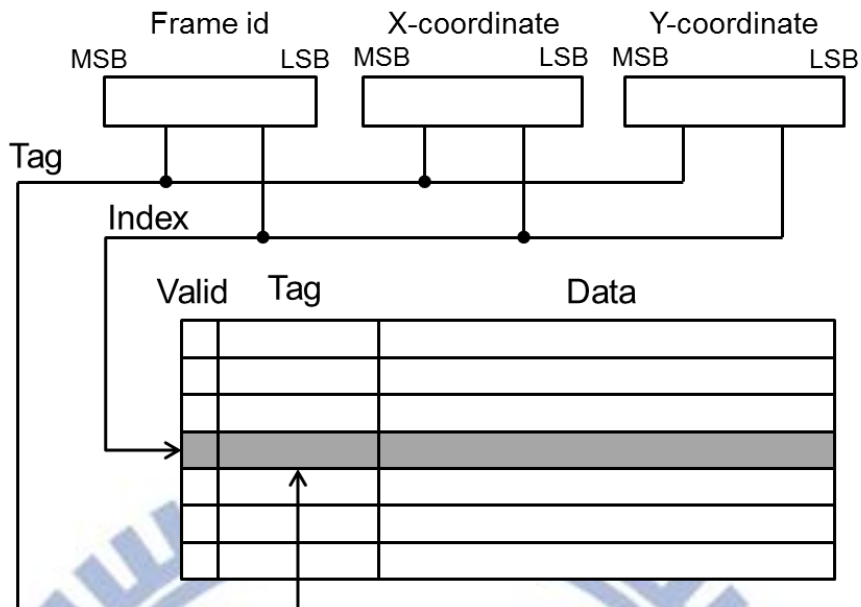


Figure 3-10 Block diagram of Fragmented Reference Frame Storage.

為了減少 FRFS 中發生 conflicts 的情況，我們希望將每一個 blocks 平均散佈於每一個 set 中。根據觀察每一張 frame 在 PRT 之後所留下 blocks 的分佈情形，我們發現在 frame 內部，這些在 PRT 之後仍須使用的 blocks 具有群聚性。另外，在相鄰的 frame 間，所留下的資料位置具有重疊性。根據上述所觀察到的特性，我們選擇以 frame id 以及 pixel 位置較為 lower bits 的部分作為 Index，如此可以盡可能將資料平均打散於各個 set 中。

對於儲存於 FRFS 的 blocks 而言，有兩個情況會將此 block 移除，其中包含：1. 此 block 已經無效(已經超過傳統 DPB 對於此 block 移除的時間)、2. 當發生 address conflict 時(當該 set 內所有空間已滿，並有新的 block 將要存入時)。以下我們分別對上述兩點做說明。

1. Block 已經無效:

當 FRFS 內的 block 已經超過傳統 DPB 對此 block 移除的時間後，此 block 就不可能再被使用。而對於每一張 frame 而言，解碼完後續 16 張 frame 之後，此張 frame 的所有 blocks 將不再被使用。Figure 3-11 說明我們何時移除已經無效

的 frames，以 frame F_0 為例，當 F_{16} 解碼完之後， F_0 就不再會被使用到，故我們從 t_{17} 開始將 F_0 移除，其他 frame 以此類推。

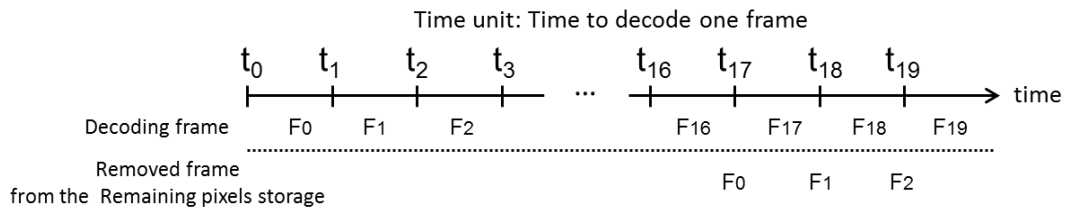


Figure 3-11 Time to remove the invalid blocks of frames in the FRFS.

2. 當發生 Address conflict 時:

當某一個 set 中已經沒有儲存空間且又有新的 block 即將存入時，FRFS 必須選擇該 set 其中一筆資料作為移除的對象。而考量 video 中鄰近 frame 間高度像似之特性，反觀越老的 block 其被參考的機會越低，因此我們選擇以 FIFO(First-In-First-Out)的 replacement policy 作為準則。

3.5 Compensated method while the conflicts misses of FRFS happened

由於我們選擇以 indexed table 作為儲存 fragmented frames 的資料結構，因此，可能會發生 conflict misses 的情況，當此情況發生就有可能影響到後續解碼的圖片品質，造成某些圖片會有部分資料移時的情況。對此，設計一個補償機制以減少 image quality 的下降幅度是必要的。

當 FRFS 發生 conflict misses 時，會造成所需參考的 pixels 遺失而影響圖片品質，而我們的想法是提供與遺失 pixels 較為相似的資料作為補償，盡可能避免 image quality 大幅下降的情況發生。由於發生 conflict misses 後需要找資料作彌補，可能會影響到 decoding time，因此會造成我們沒有辦法直接在 FRFS or CRFS 中，找到與遺失的 pixels 最相似的 pixels 作補償，只能退而求其次以猜測的方式找到相似的 pixels 來彌補。基本上，我們可由 temporal locality(鄰近圖片相似之特性)以及 spatial locality 兩方面找資料作彌補(同張 frame 鄰近區域相似的特性)。

針對此兩方面來找資料作彌補，我們最終決定是以 temporal locality(鄰近圖片相似之特性)方面來彌補資料，而捨棄掉以 spatial locality 方面作補償，其主要是我們發現同一張 frame 內各個區域實際上其 pixel 值差異不盡相同，因為一張 frame 可能有多個物體為主體的情況，若以同一張 frame 鄰近區域作彌補的話，其差異可能較大，此部份我們經過實驗觀察，發現以鄰近圖片作彌補的效果來的比以同張 frame 作彌補來的好，但仍會因 videos 的特性不同而有所改變。

對於 temporal locality 方面，我們該選擇以哪一張 frame 的哪些 pixel 作為補償對象較能減少 image quality 下降幅度，針對此問題我們在決策時遵循兩個原則：

1. Modified DPB 內的其他 frames 哪些比較相似(相較於遺失的資料)。
基於根據鄰近圖片(display order)相似的特性，由最鄰近 frames 接近與遺失資料相同區域的 pixels 作彌補是較為合適的。
2. 由於最鄰近的 frames 其內部 pixels 仍然有可能遺失，因此哪張鄰近的 frames 其存在 modified DPB 的機率較高，此部分也是我們關注的地方。由於 Modified DPB 由 CRFS 和 FRFS 所組成，因此我們也會選擇從哪一個 storage 找資料作彌補較為合適。

每一個遺失的 pixel 基本上會有前後最鄰近(display order)的兩張 frames，而此兩張 frame 有較大的機會已存入 FRFS 當中。然而按照 display order 後面最鄰近的 frame，其存在 FRFS 的機率相較於前一張最鄰近的 frame 而言是較高的，因為 FRFS 會以 decoding order 移除掉不再使用 frame，其前一張最鄰近 frame 較有可能先被移除，因此我們主要以後一張 frame 作為彌補的對象。

當發生 conflict misses 後，欲找尋資料作彌補的時候，我們所擁有的資料為遺失 pixel 所屬的 frame type、目前解碼 block 的 Motion Vector、目前解碼 block 所屬的座標以及 B frame 的數量，根據以上之資料作為 input，我們整理出選擇以哪個 pixel 作為彌補的通式。

首先，我們先在下述 Define 中，定義所需的參數。Equation(1)根據不同的 frame type 列出其所彌補的 pixel 座標。例如：目前所遺失的資料的 frame type 為 I frame 且此 frame 的 id 為 m (decoding order)，則我們會去此 I frame 後面最鄰近的 frame (Next B frame (frame id: $m+2$)) 內找資料做彌補。至於選擇此張 frame 的哪個 pixel 做彌補對象，我們會根據其 Motion Vector (目前正在解碼 block 的 MV)，並利用內插法 (因為已經與原始參考的 frame 不相同，因此需要根據 frame id 的差距，調整 motion vector 的值)，去找到我們認為合適的區域。

Define

1. T_i : The frame type of frame i
2. m : The frame id of missing pixels
3. c : The frame id of currently decoded block

 $\Rightarrow T_m$: The frame type of frame m

 $\Rightarrow T_c$: The frame type of frame c
4. MV_x : The x - component of Motion Vector of currently decoded block
5. MV_y : The y - component of Motion Vector of currently decoded block
6. X : The X -coordinate of currently decoded block
7. Y : The Y -coordinate of currently decoded block
8. n : #B frames between I and P frame
9. RF : The reference pixel of FRFS when the conflict misses happened
10. RC : The reference pixel of CRFS when the conflict misses happened
11. $RP(a,b,c)$: The frame id a and coordinate(b, c) of reference pixel

Equation (1): The reconstructed region of FRFS when the conflict misses happened

$$(1) RGF(T_m, MV_x, MV_y, X, Y, m, c)$$

$$= RP(a,b,c)$$

=

$$\left\{ \begin{array}{l} \left(m + 2, (MV_x + X) * \frac{c - (m + 2)}{c - m}, (MV_y + Y) * \frac{c - (m + 2)}{c - m} \right), \text{if } T_m \text{ is I frame} \\ \left(m + 1, (MV_x + X) * \frac{c - (m + 1)}{c - m}, (MV_y + Y) * \frac{c - (m + 1)}{c - m} \right), \text{if } T_m \text{ is } B_1 \sim B_{n-1} \text{ frame} \\ \left(m - n, (MV_x + X) * \frac{c - (m - n)}{c - m}, (MV_y + Y) * \frac{c - (m - n)}{c - m} \right), \text{if } T_m \text{ is } B_n \text{ frame} \\ \left(m + n + 2, (MV_x + X) * \frac{c - (m + n + 2)}{c - m}, (MV_y + Y) * \frac{c - (m + n + 2)}{c - m} \right), \text{if } T_m \text{ is P frame} \end{array} \right.$$

但補償方法若只選擇以 FRFS 內的 frames 作彌補，仍會發生欲彌補的 pixels 資料遺失的情況，而影響到整個 decoding time(因為需要重新找尋資料作彌補)以及圖片品質。為了解決掉一直找不到資料作彌補的狀況，我們的想法是同時去 CRFS 中找尋合適的區域，作為彌補之對象，雖然 CRFS 內所儲存的 frames 相較於遺失 pixel 的 frame id 較遠，但由於 CRFS 中所儲存的 frames 都是完整的，因此必定能找到所要彌補的 pixels。根據上述兩個原則，Equation(2)列出我們由 CRFS 中選擇哪張 frame 作為彌補對象。

Equation (2): The reconstructed pixel of CRFS when the conflict misses happened

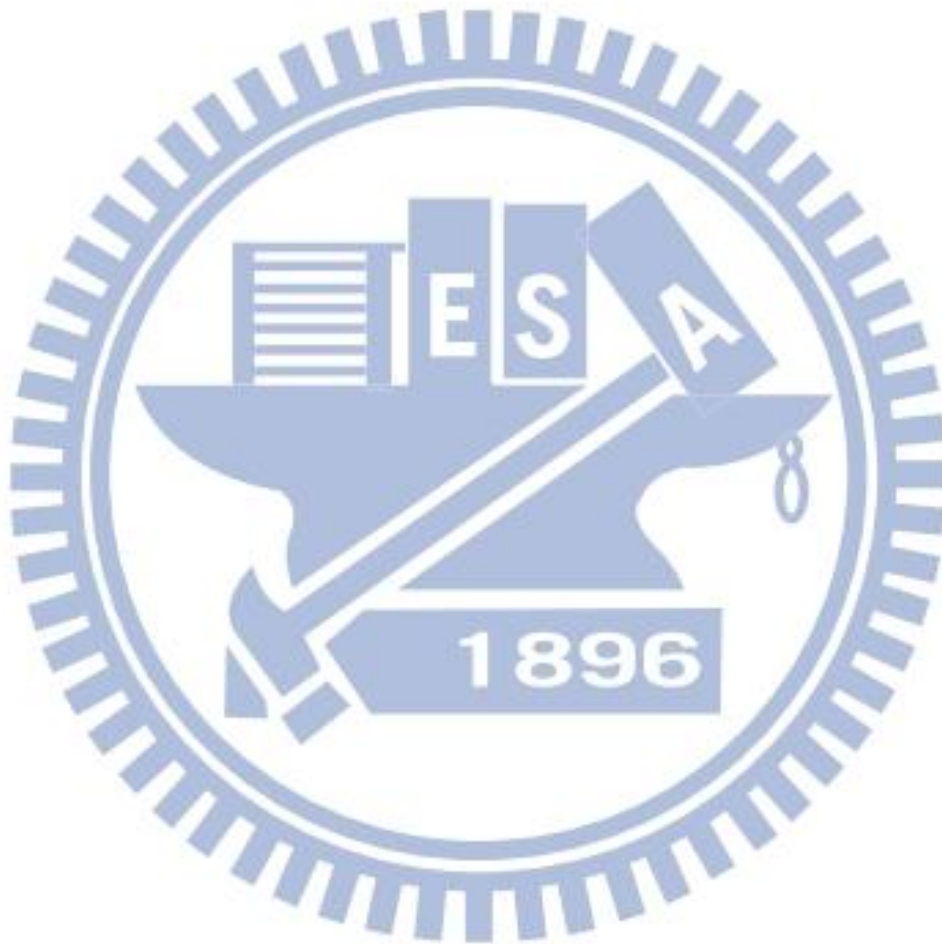
(2) RGC(T_c , MV_x , MV_y , L_x , L_y , c , m)

=RP(a,b,c)

=

$$\left\{ \begin{array}{l} \left(c - 1, (MV_x + X) * \frac{c - (c - 1)}{c - m}, (MV_y + Y) * \frac{c - (c - 1)}{c - m} \right), \text{if } T_c \text{ is } B_1 \text{ frame} \\ \left(c - 1, (MV_x + X) * \frac{c - (c - 1)}{c - m}, (MV_y + Y) * \frac{c - (c - 1)}{c - m} \right), \text{if } T_c \text{ is } B_2 \sim B_n \text{ frame} \\ \left(c - n - 1, (MV_x + X) * \frac{c - (c - n - 1)}{c - m}, (MV_y + Y) * \frac{c - (c - n - 1)}{c - m} \right), \text{if } T_c \text{ is P frame} \end{array} \right.$$

根據統計，由 FRFS 中找尋資料作彌補的效果會優於從 CRFS 中找資料做彌補，其原因是 FRFS 內的 frames 相較於遺失 pixel 的 frame id 較為鄰近。因此我們的彌補方法是同時去 FRFS 與 CRFS 中找尋欲彌補之資料，若 FRFS 找到欲彌補之資料，則以此資料作彌補，否則，則以 CRFS 作彌補。如此作法，在不影響 decoding time 的情況下，可維持一定的圖片品質。在 Chapter 4 的分析中，我們會證明此彌補方法並不會影響 decoding time。



Chapter 4 Simulation

在此章節中，針對我們所提出的設計，去分析在不同儲存空間下，對於圖片品質的影響。另外，對於我們所提出設計，我們分別針對時間、硬體成本以及所消耗之電量等部分進行分析討論。

4.1 Experiment

首先，先介紹我們所使用的模擬環境，包括 JM 16.2 software[6]用來模擬 H.264 編解碼端的過程，以及 CACTI 5.3[7](Technology:32 nm)用來評估 memory access time、chip area 以及 power 等項目。Table 4-1 為我們實驗時所使用的 benchmarks，每一個 video 各 100 張 frames，其解析度為 176*144(QCIF)與 352*288(CIF)。

Table 4-1 Common video sequences used in experiment.

Benchmarks	Resolutions	Number of frames
Carphone	QCIF	100
Crew	QCIF、CIF	100
Ice	QCIF、CIF	100
Foreman	QCIF	100
Highway	QCIF	100
City	QCIF、CIF	100
Mother-daughter	QCIF	100
Walk	QCIF	100
Suzie	QCIF	100
Coastguard	QCIF	100
Bridge-close	QCIF	100
Claire	QCIF	100
Container	QCIF	100
Grandma	QCIF	100
Hall	QCIF	100
Miss-america	QCIF	100
Silent	QCIF	100
Trevor	QCIF	100
Akiyo	QCIF	100
Salesman	QCIF	100
News	QCIF	100
Bride-far	QCIF	100

對於 image quality 的評估方面，我們以 Peak signal-to-noise ratio(PSNR)作為衡量 image quality 的標準，PSNR 值越高越好，標準 H.264 所解碼出的 frames 其 PSNR 大致上介於 30(dB)~50(dB)之間，一般而言，當 PSNR>30(dB)時，視為人類可接受的範圍[8]。Table 4-2 列出標準 H.264 decoder 在上述 benchmarks 下平均所能達到的 image quality。

Table 4-2 Average PSNR of standard H.264 decoder

	Y(Luma)	U(Chroma)	V(Chroma)
H.264 decoder	38.67(dB)	41.53(dB)	42.15(dB)

在 chapter 3 中，由於我們須增加 PRT table 用以儲存每一個 pixel 在 PRT 之後是否仍需留下的資訊，而用不同的 block sizes 做為紀錄的單位會影響到 PRT table size 以及所需留下的資料量，因此，我們需要去選擇以何種 block size 做為紀錄的單位較為合適。Table 4-3 以及 Table 4-4，我們分別列出 Modified DPB 及 Storage overheads 中，會受不同 block sizes 作為記錄單位而受影響的項目。

Table 4-3 The issues are affected by different block sizes of Modified DPB.

Modified DPB	The data needed to be kept	The storage size will be affected by different block sizes
Complete Reference Frame Storage	Pixels of 3 complete frames	✗
	PRT information of 3 frames	✓
Fragmented Reference Frame Storage	Pixels of 14 fragmented frames	✓

Table 4-4 The issues are affected by different block sizes of storage overheads.

Storage overheads	The data needed to be kept	The storage size will be affected by different block sizes
PRT table	PRT information of 17 frames	✓
Input buffer	17 encoded frames	✗

Table 4-5 我們分別列出 PRT table 以及 CRFS 在不同 block sizes 做為紀錄單位下，所需的 storage sizes(Luma parts)。Table 4-6 為儲存 Chroma parts 所需的儲存空間。

Table 4-5 Storages sizes of PRT table and CRFS with luma parts for different block sizes.

Size of block	PRT table size (Luma)	Size of Complete Reference Frame Storage	
		pixels(Luma)	PRT info. (Luma)
1 pixel	1.42	2.66	0.33
2*2 pixels	0.36	2.66	0.08
4*4 pixels	0.09	2.66	0.02
8*8 pixels	0.023	2.66	0.005

Unit: size of storing one frame

Table 4-6 Storages sizes of PRT table and CRFS with chroma parts for different block sizes.

Size of block	PRT table size (chroma)	Size of Complete Reference Frame Storage	
		Pixels (chroma)	PRT info. (chroma)
1 pixel	0.36	1.34	0.08
2*2 pixels	0.09	1.34	0.02
4*4 pixels	0.023	1.34	0.005
8*8 pixels	0.006	1.34	0.00125

Unit: size of storing one frame

Figure 4-1 顯示出根據不同的 storage sizes 與 block sizes 下，對於 image quality(Luma)的影響。橫軸為 storage size，是由 PRT table size、CRFS size 以及 FRFS size 加總而得(僅儲存 luma 相關的資訊)。圖中每一條線的起始點，其 storage size 是由 PRT table size 以及 CRFS size 所加總。而隨著 FRFS 的儲存空間增加，所能達到的 PSNR 值也隨之上升。由圖中可發現當沒有 FRFS 時，其所需能達到的 PSNR(Y)=31.07(dB)，此值已經可視為人類可接受的 image quality。另外，可發現選擇以 4*4 block 作為紀錄 PRT 的單位時，在達到一定的 image quality 下，相較於其他的 block 而言，其所需的 storage size 較小。因此，最終我們決定以 4*4 block 作為紀錄單位。另外，我們推薦選擇以平均線的轉折點(PSNR=36.97 dB)

作為所能達到的 image quality 較為合適，因為在此點之後增加 storage size 其 image quality 上升的幅度已趨緩。

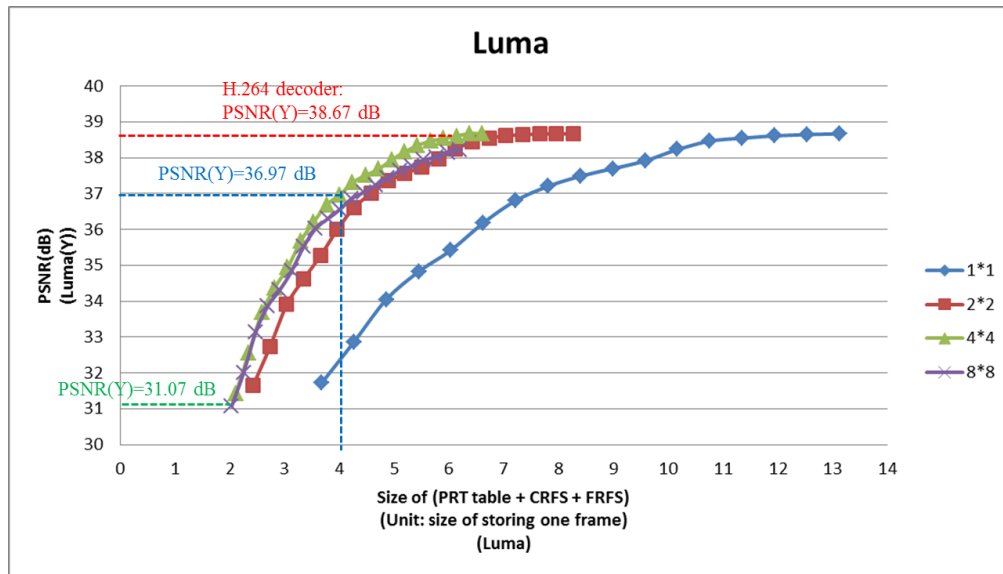


Figure 4-1 Effect of PSNR(Luma) value for different storage sizes and block sizes.

Figure 4-2 及 Figure 4-3 顯示出根據不同的 storage sizes 與 block sizes 下，對於 image quality(Chroma)的影響。可發現當選擇 2*2 block or 4*4 block 作為紀錄 PRT 資訊的單位時，在大部分的情況下，其所達到相同的 image quality 下，相較於其他 block 而言，所需的 storage sizes 最小。另外，由於使用 4*4 block 做為紀錄 PRT 資訊的單位，相較於 2*2 block，其 PRT table 的 storage size 會較小，因此，最終我們決定以 4*4 block 作為紀錄單位。我們亦推薦選擇平均線的轉折點作為所能達到的 image quality。

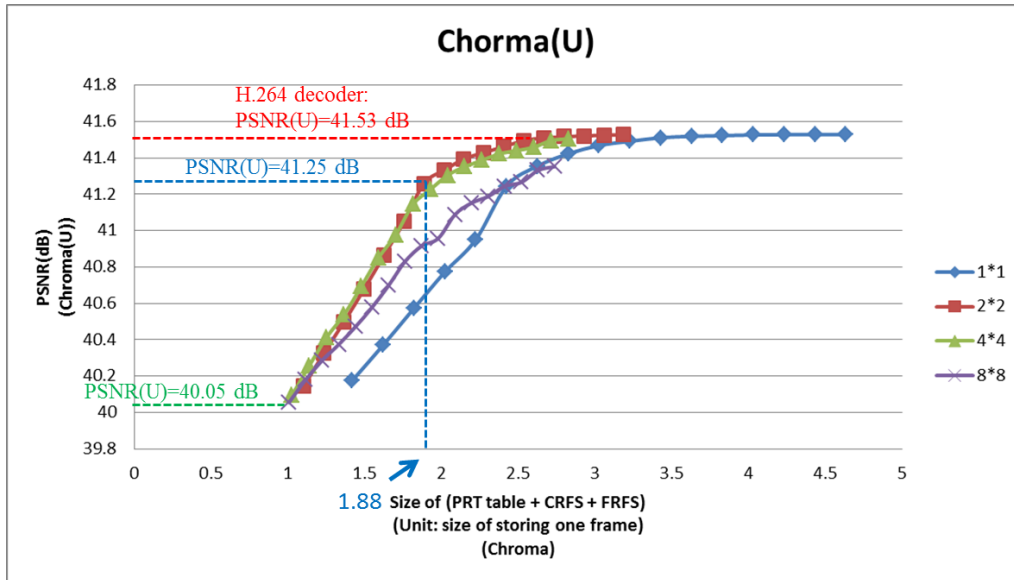


Figure 4-2 Effect of PSNR(Chroma(U)) value for different storage sizes and block sizes.

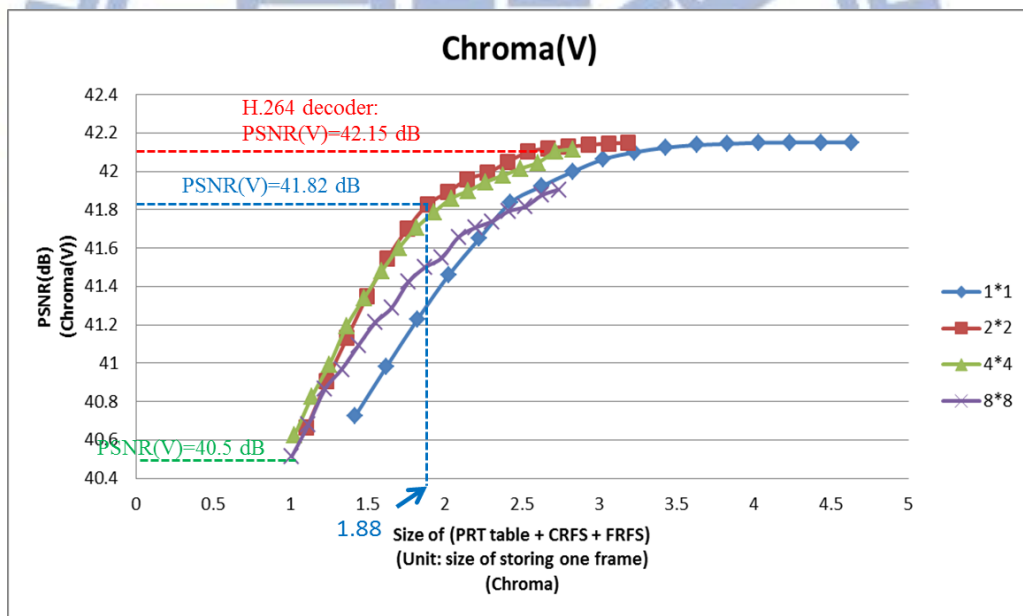


Figure 4-3 Effect of PSNR(Chroma(V)) value for different storage sizes and block sizes.

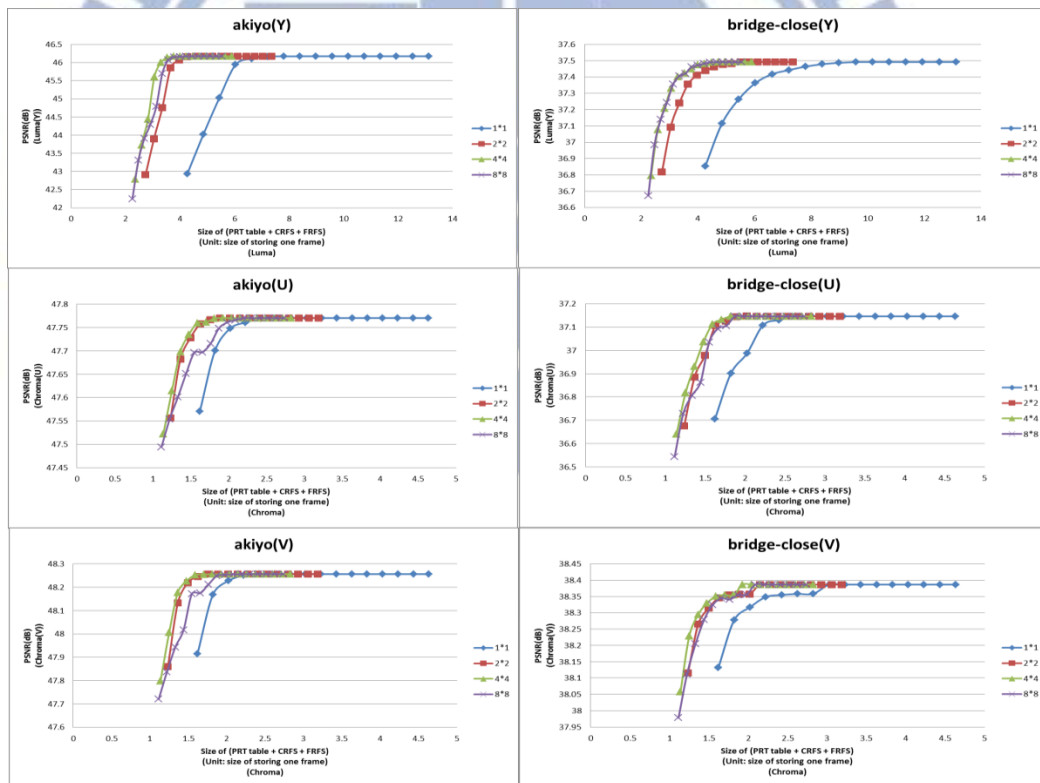
Table 4-7 為我們所提出的設計所需的 DPB sizes 以及 Storage overheads，總共所需的儲存空間為 6.407 張 frame sizes，相較於傳統 DPB 需要儲存 17 張 frames，總共可省下 62.4% 的儲存空間。在 image quality 方面，相較於標準 H.264 解碼出

的圖片品質，會下降 4.1%。在 Luma 部分，PSNR 降低了 1.7(dB)，在 Chroma 部分，則 PSNR 則降低了 0.33(dB)。

Table 4-7 Overall DPB sizes and storage overheads.

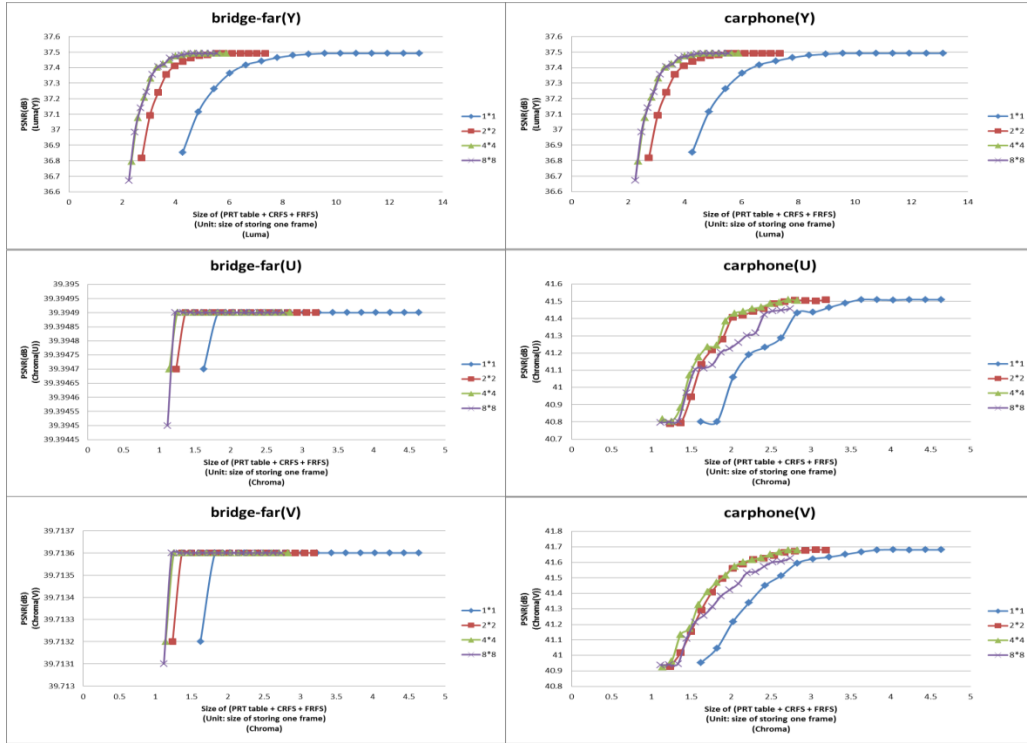
	Modified DPB				Storage overhead		Input Buffer	Total
	CRFS		FRFS		PRT Table			
	Luma	Chroma	Luma	Chroma	Luma	Chroma		
Storage size	2.015	1.004	1.895	0.85	0.09	0.023	0.53	6.407

Figure 4-4 附上各個 video 在我們的設計中所能達到的 image quality。



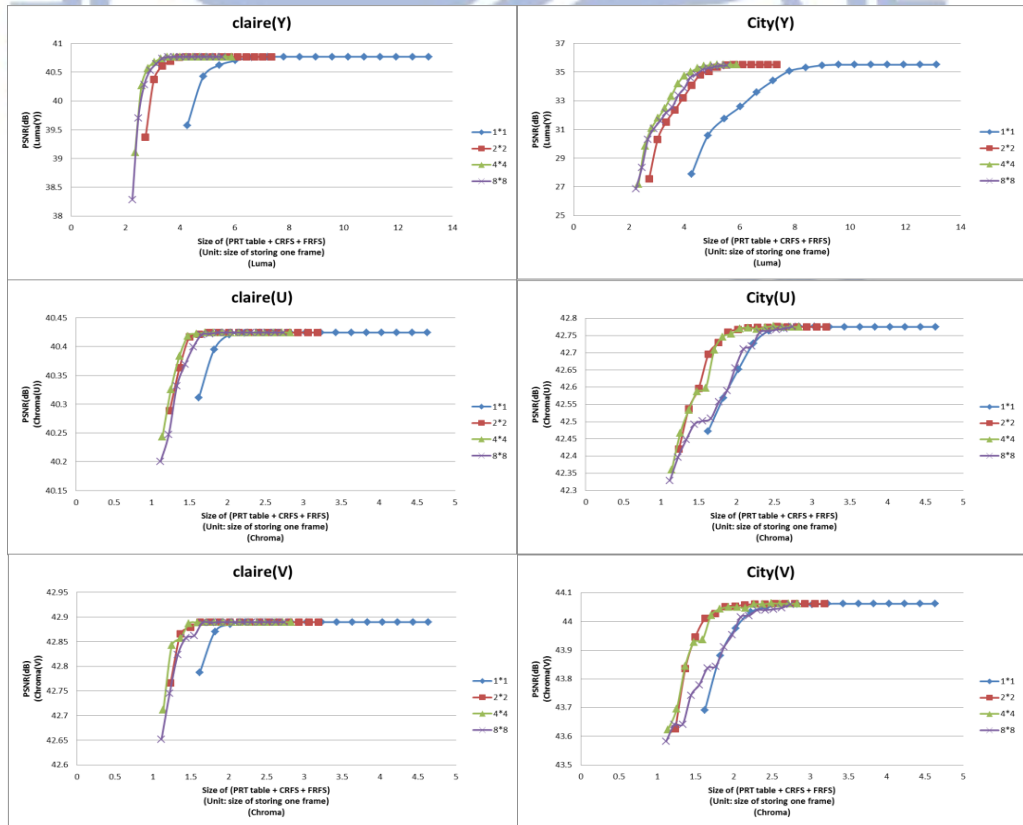
(a)

(b)



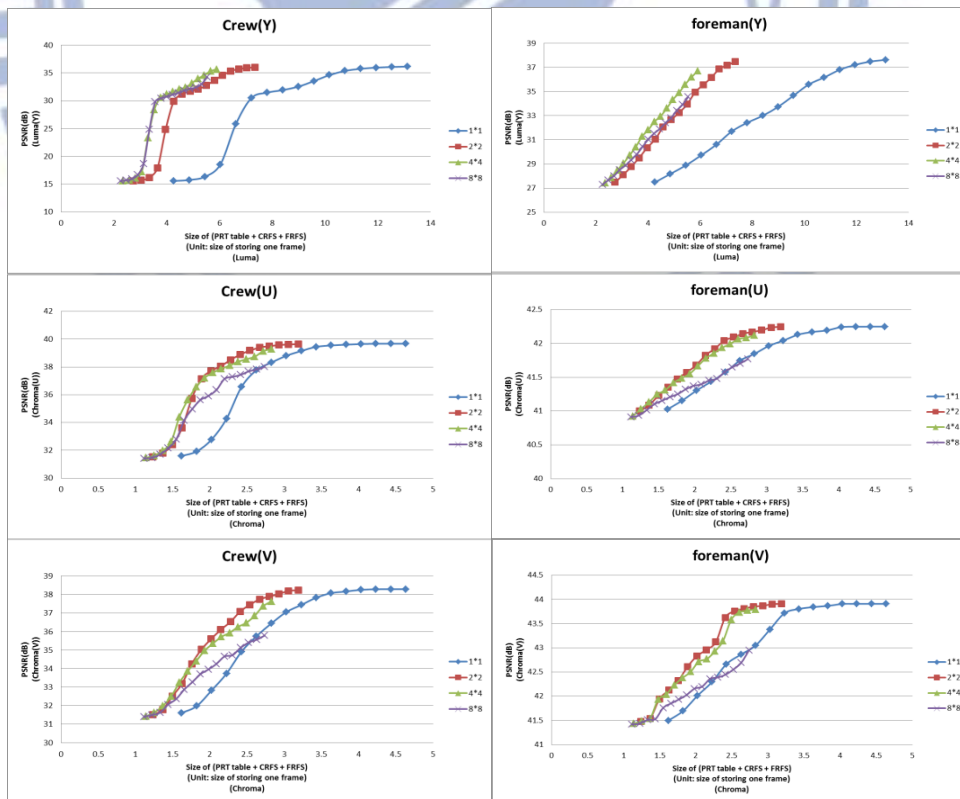
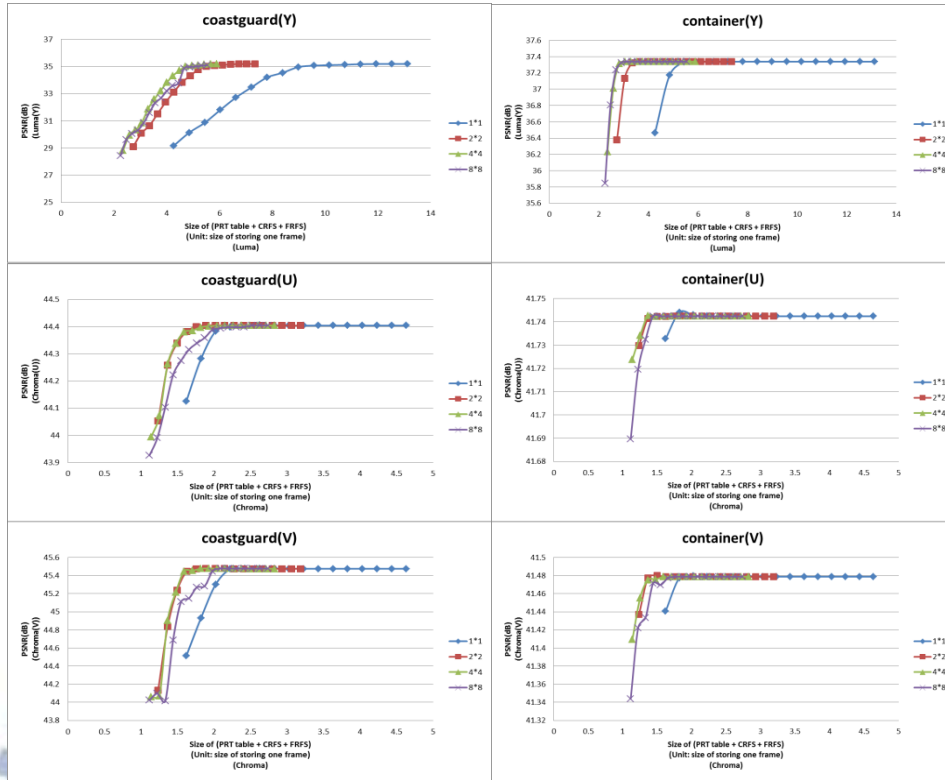
(c)

(d)



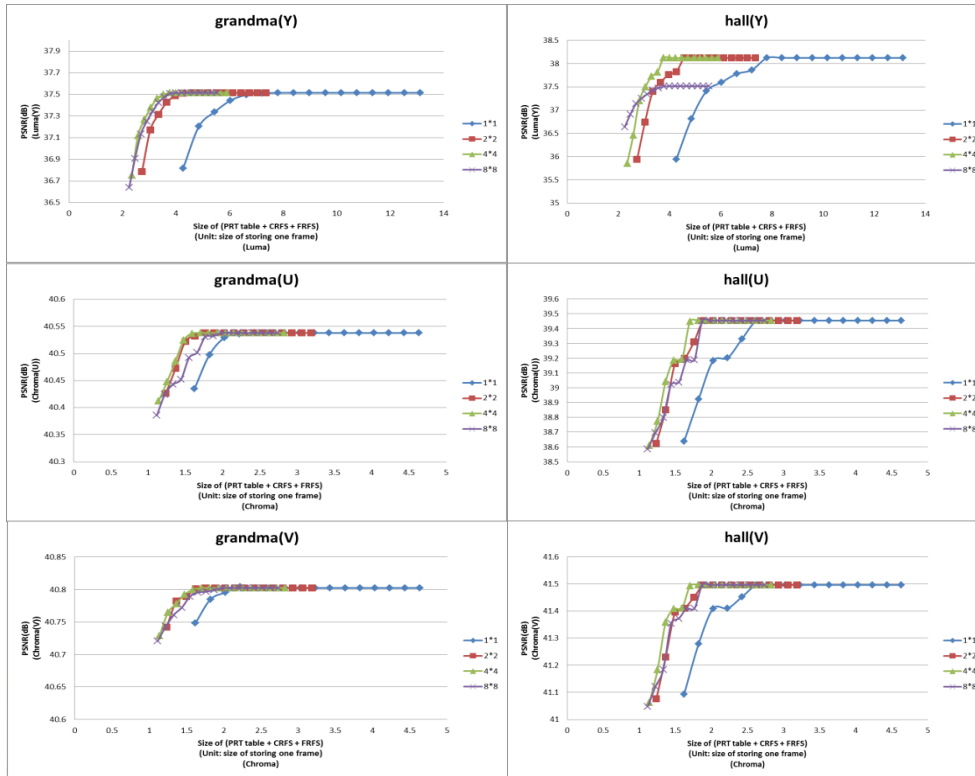
(e)

(f)



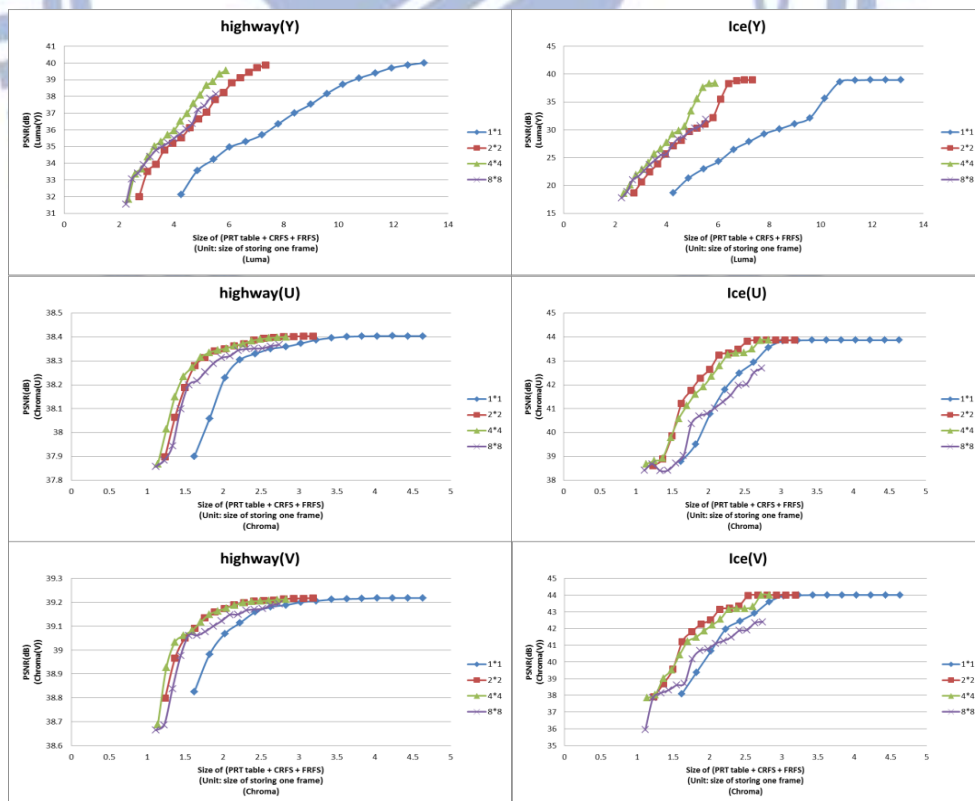
(i)

(j)



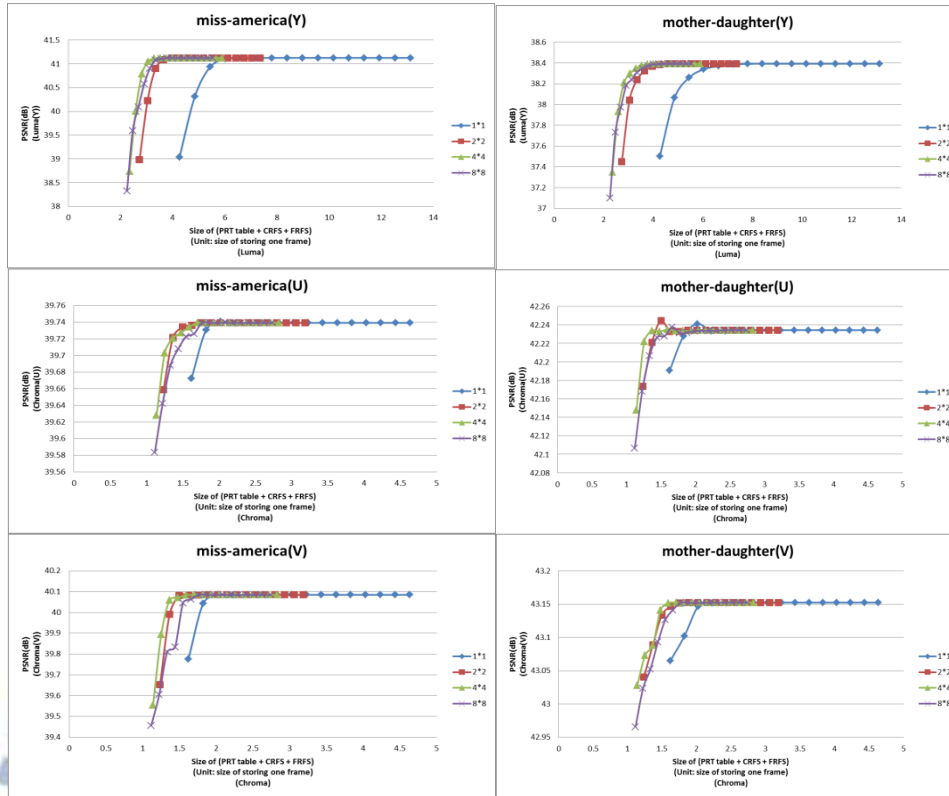
(k)

(l)



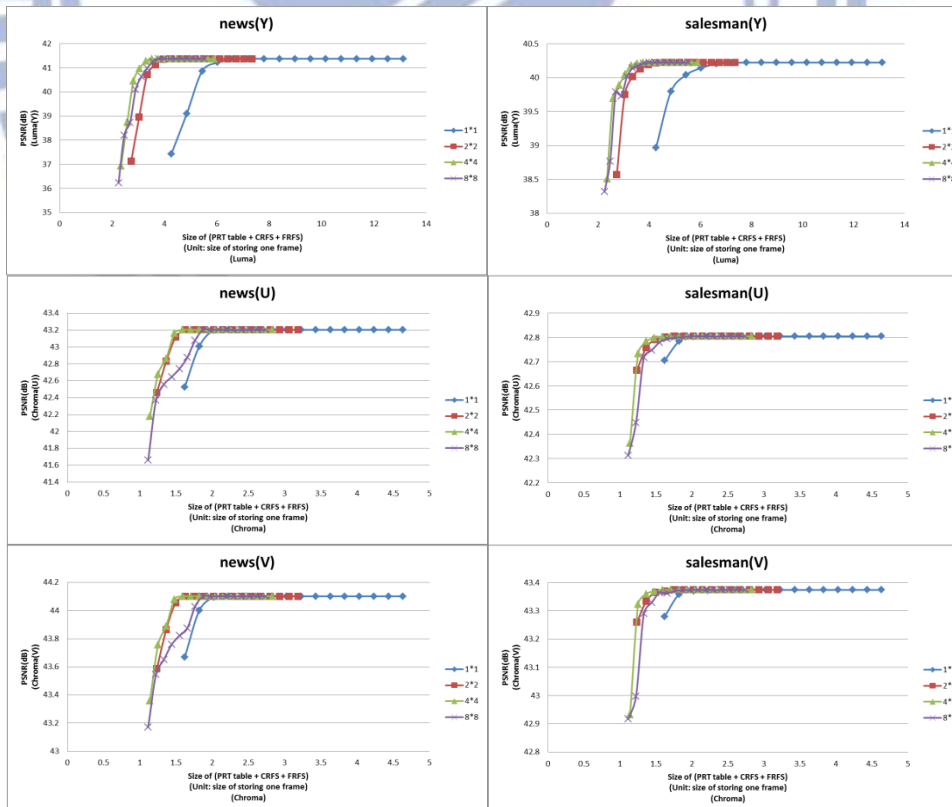
(m)

(n)



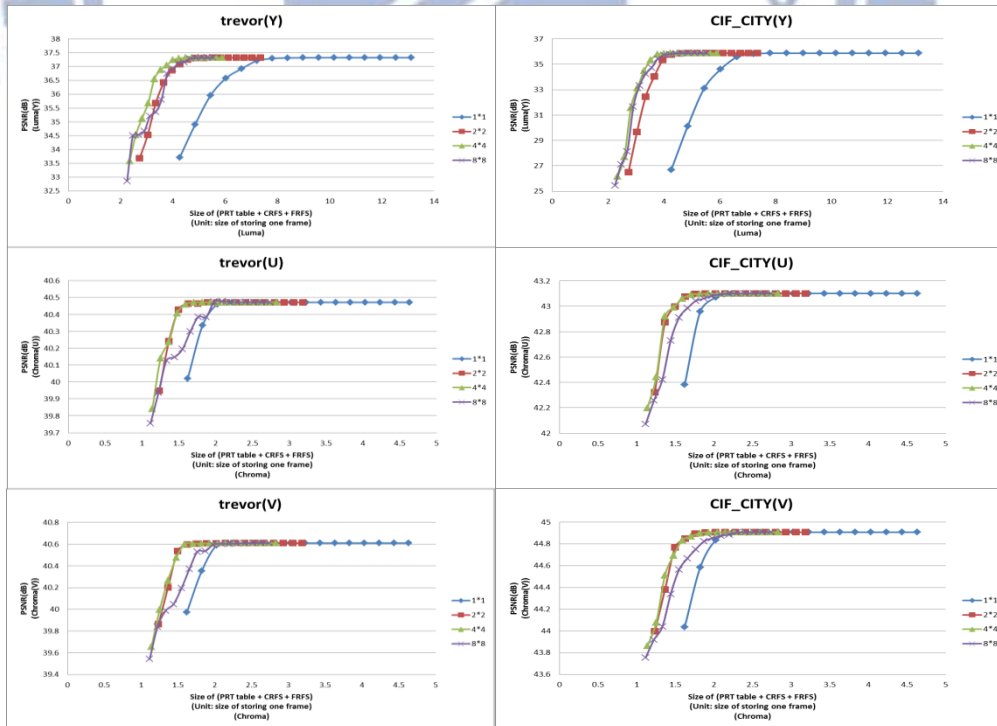
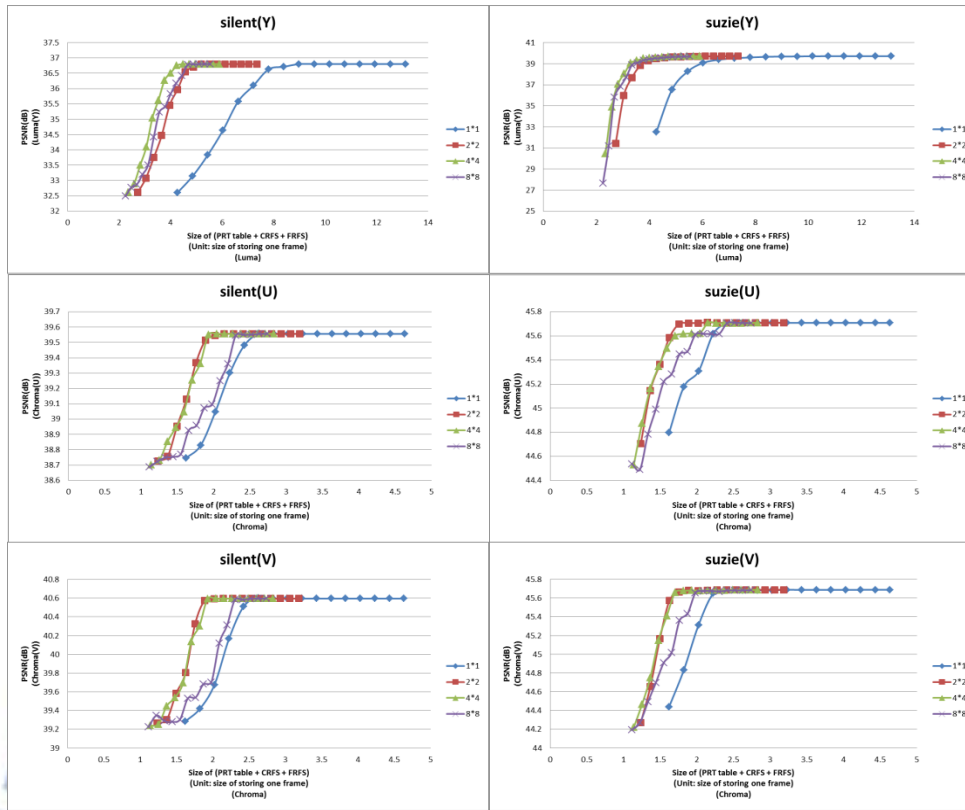
(o)

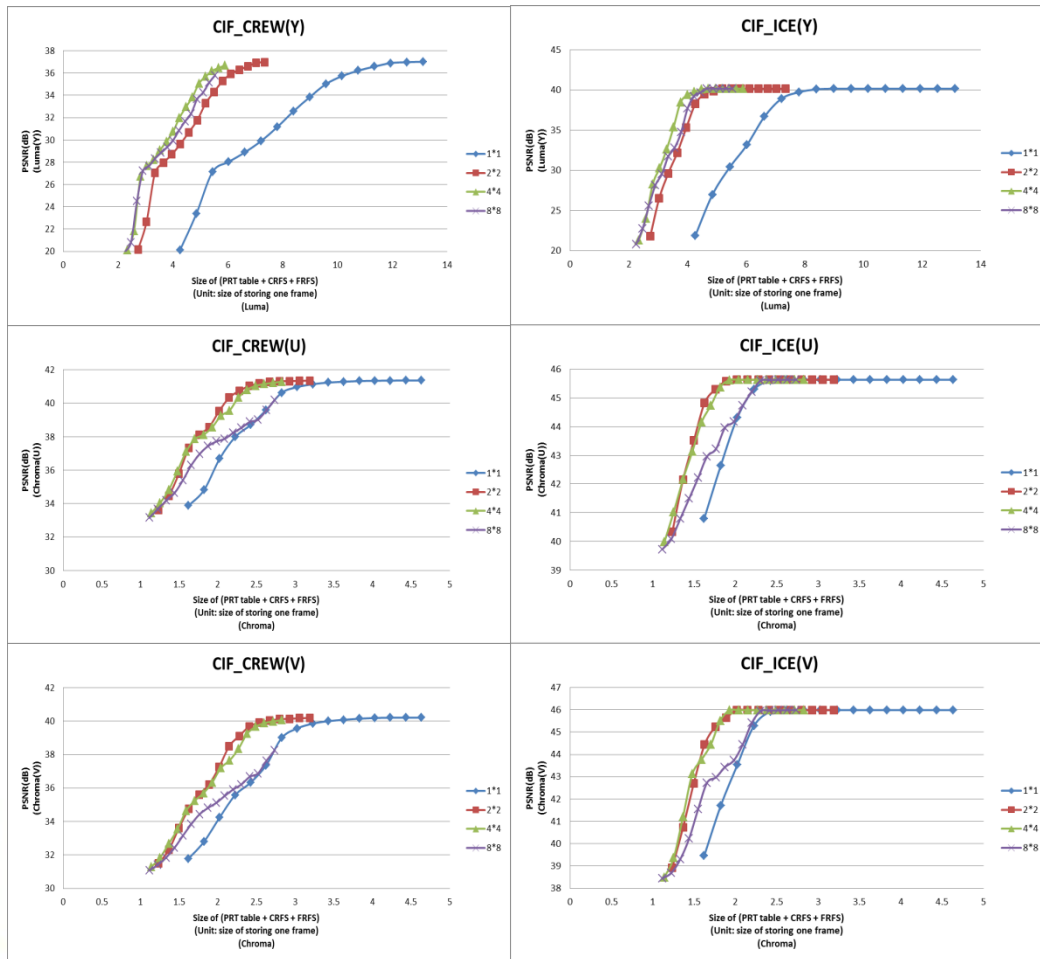
(p)



(q)

(r)





(w)

(x)

Figure 4-4 Effect of PSNR(Y、U、V) value for different storage sizes and block sizes.

由上述個別 benchmark 可發現，對於 Carphone、Crew、Ice、Foreman、Highway... 等 videos 而言，由於這些 videos 屬於 frame 間場景移動速度較快的類型，因此會造成每一張 frame 在 PRT 之後所需留下的資料比其他 videos 多，因此需要較大的 FRFS 的 size，才可以保留仍需使用的 pixels 以至於不發生 data loss 的情形。而對於 frame 間移動速度較慢的 videos(例如: Bridge-close、Container、Bridge-far、Akiyo.....等)，可發現在 PRT 留下的資料相對較少，因此僅需要較少的 FRFS size 即可保留仍須留下的 pixels。對於相同 Video 不同解析度的圖片測試(例如: Crew、Ice、City)，發現對於 storage size 與 image quality 的關聯雷同，因此，真正影響

image quality 好壞的因素，推測仍與 video 的性質關聯性較高。

根據上述的統計圖表發現，對於 Foreman 此 video 而言，相較於其他所測試的 videos，其所需要的 FRFS 的 storage size 是最大的，其 total storage size 約需 12.407 張 frame size(FRFS size 約需要 8.75 張 frame size)才不會造成 image quality 下降的情形發生，而我們所選擇的 total storage size 約需 6.407 張 frame size(FRFS size 約需 2.75 張 frame size)，對於此 video 會造成約 11%(-4.3 dB)的 image quality 下降，雖然 image quality 下降了 11%，但仍是接受的範圍(一般 image quality 可接受範圍>30dB)。

4.2 Analysis of proposed design

在此節當中，我們將針對所提出的設計，在時間、硬體成本以及 Power 的消耗上進行分析討論。Table 4-8 列出針對我們所提出的設計，所設定的環境參數。

Table 4-8 Environment parameters of CACTI

CACTI 5.3	
Technology Node(nm)	32
Nr. of Banks	1
Storage size (Unit: size of storing one frame) (Size of one frame: 1920*1080*1.5 Bytes)	Traditional DPB: 17 frames CRFS: 3.019 frames FRFS: 2.755 frames Input Buffer: 0.53 frames PRT table: 0.113 frames
#R/W ports	Traditional DPB: 2 Read ports, 1 Write port CRFS: 2 Read ports, 1 Write port FRFS: 1 R/W port, 1 Read port, 1 Write port
Temperature(k)	300

4.2.1 Timing analysis of proposed design

在此段落中，我們將分析所提出的設計在時間上的可行性。針對此部分，我們設計的限制在於為了不降低 decoding performance。因此我們需要去分析每一張 frame 額外增加或者是修改過的 stages，是否可以 fit 進原本 H.264 的 stage 內 (one stage: Time to decode one frame (1s/30))。

根據 Figure 3-2，每一張 frame 需經過 partial decoding stage(額外增加的 stage)、decoding stage(修改過的 stage)、Output & Partial Frame Removed stage(額外增加的 stage)以及 Whole frame remove stage(修改的 stage)。我們將針對上述每一個 stage，分析是否可以 fit 進 Time to decode one frame 內。

A. Partial Decode(P-D) stage

Tasks:

1. Partial decode one frame
2. Update PRT Table
3. Store one encoded frame into Input Buffer

上述三個工作流程為: Task1=>Task2(依序執行), Task3 可與 Task1 同時執行

此部份我們須分析是否 $\text{Time to execute the Task1 and Task2} < \text{Time to decode one frame}$

由於 partial decoder 是傳統 H.264 decoder 中 entropy decoder 的簡化，因此我們可以假設 Execution time of Partial decoder $<$ Execution time of Entropy decoder。

另外，相較於傳統 H.264 的 motion compensation(MC)，其 update the PRT table 所需的工作較為簡單，因此可以假設 Execution time of record the PRT info. of blocks $<$ Execution time of MC。因此可證明 Time to partial decode one frame+

Time to update the PRT table < Time to decode one frame 。

Time to execute task3 < Time to decode one frame

Table 4-9 為 Time to store one encoded frame into Input Buffer 所需的時間，可發現遠小於 Time to decode one frame 所需的時間，因此 Partial decode stage 是可以 fit 進原本的 stage 內。

Table 4-9 Timing analysis of writing one encoded frame into Input Buffer.

Time to write one encoded frame into Input Buffer	Time to decode one frame (Assume: fps is 30)
$(1920*1080*1.5*0.1)/16*1.42=27604(\text{ns})$	33,000,000(ns)

B. Decode(D) stage

在解碼一張 frame 中，相較於傳統 H.264 decoder，我們僅改變了 DPB access 的部分，因此僅需評估 modified DPB 的 access time 是否少於傳統 DPB 的 access time。Time to access modified DPB 可分成下列三個 case

Case 1: Hit in CRFS (T1)

Case 2: Hit in FRFS (T2)

Case 3: Miss in FRFS(此部分所需的時間為 $T2+\text{Max}(T1,T2)$)(此為 critical path)

Table 4-10 列出 access 不同 storage 所需的時間，可發現 Case 3 所需的 access time 亦小於傳統 DPB 的 access time。因此，我們可確認在此 stage 是不會增加解碼一張 frame 所需的時間。

Table 4-10 Access time of modified DPB and traditional DPB.

	Complete Reference Frame Storage (T1)	Fragmented Reference Frame Storage (T2)	Traditional DPB
Access time(ns)	3.84	3.81	8.35

C. Output and Partial Frame Remove (O&PRT) stage

Tasks:

1. Load all pixels of the frame reaching PRT from CRFS (T1)
2. Store useful pixels in to FRFS (T2)
3. Output pixels of the frame for display
4. Remove one frame from CRFS

上述三個工作流程為: Task1=>Task2、Task3 以及 Task4(Task2、Task3、Task4 可同時執行)

Task3 所需的時間可無須重視，因為傳統 H.264 decoder 在一個 stages 內，也需要將一張 frame output for display。另外，我們的 modified DPB access time 相較於傳統 DPB 來的快，因此，我們輸出一張 frame 亦可在一個 stage 之內完成。

Task4 所需的時間亦可忽略，此部分可由下一張 decoded frame 的資料覆蓋掉此張 frame，因此不需要真正做移除。

我們僅需分析是否 $T1+T2 < \text{Time to decode one frame}$

Table 4-11 列出 T1 以及 T2 所需的時間，可發現 $T1+T2 < \text{Time to decode one frame}$ ，因此 O&PRT stage 可以 fit 進原本的 stage 內。

Table 4-11 Timing analysis of loading all pixels of the frame reaching PRT from CRFS and storing useful pixels into FRFS.

Time to load all pixels of the frame reaching PRT from CRFS (T1)	Time to store useful pixels into FRFS (T2)	Time to decode one frame (Assume: fps is 30)
$1920*1080/16*3.84=497664$ (ns)	$1920*1080/16*3.81=493776$ (ns)	33000000(ns)

D. Whole frame remove(R) stage

Task: Remove one invalidated frame in FRFS

此部分我們須證明:

Time to remove one invalidated frame in FRFS < Time to decode one frame

Table 4-12 顯示 Remove one invalidated frame in FRFS 所需的時間小於 decode one frame 所需要的時間。因此，R stage 可以 fit 進原本的 stage 內。

Table 4-12 Timing analysis of removing one invalidated frame in FRFS.

Time to remove one frame	Time to decode one frame (Assume: fps is 30)
$((1920*1080)/16)*1.9*3.81*2=2715768$ (ns)	33000000(ns)

根據上述分析，由於所有新增或者是修改的 stage 皆可 fit 進傳統的 stage 內，因此，我們所提出的設計並不會影響到 decoding performance。

4.2.2 Hardware costs of proposed design

此節主要針對我們所提出的設計與傳統 H.264 decoder 在 hardware costs 上的分析比較。Table 4-13 以及 Table 4-14 我們分別列出傳統 H.264 decoder 與我們所提出的設計所需的 hardware cost。傳統的 H.264 decoder 所需的 chip area 總共約為 385.4961(mm²)，而我們所提出的設計其所需的 chip area 約為 128.49(mm²)，我們所提出之設計約可省下 66.7%的 chip area。

Table 4-13 Hardware cost of traditional H.264 decoder.

	Trad. DPB	Decoding Logics
Storage size	17	X
Read / Write Ports	2 read port, 1 write port.	
Chip Area (mm ²)	380.7	4.79

Table 4-14 Hardware cost of proposed design.

	Modified DPB		Storage overhead		
	CRFS	FRFS	Input Buffer	PRT	Entropy Decoder
Storage size	3.019	2.755	0.53	0.113	X
Read / Write Ports	2 read ports, 1 write port	1 write port, 1 read port, 1 R/W port	1 read port, 1 write port	1 read port, 1 write port	
Chip Area (mm ²)	62.6	51.7	6.69	1.5	1.2

4.2.3 Power analysis of proposed design

在此節中，將分析我們所提出的設計與傳統 H.264 decoder 所需的 Power 消耗。由 Table 4-15 中，列出傳統 H.264 decoder 在 dynamic power 以及 static power 上分別所需要的 power，在每秒解碼 30 張 frame 的情況下，總共所需 12.162(Watt)。而在 Table 4-16 中，列出我們所提出的設計在 dynamic power 以及 static power 上分別所需要的 power，在每秒解碼 30 張 frame 的情況下，總共所需 5.337(Watt)。相較於傳統 DPB 在 static power 上的消耗，由於我們降低了 DPB 所需的儲存空間，因此有了顯著的改善。最終，我們所提出的設計可以減少 56.2% 的 power consumption。

Table 4-15 Power analysis of traditional H.264 decoder.

(Assume:30 fps)	Dynamic power(Watt)
Decode frames (excluding DPB access)	
DPB access	0.204
Output frames for display	0.102

	Static power
H.264 decoder (excluding DPB)	
Traditional DPB	11.67

Power = 0.186 (Watt)

Table 4-16 Power analysis of proposed design.

(Assume:30 fps)	Dynamic power(Watt)
Decode frames (excluding DPB access)	
DPB access	0.076
Partial pre-decode frames	
Saving useful pixels of frames from CRFS to FRFS (including output frames for display)	0.0395
Remove frames from FRFS	0.04
Read encoded frames from Input Buffer	0.0052

(Assume:30 fps)	Static power(Watt)	
H.264 decoder (excluding DPB)		
Pre-decoding mechanism	Partial decoder	
	PRT Table	0.07
	Input Buffer	0.35
Modified DPB	CRFS	2.19
	FRFS	2.34

Power = 0.186 (Watt)
Power = 0.041 (Watt)

Chapter 5 Conclusion

在此篇 paper 中，我們針對 frame 間 pixels 的參考行為進行分析，對於每一張 frame 而言，找到適當的時機點僅保留住少量仍須被使用的 pixels，並提出一個 pre-decoding mechanism 用以確認哪些 pixels 在該時機之後仍須留下，另外，提出一個 modified DPB architecture 用以管理資料。此外，我們根據 streaming data 的特性，提出一個適合的 decoding pipeline，用以達到穩定的 output rate。在不影響 decoding performance 的情況下，相較於 H.264 decoder，我們的設計在 image quality 損失 4.1% 下，能省下 62.4% 的記憶體空間。在 Hardware costs 以及 power consumption 部分，相較於傳統 H.264 decoder，能省下 66.7% 的 chip area 以及 56.2% 的 power consumption。

我們的設計能給予使用者在其所想達到的 image quality 下，其相對應需要付出的 hardware costs 為多少，此部份我們給予使用者自行去選擇。另外，此設計因為可以大幅減少所需的 chip area 以及電量的消耗，對於嵌入式系統而言是特別有幫助的。另外，由於未來的圖片解析度只會越會來大，由於此設計大幅縮減所需的 chip area，也就代表在與 H.264 decoder 幾乎相同的 chip area 下，採用我們的設計可以解碼更高品質的圖片。

針對我們的設計，目前仍有一些可以加強的部分，在 Fragmented Reference Frame Storage 方面，我們提出一個簡單的 indexed table 來管理，若可針對此 storage 提升 hit rate，則有機會以更小的儲存空間來得到更好的 image quality。所有關於改善 cache design 的方法，都有機會應用在此 indexed table 上。另外，在資料彌補方面，我們提出較簡單的設計，以鄰近圖片相似的特性取鄰近圖片的 pixels 作為彌補，而此部分仍然有加強的空間，若可找到與遺失資料更相似的區域，則有機會可以達到更好的 image quality。

References

- [1] Tung-Chien Chen; Chung-JrLian; Liang-Gee Chen;"Hardware architecture design of an H.264/AVC video codec,"*Design Automation, 2006. Asia and South Pacific Conference on* , vol., no., pp.8 pp., 24-27 Jan. 2006
- [2] QiuShen,Wang,Ye-Kui,Hannuksela,MiskaM,Houqiang Li, Yi Wang,"Buffer requirement analysis and reference picture marking for temporal scalable video coding," *Packet Video 2007*
- [3] H. Schwarz, D. Marpe, and T. Wiegand,"Hierarchical B Pictures", Joint Video Team (JVT) document JVT-P014, Poznan, Poland, July, 2005
- [4] Iain E. G. Richardson " H.264 and MPEG-4 video compression" John Wiley & Sons Ltd 2003
- [5] Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification(ITU-T Rec. H.264/ISO/IEC 14496-10 AVC)
- [6] M16.2 Reference Software. [Online]. Available : http://iphome.hhi.de/suehring/tml/download/old_jm/
- [7] S. Thoziyoor, N. Muralimanohar, J. H. Ahn, and N. P. Jouppi, "CACTI 5.3", Technical Report. HPL-2008-20. 2008.
- [8] Gangadharan, D.; Phan, L.T.X.; Chakraborty, S.; Zimmermann, R.; Insup Lee; , "Video Quality Driven Buffer Sizing via Frame Drops," *Embedded and Real-Time Computing Systems and Applications (RTCSA)*
- [9] Q. Huynoh-Thu amd M. Ghanbari, "Scope of validity of psnr in image/video quality assessment," *Electronics Letters*, vol. 44, no.13, 2008