國立交通大學

電機學院 IC 設計產業研發碩士班

碩　士　論　文

應用於行動式視訊裝置之精簡化圖樣比對之嵌入式編解碼器

An Embedded Codec Based on Reduced Patterns Comparison for Mobile Video Devices

學生 ： 楊均宸

指導教授 ： 李鎮宜 教授

中華民國九十八年十二月

應用於行動式視訊裝置之精簡化圖樣比對之嵌入式編解碼器

An Embedded Codec Based on Reduced Patterns Comparison for Mobile Video Devices

研 究 生：楊均宸　　Student：Jun-Chen Yang

指導教授：李鎮宜　　Advisor：Dr. Chen-Yi Lee

國 立 交 通 大 學
電機學院 IC 設計產業研發碩士班
碩 士 論 文

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

Industrial Technology R&D Master Program on
IC Design

December 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年十二月

一應用於行動式視訊裝置之精簡化圖樣比對之嵌入式編解碼器

學生：楊均宸　　　　　　　　　　指導教授：李鎮宜 教授

國立交通大學電機學院產業研發碩士班

摘要

隨著多媒體、通訊系統與半導體製程的進步，行動式視訊裝置功能越來越強大，也造成有大量的資料需要傳送或儲存以及電池續航力的問題。然而有限的儲存元件、頻寬與電池容量的限制下，直接地限制了大部分行動式視訊裝置的應用，於是可以達成高品質及低儲存空間的高效率資料壓縮與解壓縮演算法是非常重要的。

我們提出一適合嵌入於行動式視訊裝置之有失真嵌入式壓縮器/解壓縮器來減少晶片與外部記憶體間所需的資料傳輸量，以達成減低頻寬使用、縮小對外部記憶體的空間需求以及降低能量消耗。

在我們提出的演算法中，是以位元平面截斷編碼(BPTC)與預先定義的位元平面(圖樣)所構成。在維持壓縮率為 2 的前提下，將一 4x2 的像素陣列，由 64 位元壓縮為 32 位元的封包。首先將一 4x2 陣列以位元平面截斷編碼找出起始平面，再根據我們設定的門檻來選擇使用圖樣比對編碼(PCC)或是一倍壓縮與二倍壓縮平均對剩下的位元平面作處理，最後封存編碼後之像數至外部記憶體。

我們提出的硬體架構可以在 100 MHz 的操作頻率下，支援每秒三十張的高解析度電視規格(HD 1080)以及 150 MHz 的操作頻率下，支援 H.264/SVC 規格下，

雙層每秒三十張的 HD720/HD1080。由於壓縮率固定為 2 倍，可輕易地轉換記憶體位址並支援動作補償單元(Motion Compensation)的亂數存取。壓縮一巨型區塊(Macro Block)需要 32 個週期，解壓縮一個巨型區塊(Macro Block)只需 16 個週期。對於記憶體的存取次數節省了將近 50%，降低了相當可觀的能量耗損。

An Embedded Codec Based on Reduced Patterns Comparison for Mobile Video Devices

Student: Jun-Chen Yang Advisor: Dr. Chen-Yi Lee

Industrial Technology R & D Master Program of
Electrical and Computer Engineering College
National Chiao Tung University

ABSTRACT

With the development of multimedia, communication system and semiconductor progress, the functions of mobile video applications are getting stronger and stronger, resulting in the problems on huge volume of data transmission, storage and battery endurance concerns. Under the constraints of limited storage components, bandwidth and battery capacity, most of the applications of mobile video devices are restricted. Therefore, the algorithm which can achieve high quality and little storage space with high efficiency of data coding and decoding is very important.

We propose a lossy embedded compressor/de-compressor which is suitable for embedding into mobile video devices to reduce the data transmission between chip and external memory, in order to reduce utilization of bandwidth, volume of external memory and power consumption.

In the proposed algorithm, we adopt Modified Bit Plane Truncation Coding (MBPTC) and Predefined Bit Planes (Patterns). Under the premise of compression ratio as 2, we compress one 4x2-pixel array from 64 bits to 32 bits into one packet. The Modified Bit Plane Truncation Coding (MBPTC) calculates the Start Plane (SP) of the

4x2-pixel array first. Then, selecting Patterns Comparison Coding (PCC) or 1x and 2x Average according to the coding threshold we setup to compress residual bit planes and last, packs the compressed pixels to external memory. The average PSNR loss of proposed algorithm is 5.98 dB.

The hardware architecture we proposed is able to support HD 1080@100 MHz of 30 frames per second for HDTV specification and HD 720/HD1080@150 MHz of 30 per frames second in double layers for H.264/SVC specification. Because the compression ratio is fixed as two, it is easy to re-map memory address and support random access of Motion Compensation (MC).

To compress a Macro Block takes 16 cycles while to decompress a Macro block takes only 16 cycles. It saves 48.7% of memory accesses on the average, leading to save considerable power consumption.

誌　　　　謝

還記得當初剛進交大在尋找指導教授時，老師願意收留後進，後來繞了一圈後進還是回歸在這裡，真的感謝老師還是願意收留學生。在交大的這段時間裡，在 SI2 Lab 的時光是後進在人生中最寶貴的！非常感謝李鎮宜教授對於後進的教導與栽培，讓後進在這段時間裡面獲益良多。老師，非常地謝謝您！

在這段時間裡，一路上受到許許多多人的幫助。感謝鍾崇斌教授及其高徒蔣昆成學長當初在生活上的幫助，感謝蔣迪豪教授、蔡文祥教授及其高徒李哲瑋學長於論文上的指點，以及黃聖傑助理教授於課業上的解惑。非常感謝我們 Multi-media Group 整個團隊，尤其是李曜學長與歐陽在研究領域上的協助與指導，讓後進研究過程的困難得以順利度過。義澤，沒有你的搞笑我哪有力挑燈夜戰呢？顏魚，沒有你的幫忙公式哪會這麼快出來呢？Arryz，感謝你在程式上的解答。Libra，感謝你時常的關心，以及浩民的反向鼓勵。義閔、柏均、人偉、洋蔥、小馬、佳龍、欣儒、建辰、博彥、H.D.以及 SI2 的每個成員們，還有子菁、伶霞、易蓁、美玲四位美麗的助理們，感謝這些有你們陪伴的日子。立偉、旭萍、柏頤、群旻、延曦、柏仁、裕昇、泰霖、肇廷、佩瑾、雅弦、文玲，感謝你們的支持。Sharon、KiKi、Jessie、Alice、Fay、Jamie，謝謝妳們的協助。BOGI、球、Navi、老猴、Ju、大熊，感謝你們所做的一切。

最後我要感謝我的家人以及這一路上我知道的、我不知道的人事物。還記得國小國語的課文《謝天》中讀到：「要感謝的人太多了，就感謝天吧！」

Index

Figures List

Tables List

Chapter 1 Introduction

1.1 Motivation

To improve the video coding efficiency, diminishing the data correlation of the temporal redundancy in each frame is widely used in the latest video coding standard, such as H.264/AVC [1]-[2]. However, it causes a large amount of data transmission between chip and external memory. In addition, the rapid and huge amount of data accesses from Motion Compensation (MC) consuming the majority of system power is another serious problem.

For a mobile video device, power consumption is the most critical issue that people concern about. Many low power techniques have already been proposed to reduce power consumption, but data transmission still dominates huge amount of system power. Hence, reduce data access between chip and external memory is the critical consideration in a mobile video device.

Although the mobile video devices are suffered from limited battery capability, the visual quality requirement is not as high as high resolution applications. Therefore, the embedded compression is suitable to lessen the volume of data access and the size of off-chip memory under the premise of maintaining acceptable visual quality.

The mobile video devices are more and more important due to their various functions at the present time. Reducing the usage of bandwidth and the required resource of hardware in the mobile video devices is a critical topic.

1.2 Thesis Organization

This thesis is organized as follows. In Chapter 2, we introduce the basic compression methods and review previous works. Chapter 3 explains the proposed lossy embedded compression algorithm. To integrate the proposed design into H.264/AVC and H.264/SVC decoders, there are some restrictions which must be specified. Under these restrictions, the proposed algorithm needs to be modified to fit for them, and the implementation and verifications of the proposed design are described in Chapter 4. The performance comparison and experimental results are shown in Chapter 5. Last, Chapter 6 gives the conclusion and future works.

Chapter 2
Previous Works

In general, the embedded compression methods are classified into two categories: lossless embedded compression and lossy embedded compression. The algorithms that have been proposed before are briefly described in Sections 2.1 and 2.2 and summarized in Section 2.3.

2.1 Lossless Compression Method

Many lossless compression methods have been proposed before. It is obvious that lossless compression methods [3] reserve the information while truncating the size of data, so there has no quality loss of data.

However, some problems of lossless compression are so fatal that it’s not suitable for system integration application. The lossless compression suffers from variable length of lossless compressed data that we cannot regularly control the compression ratio, frame memory size and bandwidth requirement. These disadvantages are also attributed to the needs of memory to prepare for the worst case of data access and the unknown size of data. For the reasons mentioned above, we decide to develop a lossy embedded compression method in this research.

2.2 *Lossy Compression Method*

There exists an important characteristic of lossy compression methods [4]-[15] which differs them from lossless compression methods. The characteristic of fixed compression ratio allows us to improve the disadvantages of lossless compression methods mentioned previously. Although lossy embedded compression algorithm will sacrifice tolerable visual quality, the reduced power consumption memory size and bandwidth requirement is more attractive for mobile video devices.

2.2.1 Transform-based Compression methods

The main function of transform-based compression methods is to convert the signal from time domain to frequency domain. In addition, transform-based compression methods can gather the energy to up-left corner. In human visual system, the lower frequency component is more important than the higher frequency component. It is a critical feature that we can employ to efficiently compress the amount of data, such as in [4]-[5].

In [4], the research utilizes the Hadamard transform and the quantization of Golomb-Rice Coding (GRC). Golomb-Rice Coding is an efficient compression scheme because it can provide the compression ability which approximates the Huffman Coding by selecting K factors. Hence, this paper focuses on low complexity so it fixes the K values based on simulation. It can be operated at 100 MHz and the processing cycle of a macroblock (MB) on embedded encoder/decoder is 16/16 cycles, respectively.

To improve the performance of [4], [5] proposed another transform-based compression method. It adopts Discrete Cosine Transform (DCT) instead of Hadamard Transform and Modified Bit Plane Zonal Coding (MBPZC) instead of Golomb-Rice Coding. The compression ability of Bit Plane Zonal Coding depends on distribution of “1” in each bit plane. In [5], it exploits a Variable Length Coding (VLC) codebook to improve the performance of BPZC. It can be operated at 100 MHz and the processing cycle in a macroblock (MB) on encoder/decoder is 72/34 cycles, respectively. Although [5] increases the value of PSNR, it also increases the overhead of hardware complexity and coding latency heavily.

2.2.2 Differential Pulse Code Modulation (DPCM) Compression

Theoretically, there exists approximate difference between each two neighboring pixels in original pictures. Therefore, Differential Pulse Code Modulation (DPCM) compression utilizes previous feature to achieve high compression efficiency, but also causes the strong data dependency over correlated data.

In [6], the research adopts DPCM as basic compression scheme. It improves the performance of DPCM compression through exploiting the intra prediction modes from H.264 video coding standard to search the best course. The modified DPCM is much more adopted than [4] and [5] in each video sequence.

However, DPCM compression needs to gather each difference into limited data budget and those differences are not always as small as we expect. Moreover, to derive the best performance, the DPCM-based compression method needs several repetitions to obtain the best quantization level for every difference. These disadvantages make the algorithm unable to exploit the pipeline scheme and lead to longer coding latency. In the view point of system, we need to increase the operation frequency or reduce the system throughputs to achieve the DPCM-based compression method. That's why we do not adopt it into our proposed algorithm.

2.2.3 Block Truncation Coding (BTC) Compression

The traditional Block Truncation Coding (BTC) [7] compression is a two-level non-parameter quantizer. As shown in Figure 1, BTC algorithm partitions an input image into several 4x4-sized blocks and each block is coded respectively. It computes the average value of pixels (AVG) within each block by (1). For each pixel with value is P_n within a block. First, the values of pixels greater than AVG are marked as "1" and the rest are marked as "0" to form the Bit-Map. Then, the average value of 1-marked and 0-marked pixels will be generated as High Eigen-value of Group (EOG) and Low EOG, respectively. Finally, each 4x4-sized block can be compressed into High-EOG, Low-EOG and Bit Map. The data coded format of BTC algorithm is shown in Figure 2.

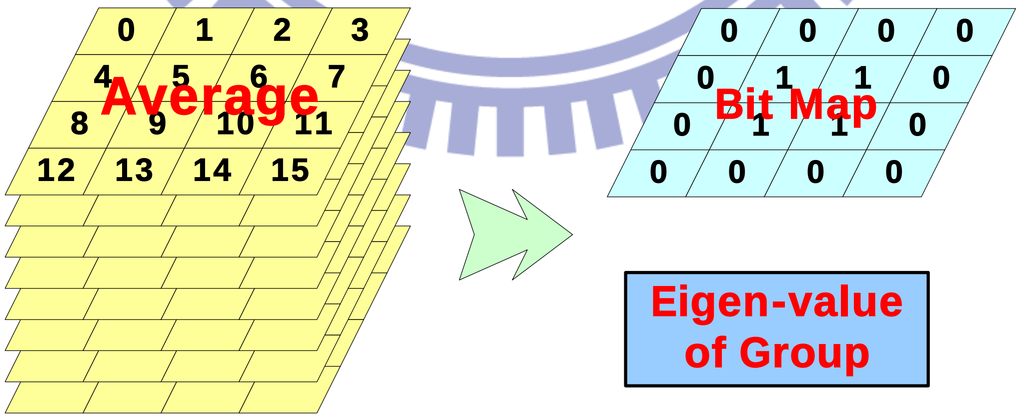


Figure 1: Block Truncation Coding (BTC) Algorithm

$$AVG = \frac{1}{16}\sum_{n=1}^{16} P_n \tag{1}$$

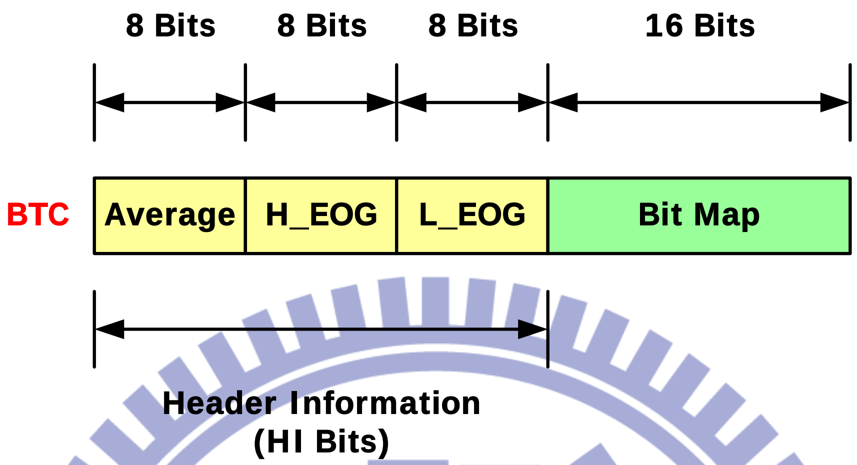


Figure 2: The Coded Format of BTC Algorithm

There exist hardware requirements and coding latency problems in BTC algorithm that it's not suitable to be embedded into the H.264/SVC system. In hardware design, multiplication and division operations will increase the requirements of hardware. Besides, the Bit-Map occupied 16 bits is another serious problem for limited compressed data budget.

Therefore, [8] proposed a method to improve the BTC algorithm with predefined bit planes as shown in Figure 3. To invert these 32 predefined bit planes, we can obtain the other 32 patterns. By comparing each Bit-Map with 64 predefined bit planes to get the highest similar index of predefined bit planes, we can reduce the amount of data. The coded format of BTC algorithm with 64 predefined bit planes is shown as Figure 4.

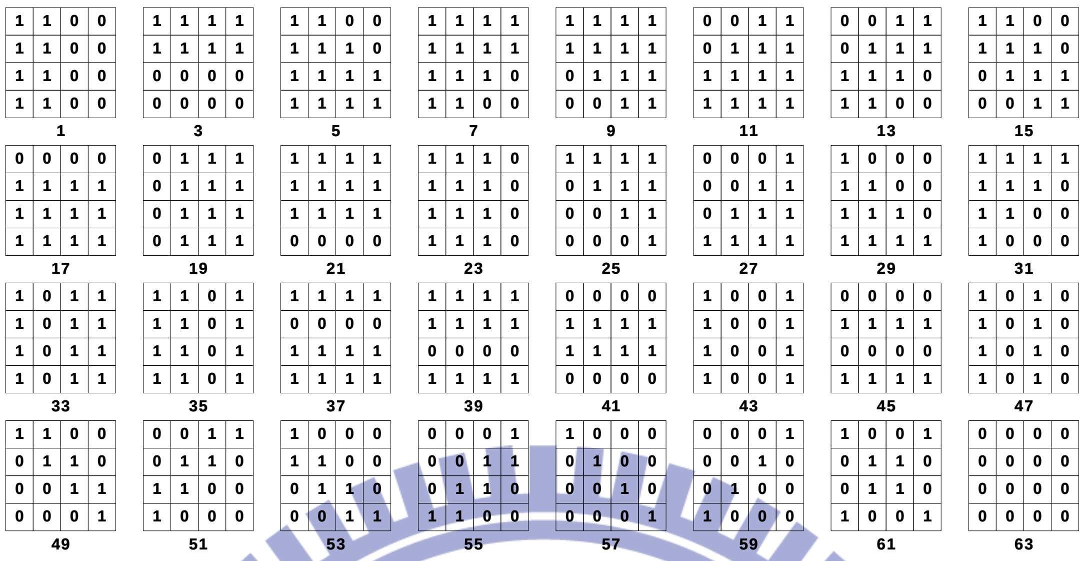


Figure 3: 64 Predefined Bit Planes [8]

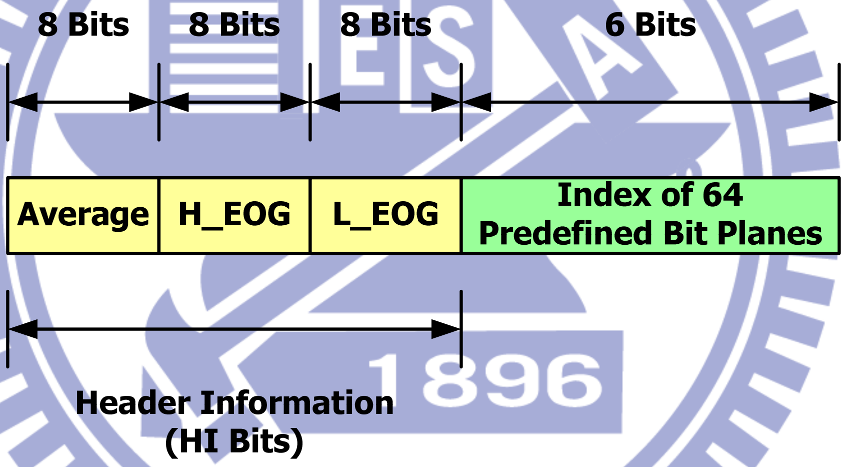

8 Bits
8 Bits
8 Bits
6 Bits
Average
H_EOG
L_EOG
Index of 64 Predefined Bit Planes
Header Information (HI Bits)

Figure 4: The Coded Format of BTC Algorithm with 64 Predefined Bit Planes [8]

However, comparing each Bit-Map with 64 predefined bit planes may increase the coding latency. In [9], it reduced the 64 predefined bit planes of [8] to 10 patterns as shown in Figure 5. By shifting and rotating, we can obtain the complete 32 predefined bit planes from 10 patterns. By reducing the number of predefined bit planes, [9] maintains acceptable visual quality to reduce the amount of data and coding latency. The coded format of BTC algorithm with 32 predefined bit planes is shown in Figure 6.

$$\underset{1}{\begin{array}{|c|c|c|c|}\hline 0&0&0&0\\\hline 0&0&0&0\\\hline 0&0&0&0\\\hline 0&0&0&0\\\hline\end{array}}\quad \underset{1}{\begin{array}{|c|c|c|c|}\hline 1&1&1&1\\\hline 1&1&1&1\\\hline 1&1&1&1\\\hline 1&1&1&1\\\hline\end{array}}\quad \underset{4}{\begin{array}{|c|c|c|c|}\hline 0&0&0&1\\\hline 0&0&0&1\\\hline 0&0&0&1\\\hline 0&0&0&1\\\hline\end{array}}\quad \underset{3}{\begin{array}{|c|c|c|c|}\hline 0&0&1&1\\\hline 0&0&1&1\\\hline 0&0&1&1\\\hline 0&0&1&1\\\hline\end{array}}\quad \underset{2}{\begin{array}{|c|c|c|c|}\hline 0&1&1&1\\\hline 0&1&1&1\\\hline 0&1&1&1\\\hline 0&1&1&1\\\hline\end{array}}$$

$$\underset{6}{\begin{array}{|c|c|c|c|}\hline 0&0&1&1\\\hline 0&0&0&1\\\hline 0&0&0&0\\\hline 0&0&0&0\\\hline\end{array}}\quad \underset{6}{\begin{array}{|c|c|c|c|}\hline 0&1&1&1\\\hline 0&0&1&1\\\hline 0&0&0&1\\\hline 0&0&0&0\\\hline\end{array}}\quad \underset{4}{\begin{array}{|c|c|c|c|}\hline 1&1&1&1\\\hline 0&0&0&0\\\hline 0&0&0&0\\\hline 0&0&0&0\\\hline\end{array}}\quad \underset{3}{\begin{array}{|c|c|c|c|}\hline 1&1&1&1\\\hline 1&1&1&1\\\hline 0&0&0&0\\\hline 0&0&0&0\\\hline\end{array}}\quad \underset{2}{\begin{array}{|c|c|c|c|}\hline 1&1&1&1\\\hline 1&1&1&1\\\hline 1&1&1&1\\\hline 0&0&0&0\\\hline\end{array}}$$

Figure 5: 10 Patterns for 32 Predefined Bit Planes [9]

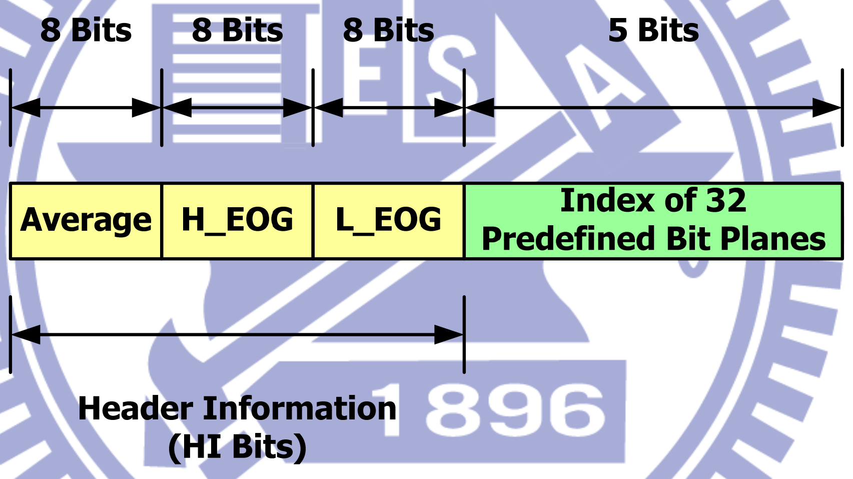
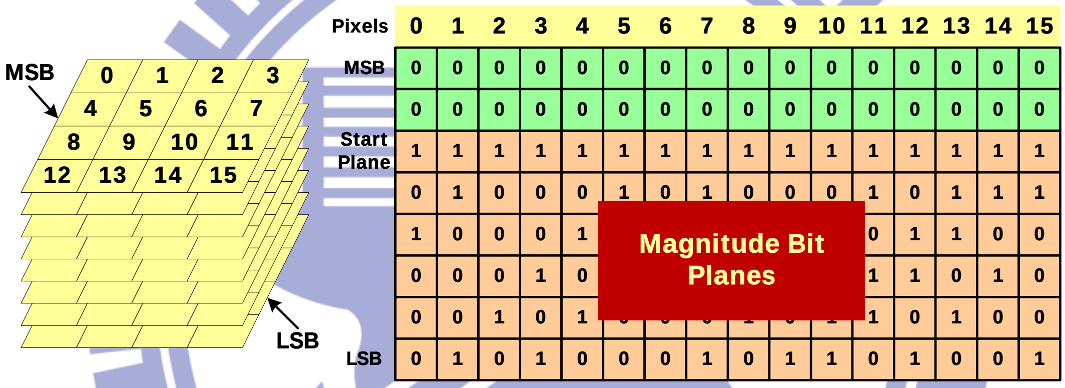

8 Bits
8 Bits
8 Bits
5 Bits
Average
H_EOG
L_EOG
Index of 32 Predefined Bit Planes
Header Information (HI Bits)

Figure 6: The Coded Format of BTC Algorithm with 32 Predefined Bit Planes [9]

Both [8] and [9] are limited by BTC algorithm; the coding latency is still too long to be well-embedded into the target H.264/SVC system. However, through [8] and [9], we find a way to utilize the predefined bit planes to reduce the coding latency and the amount of data.

2.2.4 Bit Plane Truncation Coding (BPTC) Compression

[7]-[9] imply the compression schemes from different viewpoints in [10]. As shown in Figure 7, BPTC algorithm partitions one 4x4-sized block into 8 bit planes first. Then, it starts to search for the Start Plane (SP) which is the highest plane without all-zero from MSB to LSB. Finally, BPTC algorithm will save Magnitude Bit Planes with 60 bits and truncate the rest data.



Pixels	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MSB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Start Plane	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0	1	0	0	0	1	0	1	0	0	0	1	0	1	1	1
	1	0	0	0	1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	1	1	0	0
	0	0	0	1	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1	1	0	1	0
	0	0	1	0	1	0	0	0	1	0	1	1	0	1	0	0
LSB	0	1	0	1	0	0	0	1	0	1	1	0	1	0	0	1

Figure 7: Bit Plane Truncation Coding (BPTC)

As shown in Figure 8, the BPTC algorithm is similar to lossless compression method, but the coding latency is shorter.

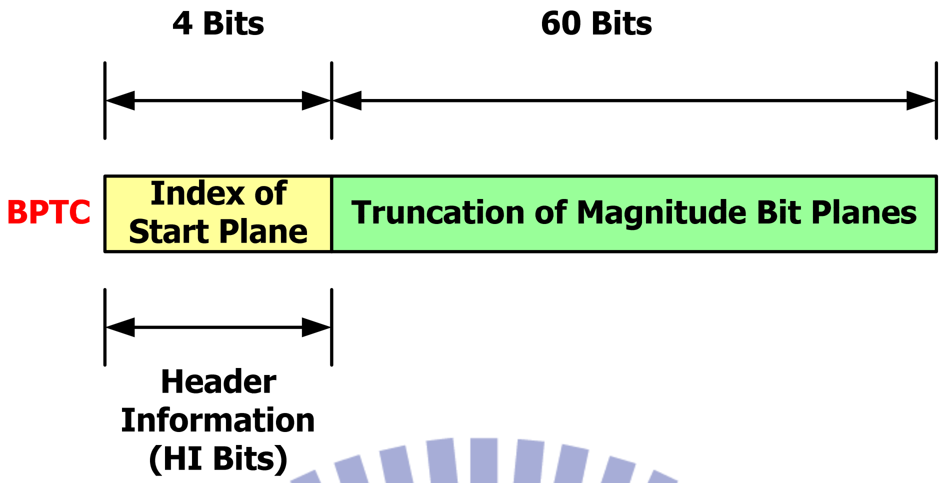

4 Bits
60 Bits
BPTC
Index of Start Plane
Truncation of Magnitude Bit Planes
Header Information (HI Bits)

Figure 8: The Coded Format of BPTC Algorithm

2.2.5 Other Compression Methods

There are more and more lossy compression methods proposed, such as the adaptive DPCM in [11], the adaptive Vector Quantization (VQ) in [12], and the down-sampling based compression scheme in [13]. Discrete Wavelet Transform (DWT) with Set Partitioning of Hierarchical Trees (SPIHT) in [14] provides another transform-based compression approach which is able to perform lossy and lossless with the same architecture. [15] proposes two lossy compression methods and adopts a pre-determining mechanism to select which one to utilize. It declares that the mechanism can reach better performance by selecting adaptive algorithm to suit for different video sequences.

According to the above discussion, no matter which one of lossy compression methods it is, the characteristic in common is to preserve the visual quality as complete as possible within limited data budget to avoid drift effect. Therefore, how to preserve complete visual quality under the restriction of maintaining compression ratio (CR) is the critical challenge of lossy compression methods.

2.3 Summary

In previous discussions, we classify the existing compression methods into two basic categories and introduce them briefly. In lossy compression methods, there are two main advantages of fixed compression ratio and the amount of coded data. However, higher visual quality usually accompanies with huge time consumption while low complexity brings lower visual quality. Most of preceding compression schemes derives higher performance from enlarging requirement of buffer and processing cycle, but it magnifies the obstruction to embed the extra function into a H.264 system. Even though slowing the operation frequency is able to fix prior problem, the former compression schemes still raised other problems such as decreasing coding throughput.. It is easy to be embedded into a H.264 decoder system that some of lossy compression methods are low complexity and coding latency.

For real time and mobile video devices, low latency and power consumption are the basic requirements. Diminish the overhead from original system is another objective. Therefore, an optimal trade-off among low latency, low power consumption, low complexity and high performance is our design challenge on embedded compressor/de-compressor.

Chapter 3
Proposed Algorithm

3.1 Overview

Data compression has been developed for a long time. From those proposed algorithms, we know that enhancing the complexity can obtain high performance. According to the restrictions of embedding an extra function into H.264 decoder system in chapter 2, low latency and power consumption are the basic requirements. In this chapter, we will discuss those in detail.

Actually, because blocked-based compression methods match the block-oriented structure in H.264 system, they are the most compatible schemes. However, overhead is another serious problem in H.264/SVC system embedment. The overhead is defined as follows: the ratio between the number of pixels that are actually accessed during the motion compensation of a block and the number of pixels that are really useful in the reference block [5]. Originally, the ratio is always 1 for the pixels accessed on demand. In the system with block-based algorithms, the ratio is always superior to 1 due to the individuality of block-based compression method. The notion of pixel-based and block-based compression methods are shown in Figure 9.

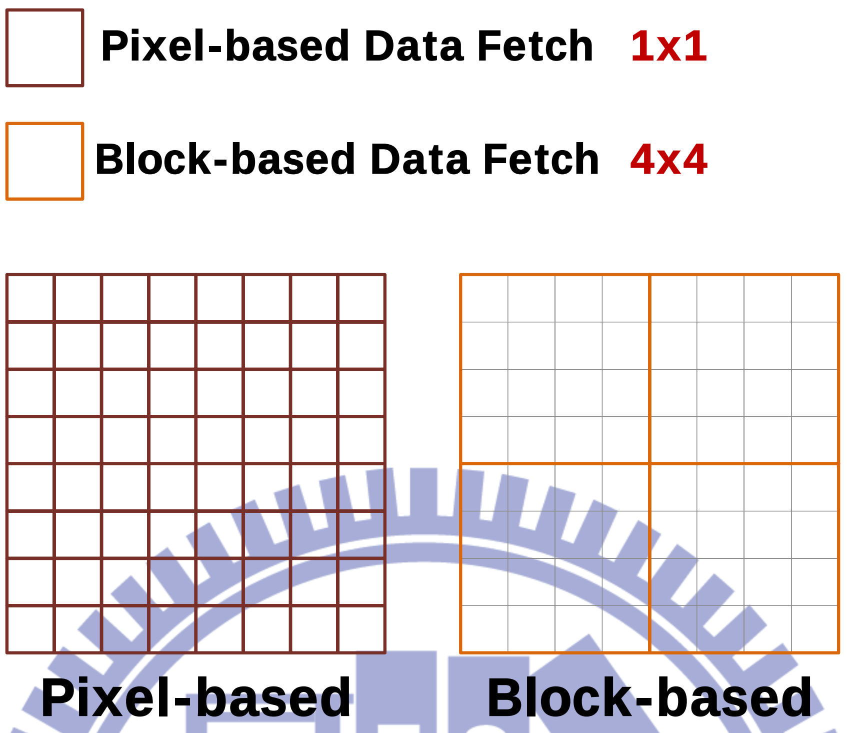


Figure 9: Pixel-based and Block-based

In the standard of H.264, a 16x16-sized macro block (MB) can be partitioned into several 8x8, 8x16 or 16x8 blocks during the process of Motion Compensation (MC). In addition, an 8x8 block can be sub-partitioned into 4x4, 4x8 or 8x4-sized sub-blocks in advance. An example of overhead is shown in Figure 10. The overhead is occurred while the compensated block is not aligned on the coded block grid. The four coded blocks need to be loaded and then decoded to derive the required data. It must be loaded 256 pixels to obtain the 16 required pixels if the compensated block is 4x4-sized block and the EC method is 8x8-sized block-based. In this example, the overhead is 16 cycles. The relativity between gain of memory access and compression ratio of EC is not direct due to the overhead.

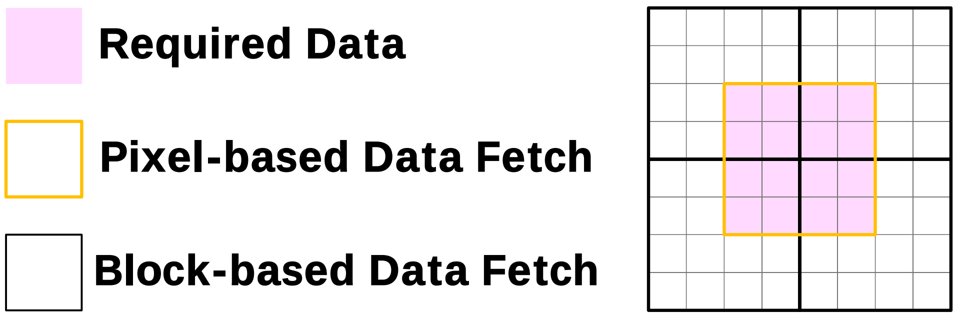


Figure 10: An Example of Data Fetching Overhead

In [15], it provides the statistic material of overhead. The relativity between overhead and coding bit-rate simulation with Stefan Sequence in three kinds of EC block-grid is shown in Figure 11. Table 1 summarizes the statistical result simulated with six sequences.

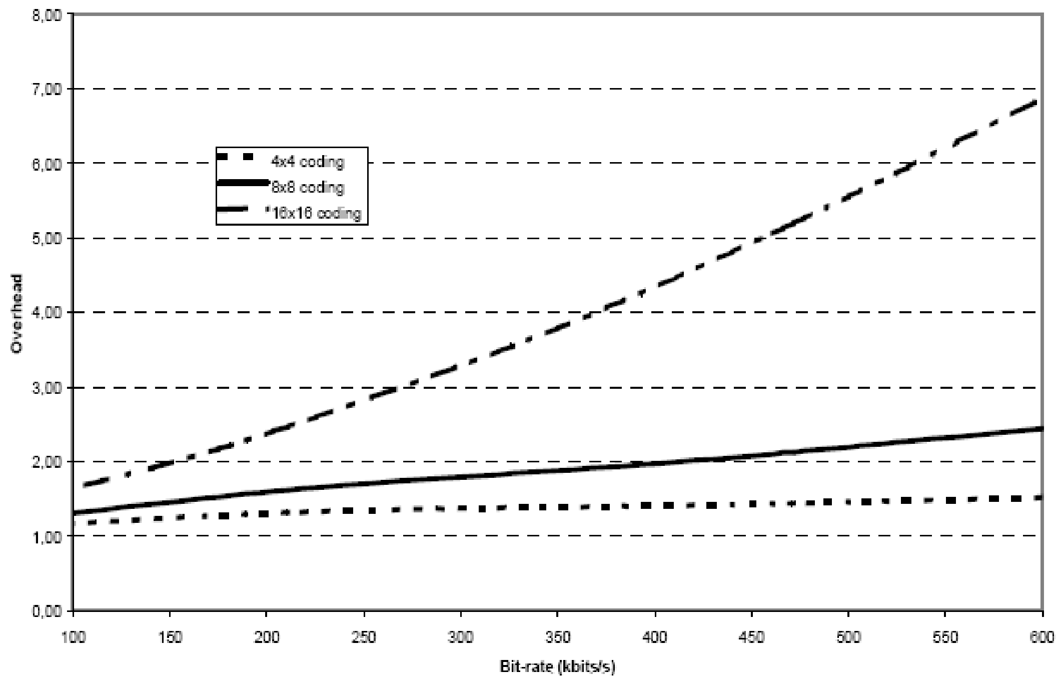


Figure 11: Relativity between Overhead and Coding Bit-rate Simulation (Stefan Sequence)

Table 1: Overhead with EC Block-grid for Each Sequence

Sequence	4x4 Block-grid	8x8 Block-grid	16x16 Block-grid
Foreman	1.31	1.77	3.69
Flower	1.30	1.74	3.77
News	1.14	1.51	2.78
Silent	1.17	1.50	3.22
Stefan	1.51	2.44	6.95
Weather	1.17	1.49	3.18
All (Average)	1.27	1.73	3.93

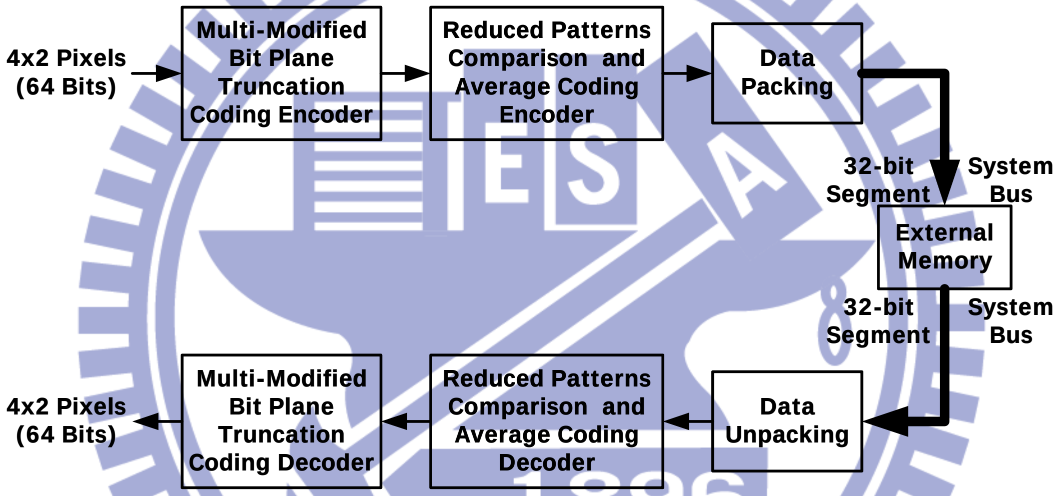
3.2 *Algorithm of Embedded Compression*

Under the restriction of mobile video devices, we know that low latency and power consumption are the basic requirements. Although utilizing transform-based algorithms, such as [4], [5] and [14], as the first step before compression methods can derive higher visual quality, we need to adopt other algorithms to lessen the coding latency.

We adopt pattern-based and 4x2 block-grid as the proposed coding algorithm including Modified Bit Plane Truncation Coding (MBPTC) and Patterns Comparison Coding (PCC). The main reason relies on the lower overhead than the statistical results described in previous section. In MBPTC algorithm, we define to search the Start Plane (SP) in 4 continuous layers which are close to MSB with 2 bits. We also change coding unit to improve the PCC algorithm.

To combine MBPC with PCC is the basic notion of the proposed algorithm. Modified Bit Plane Truncation Coding (MBPTC) and Reduced Patterns Comparison

Coding (RPCC) algorithm are the critical features. No matter Multi-MBPTC or RPCC is considerably efficient and suitable to be utilized in software and hardware. The flow chart of proposed algorithm is shown in Figure 12. There contains no iteration in the proposed algorithm which is a single-way and open-loop coding method. The bit planes are coded by Multi-MBPTC to search the SP. Then the remaining bit planes are coded by RPCC to derive the Pattern Indices (PIs). Development of the proposed algorithm will be introduced in the following sections.



4x2 Pixels (64 Bits)
Multi-Modified Bit Plane Truncation Coding Encoder
Reduced Patterns Comparison and Average Coding Encoder
Data Packing
32-bit Segment
System Bus
External Memory
32-bit Segment
System Bus
Data Unpacking
Reduced Patterns Comparison and Average Coding Decoder
Multi-Modified Bit Plane Truncation Coding Decoder
4x2 Pixels (64 Bits)

Figure 12: Flow Chart of Proposed Embedded Compression

3.2.1 Fully Patterns Comparison Coding

As shown in Figure 13, fully Patterns Comparison Coding algorithm includes MBPTC and 2x2-based PCC. Because PCC can preserve and compress five bit planes, we just compress the residual bit planes. Therefore, we adopt MBPTC algorithm to compress the 4 bit planes which are close to MSB.

There are 65536 (2^{16}) kinds of patterns in each 4x4 bit planes. [8] and [9] employ

64 and 32 patterns to compress the amount of data, respectively. The more quantities of patterns there is, the better performance there will be. However, it will make comparison more difficult. Therefore, the 2x2-based patterns are proposed which are shown in Figure 14 to improve the problem. A bit plane is partitioned into four 2x2-sized parts and compared with them with the proposed 8 patterns at the same time. As the simulation result shown in Figure 15, while the error rate occurred, there is a 1-bit error in a part at most. We exploit previous method to extend the quantities of patterns close to 4096 (2^{12}) categories in each bit plane. Hence, we reduce the coding latency to 4 cycles and maintain acceptable visual quality under CR as 2. We show the coded format of data for Fully Patterns Comparison Coding in Figure 16. PSNR loss of this algorithm is 8.18 dB for simulation in H.264 system with the following video sequences: akiyo, flower, football, foreman, mobile_calendar, carphone, canoa, coastguard, waterfall and tempete in CIF format.

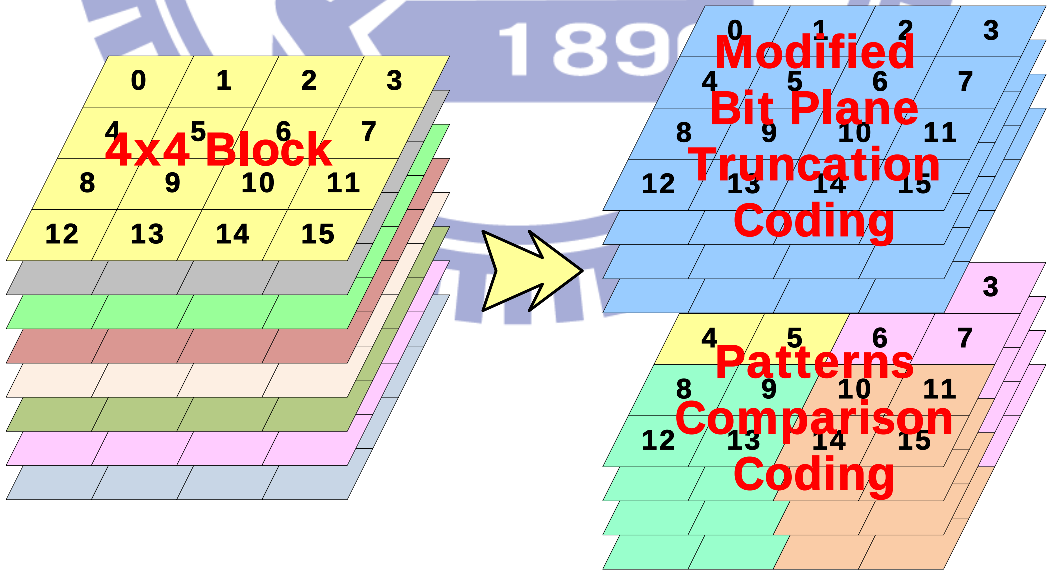

4x4 Block
Modified Bit Plane Truncation Coding
Patterns Comparison Coding
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Figure 13: Fully Patterns Comparison Coding Algorithm

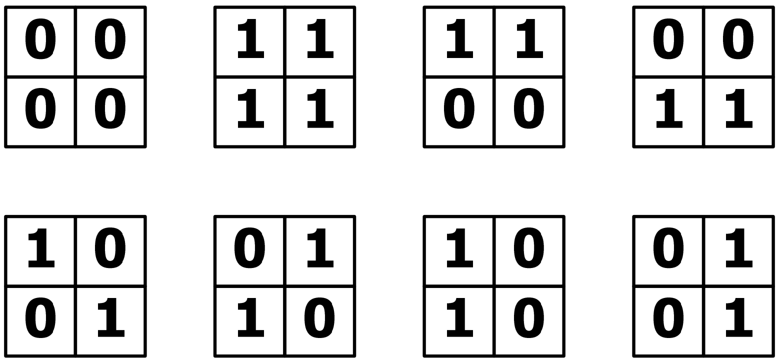


Figure 14: 8 2x2-based Patterns

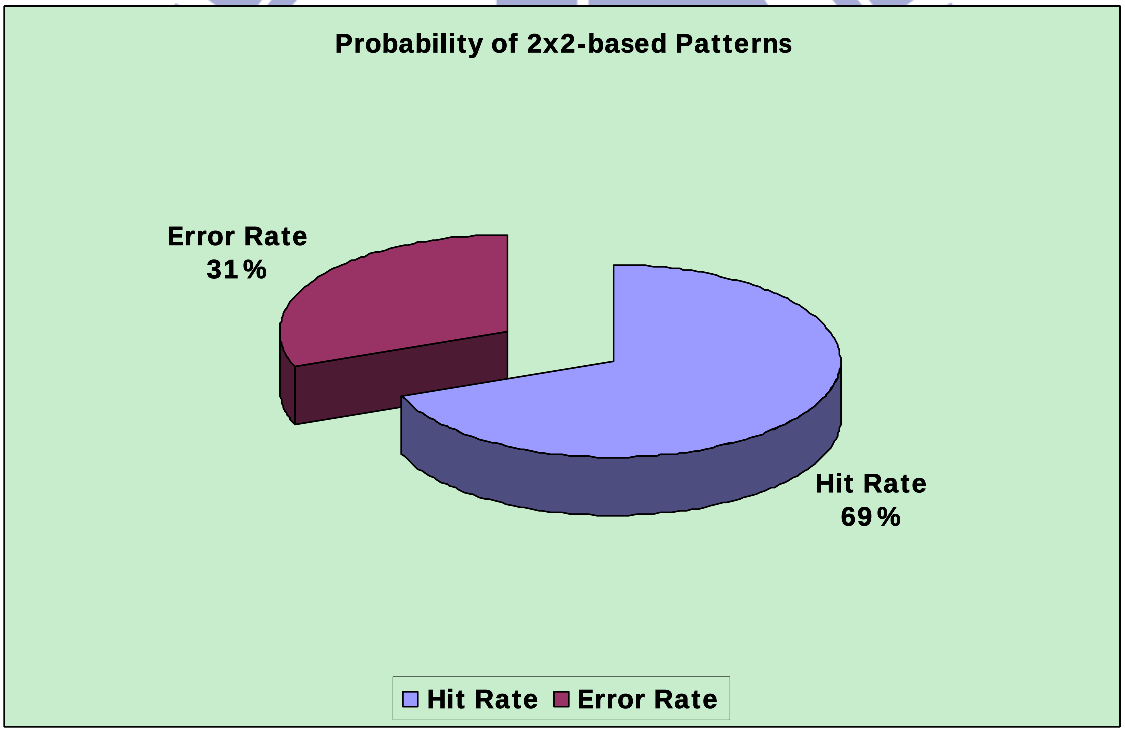


Figure 15: Probability of 2x2-based Patterns

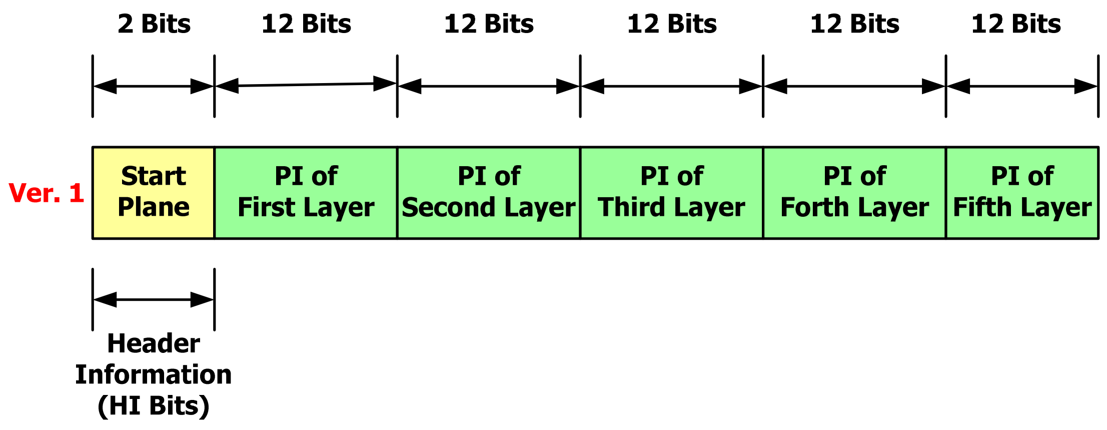


Figure 16: The Coded Format of Fully Patterns Comparison Coding Algorithm

3.2.2 Patterns Comparison and Average Coding

Although Fully Patterns Comparison Coding algorithm provides acceptable visual quality, it is a serious problem that the frames are not very smooth. That is because the 2x2-based patterns cannot contain all cases in a sequence causes. Hence, we exploit the average coding to smooth the frames and show it in Figure 17. We partition the four bit planes which are close to MSB into four parts and employ 2x2-based PCC algorithm to compress them to obtain the PIs. Then we partition the residual bit planes into four parts and compute the average value with 4 bits in each part. The coded format of data for Patterns Comparison and Average Coding Algorithm is shown in Figure 18. According to the simulation result, PSNR loss is 7.80 dB in this algorithm.

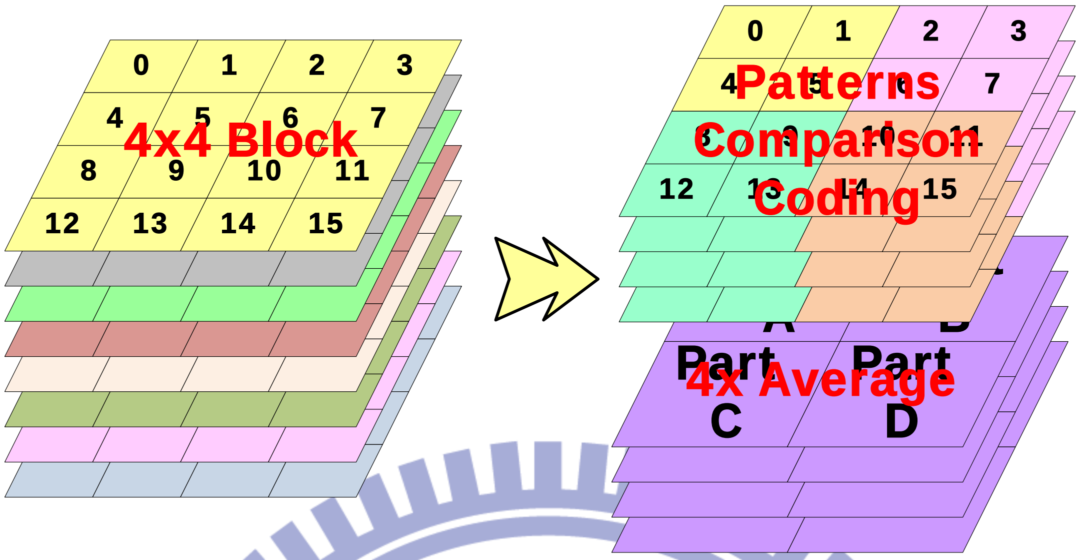


Figure 17: Patterns Comparison and Average Coding Algorithm

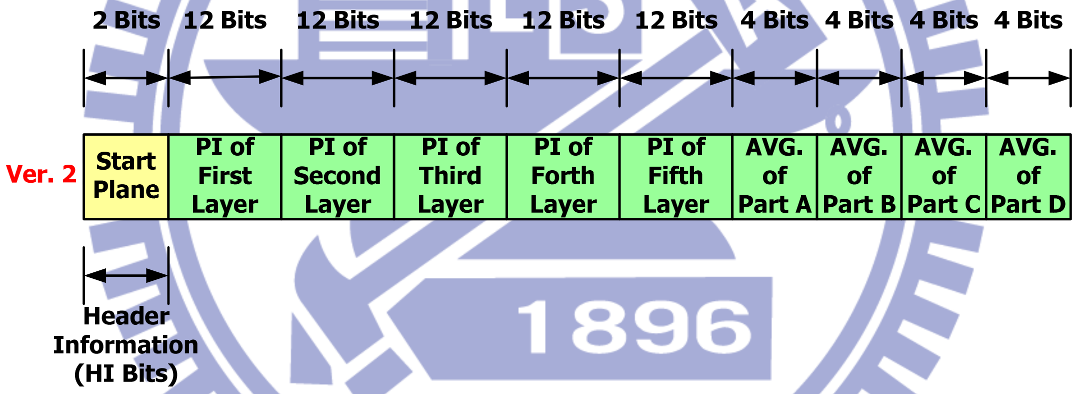
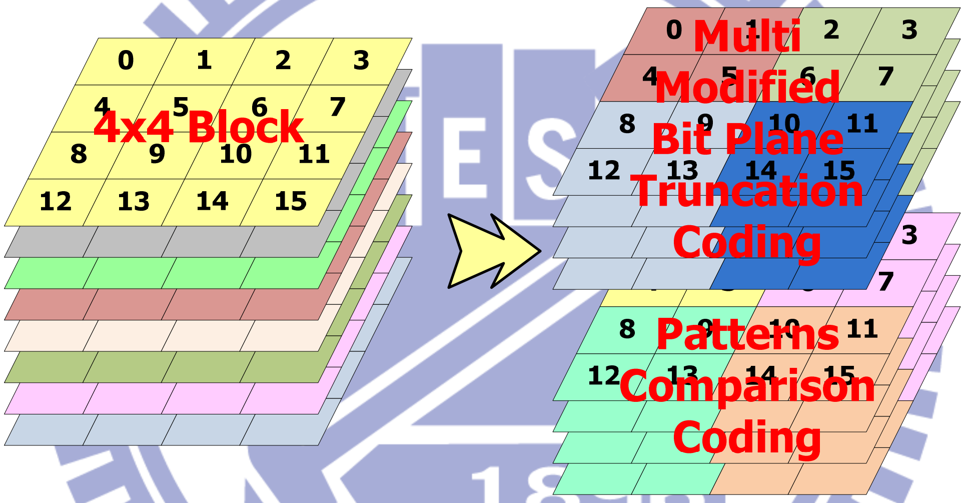


Figure 18: The Coded Format of Patterns Comparison and Average Coding Algorithm

3.2.3 Patterns Comparison and Multi-MBPTC Coding

With the respect of system, the higher the coding latency is, the higher overhead there will be. Therefore, we still need to reduce coding latency. According to Table 1, we know the smaller block-grid there is, the less overhead there will be. Hence, we exploit the feature to reduce coding latency and adopt the 4x2 block-grid. Although the 2x2-based PCC algorithm is suitable for 4x2 block-grid, the MBPTC algorithm needs to be modified. We propose the Multiple Modified Bit Plane Truncation Coding (Multi-MBPTC) algorithm to replace with MBPTC algorithm. In Figure 19, we divide a 4x4 block into four 2x2-sized parts which are partitioned into eight bit planes in advance. Multi-MBPTC algorithm searches the SPs in each part which is close to MSB and the 2x2-based PCC algorithm compress each part which is close to LSB. The coded format of data for Patterns Comparison and Multi-MBPTC Coding algorithm is shown in Figure 20. Although the coding latency is lessened, the PSNR loss is also enlarged to 8.95 dB.



0 1 2 3
4 5 6 7
8 9 10 11
12 13 14 15
4x4 Block
Multi Modified Bit Plane Truncation Coding
Patterns Comparison Coding

Figure 19: Patterns Comparison and Multi-MBPTC Coding Algorithm

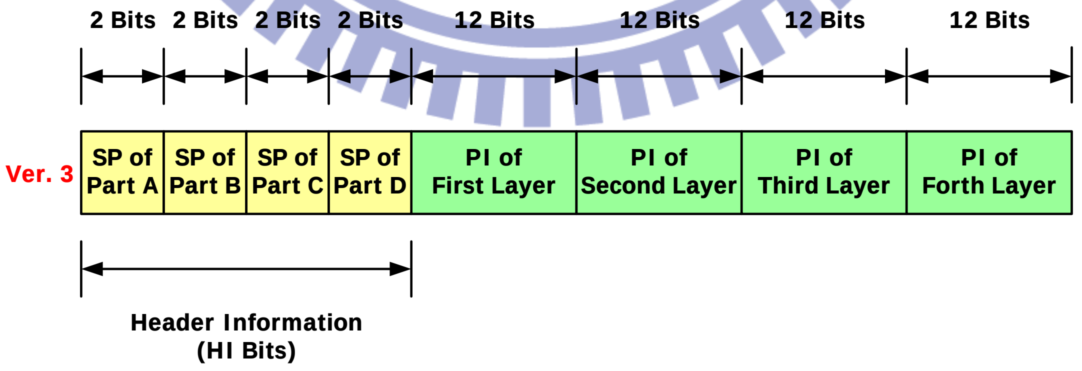


Figure 20: The Coded Format of Patterns Comparison and Multi-MBPTC Coding Algorithm

3.2.4 Reduced Patterns Comparison and Average Coding

In previous algorithms, we know that Multi-MBPTC algorithm can reduce the coding latency and average coding can smooth the frames. However, to overuse Multi-MBPTC algorithm will reduce visual quality. Therefore, we propose a 4x1-based PCC algorithm which is combined with average coding, as called Reduced Patterns Comparison Coding (RPCC) algorithm to improve previous disadvantages.

In practice, the higher the hit rate is, the higher PSNR we get. Additionally, according to simulation result, it achieves higher hit rate in 4x1-based patterns as shown in Figure 21. Thus, we adopt 4x1-based patterns to modify the PCC algorithm.

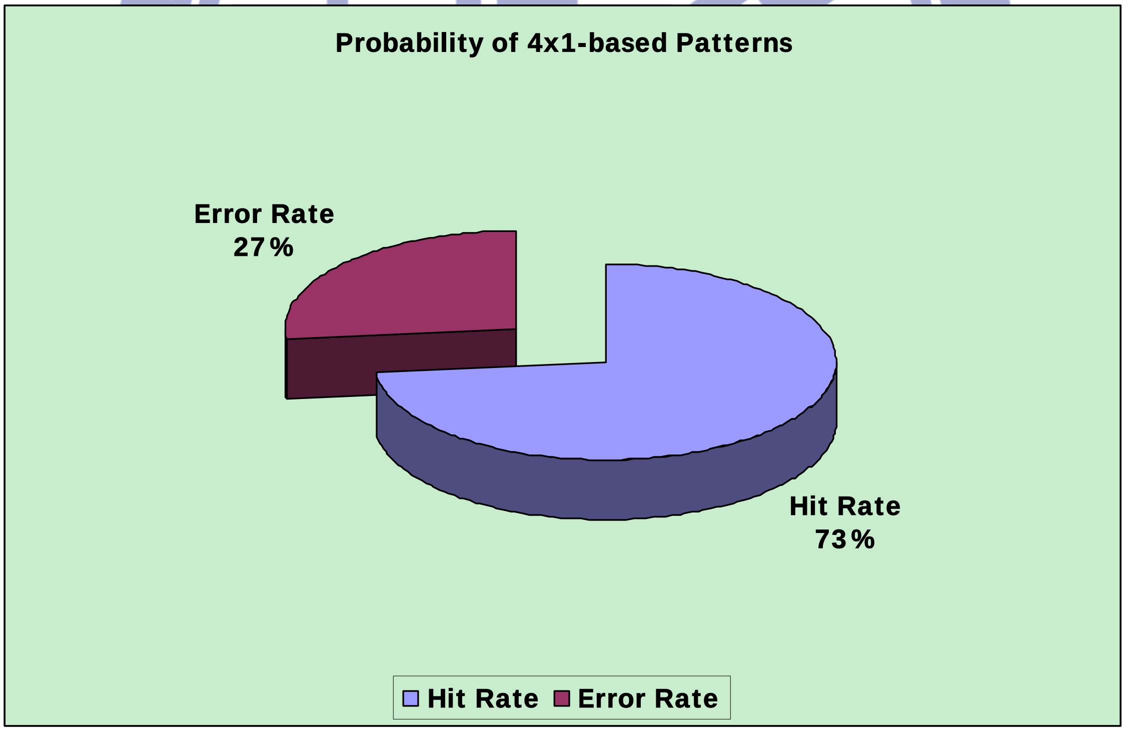


Figure 21: Probability of 4x1-based Patterns

As Figure 22, we partition 4x1 section with 4 bits into four 4x1 layers. According to the threshold, RPCC algorithm will adopt left or right strategy. While we set the threshold to level 2, RPCC algorithm compares layer 1 and layer 2 with eight 4x1-based patterns, as shown in Figure 23, at the same time. If there is no error in layer 1 and layer 2, RPCC algorithm will adopt left strategy to compress the 4x1 section. Otherwise, RPCC algorithm will adopt right strategy. According to the simulation result with different thresholds, while the right strategy is adopted, the right strategy is often in worse case. We exploit this feature to improve the drawback in 4x1-based PCC algorithm.

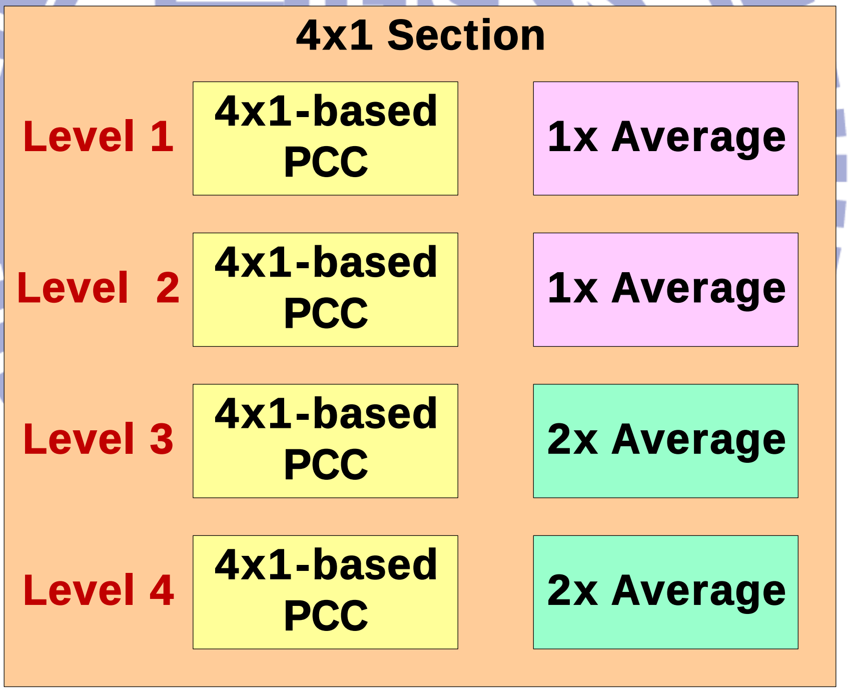


Figure 22: Reduced Patterns Comparison Coding Algorithm

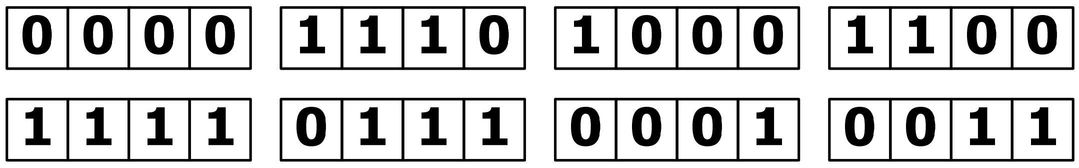


Figure 23: Eight 4x1-based Patterns

The proposed algorithm is including Multi-MBPTC, RPCC and average coding algorithm. As shown in Figure 24, we partition a 4x2 block into eight bit planes which are coded by MBPTC algorithm to derive SP. Then RPCC algorithm compresses four bit planes which are closed to SP and divided into two 4x1 sections. Finally, we compute the average value of residue after RPCC algorithm coding in each 2x2 part with 2 bits. The coded format of data for the proposed algorithm is shown in Figure 25. Strategy bits which indicate the compression strategy of each 4x1 part are coded by RPCC algorithm. The proposed algorithm reduces PSNR loss to 5.98 dB and maintains the 4x2 block-grid.

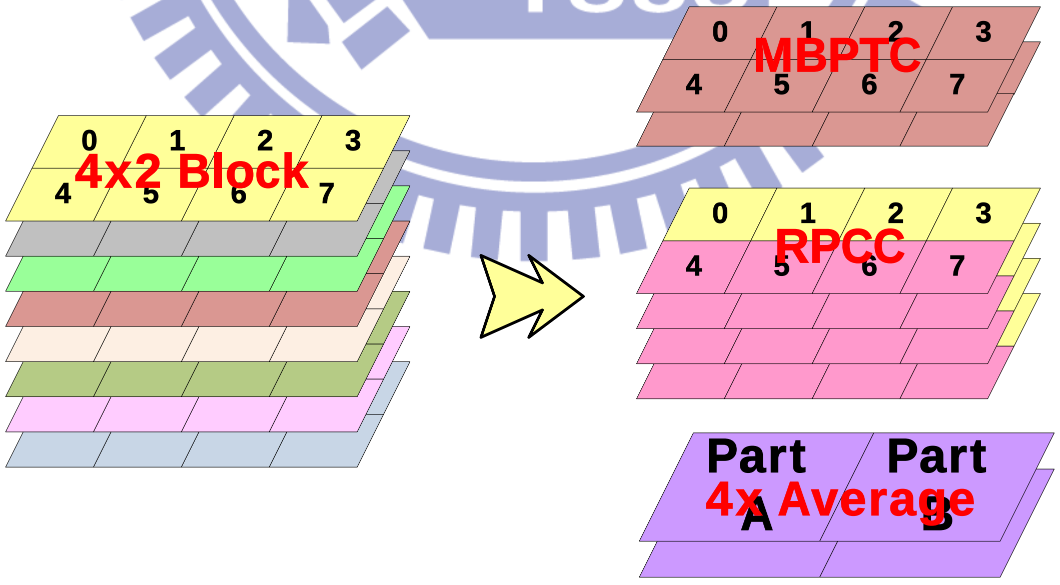


Figure 24: Reduced Pattern Comparison Coding Algorithm

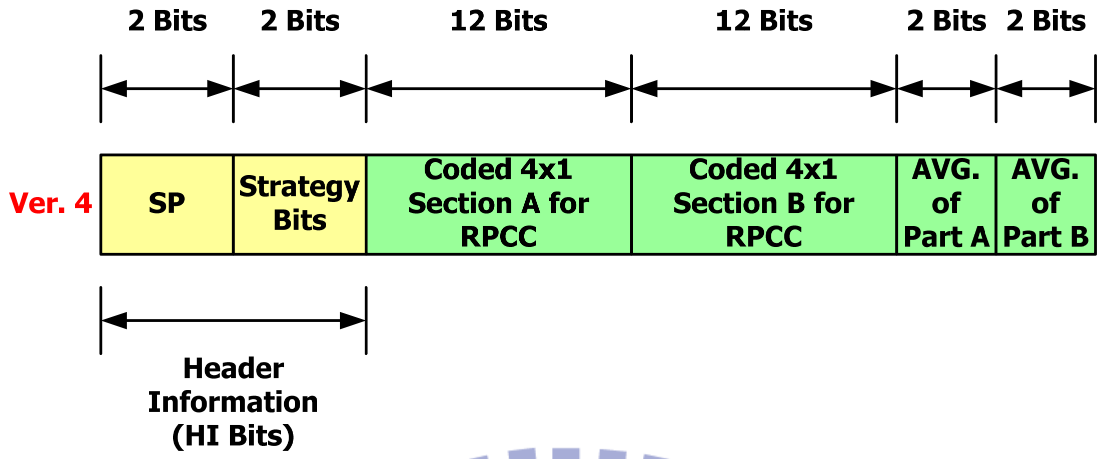


Figure 25: The Coded Format of Reduced Patterns Comparison Coding Algorithm

3.2.5 Formula

We derive the formula (2) from the simulation result. It is about the PSNR loss of 4x1-based PCC algorithm. *i* is the number of 4x1 error bit plane. P_m is the error rate shown in Figure 21 and P_n is the ratio of error rate per position in each 4x1 bit plane as described in Table 2. As described in the previous section, we can setup the different thresholds (Level 0~4) in 4x1-based PCC algorithm to get corresponding weight (W_i) as described in Table 3. We can exploit the formula to estimate for the PSNR loss in 4x1-based PCC algorithm while the previous parameters are modified. Figure 26 shows the distribution of PSNR loss in all thresholds. It helps us to simplify the improving procedure in the algorithm.

$$\textbf{PSNR Loss (4x1-based PCC)} = \sum_{i=0}^{4} \left[C_i^4 (P_m \cdot P_n)^i \cdot (1 - P_m \cdot P_n)^{4-i} \right] \cdot W_i \qquad (2)$$

Table 2: Ratio of Error Rate per Position in each 4x1 Bit Plane

P_n	Error Rate (%)	Total Ratio (%)
P_0	8.68	32.54
P_1	4.65	17.43
P_2	4.65	17.43
P_3	8.70	32.60

Table 3: Weight Under Different Thresholds

W_i	Level 0	Level 1	Level 2	Level 3	Level 4
W_0	0	0	0	0	0
W_1	240	112	48	16	0
W_2	720	224	48	0	0
W_3	720	112	0	0	0
W_4	240	0	0	0	0

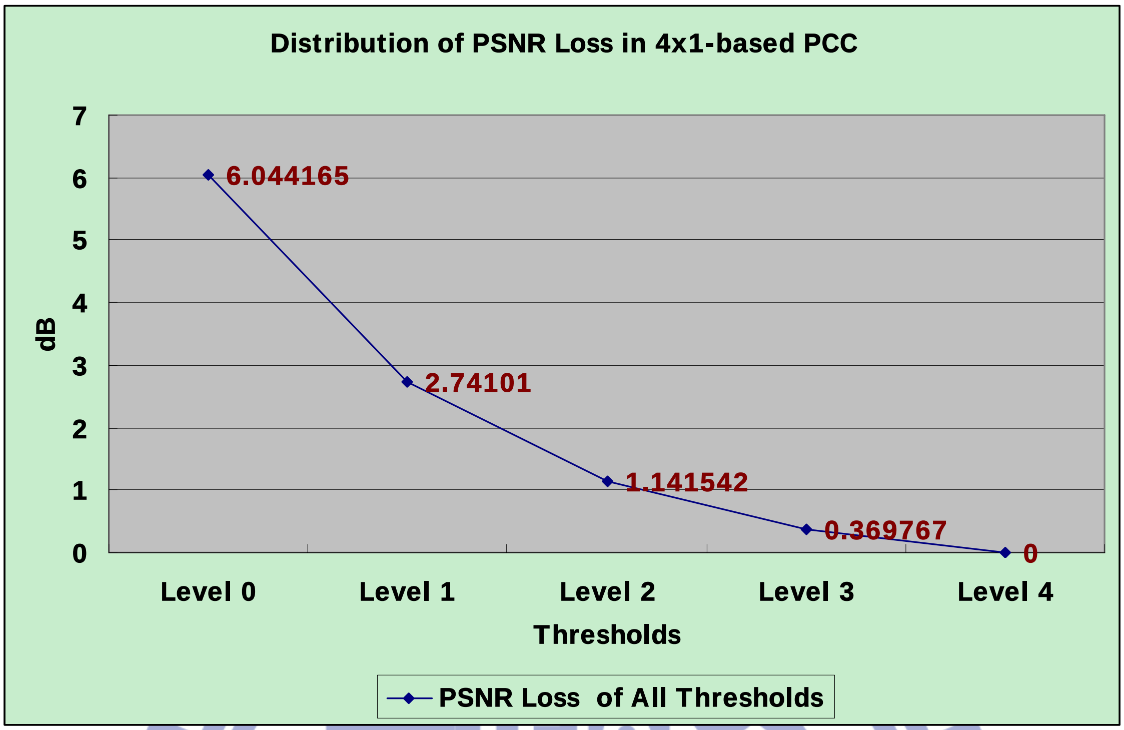


Figure 26: Distribution of PSNR Loss in 4x1-based PCC Algorithm

3.3 *Summary*

In this section, we show all results and focus on comparing the visual quality with those above-mentioned algorithms. Table 4 shows the comparison results. Figure 27 is the worst case and Figure 28 is the best case in different algorithms.

In Figure 27, there are many bright spots in the coded sequence of fully Patterns Comparison Coding (PCC) algorithm. Although the value of PSNR is acceptable, the visual quality is not enough to accept. Therefore, we employ average coding instead of MBPTC to smooth the frames as shown in Figure 27. Even though it lessens the PSNR loss and smoothes frames, we still cannot clearly to recognize the details in frames. Multi-MBPTC algorithm provides a method in recognition of edge. Nevertheless, to overuse the Multi-MBPTC algorithm will enlarge PSNR loss heavily. Thus, the algorithm we proposed is to combine the advantages of these previous algorithms.

With the respect of system, we adopt 4x2 block-grid to reduce coding latency further. Fig. 27 shows the details of the proposed algorithm in frames which are clearer than those previous methods.

Table 4: Comparison of Previous Algorithms

Algorithms	PSNR Loss (dB)
[8]	13.63
[9]	16.46
Fully Patterns Comparison Coding	8.18
Patterns Comparison & Average Coding	7.80
Patterns Comparison & Multi-BPTC Coding	8.95
Proposed	5.98

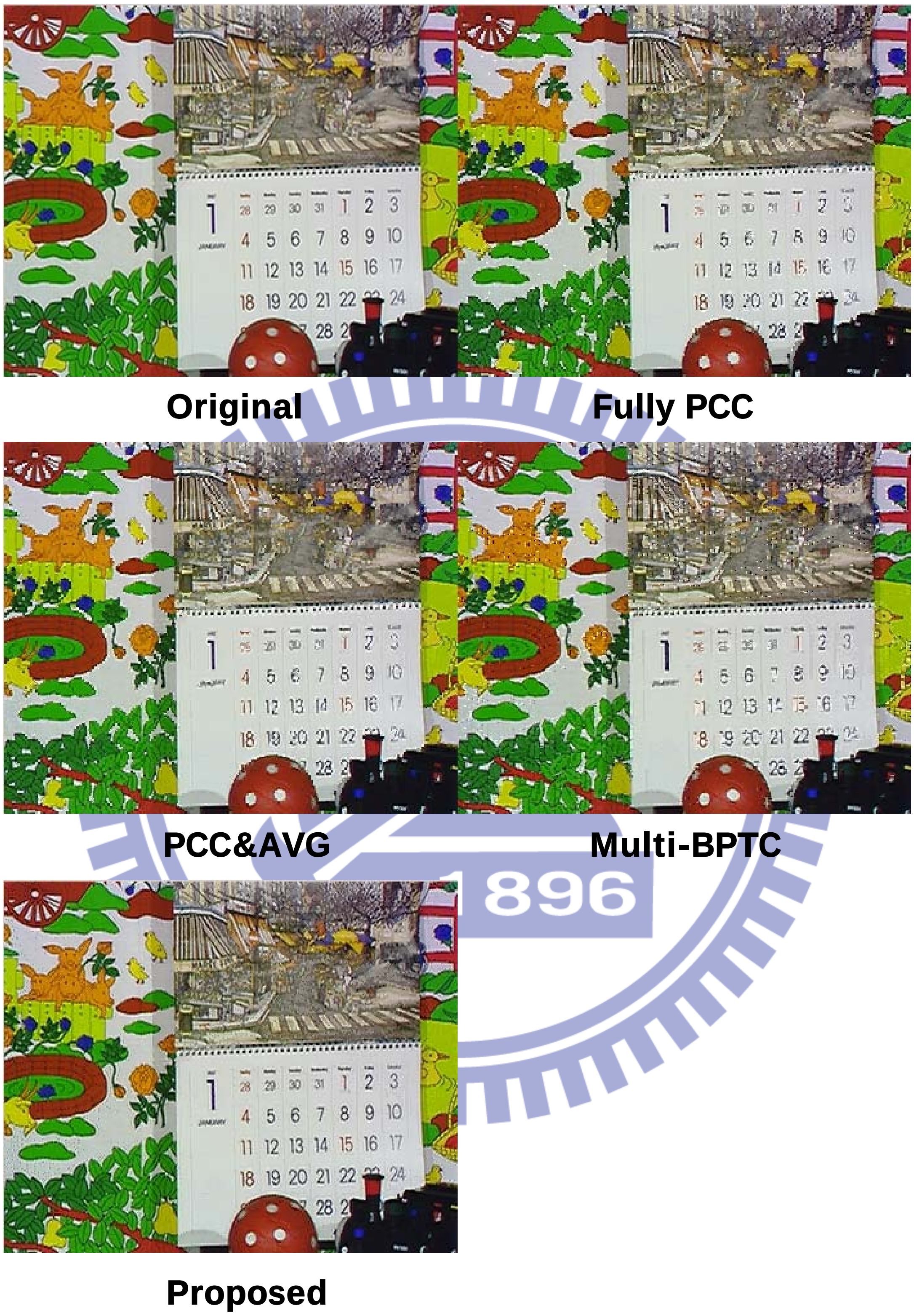


Figure 27: Worst Case in Different Algorithms

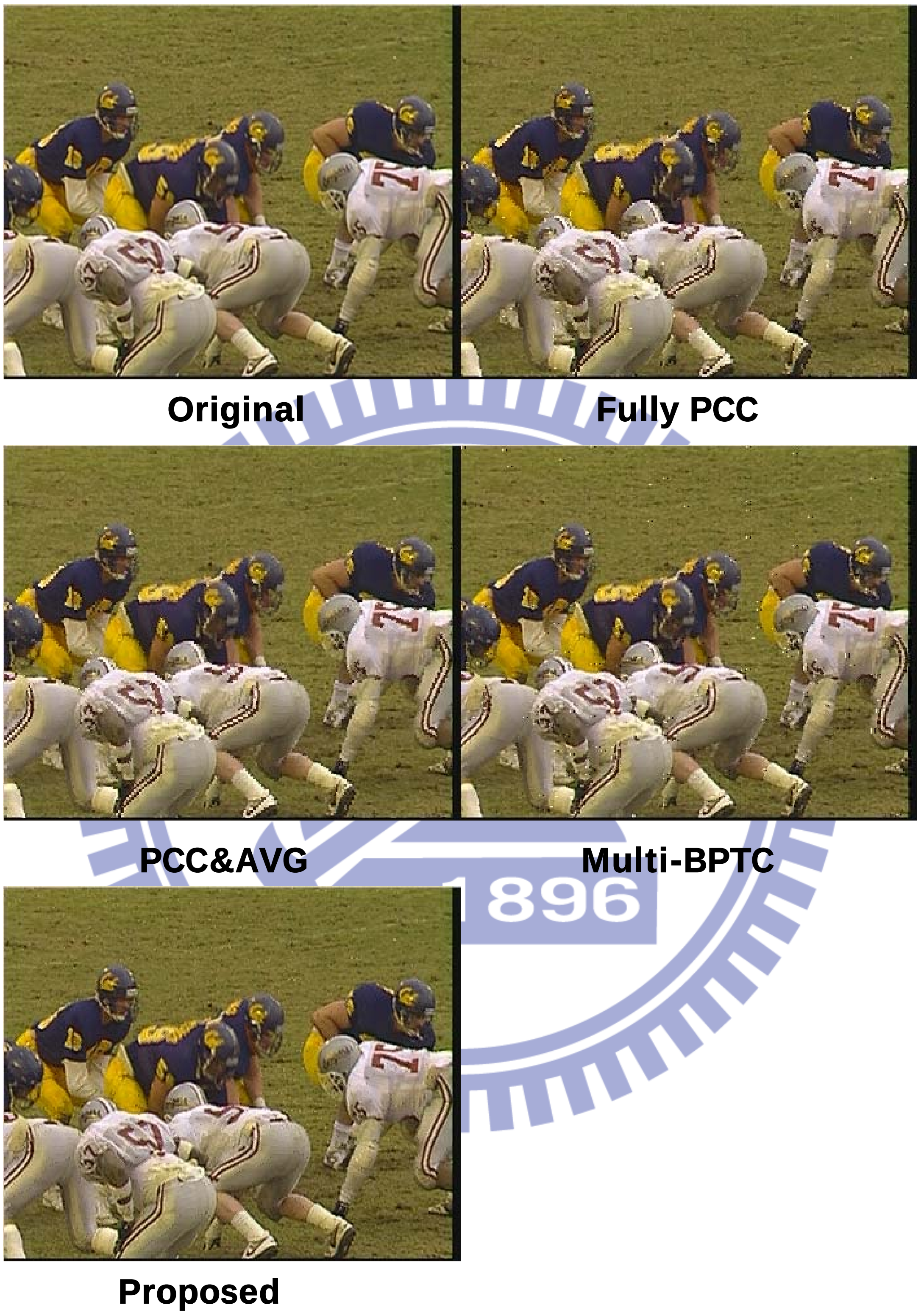


Figure 28:Best Case in Different Algorithms

Chapter 4
Proposed Architecture

In sections 4.1 and 4.2, the hardware design of the proposed embedded compressor and de-compressor are introduced separately. The implementation and verification are described in section 4.3.

4.1 Architecture of Encoder

The overall diagram of the proposed embedded compressor is shown in Figure 29 and we will introduce these blocks in the following sections.

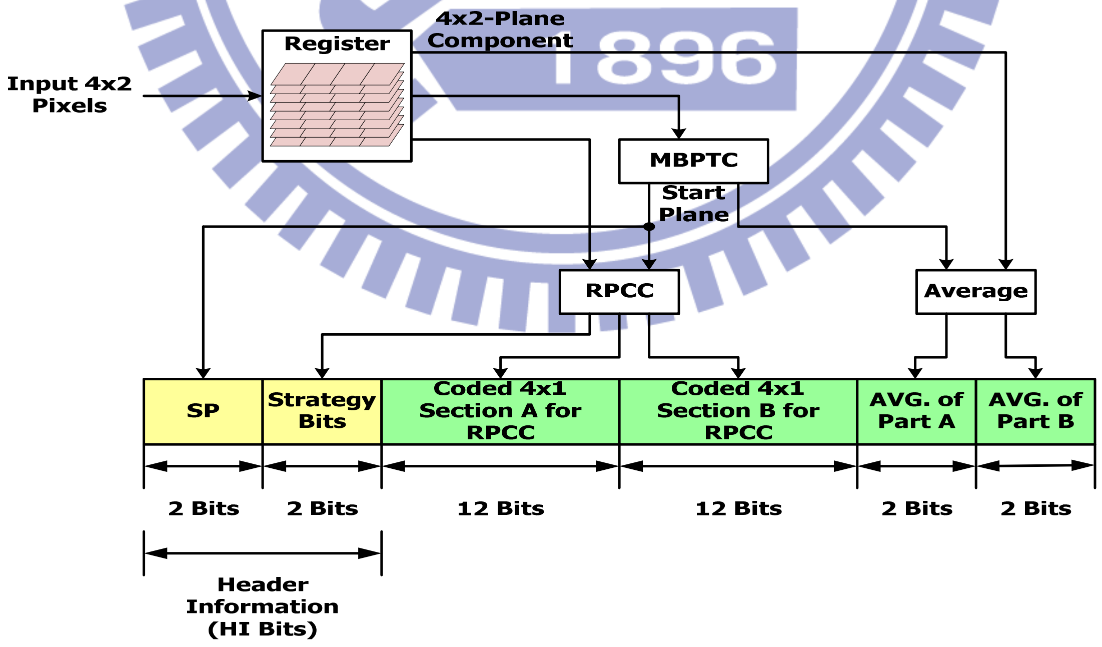


Figure 29: Overall Diagram of Embedded Compressor

4.1.1 Architecture of Modified Bit Plane Truncated Coding Encoder

The hardware design of MBPTC is improved from original BPTC. It is a combinational block to deal with 4x2 pixels to obtain Start Plane (SP) and 4x2-plane component for each 4x2 array. In Figure 30, we employ three 8-input OR gates as thresholds to control the value of SP. The bits of layer 1, 2 and 3 are used to be input of 8-input OR gate individually.

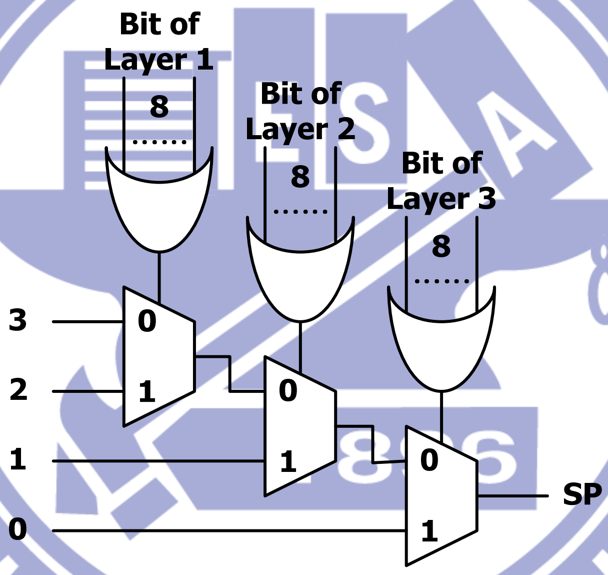


Figure 30: The Hardware Architecture of MBPTC

4.1.2 Architecture of the Reduced Patterns Comparison Coding

RCPP is a combinational block to deal with coded data by MBPTC. As shown in Figure 31, SP selects four layers to be compressed and threshold is exploited to choose the strategy to be adopted. The SP is produced by MBPTC and the threshold is defined by users with different levels as described in Table 3. (Here we adopt Level 2)

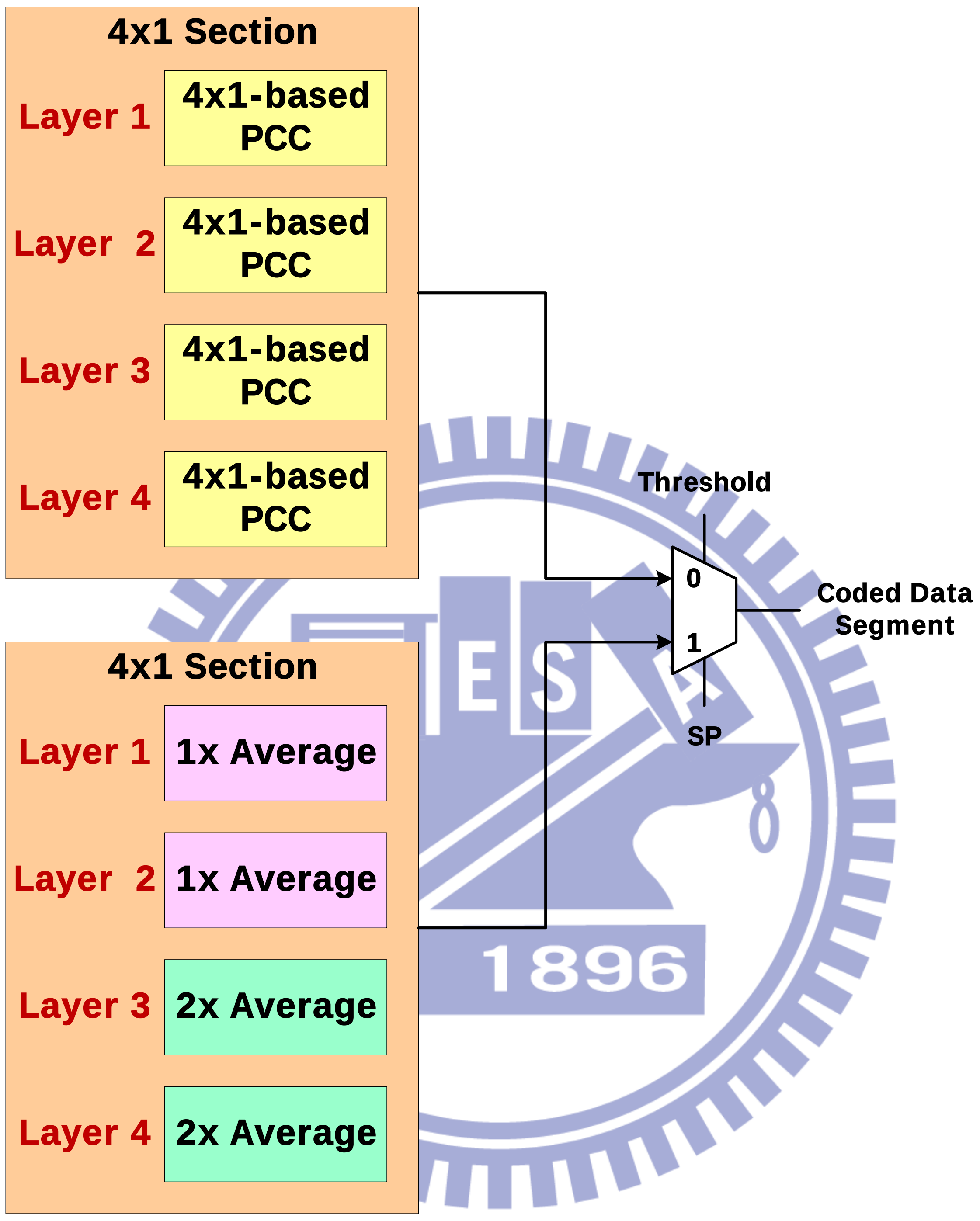


Figure 31: The Hardware Architecture of RPCC

4.1.3 Overall Encoder Design

The actual architecture of compressor design is shown in Figure 32. It takes one cycle to deal with 4x2 block. Here each Macro Block takes 16 cycles to be encoded.

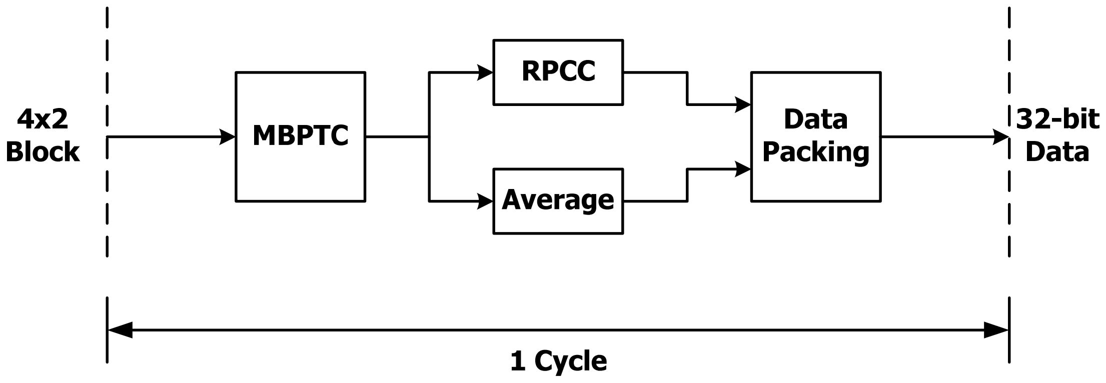


Figure 32: Encoder Design

4.2 *Architecture of Decoder*

Overall diagram of the embedded compressor is shown in Figure 33 and we will introduce these blocks in the following sections.

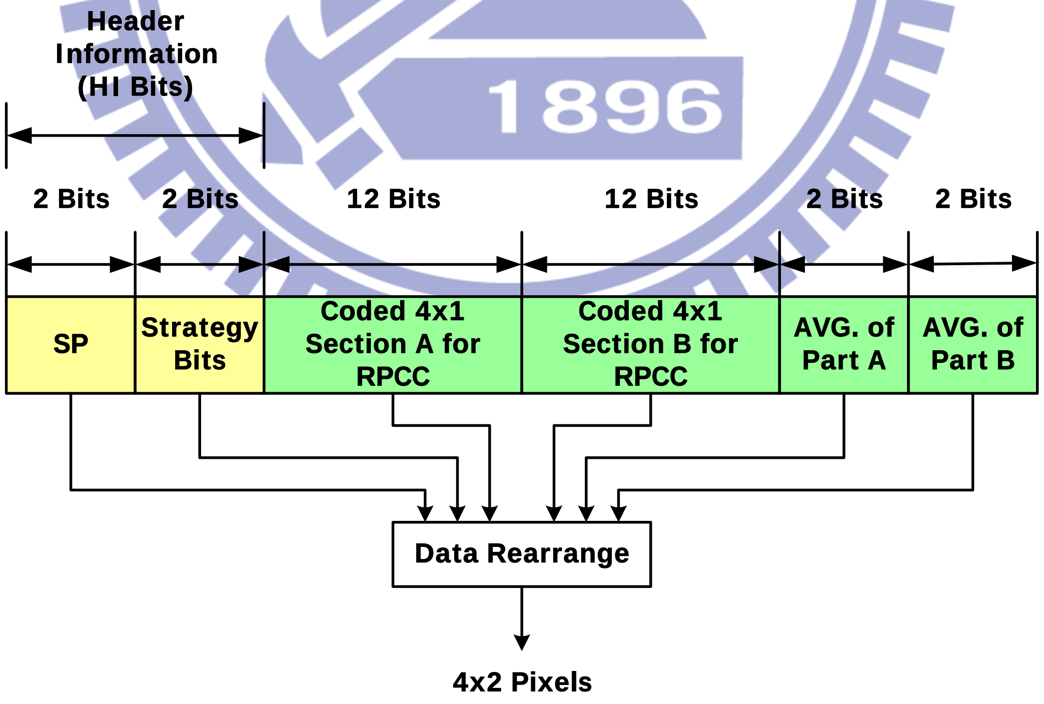


Figure 33: The Overall Diagram of Embedded Decompressor

4.2.1 Data Rearrange

Data unpacking is a simple reverse process of encoding. The less calculation needed on decoder and the information is ready are the main dissimilarities and lead to smaller gate count compared with encoder. Thus, decoder focuses on putting the data on proper positions.

According to coded data format, the SP selects the initial bit plane of decoding. The continuous four layers are then placed on corresponding positions depending on strategy bits. Afterward AVG. of part A and B are placed on the continuous two bit planes after the four layers.

4.2.2 Overall Decoder Design

For providing data to Motion Compensation (MC), the decompressor needs to support higher throughput. The actual architecture of decompressor design is shown in Figure 34. A 4x2 block takes one cycle to be decoded. Under the design, each Macro Block takes 16 cycles to be decoded.

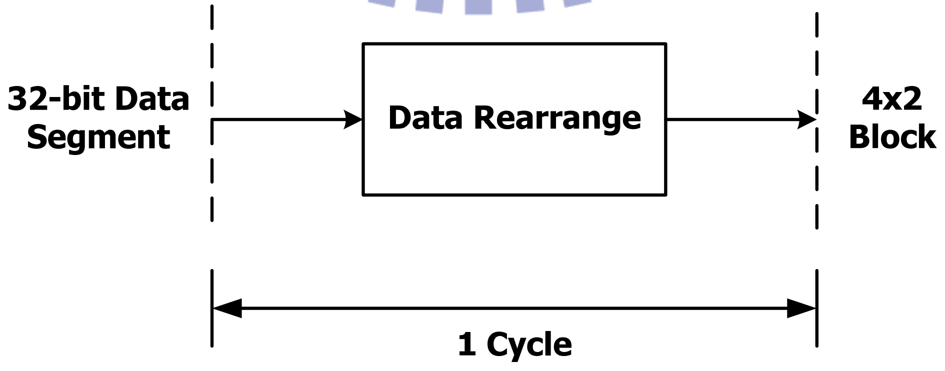


Figure 34: Decoder Design

4.3 *Implementation and Verification*

In this thesis, we propose a flexible algorithm to achieve good coding efficiency. It reduces the usage of bandwidth and the required resource of hardware in the mobile video devices is a critical topic and it suits to be integrated into any mobile video decoder. We will introduce the Specifications of the proposed hardware design and verification in the following sections.

4.3.1 Implementation

The specifications of the proposed architecture are described in Table 5. The proposed architecture is synthesized with UMC 90-nm CMOS standard-cell library and operated at 5, 100 and 150 MHz for different specifications, respectively. For compressor/decompressor, the gate counts are 1.8/1.3 K respectively and the latencies are 1 cycle per MB.

Table 5: Specification of Hardware Design

	Specification	
Synthesis Process	**UMC 90 nm**	
Function Unit	**Compressor**	**Decompressor**
Operate Frequency	**CIF @ 5 MHz** **HD 1080 AVC @ 100 MHz** **HD 1080 / HD 720 SVC @ 150 MHz**	
Latency / MB	**1 Cycle**	**1 Cycle**
Gate Counts	**1.8 K**	**1.3 K**
Power Consumption	**228 uW**	**130 uW**

4.3.2 Verification

Figure 35 shows the flow of verification. We utilize software and hardware to verify the proposed algorithm. The patterns are created by software and applied as the input of hardware designs. Then the software calculates the answer to be compared with result of hardware and the result is stored in memory. Afterward the coded data is accessed by software and hardware decompressor from memory. We check the coded data to confirm the result is matched in software and hardware.

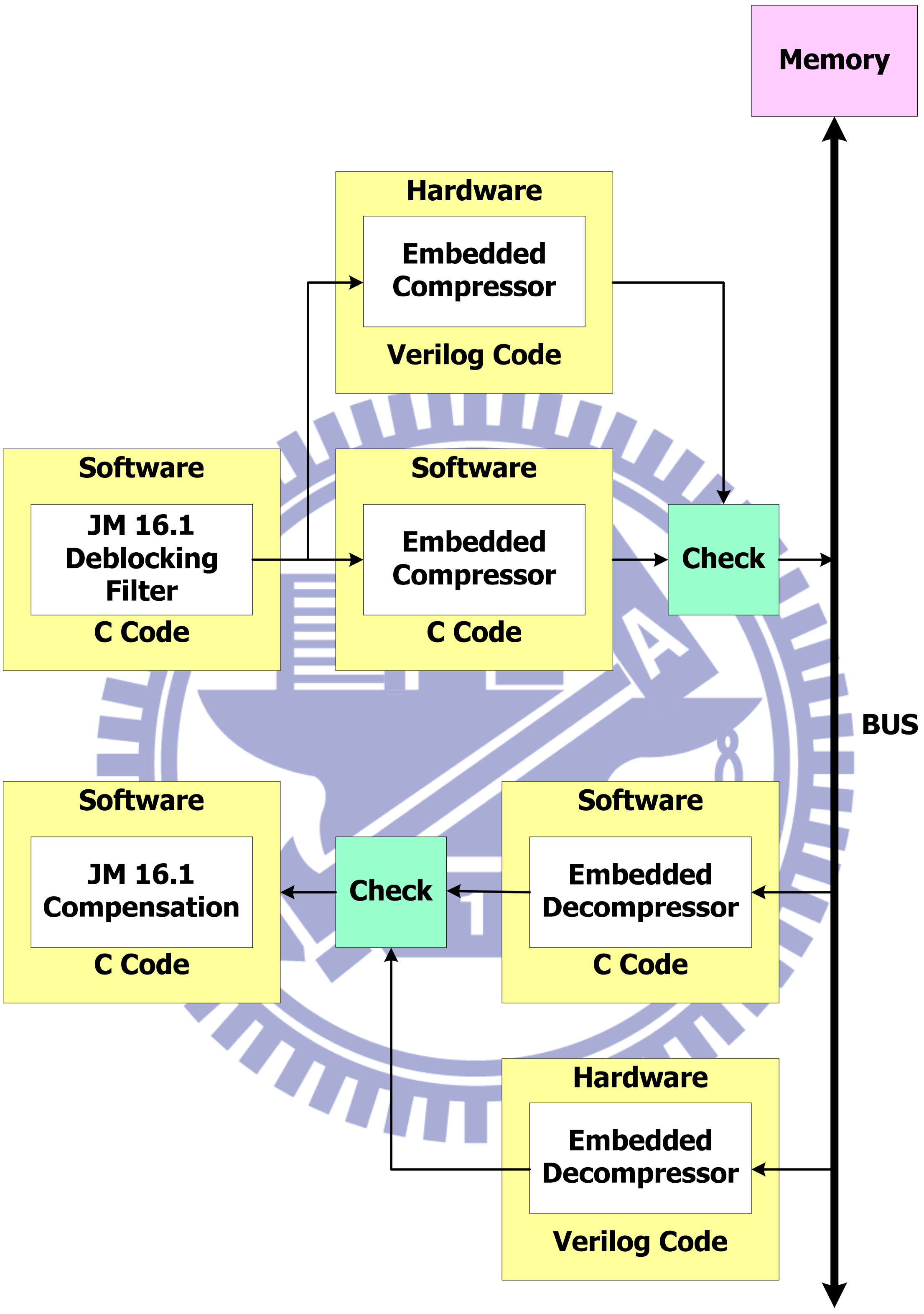


Figure 35: Flow of Verification

The previous MHT work [4] costs 20K gate counts in dealing with a 1x8 pixels array and previous DCT work [5] costs 30K gate counts in dealing with a 4x4 block while the proposed architecture costs 3.1K gate counts in dealing with a 4x4 block. The peak power consumption of proposed embedded codec @ 100 MHz is 358 uW. Thus, the proposed algorithm not only gains 5.95 dB in quality loss but achieves an area-efficient and power-aware hardware implementation.

Chapter 5
System Integration

In this chapter, we show the experimental results including the cycle analysis, bandwidth and power consumption. In addition, we will discuss the problems occurred during integration in detail.

5.1 *Adopted H.264 Hardware System*

Figure 36 shows the overall block diagram of this system. The adopted H.264 decoder works at 5, 100 and 150 MHz respectively to perform CIF, HD 1080 AVC, HD 1080/720 SVC at 30 frames/per second (FPS). The embedded compressor compresses the data from deblocking filter into 64-bit data segment which is stored in external memory. The embedded decompressor decompresses the coded data segment from off-chip memory into 4x2-sized block which is sent to Motion Compensation. The bandwidth of system bus is 32 bits and the external memory is 32 bits per entry.

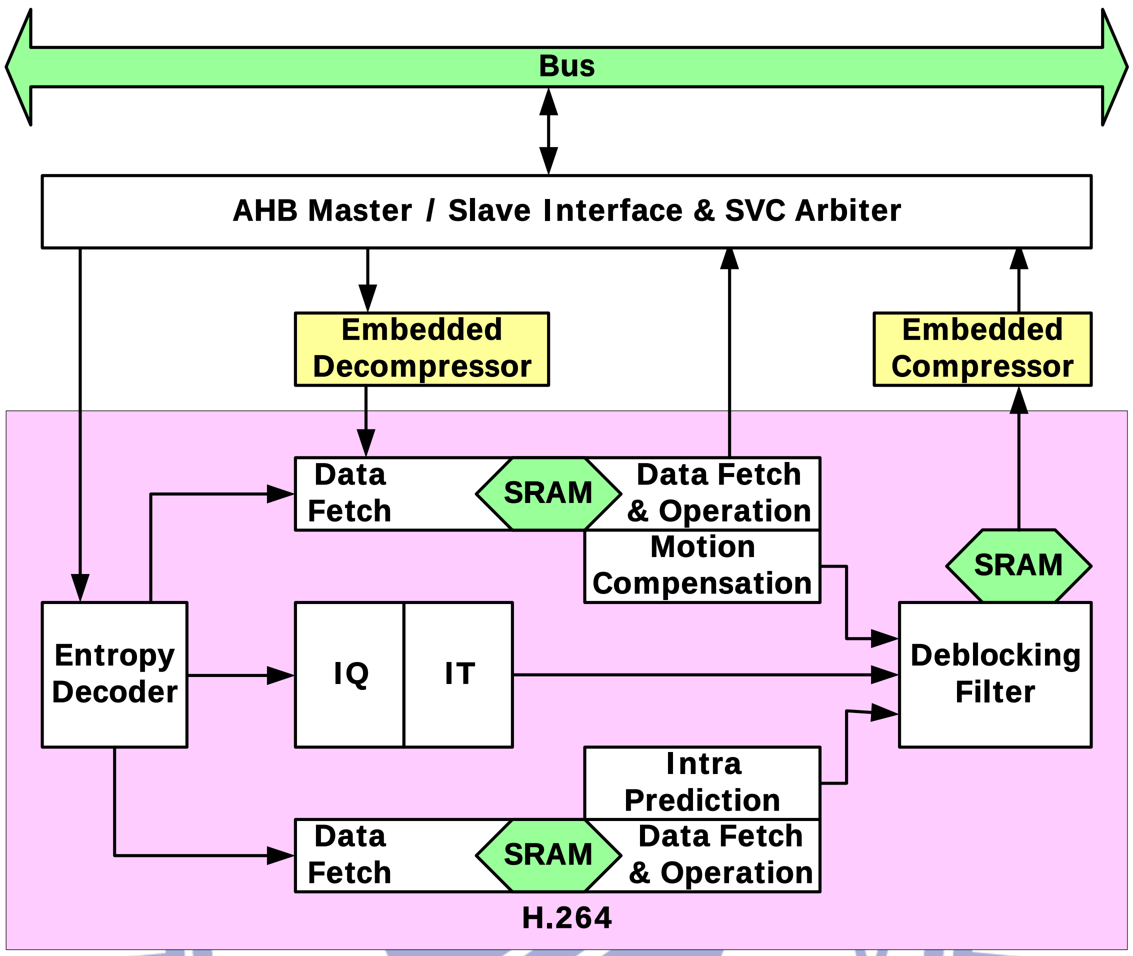


Figure 36: Block Diagram of the Video Decoder System

The embedded codec can be considered as an interface between the chip and external memory. The system interface for embedded codec is shown in Figure 37. Because deblocking Filter sends out four pixels per cycle, the best processing latency for each pipeline stage of embedded compressor is less or equal to four cycles to avoid idle delay at the input of embedded compressor. In addition, the input bandwidth of MC in original system is 4 pixels per cycle, thus the embedded decompressor outputs four pixels per cycle at least. Due to the fixed compression ratio as 2, the address converter is easy to implement.

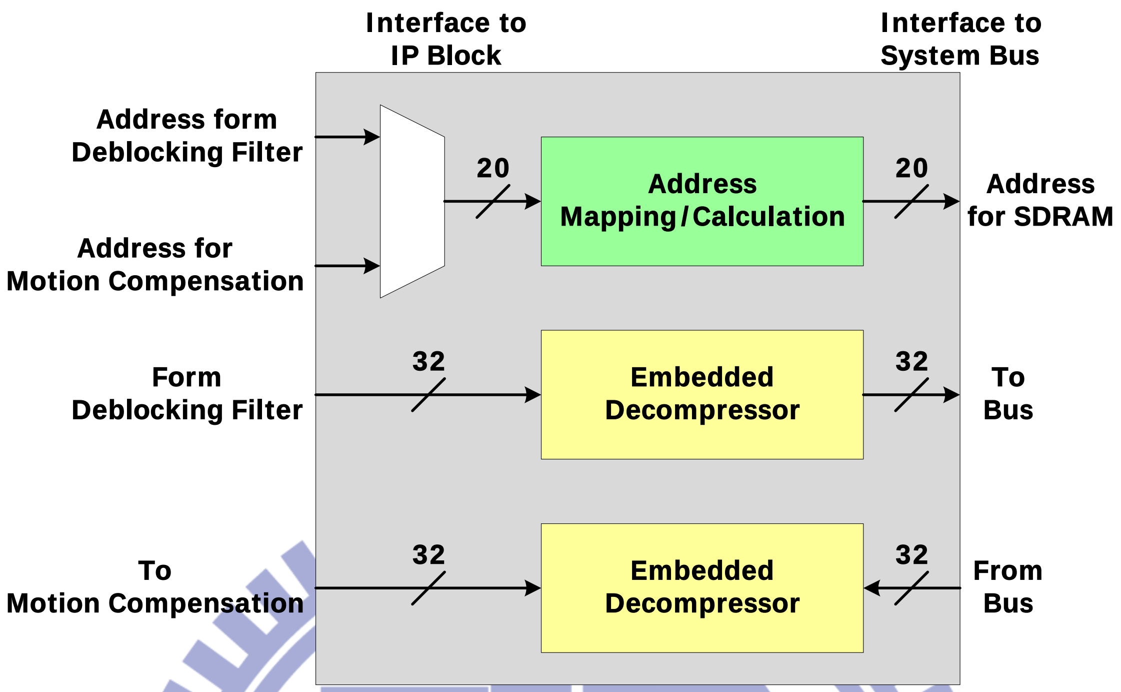


Figure 37: The System Interface for Embedded Codec

5.2 *Access Analysis*

As shown in Figure 38, the related accesses of EC are partitioned into write accesses and read accesses. Write accesses from deblocking filter write the data to external memory and read accesses read the data from external memory to MC.

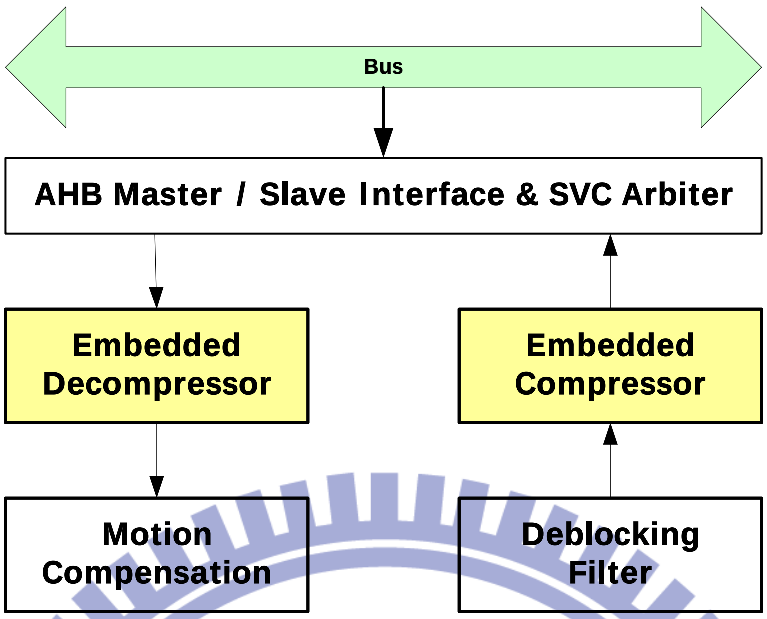


Figure 38: Data Flow of Related Accesses of EC

Many methods have been proposed to improve embedded compression and all of them aim to improve the performance of embedded compression. However most of performance measured by these methods is fragmental, lacking verification from system level. In addition, we expect to precisely estimate the amount of read/write accesses on system view point. Thus, we employ "CoWare" to deal with the complicated problems. As shown in Figure 39, CoWare provides many functions to simulate a complete system and the user-defined means user's design.

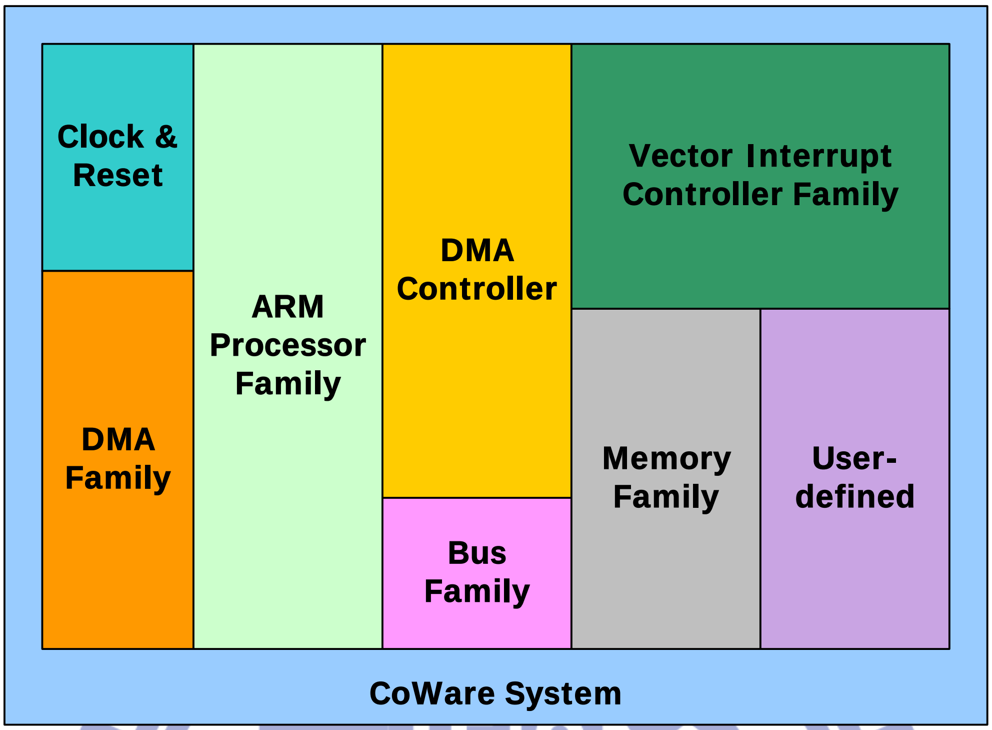


Figure 39: Schematic Diagram for CoWare System

It makes more efficient that we can change the user-defined field relied on our demands. We add the proposed design and H.264 system into user-defined field as shown in Figure 40. The AMBA interface between CoWare and user-defined is coding in System C and it provides a protocol to commutate each other. Furthermore, user-defined means all designs in this filed need to be coded in Verilog.

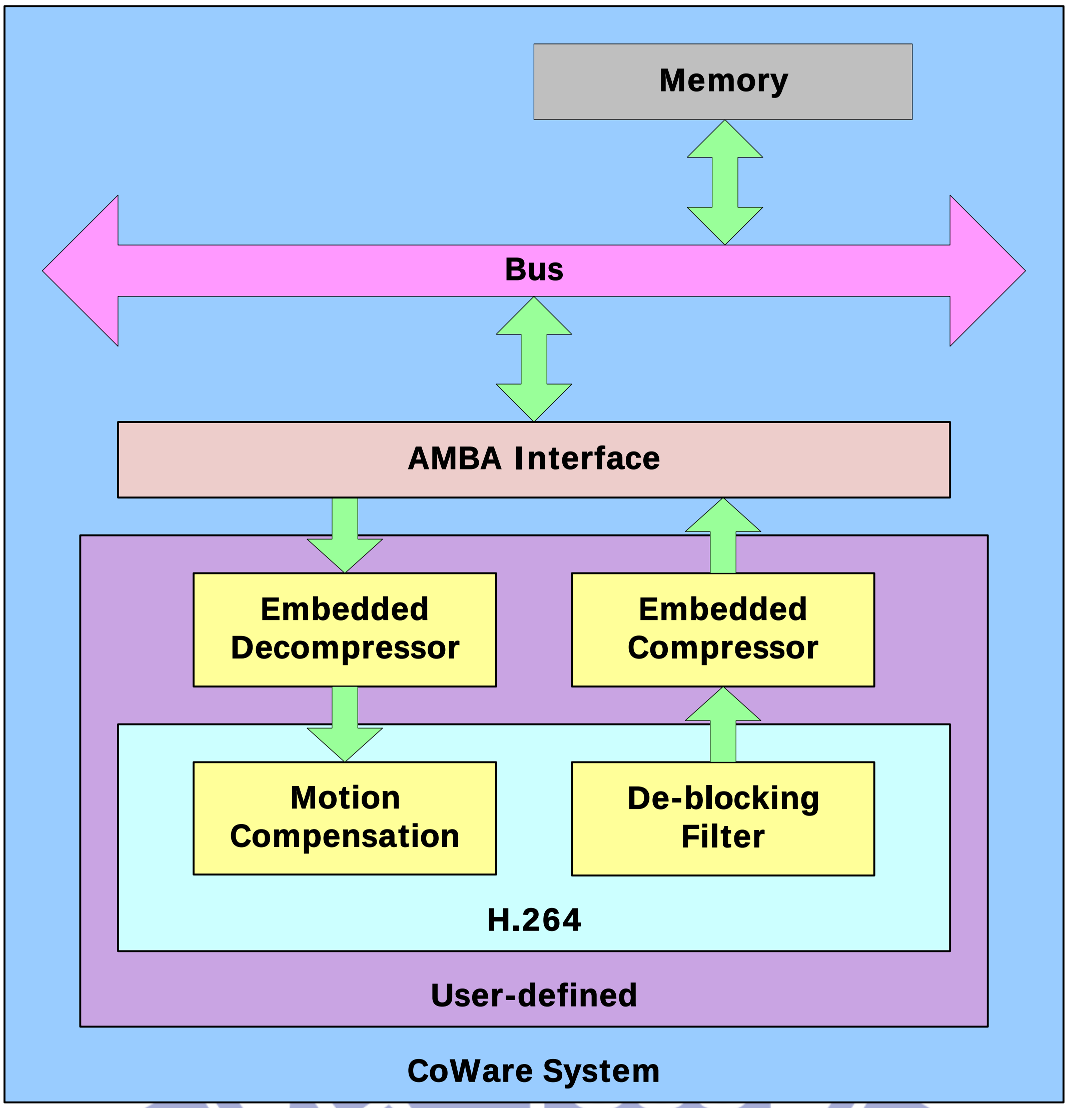


Figure 40: Complete Block Diagram for CoWare System

Figure 41 shows the block diagram in CoWare system in practice. SI2 H.264 video decoder employs single bus/memory. The external memory is adopted 128Mb Mobile LPSDR: MT48H4M32LFB5-6 [17] produced by Micron and the bus protocol exploited AMBA 2.0 with 32bits width.

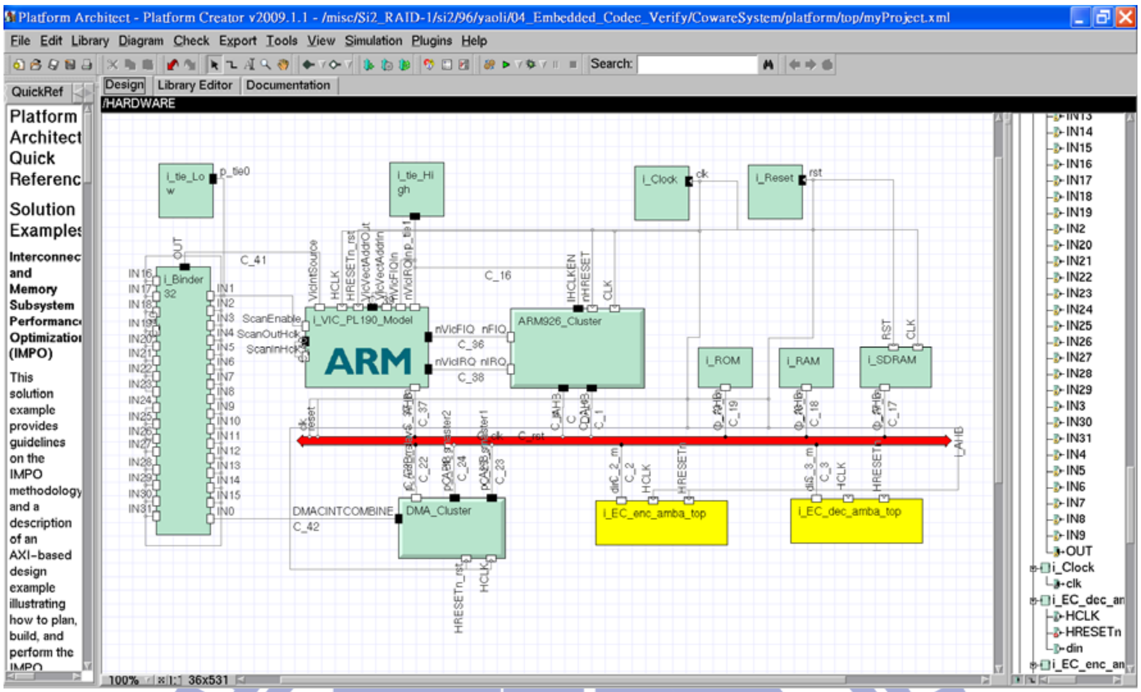

Platform Architect - Platform Creator v2009.1.1 - /misc/Si2_RAID-1/si2/96/yaoli/04_Embedded_Codec_Verify/CowareSystem/platform/top/myProject.xml
HARDWARE
ARM
i_VIC_PL190_Model
ARM926_Cluster
DMA_Cluster
i_ROM
i_RAM
i_SDRAM
i_EC_enc_amba_top
i_EC_dec_amba_top
i_Clock
i_Reset
i_Binder
i_AHB

Figure 41: Block Diagram in CoWare System

We adopt CoWare to verify our design as shown in Figure 42 and Figure 43 shows the data access trace.

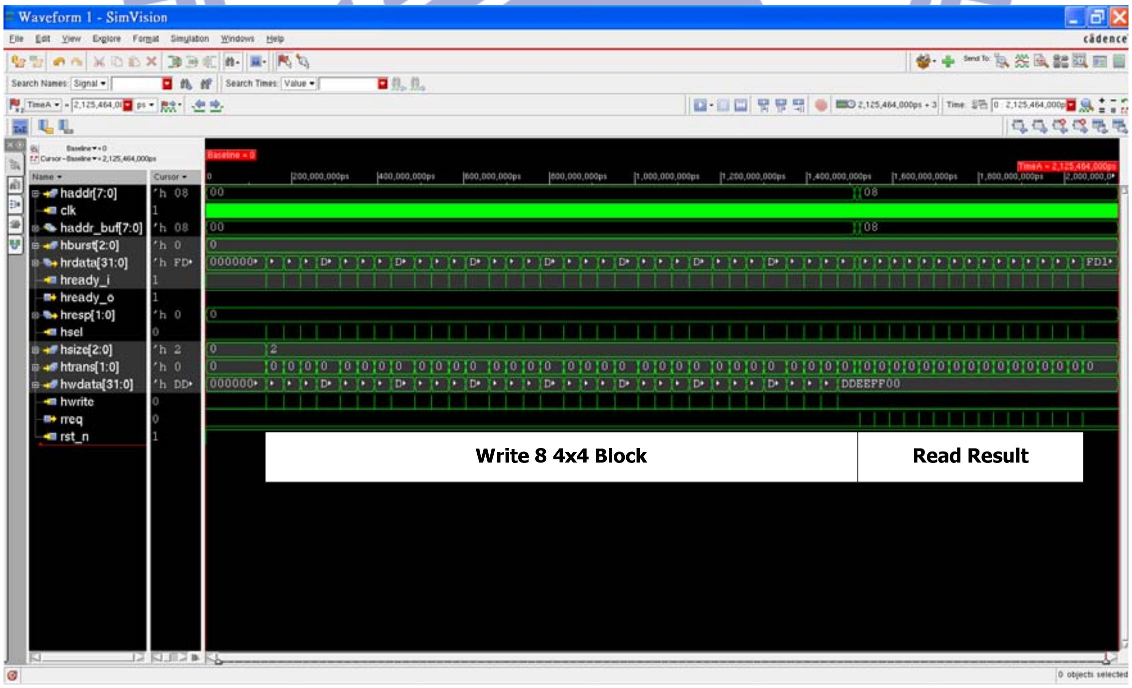


Figure 42: Embedded Compressor Waveform over CoWare System

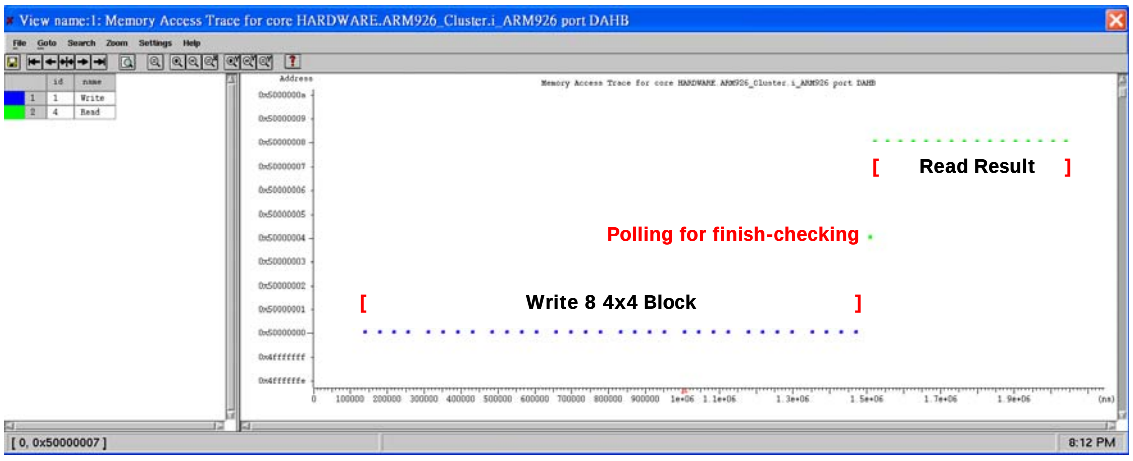

View name:1: Memory Access Trace for core HARDWARE.ARM926_Cluster.i_ARM926 port DAHB
File Goto Search Zoom Settings Help
id name
1 1 Write
2 4 Read
Address
Memory Access Trace for core HARDWARE.ARM926_Cluster.i_ARM926 port DAHB
0x5000000a
0x50000009
0x50000008
0x50000007
0x50000006
0x50000005
0x50000004
0x50000003
0x50000002
0x50000001
0x50000000
0x4fffffff
0x4ffffffe
0 100000 200000 300000 400000 500000 600000 700000 800000 900000 1e+06 1.1e+06 1.3e+06 1.5e+06 1.7e+06 1.9e+06 (ns)
[Read Result]
Polling for finish-checking
[Write 8 4x4 Block]
[0, 0x50000007]
8:12 PM

Figure 43: Data Access Trace

5.2.1 Write Access

Because the proposed algorithm provides fixed CR as two, write accesses are easy to be analyzed. The access times after adding EC are always half of original system. The reduction ratio of write access is 50%.

5.2.2 Read Access

The required read accesses from MC are much more complex. MC requests data which is based on Motion Vector (MV). The value of x and y in MV (x, y) are classified into three categories: align, not align and sub-pixel.

(1) Align: the value is quadruple. It fits with a 4x4 coded block-grid.

(2) Not Align: the value is not quadruple but is an integer. It may span two 4x4 block-grids due to needed four pixels.

(3) Sub-pixel: the value is not integer and accurate at 1/2 or 1/4. It needs nine pixels to be interpolated into four pixels.

As we discussed in the previous chapters, embedded compressor confronts overhead problems and the overhead ratio connects to the coding unit directly. Because the block-grid of our system is 4x1 block-based not pixel-based, the overhead problems can be simplified and analyzed as described in Table 6 and we will simply introduce two cases: the worst case and the best case.

Table 6: All Cases of Read Access Required by MC with/without EC

Case of MV (x , y)	Access Cycles for System Without EC	Access Cycles for System With proposed EC	Reduction of Access Cycles Without EC (%)	Probability of Each case (%)
(Align , Align)	4	2	50	33
(Align , Not Align)	4	2 / 3	50 / 25	0.4
(Align , Sub)	9	5	44.4	5.1
(Not Align , Align)	8	4	50	4.5
(Not Align , Not Align)	8	4 / 6	50 / 25	0.4
(Not Align , Sub)	18	10	44.4	5.4
(Sub , Align)	12	6	50	23.5
(Sub , Not Align)	12	6 / 9	50 / 25	1.81
(Sub , Sub)	27	15	44.4	25.8
AVG.	13.2	6.8 ~ 6.9	49.1 ~ 48.3	

The worst condition is the sub-pixel case as shown in Figure 44. Both x and y are not integers in MV (x, y). The 4x4 block in worst case needs a 9x9 block to complete the motion compensation. While original system needs 27 cycles to deal with this case, embedded compressor takes 15 cycles to do that.

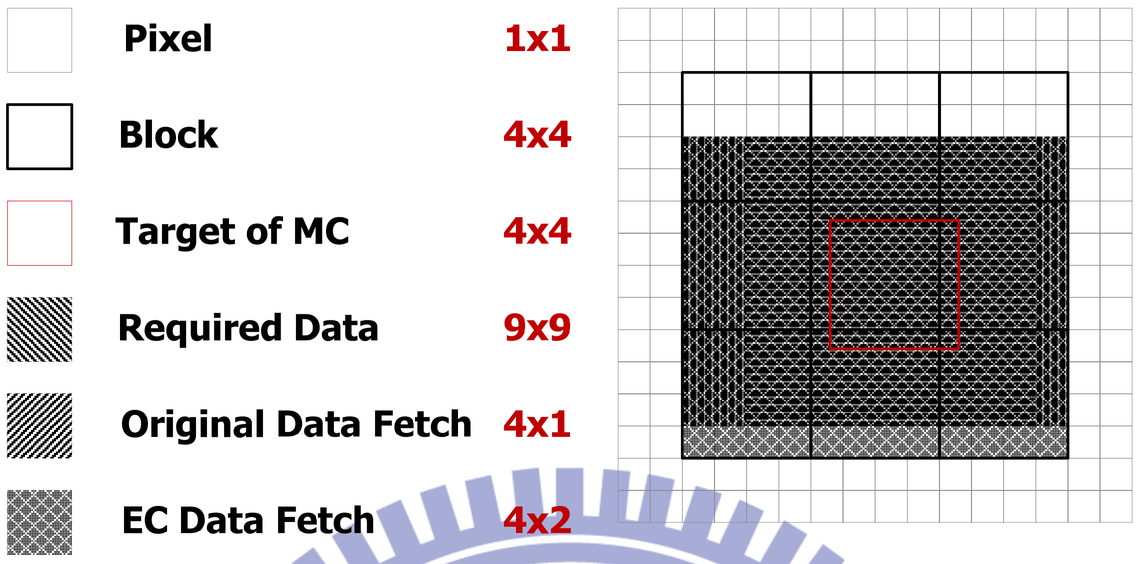


Figure 44: Worst Case on Fetching

We show the best case in Figure 45. If the required 4x4 blocks of MC are aligned with the coded 4x4 blocks, original system with/without embedded compressor needs 2/4 cycles to deal with the case.

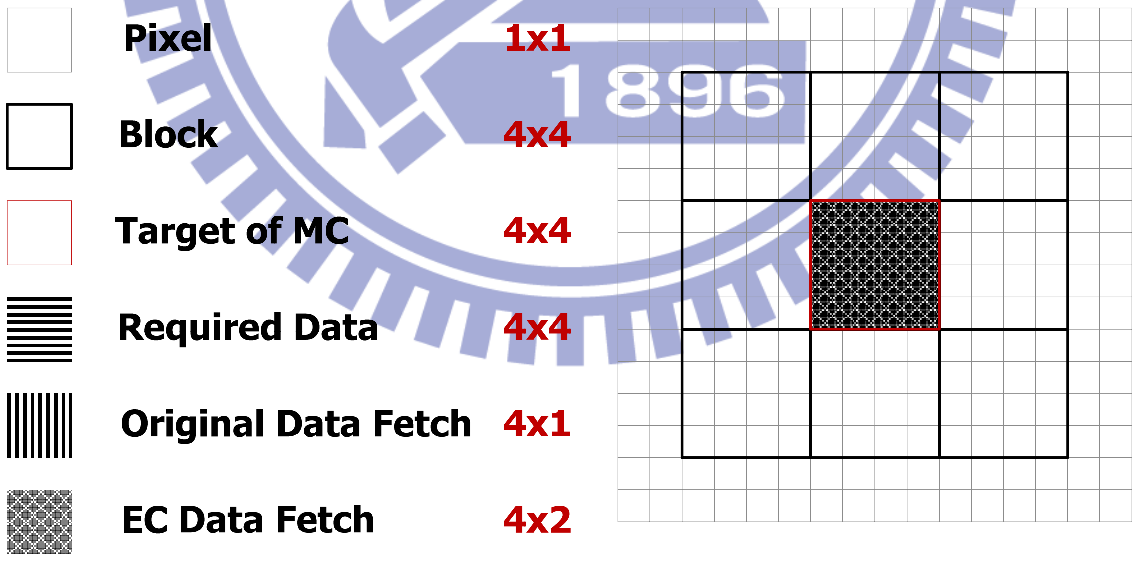


Figure 45: Best Case on Fetching

There are three special cases including (Align, Not Align), (Not Align, Not Align) and (Sub, Not Align). In Figure 46, we explain one of these special cases as an example. If required data of MC is not fit for 4x2 block-grids as the proposed algorithm adopts, it may increase an extra access.

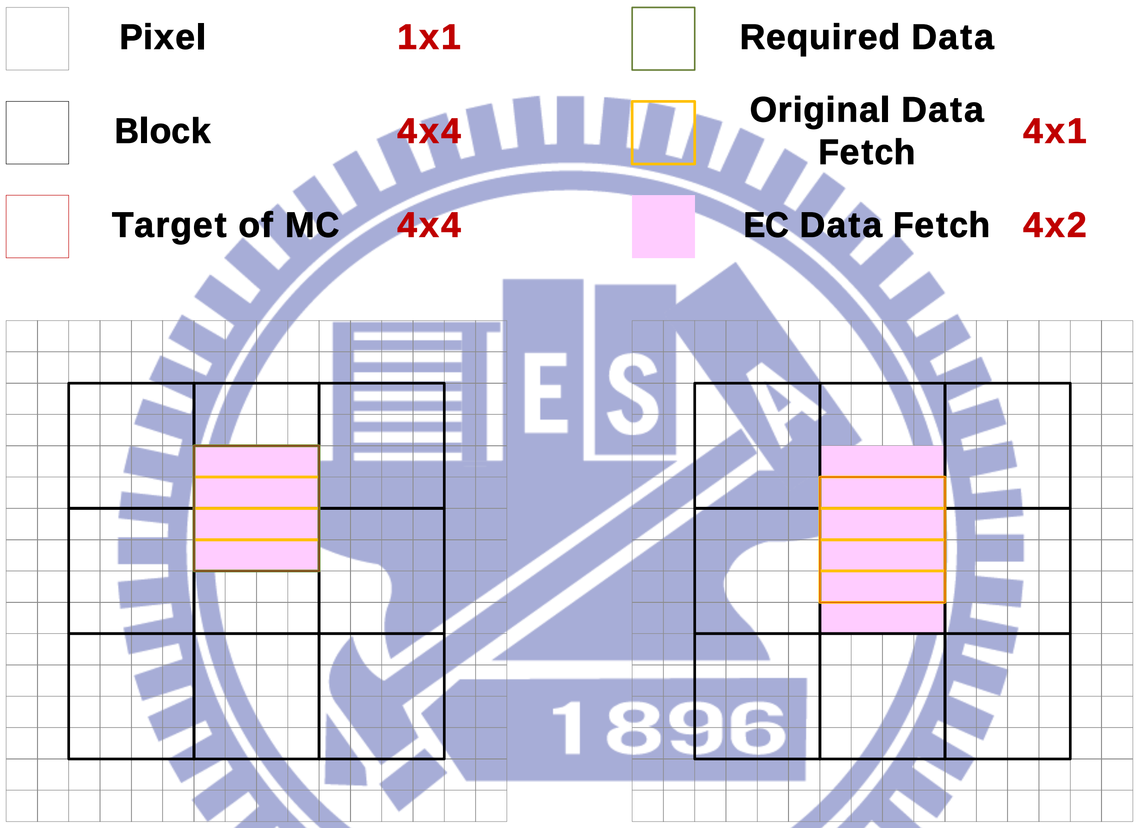


Figure 46: An Example of Special Cases (Align, Not Align)

The probabilities of each case are obtained from simulation on four sequences (Akiyo, Foreman, Stefan and Mobile Calendar) which are formed by GOP 30, three hundred frames each. According to the probabilities, the average reduction rate on read accesses is 50% of original accesses.

5.3 Processing Cycle Analysis

There is a main restriction in this original system that we do not respect to modify the data input mechanism of MC. Under this constraint, we need to insert a register between MC and embedded decompressor. Because this solution will increase latency, we must recalculate the processing cycles as described in (3).

$$\text{Processing Time for MC with EC} = \text{Delay for EC Decoder} + \text{Processing Time for MC without EC} \quad (3)$$

We can derive the new processing cycles for all cases from (3) as described in Table 7. MC with EC are much less than 25 cycles excluding the (sub, sub) case. The average processing cycles for MC with EC are 17.4 cycles. Although the (sub, sub) case more than 25 cycles, there are available cycles from other modes. Thus, the proposed EC can be embedded into original system in practice.

Table 7: All cases of Processing Cycle Analysis for Embedded Compressor

Case of MV (x , y)	Number of Blocks	Delay for our EC Decoder	Reduction Ratio of Delay for our EC (%)	Processing Cycles for MC without EC	Processing Cycles for MC with our EC	Probability of Each case (%)
(Align , Align)	1	2	50	4	6	33
(Align , Not Align)	2	2	60	4	6	0.4
(Align , Sub)	3	3	40	9	12	5.1
(Not Align , Align)	2	3	40	8	11	4.5
(Not Align , Not Align)	4	4	42.9	8	12	0.4
(Not Align , Sub)	6	6	14.3	18	24	5.4
(Sub , Align)	3	3	50	12	15	23.5
(Sub , Not Align)	6	6	33.3	12	18	1.81
(Sub , Sub)	9	8	11.1	27	35	25.8
AVG.	4.1	4.2	31.1	13.2	17.4	

5.3.1 Ratio of Access Reduction

The access ratio is defined as (4). According to the simulation result, the ratio of read accesses with/without EC is 51.7%, and the ratio of write accesses with/without EC is fixed as 50%. In addition, the average ratio of read/ write accesses is about 3.51. Thus, the overall ratio of access (with/ without EC) is recalculated as (5).

$$\text{Access Ratio} = \frac{\text{Mem\_Read\_EC} + \text{Mem\_Write\_EC}}{\text{Mem\_Read\_Ori.} + \text{Mem\_Write\_Ori.}} \tag{4}$$

$$\begin{aligned}\text{Overall Access Ratio} &= \frac{0.517 \times (\text{Mem\_Read\_Ori.}) + 0.5 \times (\text{Mem\_Write\_Ori.})}{\text{Mem\_Read\_Ori.} + \text{Mem\_Write\_Ori.}} \\ &= \frac{0.517 \times (3.51) + 0.5 \times (1)}{3.51 + 1} = 51.3\ \%\end{aligned} \tag{5}$$

Thus, the reduction ratio on memory accesses is shown in (6).

$$\begin{aligned}\text{Average Reduction Ratio} &= 1 - \text{Overall Access Ratio} \\ &= 1 - 0.513 = 48.7\ \%\end{aligned} \tag{6}$$

The average reduction ratio is about 48.7%.

5.3.2 Simulation Result on Power Consumption

We adopt the system-power calculator as [16] as power model of external memory and set the parameter according to [17]. The utilization of memory is simulated on CIF @ 4.8 MHz, HD 1080 / AVC @ 100 MHz and HD 1080 + HD 720 / SVC @ 150 MHz. We show the results in Figure 47, Figure 48 and Figure 49 respectively. There are core power of H.264 decoder, SDRAM background power and SDRAM access power (read/write) which are operated at different frequencies. Although the EC operated at 5 MHz, 100 MHz and 150 MHz consumes 0.0116 uW, 0.238uW and 0.358 uW respectively, it reduces power of each for 1 mW @ CIF (37 %), 16.15 mW @ HD 1080 / AVC (28.4 %) and 32.8 mW @ HD 1080 + HD 720 / SVC

(38.4 %). It is very obvious that the average available cycles for a 4x4 block on these video formats are the same. In addition, the access ratio on read/write is slightly different due to different test sequences. Thus, the amount of reduced power is almost proportional to the frame size.

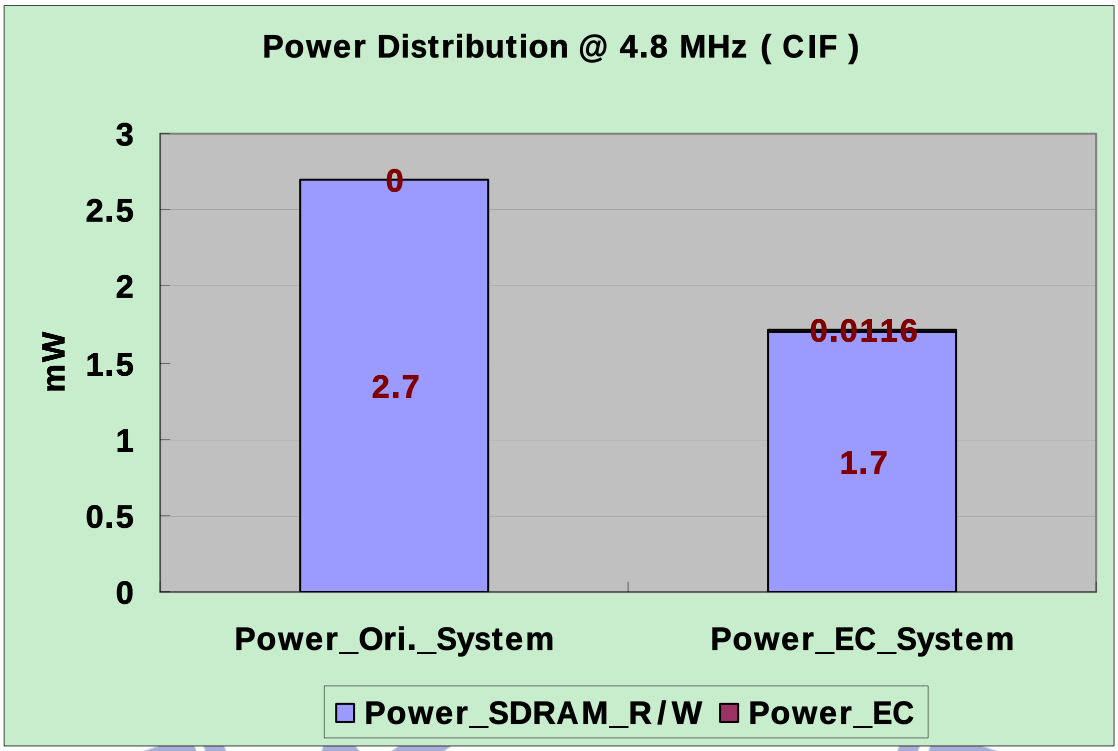


Figure 47: Power Analysis on CIF @ 4.8 MHz

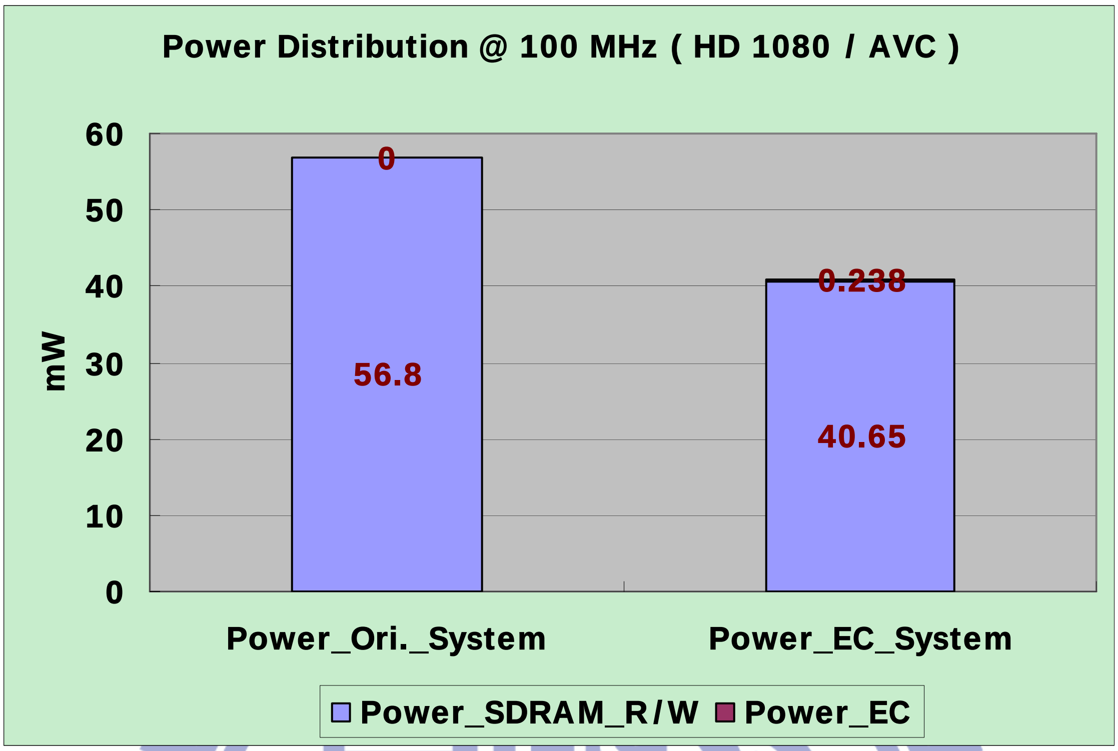


Figure 48: Power Analysis on HD 1080/AVC @ 100 MHz

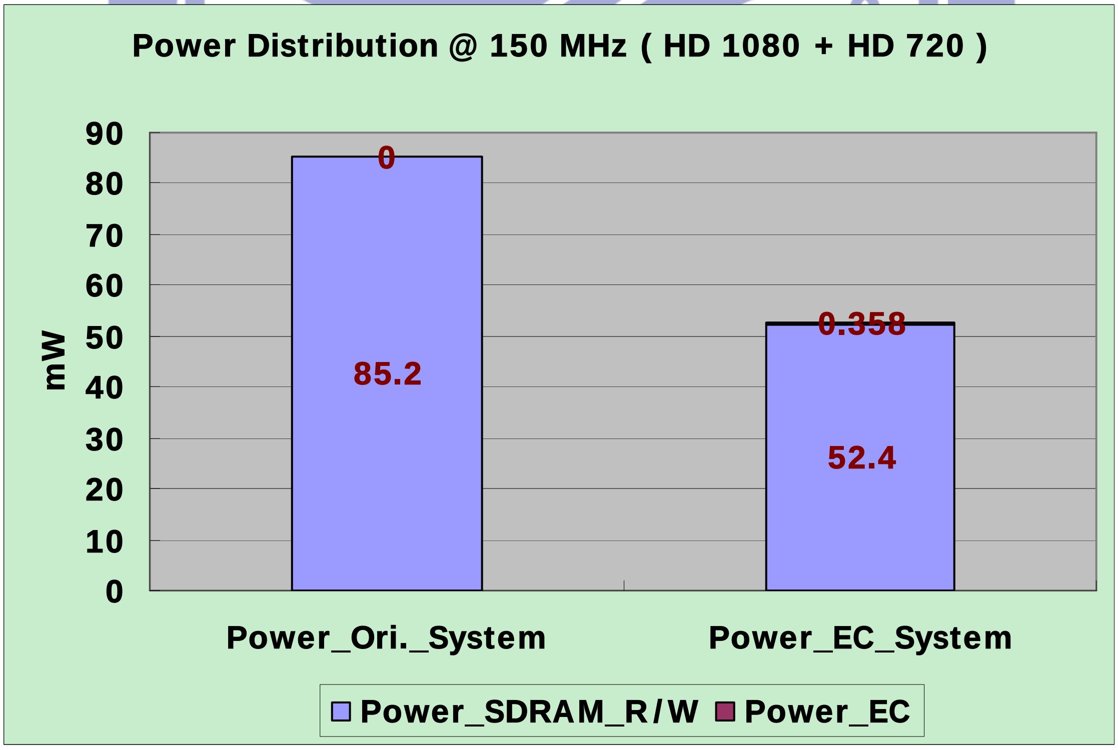


Figure 49: Power Analysis on HD 1080 + HD 720/SVC @ 150 MHz

Chapter 6
Conclusion and Future Works

6.1 Conclusion

In this thesis, we have proposed a flexible algorithm whose compression ratio is fixed as 2. This coding efficiency suits for any mobile video device. With these advantages of the proposed EC engine, we can lessen the size of external memory and bandwidth utilization to achieve the goal of power saving. Due to the fixed Compression Ratio, the proposed function is easy to be integrated with an H.264 system. The proposed architecture is synthesized with 90-nm CMOS standard-cell library and the gate counts of the proposed algorithm for embedded compressor/decompressor are 1.8K/3.1K respectively. The average PSNR loss of proposed algorithm is 5.98 dB. The working frequencies are 5 (CIF), 100 (HD 720) and 150 (HD 1080 + HD720) MHz depending on different operation modes. The proposed algorithm compresses a MB takes 16 cycles while to decompress a MB takes only 16 cycles. It saves 48.7% of memory accesses on the average, leading to save considerable power consumption.

6.2 *Future Work*

The main objective is improving coding efficiency because reducing quality loss is the only way to improve error propagation. Thus, we proposed two improving directions of coding efficiency.

First, we combine the 4x1-based Patterns Comparison Coding (PCC) and average coding into the proposed Reduced Patterns Comparison Coding (RPCC) method. With the adopted threshold, although average coding exactly solves the worse cases of PCC methods in visual quality, it will increase PSNR loss. Therefore, we can develop a more efficient and adaptive scheme by combining PCC with other embedded methods which can be modified to fix into 4x2 block-grid or improving the adopted threshold.

Second part is patterns adaptive. The proposed algorithm includes 4x1-based PCC and average coding scheme, which is two-bit-planes based coding method. According to simulation result, the certain of patterns will appear at the certain of bit planes. Thus, we can improve the method by comparing different layers with different patterns for PCC.

Bibliography

[1] ITU-T Recommendation H.264 and ISO/IEC 14496-10, Advanced Video Coding for Generic Audiovisual Services, May 2003.

[2] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, “Overview of the H.264/AVC video coding standard,” IEEE Trans. Circuits Syst. Video Technol., vol. 13, no. 7, pp. 560–576, July 2003.

[3] R. Manniesing, R. Kleihorst1, R. V. Vleuten1, and E. Hendriks, “Implementation of lossless coding for embedded compression,” IEEE ProRISC, 1998.

[4] T. Y. Lee, “A New Frame-Recompression Algorithm and its Hardware Design for MPEG-2 Video Decoders,” IEEE Trans. CSVT, vol. 13, no. 6, pp. 529-534, June 2003.

[5] Y. -D. Wu, Y. Li, C. -Y. Lee, “A Novel Embedded Bandwidth-Aware Frame Compressor for Mobile Video Applications”, ISPACS’09, pp. 1-4, Feb. 2009.

[6] Yongie Lee, et al, "A New Frame Recompression Algorithm Integrated with H.264 Video Compression," IEEE Circuits Sys. ISCAS Vol. 6, pp. 6110-6113, May 2007.

[7] E. J. Delp and O. R. Mitchell, “Image Compression Using Block Truncation Coding”, IEEE Trans. Commun., Volume 27, Issue 9, pp. 1335 - 1342, Sep. 1979.

[8] C. -K. Yang and W. -H. Tsai, “Improving block truncation coding by line and edge information and adaptive bit plane selection for gray-scale image compression”, Volume 16, Issue 1 pp. 67 – 75 Jan. 1995.

[9] Amarunnishad T. M., Govindan V. K., and Abraham T. Mathew, “Block Truncation Coding Using a Set of Predefined Bit Planes”, ICCIMA, Volume 3, pp. 73 – 78, Dec. 2007.

[10] R. Dugad and N. Ahuja, “A Fast Scheme for Image Size Change in the Compressed Domain,” IEEE Trans. CSVT, vol. 11, no. 4, pp. 461-474, April 2001.

[11] D. Pau et al., “MPEG-2 Decoding with a Reduced RAM Requisite by ADPCM Recompression before Storing MPEG Decompressed Data,” U.S. patent 5838597, Nov. 1998.

[12] R. Bruni et al., “A novel adaptive vector quantization method for memory reduction in MPEG-2 HDTV decoders,” in Proc. Int. Conf. Consumer Electronics, pp. 58-59, 1998.

[13] R. Dugad and N. Ahuja, “A Fast Scheme for Image Size Change in the Compressed Domain,” IEEE Trans. CSVT, vol. 11, no. 4, pp. 461-474, April 2001.

[14] C. –C. Cheng, P. -C. Tseng, and L. –G. Chen, “Multimode Embedded Compression Codec Engine for Power-Aware Video Coding System”, TCSVT, Volume 19, pp. 141 – 150, Feb. 2009.

[15] A. Bourge and J. Jung, “Low-Power H.264 Video Decoder with Graceful Degradation,” SPIE Proc. Visual Communications and Image Processing, vol. 5308, pp. 372-383, Jan. 2004.

[16] Micron® Technology Inc. The Micron® System-Power Calculator: SDRAM. [Online Available]: http://www.micron.com/support/part_info/powercalc

[17] Micron® Technology Inc. MT48H4M32LFB5-6 128Mb Mobile LPSDR. [Online Available]: http://www.micron.com/products/partdetail?part=MT48H4M32LFB5-6

作　者　簡　歷

姓名　　：楊均宸

戶籍地　：台灣省台北市

出生日期：1980 年 12 月 15 日

學歷：

1995 年 9 月~1998 年 6 月　台北市立內湖高工　電子科

2000 年 9 月~2004 年 6 月　私立龍華科技大學　電子工程系

2004 年 9 月~ 2009 年 12 月　國立交通大學 IC 設計產業研發碩士專班