國 立 交 通 大 學

電子工程學系 電子研究所碩士班

碩 士 論 文

適用於H.264/MPEG4-AVC及其可調式視訊編碼之移動估測設計

Motion Estimation Design for H.264/MPEG4-AVC Video Coding and Its Scalable Extension

研 究 生：曹克嘉

指導教授：張添烜教授

中 華 民 國 一百 年 十一 月

適用於 H.264/MPEG4-AVC 及其可調式視訊編碼之移動估測設計

Motion Estimation Design for H.264/MPEG4-AVC Video Coding and Its Scalable Extension

研 究 生: 曹克嘉　　Student: Ko-Chia Tsao
指導教授: 張添烜　博士　　Advisor: Tian-Sheuan Chang

國　立　交　通　大　學
電子工程學系　電子研究所碩士班
碩　士　論　文

A Thesis
Submitted to Department of Electronics Engineering & Institute of Electronics
College of Electrical and Computer Engineering
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of Master
In
Electronics Engineering
January 2010
Hsinchu, Taiwan, Republic of China

中華民國　一百年 十一月

適用於 H.264/MPEG4-AVC 及其可調式視訊編碼之移動估測設計

研究生：曹克嘉　　　指導教授： 張添烜　博士

國　立　交　通　大　學
電機學院　電子工程學系　電子研究所

摘　　要

移動估測在視訊編碼的過程中，具有非常高的複雜度，因此成為即時影像編碼的瓶頸，在高壓縮律規格(H.264/AVC)的可調式視訊編碼中，由於其額外引用的層間預測編碼，使得原本整數點移動估計之高頻寬存取所帶來的問題更加嚴重。因此，本論文引用一能有效改善層間預測高頻寬存取的演算法並提出相對應的硬體架構，此硬體架構使得整數點移動估測和層間預測能併行運算並共用運算時所需的資料。此外，為了改善分數點移動估測的高複雜度和高計算量，本論文引用了一分數點快速演算法並提出相對應的硬體架構，此提出之架構與先前架構相比運算速度可增加三倍。由於多種移動向量和來自於層間預測的多種編碼方式，使得分數點移動估測的運算量和運算時間大為增加，為了更進一步減少分數點移動估測的運算時間和運算量，本論文引用了一能有效篩選欲執行分數點移動估測的編碼方式之演算法，並且將其延伸與多層解析度移動估測演算法之結果一併考慮，進而提出一種能從不同層解析度編碼方式之中有效篩選欲執行分數點移動估測的編碼方式之演算法，經過此演算法，相較於原先的最多 20 種編碼方式，此演算法篩選至僅僅 3 種編碼方式須要被執行分數點移動估測，此演算法相較於先前無篩選的做法訊雜比下降了 0.106dB 而位元率增加了 3.542%。

Motion Estimation Design for H.264/MPEG4-AVC Video Coding and Its Scalable Extension

Student：Ko-Chia Tsao　　Advisors：Dr. Tian-Sheuan Chang

Department of Electronics Engineering
Institute of Electronics
National Chiao Tung University

Abstract

Motion estimation is (ME) is the most complex part and the bottle neck of a real time video encoder. The adoption of inter-layer prediction (IL prediction) in H.264/AVC SVC extension even increases the computing time and memory bandwidth of ME. Thus, we adopted the previous data efficient inter-layer prediction algorithm [4] to save the memory bandwidth. In this thesis, we propose the corresponding hardware architecture for inter-layer prediction which can process INTER mode and different inter-layer prediction modes in parallel to save the computing time and memory bandwidth. Furthermore, in order to reduce the high complexity and computation of FME, we adopt the Single-Pass Fractional Motion Estimation (SPFME) as our fast FME algorithm in our FME process. We then propose the corresponding FME hardware architecture for SPFME according to the previous architecture of FME design [3]. Compared with the previous architecture, our proposed architecture can speed up to four times faster. There are many prediction modes due to the adoption of inter-layer prediction and different block types. Thus, to further reduce the complexity and computing time of FME, we adopt the pre-selection algorithm of Li's to eliminate some prediction modes from FME process. However, the Parallel Multi-Resolution Motion Estimation (PMRME) algorithm [1] is adopted in our IME process. Hence, we further propose a multi-level mode filtering scheme to select 3 prediction modes from 3 different search levels. Finally, we integrate the adopted IL prediction, mode filtering, and the SPFME algorithm. The simulation results shows that the proposed function flow with mode filtering can achieve average 3.542% of bit-rate increment and 0.106dB of PSNR degradation in CIF sequence for 2 spatial layers. The implementation results of the whole ME architecture is also shown. It can support CIF+480p+1080p video @60 fps under 135MHz.

致 謝

首先，我要感謝我的指導教授-張添烜博士，老師不厭其煩的指導使我在研究的道路上獲益良多，每每遇到挫折時老師也會給予我適時的協助和關心，而平時討論和解決研究的問題時，老師也給予我極大的空間去發揮和思考，讓我在學習的過程中發現更多無限的可能，因此，我對張教授的感激之情溢於言表。此外，也謝謝百忙之中抽空來指導我的口試委員們-交大電子李鎮宜教授和清大電機陳永昌教授，你們寶貴的意見更加彌補了我論文的不足。

感謝 VSP 實驗室的各位學長姐和學弟妹，尤其是帶領我進入視訊編碼領域的李國龍學長，謝謝你總是帶給實驗室許多歡樂氣氛，每當有問題請教時你也總是很有耐心的引導我找到答案，也謝謝你總是教我許多有用的研究技巧和方法，讓我得以順利完成此篇論文。同時也謝謝曾宇晟、王國振學長、陳易群學長在我研究的生活中帶給我許多寶貴的意見和經驗，和你們聊天總是使我覺得很充實又開心。謝謝學弟妹輔仁、兆傑、珊榕、碩文，你們總是在我需要幫助時伸出援手，讓我在最後寫論文的這段期間得以順利進行；也謝謝效瑜、沛澤、炫谷、奕晴，你們認真的研究態度也使我深夜在實驗室寫論文得以不孤單。謝謝亮齊、孟勳、英佑兩年來的陪伴，你們不管在我修課或研究都帶給我許多幫助，很懷念和你們忙裡偷閒去唱歌的時光，這些都讓我的碩班生涯增色不少。

謝謝我的女友，謝謝妳一直以來的支持，在妳自己也忙於研究所考試之餘還要不停鼓勵我，因為妳的鼓勵和陪伴讓我得以順利完成我的論文。也謝謝美國影集越獄風雲，因為有你的陪伴讓我在最後寫論文的其間不枯燥乏味，使我在回到家時得以放鬆心情。當然也要謝謝默默支持我的家人。

最後，把本論文獻給所有愛我及所有我愛的人。

Contents

List of Figures

List of Table

Chapter 1. Introduction

1.1. Overview of SVC

In these years, video coding has been developed rapidly in order to satisfy a variety of applications range from mobile device display to high-definition TV. As a result, many video coding standards have been standardized to increase compatibility among different video applications. One of the state-of-the-art video coding standards called H.264/AVC, which was standardized by Joint Video Team (JVT), can achieve amazing compression ratio compared with traditional video coding standards thanks to the adoption of many different optimization techniques.

However, to further satisfy the requirement of end user heterogeneity, an advanced video coding standard called Scalable Video Coding (SVC) [1], as an extension of H.264/AVC, has been standardized.

SVC supports three scalabilities, which are temporal, spatial, and quality scalability. Temporal scalability supports different frame rate by using hierarchical B structure. Quality scalability is achieved by Fine-Grain Scalability (FGS), Coarse-Grain Scalability (CGS) or Medium-Grain Scalability. Spatial scalability is supported by varying frame resolutions.

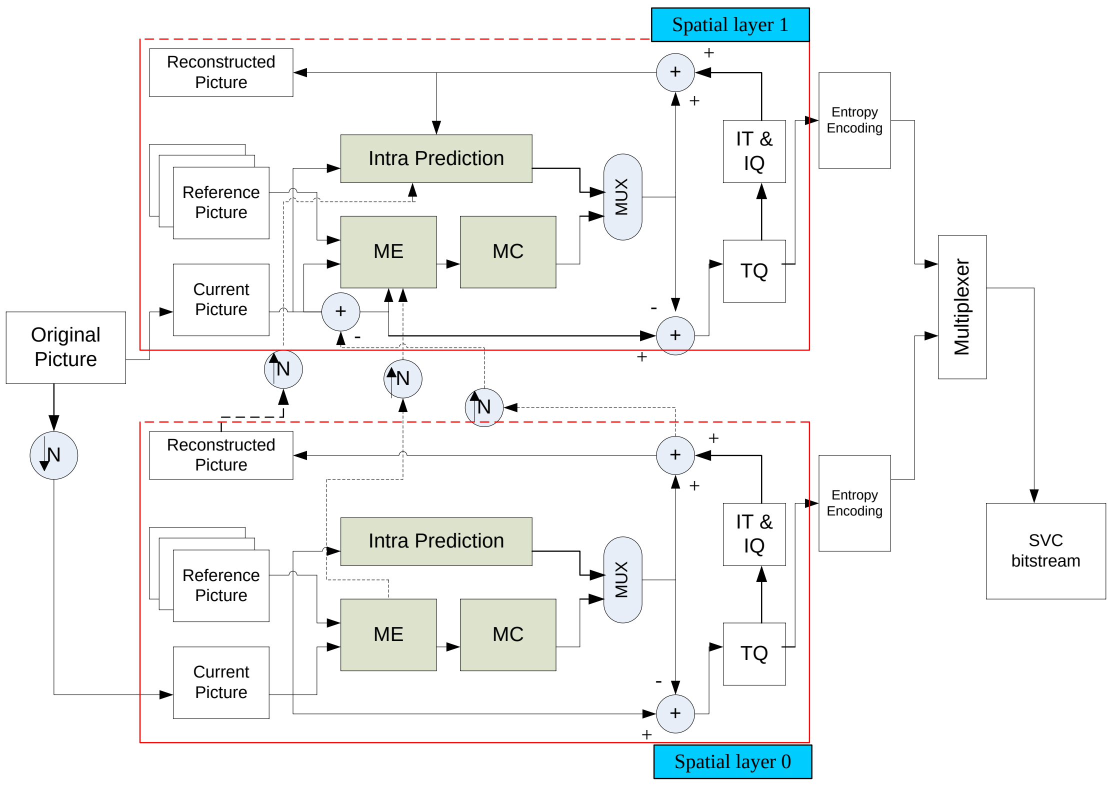


Fig. 1 Structure of SVC encoder

The basic structure of SVC encoder with two spatial layers is shown in Fig. 1. The intra-layer prediction mode is used both in base layer (BL) and enhancement layer (EL). However, for the high correlation between BL and EL, the inter-layer prediction mode is also supported in EL process by reusing the coding information from BL.

In the first step of SVC encoding process, the original input sequence is down-sampled N times to fit the size of BL input. Then the BL sequence is encoded by typical H.264/AVC encoding process. After BL is encoded, the EL takes the up-sampled encoded information from BL as reference to do the inter-layer prediction.

1.2. Organization of this thesis

The organization of this thesis is as follows: In chapter 2, we introduce the related works of this thesis. Afterwards, we proposed a fast mode filtering algorithm for our IME architecture with the adopted pre-selection algorithm of LI's work[7]. In chapter 4, we propose our architecture of the adopted efficient inter-layer prediction algorithm as well as the architecture of the adopted fast algorithm of FME. Then, in chapter 5, we list several simulation results to demonstrate our proposed mode filtering algorithm. Some hardware implementation results of our motion estimation are also listed in chapter5. In the end, a conclusion is given in chapter 6.

Chapter 2. Related work overview

2.1. Parallel multi-resolution motion estimation (PMRME)[1]

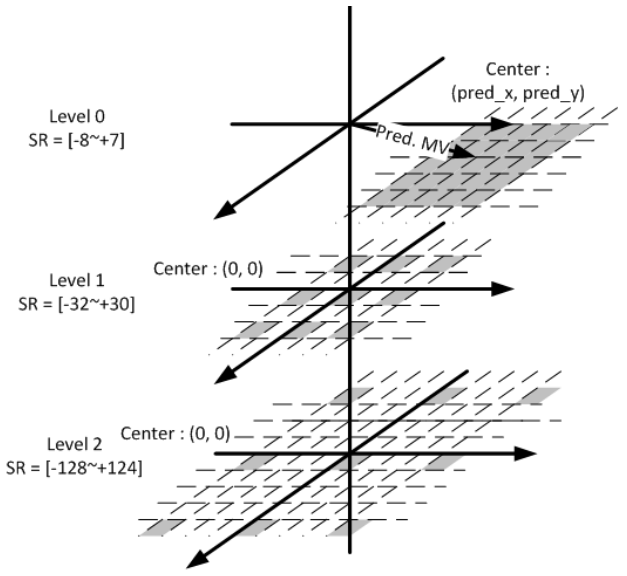


Fig. 2 illustration of parallel multi-resolution motion estimation

Parallel multi-resolution motion estimation (PMRME) includes three independent levels for search, as illustrated in Fig. 2.

Level 2 is the coarsest level. It has the largest SR, [-128,124], and its search center is located on the original point (0, 0) to enable regular data reuse between successive MB processing. This level uses the 16:1 sampling for its ratio, thus, the only mode in level 2 is 16x16.

Level 0 is the finest level, which has SR for [-8, 7]. We choose the motion vector predictor (MVP) as its center because it has high probability to be the final MV. In this level, we do not subsample data, thus, there would be variable block size modes in level 0. We here take the MVP of the top left block as the MVP_INTER of whole macro block to simplify the process and compensate the motion vectors of all blocks after motion estimation is over.

Level 1 has the SR between level 0 and level 2, which is [-32, 30].The search center of level 1 is also set to be on (0, 0) for the same reason of level 2. This level uses 4:1 sampling and thus will have 16x16, 16x8, 8x16, 8x8 modes to choose from.

These three levels have different characteristics and can properly complement to each other. Level 0 can find the best matching block of those with low motion. Level 2, on the contrary, is suitable for high motion block but with the coarsest accuracy. The characteristics of level 1 is among level 0 and level 2, which has smaller SR than level 2 but more accuracy than level 2.

In level 0, after searching all positions, there will be a motion vector difference (mvd) which indicates the difference between the final MV and the MVP_INTER. Thus, we only have to transmit the MVP_INTER and mvd after the encoding is over so that the decoder can get the final MV position. As for level 1 and level 2, the final MV is the mvd relative to zero (0, 0) position.

The advantages of PMRME is that level 1 and level 2 can enhance data reusing by setting search center on (0, 0).Moreover, level 1 and level 2 have larger search range and thus can compensate the drawback of level 0 whose search range is too small to find the best matching block with high motion. With these two large search levels, the motion vectors can rapidly converge to a proper position thus can compensate the effects from level 0 MVP, as illustrated in Fig. 3.

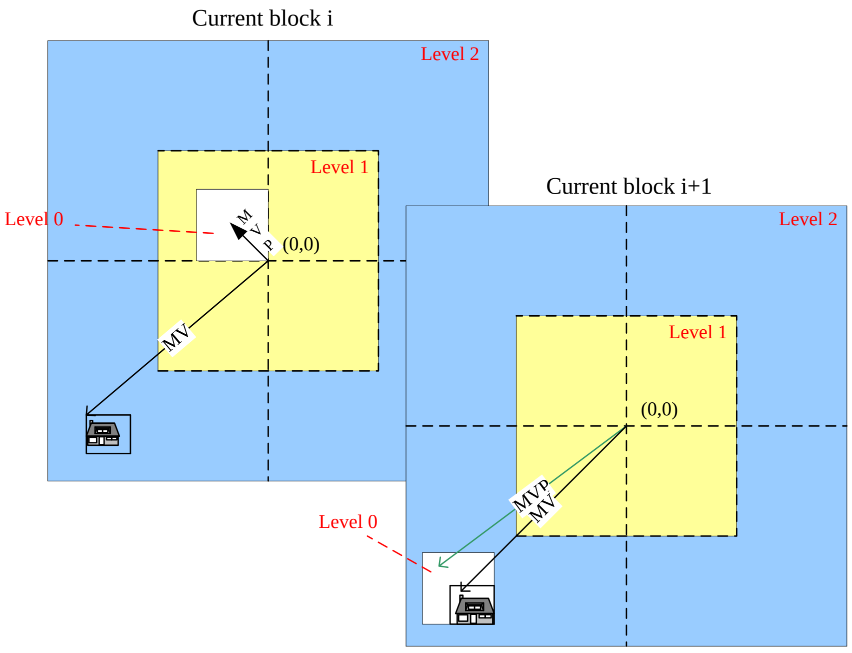


Fig. 3 the concept of PMRME (the MV can be rapidly converged.)

2.2. Data efficient Inter-layer prediction algorithm

Inter-layer (IL) prediction is adopted in SVC to reduce the redundancy existed between spatial layers. However, IL prediction also causes additional memory bandwidth and computational requirements. We adopt our data efficient IL prediction algorithm to reduce the data access requirement. IL prediction includes inter-layer residual (ILR), inter-layer motion (ILM) and inter-BL (IBL) mode and the combination of them, as illustrated in Fig. 4.

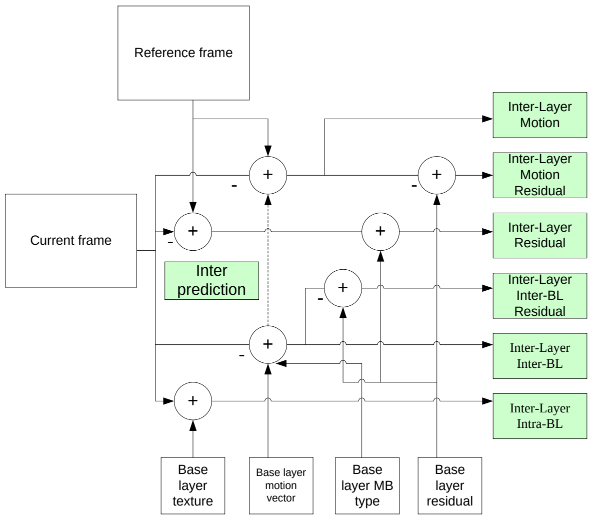


Fig. 4 different modes of the Inter-Layer prediction (IL prediction)

In our IL prediction algorithm, ILR can be achieved by only additionally subtracting the up-sampled base layer residual from current coding pixels after current coding pixels subtracts the reference data, as illustrated in Fig. 5(a)(b).

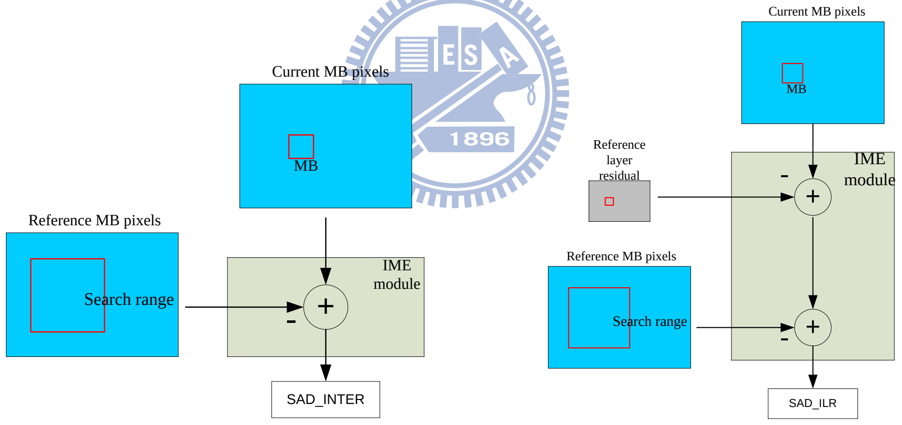


Fig. 5 (a)the SAD calculation of INTER (b) the SAD calculation of ILR

The concept of ILM is to use the up-sampled motion vectors from base layer as the motion vector predictors (MVPs) of enhancement layer, which is based on the assumption that the motion vector of base layer could be quite approximate to the one of enhancement layer.

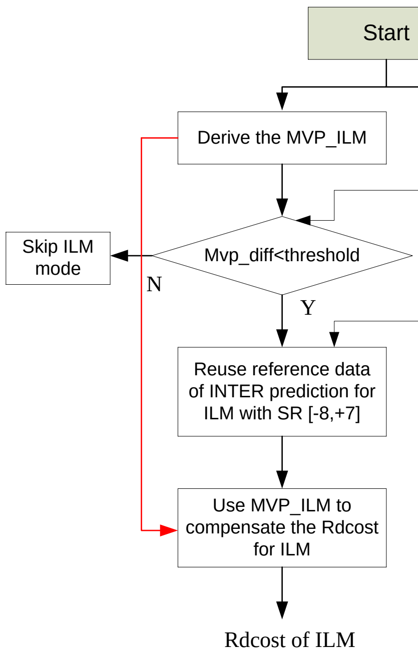


Fig. 6 adopted inter-layer motion prediction algorithm for every block in a macro block. The threshold is set to be 8. The MVP_ILM is based on a 4x4 block, thus, a macro block has sixteen MVP_ILMs.

Our adopted ILM scheme is illustrated in Fig. 6. It takes advantage of the characteristic that the difference between motion vector predictors of ILM (MVP_ILM) and motion vector predictors of INTER (MVP_INTER) is highly possible to be small, so we can apply the search area centered on MVP_INTER to find out best MV of both INTER mode and ILM mode. Moreover, a simulation was conducted to find out the most suitable search range so that the MV_ILM would be highly possible to be within the search range of INTER mode. In this way, INTER mode and ILM mode can share the same search data to reduce the data access requirement.

According to the results of our previous simulation, we set the search range to be [-8, 7]. And the condition for the execution of the ILM process is that the difference between MVP_ILM and MVP_INTER has to be smaller than 8 to assure the final MV of ILM would be inside the search area of INTER mode. In addition, since the search range is only [-8, 7], we can further save the computing time and reduce power consumption.

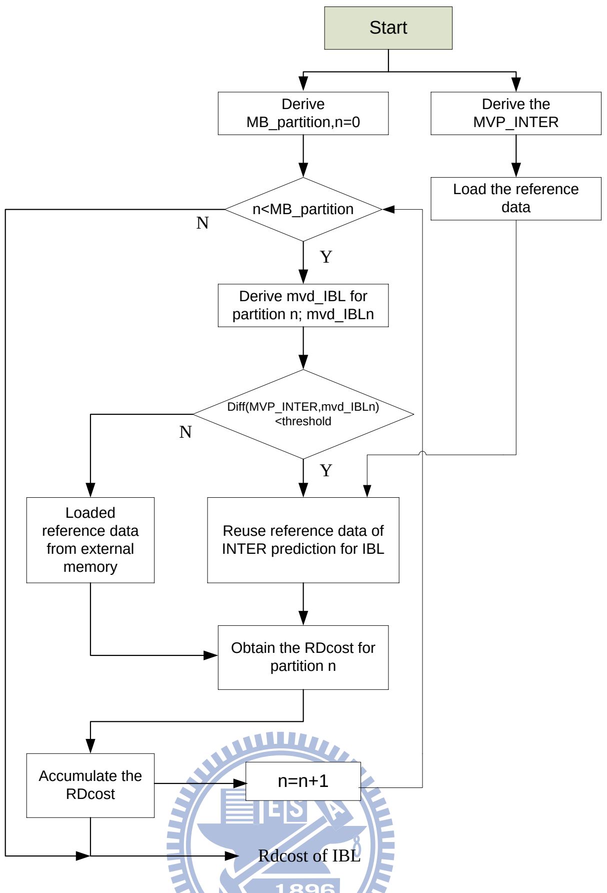

Start
Derive MB_partition,n=0
Derive the MVP_INTER
Load the reference data
n<MB_partition
N
Y
Derive mvd_IBL for partition n; mvd_IBLn
Diff(MVP_INTER,mvd_IBLn) <threshold
N
Y
Loaded reference data from external memory
Reuse reference data of INTER prediction for IBL
Obtain the RDcost for partition n
Accumulate the RDcost
n=n+1
Rdcost of IBL

Fig. 7 the function flow of inter-BL mode(IBL)

The IBL scheme is illustrated in Fig. 7. It takes the up-sampled partition from base layer and uses the same MVP as ILM. Since the IBL mode has high probability to be selected as best mode in enhancement layer, it doesn't skip IBL mode to avoid great performance degradation, instead, it fetches external memory to load reference search data when the difference between MVP_INTER and MVP_ILM is too large. Otherwise, IBL reuses the reference search data from INTER prediction to reduce the data access requirement.

The ILR, ILM, IBL mode can be combined together to further reduce redundancy between spatial layers. Thus, in this thesis, we will have INTER, ILR, ILM, ILMR, IBL and IBLR mode after different modes combined together, as illustrated in Fig. 2.4. The corresponding architecture of our IL prediction will be discussed in detail in chapter 4.

2.3. Fast FME algorithm- Single-pass Fractional Motion Estimation (SPFME)

SPFME is a one-iteration search method which is used as a fast fractional motion estimation. SPFME uses the MVP position and the zero (0, 0) position to set a ten–points search pattern.

The SPFME needs a MVP position to locate the search point of the predicted fractional motion vector (*pred frac mv*) and the other four points around it. The way we adopt to produce the *pred frac mv* is the same as the way adopted in [5]. In H.264/AVC, the predicted motion vector (*pred mv*) is defined as the median of three neighboring motion vectors. The *pred frac mv* is extracted from *pred mv* and the best integer motion vector (mv),

$$MVP\ position = pred\ frac\ mv = (pred\ mv - mv)\ modulo\ \beta \qquad (2.1)$$

where modulo β operation is applied to obtain the fractional component by removing the integer part. The number "β" is decided by the precision, β=4 in 1/4 pel case and β=8 in 1/8 pel case. The basic idea of obtaining the *pred frac mv* according to the equation (1) is based on the assumption that most of the best fractional motion vectors (*best frac mv's*) lies on either *pred frac mv* or its four neighbors (top, down, left and right).

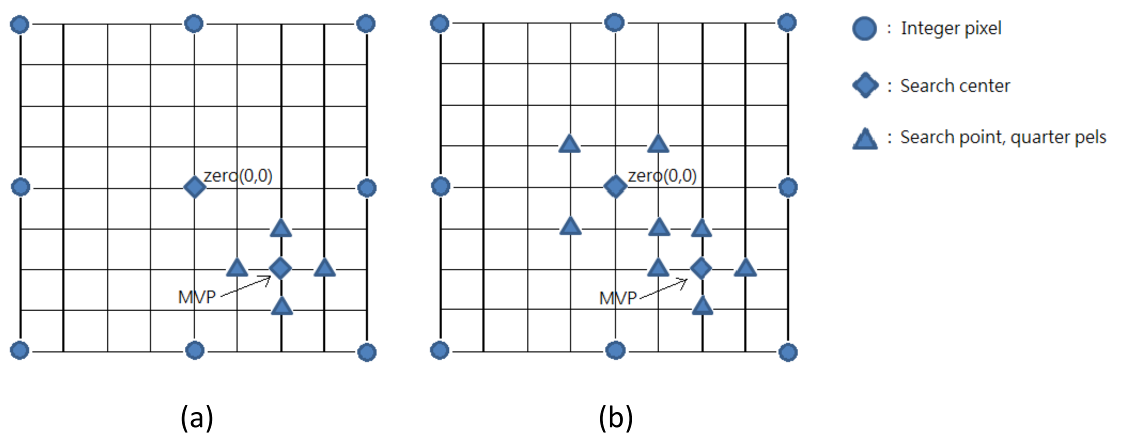


Fig. 8 Different search patterns are shown. Circle, diamond and triangle denote integer point, search center and quarter-pel location. (a) is SIFME, proposed by our early algorithm, searches zero position, MVP position and its four neighbors (up, down, right and left) (b) is SPFME, denoting Kyung's algorithm, with four more points around the zero position (up left, up right, down left, and down right) than (a).

SPFME has a ten-points search pattern, as illustrated in Fig. 8. The pattern includes the zero (0, 0) position with its four neighbors (up left, up right, down left and down right) around it, and the MVP position with its four neighbors (top, down, left and right).

SPFME improves SIFME[3] by adding more points around the zero position and thus increase hit rate of the best MV, as illustrated in Fig., . Furthermore, it can be easily implemented by our previous architecture of SIFME algorithm without increasing computing time because of the parallel calculation between different search points. The proposed architecture design of SPFME will be discussed in detail in chapter 4.

Chapter 3. Mode filtering for IME

3.1. The Matching Criteria

In this section, we use RDcost to decide the final prediction mode. The function of RDcost is listed as follow:

$$J = D + \lambda \bullet R \qquad (3.1)$$

Where J denotes RDcost, λ represents Lagrangian parameter, D is the distortion between current and reference data, and R refers to rate derived by computing the difference between selected motion vector (MV) and motion vector predictor (MVP). In the following section, the "$\lambda \bullet R$" will be simply called "MVcost". Thus, the function of RDcost can be shorten to:

$$J = D + MVcost \qquad (3.2)$$

The D term is acquired by calculating sum of absolute differences (SAD) in IME and sum of absolute transformed differences (SATD) in FME.

3.2. Motivation of mode filtering

In H.264 video coding standard, variable block size (16x16, 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4) motion estimation is supported and every block of every partition size goes through integer motion estimation (IME) and fractional motion estimation (FME) processing, as illustrated in Fig. 9. Thus, 41 blocks would go through IME and FME processing to derive best partition and best motion vectors of a macro block. Furthermore, the computational complexity is even increased due to the adoption of inter-layer prediction in scalable video coding (SVC), including ILR, ILM and ILMR. Therefore, there are overall 41x4=164 blocks that would have to be examined by IME and FME process, as illustrated in Fig. 10.

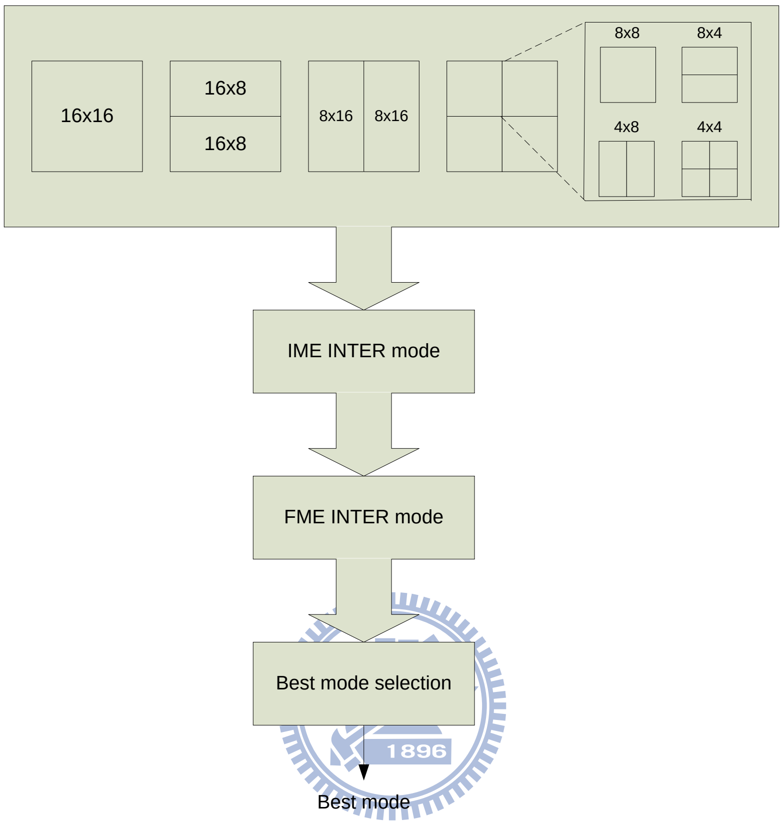
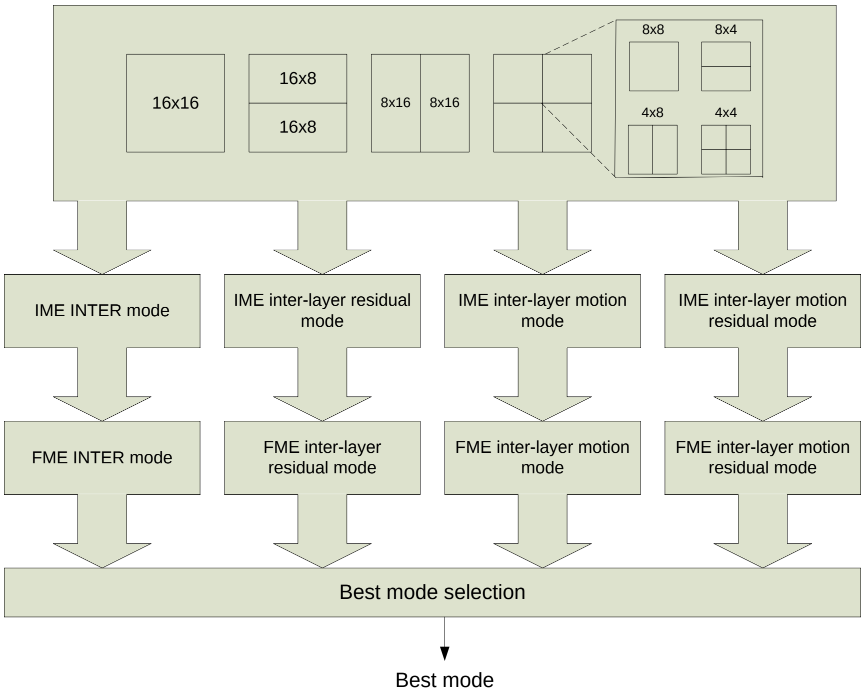


Fig. 9 mode selection process of H.264



16x16
16x8
16x8
8x16
8x16
8x8
8x4
4x8
4x4
IME INTER mode
IME inter-layer residual mode
IME inter-layer motion mode
IME inter-layer motion residual mode
FME INTER mode
FME inter-layer residual mode
FME inter-layer motion mode
FME inter-layer motion residual mode
Best mode selection
Best mode

Fig. 10 mode selection for SVC

To reduce the complexity, partition beyond 8x8 is simplified into submode after IME stage in H.264/AVC, which is derived from 4x4, 4x8, 8x4 and 8x8 mode, as illustrated in Fig. 11. Thus, there are only 16x16, 16x8, 8x16, and submode to be examined in FME stage, namely, 21x4=84 blocks at most and 9x4=36 blocks at least to examined in FME stage.

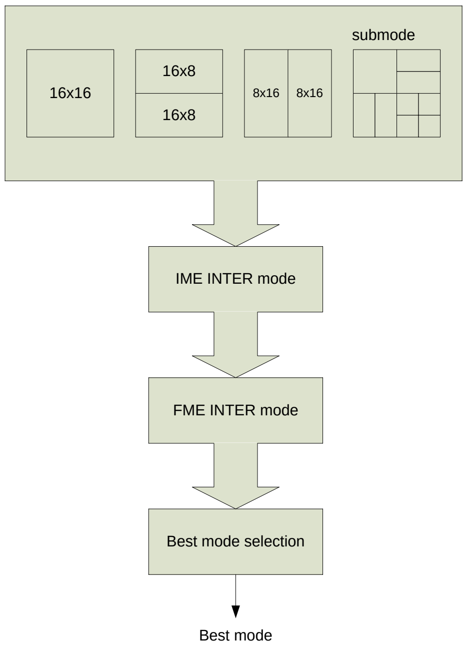


Fig. 11 mode selection for H.264/AVC

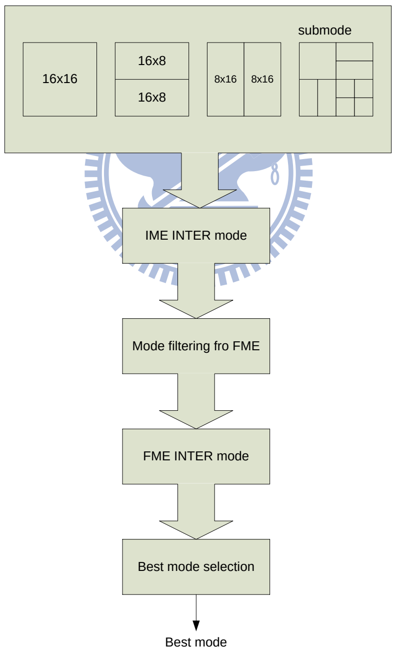


Fig. 12 mode pre-selection for H.264/AVC

Some works [10],[11],[12] have been proposed to speed up the FME process; however, to further reduce the complexity and computing time of FME processing, instead of checking all modes from IME, we think of pre-filtering modes from IME to reduce the number of modes examined in FME. There are some researches [8],[9] on mode-filtering for H.264/AVC, as illustrated in Fig. 12. Nevertheless, those mode-filtering schemes are only for INETR prediction. Since the inter-layer prediction is adopted in our IME process, we adopt the mode pre-selection scheme for inter-layer prediction from the previous work [7] to reduce the numbers of modes in FME stage.

Although we can reduce the numbers of modes from IME by Li's work, we still have nearly 8 modes left after the pre-selection process in our IME stage. To further reduce the computing time of FME as well as considering the hardware implementation of FME, we try to reduce modes for FME to only 3 modes. Thus, after introducing the adopted Li's mode pre-selection algorithm, we still need to handle the remaining modes of inter-layer prediction and the other two modes from level 1 and level 2 in IME stage before entering into FME stage.

The rest of this chapter will be as follows. In 3.2, we introduce the mode pre-selection algorithm by Li. Afterwards, we propose an algorithm to deal with the rest of IME modes in 3.3 so that there will be only 3 modes left entering into FME stage. In 3.4, we further take advantage of the characteristics of inter-BL (IBL) and inter-BL residual (IBLR) mode to reduce the number of mode from IME. Simulation results of the overall mode filtering algorithm will be shown in Chapter 5 to demonstrate the efficiency of the whole mode filtering flow.

3.3. Efficient pre-selection algorithm for fractional motion estimation in H.264/AVC scalable video extension

3.3.1. Observation and analysis

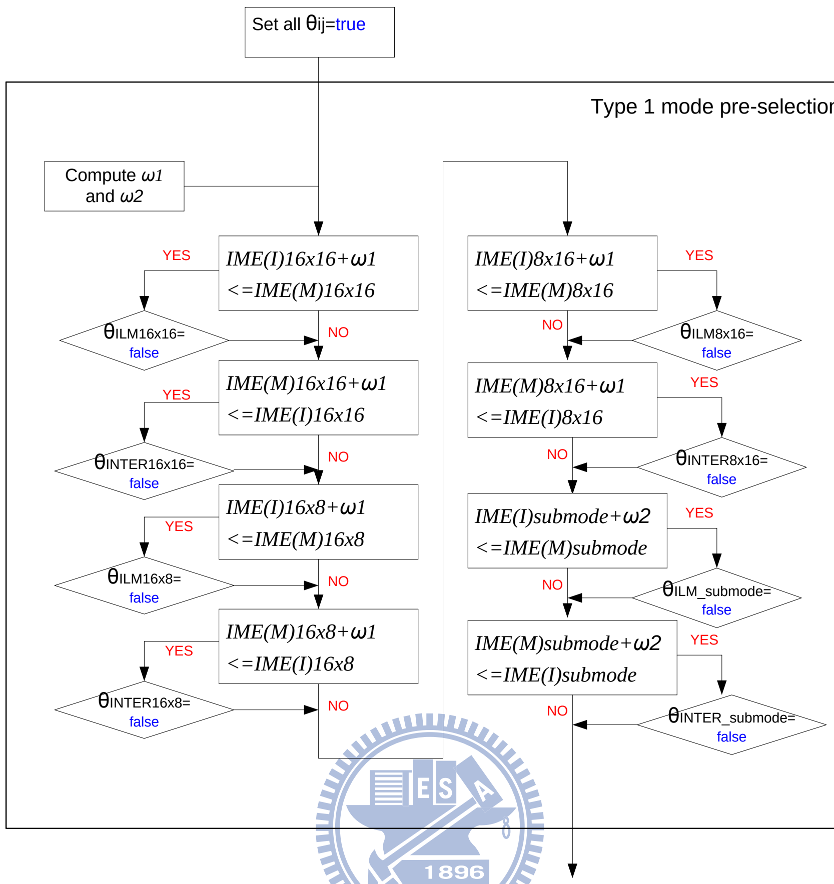
The pre-selection algorithm [7] is based on the observation of the RDcost between IME and FME of different prediction modes. Li divided these prediction modes into four types to compare the RDcost of them. They are *"INTER versus Inter-layer motion (Type 1)"*, *"Inter-layer residual versus Inter-layer motion residual (Type 2)"*, *"INTER versus Inter-layer residual (Type 3)"*, and *"Inter-layer motion versus Inter-layer motion residual (Type 2)"*.Here we only consider the *Type 1* and *Type 2* algorithm for convenience. Thus, we will only introduce the *Type 1* and *Type 2* algorithm in next section.

Li define the term called "spatial locality" to indicate that if a macro block whose RDcost of IME is very close to the RDcost of FME. That is, the RDcost won't change a lot after the FME process. Li found that most of macro blocks have high spatial locality for block size of 16x16, 16x8, and 8x16. Thus, for example, if the IME RDcost of INTER mode is sufficiently larger than IME RDcost of ILM mode for block size of 16x16, it has high probability that the FME RDcost of INTER mode is larger than the FME RDcost of ILM mode for block size of 16x16, which can be illustrated as follows:

$$\begin{gathered}\text{if}\left(\text{RDcost}_{\text{IME}\,(\text{INTER}_{16\text{x}16})} + \text{threshold} \leq \text{RDcost}_{\text{IME}(\text{ILM}_{16\text{x}16})}\right) \\ => \text{RDcost}_{\text{FME}(\text{INTER}_{16\text{x}16})} \leq \text{RDcost}_{\text{FME}(\text{ILM}_{16\text{x}16})} \qquad (3.3)\end{gathered}$$

As for submode, it should be treated individually since the spatial locality of it isn't obvious. The *Type 1* and *Type 2* mode pre-selection algorithms are described in the following section.

3.3.2. The pre-selection algorithm for inter-layer prediction



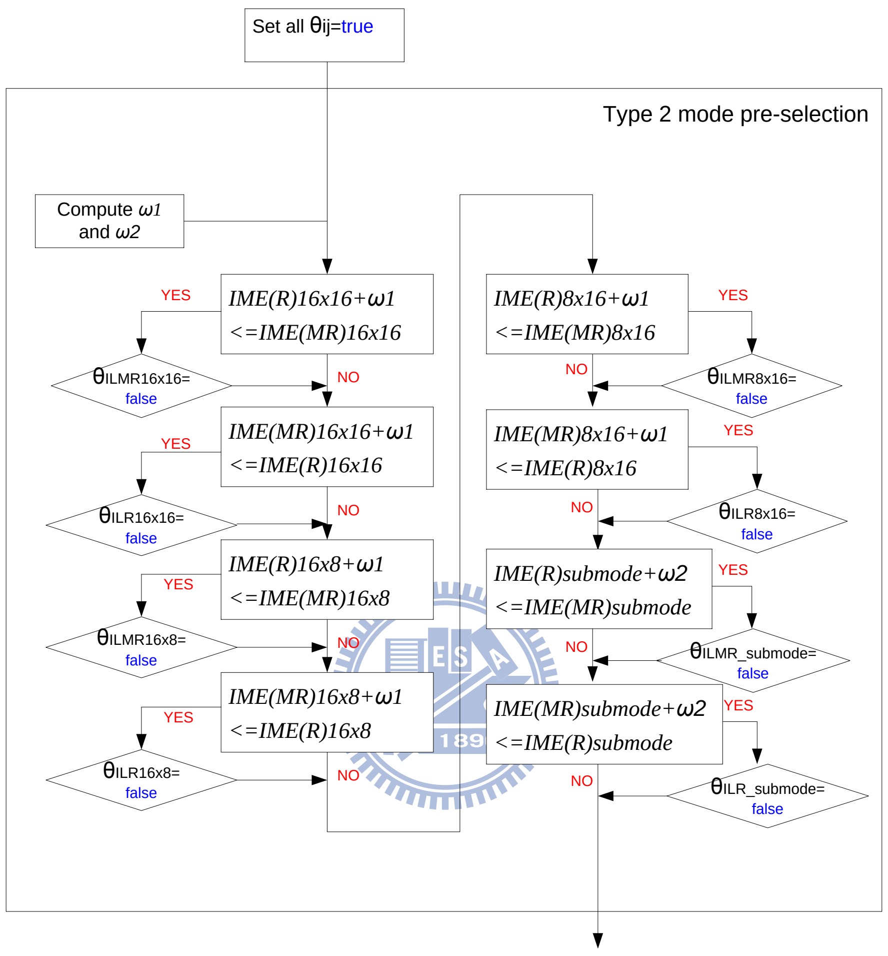
Set all θij=true
Type 1 mode pre-selection
Compute ω1 and ω2
IME(I)16x16+ω1 <=IME(M)16x16
YES
θILM16x16= false
NO
IME(M)16x16+ω1 <=IME(I)16x16
YES
θINTER16x16= false
NO
IME(I)16x8+ω1 <=IME(M)16x8
YES
θILM16x8= false
NO
IME(M)16x8+ω1 <=IME(I)16x8
YES
θINTER16x8= false
NO
IME(I)8x16+ω1 <=IME(M)8x16
YES
θILM8x16= false
NO
IME(M)8x16+ω1 <=IME(I)8x16
YES
θINTER8x16= false
NO
IME(I)submode+ω2 <=IME(M)submode
YES
θILM_submode= false
NO
IME(M)submode+ω2 <=IME(I)submode
YES
θINTER_submode= false
NO

Fig. 13 the *Type 1* mode pre-selection algorithm. The term *"I"*, and *"M"* refers to INTER, and ILM mode, respectively. *IME(I)16x16, IME(I)16x8, IME(I)8x16,* and *IME(I)submode*, are 16x16 RDcost, 16x8 RDcost, 8x16 RDcost, and submode RDcost of IME for INTER mode. *IME(M)16x16, IME(M)16x8, IME(M)8x16,* and *IME(M)submode*, are 16x16 RDcost, 16x8 RDcost, 8x16 RDcost, and submode RDcost of IME for ILM mode. ω1 is the threshold for block size of 16x16, 16x8 and 8x16 ; ω2 is the threshold for submode.

The *Type 1* mode pre-selection algorithm is as illustrated in Fig. 13. θij indicates the "mode_flags" of different prediction modes and different block types. The term "i" refers to the prediction mode and the term "j" represents the block types. In Fig, the thresholds ω1 and ω2 are calculated as follows:

$$\omega = \begin{cases} \omega_1 = \frac{1}{3}\sum_{m\in\{16\times16,16\times8,8\times16\}} |IME(I)_m - IME(M)_m|_{Mode\in\{16\times16,16\times8,8\times16\}} \\ \omega_2 = 0, \ \ Mode \in \{Submode\} \end{cases} \tag{3.4}$$

where the ω1 is the threshold for block size of 16x16, 16x8 and 8x16 because of the spatial locality of them. In the begging of the algorithm, we first set **θij (**mode_flag**)** true for all i and j. After the *Type 1* algorithm, some of the mode_flags would become false.



Set all θij=true
Type 2 mode pre-selection
Compute ω1 and ω2
IME(R)16x16+ω1 <=IME(MR)16x16
YES
θILMR16x16= false
NO
IME(MR)16x16+ω1 <=IME(R)16x16
θILR16x16= false
IME(R)16x8+ω1 <=IME(MR)16x8
θILMR16x8= false
IME(MR)16x8+ω1 <=IME(R)16x8
θILR16x8= false
IME(R)8x16+ω1 <=IME(MR)8x16
θILMR8x16= false
IME(MR)8x16+ω1 <=IME(R)8x16
θILR8x16= false
IME(R)submode+ω2 <=IME(MR)submode
θILMR_submode= false
IME(MR)submode+ω2 <=IME(R)submode
θILR_submode= false

Fig. 14 the *Type 1* mode pre-selection algorithm, the term *"R",* and *"MR"* refers to ILR, and ILMR mode, respectively. *IME(R)16x16, IME(R)16x8, IME(R)8x16,* and *IME(R)submode*, are 16x16 RDcost, 16x8 RDcost, 8x16 RDcost, and submode RDcost of IME for ILR mode. *IME(MR)16x16, IME(MR)16x8, IME(MR)8x16,* and *IME(MR)submode*, are 16x16 RDcost, 16x8 RDcost, 8x16 RDcost, and submode RDcost of IME for ILMR mode. ω1 is the threshold for block size of 16x16, 16x8 and 8x16 ; ω2 is the threshold for submode.

The *Type 2* mode pre-selection algorithm is as illustrated in Fig. 14. The thresholds here are calculated as follows:

$$\omega = \begin{cases} \omega_1 = \frac{1}{3}\sum_{m\in\{16\times16,16\times8,8\times16\}} \left| IME(R)_m - IME(MR)_m \right|_{Mode\in\{16\times16,16\times8,8\times16\}} \\ \omega_2 = 0,\ \ Mode \in \{Submode\} \end{cases} \quad (3.5)$$

After the *Type 1* algorithm and *Type 2* algorithm, we can skip modes with false mode_flags, which are more unlikely to be the best mode after FME stage. Thus, we can reduce the computing time of FME.

3.4. Proposed mode filtering algorithm for multi-level

After going through Li's mode pre-selection algorithm, there are still nearly 8 modes left in average for level 0 in IME. Except for the rest of level 0 modes, level 1 and level 2 have their own prediction modes, too. Thus, in this section, we focus on how to process the rest of modes.

To assure that the mode filtering would be accurate enough, we deal with level 0 modes and level1, level 2 modes separately. For level 0, we choose three best modes from the modes left according to RDcosts performance. Afterwards, we will have 3 modes from level 0, one mode from level 1, and one mode from level 2. To further reduce the number of modes to only 3, we propose a fast algorithm to select 3 modes among these 5 modes. The way we select the 3 best modes is illustrated in Fig. 15.

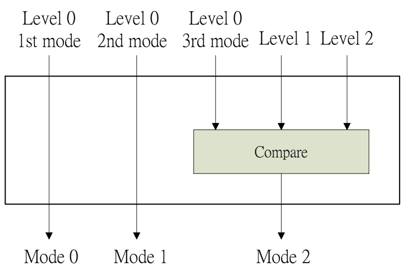


Fig. 15 the function of mode filtering between multi-levels

In the algorithm, we set the modes with the smallest RDcost and the second small RDcost from level 0 as the first and the second candidate for the final modes. As for the third candidate for final mode, we compare the RDcost of the third mode of level 0, level 1 RDcost, and level 2 RDcost to determine the best as the third candidate for the final mode. The reason we keep 2 candidates from level 0 is due to its high prediction accuracy compared with the other two levels, since the other two levels use sub-sampled data to calculate RDcosts. In this way, we can quickly decide the candidates for the final modes in FME as well as make sure that the candidates from IME are accurate enough.

3.5. Mode filtering by IBL and IBLR mode

IBL mode and IBLR mode are two special modes in inter-layer prediction since they directly take the up-sampled MVP_ILMs as their final MVs. In this way, IBL and IBLR will load data according to the final MVs and produce the RDcost without searching. Moreover, since the MVP_ILMs are in 1/4-pels unit, the RDcosts of IBL and IBLR after IME stage will be the same as the RDcosts after FME stage. Hence, we don't need to put IBL and IBLR modes into FME if either of them is in the modes left after IME stage. Namely, we can skip IBL or IBLR mode when they are in the last 3 modes from IME, as illustrated in Fig. 16.

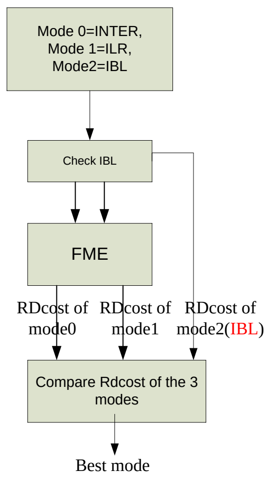


Fig. 16 the function flow of mode filtering by IBL mode

In Fig. 16, mode 2 is IBL mode while the other two is not, thus, we skip mode2 and directly send the RDcost of mode 2 to the compare block. That is, in this case, we only have to process 2 modes in FME. Moreover, if IBL and IBLR are both in the 3 modes from IME, we can further reduce the number of modes to only 1. With this IBL skip mechanism, we will have at most 3 modes and at least 1 mode to process in FME stage.

Chapter 4. Hardware architecture design

4.1. Design Spec

The desired system specification is described as follows: an SVC encoder works under 135 MHz clock frequency with 3 quality layers, 3 spatial layers (CIF, SD 480p, and HD 1080p), and frame rate is set to be 60fps. To achieve this tough specification, we encode two frames at the same time, thus, two MBs from different frames will be encoded in parallel.

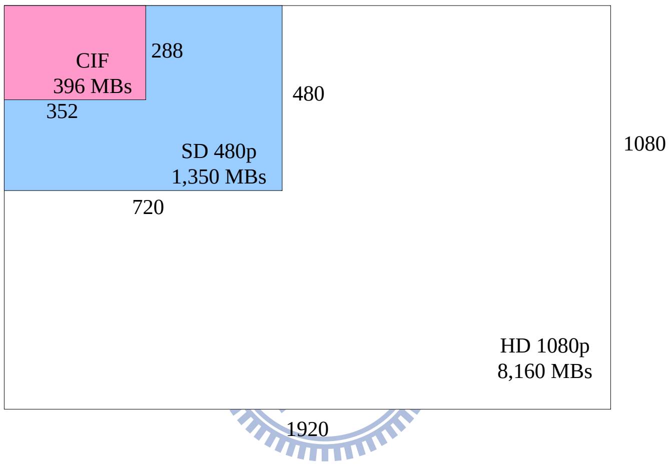


Fig. 17 the three spatial layers for our spec

According to the spec and Fig. 17, we can deduce the needed cycle time as calculated below:

$$396 + 1{,}350 + 8{,}160 = 9906\ MBs \qquad (4.1)$$

Since we encode two frames simultaneously with 60fps, and the frequency is 135 MHz, the cycles for encoding a MB will be:

$$135M \div \left((9906 \div 2) \times 60\right) = 454\ cycles \quad .. \qquad (4.2)$$

According the spec, our design is implemented through pipelined stages as shown in Fig. 18. The time of one pipelined stage is 450 cycles. We have two sets of IME, inter-layer prediction, mode filtering, FME, intra prediction, Deblocking, and Entropy coding modules to process two MBs of different frames at the same time. As for the other modules, we only have one set of them since they are able to process two MBs' data in time.

The first stage is the IME process with the IL prediction. Mode filtering is also in the first stage after the IME and IL prediction. After stage 1 finishes, it pass the best 3 prediction modes and their MVs to the second stage. FME and intra prediction are in the second stage with the transform, quantization, inverse transform, and inverse quantization modules. After the second stage, best residuals, best mode, and best MVs are sent into the third stage. CGS, reconstruction, and deblocking are processed in the third stage. Finally, in the fourth stage, entropy coding would be processed to get the final output.

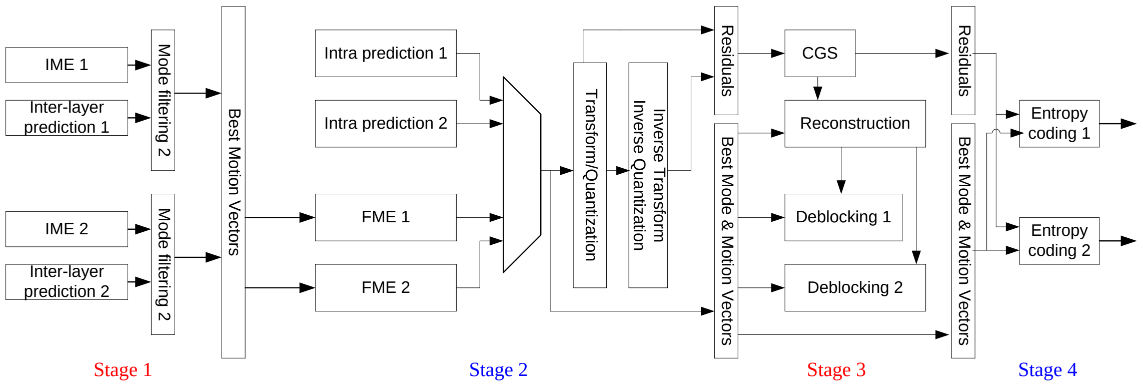


Fig. 18 the pipelined architecture of our H.264/AVC scalable video encoder

4.2. Architecture design of IME

4.2.1. Overview of PMRME architecture design

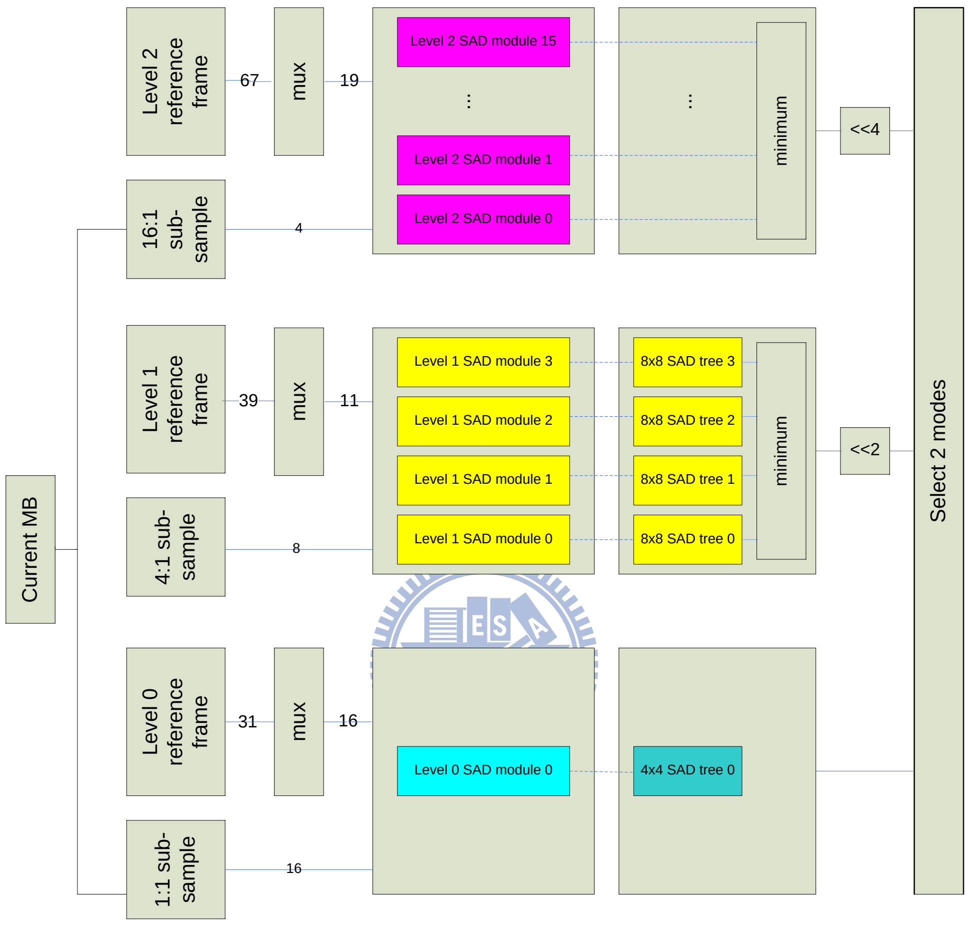


Fig. 19 the PMRME architecture(the number on the line is the number of pixels)

Fig. 19 shows the PMRME [1] architecture and one 16x16 current block data is shared for the three levels with different sample ratios. After the reference selection module, "Level X (0, 1 or 2) ME module" calculates distortion then output the outcome to the "Level X tree module". The "Level X tree module" is in charge of summing up SADs to further generate the SADs of different block sizes as well as add MVCOST to distortion to form RDcost. After costs from three levels are produced, the best two candidates will be selected to enter into FME process.

The reference selection modules for different levels have different bandwidths due to two factors, different search ranges and parallelism of SAD calculation. Level 0 searches one position at a time. Thus, level 0 reference data inputs 16 pixels to “Level 0 ME module” and allocate the whole 16 pixels to a “Level 0 SAD module”.

Level 1 searches four positions at a time to speed up the processing. Thus, level 1 reference data inputs 8+3=11 pixels to “Level 1 ME module” and allocate to four “Level 1 SAD modules”, each one has 8 pixels input, as illustrated in Fig. 20.

Level 2 searches sixteen positions at a time to speed up the processing. Thus, level 2 reference data inputs 4+15=19 pixels to “Level 2 ME module” and allocate to sixteen “Level 2 SAD modules”, each one has 4 pixels input, as illustrated in Fig. 21.

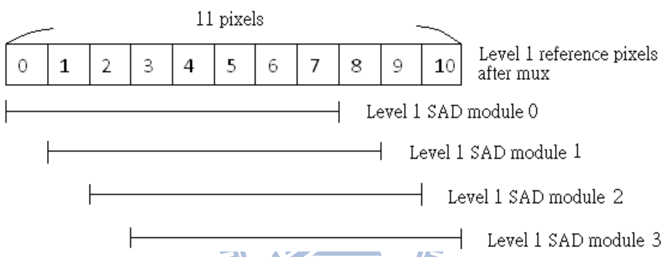


Fig. 20 level 1 reference data allocation for “Level 1 SAD modules”

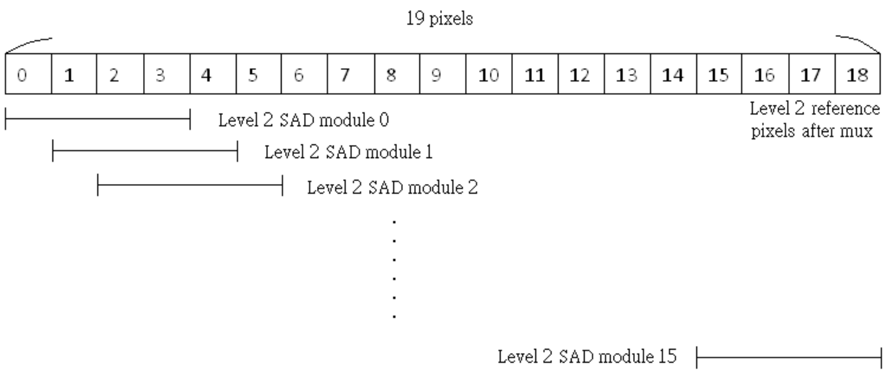


Fig. 21 level 2 reference data allocation for “Level 2 SAD modules”

Every level has its own “ME module”. The hierarchy of ME module is: ”ME module” >”SAD module” >”Row ME module”> ”Primitive module.” In this way, the “SAD module” and “row ME module” of each level can be shown below. The primitive module is shown in Fig. 22.

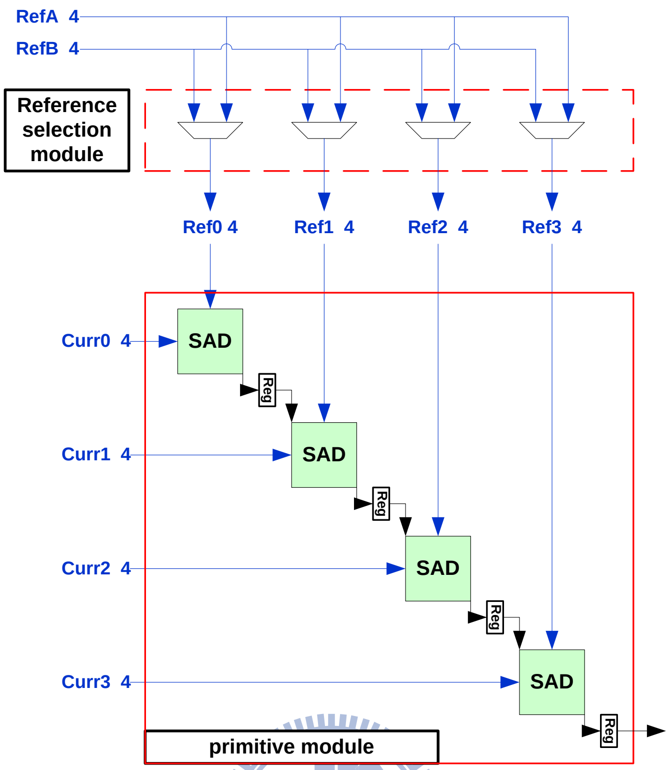


Fig. 22 the primitive module. RefA and RefB are input for reference data, Cur0~Cur3 is the input for current block data (the number on the line means the number of pixels)

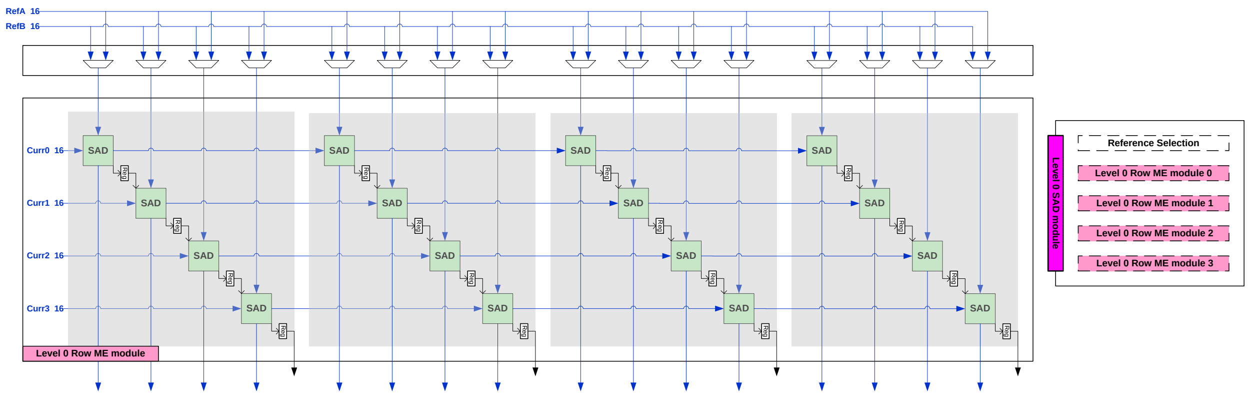


Fig. 23 “Level 0 Row ME module” (left) and “Level 0 SAD module” (right)

“Level 0 ME module” is shown in Fig. 23. It has one “Level 0 SAD module”, and the “Level 0 SAD module” has four “level 0 Row ME modules”. Every “level 0 Row ME module" has four "primitive modules" and every "primitive module" can process a 4x4 block data. Thus, a "level 0 ME module" has 16 "primitive modules" in total.

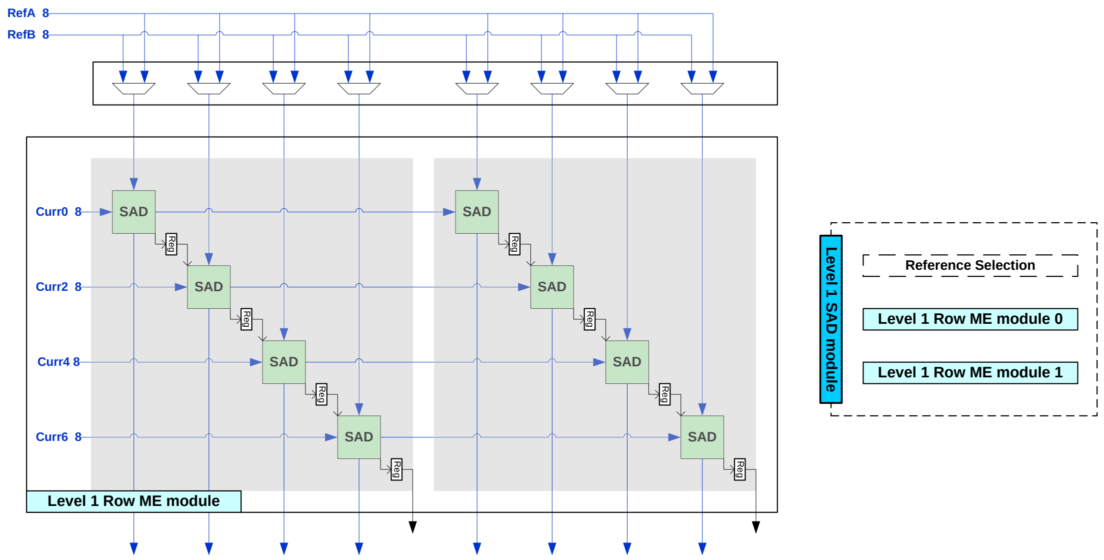


Fig. 24 "Level 1 Row ME module" (left) and "Level 1 SAD module" (right)

"Level 1 ME module" is shown in Fig. 24. It has four "Level 1 SAD modules" to process four points searching in parallel. And every "Level 1 SAD module" has two "level 1 Row ME modules". Every "level 1 Row ME module" has two "primitive modules" and every "primitive module" can process a 4x4 block data. Thus, a "level 1 ME module" has sixteen "primitive modules" in total.

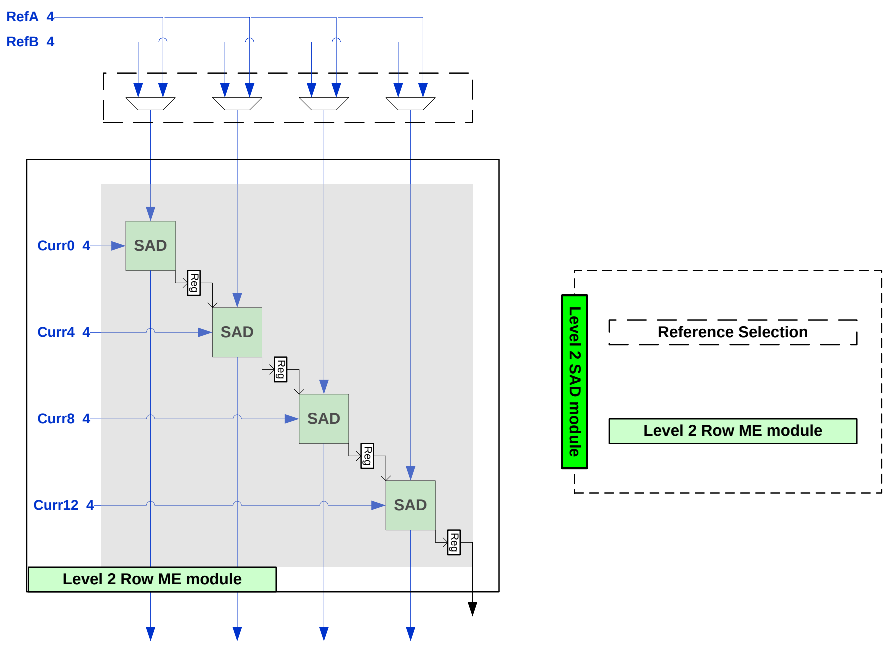


Fig. 25 “Level 2 Row ME module” (left) and “Level 2 ME module” (right)

“Level 2 ME module” has sixteen “Level 2 SAD modules” to process sixteen points searching in parallel, and every “Level 2 SAD module” has one “level 2 Row ME module”. Every “level 2 Row ME module” has one “primitive module” and every “primitive module” can process a 4x4 block data. Thus, a “Level 2 ME module” has sixteen “primitive modules” in total.

The SAD tree for level 0 is a “4x4 SAD tree” since level 0 has various combinations of seven kinds block types and needs the basic unit to compose SADs for all partitions. As for level 1, a ”8x8 SAD tree” is used because level 1 only has block types with sizes beyond 8x8 due to the sampling ratio. Since level 2 only has block type of 16x16, it doesn’t need a SAD tree to calculate RDcosts for different block sizes. The architecture of “4x4 SAD tree” and “8x8 SAD tree” are shown Fig. 26 and Fig. 27.

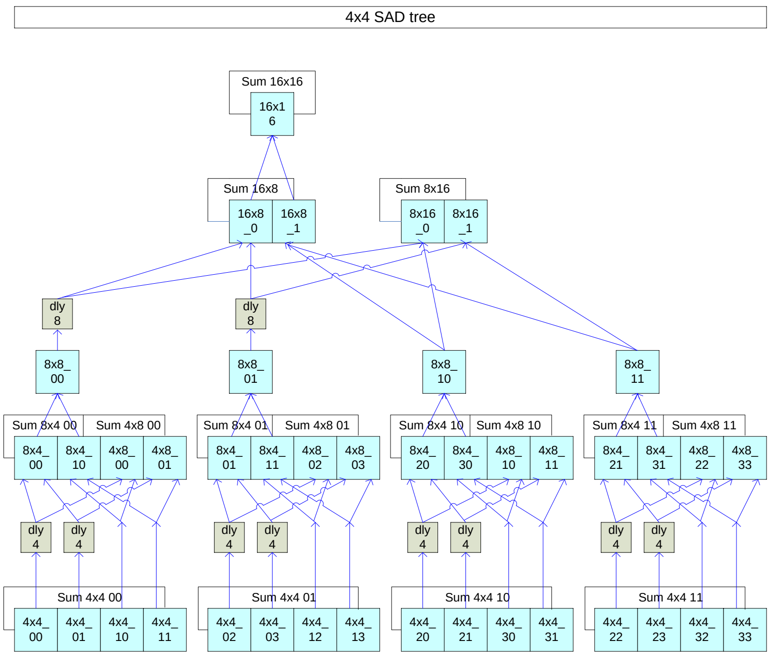


Fig. 26 “4x4 SAD tree for level 0”

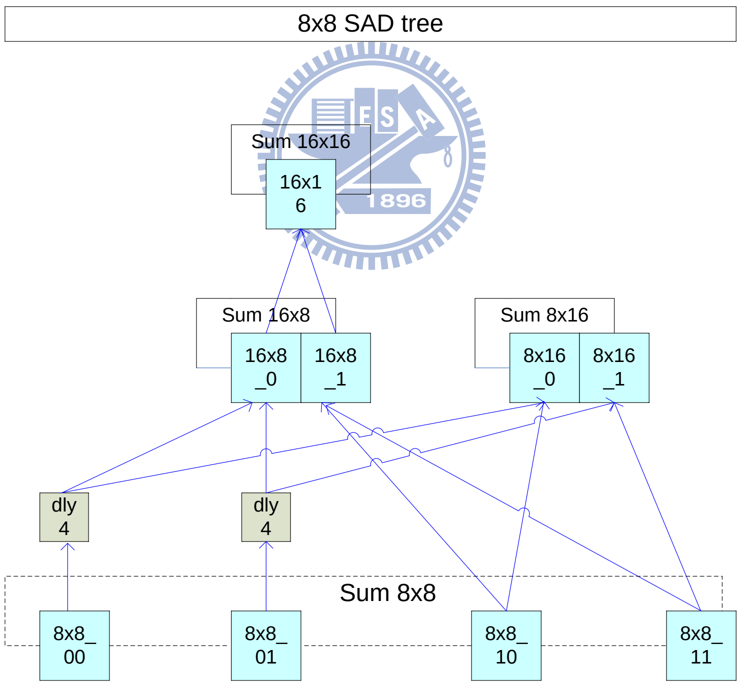


Fig. 27 the “8x8 SAD tree for level 1”

4.2.2. Proposed architecture design for inter-layer prediction

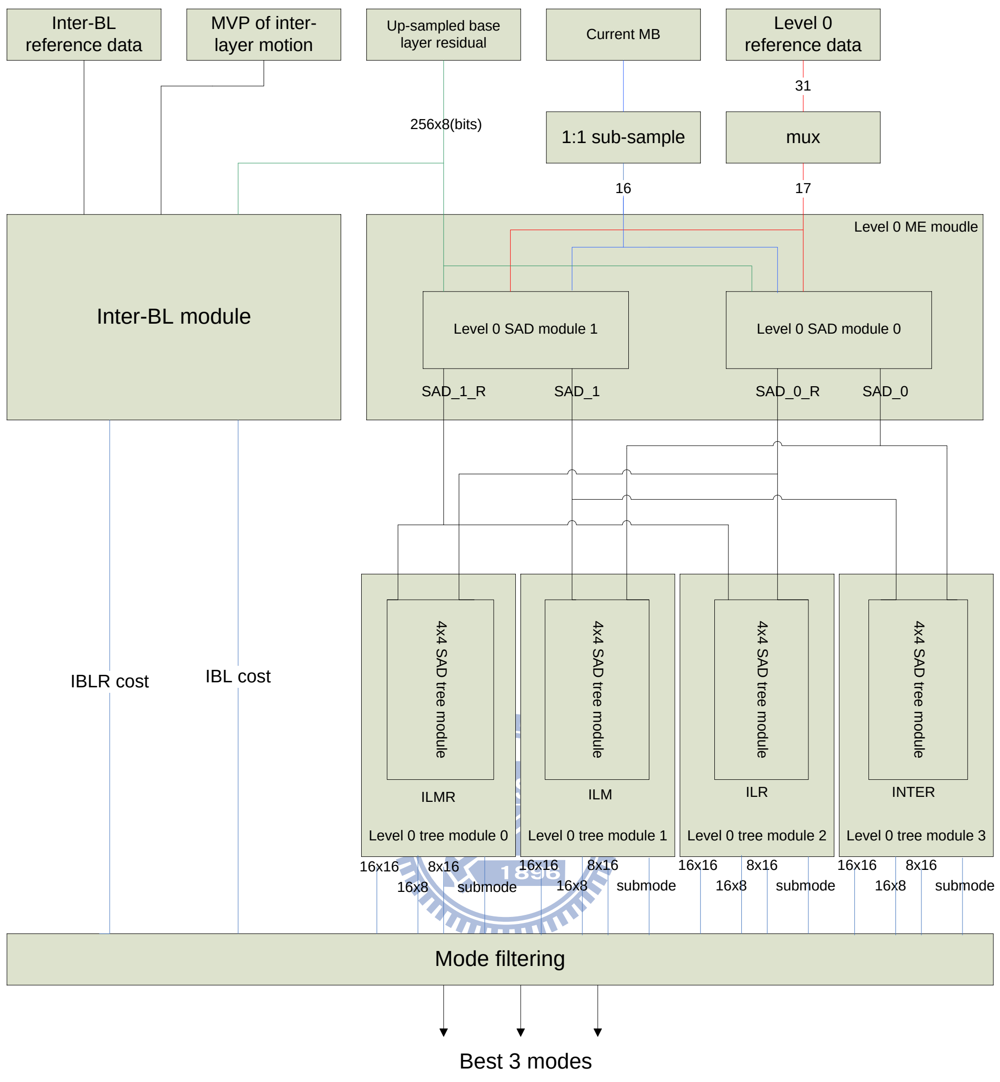


Fig. 28 the proposed inter-layer prediction architecture

Fig. 28 shows the proposed inter-layer (IL) prediction architecture design based on our data efficient inter-layer prediction algorithm. The IL architecture is implemented in level 0 since the data during RDcost calculation of level 1 and level 2 are already sub-sampled. Thus, the reference data of IL prediction is loaded from level 0 reference frame and the current MB data is loaded from 1:1 sub-sample module.

First, to meet our spec, we duplicate a “Level 0 SAD module” to simultaneously process SAD calculations of two positions. Therefore, we will have two modules, “Level 0 SAD module 0” and “Level 0 SAD module 1” during the SAD calculation stage in level 0.

Moreover, since the only difference between INTER mode and inter-layer residual (ILR) mode is that ILR needs to additionally subtract up-sampled residual from current MB data, we input up-sampled residual from base layer into “Level 0 ME module” and allocates it to both “Level 0 SAD module 0” and “Level 0 SAD module 1” to additionally get SADs of ILR mode. In this way, a “Level 0 SAD module” can produce four 4x4 SADs of ILR and four 4x4 SADs of INTER for two position every four cycles, as illustrated in Fig. 29. In Fig. 29, the upBR represent the up-sampled residual from base layer and the Curr refers to current MB data. The “RefA” and “RefB” indicate the reference data of different banks from level 0 SRAM. The SAD module will calculate both (curr-ref) and (curr-ref-upBR) in the same time.

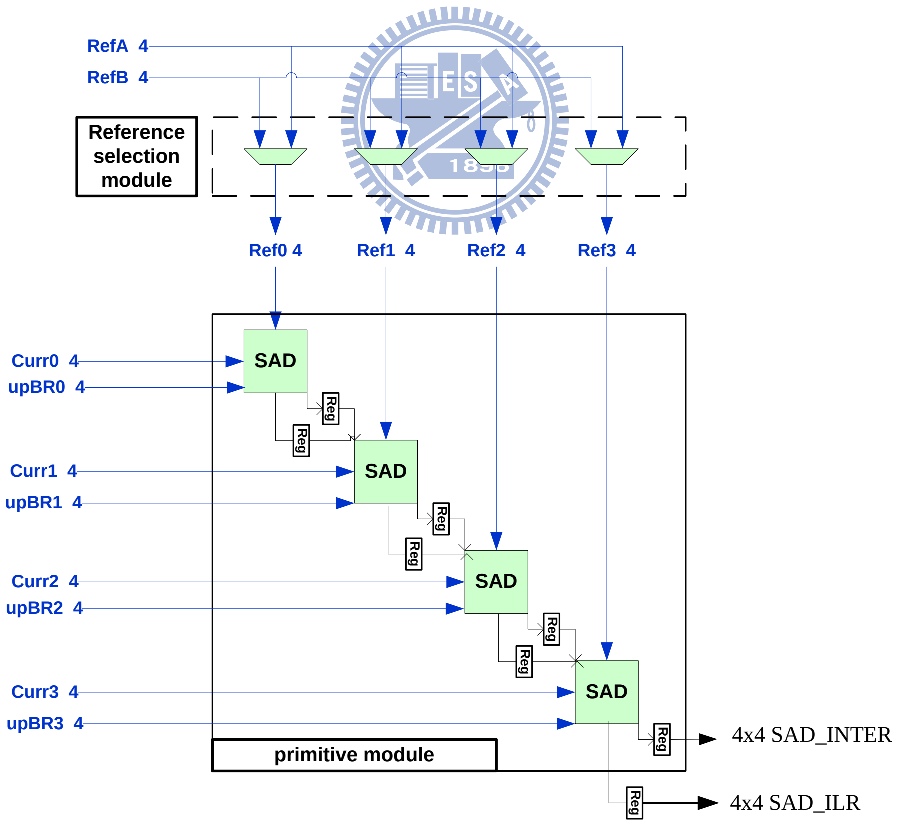


Fig. 29 the primitive module with inter-layer residual mode. The output will be 4x4 SAD of INTER mode and 4x4 SAD of inter-layer residual mode.

After the SADs are produced by "Level 0 ME module", there will be SAD_0, SAD_0_R, SAD_1 and SAD_1_R which indicate SAD of first position, SAD of first position for ILR mode, SAD of second position and SAD of second position for ILR mode, respectively, and each of them contains four 4x4 SADs.

To implement INTER, ILR, ILM and ILMR, we duplicate three more "4x4 SAD trees" as mentioned before. Thus, there are four "Level 0 tree modules" and each of them is responsible for producing 16x16, 16x8, 8x16 and submode block type RDcosts of its prediction mode.

Furthermore, to compare costs of two different positions at a time, we duplicate a "4x4 SAD tree" to process RDcosts of two different positions at the same time. Thus, two search positions as well as different prediction modes (INTER, ILR, ILM or ILMR) can be processed simultaneously.

While the SAD tree is forming SADs of different block types, the RDcost of different block types are also formed. RDcosts of two positions of a block are compared so that the better position and its RDcost would be saved. The best position and its RDcost would be change until better RDcost of a position formed.

After the process of "Level 0 tree module" is over, there will be sixteen RDcosts including of 16x16, 16x8, 8x16 and submode of INTER, ILR, ILM and ILMR prediction mode. Accompanied with inter-BL mode, inter-BL residual mode, INTER mode of level 1 and INTER mode of level 2, the twenty modes would be filtered and selected by a pre-selection module. Before enter into FME process, there are only three modes left to be processed. The architecture of pre-selection process will be discussed in detailed in latter section.

As for the inter-BL prediction (IBL), instead of searching for best position, it simply uses the MVPs of ILM as its final motion vectors. Thus, it has a quite different architecture from other prediction modes, as illustrated in Fig. 30. In the following section, we simply call the MVPs of ILM as "mvd_IBLs" since they are the final MV of IBL mode.

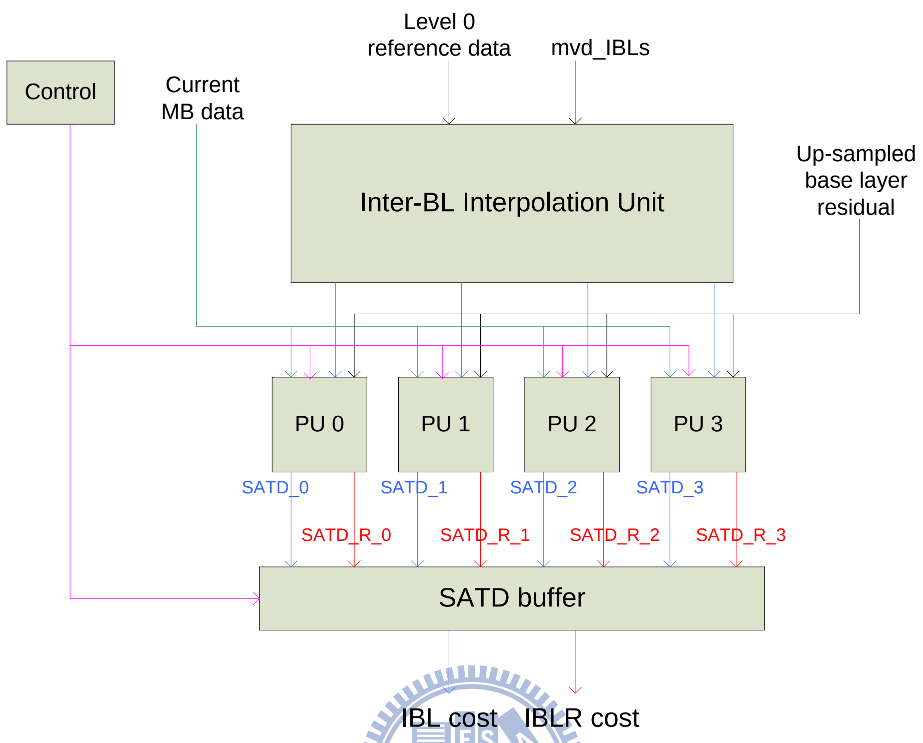


Fig. 30 the architecture of Inter-BL

The architecture of IBL mode includes three parts, Interpolation, PU and SATD buffer. Since mvd_IBLs are in 1/4 unit, we have to interpolate the reference data accessed from memory to produce fractional-pels . After interpolation, each PU calculates its satd and passes to SATD Buffer. SATD buffer then sums them up to acquire the final cost of IBL.

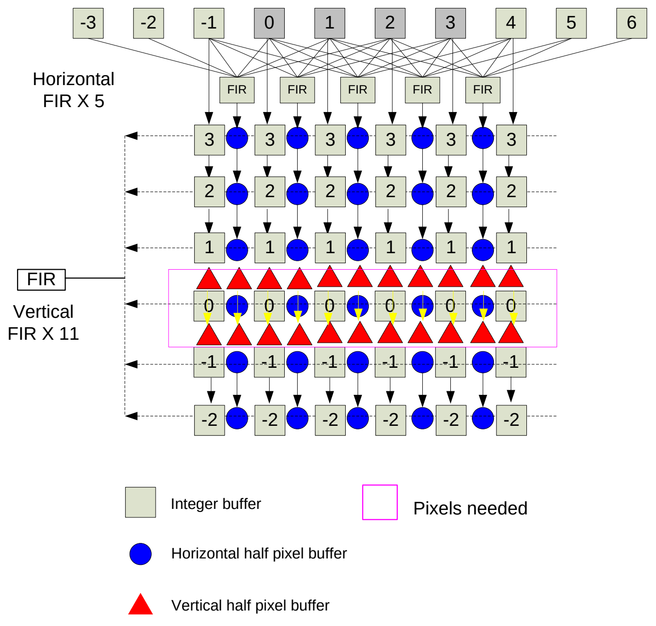
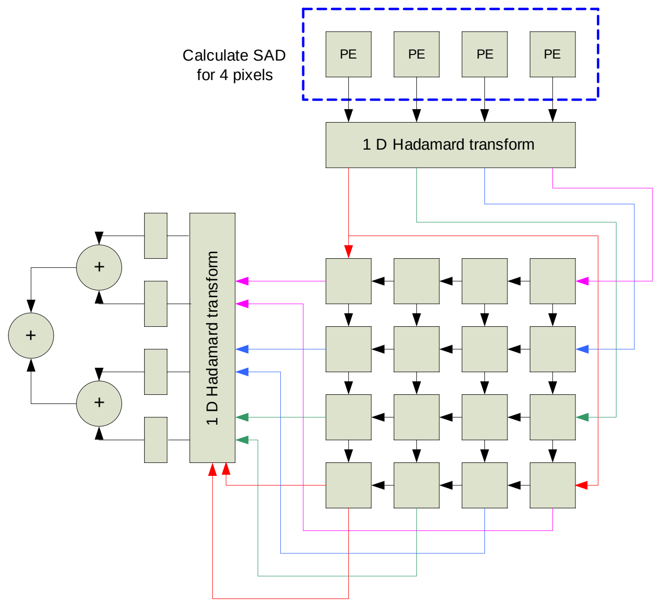


Fig. 31 Architecture of interpolation unit. FIR is the 6 tap 1D filter.

The interpolation unit is shown in Fig. 31. It loads the a row of reference data cycle by cycle, which contains 10 adjacent integer points, and immediately interpolates horizontal half pixels by the five FIR filters. Thus, there will be 11 points being shifted cycle by cycle in the interpolation buffer. We use the same way to interpolate the vertical half pixels. After 7 cycles, all the half pixels we need are already ready, then we can choose two rows from them, each contains a 4x1 row half pixels, and use the two candidates to interpolate the quarter pixels according to the mvd_IBLs.



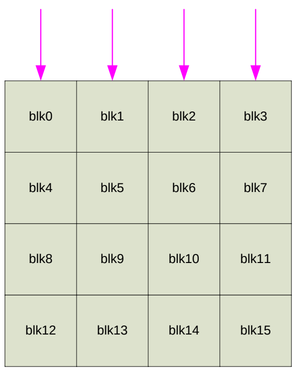


Fig. 32 (a) 4x4 block PU, a PE calculates the difference between reference and Current MB data of one pixel. (b) IBL processing order of 4x4 blocks, same rows would be process in the same time

The architecture of PU is shown in Fig. 32(a). It contains four processing elements (PE), 2-D Hadamard transform decomposed by two 1-D Hadamard transform and a transpose register array, which can continually process four pixels in each cycle without any latency.

To speed up, we use four 4x4 interpolation units to produce four 4x1 row data, each row for different 4x4 blocks. The order of processing for different 4x4 blocks is illustrated in Fig. 32(b). We first interpolate the fractional pixels of the first row of block 0, block 1, block 2 and block 3 simultaneously. While PUs calculating the SATD of block 0, block 1, block 2 and block 3, the interpolation unit keeps interpolating the fractional pixels of block 4, block 5, block 6 and block 7. In this way, SATD buffer can add only four times and get the cost of IBL and IBLR.

To further reduce the complexity of hardware design, we simplify the mechanism of IBL when the reference data cannot be found in the local memory. Instead of fetching memory from external memory, we skip IBL mode when some specific conditions meet. The conditions are as follows:

$$
\begin{aligned}
&(((mvdx\_IBL[0] - MVPx\_INTER)<=threshold) \quad \&\& \\
&((mvdx\_IBL[1] - MVPx\_INTER)<=threshold) \quad \&\& \\
&\qquad\qquad\vdots \\
&((mvdx\_IBL[15] - MVPx\_INTER)<=threshold) \quad \&\& \\
&((mvdy\_IBL[0] - MVPy\_INTER)<=threshold) \quad \&\& \\
&((mvdy\_IBL[1] - MVPy\_INTER)<=threshold) \quad \&\& \\
&\qquad\qquad\vdots \\
&((mvdy\_IBL[15] - MVPy\_INTER)<=threshold) \quad) = 1
\end{aligned}
\tag{4.3}
$$

Where mvdx[i] for i = 0~15 is the vertical elements of mvd_IBLs and mvdy[i] for i = 0~15 is the horizontal elements of mvd_IBLs.

Therefore, all the elements of the differences between mvd_IBLs and MVP_INTER should be below threshold, otherwise, we skip IBL mode. Thus, in our design, we check if the conditions mentioned above hold. If the conditions hold, we set the final cost of IBL extremely large so it won’t be chosen.

Fig. 33 shows the proposed overall architecture design for IME. In level 0, we have INTER, ILR, ILM, and ILMR mode, and each one has RDcosts of 4 different block type. With the IBL and IBLR mode, there will be 18 modes from level 0 in IME. We first do the mode pre-selection of LI’s to eliminate some modes from the 18 modes. And then, we directly reduce the number of modes to 3 by comparing the RDcosts of the modes left. Afterwards, we use our proposed fast multi-level mode filtering algorithm to select 3 best modes from different levels. By the whole mode filtering process, there will be only 3 modes as the IME output in the end of IME stage.

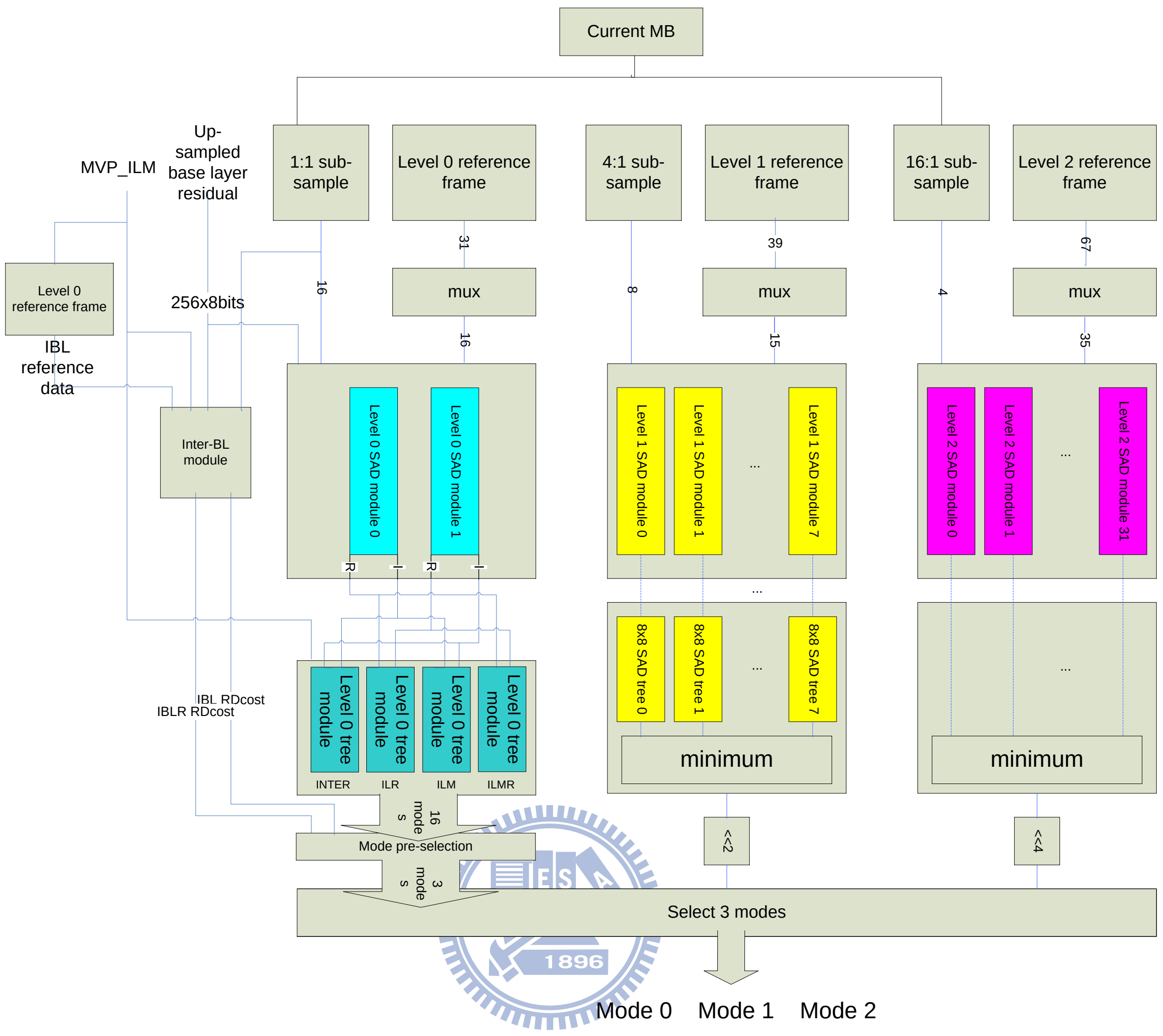


Fig. 33 the proposed PMRME architecture with inter-layer prediction

4.2.3. Search scheduling of IME

The search scheduling is adapted from the previous work [1]. Fig. 34 shows the search flow in level 0. For the current MB, we separate it into 16 row packages; while the reference data are cut into many overlapped row packages (17x31, 17 is because of the search range [-8, 7] and we search 2 positions in the same time, 0~15 for the first position and 1~16 for second position. 31 comes from the data of the search range need, that is 16+8+7=31.). The SRAM will be discussed in 4.5.

Fig. 36 shows the pipelined search schedule of level 0. The pipeline only takes 143 cycles to read all row packages needed during the level 0 searching. In Fig, C0~C15 represent the sixteen 16-pixels current MB rows. R(-8,-8)~R(7,7) indicate the 17-pixels rows from the SRAM according to the search position. Different from previous work, we search two positions in the same time, thus, the R(-8,-8) row package also includes the row of (-7,-8) position; that is, the row package of R(-8,-8) contain the 16-pixels row packages of R'(-8,-8) and R'(-7,-8), where R'(X,Y) represents a 16-pixels row package of position (X,Y). The row package is shared as illustrated in Fig. 35.

In the first cycle, we will get the SAD of [C0, R'(-8,-8)] by "level 0 SAD module 0" and the SAD of [C0, R'(-7,-8)] by "level 0 SAD module 1". In the following cycle, we will get the SAD of [C0, R'(-8,-7)] and {[C1, R'(-8,-7)]+ [C0, R'(-8,-8)]} by "level 0 SAD module 0", [C0, R'(-7,-7)] and { [C1, R'(-7,-7)] + [C0, R'(-7,-8)]} by "level 0 SAD module 1" , and so forth. Until the 4th cycle, we can have the SADs of 4x4_00 block, 4x4_01 block, 4x4_02 block, and 4x4_03 block as mentioned in the 4x4 SAD tree in Fig. 26 of search position (-8,-8) and (-7,-8). Moreover, since the search is fully pipelined, we can acquire result of every search point in every cycle after the 4th cycle.

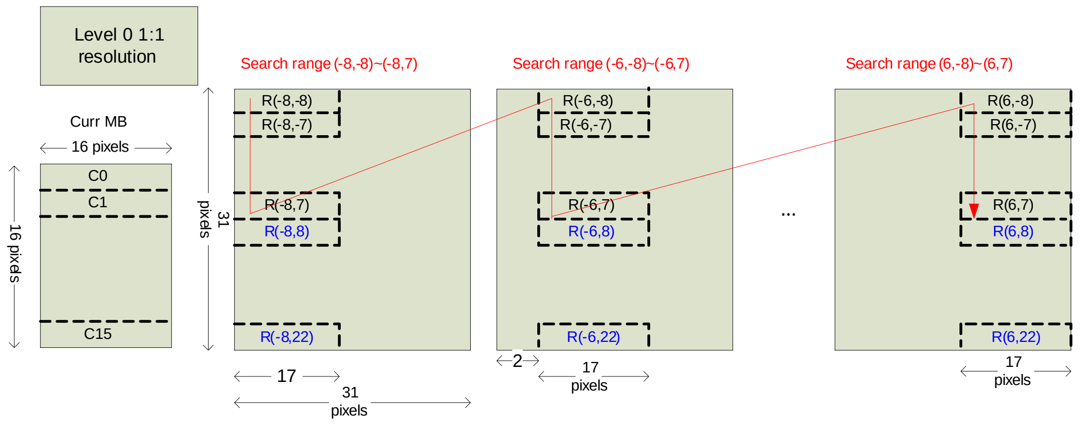


Fig. 34 the reference control of level 0

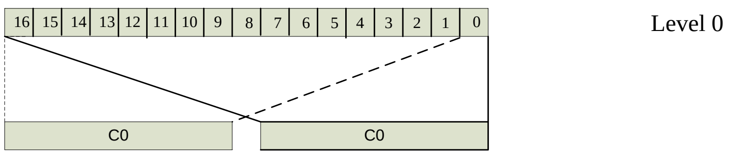


Fig. 35 parallel data reuse in level 0. A row package is shared between 2 search points

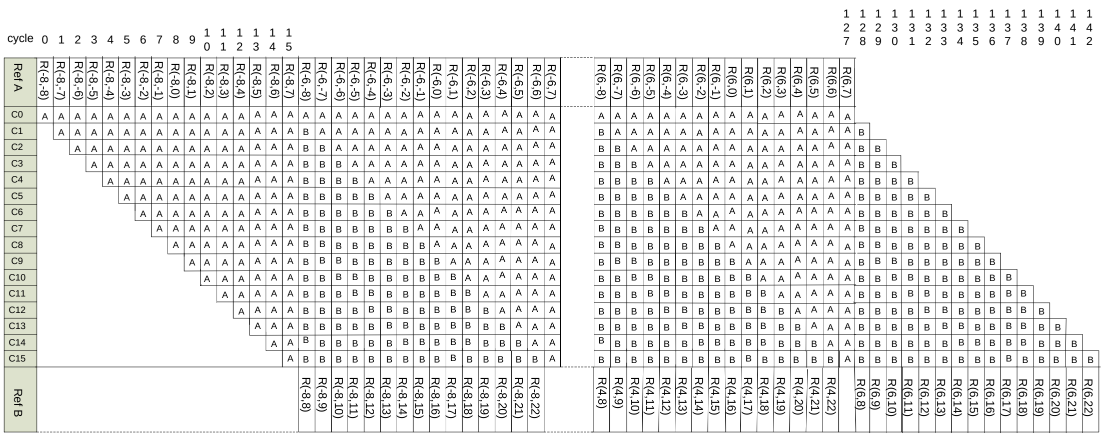


Fig. 36 the pipelined search schedule oh level 0

Fig. 37 shows the search scheduling of level 1. Since level 1 has quite larger search range and, it's adapted from previous work [1] to accelerate the processing. We speed up the level 1 searching process by the parallel calculations of different search positions like we mentioned before. Instead of searching 2 positions at a time, we here search 8 positions in parallel in level 1. For example, [C0,R(-32,-32)] are capable of dealing with the partial SAD of search points (-32,-32), (-30,-32), (-28,-32), (-26,-32), (-24,-32), (-22,-32), (-20,-32), and (-18,-32) at the same time, as illustrated inFig. 38. Fig. 39 shows the pipelined search schedule of level 1. The whole pipeline can be done in 131 cycles

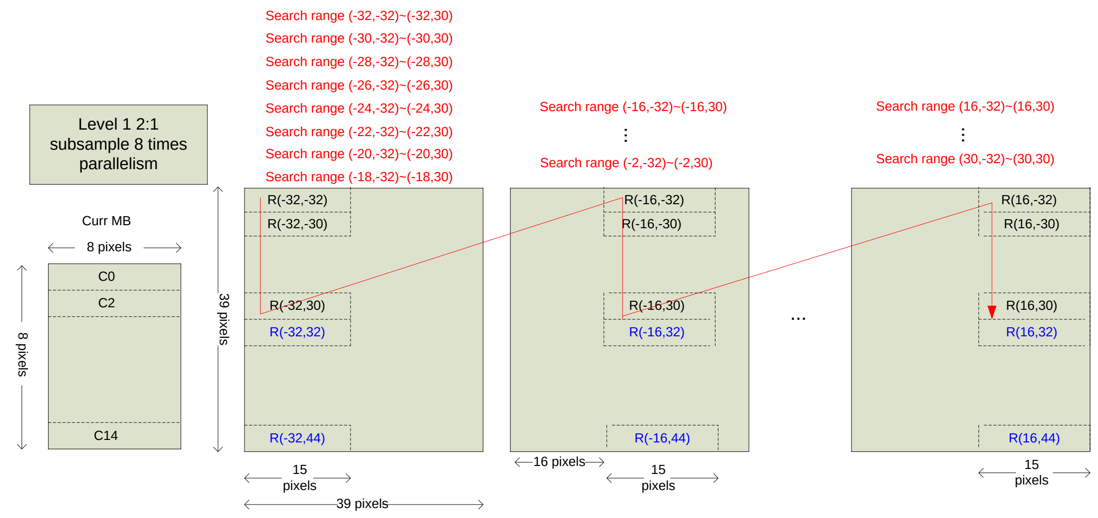

Search range (-32,-32)~(-32,30)
Search range (-30,-32)~(-30,30)
Search range (-28,-32)~(-28,30)
Search range (-26,-32)~(-26,30)
Search range (-24,-32)~(-24,30)
Search range (-22,-32)~(-22,30)
Search range (-20,-32)~(-20,30)
Search range (-18,-32)~(-18,30)
Search range (-16,-32)~(-16,30)
Search range (-2,-32)~(-2,30)
Search range (16,-32)~(16,30)
Search range (30,-32)~(30,30)
Level 1 2:1 subsample 8 times parallelism
Curr MB
8 pixels
C0
C2
C14
39 pixels
R(-32,-32)
R(-32,-30)
R(-32,30)
R(-32,32)
R(-32,44)
R(-16,-32)
R(-16,-30)
R(-16,30)
R(-16,32)
R(-16,44)
R(16,-32)
R(16,-30)
R(16,30)
R(16,32)
R(16,44)
15 pixels
16 pixels
…

Fig. 37 the search flow of level 1

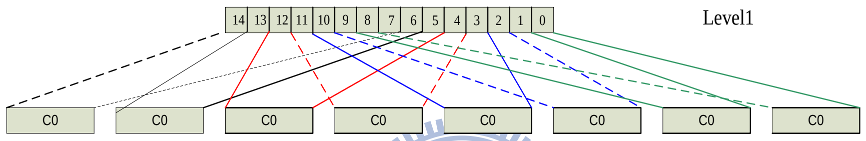


Fig. 38 parallel data reuse in level 1

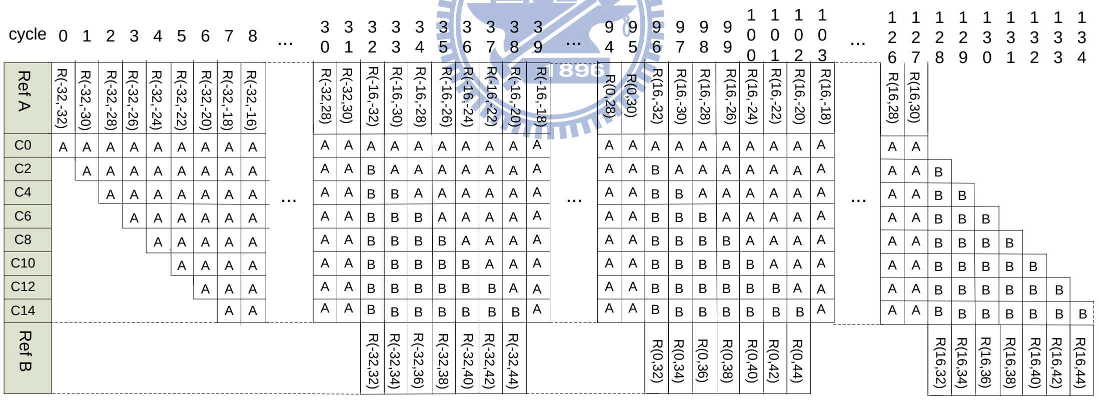
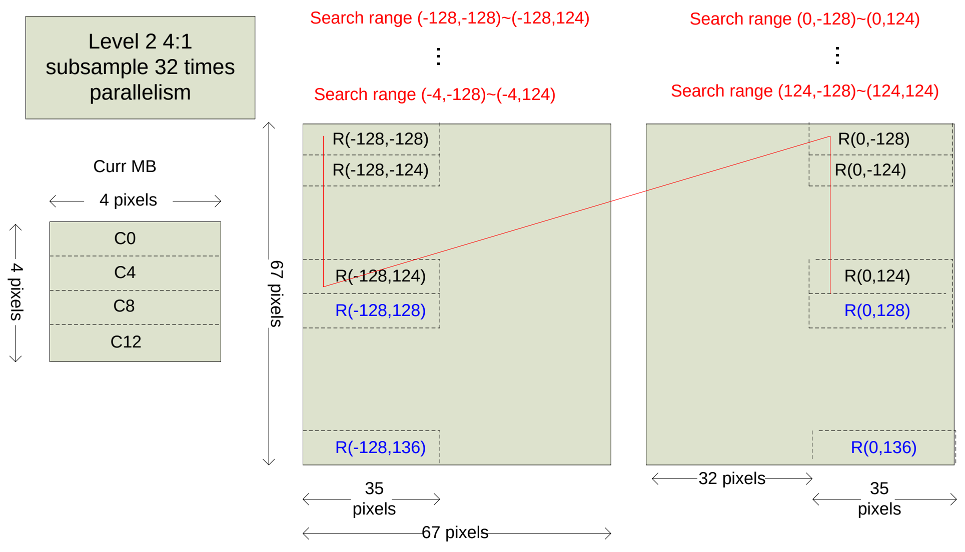


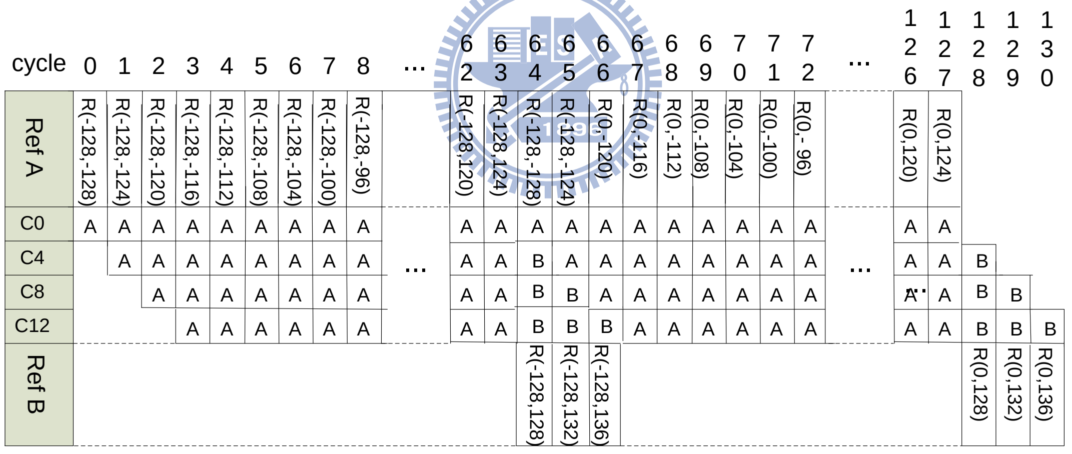
Fig. 39 the reference control of level 1

As for level 2, the pipelined search schedule and search flow is quite like level 1. However, the search range for level 2 is even larger, thus, we have to use 32 times parallelism to speed up the searching. As a result, we set the row package of level2 as a 35 pixels (4+31) row so that every row package can acquire 32 SAD from different search points. The search scheduling of level 2 is shown in Fig. 40 and the pipelined search schedule is shown in Fig. 41. In this degree of parallelism, it only takes 134 cycles to finish the pipeline of level 2.



Level 2 4:1 subsample 32 times parallelism
Search range (-128,-128)~(-128,124)
Search range (-4,-128)~(-4,124)
Search range (0,-128)~(0,124)
Search range (124,-128)~(124,124)
Curr MB
4 pixels
C0
C4
C8
C12
67 pixels
R(-128,-128)
R(-128,-124)
R(-128,124)
R(-128,128)
R(-128,136)
R(0,-128)
R(0,-124)
R(0,124)
R(0,128)
R(0,136)
35 pixels
67 pixels
32 pixels
35 pixels

Fig. 40 the search flow of level 2



cycle
0 1 2 3 4 5 6 7 8 … 62 63 64 65 66 67 68 69 70 71 72 … 126 127 128 129 130
Ref A
R(-128,-128) R(-128,-124) R(-128,-120) R(-128,-116) R(-128,-112) R(-128,-108) R(-128,-104) R(-128,-100) R(-128,-96)
R(-128,120) R(-128,124) R(-128,-128) R(-128,-124) R(0,-120) R(0,-116) R(0,-112) R(0,-108) R(0,-104) R(0,-100) R(0,-96)
R(0,120) R(0,124)
C0
C4
C8
C12
A
B
…
Ref B
R(-128,128) R(-128,132) R(-128,136)
R(0,128) R(0,132) R(0,136)

Fig. 41 the reference control of level 2

Thanks to the parallelism we use in all levels, the pipelined search schedules of all three levels can finish the in 142 cycles. The corresponding SRAMs of level 0, level 1, and level 2 will be discussed in 4.5.

4.3. Architecture design of FME

In this chapter, we propose a new FME architecture design according to the fast algorithm of SPFME [5] as mentioned in previous chapter. Since the only difference between SPFME and our early fast algorithm is that SPFME searches four more position around the zero (0, 0) position in 1/4-pel unit, the architecture design of SPFME could be similar to the one of SIFME. Furthermore, we had proposed an architecture design for SIFME before. Thus, the architecture of SPFME can be easily implemented from the design of SIFME.

In the following section, the overview of FME architecture design for SIFME would be introduced in 4.3.1. In 4.3.2, the proposed SPFME architecture design would be discussed in detailed.

4.3.1. Overview of previous FME architecture design

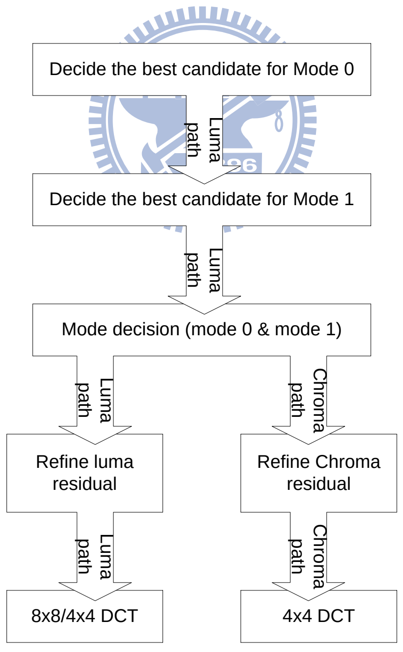


Fig. 42 function flow of previous FME stage

Fig. 42 shows our previous FME function flow. It is based on our previous fast algorithm for FME, which is the SIFME as mentioned before. It can be divided into two paths: luma path and chroma path. The luma path includes three parts: candidate decision, mode decision and luma residual generation. Since the previous version of our IME would pass two best modes to FME after the mode filtering technique, FME only has to calculates two modes, mode0 and mode1 in the luma path. The chroma path only includes the refinement process to generate chroma residuals.

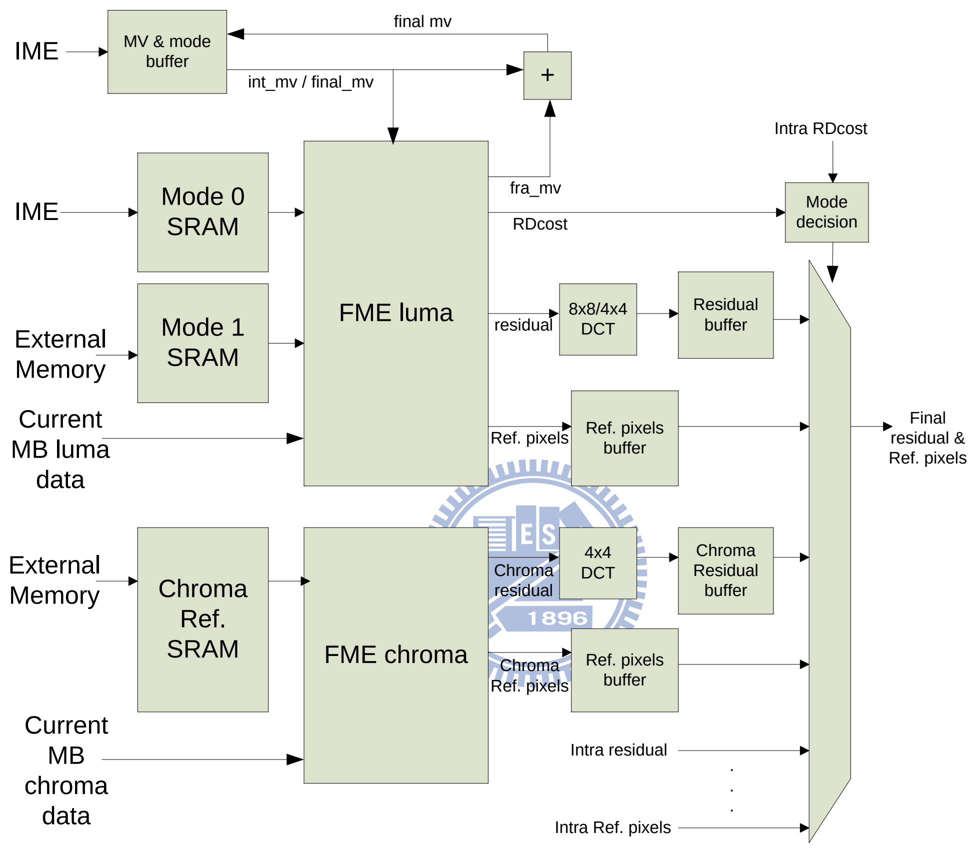


Fig. 43 the previous FME architecture design

Fig. 43 shows the previous architecture design of FME. The luma path consists of Mode 0 Reference SRAM, Mode 1 Reference SRAM, the “FME luma” module and the “8x8/4x4 DCT” module. The chroma path includes the Chroma Reference SRAM, the “FME chroma” module and the “4x4 DCT” module.

The luma path contains two reference SRAM. This is because of the previous subsample strategy from IME stage. In early IME process, there would be two modes passed to FME after the mode filtering process. In the mode filtering process, the first mode must be the best mode from level 0. The second mode is selected from the second mode of level 0, the level 1 mode and level 2 mode. Thus, one SRAM stores luma reference pixels coming from IME stage and the other holds pixels from external memory.

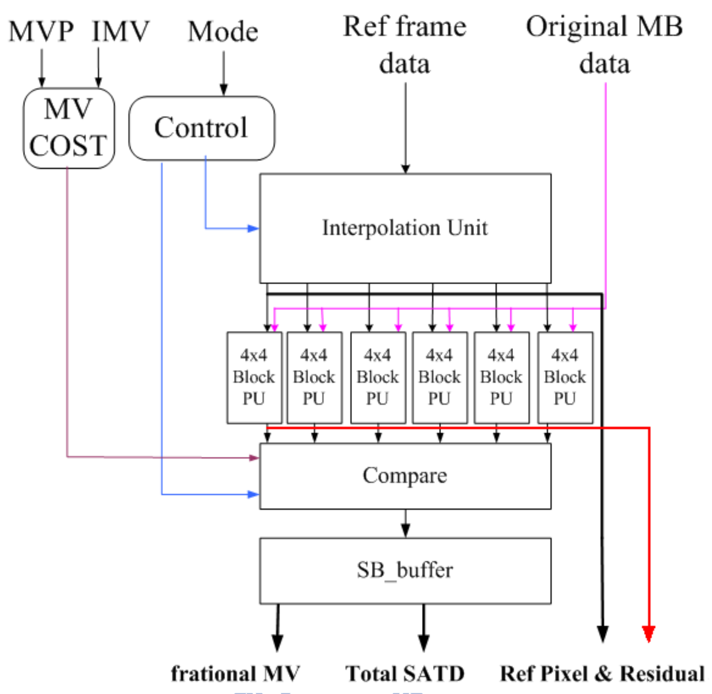


Fig. 44 the architecture of FME luma module

Fig. 44 shows the architecture of “FME luma” module. The “FME luma” module includes “Interpolation unit” module, “PU” module, “Compare” module, “SB_buffer” module, ”MV COST” module, and “Control” module. The “Interpolation unit” is responsible for interpolate fractional pixels, including 1/2 pixels and 1/4 pixels.

The “FME luma” module has six PUs due to the six search positions in SIFME. Each of them is capable of processing a 4x4 block SATD. The SATDs of all six candidates are sent to the “Compare” module and each of them would be added by its rate (MVcost) to become RDcost. The “Compare” module compares all six RDcosts and decides the best candidate. After the best candidate of each mode is chosen, “SB_buffer” module would compare the RDcost of the two modes and decides the best one to send out. In the end, the refinement step is carried out to get the final residual according to the outcome of “SB_buffer” module.

The overall flow of FME process is as follows: First, the “MV buffer” buffers the motion vectors from IME and pass them to “FME luma” module in the same time. “FME luma“ module calculates SATD of six candidates and selects the best one of all blocktypes of a mode. After RDcosts of two modes are generated, “FME luma” would decide the best mode to recalculate residual and reference pixels of the best candidates of the best mode. The residual is then passed to “8x8/4x4 DCT” module to do the DCT transform. In the same time, we load reference chroma data to “Chroma Ref. SRAM” according to the best motion vectors from luma path and calculate chroma residual and chroma interpolated reference pixels in the “FME chroma” module.

In the end of FME stage, the RDcost of FME is compared with the RDcost of intra mode in “mode decision” module to decide whether current macro block is coded in INTRA mode or INTER mode.

4.3.2. Proposed FME design

4.3.2.1. Overall architecture and primary modules

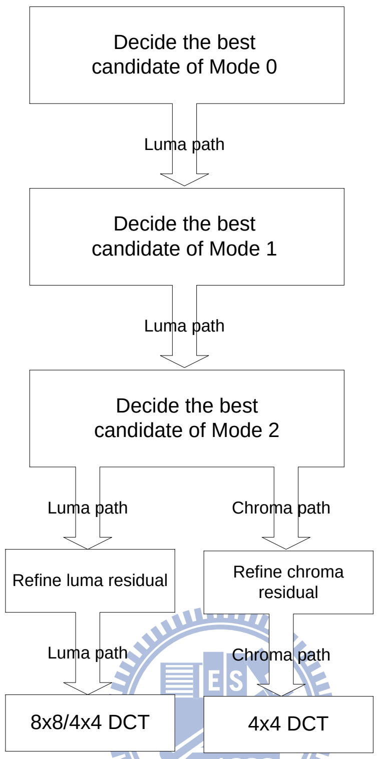


Fig. 45 the new FME function flow

The new flow is shown in Fig. 45 . The differences between the new flow the previous one are that the luma path of new flow processes 3 modes and each mode could be in inter-layer prediction mode. As for the chroma path, the function flow is the same as the previous one.

The proposed FME design has some different characteristics from previous one. First, there are four more points to search than the SIFME algorithm. Thus, we have to add more "PU" modules to process these ten points in parallel. Moreover, since we adopt IL prediction in our IME stage, we have to consider features of IL mode in our FME design, including inter-layer residual mode (ILR), inter-layer motion mode (ILM), inter-layer motion residual mode (ILMR), and inter-BL mode (IBL).

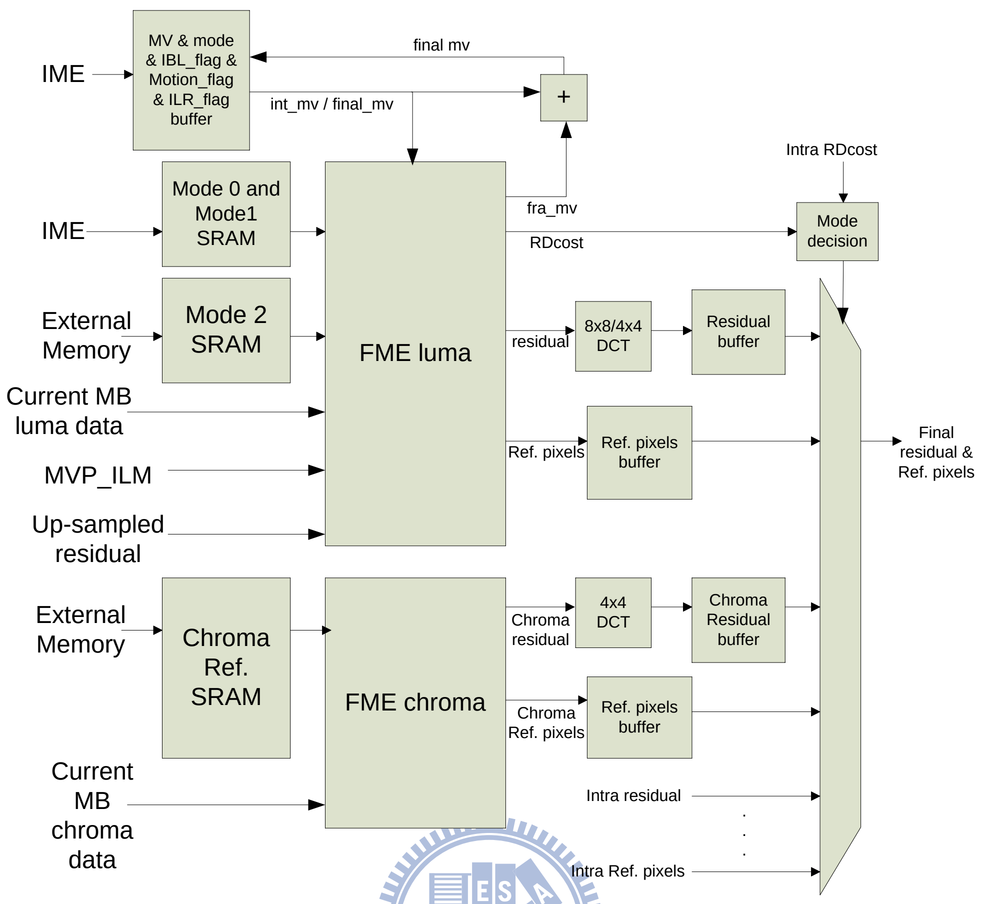


Fig. 46 the architecture of proposed FME architecture. To process IL mode from IME, we additionally input MVP_ILMs and up-sampled residual into FME luma module, motion_flag and IBL_flag into the MV buffer

Fig. 46 showed the proposed architecture. The difference between the original one and the new one are few. However, to process the IL mode, we input MVP_ILMs and up-sampled residual into the new "FME luma" module to calculate RDcosts of ILM mode, ILR mode, and ILMR mode. Moreover, the MV buffer now buffers motion vectors of three modes from IME as well as their motion_flags, ILR flags, and IBL flags to indicate the prediction type of each mode.

There are two SRAM for FME architecture. One is for mode0 and mode1 since we choose the best two modes from IME in level 0. The third mode is selected from the third mode in level 0, level 1 mode, and level 2 mode to as mentioned in chapter 3. Thus, the "mode 0 and mode 1 SRAM" load pixels from IME stage and the "mode 2

SRAM" holds pixels from external memory.

4.3.2.2. FME luma module

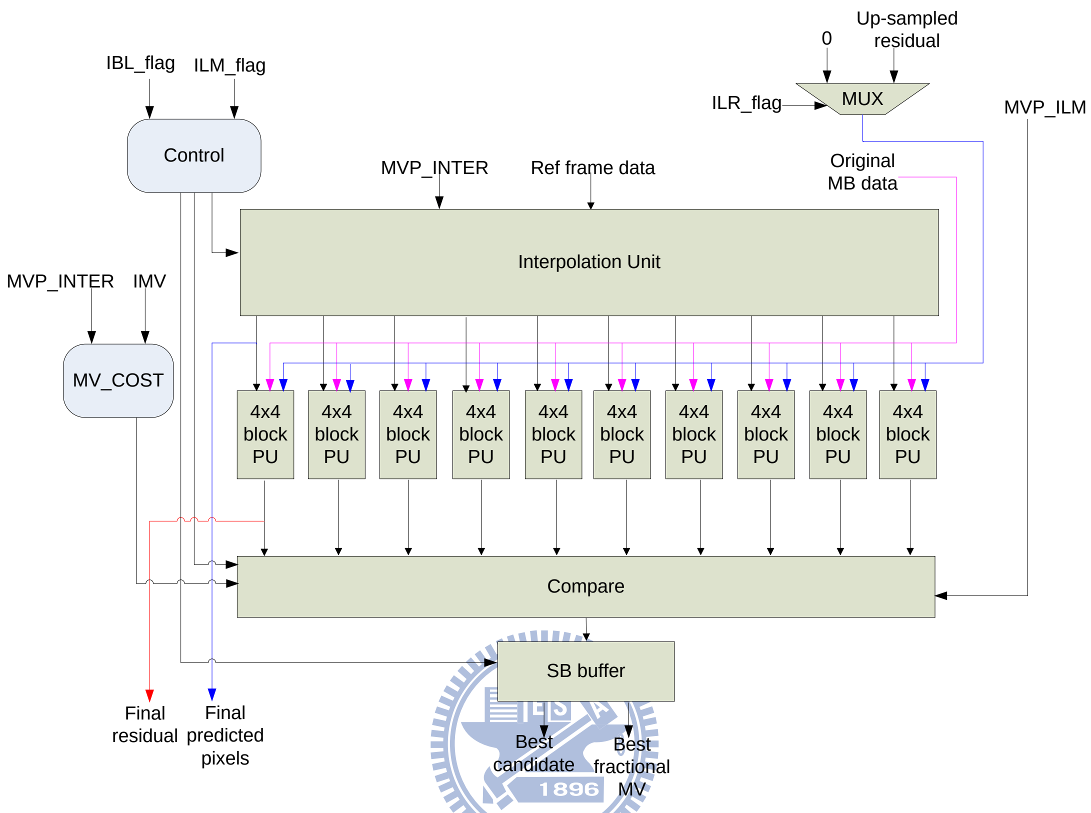


Fig. 47 the proposed FME luma hardware

The SPFME is implemented in our proposed "FME luma" module. Fig. 47 shows the architecture of the "FME luma" module of SPFME. We add up-sampled residual, MVP_ILM, "ILR_flag", "ILM_flag", and "IBL_flag" as inputs of "FME luma" module to process different modes from IME. Moreover, we add four more PUs to respectively calculate the up left, up right, down left and down right position around zero (0, 0). These ten PUs will calculate the SATDs of different position, and each SATD will be combined with its own MVcost to get the RDcosts in the "Compare" module.

The up-sampled residual is input of the PU module through a multiplexer. The "ILR_flag" indicates that whether the processing mode is ILR mode or not. If the "ILR_flag" is 1, the processing mode is ILR mode, otherwise, the processing mode is

INTER mode. Thus, the multiplexer will output up-sampled residual to PUs if "ILR_flag" is 1. Otherwise, the multiplexer will output zero so that the SATD of PU equals the SATD of INTER mode.

The MVP_ILM and MVP_ILM are used to calculate the RDcost when the present mode is inter-layer motion (ILM) or inter-layer motion residual (ILMR) mode. The "Control" module will decide whether the ILM mode should be processed. If ILM_flag equals 1, the "Compare" module will combine the SATD as well as MVcost derived from MVP_ILM and fractional mvd to get the RDcost. Otherwise, the MVcost will be derived from MVP_INTER and fractional mvd so that we will have different RDcost.

4.3.2.3. The parallel processing architecture of interpolation unit

Our interpolation unit is adapted from previous work [3]. However, in order to meet our spec, we have to further increase the speed of interpolation process. Thus, we need to make interpolation unit capable of processing four times the data than before.

Fig. 48 shows the architecture of our interpolation unit. Since we add more buffers in the interpolation unit, the interpolation unit now can buffer four 10-pixel rows data every cycle. First, we interpolate the horizontal half pixels of each row by five 1-D FIR filters from 10 adjacent integer points. Thus, after the first cycle, there will be four integer pixel rows and four horizontal half pixel rows in the buffers. Until all the buffers are full, we can decide the two candidates to interpolate the quarter pixels according to *pred frac mv* and the zero (0, 0) position through bilinear filters.

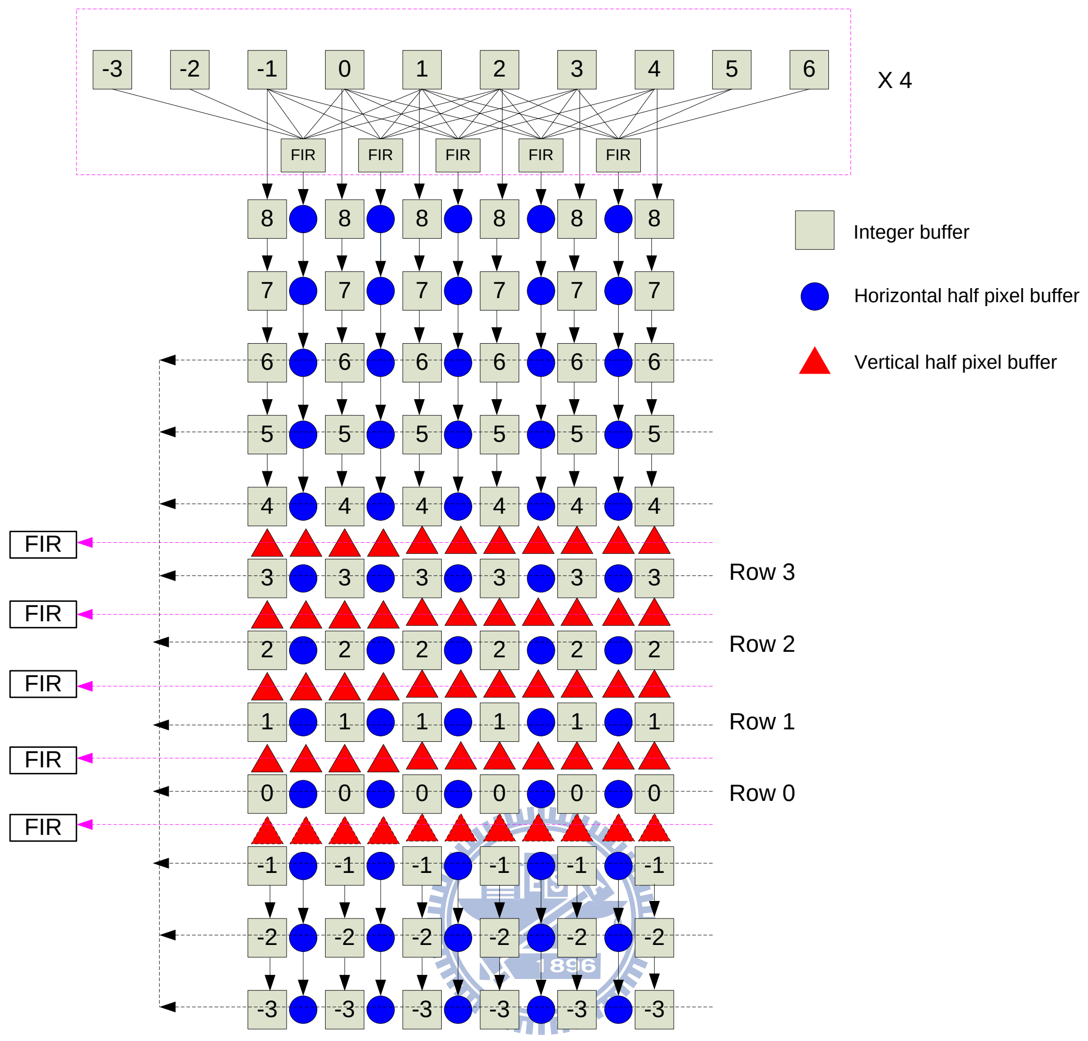


Fig. 48 the interpolation unit architecture

To interpolate four rows pixels cycle by cycle, we have five 1-D vertical FIR filters in every column to calculate the vertical half pixels. Therefore, there are total 5x4(row)=20 horizontal FIR filters and 5x11(column)=55 vertical FIR filters in the interpolation unit.

With the pipeline, our interpolation unit can interpolate four rows data cycle by cycle. Since the interpolation is designed in 4x4 block unit, the cycle time to interpolate a block data is associated with the height of block. For example, a 4x4 block need (10/4) + 1=3 cycles for interpolation because it needs 3 cycles to buffer all the 10 rows pixels, which are a 4x4 block interpolation needs. The cycle time for interpolation of other blocktype is listed in Table 1.

Table 1 cycles for interpolation of different block types

Blocktype	16x16	16x8	8x16	8x8	8x4	4x8	4x4
Cycles	6x4=24	4x4x2=32	6x2x2=24	4x2=8	3x2=6	4	3

4.3.2.4. The skip IBL mode for FME

When one of the processing modes is IBL or IBLR (inter-BL residual) mode, we skip the processing of it because of the fact that the IBL mode takes the MVP_ILMs as mvd_IBLs. And the MVP_ILMs are in 1/4 unit already. Thus, IBL mode doesn’t have to go through the FME process again.

However, the original design sets the number of modes in FME process to be a constant value two. Hence, we re-design the “Control” module to process uncertain number of modes. Fig. 49(a) shows the proposed function flow of FME without IBL or IBLR mode and proposed function flow of FME with IBL or IBLR mode is shown in Fig. 49(b). In Fig. 49(b),“Fst_IBL”, “Sec_IBL”, and “Thd_IBL” represent IBL flag of the first mode, the second mode, and the third mode from IME, respectively. There could be many situations of the processing number of modes, 1 mode at least and 3 modes at most since that there are at most two inter-BL mode, which are IBL and IBLR.

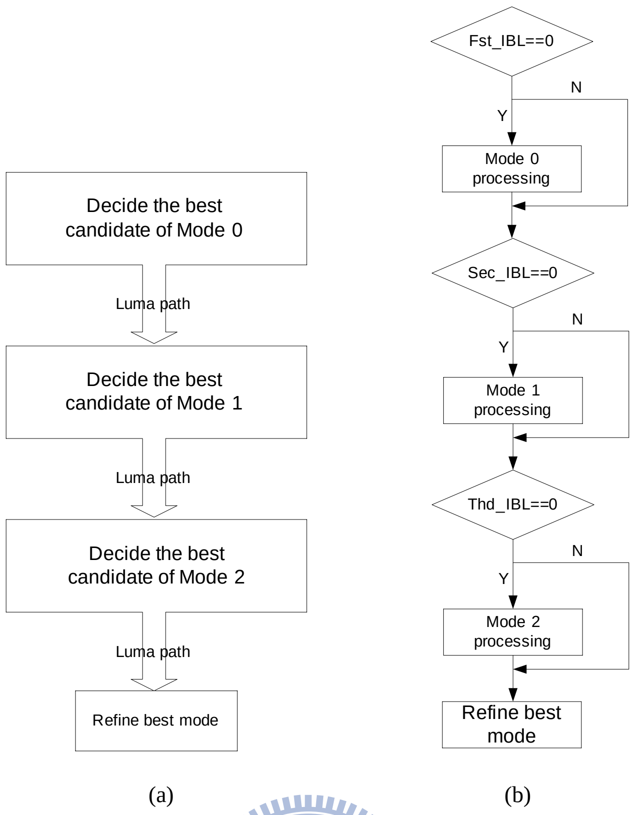


Fig. 49 (a) the new flow of FME luma path without IBL mode (b) the new flow of FME luma path with IBL or IBLR mode

Though we skip FME when the mode is IBL or IBLR, we still need their information in the refine stage. Thus, if the RDcost of IBL mode or ILR mode is the smallest, we directly take the MVP_ILMs as the final MVs and do the refine to get residual and predicted pixels.

4.4. Reference SRAMs

4.4.1. Level 0 and FME SRAM

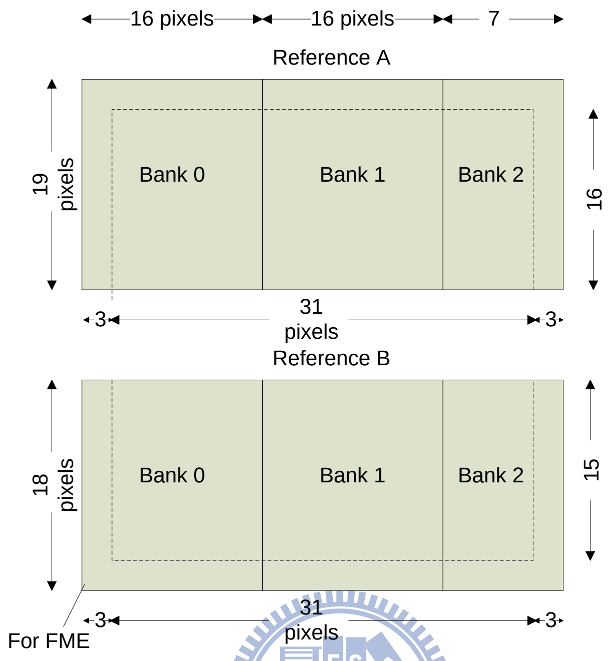
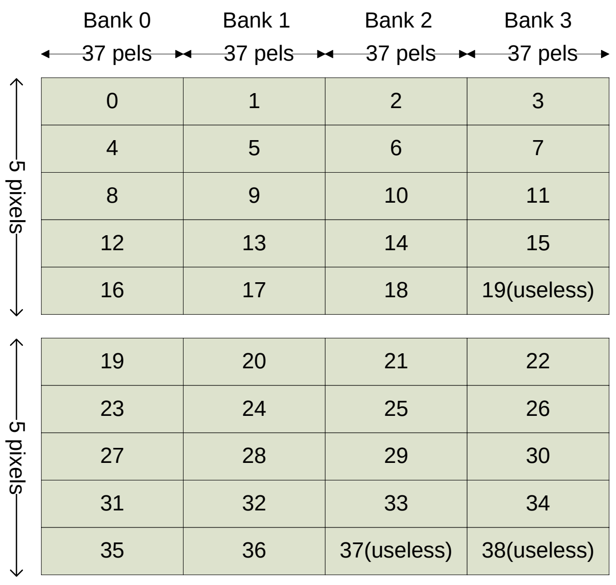


Fig. 50 previous IME level 0 SRAM and its bank

The reference SRAM of level 0 in IME is the same as the SRAM of FME. To meet the needs of processing four rows of FME at a time, we adapt the reference SRAM from previous work [1]. Fig. 50 shows the previous reference SRAM for level 0. It is cut into two parts, reference A and reference B, and each part contains three banks. The reason that the SRAM is cut into two parts is to realize fully pipelined data flow for motion estimation as illustrated in Fig. 36. Each part has width of 37(pixels) x 8(bits) = 296 (bits). Reference A has height of 19 words and reference B has height of 18 words. Thus, the whole memory size is 37x37=1369 bytes.



0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19(useless)

19	20	21	22
23	24	25	26
27	28	29	30
31	32	33	34
35	36	37(useless)	38(useless)

Fig. 51 the proposed SRAM for FME and level 0 in IME

To meet the needs of FME, we divide the original SRAM into four banks. We divide the SRAM according to their address. The way we divide banks is as follows:

Bank 0= { {i modulo 4=0, i=0~18},{ (i-19) modulo 4=0, i=19~36} };

Bank 1= { {i modulo 4=1, i=0~18},{ (i-19) modulo 4=1, i=19~36} };

Bank 2= { {i modulo 4=2, i=0~18},{ (i-19) modulo 4=2, i=19~36} };

Bank 3= { {i modulo 4=3, i=0~18},{ (i-19) modulo 4=3, i=19~36} };

where i refers to the index of word in the SRAM. When i=19, we write the row to both reference A and reference B so that the pipeline can be achieved. In this way, we can load successive four rows data from the SRAM without confliction.

4.4.2. SRAMs of level 1 and level2

The SRAM of level 1 and level 2 are the same as the previous work [1], as illustrated in Fig. 52(a) and Fig. 52(b). In Fig. 52(a), we can see that level 1 SRAM has

39 pixels in height and 40 pixels in width. The reason for the width to be 40 pixels is that it can be divided into 5 banks and each of them has 8 pixels in width. Moreover, 8 pixels is the width of a macro block for level 1, thus, we only have to refresh one bank to process the next macro block and the other four banks can be reused. Level 2 SRAM has 67 pixels in height and 68 pixels in width with 17 banks. The reason for the width to be 68 pixels is the same as level 1. Alike level 1, we only have to update one bank for the searching of every macro block.

The only difference from the previous one is that we access longer row package from both the SRAM of level 1 and level 2. For level 1 we access 15 (8+7) pixels at a time to do 8 times parallelism of search. For level 2, we load 35 (4+31) pixel to search 32 positions at the same time.

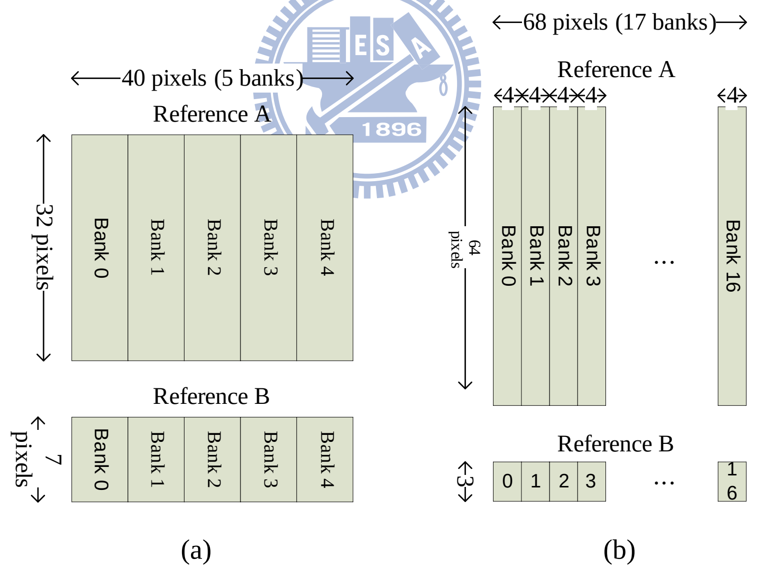


Fig. 52 (a) the SRAM of level 1 (b) the SRAM of level 2

4.5. Memory schedule

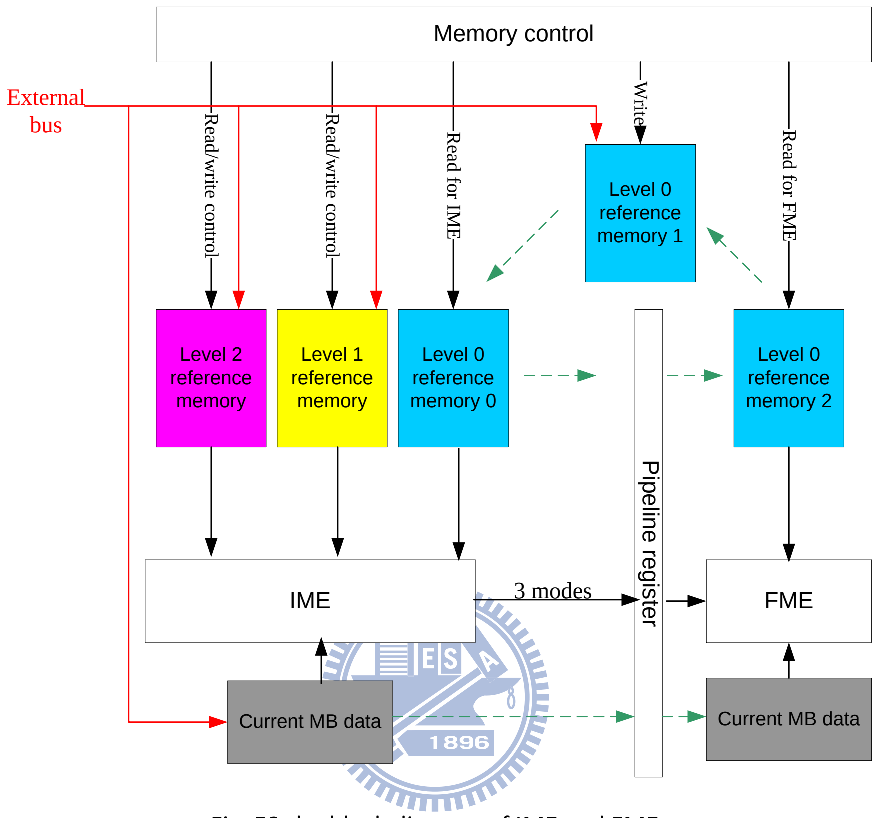


Fig. 53 the block diagram of IME and FME

The memory schedule of IME and FME is the same as the previous work [1], as illustrated in Fig. 53. The difference is that the level 0 SRAM is changed into more banks as mentioned in previous section.

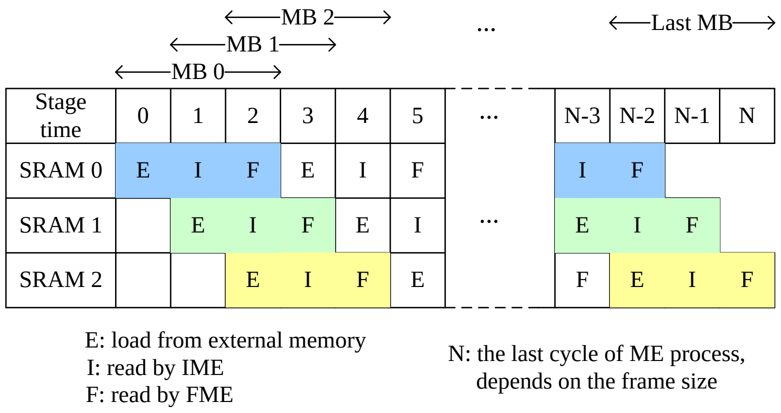


Fig. 54 the ping-pong buffer concept of level 0 SRAMs. Stage time here is 450 cycles.

There are 3 identical level0 SRAMs to enable the ping-pong buffer concept. The change of memory state is shown in Fig. 54. When the first MB finishes IME process, the SRAM for the first MB read by IME process is now changed to be read by FME process. In the same time, the original FME SRAM is now changed to load the third MB data from external memory. As for the SRAM which were loading the second MB data from external memory is now changed to read by IME process.

In case that the FME stage will have to load to many data from external memory and cause too much data loading traffic, the author of the previous work set the first mode from IME to be inside the search range centered on MVP_INTER, while the other mode might be outside the search range. By our mode filtering algorithm and acceleration of FME hardware, although we now have 3 modes from the IME stage, the external memory loading traffic can still be little. Moreover, the adoption of our efficient algorithm for inter-layer prediction can further prevent the loading traffic of external memory caused by inter-layer prediction.

Chapter 5. Simulation and Implementation results

5.1. Simulation results

In this section, we list several simulation results to demonstrate the performance of our whole mode filtering process. Since we adopt Li's algorithm in the C model of [4], we simply use our previous C model of [4] without mode filtering as the reference software. The simulation setting is summarize in Table 2

Table 2. simulation settings

Reference software	Ref[4]	
QP	18,28,38	
Frame size in spatial base layer	QCIF	
Frame size in spatial enhancement layer	CIF	
Frames to be encoded	150 for Table 3,100 for Table 4	
Frame rate	30	
Adaptive inter-layer prediction	ON	
Multi-resolution	Table 3:Off	Table 4:On
Search range size	±8	
GOP	8	
Reference frame number	2	
Test sequence	Akiyo, Coastguard, Football, Foreman, Mobile, News	

Table 3 shows performance of different selected number of candidates from IME after pre-selection scheme [7] without PMRME for CIF as EL and QCIF as BL. When choosing 3 modes from IME, the RD performance is 4.617% in bit-rate increase and 0.031 dB in PSNR degradation.

Table 3. mode selection performance after pre-selection algorithm without PMRME for CIF as EL and QCIF as BL

Sequence	QP		3 modes	4 modes	5modes	6modes
Akiyo	18	*PSNR(dB)*	-0.0872	-0.0695	-0.0713	-0.066
		Bit-rate(%)	4.222	3.778	3.556	2.889
	28	*PSNR(dB)*	-0.0457	-0.0225	-0.0144	-0.0113
		Bit-rate(%)	10.256	10.256	10.256	9.402
	38	*PSNR(dB)*	-0.0472	-0.0584	-0.0401	-0.0462
		Bit-rate(%)	18.421	15.789	15.789	15.789
Coastguard	18	*PSNR(dB)*	-0.0078	-0.003	-0.0023	-0.0057
		Bit-rate(%)	1.394	1.016	0.784	0.580
	28	*PSNR(dB)*	-0.0203	-0.0115	-0.0046	-0.003
		Bit-rate(%)	1.229	0.850	0.567	0.567
	38	*PSNR(dB)*	-0.0245	-0.0228	-0.019	-0.0166
		Bit-rate(%)	3.306	3.719	3.719	3.305
Football	18	*PSNR(dB)*	-0.0106	-0.0068	-0.0048	-0.0042
		Bit-rate(%)	0.391	0.335	0.307	0.279
	28	*PSNR(dB)*	-0.0486	-0.0452	-0.0439	-0.042
		Bit-rate(%)	0.493	0.352	0.352	0.352
	38	*PSNR(dB)*	-0.022	-0.0202	-0.0266	-0.0274
		Bit-rate(%)	1.727	1.727	1.535	1.535
Foreman	18	*PSNR(dB)*	-0.0202	-0.0084	0.0028	0.0064
		Bit-rate(%)	2.065	1.630	1.25	1.086
	28	*PSNR(dB)*	-0.0513	-0.0391	-0.0362	-0.028
		Bit-rate(%)	2.515	2.096	1.886	1.886
	38	*PSNR(dB)*	-0.0351	-0.0494	-0.0462	-0.0479
		Bit-rate(%)	10.666	10	9.333	9.333
Mobile	18	*PSNR(dB)*	-0.0232	-0.016	-0.0125	-0.0126
		Bit-rate(%)	2.049	1.627	1.085	0.844
	28	*PSNR(dB)*	-0.0295	-0.0188	-0.0076	-0.0037
		Bit-rate(%)	2.900	2.235	1.631	1.389
	38	*PSNR(dB)*	-0.0591	-0.0341	-0.0174	-0.0095
		Bit-rate(%)	3.611	3.333	2.777	2.777
News	18	*PSNR(dB)*	0.0265	0.0339	0.0312	0.0382
		Bit-rate(%)	2.358	1.768	1.650	1.415
	28	*PSNR(dB)*	-0.0615	-0.0433	-0.0364	-0.0396

		Bit-rate(%)	5.158	5.158	4.761	4.365
	38	*PSNR(dB)*	0.0123	0.0119	0.015	0.0009
		Bit-rate(%)	10.344	10.344	9.195	9.195
Average	18	*PSNR(dB)*	-0.020	-0.011	-0.009	-0.007
		Bit-rate(%)	2.080	1.692	1.438	1.182
	28	*PSNR(dB)*	-0.042	-0.030	-0.023	-0.021
		Bit-rate(%)	3.758	3.491	3.242	2.993
	38	*PSNR(dB)*	-0.029	-0.028	-0.022	-0.024
		Bit-rate(%)	8.012	7.485	7.058	6.989
	PSNR(dB)		-0.031	-0.023	-0.019	-0.018
	Bit-rate(%)		4.617	4.223	3.913	3.721

In Table 4, we list the RD performance of the final mode filtering scheme. The reference software is the previous work [4] with IL prediction and PMRME without mode filtering. The average PSNR degradation is 0.106dB and increase of bit-rate is 3.542%. In Fig. 55-Fig. 60 are the R-D curves of different sequences for 100 frames with QP=18, 28, 38.

Table 4 the RD performance of the final mode filtering with IL prediction and PMRME

QP		Akiyo	Coastguard	Football	Foreman	Mobile	News	Average
18	*PSNR(dB)*	-0.0609	-0.010	-0.098	-0.024	-0.014	-0.001	-0.034
	Bit-rate(%)	4.416	0.294	-1.454	2.075	2.662	2.280	1.712
28	*PSNR(dB)*	0.000	-0.019	-0.069	0.042	-0.036	-0.022	-0.017
	Bit-rate(%)	0.000	-0.679	-1.091	1.823	3.211	4.117	1.230
38	*PSNR(dB)*	-0.447	-0.063	-0.028	-0.322	-0.035	-0.701	-0.266
	Bit-rate(%)	22.222	-2.395	0.259	8.490	3.734	13.793	7.683
Ave.	*PSNR(dB)*	-0.106						
	Bit-rate(%)	3.542						

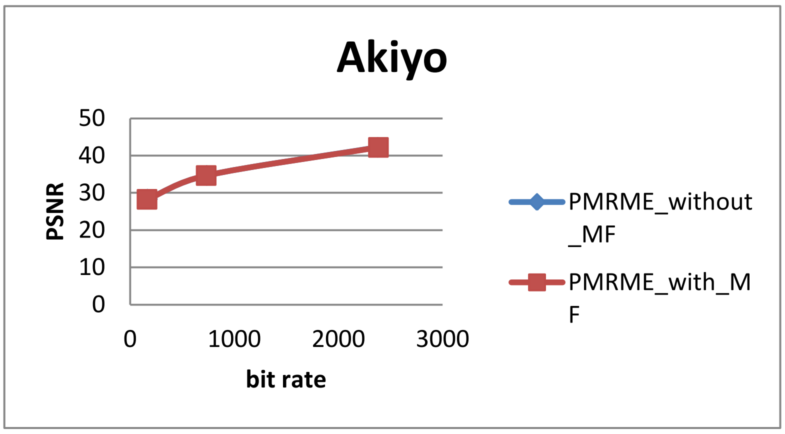


Fig. 55 the performance of our mode filtering for akiyo_cif

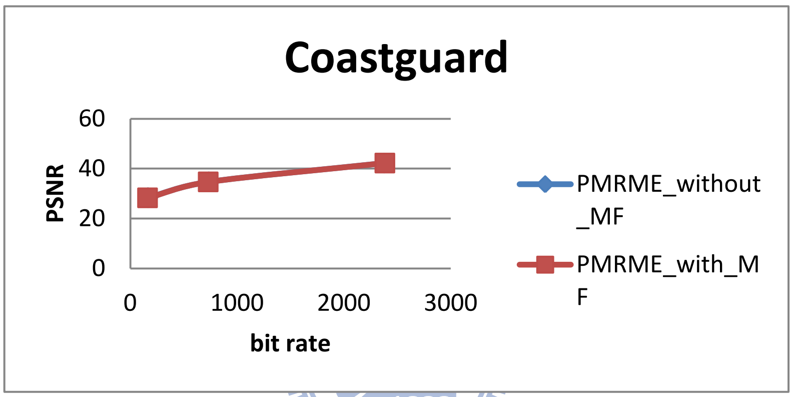


Fig. 56 the performance of our mode filtering for coastguard_cif

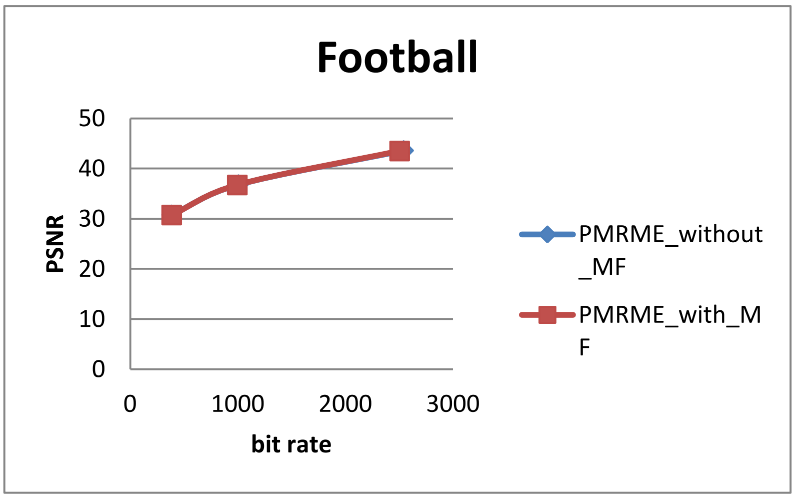


Fig. 57 the performance of our mode filtering for football_cif

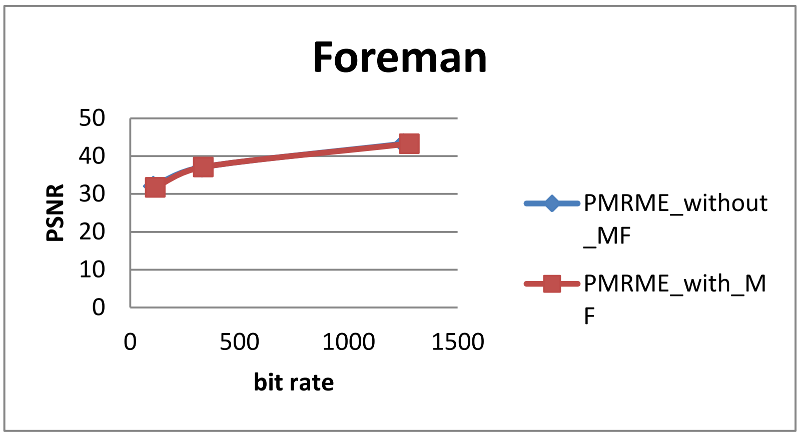


Fig. 58 the performance of our mode filtering for foreman_cif

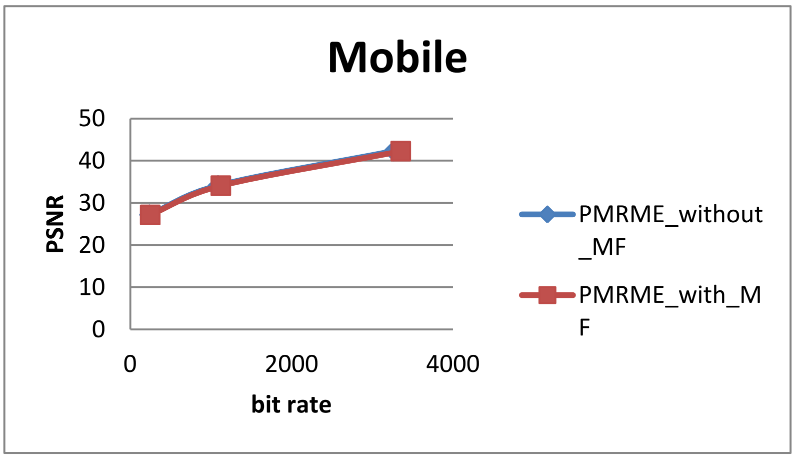


Fig. 59 the performance of our mode filtering for mobile_cif

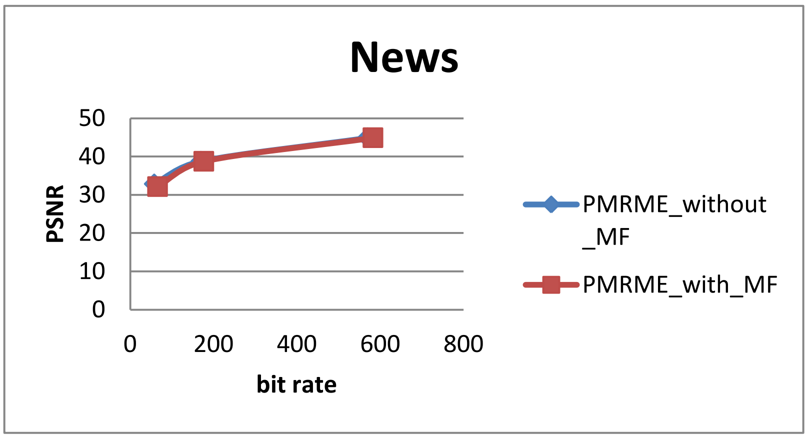


Fig. 60 the performance of our mode filtering for news_cif

5.2. Hardware implementation results

The proposed architecture is implemented by Verilog and synthesis in UMC 90nm technology at 142MHz.

Table 5. synthesis result of the PMRME in UMC90

Unit	Gate Count in 142 MHz
Level_0	166,085
Level_1	138,350
Level_2	112,558
Others	19,114
Total	436,107

Table 5 lists the synthesis result of IME in our design. It's almost double the area of the previous work [2] due to the adoption of IL prediction and the increase of parallelism for searching. Table 6 shows the comparison between different IME architecture. Our design includes the IL prediction and multi-resolution while the gate count is still acceptable. Moreover, our design can achieve CIF+480p+1080p@60fps since we have double IME hardware to process two MBs at the same time. The cycle time of our design is also quite short due to the adopted PMRME design and the parallel calculations of different search points.

The design of [13](a) and [13](b) both have small local SRAM. However, their areas are both extremely large. In comparison, although the design of [14] has larger search range, it also needs larger area cost and bigger local memory. Though the design of [15] has larger search range and smaller gate count, it only has one reference frame and the encoding block types only include block size over 8x8. Furthermore, the cycle times of [15] is relatively long compared with ours.

Table 6 the comparison of different IME architectures

	[13](a)	[13](b)	[14]	[15]	**Proposed**
technology	TSMC .18µm	TSMC .18µm	TSMC .18µm	TSMC .18µm	**UMC 90nm**
Max. Supporting Resolution	1080p@30fps	1080p@30fps	1080p@30fps	1080p@30fps	**CIF+480p+1080p@60fps**
# of reference frame	2	1	2	1	**2**
Search algorithm	Full search	Full search	Multi-resolution	Sub-sampling	**Multi-resolution + IL prediction**
Block sizes	all	all	16x16,16x8, 8x16, 8x8	16x16,16x8, 8x16, 8x8	**16x16,16x8, 8x16, submode**
IL prediction	N/A	N/A	N/A	N/A	√
Max Search Range	H:±64 V:±64	H:±128 V:±64	H:±256 V:±192	H:±192 V:±128	**H:±128 V:±128**
Gate count(K)	1449	1511	460x2	486	**436.1x2**
Local memory(KB)	2.97	1.61	96x2	40 (dual port)	**30.384**
Frequency(MHz)	130	130	200	200	**142**
Latancy(cycles)	N/A	N/A	756	960	**128~332(B-frame+IBL)**

The implementation results of FME_luma are listed in Table 7. It shows that the ten PUs occupy the largest area and the IE occupies second largest area due to the parallelism of 4 successive rows processing.

Table 7. synthesis result of the FME_luma module in UMC90

Unit	Gate Count in 142 MHz
Control	582
MV_COST	6,560
Interpolation unit(IE)	85,126
4x4 Block PU(*10)	102,560
Compare unit(COMP)	4,011
SB_buffer	10,687
Others	3,695
Total	213,221

The implementation results of FME top are listed in Table 8. It shows that the FME_luma occupies the largest area.

Table 8. synthesis result of the FME top module in UMC90

Unit	Gate Count in 142 MHz
FME_luma	21,3221
luma_ctrl	16,624
chroma	2,030
Chroma ctrl	1,700
MV_buf	10,507
MC_buf	15,788
8x8/4x4 DCT	13,309
DCT_buf	31,492
others	5,478
Total	310,194

Table 9 shows the comparison between different FME designs. Our FME design holds the best frame rate due to double hardware policy. It is also capable of dealing with two reference frames to enhance the encoding performance. While the gate count is smaller in the design of [16], it can only deal with block sizes over 8x8. Moreover, since [16] deal all the block sizes over 8x8, the latency will be fixed as 256 while ours is flexible. As for [17], its latency is too long compared with ours since we accelerate the FME by process 4 rows in parallel.

Table 9. comparison between different FME design

	[16]	[17]	**Proposed**
technology	TSMC .13µm	Chartard .18 µm standard CMOS1P5M	**UMC 90nm**
Max. Supporting Resolution	1080p@30fps , QFHD@24fps	1080p@30fps	**CIF+480p+1080p@60fps**
# of reference frame	1	N/A	**2**
Search algorithm	SPFME	Full search	**SPFME**
Max Search Range	H:±64 V:±64	H:±128 V:±64	**H:±128 V:±128**
Block sizes	16x16, 16x8, 8x16, 8x8	all	**all**
Gate count(K)	134	412	**310.2x2**
Local memory(KB)	N/A	9.1	**5.92**
Frequency(MHz)	250	200	**142**
Latancy(cycles/MB)	256	862	**Worst:384 /Best:96**

Table 10 lists the overall synthesis results of our ME design. The gate count is 847.3×2 (K) because of the double hardware policy we adopt.

Table 10. the overall synthesis result of ME

PMRME	436,107
FME_top	310,194
MEM_top	85,462
Curr_buf	15,585
Total	847,348
In design	847,348*2 (for 2 MBs at the same time)

Chapter 6. Conclusion and future work

The major contributions of this thesis are summarized into following three parts:

First, in Chapter 3, we adopt the pre-selection algorithm by Li's [7] and propose a mode filtering scheme for our IME with IL prediction and PMRME concept. The number of prediction modes is reduced to only 3 by our mode filtering scheme. The RD performance compared with reference software [4] is 3.542% in bit-rate increase and 0.106 dB in PSNR degradation.

Second, we propose the hardware architecture for the adopted efficient IL prediction algorithm [4]. The proposed IL architecture has three advantages: first, the reference data can be shared between INTER and IL prediction, thus, we can save the high memory bandwidth caused by different prediction modes. Second, the inter-layer residual (ILR), inter-layer motion (ILM), and inter-layer motion residual (ILMR) mode from IL prediction can be processed with INTER in parallel due to the data sharing scheme. Hence, we can reduce the computing time of IME stage. Last, the ME module of INTER mode can be shared with IL prediction due to the data sharing scheme, thus, possible gate count from different prediction modes can be saved.

Third, in Chapter 4, we propose the architecture for the adopted fast FME algorithm SPFME according to the previous work [3]. The new architecture is four times faster than the previous one due to the parallelism of interpolations of different row data, thus, it can process 3 candidates from IME. To achieve the speed up of FME, we further cut the SRAM of FME into 4 banks to fetch data from successive 4 rows.

In the future work, the area of hardware is still an issue. Each element of our ME design can still be optimized to get smaller area.

Reference

[1] T. Wiegand, G. Sullivan, J. Reichel, H. Schwarz and M. Wien, ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6: JVT-X201 'Joint Draft ITU-T Rec. H.264 | ISO/IEC 14496-10/Amd.3 Scalable video coding,' 24th Meeting, Geneva, Switzerland, Jun.29-Jul.5, 2007.

[2] C.-C. Lin, Y.-K. Lin and T.-S. Chang, "PMRME: A parallel multi-resolution motion estimation algorithm and architecture for HDTV sized H.264 video coding," *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Proceeding,* vol.2, pp.II-385-II-388, April 2007.

[3] Tzu-Yun Kuo, Yu-Kun Lin, and Tian-Sheuan Chang, "SIFME: A single iteration fractional-pel motion estimation algorithm and architecture for HDTV sized H.264 video coding," *IEEE Int. Conf. Acout., Speech, Signal Process*, vol. 1, pp. 1185–1188, 2007.

[4] Hsiao-Shan Huang, Gwo-Long Li, and Tian-Sheuan Chang, "Low Memory Bandwidth Prediction Method for H.264/AVC Scalable Video Extension," *Proceeding of APSIPA Annual Summit and Conference*, pp. 294-298, Oct. 2009.

[5] Giwon Kim, Jaemoon Kim, Student Member, IEEE, Chong-Min Kyung, Fellow, IEEE, "A single-pass fractional motion estimation architecture for H.264 video codec," *Proceeding of IEEE International Conference on Multimedia and Expo*, pp. 661-666, July 2010

[6] ISO/IEC International Standard 14496-10, Information technology – Coding of audio-visual objects – Part 10: Advanced Video Coding, third edition, Dec. 2005, corrected version, March 2006.

[7] Gwo-Long Li and Tian-Sheuan Chang, "An Efficient Mode Pre-Selection Algorithm for H.264/AVC Scalable Video Extension Fractional Motion Estimation,"

IEEE International Conference on Digital Signal Processing, Corfu, Greece, July 2011.

[8] C.-C. Yang, K.-J. Tan, Y.-C. Yang, and J.-I. Guo, "Low complexity fractional motion estimation with adaptive mode selection for H.264/AVC," *Proceeding of IEEE International Conference on Multimedia and Expo*, pp.673-678, July 2010.

[9] C.-C. Lin, Y.-K. Lin, and T.-S. Chang, "A fast algorithm and its architecture for motion estimation in MPEG-4 AVC/H.264," in Proceeding of *Asia Pacific Conference on Circuits and Systems*, pp.1250-1253, December 2006.

[10] H. Nisar and T.-S. Choi, "Fast and efficient fractional pixel motion estimation for H.264/AVC video coding," in Proceeding of *IEEE International Conference on Image Processing*, pp.1561-1564, October 2008.

[11] C.-Y. Kao, C.-L. Wu, and Y.-L. Lin, "A high-performance three-engine architecture for H.264/AVC fractional motion estimation," *IEEE Transactions on Very Large Scale Integration System*, vol.18, no.4, pp.662-666, April 2010

[12] Y.-J. Wang, C.-C. Cheng, and T.-S. Chang, "A fast algorithm and its VLSI architecture for fractional motion estimation for H.264/MPEG-4/AVC video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol.17, no.5, pp.578–583, May 2007.

[13] Chao-Yang Kao and Youn-Long Lin, Senior Member, "A memory-efficient and highly parallel architecture for variable block Size integer motion estimation in H.264/AVC," *IEEE Very Large Scale Integration (VLSI) Systems,* vol.18, issue.5, pp. 866–874, June 2010.

[14] Xianghu Ji, Chuang Zhu, Huizhu Jia, Xiaodong Xie, Haibin Yin, "A hardware-efficient architecture for multi-resolution motion estimation using fully reconfigurable processing element array," *2011 IEEE International Conference Multimedia and Expo (ICME),*pp. 1–6 , 11-15 July 2011.

[15] Z.Y. Liu, Y. Song, M. Shao, S. Li, L.F. Li, Ishiwata, S., Nakagawa, M., Goto, S., Ikenaga T., “HDTV 1080P H.264/AVC encoder chip design and performance analysis,” *IEEE J. Solid-State Circuits*, vol. 44, no. 2, 816 pp. 594–608, Feb. 2009.

[16] Giwon Kim, Jaemoon Kim, Chong-Min Kyung, “A low cost single-pass fractional motion estimation architecture using bit clipping for H.264 video codec,” *2010 IEEE International Conference Multimedia and Expo (ICME)*, pp. 661–666, 19-23 July 2010

[17] Nam Thang Ta, Jim Rim Choi, Jae Hoon Kim, Seon Cheol Hwang, Shi Hye Kim, “Fully parallel fractional motion estimation for H.264/AVC encoder,” *2009 Intelligent Computing and Intelligent Systems (ICIS) conference,* vol. 4, pp. 306–309, 20-22 Nov. 2009