國立交通大學

電子工程學系 電子研究所碩士班

碩　士　論　文

應用於SVC視訊編碼標準之空間可適性內幀解碼器設計

Design of An Intra Predictor with Spatial Scalability for Scalable Video Decoding

學生 ： 賴昱帆

指導教授 ： 李鎮宜 教授

中華民國九十八年七月

應用於 SVC 視訊編碼標準之空間可適性內幀解碼器設計

Design of An Intra Predictor with Spatial Scalability for Scalable Video Decoding

研 究 生：賴昱帆　　Student：Yu-Fan Lai

指導教授：李鎮宜 博士　　Advisor：Dr. Chen-Yi Lee

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 Doctor of Philosophy

in

Electronics Engineering
Jul 2009
Hsinchu, Taiwan, Republic of China

中華民國九十八年七月

應用於SVC視訊編碼標準之空間可適性內幀解碼器設計

學生：賴昱帆　　　　　　　　　　指導教授：李鎮宜 教授

國立交通大學

電子工程學系 電子研究所碩士班

摘要

基於 H.264 視訊標準下之可適性視訊壓縮標準(SVC)，是新一代的視訊壓縮規格。與之前的標準相比，在同一種 profile 下，SVC 支援了更多的可適性演算法來大大提升視訊的壓縮率。在空間可適性方面，SVC 跟隨著以往多層的演算架構。向量移動補償和內幀預測同樣存在在每一層空間層裡面，就像以往單層的情況一樣。但是，相比較於每一層都是獨立的 simulcast 來說，為了要更加提高壓縮的效率，因此在 SVC 中引進了一種層與層之間的預測，這種新加入的方法就叫做*層間預測 (Inter-layer Prediction)*。

在 SVC 的架構當中，特別的是基層(Base Layer)要能夠與傳統 H.264 的標準相符合，並且要能夠解出正確的資料。因此，一個完整的 SVC 解碼器不但是要能夠解碼出原來 H.264 的位元流，同時也要能夠解碼出 SVC 格式下的位元流。基於上述，我們提出了一個用在 high profile SVC 解碼器上的內幀預測架構，而這個架構主要是由兩部分組成：基本內幀預測以及 Intra_BL 內幀預測。基本內幀預測是用來處理傳統 H.264 的內幀區塊，在 High profile 的規格下，它支援了 macroblock-adaptive frame field (MBAFF)

的視訊格式，為了減少暫存器的使用量，我們透過重覆使用上方、左方以及角落的暫存器，來優化對圖素的存取，必且提升了存取上面的使用效率。另一個在 High profile 下支援的演算法為 Intra_8x8，我們利用提出的 base-mode 預測器來簡化 RSFP (Reference Sample Filtering Process)的過程，同時也優化了暫存器的使用量以及處理的時間。

另一方面，對於處理在 SVC 中提供的新內幀預測模式 Intra_BL，我們也提出了另一個預測模組，叫做 Intra_BL 內幀預測模組。這個 Intra_BL 內幀預測模組是由 banked SRAM、橫向的基本插補運算單元、縱向的基本插補運算單元以及縱向的延伸式插補運算單元所組成。對於這些插補運算單元，我們也優化了其對面積上面的設計。基於這個初步設計的 Intra_BL 內幀預測模組，我們更進一步提出了對於低功率消耗上的改良式低功耗 Intra_BL 內幀預測模組設計。主要的改進分為兩部分：第一部分為在原來記憶體階級中，我們加入了第二級暫存器的設計；第二部份為我們利用相等特性來修改了基本插補運算單元的處理流程。加入了這兩種方法，我們可以節省全部功率消耗的 46.43%。

最後，我們利用 90 奈米製程技術實作了整個 SVC 內幀預測架構，跑在頻率為 145 MHz 情況下，總面積為 42756 NAND2 CMOS gates。另外在功率損耗上面，分別跑在頻率為 100 MHz 的 H.264 規格下以及頻率為 145 MHz 的 SVC 規格下，功率消耗為 0.292 mW 以及 2.86 mW。而這個設計可以在頻率為 100 MHz 的 H.264 視訊標準下達到每秒 30 張 HD1080 的畫面大小，並且可以在頻率為 145 MHz 的 SVC 視訊標準下最大支援到每秒 30 張 HD720 和 HD1080 兩層空間層的解碼。

Design of An Intra Predictor with Spatial Scalability for Scalable Video Decoding

Student : Yu-Fan Lai   Advisor : Dr. Chen-Yi Lee

Department of Electronics Engineering

Institute of Electronics

National Chiao Tung University

ABSTRACT

Scalable Video Coding (SVC) extension of the H.264/AVC is the latest standard in video coding. It has achieved significant improvements in coding efficiency with an increased degree of supported scalability relative to the scalable profiles of prior video coding standards. For supporting spatial scalable coding, SVC follows the conventional approach of multilayer coding. In each spatial layer, motion-compensated prediction and intra-prediction are employed as for single-layer coding. But in order to improve coding efficiency in comparison to simulcasting different spatial resolutions, additional so-called *inter-layer prediction* mechanisms are incorporated.

In particular, H.264/AVC compatible bitstream needs to be decoded in the base layer of SVC. Therefore, a SVC decoder must support both traditional H.264 decoding and SVC extension decoding. Specifically, we propose a high profile SVC intra prediction engine which is composed of two major prediction modules, basic prediction module and Intra_BL prediction module. Basic prediction module is used to decode the traditional H.264 intra prediction. In order to reduce the buffer size for supporting macroblock-adaptive frame field (MBAFF) coding which is supported in high profile, we optimize the buffer size via upper, left, and corner data reuse sets (DRS) to reuse the pixels and improve the cost and access efficiency. In Luma_8x8 decoding process, we simplify the RSF process via a base-mode predictor and optimize the processing latency and buffer cost.

For the Intra_BL prediction module which is used to decode the new intra prediction type called “Intra_BL”, we propose an Intra_BL prediction engine that consists of banked SRAM, basic horizontal interpolator, basic vertical interpolator and extended vertical interpolator. We also optimize the architecture of interpolators to have better area efficiency than direct implementation. Based on our preliminary Intra_BL prediction module design, we further propose a power efficient Intra_BL prediction module. By applying a second stage of register sets in memory hierarchy and equality determination before basic interpolation process, a total of 46.43% power consumption can be reduced.

Finally, the architecture of this power efficient SVC intra prediction engine is implemented in a 90nm technology with a total area of 42756 NAND2 CMOS gates under working frequency of 145 MHz. The power consumption is 0.292 mW and 2.86 mW under frequency of 100 MHz and 145 MHz for H.264 and SVC, respectively. This design can achieve real-time processing requirement for HD1080 format video in 30fps under the working frequency of 100MHz in H.264, and for a maximum two spatial layers with HD720 and HD1080 scalable format video in 30fps under the working frequency of 145MHz in SVC.

誌　　　　謝

拿到老師及口試委員們簽名的口試通過單，這一刻我才意識到，我真的畢業了！我非常榮幸能夠在 Si2 實驗室畢業，能夠成為 SI2 的一份子，我真是倍感驕傲。

首先我要謝謝這兩年不斷指導我的**李鎮宜教授**，老師總是用很親切的態度指導我，給了我很多研究上面的觀念，處理、分析問題的方法，以及報告的技巧…等等。雖然老師平常有很多的事要處理，不過老師總是一一回覆每個學生的問題。能跟著老師做研究，不僅是在研究上面有很大的幫助，在做人做事以及平常生活的態度上面，更是有所收穫，謝謝老師！另外要感謝的是**蔣迪豪教授**，雖然蔣老師平常也忙於公司上面的事情，但是只要老師有控，就會來參加我們group的meeting，並且從業界的角度來給予我們不一樣的建議。還有要感謝我們SI2 的大學長**鍾菁哲助理教授**，願意抽空回來當我的口委。雖然他目前在中正大學任職，但是學長一直不斷為著SI2 做很多事情。他總是默默付出，如果可以的話真希望學長可以留下來！

再來我要感謝帶領我們 multimedia group 的李曜學長，他親切熱心的態度，總是讓我對他充滿感激。不管再小、再基本的問題，他總是很有耐心的指導我。總是把別人的事情放第一，自己放第二的他，是我永遠敬佩的學長。另外也要謝謝同 group 的學長學弟，Mingle、韋磬、阿德、建州、明諭、均宸、浩民、建辰以及昱錚，在我研究之路上給我很多支持以及鼓勵，也帶了很多歡樂給我，謝謝你們！

在這也要謝謝其他 SI2 的學長及同學，阿龍、義閔、義澤、建螢、點子、Circus、Fishya、豪哥、柏均、芳年、小馬、欣儒、智超、耀琳、佳龍，在我遇到問題的時候，總是提供幫助給我，SI2 真是讓人不捨離開的地方！搞不好過幾年我就想回來唸博班了，Kelvin 等我阿，別太早走喔！

最後我要謝謝我的家人以及教會的朋友們，每個禮拜回去總是得到很多的關心以及支持，讓我能夠充滿精神開始下禮拜的研究。這一切的一切，有太多說不完的感謝，感謝上帝！

Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Motivation

H.264/AVC [1]-[2] is the latest international video coding standard from MPEG and ITU-T Video Coding Experts Group. It consists of three profiles which are defined as a subset of technologies within a standard usually created for specific applications. Baseline and main profiles are intended as the applications for video conferencing/mobile and broadcast/storage, respectively. Considering the H.264-coded video in high profile, it targets on the compression of high-quality and high-resolution video and becomes the mainstream of high-definition consumer products such as HD DVD and Blu-ray disc. However, high-profile video is more challenging in terms of implementation cost and access bandwidth since it involves extra coding engine, such as macroblock-adaptive frame field (MBAFF) coding (using a macroblock pair structure for pictures coded as frames, allowing 16×16 macroblocks in field mode) and Luma 8×8 intra coding, for achieving high performance compression.

On the other hand, considering that future applications will support a different range of display resolutions and transmission channel capacities, the Joint Video Team (JVT) has developed a scalable extension [3]-[4] based on the state-of-the-art H.264/AVC standard which was completed in November 2007. This extension is commonly known as Scalable Video Coding (SVC), and it provides multiple display resolutions within a single compressed bit-stream, which is referred to *spatial scalability* in SVC. Additionally, the SVC extension supports combinations of *temporal scalability* (frame rate) and *quality scalability* (fidelity enhancement under the same resolution) with the spatial scalability feature. This is achieved while balancing both decoder complexity and coding efficiency.

The need for spatial scalability is motivated by the resolutions variety of current display devices. Specifically, larger format, high definition displays are becoming commonly used in consumer applications, with displays containing over two million pixels. By contrast, lower-resolution displays with between ten thousand and one hundred thousand pixels are also popular in applications due to the considerations of size, power, and weight. However, transmitting a single representation of a video sequence to the different of display resolutions in the market is impractical. For example, it is rarely reasonable to design a device with low display resolution with the capacity for decoding and downsampling high-resolution video sequence. This kind of requirement could increase the cost and power of the device in order to determine its display resolution. In addition, it may be a waste of channel capacity to send the high-resolution video bitstreams that are ultimately not shown on the display for such a device.

Therefore, the new scalable extension of H.264 standard can not only support multiple resolutions decoding but also provide better compression compared to the prior standards. Furthermore, considering that the overall complexity and other overheads will be increased in SVC, the proposed methods can be efficient to reduce the area, computational complexity, and power consumption.

1.2 Thesis Organization

This thesis is organized as follows. In Chapter 2, the intra prediction process in different profiles and some important coding tools which for achieving high performance compression will be described. For the additional features in intra prediction of SVC such as spatial scalability, new prediction mode and inter-layer intra prediction between different video types (progressive or interlace) will also be introduced. Then, the architecture design and implementation results of intra prediction engine which supports H.264/AVC high profile and scalable extension will be shown in Chapter 3 and Chapter 4, respectively. Finally, the conclusions and future works will be presented in Chapter 5.

Chapter 2
Overview the Intra Prediction in H.264/AVC and Scalable Video Coding

2.1 H.264/AVC Standard Overview

H.264/AVC is the newest international video coding standard from MPEG and ITU-T Video Coding Experts Group. It provides various kinds of powerful but complex techniques to achieve higher video compression, such as spatial prediction in intra coding, adaptive block-size motion compensation, 4×4 integer transformation, context-adaptive entropy coding, adaptive deblocking filtering, and etc.

2.1.1 Profiles and Levels

In order to achieve different specific classes of applications, the standard includes the following profiles. We take four commonly used profiles to be described.

- **Baseline Profile:** This profile is used widely for lower-cost applications with limited computing resources, such as in videoconferencing and mobile applications.
- **Main Profile:** Intended as the mainstream consumer profile for broadcast and storage applications. However, the importance of this profile is faded when the high profile was developed for those applications.
- **Extended Profile:** This profile is intended for the streaming video profile, and has relatively high compression capability and some extra methods for robustness to error and server stream switching.
- **High Profile:** The primary profile for broadcast and disc storage applications,

particularly for high-definition television applications (for example, HD DVD and Blu-ray Disc).

Figure 1 shows the difference between coding techniques in different profiles of H.264/AVC standard. To be noticed, the coding tools “Interlace” and “8x8 Integer DCT” in Main and High profiles are referred to the Macroblock-Adaptive Frame-Field (MBAFF) coding (or Picture-Adaptive Frame-Field (PAFF) coding) and Luma *Intra_8x8* prediction mode.

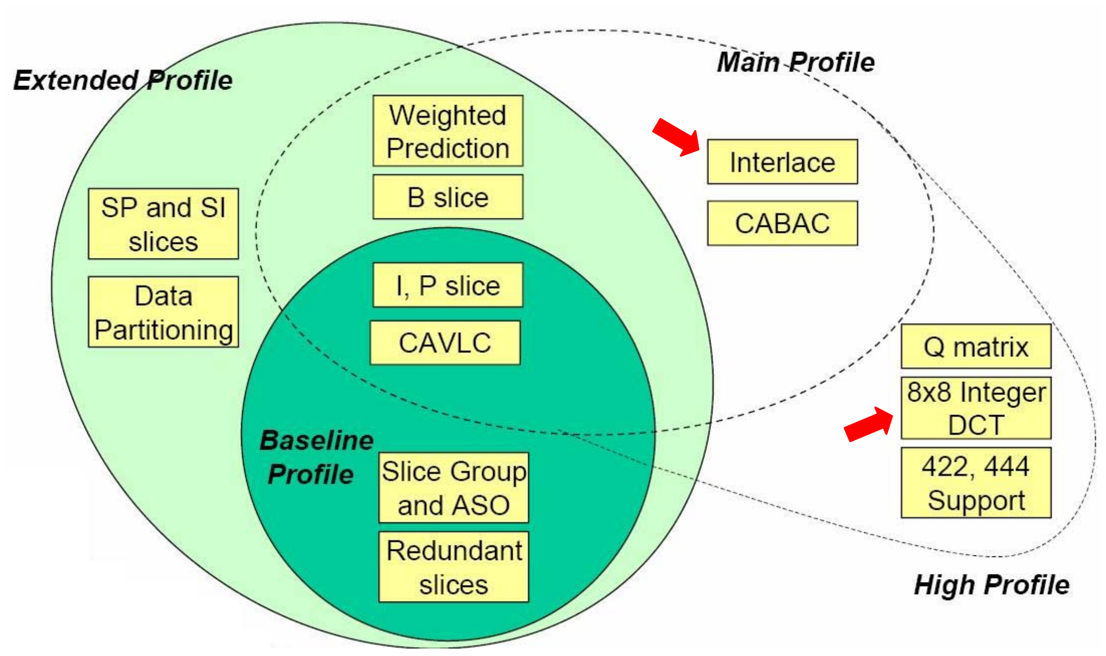


Figure 1: The contained techniques in different profiles of H.264/AVC standard.

These two additional coding tools can further increase the performance of compression. PAFF and MBAFF was reported to reduce bit rates in the range of 16 to 20% and 14 to 16% over frame-only and PAFF for ITU-R 601 resolution sequences like “Canoa” and “Mobile and Calendar”, respectively [2]. Moreover, *Intra_8x8* supports more options to predict the block, and provides more efficient compression than *Intra_4x4* and *Intra_16x16* only. For more details about coding tool difference are listed in Table 1.

Table 1: Coding tools in different profiles of H.264/AVC standard

Coding Tools	Baseline	Main	Extend	High
I and P Slices	Yes	Yes	Yes	Yes
B Slices	No	Yes	Yes	Yes
SI and SP Slices	No	Yes	No	No
Multiple Reference Frames	Yes	Yes	Yes	Yes
In-Loop Deblocking Filter	Yes	Yes	Yes	Yes
CAVLC Entropy Coding	Yes	Yes	Yes	Yes
CABAC Entropy Coding	No	No	Yes	Yes
Flexible Macroblock Ordering (FMO)	Yes	Yes	No	No
Arbitrary Slice Ordering (ASO)	Yes	Yes	No	No
Redundant Slices (RS)	Yes	Yes	No	No
Data Partitioning	No	Yes	No	No
Interlaced Coding (PicAFF, MBAFF)	No	Yes	Yes	Yes
4:2:0 Chroma Format	Yes	Yes	Yes	Yes
Monochrome Video Format (4:0:0)	No	No	No	Yes
4:2:2 Chroma Format	No	No	No	No
4:4:4 Chroma Format	No	No	No	No
8 Bit Sample Depth	Yes	Yes	Yes	Yes
9 and 10 Bit Sample Depth	No	No	No	No
11 to 14 Bit Sample Depth	No	No	No	No
8x8 vs. 4x4 Transform Adaptivity	No	No	No	Yes
Quantization Scaling Matrices	No	No	No	Yes
Separate Cb and Cr QP control	No	No	No	Yes
Separate Color Plane Coding	No	No	No	No

Predictive Lossless Coding	No	No	No	No

2.1.2 Encoder/Decoder Block Diagram

As mentioned before, H.264/AVC standard provides much higher video compression performance by some new coding tools. However, the complexity is also increased. Figure 2 shows the block diagram of H.264/AVC video encoder. An embedded decoder exists inside the encoder that calculates the motion compensation and intra prediction at the decoder side. With this embedded decoder in the encoder side, the encoder can foresee the decoded result and precisely calculate the residual pixel values without mismatch to the decoder. One of the major differences compared to previous standard is the intra prediction in H.264/AVC standard. Several prediction modes are provided for the intra prediction to highly improve the compression ability in the spatial domain. On the other side, the motion compensation supports variable block sizes, multiple reference frames, and short/long term prediction to reduce the redundancy in the temporal domain. After subtracting the input video with the predicted pixels which are from the intra prediction or motion compensation, the residual values are processed with DCT, quantization and entropy coding to reduce the coding redundancy. Moreover, H.264/AVC uses powerful block-based compression methods to increase the performance. However, another problem called “blocky effect” will be generated. In order to solve this problem, H.264 uses the in-loop filter, deblocking filter, to improve the visual quality. Finally, the coded bit stream is produced and transmitted.

On the decoder side, it lacks of the decision parts like motion estimator and the intra mode decision parts. Hence, the decoder is simpler than the encoder. Figure 3 shows the block diagram of H.264/AVC video decoder. After entropy decoding the input bit stream, the syntax elements are decoded to decide which mode is used in motion compensation or intra prediction. Then, the residual values are generated by inverse quantization and IDCT. With residual values and predicted pixel values, the video can be constructed. Finally, the deblocking filter is invoked to eliminate the blocky effects and improve the visual quality.

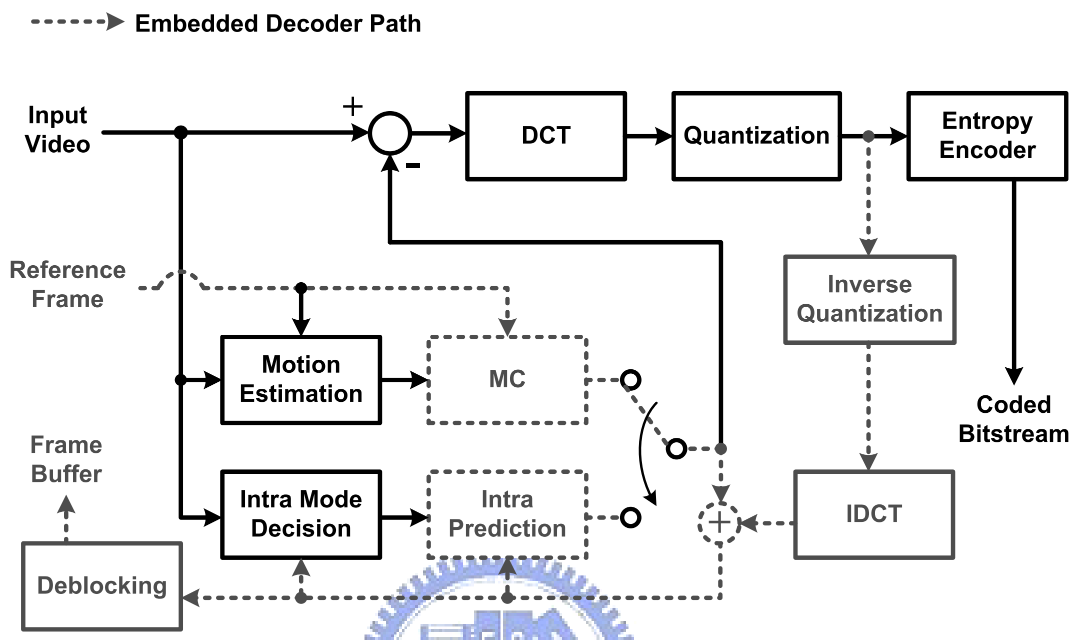


Figure 2: Block diagram of H.264/AVC video encoder.

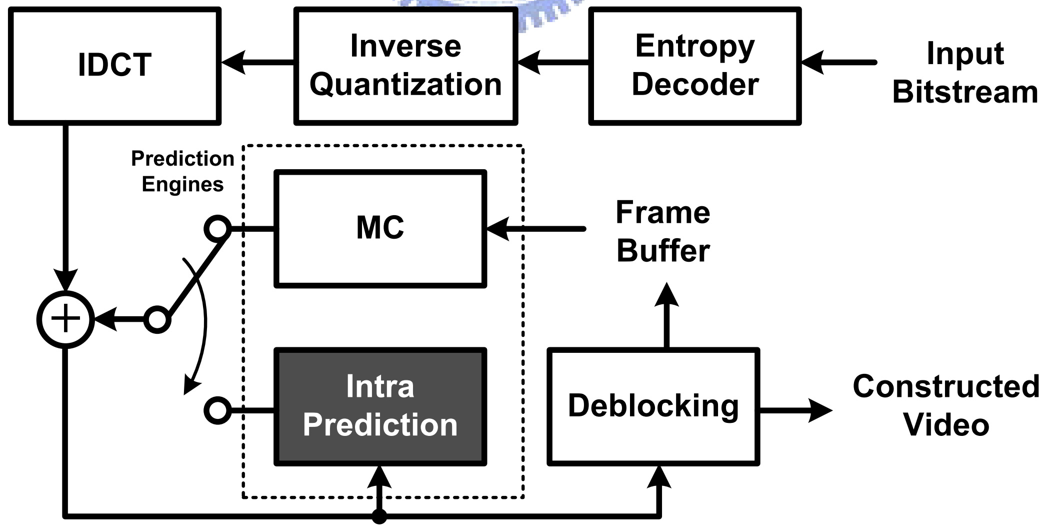


Figure 3: Block diagram of H.264/AVC video decoder.

2.1.3 Intra Block Decoding

As shown in Figure 3, the intra prediction (the grey block) is one of the major prediction engines in H.264/AVC standard. In the previous section, we have mentioned about that H.264 provides several prediction modes in intra prediction to improve the compression ability in the spatial domain. There are two classes of intra prediction modes in baseline and main profile of H.264/AVC, the *Intra_4x4* prediction mode and *Intra_16x16* prediction mode. Besides, the third class of intra prediction called *Intra_8x8* which is supported in high profile will be further introduced in Section 2.1.5.

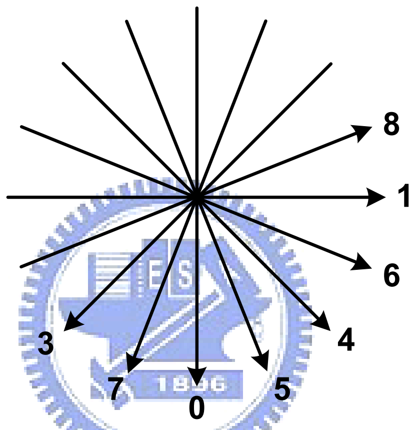


Figure 4: Intra_4x4 prediction mode directions.

There are a total of 9 optional prediction modes for *Intra_4x4* prediction mode. The arrows in Figure 4 indicate the direction of prediction in each mode. These 9 modes are vertical (0), horizontal (1), DC (2), diagonal down-left (3), diagonal down-right (4), vertical-right (5), horizontal-down (6), vertical-left (7), and horizontal-up (8), respectively.

The prediction 4x4 blocks (which are grey color in Figure 5) are calculated based on the neighboring samples labeled A-M in Figure 5, as follows. In DC modes, the intra prediction process is to calculate the mean value of neighboring pixel values. Except for DC mode, all the others are calculated by four neighboring pixel values. These four pixels are different according to the type of mode and the position in the 4x4 block.

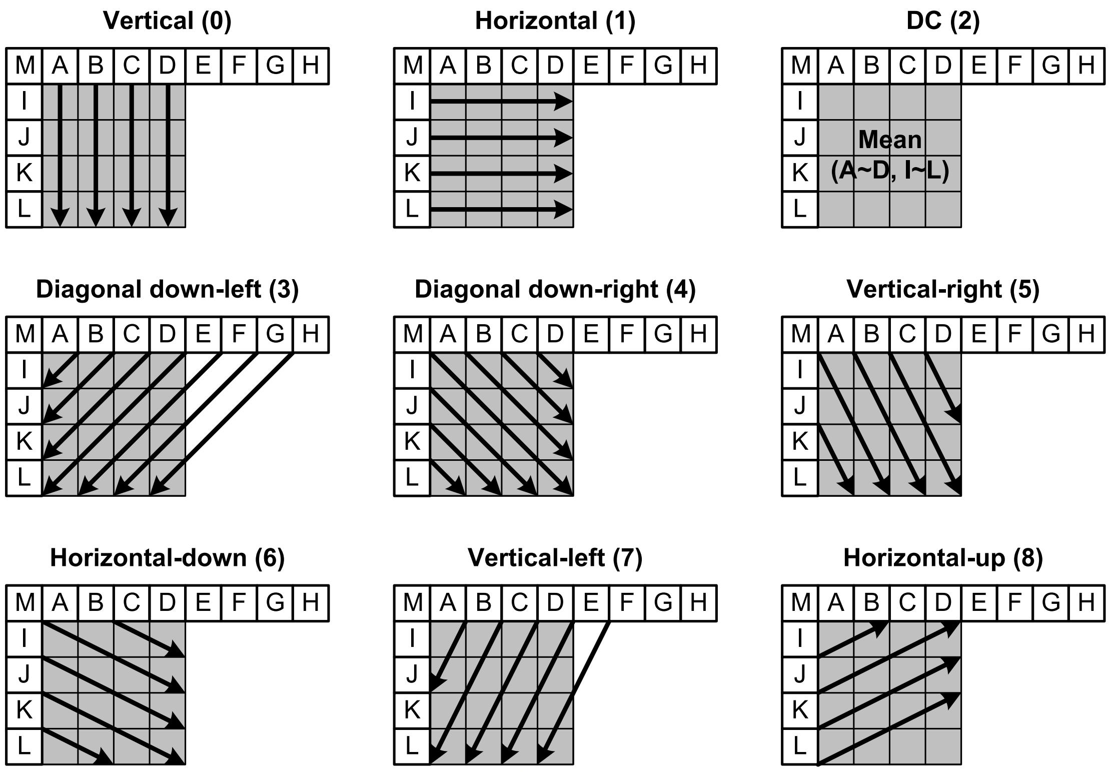


Figure 5: Intra_4x4 prediction modes based on the neighboring pixels.

In the *Intra_16x16* prediction mode class, there are a total of 4 optional prediction modes, which are vertical, horizontal, DC, and plane modes, as shown in Figure 6, respectively. The vertical, horizontal, and DC modes are very similar to the *Intra_4x4* prediction modes described before. The new prediction mode in Intra_16x16 is plane mode, which is a "linear" plane function that is fitted to the upper and left-hand samples H and V. This works well in areas of smoothly-varying luminance.

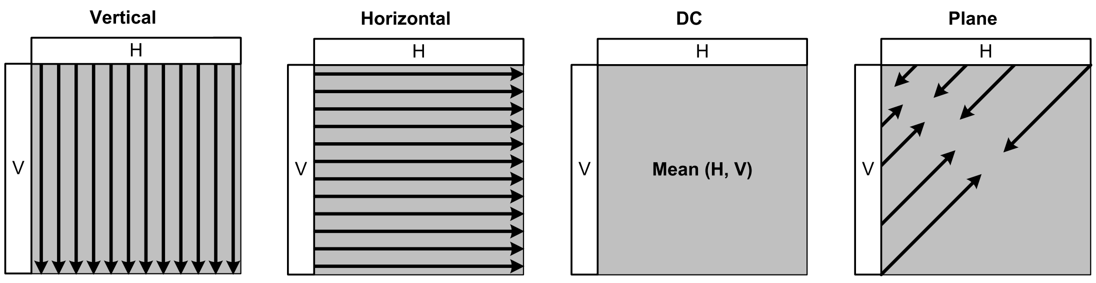


Figure 6: Intra_16x16 prediction modes based on the neighboring pixels.

For a luma samples in a macroblock, both *Intra_4x4* prediction modes and *Intra_16x16* prediction modes are valid. However, only 4 prediction modes which are very similar to the 16x16 luma prediction modes but with a little different in parameters are valid in chroma samples.

2.1.4 Macroblock-Adaptive Frame-Field Intra Decoding

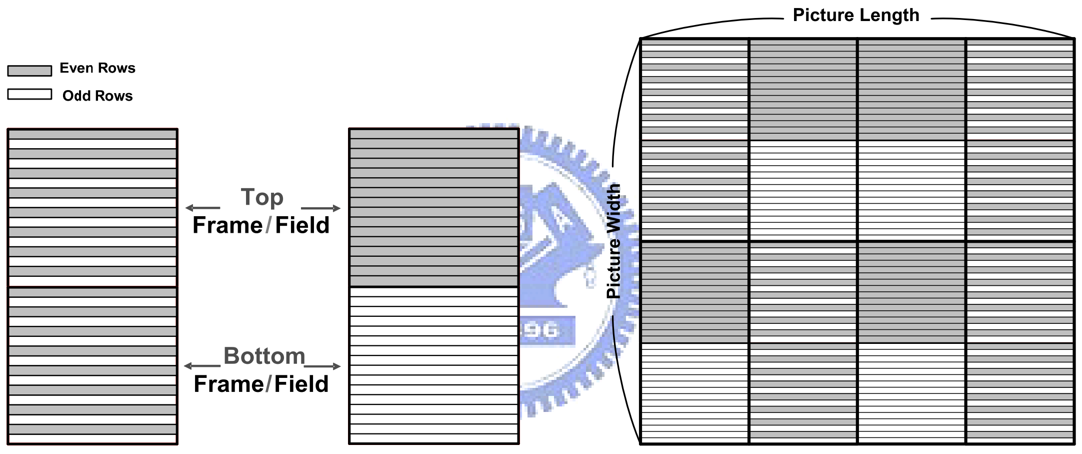
There usually exist two types of video source formats, i.e. progressive and interlaced video, for which different techniques were presented in a range of video standards, such as H.261 [5], H263 [6], MPEG-1 [7], MPEG-2 [8], and MPEG-4 [9] as well as in H.264/AVC [1] a newly released standard. In particular, a frame of interlace video sequence consists of two fields, scanned at different time instants. Therefore, a video sequence consisting of interlaced video can be compressed in many different ways. They can be grouped in two main categories: (a) Fixed Frame/Field and (b) Adaptive Frame/Field.

2.1.4.1 Types of Video Source Formats

In the first category, all the frames of a sequence are encoded in either frame or field mode. In the second category, depending on the content, one adaptively selects whether to use the field or frame structure. Adaptation can be done at either picture level or macroblock level. For the picture level adaptation of frame/field (PAFF), an input frame of a sequence can be encoded as one frame or two fields. If frame mode is chosen, a picture created by interleaving both even number pixels and odd number pixels is partitioned into some 16x16 macroblocks (MBs) and coded as a whole frame. However, in field coding, the top and bottom field pictures are partitioned into 16x16 MBs and coded as separate pictures. For MB level adaptation of frame/field coding (MBAFF), the frame is partitioned into 16x32 MB pairs, and the two MBs in each MB pair are both coded in frame mode or field mode, as shown in

Figure 7(a). Further, any combinations of MB pairs can also exist in a picture, as illustrated in Figure 7(b) (In Figure 7, we assume that picture length and width are both 64 pixels).

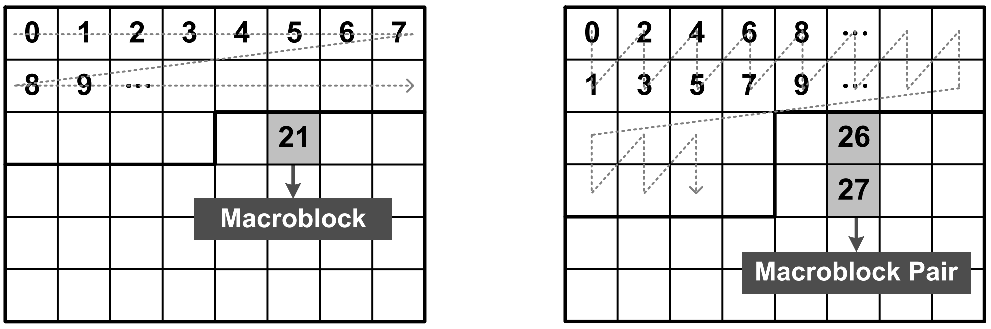
Therefore, there exist two decoding units according to the video source format. One is called MB in fixed frame/field video source format and picture level adaptive frame/field video source format, and the other is called MB pair in MBAFF video source format. The sizes of MB and MB pair are 16x16 and 16x32, respectively. The scanning order in different decoding units of a frame and the relationship between current unit and neighboring units are described in Figure 8(a) and Figure 8(b).



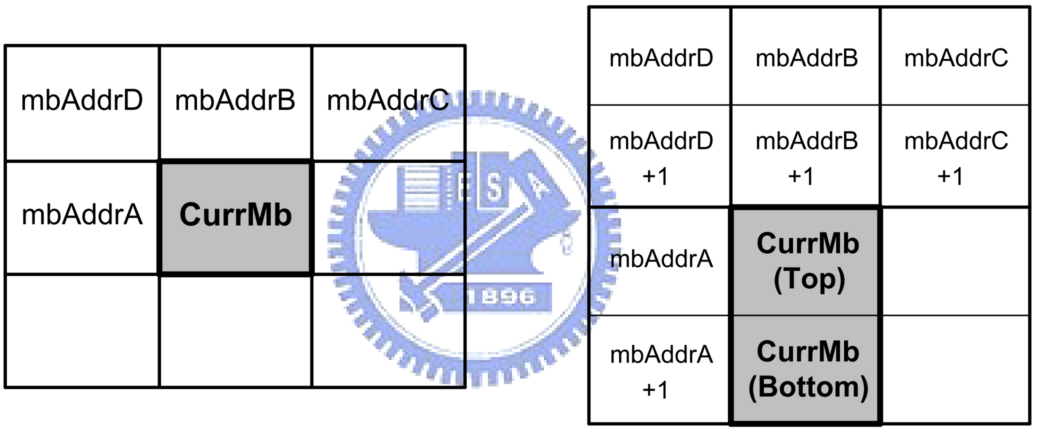
(a) (b)

Figure 7: (a) A MB pair which is coded in frame mode and field mode, and (b) an example of different combinations of MB pairs in a picture.

In a short summary about video sequence formats, it is believed that frame coding is more efficient than field coding for progressive video and static pictures in interlaced video, while field coding is more efficient for moving pictures in interlaced video. Thus, picture-adaptive frame/field (PAFF) coding is supported in H.264 to improve the coding efficiency for interlaced video with both static and moving pictures. Obviously, PAFF coding requires the encoder has the ability to adaptively select the picture coding mode, which is realized by 2-pass coding in reference software of H.264, i.e., frame mode and field mode are performed respectively, and the one with smaller rate-distortion cost is chosen as the optimal picture coding mode.



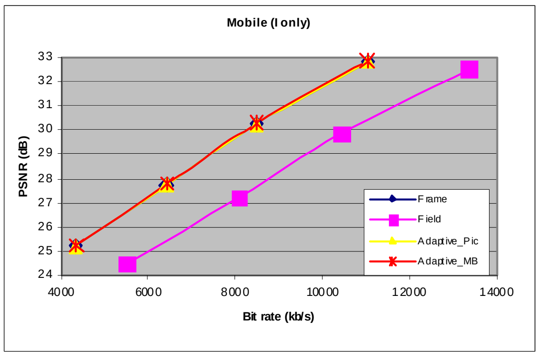
(a)



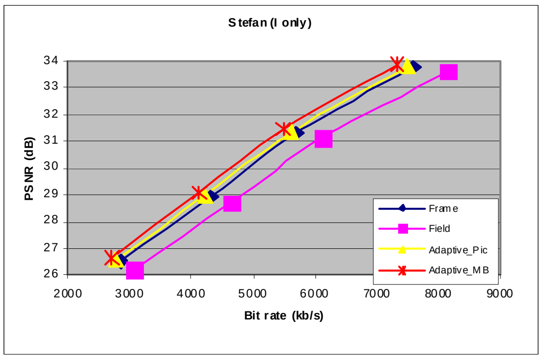
(b)

Figure 8: (a) Scanning order in different decoding units of a frame and (b) relationship between current unit and neighboring units

As an extension of PAFF, MBAFF is used to improve coding efficiency of picture with both static and moving regions. Reference software performs MBAFF coding with a 2-pass coding strategy on MB pair level. Every MB pair is coded in frame mode and field mode respectively, and the mode with smaller rate-distortion cost is selected. In general, MBAFF coding is more efficient for video coding. Figure 9 gives some simulation result about different video sequence formats in different test sequences [10].



(a)



(b)

Figure 9: PSNR vs bit rate curve for the (a) Mobile sequence and (b) Stefan sequence.

2.1.4.2 Intra Block Neighboring Information in MBAFF

As mentioned in the previous section, the even rows and odd rows will be gathered to different MBs (top or bottom MB in a MB pair) according to the different modes of a MB pair. Therefore, the neighboring pixels of an intra block will also be more complex when the MBAFF is supported.

Figure 10 shows the neighboring pixels of a 4x4 intra block in the MB pair. When the current MB pair is coded in frame mode, the neighboring pixels are similar to the baseline profile which has the unit in MB, as illustrated in Figure 10 (a). However, when the current MB pair is coded in field mode, the relationship between current 4x4 block and neighboring pixels will be more complicated, and the neighboring pixels are no more around to a 4x4 block. Depending on the 4x4 blocks in a MB pair, the neighboring pixels will be in different positions, as shown in Figure 10 (b). In a simple way to express the principle about neighboring pixels is that the pixels in top (or bottom) field of current block can only find the neighboring pixels also in top (or bottom) field of neighboring blocks. Besides, if neighboring blocks are not coded in field mode, the first step is to convert the frame mode block into field block. Then, find the corresponding neighboring pixels in the same field. Furthermore, there are a total of 4 cases between current MB pair and neighboring MB pair (frame/frame, frame/field, field/frame, and field/field). Here we just take 2 cases (frame/frame and frame/field) for an example.

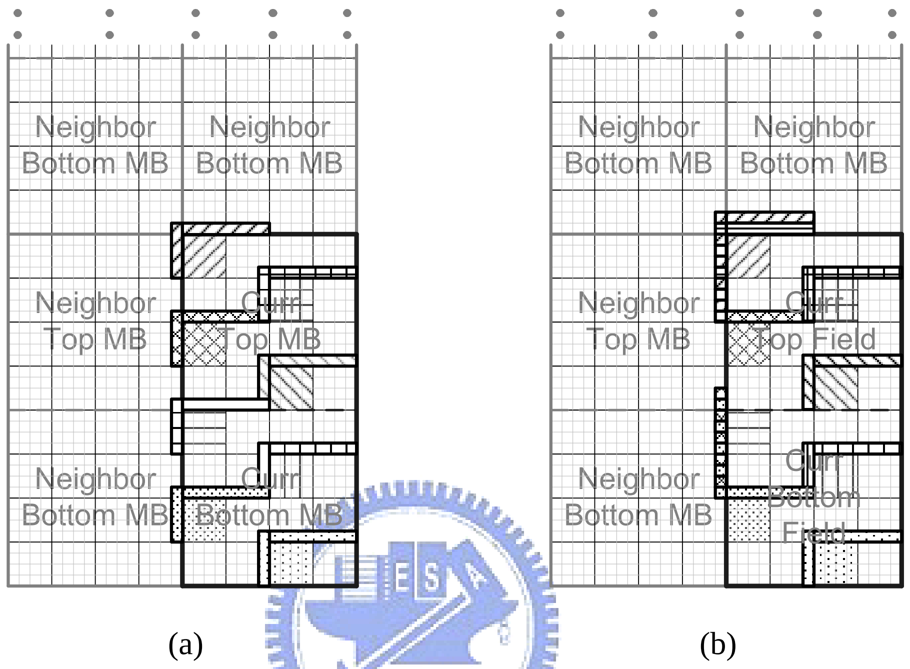
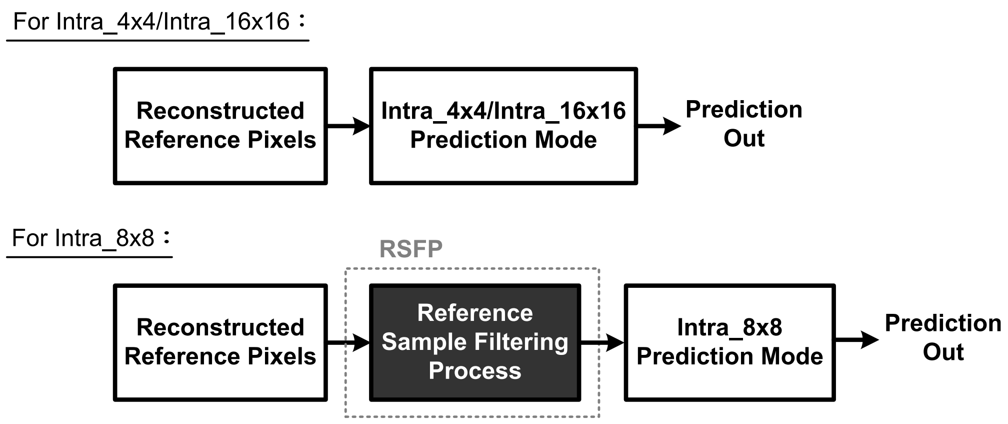


Figure 10: Neighboring information in (a) frame mode MB pair and (b) field mode MB pair (assume neighboring MB pairs are all coded in frame mode).

Obviously, in order to support MBAFF mode, it needs more side information to indicate the frame/field mode of each MB pair. Therefore, how to efficiently store and decode these neighboring pixels and MB pairs becomes an important issue in MBAFF video format.

2.1.5 Intra_8x8 Prediction Mode Decoding

For intra prediction, luma *Intra_8x8* is an additional intra block type supported in H.264 high profile which supports all features of the prior main profile. The luma *Intra_8x8* prediction is introduced by extending the concepts of *Intra_4x4* prediction but with an additional process as described in Figure 11.



For Intra_4x4/Intra_16x16：
Reconstructed Reference Pixels
Intra_4x4/Intra_16x16 Prediction Mode
Prediction Out
For Intra_8x8：
RSFP
Reconstructed Reference Pixels
Reference Sample Filtering Process
Intra_8x8 Prediction Mode
Prediction Out

Figure 11: Difference between Intra_8x8 and Intra_16x16/Intra_4x4

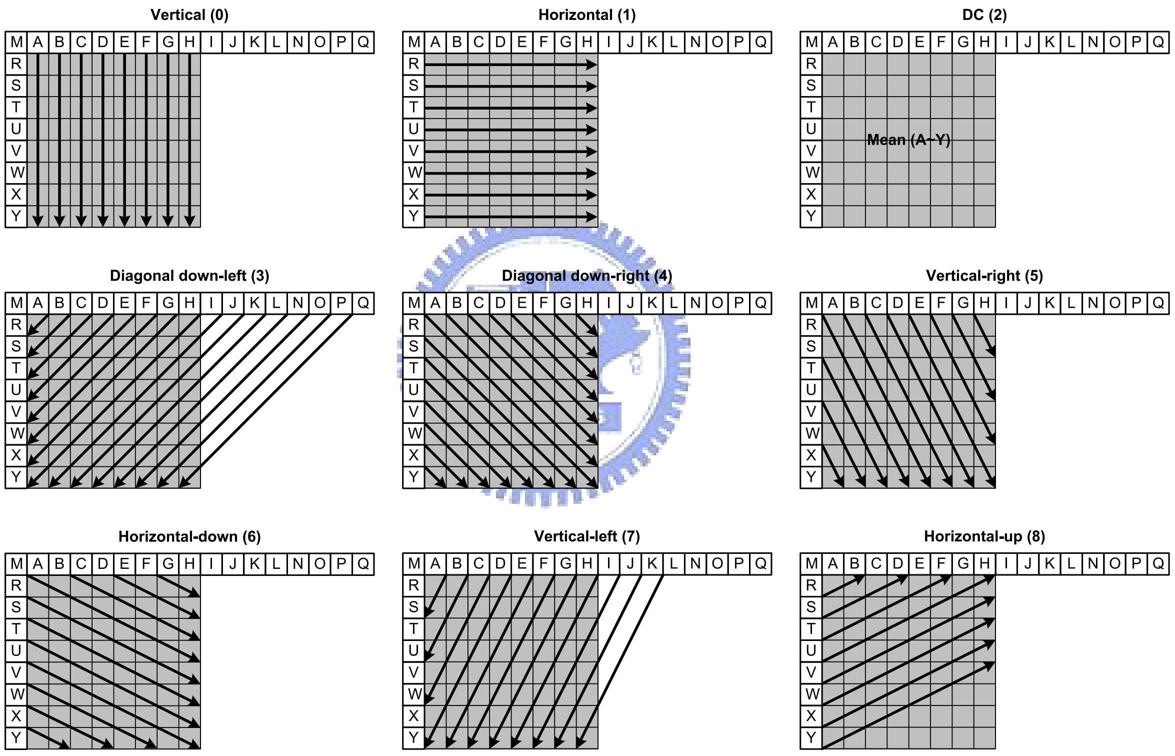


Figure 12: Nine Intra_8x8 prediction modes (note: each neighboring pixel (A~Y) is already filtered by RSFP).

Similar to *Intra_4x4*, *Intra_8x8* has nine different direction prediction modes, as shown in Figure 12. The luma values of each sample in a given 8x8 block are predicted from neighboring reconstructed reference pixels base on prediction modes. It should be noticed that before decoding an *Intra_8x8* block, there is an extra process that is different from *Intra_4x4* and *Intra_16x16*, which called reference sample filtering process (RSFP). Original pixels will be filtered first, and then using these filtered pixels to predict subsequent 8×8 blocks. For an *Intra_4x4* and *Intra_8x8* block, 13 neighbors and 25 filtered neighbors are needed, respectively.

This new intra coding tool not only improves I-frame coding efficiency significantly [11], but also induces some extra cycles and data dependency in order to generate the filtered neighboring pixels.

2.2 SVC Extension of H.264/AVC Standard Overview

Scalable Video Coding (SVC) [12] is the name given to an extension of the H.264/AVC video compression standard. SVC enables the transmission and decoding of partial bit streams to provide video services with lower temporal or spatial resolutions or reduced fidelity while retaining a reconstruction quality that is high relative to the rate of the partial bit streams. Hence, SVC provides some techniques such as graceful degradation in lossy transmission environments, bit rate, format, and power adaptation. By these techniques, SVC can enhance the transmission and storage within applications. Furthermore, SVC has achieved significant improvements in coding efficiency with an increased degree of supported scalability relative to the scalable profiles of prior video coding standards.

Scalable Video Coding as specified in extension of H.264/AVC allows the construction of bitstreams that contain sub-bitstreams which conform to H.264/AVC. For temporal bitstream scalability, it provides the presence of a sub-bitstream with a smaller sampling rate than the bitstream. When deriving the temporal sub-bitstream, complete access units will be removed from the bitstream. For spatial and quality bitstream scalability, these two scalabilities provide the presence of a sub-bitstream with lower spatial resolution or quality than the bitstream. When deriving these kinds of sub-bitstream, the NAL (Network Abstraction Layer) will be removed from the bitstream. In this case, inter-layer prediction (i.e.

the prediction between the higher spatial resolution or quality layer and the lower spatial resolution or quality layer) is typically used for improving the coding efficiency, and will be more explained in Section 2.2.2.

2.2.1 Profiles and Levels

As a result of the Scalable Video Coding extension [13], the standard contains three additional scalable profiles: Scalable Baseline, Scalable High, and Scalable High Intra. These profiles are defined as a combination of the H.264/AVC profile for the base layer and tools that achieve the scalable extension:

- **Scalable Baseline Profile**: Primarily targeted for mobile broadcast, conversational and surveillance applications that require a low decoding complexity. In this profile, the support for spatial scalable coding is restricted to resolution ratios of 1.5 and 2 between spatial layers in both horizontal and vertical direction and to macroblock-aligned cropping. Furthermore, the coding tools for interlaced sources are not included in this profile. Bit streams conforming to the Scalable Baseline profile contain a base layer bit stream that conforms to the restricted Baseline profile H.264/AVC. It should be noted that the Scalable Baseline profile supports B slices, weighted prediction, the CABAC entropy coding, and the 8x8 luma transform in enhancement layers (CABAC and the 8×8 transform are only supported for certain levels), although the base layer has to conform to the restricted Baseline profile, which does not support these tools.
- **Scalable High Profile**: Primarily designed for broadcast, streaming, and storage applications. In this profile, the restrictions of the Scalable Baseline profile are removed and spatial scalable coding with arbitrary resolution ratios and cropping parameters is supported. Bit streams conforming to the Scalable High profile contain a base layer bit stream that conforms to the High profile of H.264/AVC.

- **Scalable High Intra Profile**: Primarily designed for professional applications. Bit streams conforming to this profile contain only IDR pictures (for all layers). Besides, the same set of coding tools as for the Scalable High profile is supported.

  The coding tool differences between SVC profiles are described in Table 2.

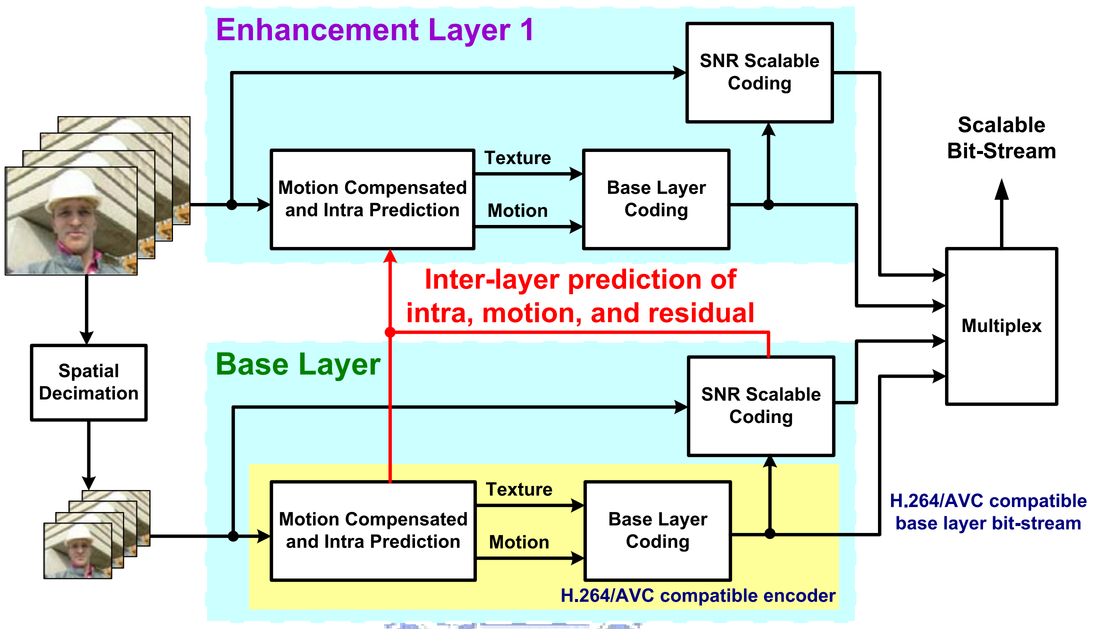
Table 2: Coding tool difference between SVC profiles

Coding Tools	**Scalable Baseline**	**Scalable High**	**Scalable High Intra**
Base Layer Profile	AVC Baseline	AVC High	AVC High Intra
CABAC and 8x8 Transform	Only for certain levels	Yes	Yes
Layer Spatial Ratio	1:1, 1.5:1, or 2:1	Unrestricted	Unrestricted
I, P, and B Slices	Limited B Slices	All	Only I Slices
Interlaced Tools (MBaff & Paff)	No	Yes	Yes

2.2.2 Encoder/Decoder Block Diagram

The example for combining spatial, quality, and temporal scalabilities is illustrated in Figure 13, which shows an encoder block diagram with two spatial layers.

The SVC coding structure is organized in dependency layers (i.e. spatial layers). A dependency layer usually represents a specific spatial resolution, and is identified by a layer identifier, *D*. The identifier *D* equals 0 for the base layer which is also a H.264/AVC compatible layer, and it is increased by 1 from one spatial layer to the next. For each dependency layer, the basic prediction engine of motion compensation and intra prediction in H.264 are used as in single-layer coding. Further, the additional inter-layer prediction which is exploited to reduce the redundancy between dependency layers will be more explained in Section 2.2.3.1. Hence, by applying this new inter-layer prediction in SVC, the coding efficiency related to spatial scalability provides higher coding efficiency than the simulcasting between different spatial resolutions.



Enhancement Layer 1
SNR Scalable Coding
Motion Compensated and Intra Prediction
Texture
Motion
Base Layer Coding
Scalable Bit-Stream
Inter-layer prediction of intra, motion, and residual
Multiplex
Spatial Decimation
Base Layer
SNR Scalable Coding
Motion Compensated and Intra Prediction
Texture
Motion
Base Layer Coding
H.264/AVC compatible base layer bit-stream
H.264/AVC compatible encoder

Figure 13: SVC encoder block diagram (note: enhancement layer can be more than 1).

In a special case that the spatial resolutions for two dependency layers are identical, in which case the different layers provide coarse-grain scalability (CGS) in terms of quality. However, this multilayer concept for quality scalable coding only allows a few selected bit rates to be supported in a scalable bit stream. Therefore, in order to increase the flexibility of bit stream adaptation and error robustness, but also for improving the coding efficiency for bit streams that have to provide a variety of bit rates, an extension of the CGS approach, which is also referred to as medium-grain quality scalability (MGS), is included in the SVC design. The difference between CGS and MGS are a modified high level signaling, which allows a switching between different MGS layers in any access unit. Finally, these scalable sub-streams will be organized to a scalable bit stream and transmitted.

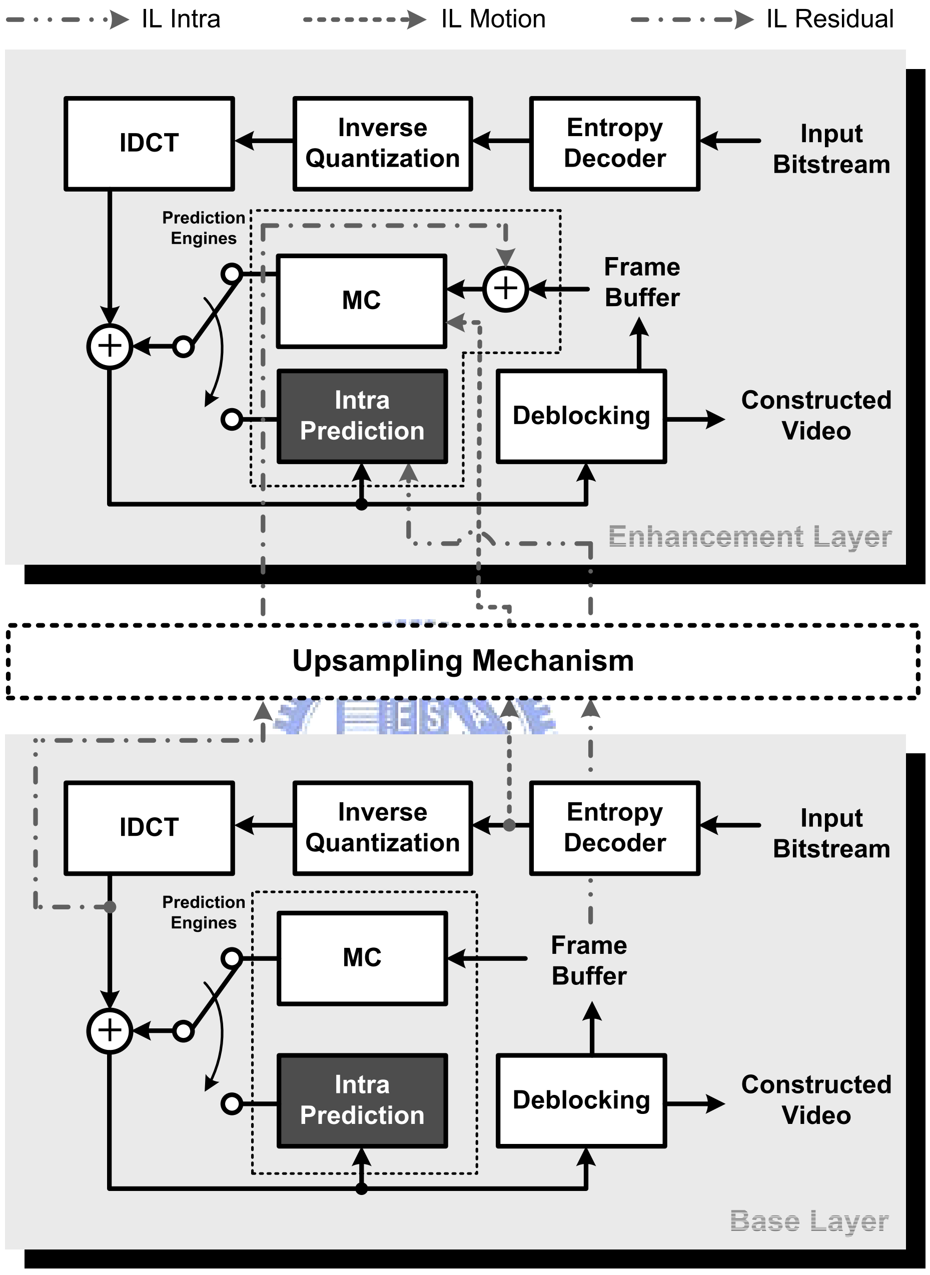


Figure 14: SVC decoder block diagram with two spatial layers.

The block diagram of SVC decoder is similar to Figure 3 in Section 2.1.2, but with some additional processes for enhancement layer decoding. When the base layer is decoding, the overall decoding flow is the same as H.264/AVC decoding flow. However, there are some additional processes chosen for decoding the enhancement layers. The block diagram of SVC decoder with two spatial layers is shown in Figure 14.

For spatial enhancement layers, SVC includes a new macroblock type which is involved by inter-layer prediction. For this macroblock type, only a residual signal but no additional side information such as intra prediction modes or motion parameters is transmitted. In intra prediction, the new prediction mode "*I_BL*" provides an additional prediction source for the scalable enhancement layer. When this new prediction mode I_BL is used in enhancement layer, the decoder will perform the following two major steps. First, it decodes the co-located blocks in the lower-resolution layer and applies a deblocking filter. Then, upsampling these samples of the lower resolution layer to form a prediction. On the other hand, the partitioning information of the enhancement layer macroblock, reference indices, and motion vectors are derived from the corresponding information of the co-located block in the reference layer for motion compensation. Furthermore, residual information of inter-picture coded MBs from a lower resolution layer can also be used for the prediction of the residual in the enhancement layer. The lower resolution layer residuals of the corresponding MBs are block-wise upsampled using a bilinear filter as a predictor for enhancement residual. Therefore, the difference between the residual of the enhancement layer which is obtained after motion compensation (MC), and the upsampled residual of a lower resolution layer is added before MC in enhancement layer. Finally, the video is reconstructed through the traditional process as H.264/AVC decoding flow.

2.2.3 Special Features in SVC

As mentioned in Section 2.2.2, SVC provides three special scalabilities, which are temporal scalability, spatial scalability, and quality scalability. Among these scalabilities, spatial scalability is the most special part in SVC. Hence, in the following section, we will focus on the spatial scalability, and introduce more detail.

2.2.3.1 Spatial Scalability

With the requirement that a single video bitstream should flexibly provide different spatial resolutions to satisfy various channel conditions and customers with different bitrates, computational, and power capabilities, spatial scalability has become an attractive feature in many video applications. SVC has been initially designed for progressive video. However, even if progressive video is becoming the favorite format for productions, broadcasting, and consumer equipments, interlaced video is still widely used in the video and will not disappear in the next few years. Therefore, interlaced video becomes an important requirement for SVC.

In the SVC standard, the lowest resolution video in a spatially scalable system is referred to as the *base layer*, and the higher resolution video is often referred to as the *enhancement layer*. The video bitstream is desired to be partitioned into lower resolution sequence which the base layer can decode independently to form a lower resolution video and the enhancement layer sequences which contain additional refinement data, and can be decoded to provide higher resolution videos. Processes that predict the data of enhancement layer from previously reconstructed data of a lower-resolution layer are referred to as *inter-layer* prediction processes, which is the major prediction mechanism that defined in SVC spatial scalability.

Furthermore, when the interlaced video is considered in SVC, the inter-layer prediction will be more complex than only support progressive video. For a SVC video coder with more than one spatial layer will encounter a total of 9 combinations of video formats between spatial layers, such as frame-frame, frame-field, frame-MBAFF, field-frame, field-field, field-MBAFF, MBAFF-frame, MBAFF-field, and MBAFF-MBAFF. However, not all of them are supported in SVC. Figure 15 shows the possible combinations of inter-layer prediction defined in SVC. It should be noticed that the field-MBAFF and MBAFF-field are not supported. Therefore, depending on the base layer and enhancement layer picture types, the decoding process also has some differences due to the neighboring information fetching.

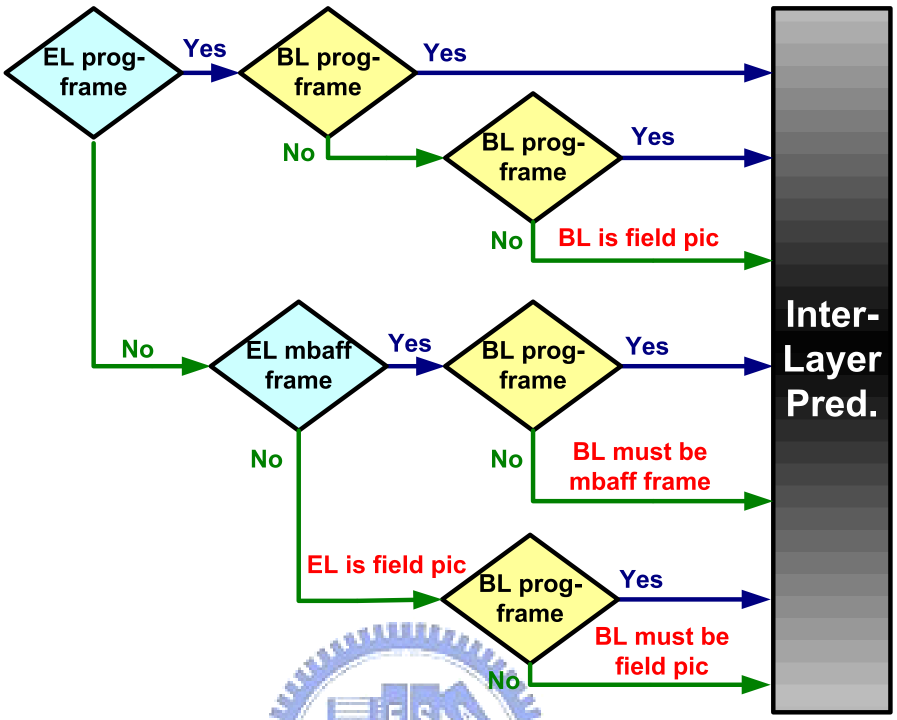


Figure 15: Possible inter-layer prediction process supported in SVC.

Besides, inter-layer prediction can be further partitioned into three cases: inter-layer intra prediction, inter-layer motion prediction, and inter-layer residual prediction. In the next section, we will focus on the inter-layer intra prediction to introduce the difference of intra prediction in H.264 and SVC.

2.2.4 Inter-Layer Intra-Prediction

For decoding an enhancement layer in SVC, if all 4x4 luma blocks of the enhancement macroblock correspond to intra-picture coded lower resolution layer blocks, the inferred macroblock type is considered to be "I_BL", a macroblock type that provides an additional prediction source for the scalable enhancement layer intra prediction, as shown in Figure 16. The decoding flow of this new prediction type will be described in Section 2.2.4.1.

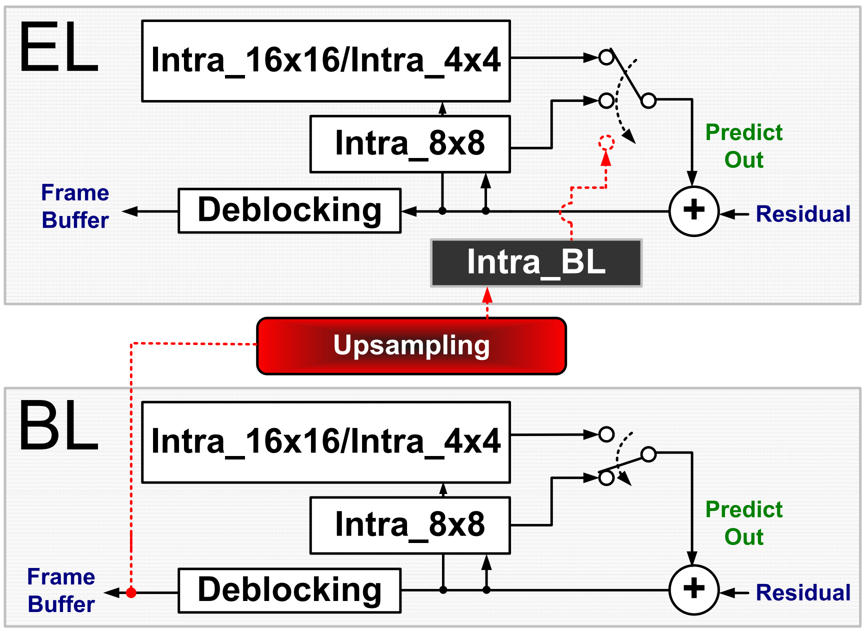


Figure 16: Intra prediction types in base layer and enhancement layer.

2.2.4.1 Intra_BL Decoding Processes

To decode this new type in enhancement layer, the block diagram of decoding process is described in Figure 17. The first step is to get the corresponding blocks in the lower resolution layer. After the reference blocks in the lower resolution layer are identified, these reconstructed samples are upsampled to the higher resolution layer. In the SVC design, the upsampling process for the luma samples consists of applying a separable four-tap poly-phase interpolation filter. The interpolation coefficient values for the filter are provided in Table 3. The chroma samples are also upsampled but with a different (simpler) interpolation filter which corresponds to bi-linear interpolation. The interpolation coefficients of this filter are also shown in Table 3.

The use of different interpolation filters for luma and chroma is motivated by complexity considerations. In prior standardized designs, the upsampling filter was only bi-linear for both luma and chroma, but resulting in significantly lower luma prediction quality. Therefore, the SVC standardizes different filters to luma and chroma samples.

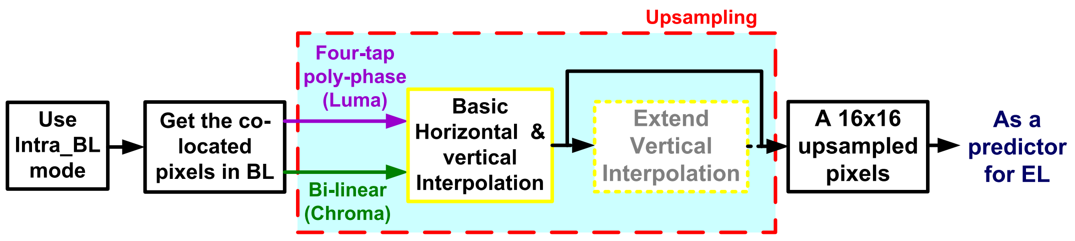

Upsampling
Use Intra_BL mode
Get the co-located pixels in BL
Four-tap poly-phase (Luma)
Bi-linear (Chroma)
Basic Horizontal & vertical Interpolation
Extend Vertical Interpolation
A 16x16 upsampled pixels
As a predictor for EL

Figure 17: Block diagram of Intra_BL decoding process.

Table 3: Interpolation coefficients for Luma and Chroma up-sampling.

Luma

Phase index	Interpolation filter coefficients			
	e[-1]	e[0]	e[1]	e[2]
0	0	32	0	0
1	-1	32	2	-1
2	-2	31	4	-1
3	-3	30	6	-1
4	-3	28	8	-1
5	-4	26	11	-1
6	-4	24	14	-2
7	-3	22	16	-3
8	-3	19	19	-3
9	-3	16	22	-3
10	-2	14	24	-4
11	-1	11	26	-4
12	-1	8	28	-3
13	-1	6	30	-3
14	-1	4	31	-2
15	-1	2	32	-1

Chroma

Phase index	Interpolation filter coefficients			
	e[-1]	e[0]	e[1]	e[2]
0	0	32	0	0
1	0	30	2	0
2	0	28	4	0
3	0	26	6	0
4	0	24	8	0
5	0	22	10	0
6	0	20	12	0
7	0	18	14	0
8	0	16	16	0
9	0	14	18	0
10	0	12	20	0
11	0	10	22	0
12	0	8	24	0
13	0	6	26	0
14	0	4	28	0
15	0	2	30	0

After upsampling, the SVC decoder also adds the residual difference information to refine the upsampled prediction as the same as in H.264. Finally, a deblocking filter that is similar to that of the ordinary H.264/AVC but with altered boundary strength calculations is applied to the decoded result.

It should be noticed that the up-sampling process in Figure 17 is consisted of two parts. One is basic interpolation and the other one is extended vertical interpolation. However, the extended vertical interpolation process is only applied to some combinations of picture type in base layer and enhancement layer, such as frame-MBAFF, MBAFF-frame, and field-frame due to the neighboring information considerations.

Chapter 3
Proposed Intra Prediction Engine with Data Reuse in H.264/AVC HP

In Chapter 1 and Chapter 2, the intra decoding process based on profile, picture type, and prediction source in H.264 and SVC is introduced. Therefore, we propose a H.264/SVC intra prediction engine which can support high profile and each inter-layer prediction type in single layer decoding and multi-layer layer decoding of H.264 and SVC, respectively.

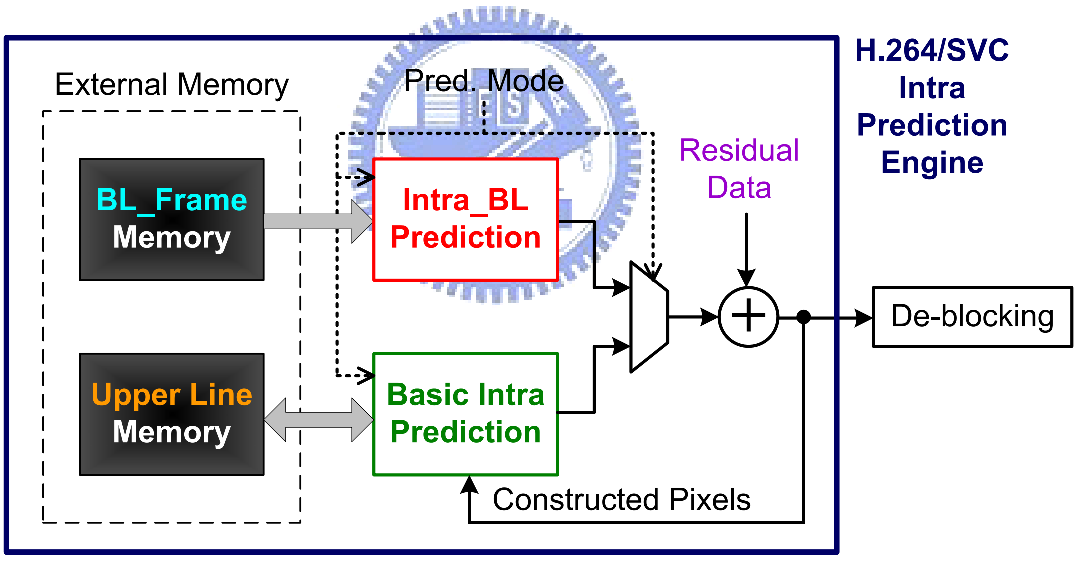


Figure 18: Block diagram of SVC intra prediction engine.

Figure 18 shows our proposed H.264/SVC intra prediction engine. The module “Basic Intra Prediction” is used to decode the traditional single layer intra prediction in H.264 or base layer intra prediction in SVC. The other module “Intra_BL Prediction” is designed to decode the new prediction type Intra_BL in enhancement layer of SVC.

In this chapter, the design of intra prediction with data reuse in H.264 high profile will be described. Another design for SVC inter-layer intra prediction will be further described in Chapter 4. To alleviate the starved bandwidth of intra compensation in high-definition video, we reuse the neighboring pixels and optimize the buffer size and access latency. In particular, a dedicated pixel buffer reuses neighboring pixel for realizing MB-adaptive frame-field (MBAFF) decoding in intra compensation. Moreover, a base-mode predictor is explored to optimize the area efficiency for reference sample filtering process (RSFP) in intra 8x8 modes. The slightly increased gates and SRAMs overhead are 10% and 7.5% as compared to the intra prediction design with main profile. Simulation results show that the proposed data-reused intra prediction module requires 14K logic gates and 688 bits SRAM, and operates on 100MHz frequency for realizing 1080HD video playback at 30fps.

3.1 Overview

As we know, H.264/AVC consists of three profiles which are defined as a subset of technologies within a standard usually created for specific applications. Baseline and main profiles are intended as the applications for video conferencing/mobile and broadcast/storage, respectively. Considering the H.264-coded video in high profile, it targets the compression of high-quality and high-resolution video and becomes the mainstream of high-definition consumer products such as Blu-ray disc. However, high-profile video is more challenging in terms of implementation cost and access bandwidth since it involves extra coding engine, such as macroblock-adaptive frame field (MBAFF) coding and 8×8 intra coding, for achieving high performance compression.

In the MBAFF-coded pictures, they can be partitioned into 16x32 macroblock pairs, and both macroblocks in each macroblock-pair are coded in either frame or field mode. As compared to purely frame-coded pictures, MBAFF coding requires two times of neighboring pixels size and therefore increases implementation cost. To cope with aforementioned problem, we propose neighboring buffer memory (include upper/left/corner) to reuse the overlapped neighboring pixels of an MB pair. Furthermore, we present memory hierarchy and pixel re-ordering process to optimize the overall memory size and external access efficiency. On the other hand, H.264 additionally adopts intra 8×8 coding tools for improving coding efficiency. It involves a reference sample filtering process (RSFP) before decoding a Luma *intra_8x8* block. These filtered pixels are used to generate predicted pixels of 8×8 blocks. Hence, the additional processing latency and cost are required, and they may impact the overall performance for the real-time playback of high-definition video. In this chapter, we simplify the RSF process via a base-mode predictor and optimize the processing latency and buffer cost. Compared to the existing design [16] without supporting intra 8×8 coding, this design only introduces area overhead of 10% and 7.5% of gate counts and buffer SRAM.

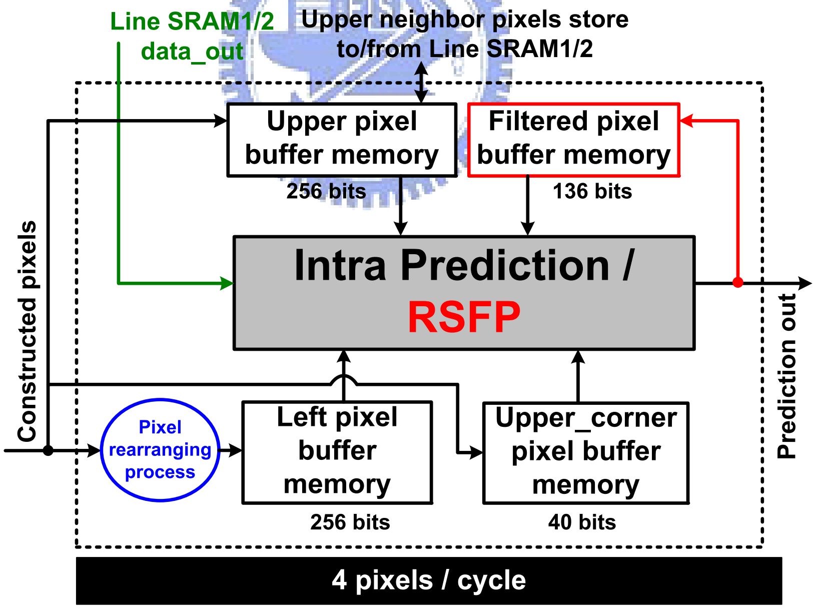


Figure 19: Block diagram of the proposed H.264 high-profile intra predictor.

Figure 19 shows the block diagram of the proposed H.264 high-profile intra compensation architecture. A pixel rearranging process, which is located on the bottom-left of Figure 19, is proposed to reduce the complexity of neighbor fetching when MBAFF coding is enabled. The signal, *Line SRAM1/2 data_out*, is directly connected to the intra prediction block for replacing the last set of upper buffer memory. As for 8×8 intra coding, a dedicated pixel buffer memory is used to store the filtered neighboring pixels and reuse the overlapped pixel data. According to the relations between Luma *intra_8x8* modes and numbers of filtered pixels which are needed in each mode, we minimize the number of stored pixels to 17 (i.e. 136 bits). The output of predicted pixel is interfaced to the filtered pixel buffer memory because RSFP is embedded in the intra prediction generator.

3.2 Memory Hierarchy

An architectural choice advocated for dealing with long past history of data is a memory hierarchy [18]. However, if the instantaneously used upper neighboring pixels are not in the neighboring information memory (i.e. a miss is occurring), the decoding process and pipeline schedule will be delayed and destroyed due to external SDRAM accessing. For more details, please refer [18]. In the intra prediction, it utilizes the neighboring pixels to create a reliable predictor, leading to dependency on a long past history of data. This dependency can be solved by storing upper rows of pixels for predicting current pixels but is a challenging issue in implementation cost and access bandwidth.

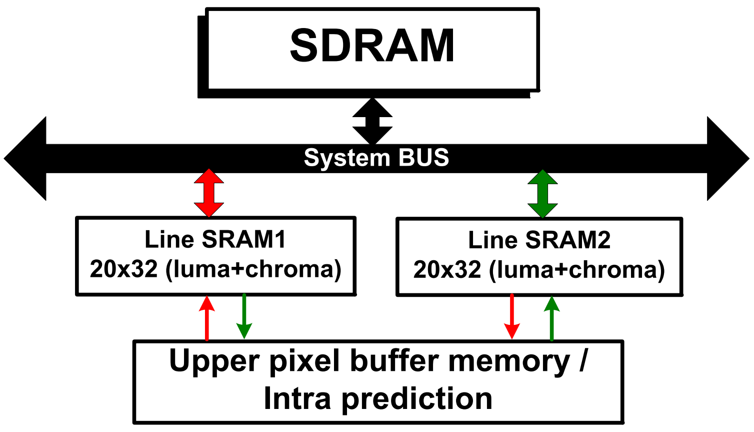


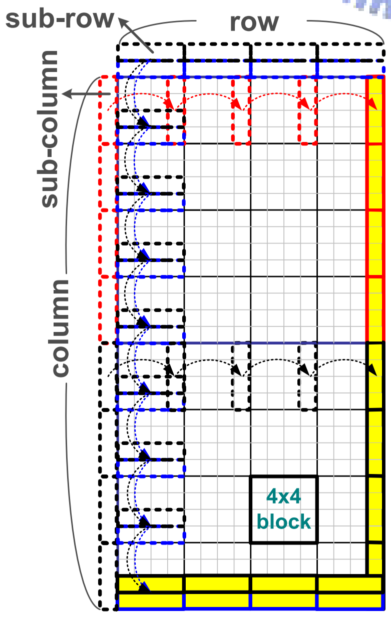
Figure 20: Memory hierarchy of H.264 high-profile intra predictor.

To optimize the introduced buffer cost and access efficiency, we use two internal Line SRAM1 and Line SRAM2 to store the Luma and Chroma upper line pixels, as illustrated in Figure 20. By the ping-pong mechanism, the upper neighboring pixels of current MB (or MB pair) are stored to one of them, and the other Line SRAM is used to store next MB of upper neighboring pixels. This memory hierarchy facilitates the internal Line SRAM size and the decoding pipeline schedule.

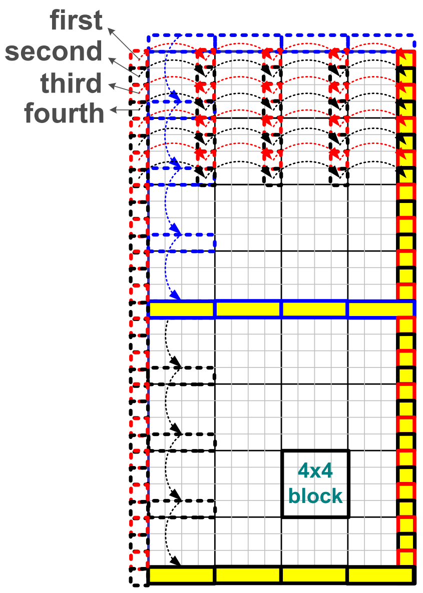
3.3 MBAFF Decoding with Data Reuse Sets

MBAFF is proposed to improve coding efficiency for interlaced video. However, it introduces longer dependency than conventional frame-coded picture. In this section, we analyze and realize it via upper, left, and corner data reuse sets (DRS) to reuse the pixels and improve the cost and access efficiency.

3.3.1 Upper Data Reuse Sets



(a)



(b)

Figure 21: The updated direction of upper/left buffer memory in (a) frame and (b) field mode MB pair.

For decoding an MBAFF-coded video, upper buffer memory is used to store the constructed upper pixels of current MB pair. These upper buffers are updated with the completion of prediction process on every 4×4 block. For each updated sub-row(s), they can be reused by the underside 4×4 blocks. According to the different prediction modes of MB pair, the upper buffer will store data from different directions. If current MB pair is frame mode, it only needs to load one row of upper buffer (16 pixels) at first, and when a 4×4 block is decoded, updating the two sub-rows in two rows (8 pixels) of upper buffer from top to down at one time, as illustrated in Figure 21(a). Finally, the new pixels in two rows will be stored to Line SRAM in chorus when MB pair is decoded.

In Figure 21(b), a field-coded MB pair needs to load two rows of upper buffer (32 pixels), two times of frame-coded MB pairs. Then, only one sub-row of upper buffer memory will be updated when a 4×4 block is decoded. Finally, the new pixels in one row of upper buffer are stored back to Line SRAM when the top MB is decoded and another row of upper buffer are stored back to Line SRAM when the bottom MB is decoded. However, considering the fifth 4×4 block, it still needs a sub-row of upper buffer to predict, as shown in Figure 22. In order to reduce the upper buffer memory size, the Line SRAM *data_out* is directly used. The only penalty to this scheme is that the Line SRAM *data_out* must hold the value until fifth 4×4 block is decoded.

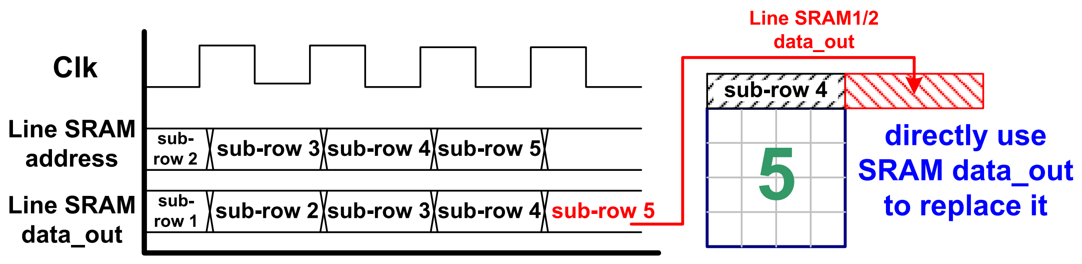


Figure 22: Line SRAM *data_out* replaces the last sub-row of upper buffer.

3.3.2 Left Data Reuse Sets

The updated direction of the left buffer is similar to that of the upper one. The direction ranges from left to right. When the left buffer is located on the right hand side of MB pair, the next MB pair can reuse these new pixels for the following prediction procedures. However, when the modes of current and previous MB pairs are different, the left neighbors of a 4x4 block will become complicated. To reduce computational complexity of this intra predictor, pixel rearranging process is exploited. If current MB pair is frame mode, each sub-column of left buffer will be updated when each 4x4 block is decoded, as shown in Figure 22(a). On the other hand, if current MB pair is field mode, first and third buffers in each sub-column of left buffer will be updated when each 4×4 block is decoded in the top MB, as shown in Figure 22(b). Second and fourth buffers in each sub-column of left buffer will be updated when each 4×4 block is decoded in bottom MB. Hence, we only need to consider what the mode current MB pair is, instead of handling four coding modes for the combination of current and previous MB pairs, and therefore the complexity can be reduced.

3.3.3 Corner Data Reuse Sets

Using corner buffer memory can efficiently reuse the upper left neighboring pixels. We change the positions of corner buffer from left [16] to top. Therefore, the total corner buffer size can be reduced by 38% (i.e. 64bits → 40bits, because the MB number in horizontal is less than that of vertical MB pair). In particular, Figure 23 shows the updating directions of corner buffer. However, because the upper neighboring pixels will be either the last row or the row prior to the last row in upper MB pair, hence the first corner of current MB pair has two processing states: reuse and reload. The first corner is reused when 1) the mode of current MB pair is identical to that of previous (left) MB pair or 2) before decoding the bottom MB of frame-coded MB pair. On the other hand, the first corner is reloaded when 1) the current MB pair has the different modes as previous (left) MB pair or 2) before decoding the bottom MB of field-coded MB pair.

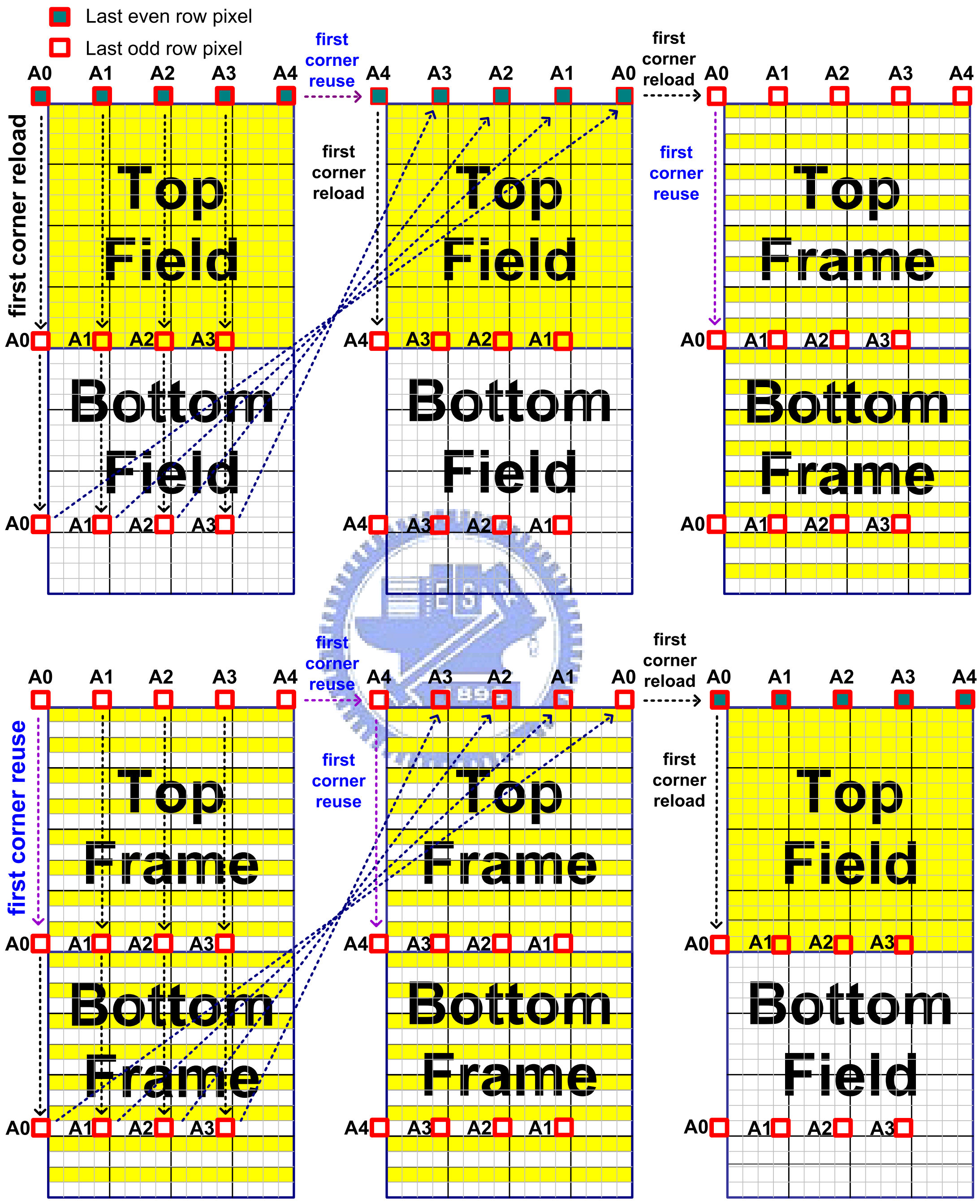


Figure 23: The updated direction of corner pixel buffers.

In summary, using neighboring buffer memory and their different directions of updates according to different modes of MB pair can reuse the neighboring pixels and improve the access efficiency. The associated pipeline structure of MBAFF decoding is shown in Figure 24. We can see that during a MB pair decoding process, the interaction between buffers and Line SRAM can be completed easily and efficiently, and the communication between another Line SRAM and external SDRAM can be done at the same time. It should be noticed that Line SRAM1/2 *data_out* must be held until the fifth 4x4 block is decoded.

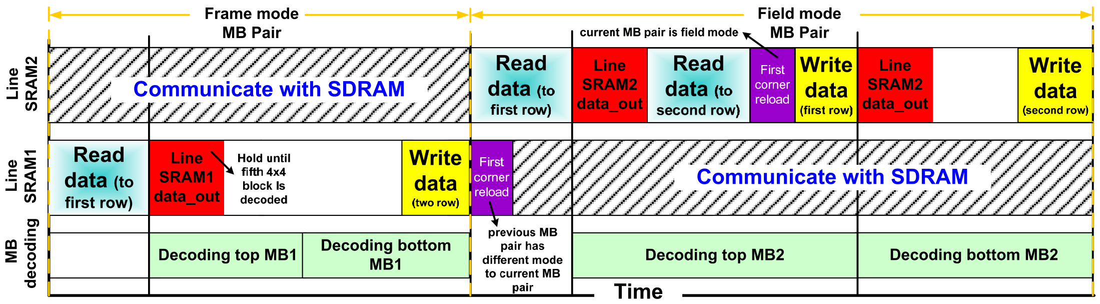


Figure 24: The pipeline scheme of MBAFF decoding.

3.4 *Intra 8x8 Decoding with Modified Base-Mode Intra Predictor*

Luma *intra_8x8* is an additional intra block type supported in H.264 high profile. Before decoding an *intra_8x8* block, there is an extra process that is different from *intra_4x4* and *intra_16x16*, which called reference sample filtering process (RSFP). Original pixels will be filtered first, and then using these filtered pixels to predict subsequent 8×8 blocks.

3.4.1 Filtered Neighboring Buffer Analyze

For an *intra_4x4* and *intra_8x8* block, 13 neighbors and 25 filtered neighbors are needed, respectively. However, according to the Luma *intra_8x8* modes, the maximum number of filtered neighbors is 17, as illustrated in Table 4. Hence, only 17 (i.e. 136 bits) filtered pixels of a 8x8 block need to be stored in our filtered pixel buffer memory instead of storing all of the 25 filtered neighbors.

Table 4: Number of filtered pixels actually needed in intra 8x8 modes.

Prediction Modes of Intra_8x8	# of filtered neighbors
0 (Vertical)	8
1 (Horizontal)	8
2 (DC)	0,8,16
3 (Diagonal down left)	16
4 (Diagonal down right)	17
5 (Vertical right)	17
6 (Horizontal down)	17
7 (Vertical left)	16
8 (Horizontal up)	8

3.4.2 Reference Sample Filter Process (RSFP)

In the *intra_4x4* process, the prediction formula of each mode except DC mode has the same form: *prediction_out = (A+B+C+D+2) >> 2*. For a four-parallel intra pixel generator, a suitable architecture design is proposed in [14], as shown in Figure 25.

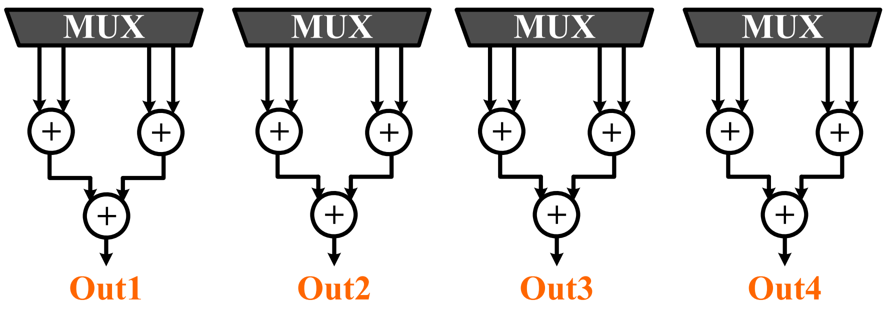
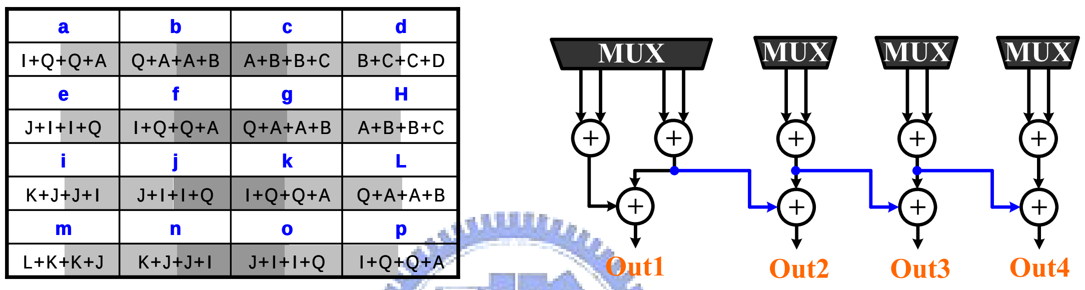


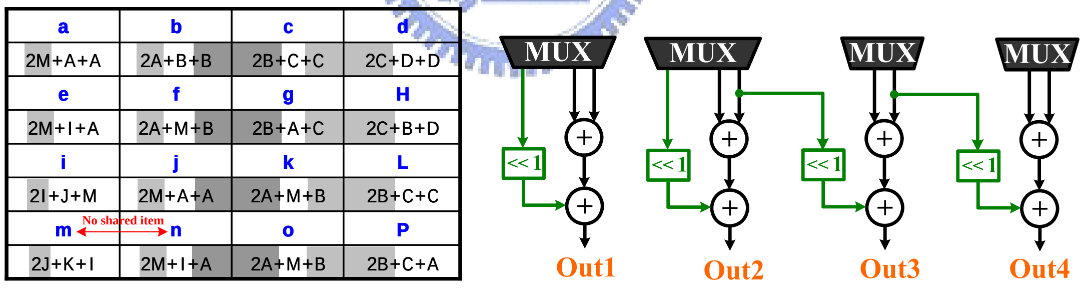
Figure 25: Intra predictor in [14].

However, if we analyze the relationship between four output pixels in each mode, some adders can be eliminated due to the shared items, so that the hardware architecture can be reduced [19], as described in Figure 26. This shared based intra predictor can predict almost every *intra_4x4* mode in one cycle. But there still exist exceptions, such as vertical-right. For example, in the vertical-right prediction mode, there is no shared item between prediction pixels m and n, as illustrated in Figure 27. Therefore, the modified intra predictor is proposed. Compared with the share-based [14]-[16][19] intra prediction generator, the proposed base-mode intra predictor not only reduces area cost (due to elimination of four adders) but also guarantees that four predicted pixels will be generated in one cycle of each *intra_4x4* modes.



a	b	c	d
I+Q+Q+A	Q+A+A+B	A+B+B+C	B+C+C+D
e	f	g	H
J+I+I+Q	I+Q+Q+A	Q+A+A+B	A+B+B+C
i	j	k	L
K+J+J+I	J+I+I+Q	I+Q+Q+A	Q+A+A+B
m	n	o	p
L+K+K+J	K+J+J+I	J+I+I+Q	I+Q+Q+A

Figure 26: Intra predictor in [19].



a	b	c	d
2M+A+A	2A+B+B	2B+C+C	2C+D+D
e	f	g	H
2M+I+A	2A+M+B	2B+A+C	2C+B+D
i	j	k	L
2I+J+M	2M+A+A	2A+M+B	2B+C+C
m	n	o	P
2J+K+I	2M+I+A	2A+M+B	2B+C+A

Figure 27: Modified base-mode intra predictor.

In particular, we use this base mode predictor to generate the four predicted pixels in one cycle. In the RSFP, the form of formula is identical to that in *intra_4x4*, and also can be rewritten to the same form *filtered_out* = *(A+2B+C+2) >> 2*. Hence, we can share the hardware resource to generate filtered pixels, as shown in Figure 28. Notice that an additional process, neighbor distribution, is needed to apply in *intra_8x8* process because we only store 17 filtered pixels instead of 25.

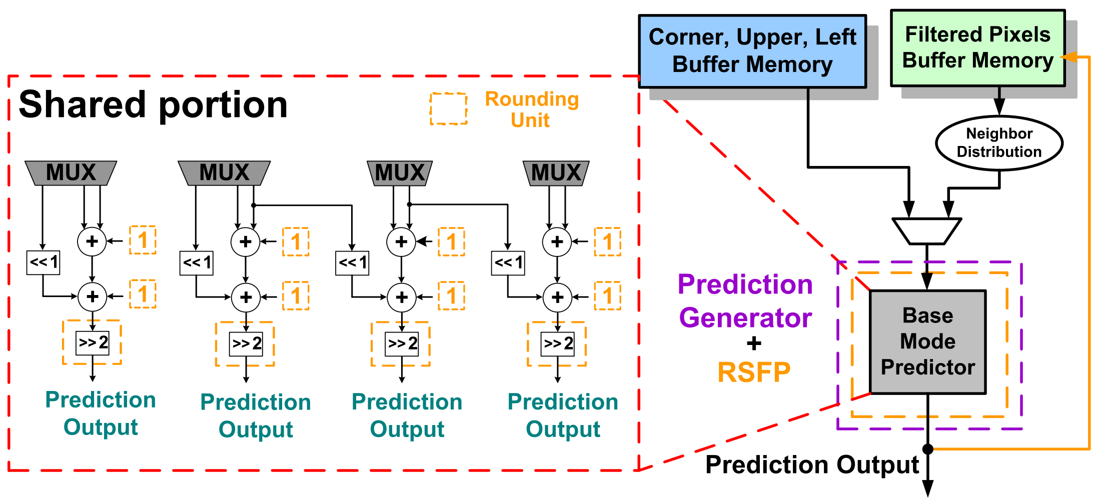


Figure 28: Architecture of an intra 8×8 decoding module.

3.4.3 Latency Reduction

In a four-parallel intra prediction module, the latency of decoding an 8x8 block will be increased to 0~5 cycles according to the different modes of 8×8 blocks. In order to reduce the latency penalty, we reserve filtered pixels when the mode of the first/third 8x8 block in a MB is equal to 3 or 7 and second/fourth 8x8 block in the same MB is equal to 0, 2 (if upper is available), or 3~7 (i.e. if so, the value N = 6. Otherwise, N = 0). During this case, when the first/third 8x8 block in a MB is decoded, these overlapped filtered pixels are directly used to predict second/fourth 8×8 block. To clarify the extra latency, Eq. (1) lists the decoding extra latency in an *intra_8x8* MB, and Table 4 summarizes the # of filtered pixels in each 8×8 intra coding mode (i.e. the value of M).

$$\textit{Extra latency} = \underline{\underline{\left\lceil \frac{M}{P} \right\rceil}} + \underline{\underline{\left\lceil \frac{M-N}{P} \right\rceil}} + \underline{\underline{\left\lceil \frac{M}{P} \right\rceil}} + \underline{\underline{\left\lceil \frac{M-N}{P} \right\rceil}}$$

8x8 block: first second third fourth (1)

$$\text{, where } \begin{cases} M = 0, 8, 16, \textit{or } 17 \\ N = 0 \textit{ or } 6 \\ P = 4 \end{cases}$$

In particular, we list some examples to clarify the processing behavior of an intra 8×8 block in Figure 29. If the modes of first and second 8×8 blocks are 3 (diagonal bottom left) and 7 (vertical left) or 3 (diagonal bottom left) and 4 (diagonal bottom right), only 10 and 11 pixels are needed to be filtered while decoding the second 8×8 block, respectively, as described in Figure 29.

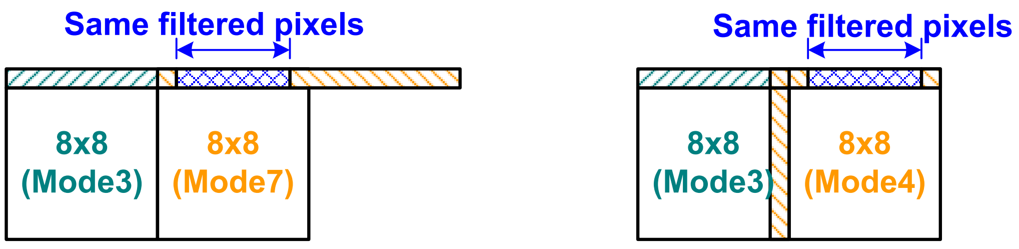


Figure 29: Behavior of shared filter.

3.5 *Simulation Result and Comparison*

To enhance system performance, our proposal is designed to optimize area, buffer size, and latency. We use two 0.64kb Line SRAMs which are connected to a 32-bits system bus to make decoding pipeline simple, and 0.688kb SRAM to store reused neighboring pixels.

Table 5: Average cycles needed for decoding a MB in different video sequences.

Test Video Sequence	Intra Prediction @ BL [17]	Proposed Intra Prediction @ HP	Cycle Overhead
Foreman	342.68	355.81	3.8%
Grandma	275.63	285.28	3.5%
Suzie	294.90	307.28	4.2%

Table 5 shows the average cycles for decoding an I-MB in different video sequences of our proposed design for 30fps HD1080 video format at working frequency of 100MHz with

MBAFF and Luma *intra_8x8*. The overhead of latency is less than 5% compared to preliminary architecture [17]. The overall area and buffer memory size for supporting H.264 BP/MP/HP are 14063 gates in UMC 0.18um technology and 688 bits, as shown in Table 6. The overheads for supporting Luma *intra_8x8* are 10% and 7.5% compared to [16].

Table 6: Comparison results.

	Chen et al. [16]	**Proposal**	**Overhead**
Profile	MP	HP	
Process	0.18um	0.18um	
Working Frequency	87M	100M	
Gate Count	12785	14063	10%
Memory (bit)	640	688	7.5%

Chapter 4
Proposed Power Efficient Inter-Layer Intra Prediction Engine in SVC

Scalable Video Coding (SVC) extension of the H.264/AVC is the latest standard both in video coding. It has achieved significant improvements in coding efficiency with an increased degree of supported scalability relative to the scalable profiles of prior video coding standards. In order to improve coding efficiency in comparison to simulcasting different spatial resolutions, additional so-called *inter-layer prediction* mechanisms are incorporated in SVC.

For the intra frame decoding, *inter-layer intra prediction* will be considered in SVC. When the prediction signal of the enhancement layer macroblock is obtained by *inter-layer intra-prediction*, the new prediction type called *Intra_BL* will be used (for which the corresponding reconstructed intra-signal of the co-located 8x8 submacroblock in reference layer is upsampled). In other words, the intra prediction types which are *intra_4x4*, *intra_8x8*, *intra_16x16* and *intra_BL* in enhancement layer are more than traditional H.264 intra prediction. However, the decoding process of *Intra_BL* is more complex than the other intra prediction modes. It is more like motion compensation, which needs interpolation process to generate the predicted pixels. The difference between them is that the interpolation coefficients in intra interpolation are not fixed. Therefore, complexity, processing time, and power consumption will become the major problems to decoder.

In this chapter, we propose an architecture design of high profile inter-layer intra prediction engine in SVC. It supports the new decoding type called *Intra_BL* and 7 possible picture type combinations between spatial layers in intra prediction. Besides, we also propose the area efficient interpolators design which can save 26% hardware area compared to direct implement. Moreover, we further propose a power efficient design, including memory hierarchy improvement and computational complexity reduction, which can save 46.43% total power consumption compared to our preliminary design.

4.1 Overview

In Chapter 3, the proposed SVC intra prediction engine is illustrated in Figure 18, and the basic intra prediction which is used to decode the traditional H.264 intra block for high profile video stream is also described. Another prediction module, Intra_BL prediction, will be described in this chapter. Besides, the power efficient proposal based on our original Intra_BL prediction will also be further described in Section 4.5.

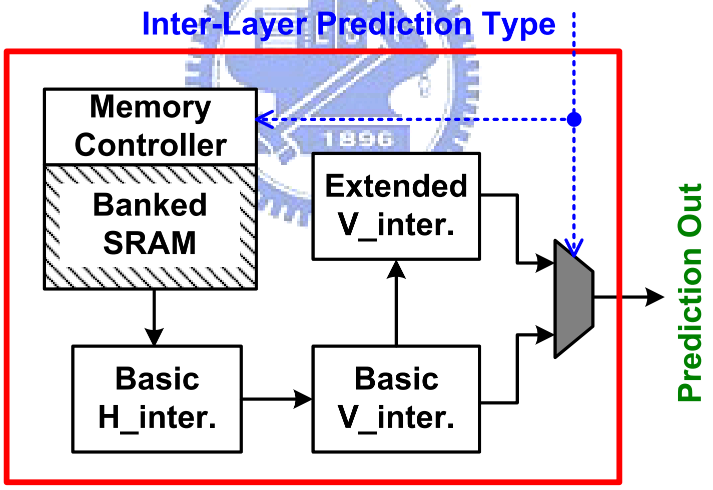


Figure 30: Proposed Intra_BL prediction decoding block diagram.

Figure 30 shows our proposed Intra_BL prediction decoding block diagram. We use a banked SRAM in our Intra_BL module to reduce the external SDRAM access times, and enhance the reuse of prediction source pixels. As mentioned in Section 2.2.3.1, there are total of 7 picture type prediction combinations between spatial layers. Hence the multiplexer and memory controller are applied to select the corresponding decoding flow and locations of prediction source pixels in the internal banked SRAM, respectively. The main interpolation architecture designs of basic horizontal interpolation (Basic H_inter.), basic vertical interpolation (Basic V_inter.), and extended vertical interpolation (Extended V_inter.) will be illustrated in Section 4.3.

4.2 System Power Modeling

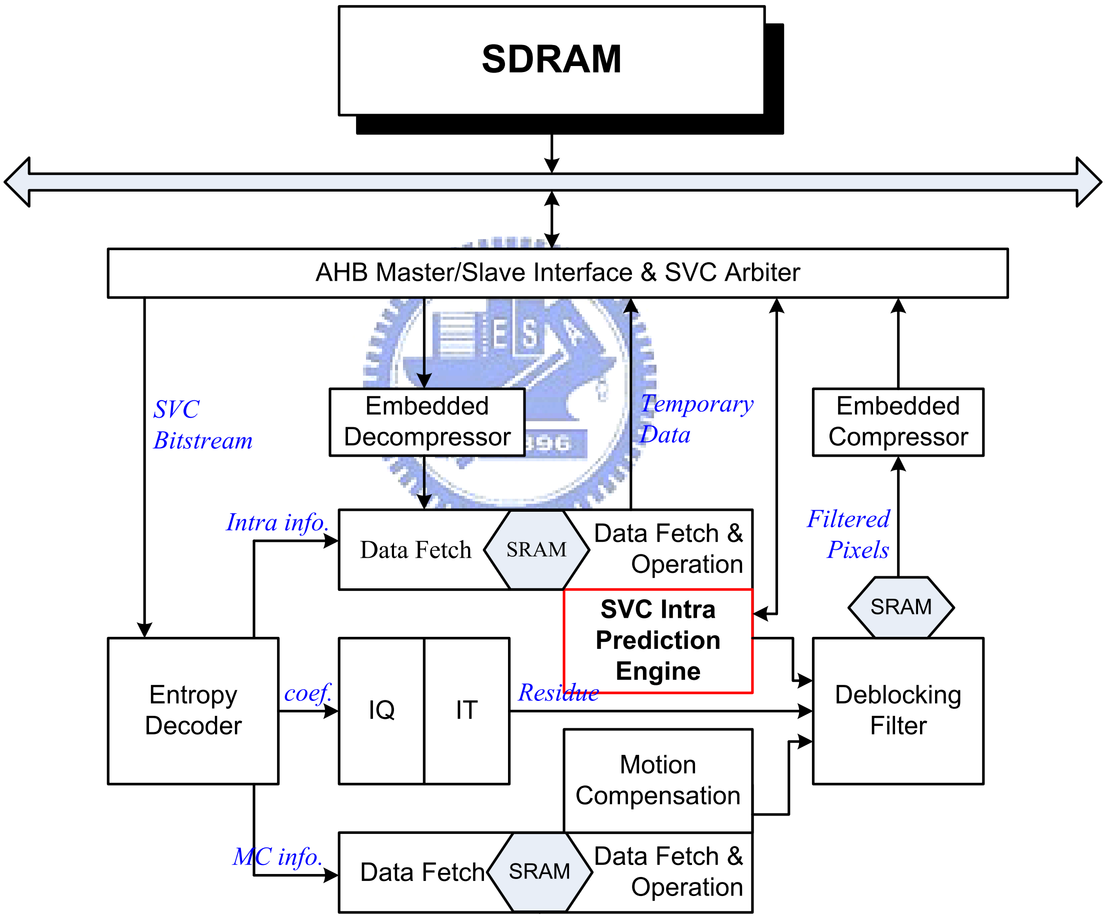


Figure 31: SVC decoder system block diagram.

The power consumption of a video coder system can be considered as a summation of on-chip and off-chip memory power. For the off-chip power consumption, during the video bitstream decoding, there are lots of data read/write from/to external SDRAM through a system bus, such as entropy decoder, motion compensation, intra prediction, and deblocking filter in H.264. However, when the new prediction type Intra_BL is taken into account in SVC, the intra prediction becomes one of the most major power consumption parts in the system due to the large data fetching from external SDRAM, as shown in Figure 31. Therefore, that will generate much more off-chip power consumption in SVC.

Consequently, we try to model the power consumption between external SDRAM and internal SRAM, choosing a most suitable way to our overall system. As for off-chip memory, the power modeling becomes more complicated. Not only data access but also I/O and background power should be concerned (see Eq. (2)).

$$P_{total} = P_{on-chip} + P_{off-chip} = P_{on-chip} + (P_{access} + P_{IO} + P_{BG}) \tag{2}$$

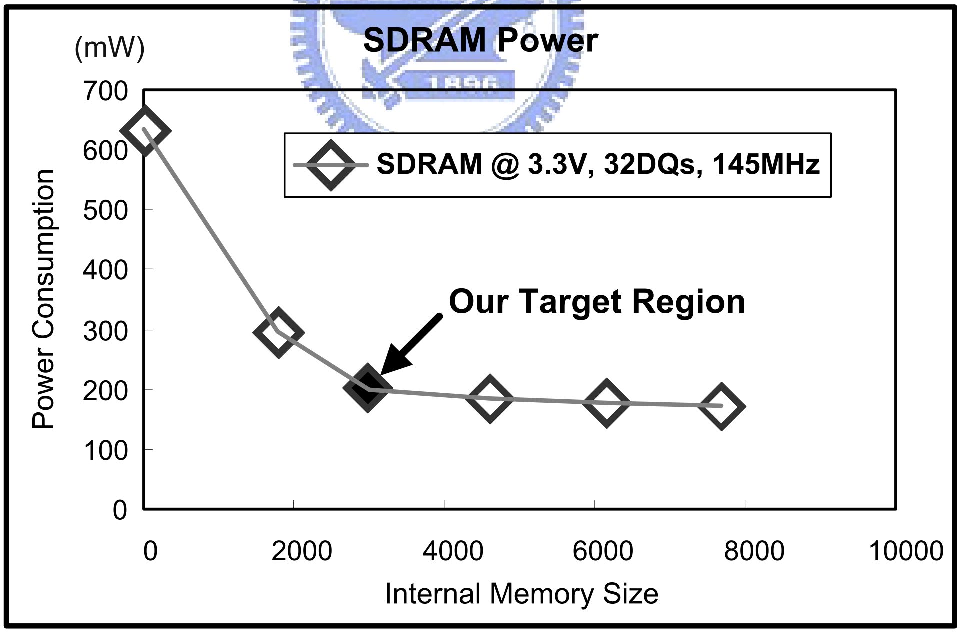


Figure 32: Analysis of power dissipation on external SDRAM and internal SRAM.

Specifically, we adopt Micron’s system-power calculator [20] to model SDRAM power. To estimate the power consumption, we exploit memory size and power calculator as the

SRAM and SDRAM power indexes. An observation is that the curve between internal SRAM and external DRAM power consumption is shown in Figure 32. The Micron's SDRAM model is chosen with supply voltage = 3.3V, ^tCK = 7ns, DQ = 32, and CK Frequency = 145 MHz. As a result, a better compromise can be selected by the marked region in Figure 32, since it achieves smaller SRAM size as well as SDRAM power dissipation. Therefore, in our proposed Intra_BL prediction module (see Figure 30), the size of banked SRAM is chosen according to the analysis of SDRAM and SRAM before.

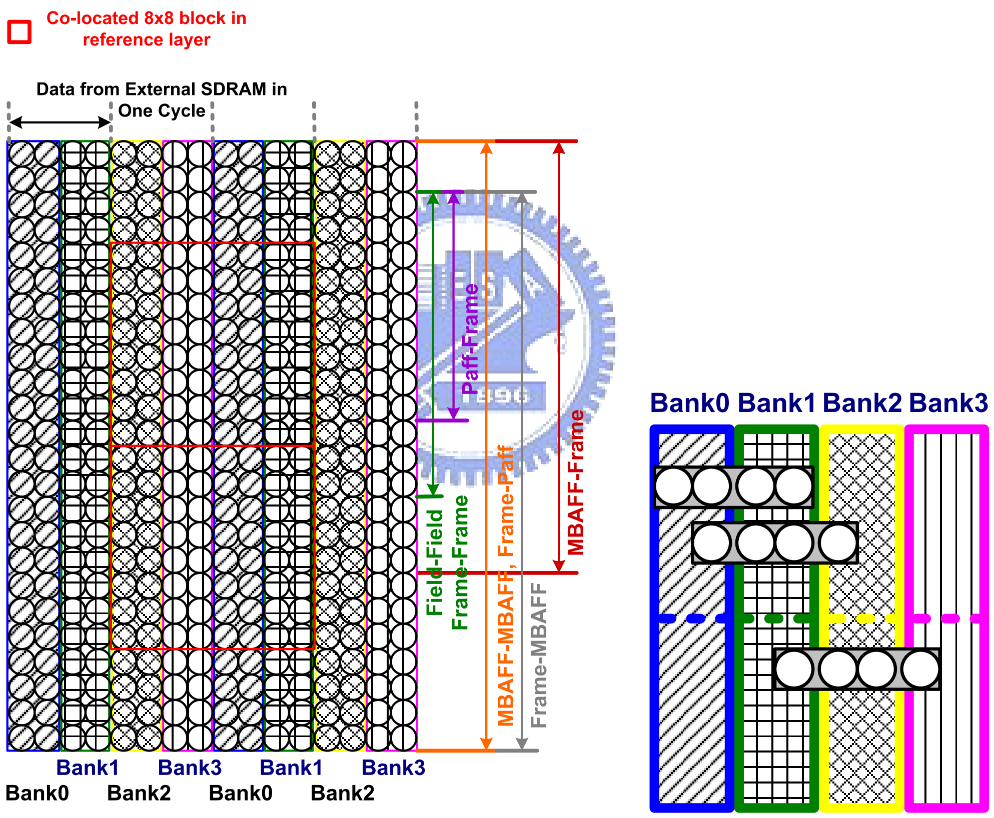


Figure 33: Banked SRAM and required region for different picture type predictions between spatial layers.

Figure 33 shows our internal banked SRAM used in Intra_BL prediction module. The total size of this banked SRAM is 3072 bits. We separate this SRAM to four banks, and each bank has 24 entries and 16 bits word length (i.e. 2 pixels). In this way, when the required reference pixels are stored into banked SRAM, fours pixels can be fetched in one cycle and interpolated, as shown in the right side of Figure 33. It should be noticed that we also mark the required pixel region in reference layer for different prediction types between spatial layers. However, due to the position of stored reference pixels in the external SDRAM, the first bank and last bank in Figure 33 are used to eliminate the redundancy and waste during the data reading from SDRAM. Moreover, this kind of partition is also convenient to update the pixel source inside the SRAM. When the current MB (or MB pair) is decoded, we update the first half part in each bank, and reuse another half part to be the new reference prediction source for the next MB (or MB pair). The corresponding update process is illustrated in Figure 34. By applying this banked SRAM, it is not only reduce the system power consumption but also efficient to data fetching and update process.

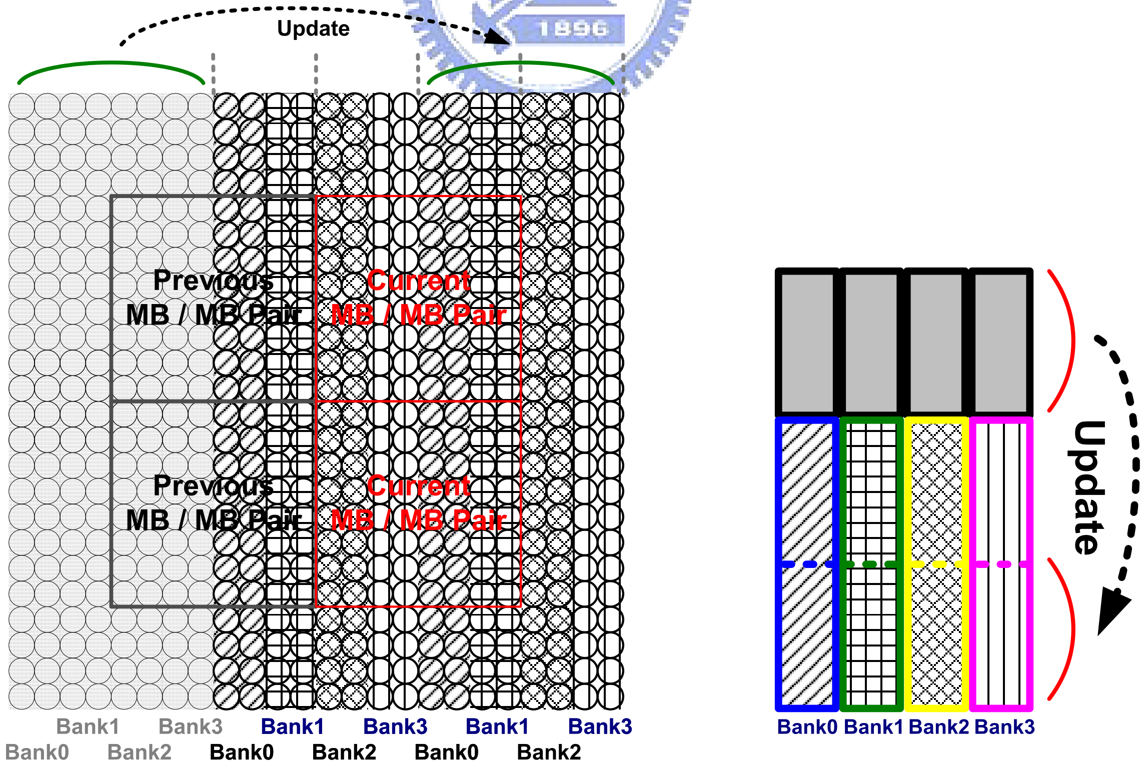


Figure 34: Updating process of banked SRAM.

4.3 Area Efficient Interpolator Design

The main interpolators in Intra_BL decoding process are basic interpolator and extended vertical interpolator. The decoding process of *Intra_BL* is illustrated in 2.2.4.1 previously. For upsampling the luma component, one-dimensional 4-tap FIR filters are applied horizontally and vertically. The chroma components are upsampled by using a simple bilinear filter. First, one of the coefficient sets in Table 3 will be chosen according to the *phase_idx* signal (i.e. the interpolation coefficients are not fixed) and then multiplying these coefficients by 4 (or 2, according to luma or chroma) reference pixels. Finally, add and shift these scaled pixels to generate one prediction out, as described in Eq. (3) and Eq. (4).

$$\begin{aligned} L\_Pred\_out1 \text{ (Luma)} = {} & L\_coef1 * L\_ref\_pixle1 + L\_coef2 * L\_ref\_pixle2 \\ & + L\_coef3 * L\_ref\_pixle3 + L\_coef4 * L\_ref\_pixle4 \end{aligned} \tag{3}$$

$$C\_Pred\_out1 \text{ (Chroma)} = C\_coef1 * C\_ref\_pixle1 + C\_coef2 * C\_ref\_pixle2 \tag{4}$$

The interpolation process mentioned above is referred to basic interpolation. Another interpolation process is called extended vertical interpolation process, which uses the outputs of basic interpolation to further generate the vertical correlative pixels in some cases of prediction types between spatial layers, as described in Eq. (5) and Eq. (6).

$$\begin{aligned} L\_Pred\_out1 \text{ (Luma)} = {} & (-3) * L\_B\_int\_out1 + 19 * L\_B\_int\_out2 \\ & + 19 * L\_B\_int\_out3 + (-3) * L\_B\_int\_out4 + 16 \end{aligned} \tag{5}$$

$$C\_Pred\_out1 \text{ (Chroma)} = C\_B\_int\_out1 + C\_B\_int\_out2 + 1 \tag{6}$$

4.3.1 Hybrid Basic Interpolator Design

Our proposed architecture, as shown in Figure 35, is divided into three parts, which are coef_generator, pixel_shifter, and scaling_engine. To be noticed, the bi-linear filter which is used for chroma sample is embedded in our architecture to reduce the area cost.

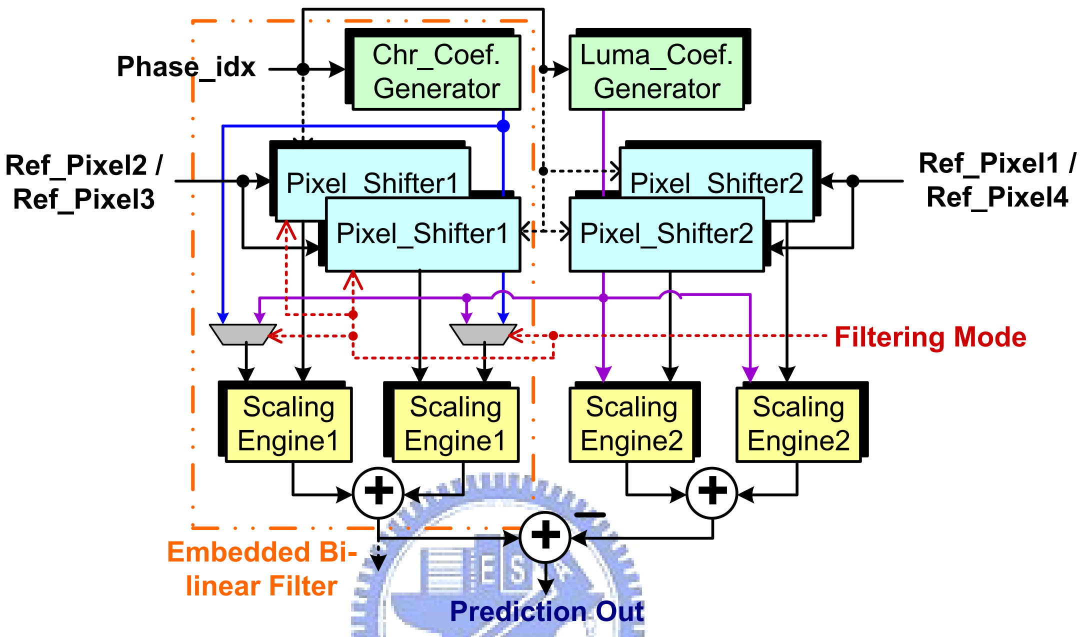


Figure 35: Proposed architecture of basic interpolator.

4.3.1.1 Coef_generator

For the chroma interpolation coefficient sets, if the coefficients are translated into binary form in Table 7, we can notice that both of the coef1 and coef2 have highly relative to the *phase_idx*. The first coefficient coef1 is the left shift of *phase_idx*, and the second coefficient coef2 is the 2's compliment and shift of *phase_idx*. Hence, we can easily use these relationships to construct our hardware architecture, as shown in Figure 36. On the other hand, the luma table has less relationship than chroma. However, we can still use some skills to reduce the area. In Table 8, if we separate the table from the ninth coefficient set, each set has the same coefficients to the symmetrical one except the coef2 and coef3 should be exchanged, so do coef1 and coef4. Therefore, taking the 2's complement of the *phase_idx* to be a control signal, we can easily merge the last 7 coefficient sets to the first 7 coefficient sets. Furthermore, the positive coefficients are used in the coef1 and coef4 instead of original negative representations. Then we separate the positive values (left part in Figure 35) and negative values (right part in Figure 35) to the two sides, and let the subtraction to the end of interpolation calculation (i.e. the last adder in Figure 35). So that before the last adder, all the wires can uses less bits due to further reduce the area cost. The luma_coef generator architecture is illustrated in Figure 37. The luma_coef set2~4 are just like set1, only the coefficients are different.

Table 7: Chroma Table in Binary Form.

Phase_idx	Chr_Coef1	Chr_Coef2
0000	000000	100000
0001	000010	011110
0010	000100	011100
0011	000110	011010
0100	001000	011000
0101	001010	010110
0110	001100	010100
0111	001110	010010
1000	010000	010000
1001	010010	001110
1010	010100	001100
1011	010110	001010
1100	011000	001000
1101	011010	000110
1110	011100	000100
1111	011110	000010

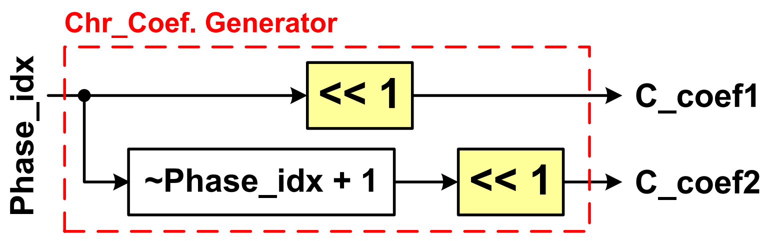


Figure 36: The architecture of chroma coef_generator.

Table 8: Luma Table in Binary Form.

Phase_idx	Lu_Coef1 (use positive)	Lu_Coef2	Lu_Coef3	Lu_Coef4 (use positive)
0000	000000	100000	000000	000000
0001	000001	100000	000010	000001
0010	000010	011111	000100	000001
0011	000011	011110	000110	000001
0100	000011	011100	001000	000001
0101	000100	011010	001011	000001
0110	000100	011000	001110	000010
0111	000011	010110	010000	000011
1000	000011	010011	010011	000011
1001	000011	010000	010110	000011
1010	000010	001110	011000	000100
1011	000001	001011	011010	000100
1100	000001	001000	011100	000011
1101	000001	000110	011110	000011
1110	000001	000100	011111	000010
1111	000001	000010	100000	000001

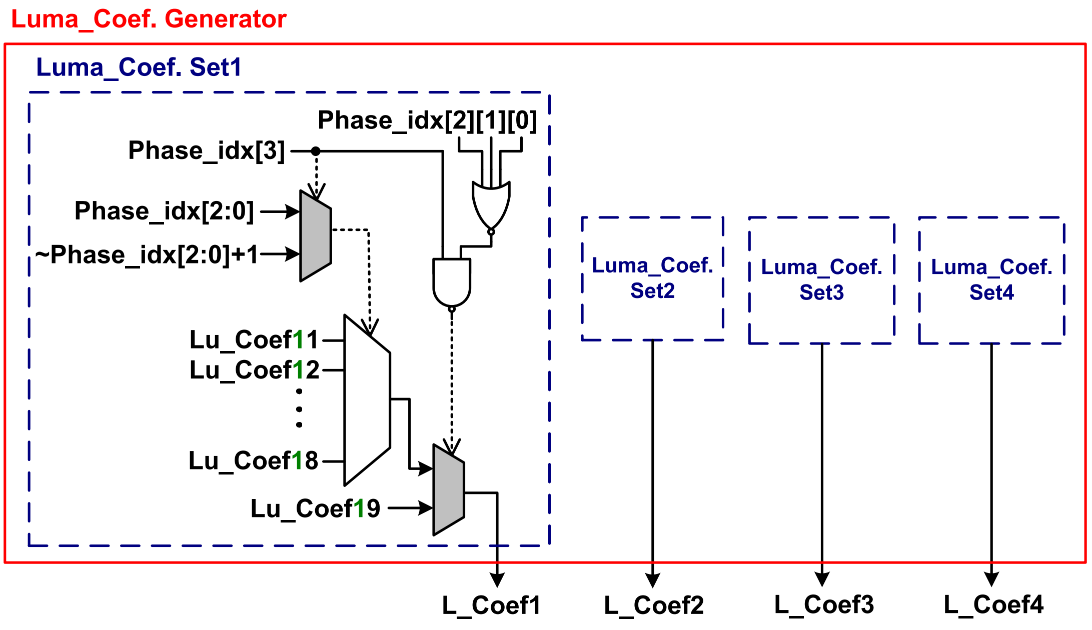
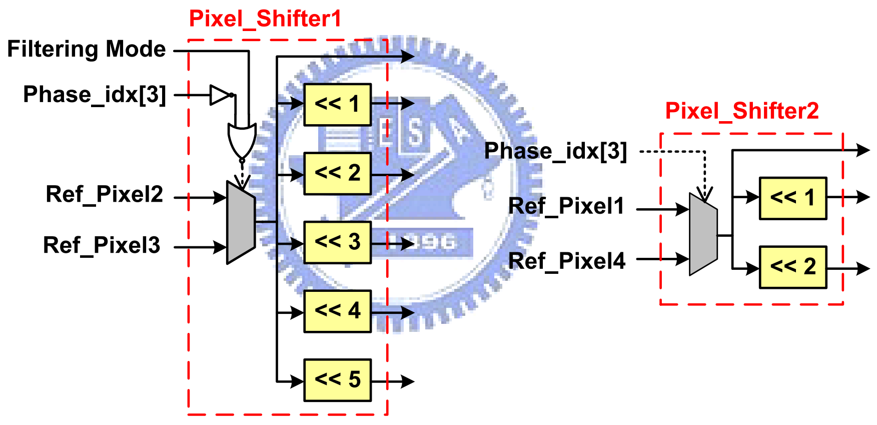


Figure 37: The architecture of luma coef_generator.

4.3.1.2 Pixel_shifter

In order to replace the multipliers, pixel_shifter1 and pixel_shifter2 are used to generate the scaled units of reference pixels. As shown in Figure 38 (a) and Figure 38 (b), pixel_shifter1 and pixel_shifter2 generate six scaled units for chr_coef1, chr_coef2, lu_coef2 and lu_coef3, and three scaled units for lu_coef1 and lu_coef4, respectively. In particular, lu_coef1 and lu_coef4 can use only three scaled units due to the positive representations. Further, the multiplexer is needed for changing the source pixels to match the right coefficients when the *phase_idx* is selected to the last half coefficient sets, which are merged to the upper coefficient sets in luma table.

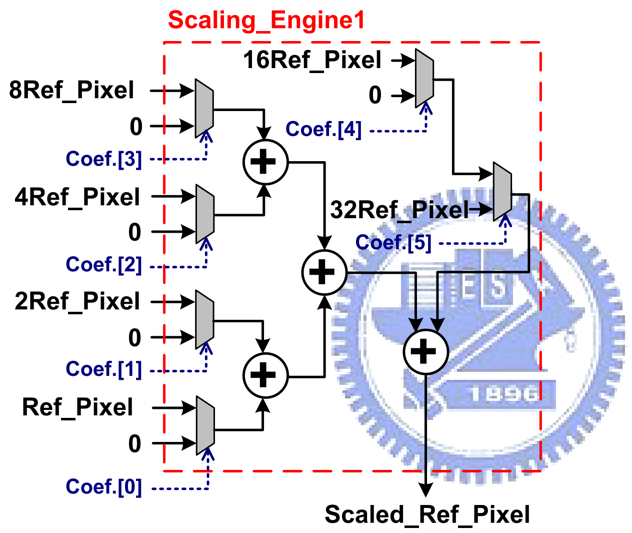


(a) (b)

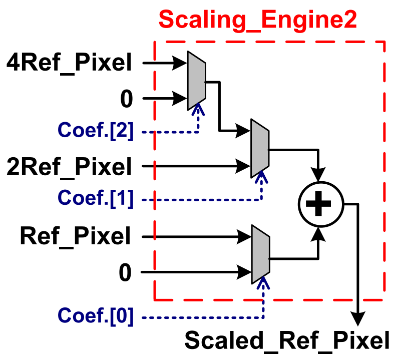
Figure 38: (a) Pixel_shifter1 generates six scaled sets, and (b) pixel_shifter2 generates three scaled sets.

4.3.1.3 Scaling_engine

The last parts are scaling_engine1 and scaling_engine2. They are composed of four adders and one adder, respectively, as shown in Figure 39 (a) and Figure 39 (b). In particular, every coefficient in the luma or chroma table can be combined with these scaled units which are after the pixel_shifters. All it has to do is to select the right combination of the scaled values in scaling_engines. However, each bit of the coefficient in binary form is just the control signal for the multiplexer in scaling_engine. For example, if the sixth coefficient of lu_coef3 is chosen, that means an eleven times of reference pixel is needed. First, we change the coefficient 11 to binary form 001011. Then, the first bit 1, second bit, and fourth bit will select the non-scaled, 2 times, and 8 times reference pixels, respectively. The other three zero bits will select the zero values. Finally, adding these selected scaled units (1+2+0+8+0+0=11) to get the final scaled result.



(a)



(b)

Figure 39: The architecture of (a) scaling_engine1 and (b) scaling_engine2.

4.3.2 Share-Based Extended Vertical Interpolator Design

Another interpolator, extended vertical interpolator, is proposed in Figure 40. We first extend and rewrite Eq. (5) to Eq. (7), and then recombine the Eq. (7) to a new form in Eq. (8). In Eq. (8), the common term (A+B+1) is merged together, and this common term also represent the equation for chroma extended vertical interpolation in Eq. (6). Therefore, due to the common term and two filters are merged together, the area cost can be more efficient.

$$
\begin{aligned}
L\_Pred\_out1 &= (-3)*L\_B\_int\_out1+(16*L\_B\_int\_out2+3*L\_B\_int\_out2) \\
&+(16*L\_B\_int\_out3+3*L\_B\_int\_out3)+(-3)*L\_B\_int\_out4+(16-3+3) \\
&=(16*L\_B\_int\_out2+16*L\_B\_int\_out3+16) \\
&-3*L\_B\_int\_out1-3*L\_B\_int\_out4-3 \\
&+3*L\_B\_int\_out2+3*L\_B\_int\_out3+3
\end{aligned} \tag{7}
$$

$$
\begin{aligned}
L\_Pred\_out1 &= 16*\left[L\_B\_int\_out2+L\_B\_int\_out3+1\right] \\
&-3*\begin{bmatrix}\left(L\_B\_int\_out1+L\_B\_int\_out4+1\right) \\ -\left(L\_B\_int\_out2+L\_B\_int\_out3+1\right)\end{bmatrix}
\end{aligned} \tag{8}
$$

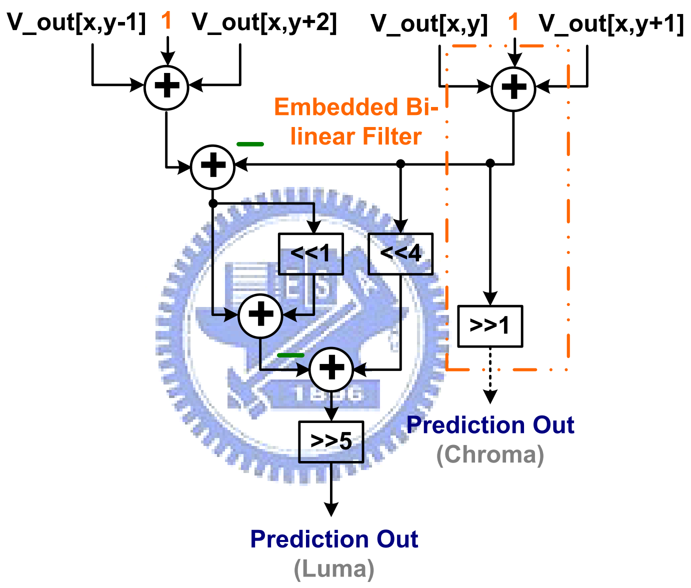


Figure 40: Proposed architecture of extended vertical interpolator.

4.4 *Short Summary*

In a short summary, we exploit an Intra_BL prediction module which contains banked SRAM, basic interpolator, and extended vertical interpolator. Our proposed basic interpolator design has better area efficiency compared to direct implementation, as shown in Figure 41. Table 9 describes the area comparisons between direct implementation and proposed basic interpolator design. Simulation result shows that 26% of total area cost can be saved in our proposed basic interpolator. The area of proposed share-based extended vertical interpolator is also shown in Table 9.

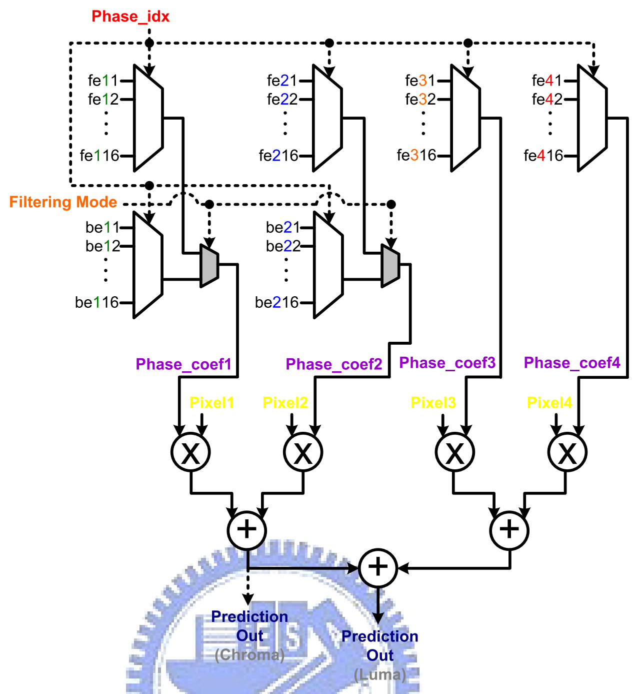


Figure 41: Direct implementation of basic interpolator.

Table 9: Simulation results of proposed basic interpolator and extended vertical interpolator.

Basic Interpolator	Direct Implementation	Hybrid	Reduction
Process	90 um		
Frequency	145 MHz		
Coefficient Generator (um^2)	436	298	31%
Pixel_Shifter + Scaling_Engine (um^2)	6382	4741	26%
Total Area	**6818**	**5039**	**26%**

	Extend Vertical Interpolator
Process	90 nm
Frequency	145 MHz
Area (um^2)	**1596**

In our SVC system, our target is to support HD720 (base layer) – HD1080 (enhancement layer) two spatial layers. However, we can also support more spatial layers such as qcif (base layer) – cif (enhancement layer1) – 4cif (enhancement layer2) – HD1080 (enhancement layer3) as long as the total MBs per second is less than 352800MBs/second ({[(1920x1088) + (1280x720)] / (16x16)} x 30 frames/s).

Therefore, for our Intra_BL prediction module, total of two basic horizontal interpolators, two basic vertical interpolators, and four extended vertical interpolators are used. The execution times for decoding a MB in each prediction type between spatial layers are listed in Table 10. It should be noticed that we calculate the processing cycle time in worst case. Hence, for the frame-MBAFF and MBAFF-MBAFF prediction types, the worst cases for decoding the enhancement blocks are bottom field MB and frame MB, respectively.

Table 10: Interpolation execution time for a MB in worst case.

	Processing Cycle in Worst Case
Frame – Frame	192
Field – Field	192
Frame – MBAFF **(bottom field MB)**	312
MBAFF – Frame	180
MBAFF – MBAFF **(frame MB)**	312
Frame – PAFF	192
PAFF – Frame	180

The detail about proposed Intra_BL prediction module is illustrated in Table 11. We use umc90 technology to simulate our work. The total gate count is 18673 under working frequency of 145 MHz, and the size of internal SRAM used is 5376 bits (luma and chroma). Among these prediction types between spatial layers, the most critical case is based on the bottom field block in enhancement layer of frame-MBAFF prediction. The result shows that the worst case of interpolation execution time in this prediction type is 312 cycles/MB.

Table 11: Proposal description.

	Proposed Intra_BL Prediction Module
Process	90 nm
Interpolator Components	Basic Horizontal x2 Basic Vertical x2 Extend Vertical x4
Interpolation Execution Time	312 cycles/MB (worst case)
Interpolator Gate Count	**10478**
Working Frequency	145 MHz
Total Gate Count	18673
Memory (bit)	5376

4.5 Memory Hierarchy Improvement and Computational Complexity Reduction

Based on our preliminary basic intra prediction and Intra_BL prediction designs in Chapter 3 and previous sections in Chapter 4, Figure 42 shows the internal core power consumptions in intra prediction at working frequency of 145 MHz. We can see that the power consumption becomes higher when the Intra_BL prediction is supported in SVC. This is because the memory fetching and computational complexity are both increased.

The power consumption of SVC intra prediction is consisted of internal SRAM power, dynamic power, and leakage power, as shown in Figure 43. Among these three parts, dynamic and SRAM power almost dominate the total power consumption. Therefore, in the following sections, we will try to reduce these two major power consumptions and further propose a power efficient Intra_BL prediction engine.

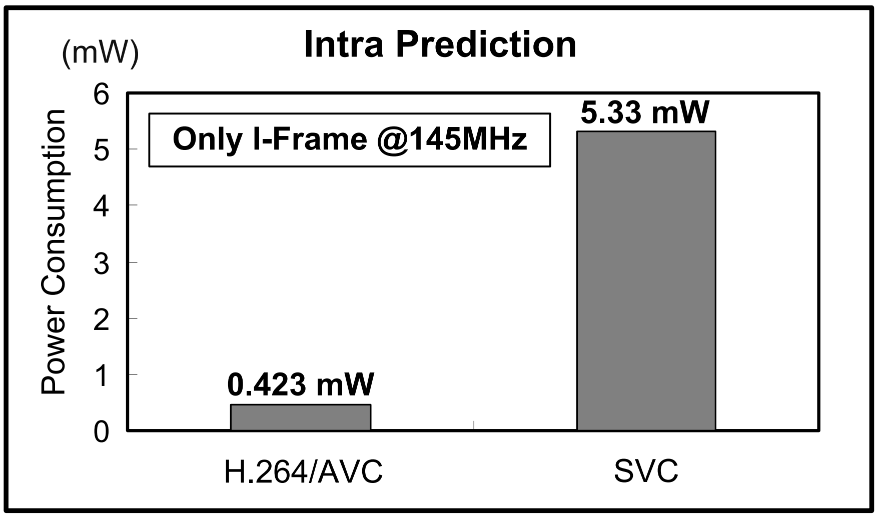


Figure 42: Intra power consumption in H.264/AVC and SVC.

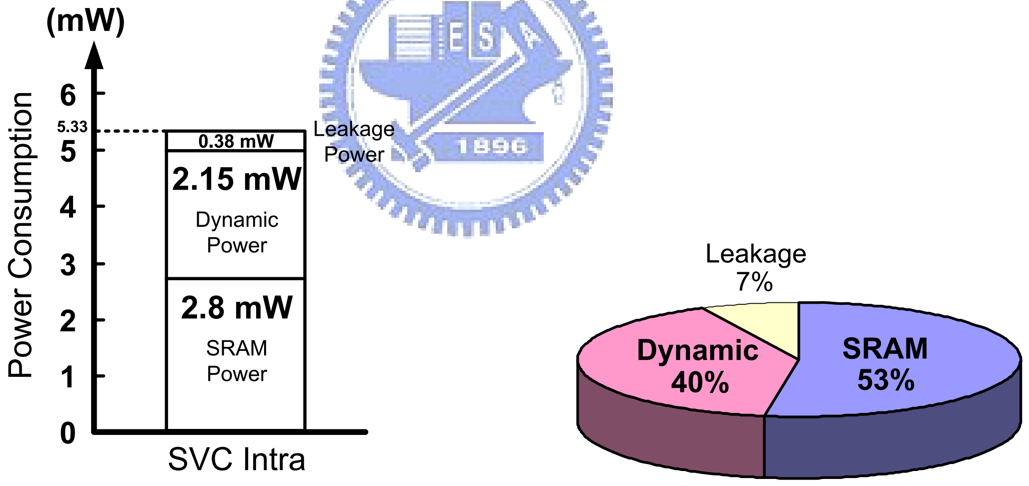


Figure 43: Power organization of Intra_BL prediction engine.

4.5.1 Internal SRAM Access Improvement

In Figure 43, internal SRAM occupies about half of total power consumption. In order to reduce this significant power, we improve the memory hierarchy in internal memory access. Four register sets are added to be the second stage of memory hierarchy, as described in Figure 44. These register sets are exploited to transform the large power consumption in

SRAM into register sets. To be noticed, in order to minimize the power consumption overhead in these register sets, gated clocks are applied to each set of register.

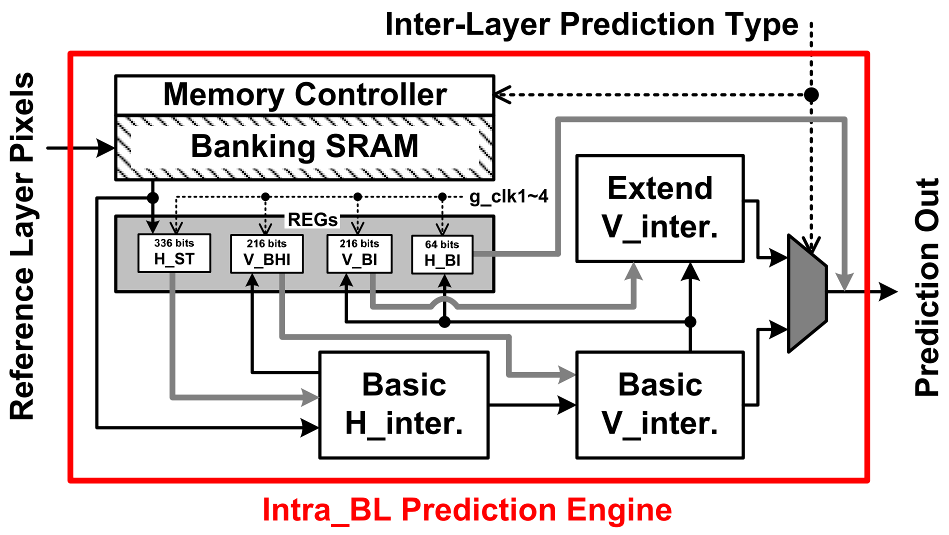


Figure 44: Memory hierarchy improvement of Intra_BL prediction engine.

These four register sets are V_BHI, V_BI, H_BI, and H_ST. The definitions of them are illustrated below.

- V_BHI: Store vertical direction values that passed through basic horizontal interpolation.
- V_BI: Store vertical direction values that passed through basic horizontal and vertical interpolations.
- H_BI: Store horizontal direction values that passed through basic horizontal and vertical interpolations.
- H_ST: Store horizontal direction values that come from internal banked SRAM.

V_BHI and V_BI are used to store and reuse the vertical related values as well as H_BI and H_ST are used to store and reuse the horizontal related values. Most of the data fetching will be transferred to register sets instead of directly reading from SRAM. For instance, we take the most critical case which uses frame picture type and MBAFF picture type for base layer and enhancement layer, respectively, and the decoding block in enhancement layer is a bottom field MB. Some definitions of representation in the following figures this example are listed in Figure 45.

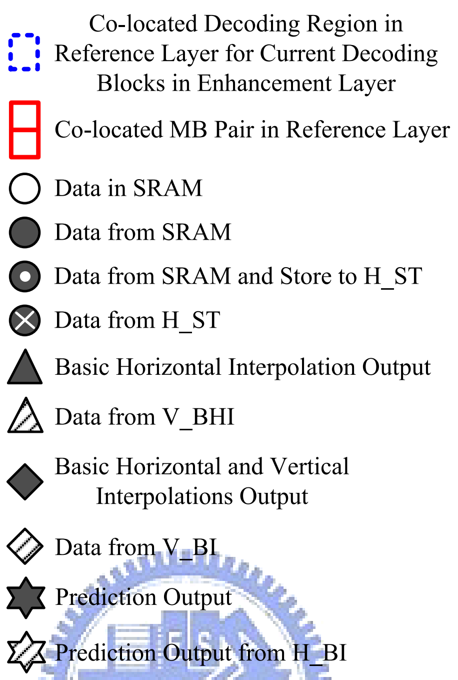
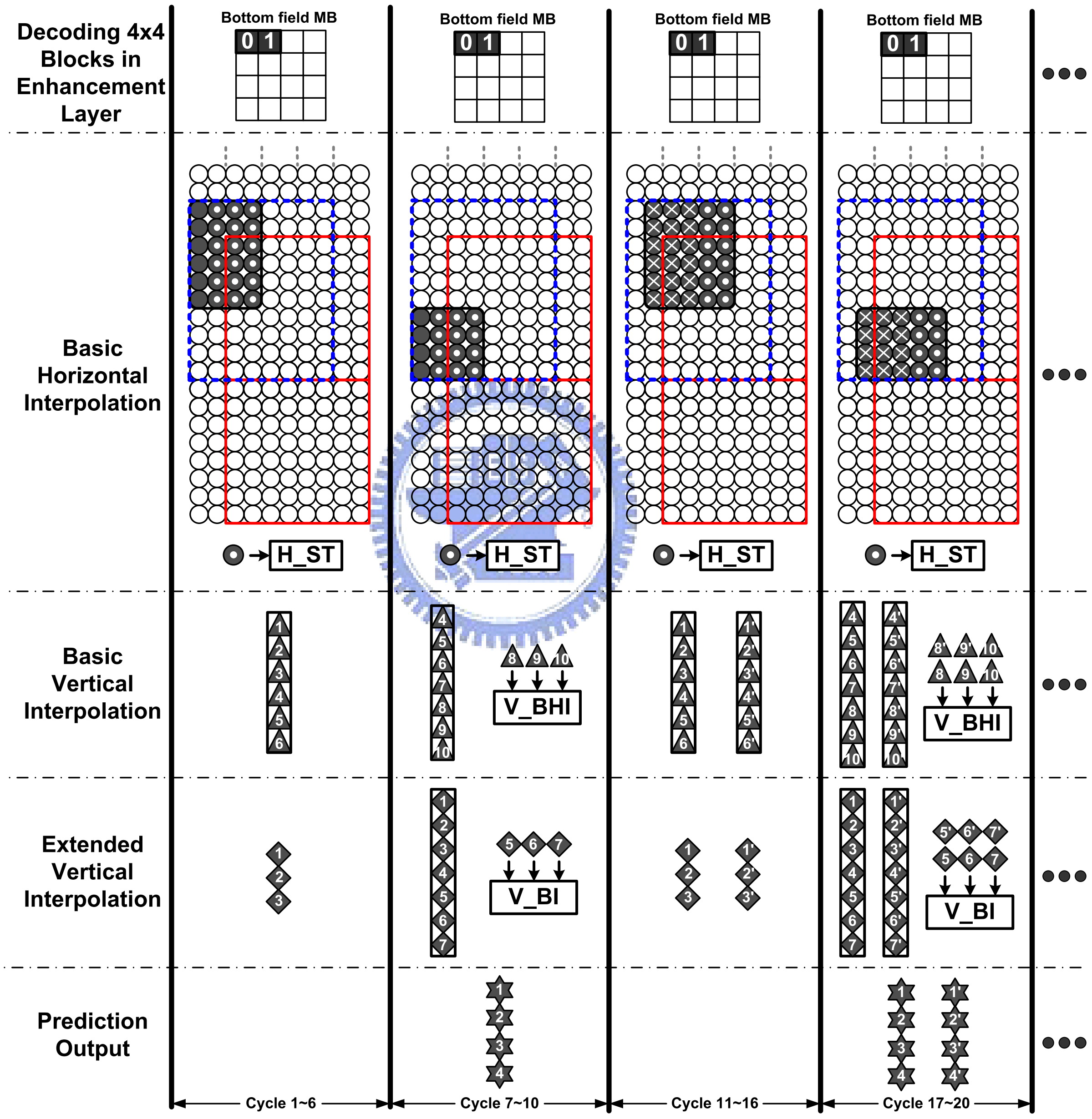

Co-located Decoding Region in Reference Layer for Current Decoding Blocks in Enhancement Layer
Co-located MB Pair in Reference Layer
Data in SRAM
Data from SRAM
Data from SRAM and Store to H_ST
Data from H_ST
Basic Horizontal Interpolation Output
Data from V_BHI
Basic Horizontal and Vertical Interpolations Output
Data from V_BI
Prediction Output
Prediction Output from H_BI

Figure 45: Some definitions of representations.

Figure 46, Figure 47, Figure 48, and Figure 49 illustrate the decoding process of this example. We take two 4x4 blocks in enhancement layer for a decoding unit. When the last three rows of co-located region in reference layer are decoded, the outputs of basic horizontal interpolation and basic vertical interpolation will be store to V_BHI set and V_BI set, respectively, and then for the next unit which is below to current unit to use. For the H_ST set, the data will be store to the H_ST set and read from H_ST set except the first basic horizontal interpolation. In particular, when the 4x4 blocks 2 and 3 are decoding in the enhancement layer, the basic horizontal interpolation does not have to start from the first row of the co-located region due to the required values have been already stored to the V_BHI and V_BI sets, as shown in Figure 47. Therefore, the required values can be directly gotten from the V_BHI and V_BI sets during cycle 31 to 42 in Figure 47. Hence, some calculations can be removed. Furthermore, the prediction outputs will also be store to H_BI at the end of the decoding unit. These stored prediction outputs can be reuse in the next decoding unit that right next to the current unit, as described in cycle 43 to 48 and cycle 63 to 66 in Figure 48 and Figure 49, respectively.



Decoding 4x4 Blocks in Enhancement Layer
Basic Horizontal Interpolation
Basic Vertical Interpolation
Extended Vertical Interpolation
Prediction Output
Bottom field MB
0 1
H_ST
V_BHI
V_BI
Cycle 1~6
Cycle 7~10
Cycle 11~16
Cycle 17~20

Figure 46: An example of using these register sets.

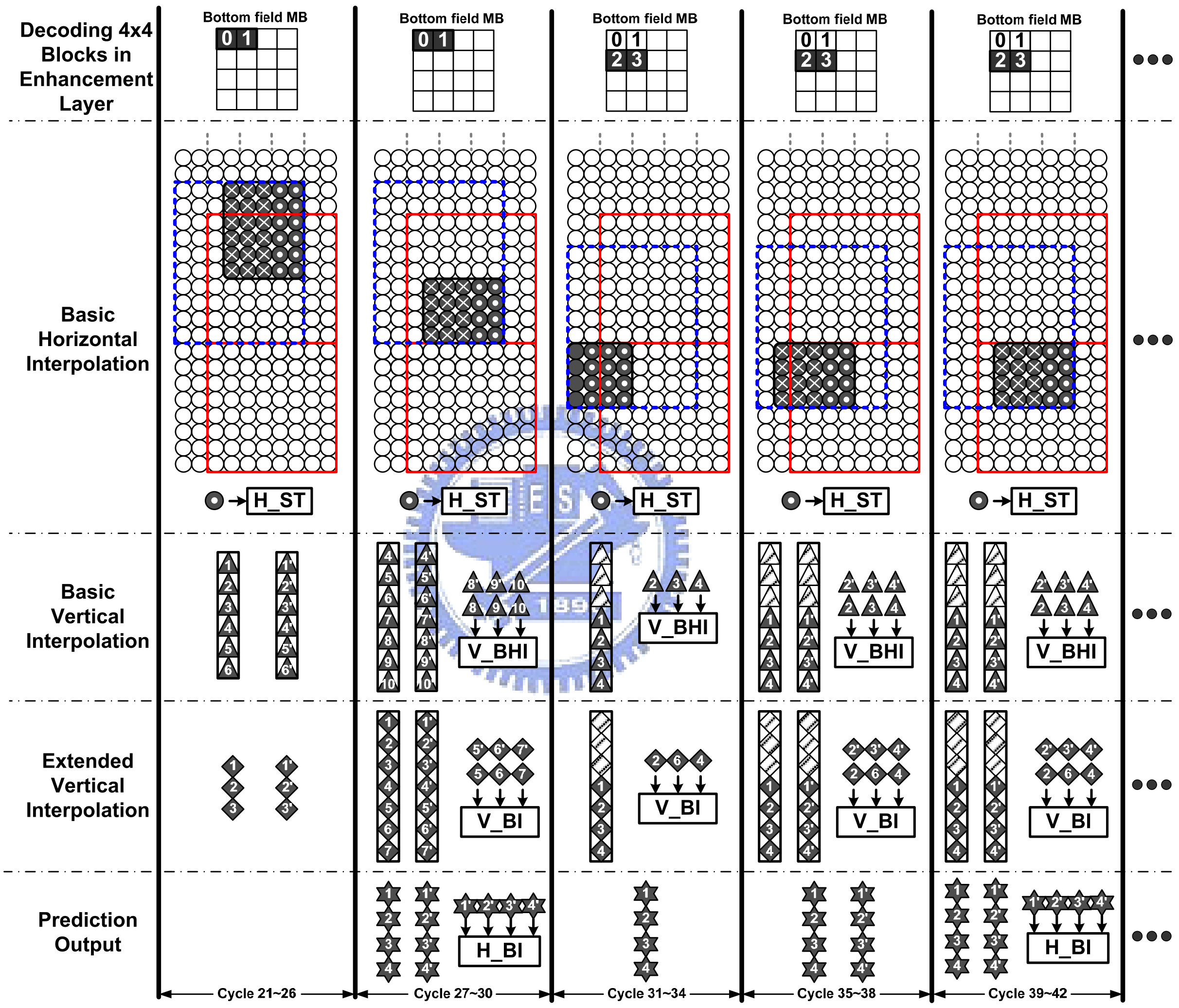

Decoding 4x4 Blocks in Enhancement Layer
Basic Horizontal Interpolation
Basic Vertical Interpolation
Extended Vertical Interpolation
Prediction Output
Bottom field MB
H_ST
V_BHI
V_BI
H_BI
Cycle 21~26
Cycle 27~30
Cycle 31~34
Cycle 35~38
Cycle 39~42

Figure 47: An example of using these register sets.

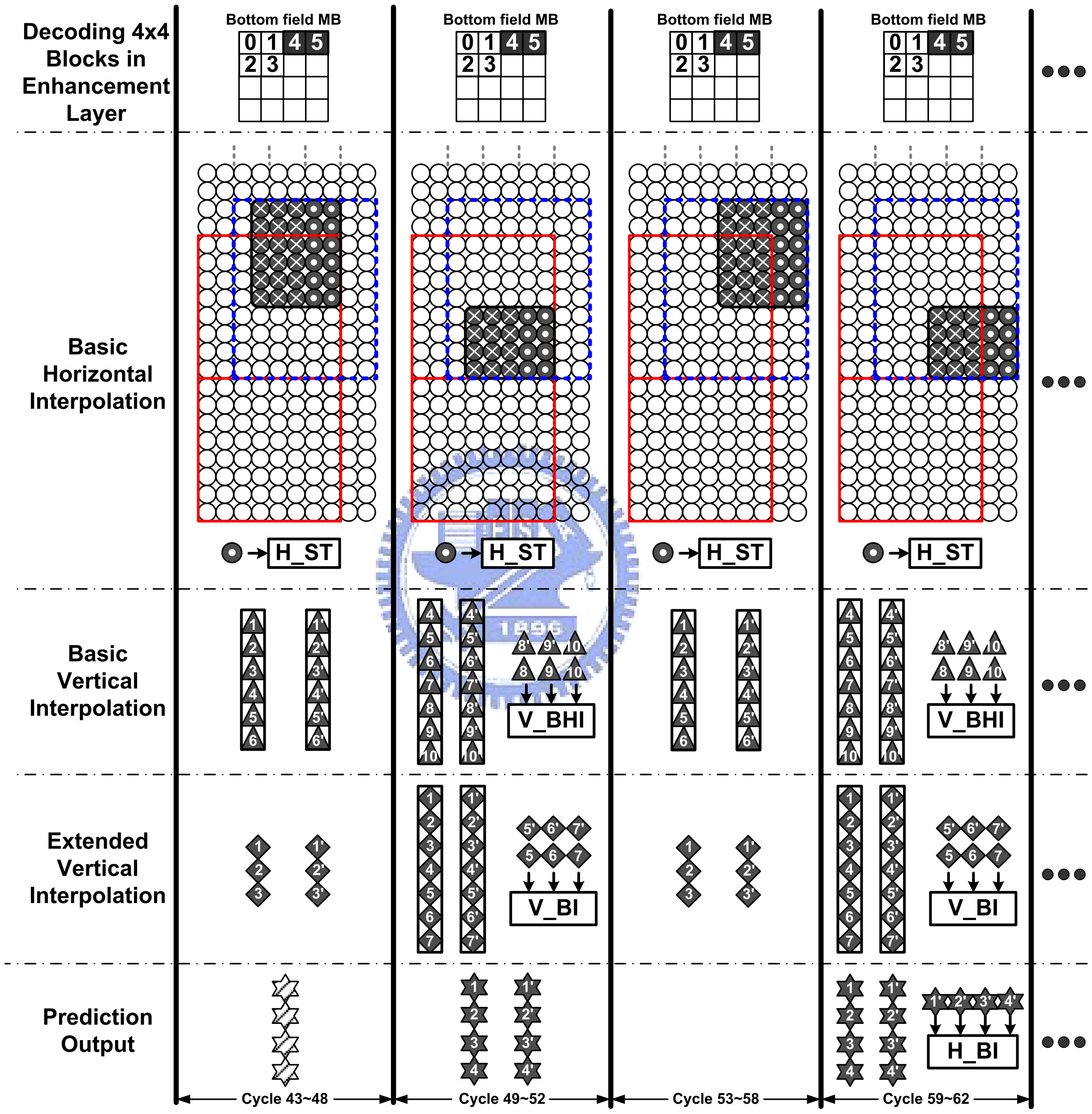

Decoding 4x4 Blocks in Enhancement Layer
Basic Horizontal Interpolation
Basic Vertical Interpolation
Extended Vertical Interpolation
Prediction Output
Bottom field MB
0 1 4 5
2 3
H_ST
V_BHI
V_BI
H_BI
Cycle 43~48
Cycle 49~52
Cycle 53~58
Cycle 59~62

Figure 48: An example of using these register sets.

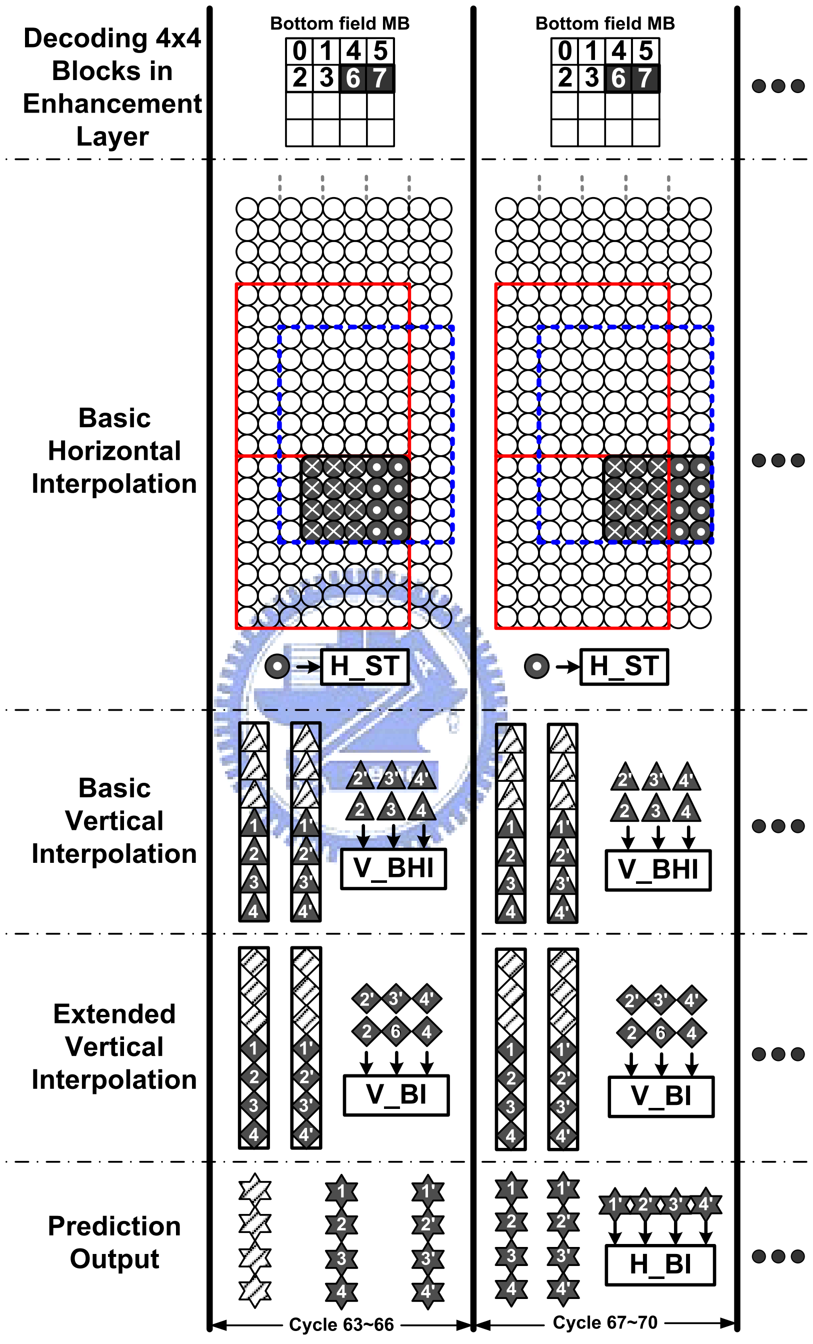


Figure 49: An example of using these register sets.

Table 12 shows the power reduction in internal SRAM when the second stage four register sets are applied to memory hierarchy. Simulation result shows that 62.3% of total power consumption in memory fetching can be reduce. Moreover, as mentioned before, some calculations can be reduced due to stored values which are already interpolated. Therefore, the second memory stage not only reduces the power consumption but also improves the execution time, as shown in Table 13.

Table 12: SRAM Power reduction by four REGs sets.

	Original	Original+ V_BHI+ V_BI	Original+ V_BHI+ V_BI+ H_BI	Original+ V_BHI+ V_BI+ H_BI+ H_ST
SRAM Power Consumption	2.8 mW	1.905 mW	1.714 mW	1.06 mW
Buffer Power Consumption	0 mW	0.069924 mW	0.07574 mW	0.115084 mW
Power Reduction	0%	32%	36.1%	62.3%

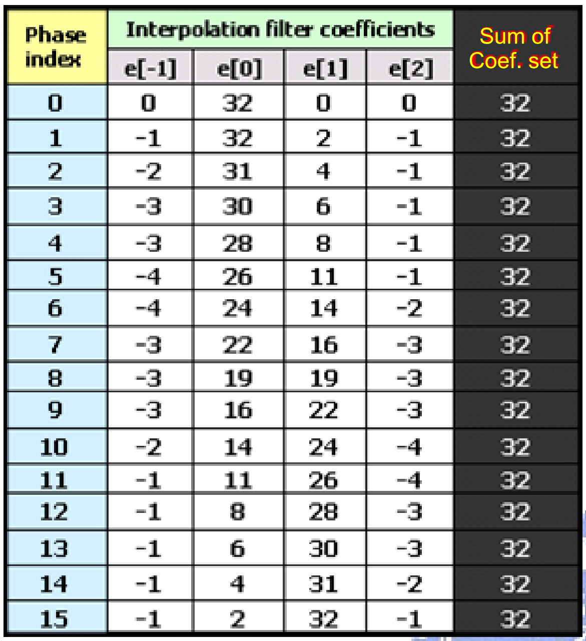
Table 13: Processing cycle reduction by four REGs sets (frame - MBAFF).

	Original	Original+ V_BHI+ V_BI	Original+ V_BHI+ V_BI+ H_BI	Original+ V_BHI+ V_BI+ H_BI+ H_ST
Processing Cycles for a MB	312	192	170	170
Processing Cycle Reduction	0%	38.46%	45.51%	45.51%

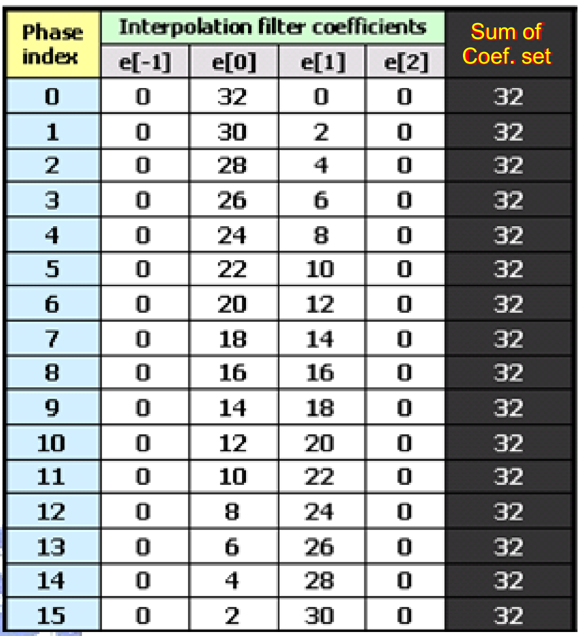
4.5.2 Intra_BL Decoding Flow Modification

Another major power consumption part is dynamic power consumption. In this section, we try to reduce the computational complexity of interpolation process and modify the Intra_BL decoding flow. In Section 4.3, Eq. (3) and Eq. (4) illustrate the basic interpolation process equations. However, the interpolation coefficient sets in both four-tap poly-phase filter and bi-linear filter have the same property, as described in Table 14.

Table 14: Property of coefficient sets in luma and chroma filter tables.



Phase index	Interpolation filter coefficients				Sum of Coef. set
	e[-1]	e[0]	e[1]	e[2]	
0	0	32	0	0	32
1	-1	32	2	-1	32
2	-2	31	4	-1	32
3	-3	30	6	-1	32
4	-3	28	8	-1	32
5	-4	26	11	-1	32
6	-4	24	14	-2	32
7	-3	22	16	-3	32
8	-3	19	19	-3	32
9	-3	16	22	-3	32
10	-2	14	24	-4	32
11	-1	11	26	-4	32
12	-1	8	28	-3	32
13	-1	6	30	-3	32
14	-1	4	31	-2	32
15	-1	2	32	-1	32



Phase index	Interpolation filter coefficients				Sum of Coef. set
	e[-1]	e[0]	e[1]	e[2]	
0	0	32	0	0	32
1	0	30	2	0	32
2	0	28	4	0	32
3	0	26	6	0	32
4	0	24	8	0	32
5	0	22	10	0	32
6	0	20	12	0	32
7	0	18	14	0	32
8	0	16	16	0	32
9	0	14	18	0	32
10	0	12	20	0	32
11	0	10	22	0	32
12	0	8	24	0	32
13	0	6	26	0	32
14	0	4	28	0	32
15	0	2	30	0	32

The sum of coefficients in each set is equal to 32. Hence, base on this property, if all of the input reference pixels are equal, then the Eq. (3) and Eq. (4) can be rewritten to Eq. (9) and Eq. (10). Therefore, the basic interpolation process can be modified to a simple shifter instead of complex operations. The modified Intra_BL decoding process and architecture are shown in Figure 50 and Figure 51, respectively.

$$\begin{aligned} \mathit{L\_Pred\_out1}\ \text{(Luma)} &= \mathit{L\_coef1} * \mathit{L\_ref\_pixle1} + \mathit{L\_coef2} * \mathit{L\_ref\_pixle2} \\ &\quad + \mathit{L\_coef3} * \mathit{L\_ref\_pixle3} + \mathit{L\_coef4} * \mathit{L\_ref\_pixle4} \\ &= 32 * \mathit{L\_ref\_pixleX} = \mathit{L\_ref\_pixleX} << 5 \quad (X = 1, 2, 3, \text{or } 4) \end{aligned} \tag{9}$$

$$\begin{aligned} \mathit{C\_Pred\_out1}\ \text{(Chroma)} &= \mathit{C\_coef1} * \mathit{C\_ref\_pixle1} + \mathit{C\_coef2} * \mathit{C\_ref\_pixle2} \\ &= 32 * \mathit{C\_ref\_pixleX} = \mathit{C\_ref\_pixleX} << 5 \quad (X = 1 \text{ or } 2) \end{aligned} \tag{10}$$

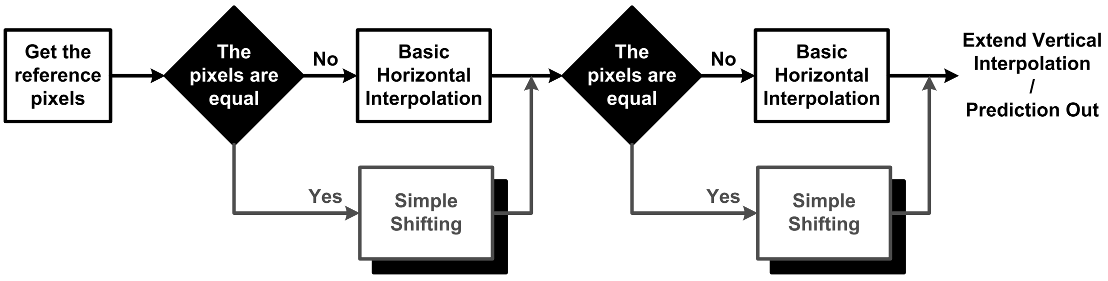

Get the reference pixels
The pixels are equal
No
Yes
Basic Horizontal Interpolation
Simple Shifting
The pixels are equal
No
Yes
Basic Horizontal Interpolation
Simple Shifting
Extend Vertical Interpolation / Prediction Out

Figure 50: Modified Intra_BL decoding process.

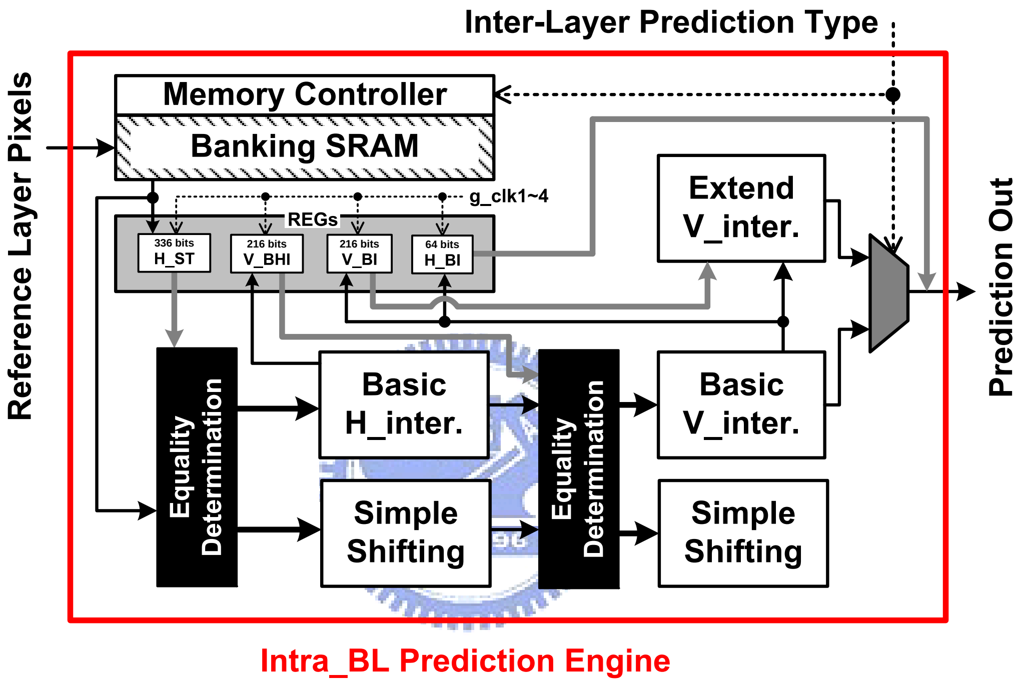


Figure 51: Proposed power efficient Intra_BL prediction engine.

The detailed description about our power efficient proposal and preliminary design is listed in Table 15. Although the total area cost in more than the preliminary design due to the second stage register sets, the power consumption is about half of original proposal. Furthermore, we also analysis the pixel equality based on the different test sequences and quantization parameters (QPs), as illustrated in Table 16. Analysis shows that the probability of equality is proportioned to QP. It should be noticed that only basic interpolation process is modified, since the probability of equality for extended vertical interpolation process is very small (<2%).

Table 15: Comparisons between new proposal and preliminary design.

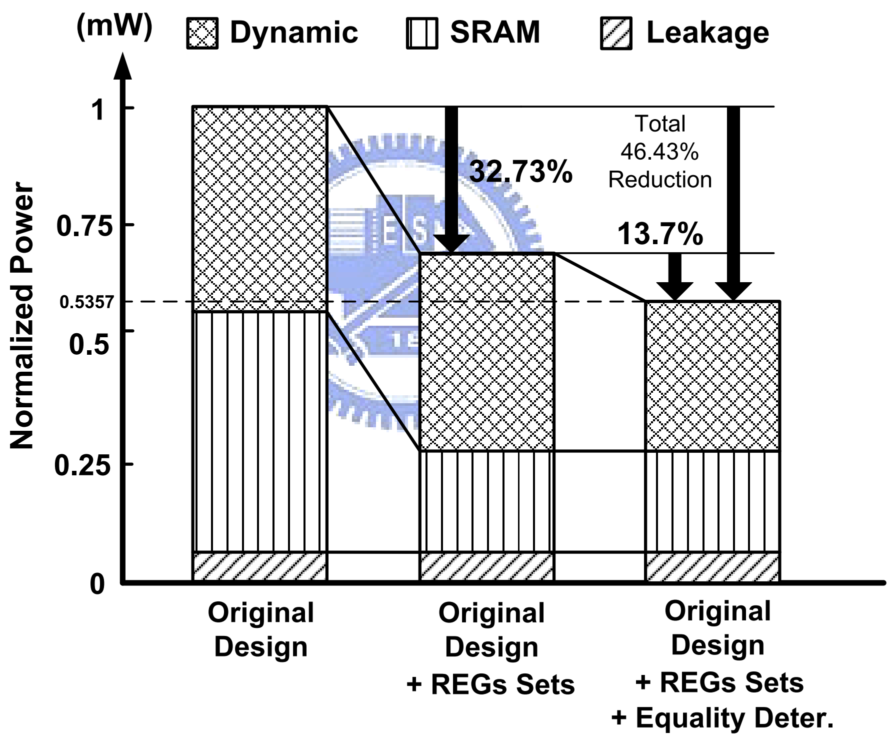
	Original Proposal	Power Efficient Proposal
Process	90nm	90nm
Interpolator Components	Basic Horizontal x2 Basic Vertical x2 Extend Vertical x4	Basic Horizontal x2 Basic Vertical x2 Extend Vertical x4
Interpolation Execution Time	312 cycles/MB (worst case)	170 cycles/MB (worst case)
Interpolator Gate Count	10478	10478
Working Frequency	145 MHz	145 MHz
Total Gate Count	18673	24793
Power Consumption @145MHz	5.33 mW	2.86 mW
Memory (bit)	5376	5376

Table 16: Probabilities of equality in different QPs and test sequences.

	QP = 28		QP = 35		QP = 42	
	H_Int.	V_Int.	H_Int.	V_Int.	H_Int.	V_Int.
Foreman	18.7%	6.9%	23.0%	10.7%	24.2%	14.4%
Akiyo	34.1%	14.0%	36.2%	18.7%	44.5%	27.3%
M & D	27.5%	20.4%	31.8%	24.1%	36.2%	30.4%
Stefan	29.9%	5.17%	33.5%	10.1%	35.8%	13.0%
Grandma	25.2%	17.9%	31.5%	23.3%	39.2%	35.9%
Table	16.2%	1.3%	43.4%	29.7%	65.8%	55.9%

4.6 *Simulation Result*

To summarize the power efficient techniques discussed in Section 4.5, Figure 52 shows the composition of the power consumption when the intra prediction runs in SVC standard at working frequency of 145 MHz. By applying second stage register sets in memory hierarchy and modified simpler interpolation decoding process, the SRAM and dynamic power are reduced due to memory access times and computational complexity. The second stage register sets not only reduce 62.3% of internal SRAM power consumption as compared to original only banked SRAM design, but also reduce the processing cycles by 45.51%. For dynamic power, modified Intra_BL decoding process can reduce 34% of power consumption due to the simpler operations. By these two methods, total power of SVC intra prediction engine can have 46.43% reduction.



***Note1*: Equality determination power reduction is based on the "Table" test sequence and QP = 42.**

Figure 52: Power reduction of proposed design.

The overall proposal description of SVC intra prediction engine is shown in Table 17. Although the area overhead of power efficient proposal is about 22% compared to original design, the total power consumption and execution time can have significant improvements with 46.43% and 45.51% reductions, respectively.

Table 17: Proposal description.

	Original Proposal	**Power Efficient Proposal**
Profile	H.264/SVC HP	H.264/SVC HP
Process	90nm	90nm
Interpolator Components	Basic Horizontal x2 Basic Vertical x2 Extend Vertical x4	Basic Horizontal x2 Basic Vertical x2 Extend Vertical x4
Interpolation Execution Time	312 cycles/MB (worst case)	170 cycles/MB (worst case)
Working Frequency	100 MHz/145 MHz	100 MHz/145 MHz
Interpolator Gate Count @ 145 MHz	10478	10478
Total Gate Count @ 145 MHz	35106	42756
Power Consumption	0.292 mW/5.33 mW	0.292 mW/2.86 mW
Memory (bit)	6064	6064

Chapter 5
Conclusion and Future Work

5.1 Conclusion

A power efficient SVC intra prediction engine is developed which supports high profile intra prediction in both H.264/AVC and SVC. This engine is consisted of two major prediction parts, which are basic prediction and Intra_BL prediction.

5.1.1 Basic Intra Prediction for H.264/AVC

We propose a high-profile intra predictor to support MBAFF and Luma *intra_8x8* decoding. The proposed memory hierarchy includes upper, left and corner memory buffer which reuses the neighboring pixels for follow-up prediction procedures. In Luma_8x8 decoding process, we propose base-mode predictors to minimize the additional hardware cost, latency penalty, and filtered pixel buffer memory size. Compared to the existing design [16] without supporting intra 8x8 coding, this design only introduce 10% and 7.5% of gate counts and SRAM overheads. The proposed design can achieve real-time processing requirement for HD1080 format video in 30fps under the working frequency of 100MHz.

5.1.2 Intra_BL prediction for SVC

We propose an Intra_BL prediction engine which supports different inter-layer prediction picture types. For the main interpolator design, the area efficient architecture is proposed by coef_generator, pixel_shifter, and scaling_engine. Coef_generator generates the each coefficient set of luma and chroma samples by some relationship between *phase_idx* and coefficient values. Pixel_shifter does some simple shifts for reference pixels, and scaling_engine used to add the each combination from the pixel_shifter, and finally output the prediction pixels. Via these methods, the area cost can be reduced 26% at working frequency of 145MHz in UMC 90nm technology compared to the direct implementation.

To reduce the total power consumption in SVC intra prediction, second stage memory hierarchy and modified interpolation decoding process are further proposed based on our preliminary design. Second stage memory hierarchy transfers the significant power consumption in SRAM into four register sets. By adding these register sets, the power consumption in SRAM and processing cycles can be reduced by 62.3% and 45.51%. Moreover, equality determination is used to simplify the computational complexity in basic interpolation. The modified Intra_BL decoding process can use simple shift instead of complex interpolation calculations. For dynamic power, 34% of power consumption can be reduced by this method. The total power consumption in SVC intra prediction engine can have 46.43% reduction by these two techniques. The proposed power efficient SVC intra prediction engine can achieve the maximum of two spatial layers with HD720 and HD1080 resolutions in 30fps under working frequency of 145 MHz.

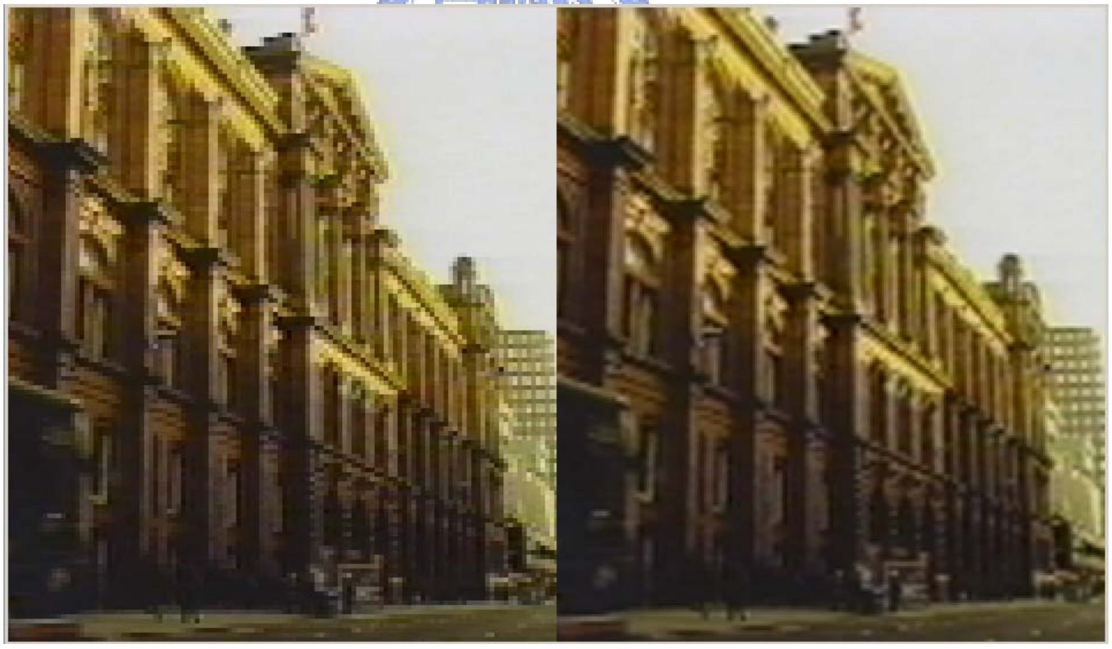
5.2 Future Work

5.2.1 Error Concealment on Base Layer

Error correction is always a special and important issue in the video decoder. However, it is also hard to find a suitable method to implement due to the error propagation. A simple way to solve this problem is adding a big frame buffer to the design. However, this method is not appropriate to VLSI implementation, since that will cause a large area cost. Therefore, for the base layer decoding which uses the traditional H.264/AVC, the error issue is still a challenge.

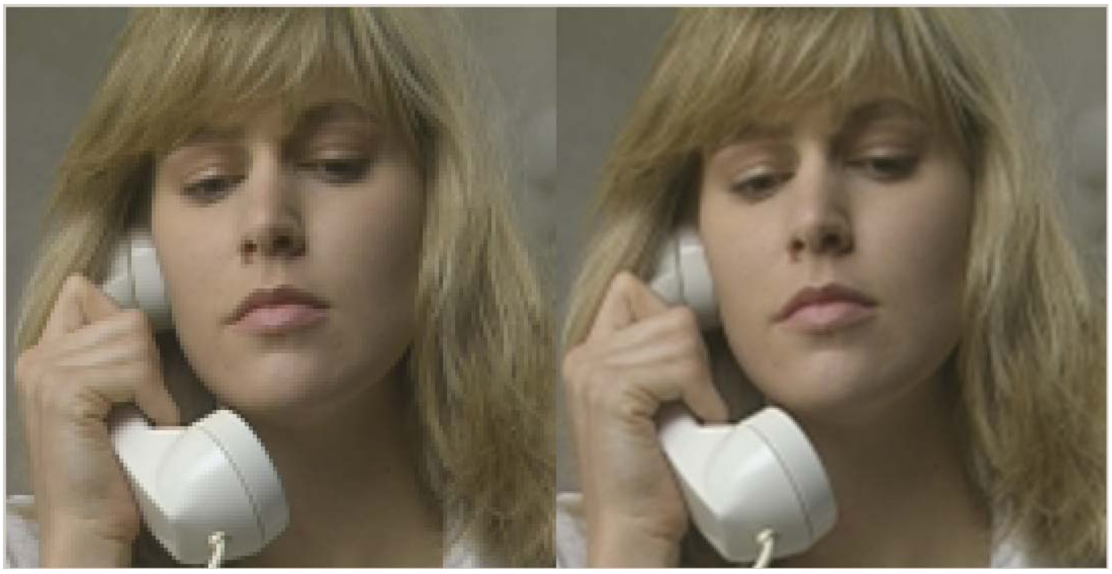
5.2.2 Error Concealment on Enhancement Layer

On the other hand, the error correction in enhancement is easier to solve compared to base layer. Since the enhancement can use the information from base layer, this characteristic can also be applied to error correction. A method to solve the error block in enhancement layer is using the co-located block region in base layer. Different from upsampling process in SVC, we can use another simpler method called one-to-four to upsample the pixel instead of complex interpolation process. For the co-located pixels in base layer, we can directly extend the one pixel to be four pixels in enhancement layer (assume that the ratio of enhancement resolution to reference layer is 2:1). Figure 53 and Figure 54 show the results of upsampling method in one-to-four method and interpolation method. Although the reconstructed picture is not smooth as interpolation method, one-to-four is a simple method and a suitable way in VLSI implementation.



(a) (b)

Figure 53: (a) One-to-four method and (b) upsampling method in SVC for test sequence “glasgow”.



(a) (b)

Figure 54: (a) One-to-four method and (b) upsampling method in SVC for test sequence “suzie”.

Bibliography

[1] Joint Video Team, *Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification*, ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, May 2003.

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

[3] T. Wiegand, G. J. Sullivan, J. Reichel, H. Schwarz, and M. Wien (Eds.), Amendment 3 to ITU-T Recommendation H.264 (2005) | ISO/IEC 14496-10:2005, *Scalable video coding*, July 2007.

[4] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 9, pp. 1103–1120, Sep 2007.

[5] ITU-T Recommendation H.261. Video codec for audiovisual services at px64 kbit/s. Helsinki, March 1993.

[6] ITU-T Recommendation H.263. Video coding for low bitrate communication. Geneva, January 1998.

[7] ISO/IEC.IS 11172:Informaton technology -coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s, 1993.(MPEG-1).

[8] ISO/IEC.IS 13818-2: Information technology -generic coding of moving pictures and associated audio information: Video, 1995. (MPEG-2 Video).

[9] Weiping Li, Jens-Rainer Ohm, Mihaela van der Schaar, Hong Jiang, Shipeng Li, “ MPEG-4 Video Verification Model version 18.0”, ISO/IEC JTC1/SC29/WG11, WG 11 document N3908, Pisa, Italy, January 2001.

[10] L. Wang, K. Panusopone, R. Gandhi, Y. Yu and A. Luthra, “Interlace Coding Tools for H.26L Video Coding,” ITU-T VCEG-N57, Pattaya, Thailand, 2001.

[11] D. Marpe and et al., “H.264/MPEG4-AVC fidelity range extensions: Tools, profiles, performance, and application areas,” in *Proc. IEEE ICIP*, vol. 1, pp. 593–596, Sept. 2005.

[12] T.Wiegand, G. J. Sullivan, J. Reichel, H. Schwarz, and M.Wien, *Joint Draft 11 of SVC Amendment*, Joint Video Team, Doc. JVT-X201, Jul. 2007.

[13] H. Schwarz, D. Marpe, and T. Wiegand, “Overview of the scalable video coding extension of the H.264/AVC standard,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, pp. 1103–1120, Sep. 2007.

[14] Yu-Wen Huang, Bing-Yu Hsieh, Tung-Chien Chen, and Liang-Gee Chen, “Hardware architecture design for H.264/AVC intra frame coder”, *ISCAS 2004.*

[15] Esra Sahin and Ilker Hamzaoglu, “An Efficient Intra Prediction Hardware Architecture for H.264 Video Decoding”, *DSD 2007.*

[16] Jia-Wei Chen, Chien-Chang Lin, Jiun-In Guo, and Jinn-Shyan Wang, “Low Complexity Architecture Design of H.264 Predictive Pixel Compensator for HDTV Application”, *ICASSP 2006.*

[17] Ting-An Lin, Sheng-Zen Wang, Tsu-Ming Liu and Chen-Yi Lee, “An H.264/AVC Decoder with 4x4 Block-Level Pipeline,” *Proc. ISCAS*, pp. 1810-1813, May, 2005.

[18] Tsu-Ming Liu and Chen-Yi Lee, “Design of an H.264/AVC Decoder with Memory Hierarchy and Line-Pixel-Lookahead,” *Journal of VLSI Signal Processing Systems*, Aug. 2007.

[19] Chun-Hao Chang, Jia-Wei Chen, Hsiu-Cheng Chang, Yao-Chang Yang, Jinn-Shyan Wang, “A QUALITY SCALABLE H.264/AVC BASELINE INTRA ENCODER FOR HIGH DEFINITION VIDEO APPLICAITONS”, *SPS 2007*.

[20] Micron® Technology Inc. The Micron® System-Power Calculator: SDRAM. [Online Available]: http://www.micron. com/products/dram/syscalc.html

作　者　簡　歷

姓名：賴昱帆

戶籍地：台灣省台北市

出生日期：1984.09.24

學歷：2000.09 ~ 2003.06 國立建國高級中學

2003.09 ~ 2007.06 國立交通大學 電子工程學系

2007.09 ~ 2009.07 國立交通大學 電子工程研究所碩士班

得　獎　事　績

2009/05　2009 全國 IC 設計競賽設計完整獎

發　表　論　文

- **Yu-Fan Lai**, Tsu-Ming Liu, Yao Li, Chen-Yi Lee, "Design of An Intra Predictor with Data Reuse for High-Profile H.264 Applications," *IEEE International Symposium on Circuit and System (ISCAS'09)*, pp. 3018-3021, May 2009.