國 立 交 通 大 學

電子工程學系　電子研究所碩士班

碩 士 論 文

H.264/AVC Scalable High Profile 解碼器之設計與實作

Design and Implementation of H.264/AVC Scalable High Profile Decoder

研究生：陳宥辰

指導教授：張添烜 博士

中華民國　九十九 年　九 月


E S A
1896

H.264/AVC Scalable High Profile 解碼器之設計與實作

Design and Implementation of H.264/AVC Scalable High Profile Decoder

研 究 生：陳宥辰　　Student: Yu-Chen Chen

指導教授：張添烜 博士　　Advisor: Tian-Sheuan Chang

國 立 交 通 大 學

電子工程學系 電子研究所碩士班

碩 士 論 文

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

中華民國 九十九 年 九 月

H.264/AVC Scalable High Profile 解碼器之設計與實作

研究生：陳宥辰　　指導教授：張添烜 博士

國立交通大學

電子工程學系電子研究所碩士班

摘　要

隨著愈來愈先進的視訊標準，視訊裝置的應用也更趨廣泛。在這些標準之中，可調性影像編碼(SVC)不僅提供高規格的影像編碼，同時也支援了時間、品質、空間上的可調性。然而這些可調性在視訊晶片的設計上會造成解碼時間、記憶體頻寬、邏輯閘成本等額外的負擔。因此，本篇論文呈現了 Scalable High profile H.264/AVC 解碼器從解碼流程分析、架構設計到模組實作的優化。

在解碼流程上，本篇論文採取先前提出的畫面幀為基礎(frame-based)之空間層解碼，並提出一個可以在記憶體頻寬和巨圖塊的處理週期分別能達到 71%和66%縮減的單次品質層解碼流程。對於在層間幀內預測的質地填充(texture padding)方面，我們提出了基本層級(BL-level)的填充流程並節省了層間幀內預測巨區塊 26%的解碼時間。

在上述流程下，本解碼器採取四級管線架構設計來增加解碼速度。第一個管線級是由三品質層平行處理的熵解碼器(Entropy Decoder)和語法解析器(Syntax Parser)所組成。第二個管線級是由殘餘重建路徑、層間預測器、以及參考像素抓取單元所組成。本論文特別針對殘餘重建路徑進行優化，以解決由可調性所造成的額外複雜度。經由實驗結果，我們所提出的平行管線架構和暫存結果重複使用

(temporal result reusing)方法相對於傳統方法能節省 54%的邏輯閘。對於層間預測，我們提出中央化的累加器型層間對應結構、簡化的多相插值器以及有效率的移動向量向上取樣器來節省邏輯閘成本和解碼時間。第三個管線級是由動作補償和幀內預測器所組成。而第四個管線級是由去區塊濾波器和質地填充器所組成。為了有效存取外部記憶體，本篇論文使用了針對可調性解碼規格客製化的記憶體要求協定。

最後，我們提出的 Scalable High profile 解碼器在 UMC 90 奈米的製程環境下總共約使用了 54 萬個邏輯閘和 3 萬 9 千個位元組的內部記憶體。其在一秒內可以處理 60 張 CIF-SD480p-HD1080p 規格和三層品質層的畫面幀。相對於較早的解碼器，本實作能在多樣可調性的基礎上提供更好的解碼效率。

Design and Implementation of H.264/AVC Scalable High Profile Decoder

Student: Yu-Chen Chen Advisor: Tian-Sheuan Chang

Department of Electronics Engineering & Institute of Electronics

National Chiao Tung University

Abstract

Video applications are everywhere with the more and more advance standards. Scalable Video Coding (SVC) not only supports high definition specifications but also provides temporal, quality, and spatial scalabilities. However, these additional scalabilities cause the decoding time, memory bandwidth, and area cost overhead in chip design aspect. Thus, this thesis presents an H.264/AVC Scalable High Profile decoder with optimizations on decoding flow, architecture design, and module implementation.

For decoding flow, this thesis adopts the previous proposed frame-based flow for spatial layer decoding, and proposes one-pass MB-based flow for quality layer decoding that saves 71% and 66% in external memory bandwidth and macroblock processing cycle respectively. For texture padding in inter-layer intra prediction, we propose BL-level padding flow that saves 26% decoding time in IntraBL coded macroblocks.

With above flow, the decoder adopts four stages pipeline architecture to enhance

the decoding throughput. The 1st stage is composed of entropy decoder and syntax parser which deal with 3 quality coefficients in parallel. The 2nd stage is composed of residual reconstruction path, inter-layer predictor, and reference pixels fetch unit. This thesis specifically optimizes the residual reconstruction path with parallel-pipeline architecture and temporal result reuse to cope with the additional complexity from SVC standard, which leads to 54% gate count savings compared with the traditional serial-pipeline architecture. For inter-layer predictor design, we propose the centralized accumulation-based CCSP concept, simplified poly-phase interpolator, and efficient MV upsampler to save the area cost and decoding time. The 3rd stage is composed of motion compensation and Intra predictor. The 4th stage is composed of the deblocking filter and the texture padder. To efficiently access external memory, a SVC-customized memory protocol is adopted in this thesis.

Finally, the proposed design Scalable High profile decoder is implemented with UMC 90nm CMOS technology, which cost 565.12k gate count, and 39.66 Kbytes on chip memory. It is capable of 60fps, CIF-SD480p-HD1080p, and 3 quality layers decoding at 135MHz. Compared to the previous designs, the proposed decoder achieves better decoding efficiency based on multiple scalabilities.

誌　謝

首先，要感謝我的指導教授－張添烜博士。張教授給予我在研究上自由揮灑的空間，並適時地提供建議和教導，並讓我能以確的方法和態度進行研究。此外，教授提供的各項訓練和資源，也讓我的研究得以順利進行。在此由衷地感謝張添烜教授。

同時也要謝謝我的口試委員們，交大電子李鎮宜教授、中央電機蔡宗漢教授，在百忙之中抽空前來口試，感謝各位教授的不吝指導讓本論文得以更加完備。

接著，我要感謝實驗室的夥伴。謝謝李國龍學長，帶我進入影像編碼的領域，並且與SVC團隊的成員們討論所遇到的種種難題。謝謝曾宇晟學長和王國振學長的指導，讓我不論在修課或研究上都能有更多的收穫。謝謝陳之悠學長、黃筱珊學姊、沈孟維學長和許博淵學長，你們細心的指導和寶貴的意見讓我受益良多。謝謝陳奕均同學、洪瑩蓉同學、許博雄同學和廖元歆同學，與你們不論一起上課研究或是聊天娛樂都會是我寶貴的回憶。還要謝謝蔡政君學長、孟勳、克嘉、英佑和亮齊等學弟，你們的幫忙讓我的實驗室生活能順利渡過。

另外，我要感謝交大棒球隊黃衫楹教練和全體成員們，你們讓我有個愉快且充實的大學和研究所生活。最後，我要感謝怡婷和我的家人們，你們的默默支持，是我能夠完成學業的最大動力。

在此，謹把這篇論文獻給所有愛我與我愛的人。

Contents

List of Figures

List of Tables

Chapter 1. Introduction

Video applications have been everywhere since advances of network bandwidth and wireless access techniques. With the prosperity of portable devices, digital televisions and internet videos, the applications of digital video become diversified. The state-of-art coding standard H.264/AVC [1] achieves high performance in bit-rate savings for these applications. Video encoder provides different bit-streams for different demands of video size, quality and frame rate. To further integrate the processing of different video demands, SVC, developed by the Joint Video Team (JVT) of ISO/IEC Motion Picture Expert Group (MPEG) and ITU-T Video Coding Expert Group (VCEG), delivers flexible scalabilities in temporal, spatial and quality domains by a single bit-stream [2]. Bit-stream of SVC is integrated at once with the benefits of coding time and complexity consumption. For the receiver side, decoder must guarantees the reconstruction of various scalabilities.

1.1. Motivation

Recently, real-time processing with HD resolution is basically boosted for video applications. To meet the processing specification, video decoders are commonly integrated to ASIC chips [3][4][5][6][7]. These designs were implemented within H.264/AVC standard however did not support the scalable extension.

The high definition and various scalabilities make SVC decoder much more complex than H.264/AVC and previous standards. Critical issues from external memory bandwidth, processing throughput, and area cost need to be solved in SVC chip designing. In order to deal with those problems, this thesis makes efforts on the

investigation of design and implementation methodology for SVC decoder.

1.2. Thesis Organization

The organization of this thesis is described as follows. Chapter 1 makes a brief introduction of SVC and motivation of this work. Chapter 2 gives an overview on SVC standard and introduces the mechanism of inter-layer prediction in SVC. In Chapter 3, the overall architecture in this work is presented with the decoding flow analysis and data path optimization. Chapter 4 shows the hardware implementation and experiment results of the proposed SVC decoder. Finally, the conclusion and future work will be given in Chapter 5.

Chapter 2. Overview of SVC Standard

With the stronger demands of multimedia applications, video coding techniques have been constantly concerned as well. The compression capability has getting more powerful with the more advanced video coding techniques. Fig. 2.1 shows the evolution history of video coding standards. For the multiple device network applications, the state-of-art standard, SVC, further provides flexible scalabilities for applications. The term "scalability" means that certain parts of the bit-stream can be removed in order to adapt to the requirements of receivers.

Fig. 2.2 gives a scalability demonstration of SVC. For encoder, temporal, spatial, and quality scalabilities are integrated in single bit-stream. This bit-stream provides selective ranges from multiple levels of frame rate, frame size, and video quality. The bit-steam is transmitted to every device by broadcasting within the communication networks. Each backend device then adaptively extracts the corresponding parts of bit-stream for their specific application. For example, HDTV extracts the highest level scalabilities with its customized purpose and high-speed transmitting channel. Cell phone extracts the lowest level part of bit-stream on the contrary.

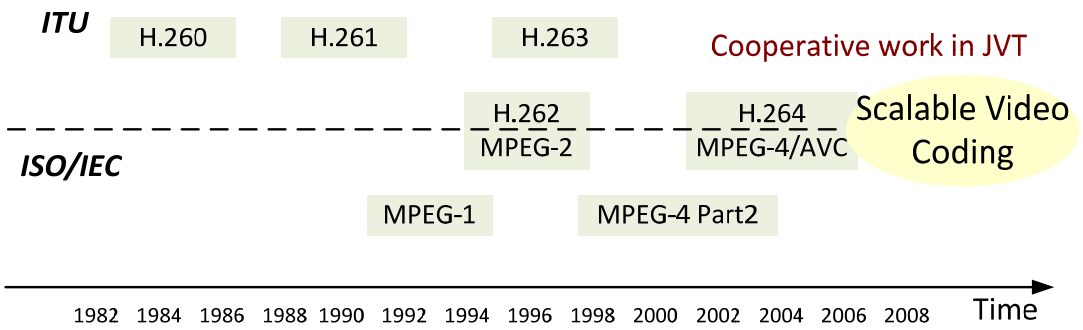


Fig. 2.1. History of video coding standards

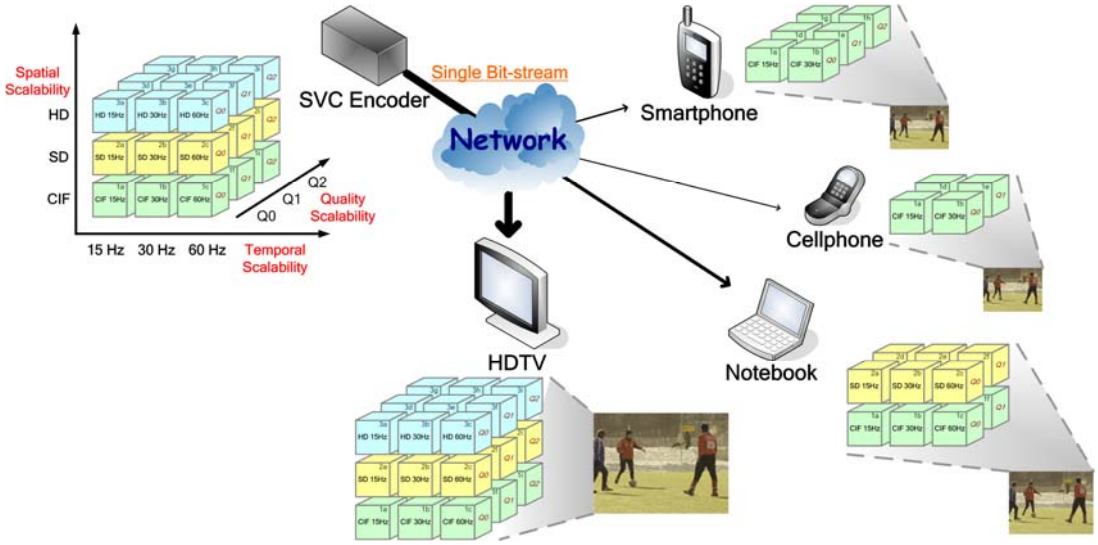


Fig. 2.2. Scalability demonstration of SVC

2.1. Fundamentals of SVC

SVC is inherited from H.264/AVC with its advanced coding techniques. In addition, it provides scalabilities in three domains where the architecture of encoder is shown in Fig. 2.3. In the beginning, the original texture is down-sampled to supportable image resolutions. Two spatial layers are configured to achieve the spatial scalability in this case. The functional behaviors within shaded region are equal to H.264/AVC. Different from H.264/AVC, the upper resolution layer (spatial enhancement layer) uses the upsampled information from lower resolution layer (spatial base layer) as prediction candidates. That is because with the scaled relationship between layers, scaled motions vectors, residuals, and texture value from base layer may predict more accurately. The prediction scheme is called inter-layer prediction (ILP) which achieves high coding performance for SVC. Progressive SNR refinement coding produces multiple levels of transformed coefficients to provide quality scalability. Because temporal scalability is already enabled by H.264/AVC, SVC only provides supplemental enhancement information to improve its usage.

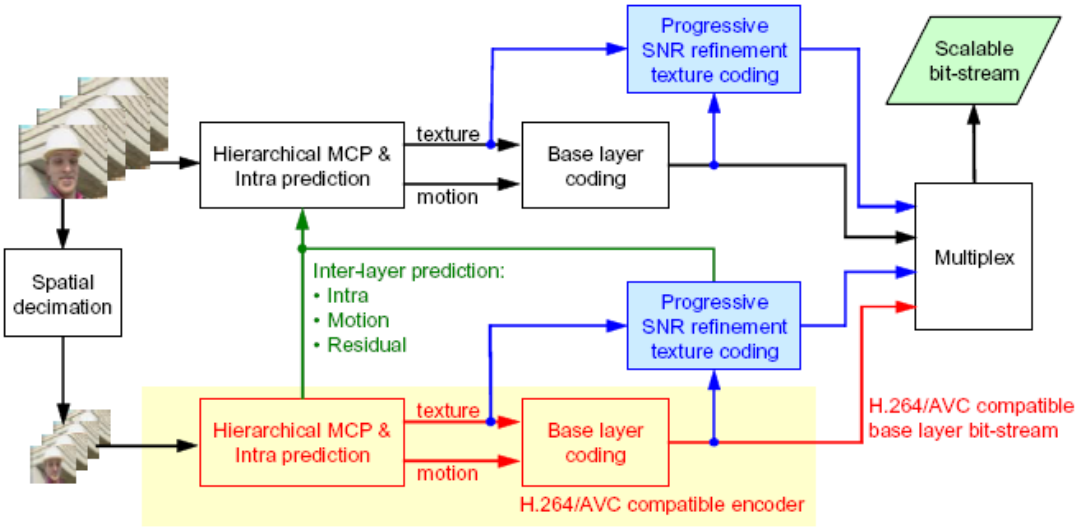


Fig. 2.3. Block diagram of SVC encoder with two spatial layers

The SVC Amendment of H.264/AVC specifies three profiles for scalable video coding [8]: Scalable Base, Scalable High, and Scalable High Intra. The coding tools supported by each profile are listed in Table. 2.1. The Scalable Baseline profile is mainly used for applications with low decoding complexity. The supportable resolution ratio between successive layers is restricted to 1.5 or 2. Thus, the complexity of ILP can be reduced by the macroblock-aligned spatial mapping. In Scalable High profile, arbitrary spatial resolution ratios can be supported. Except for spatial scalability, temporal and quality scalability are supported without any restriction in both Scalable Baseline and Scalable High Profile. Scalable Baseline profile and Scalable High profile retains all the coding tools from Baseline Profile and High Profile of H.264/AVC, respectively. Noticeably, Scalable Baseline profile supports B Slices, weighted prediction, CABAC entropy coding, and 8x8 luma transform in enhancement layers, although these coding tools are not supported in Baseline profile of H.264/AVC. Scalable High Intra profile is mainly considered for professional applications which contain only IDR pictures. Except for this restriction,

the coding tools are the same with Scalable High profile.

Table. 2.1. Profiles of SVC

Coding Tools	Baseline	High	High Intra
Arbitrary Resolution Ratio	N	Y	Y
Inter-laced Coding	N	Y	Y
Temporal Scalability Without Restriction	Y	Y	Y
Quality Scalability Without Restriction	Y	Y	Y
H.264/AVC Baseline Profile	Y	Y	Y
H.264/AVC High Profile	N	Y	Y
B Slices	Y	Y	Y
Weighted Prediction	Y	Y	Y
CABAC Entropy Coding	Y	Y	Y
8x8 Luma Transform for EL	Y	Y	Y
Only IDR pictures	N	N	Y

2.2. Features of SVC Decoder

SVC decoder not only inherits the coding techniques from H.264/AVC but also guarantees the reconstruction of spatial, temporal, and quality scalabilities as shown in Fig. 2.4. In the beginning, the extracted bit-stream with necessary scalability is entropy decoded and the parsed data are used as the source of further reconstruction. Quality scalability is mainly processed with the refinement of scaling coefficients, which brings quality disparity for different layers. With the utilization of inter-layer prediction, reference data from base layer will be upsampled as the predicted contents. The inter-layer prediction mechanism will be introduced in next section. Similar to the existing H.264/AVC standard, intra prediction and motion compensation are used in SVC with the spatial and temporal locality. The pixel sample reconstruction is

accomplished by adding residuals to those predicted samples. Finally, deblocking filter is applied to erase the blocking effect.

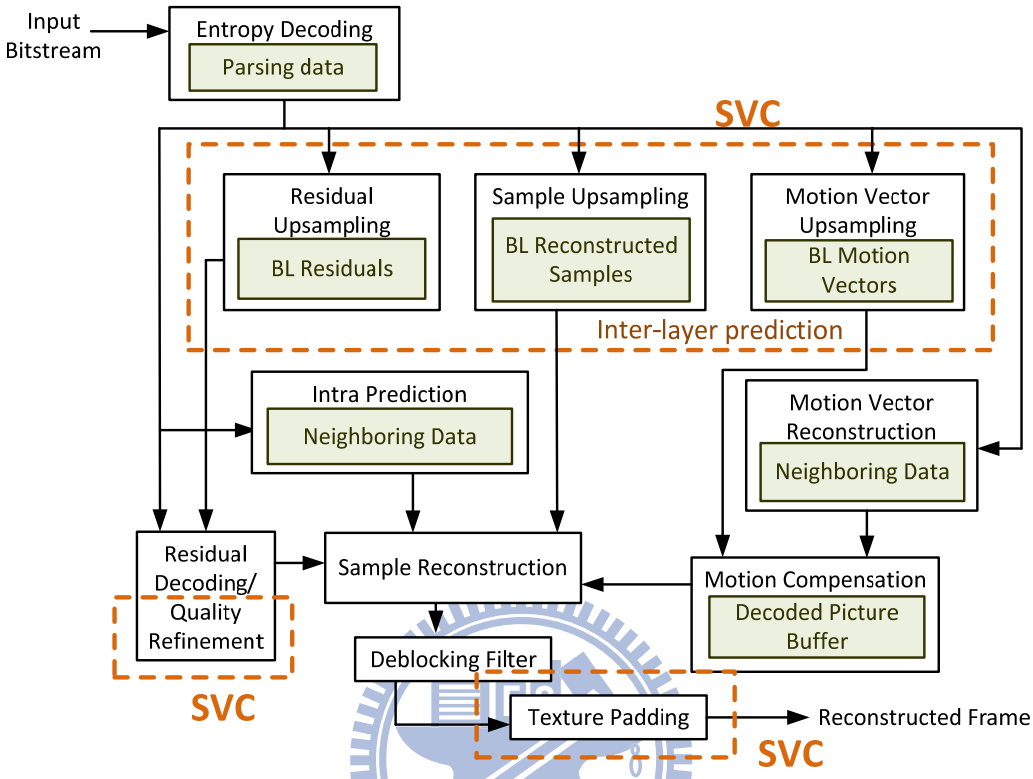


Fig. 2.4. Block diagram of SVC decoder

2.2.1. Single-loop Decoding

Single-loop decoding [9] is an advanced mechanism to lower the decoding complexity. Because the Inter-coded macroblocks only use the motion information of base layer, motion compensation in base layer can be avoided. Only Intra-coded and IntraBL-coded macroblocks are reconstructed and deblocked in spatial base layers. Thus, the complexity can be reduced due to the single motion compensation loop among spatial layers. To further integrate this mechanism with inter-layer intra prediction, texture padding is applied to extending the reference sample boundary for those regions are un-available for upsampling.

2.2.2. Key Picture Concept and Combined Scalability

Except for single-loop decoding, “key picture concept” and “combined scalability” is added to simply the coding flow of SVC. Key picture concept provides appropriate trade-off between stream-truncated drift and coding efficiency for hierarchical prediction structures. Beside, based on combined scalability concept, only quality refinement is needed inside the same dependency layer. Thus, multi-loop reconstruction flow for quality layers can be avoided by single prediction generation. For detail of key picture concept and combined scalability, please refer [10] [11].

2.2.3. Data Dependency of SVC Decoder

In summary, with simplifications mentioned above, the data dependency of SVC decoder is shown in Fig. 2.5. The solid lines represent reference direction of coefficients where should not be across spatial and temporal layers. The dotted lines indicate the reference direction of predictions: for inter-layer prediction, top quality layer reconstruction is referenced; for motion compensation, P slice references the data of bottom quality and B slice references the top ones. Based on single-loop decoding, motion compensation only executes in the top spatial layer which means there is no temporal referencing in spatial base layer. The referencing structure mentioned above makes SVC decoder simpler to be implemented.

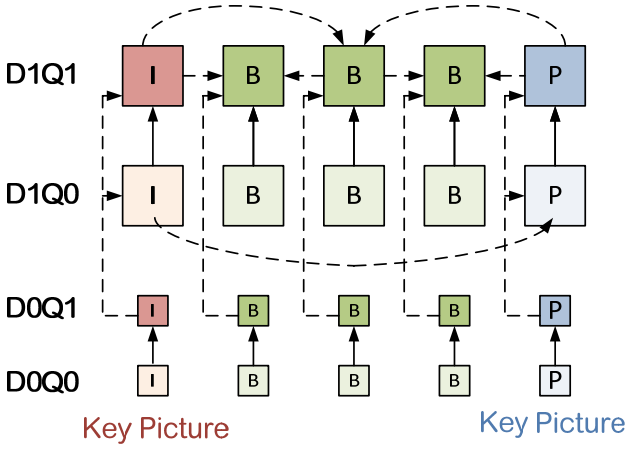


Fig. 2.5. Data dependency of SVC decoder

2.3. Inter-layer Prediction

Inter-layer prediction (ILP) is the novel coding technique which applied in SVC standard. From experimental results, it considerably lowers the bit-rate by taking advantage of the content locality between spatial layers. In addition to dyadic applications, ILP mechanism also works under the situations of arbitrary hierarchical frame sizes, which is defined in Scalable High profile. In reality, ILP adopts "Extended Spatial Scalability (ESS) [12]" approach to deal with the macroblock and data un-aligned of two spatial layers. In this section, the practical mechanism of ESS-utilized ILP in SVC decoder will be introduced.

2.3.1. Flags and Macroblock Types

The reconstruction procedure of ILP is illustrated in Fig. 2.6. In this figure, the identification of Base_mode_flag is processed in the beginning of this flow. Once this flag is set true, the current spatial enhancement macroblock will be totally inter-layer predicted by contents from base layer. Therefore, the enabling of this flag can be regarded as the appearance of a novel macroblock type "BaseMode", which has not existed in H.264/AVC or earlier standard.

After the BaseMode type is qualified, this flow goes to IntraBL identification. In this process, SVC standard chooses sixteen feature points of enhancement layer as representational positions of current macroblock. As shown in Fig. 2.7, these sample positions are going to map their specific base layer positions by calculation. One thing we can imagine is that these points belong to their corresponding macroblocks in base layer. By definition, if all the corresponding macroblocks are reconstruct-able, the current macroblock type is named as "IntraBL". Otherwise, if any corresponding macroblock of these sixteen points is inter-coded, the current macroblock is called as "InterBL". InterBL and IntraBL are the two subsets of BaseMode type. IntraBL type

adopts the upsampling samples as prediction of current macroblock entirely, and it would not be accompanied with inter-layer motion (ILMP) and inter-layer residual (ILRP) mechanisms. In the other hand, InterBL type uses the information of base more directly. Under this type, every sample within "reconstruct-able" (IntraBL or Intra) macroblocks in base layer is predicted by upsampling of inter-layer texture. And the other samples will be predicted by result of motion compensation with the scaled motion vectors.

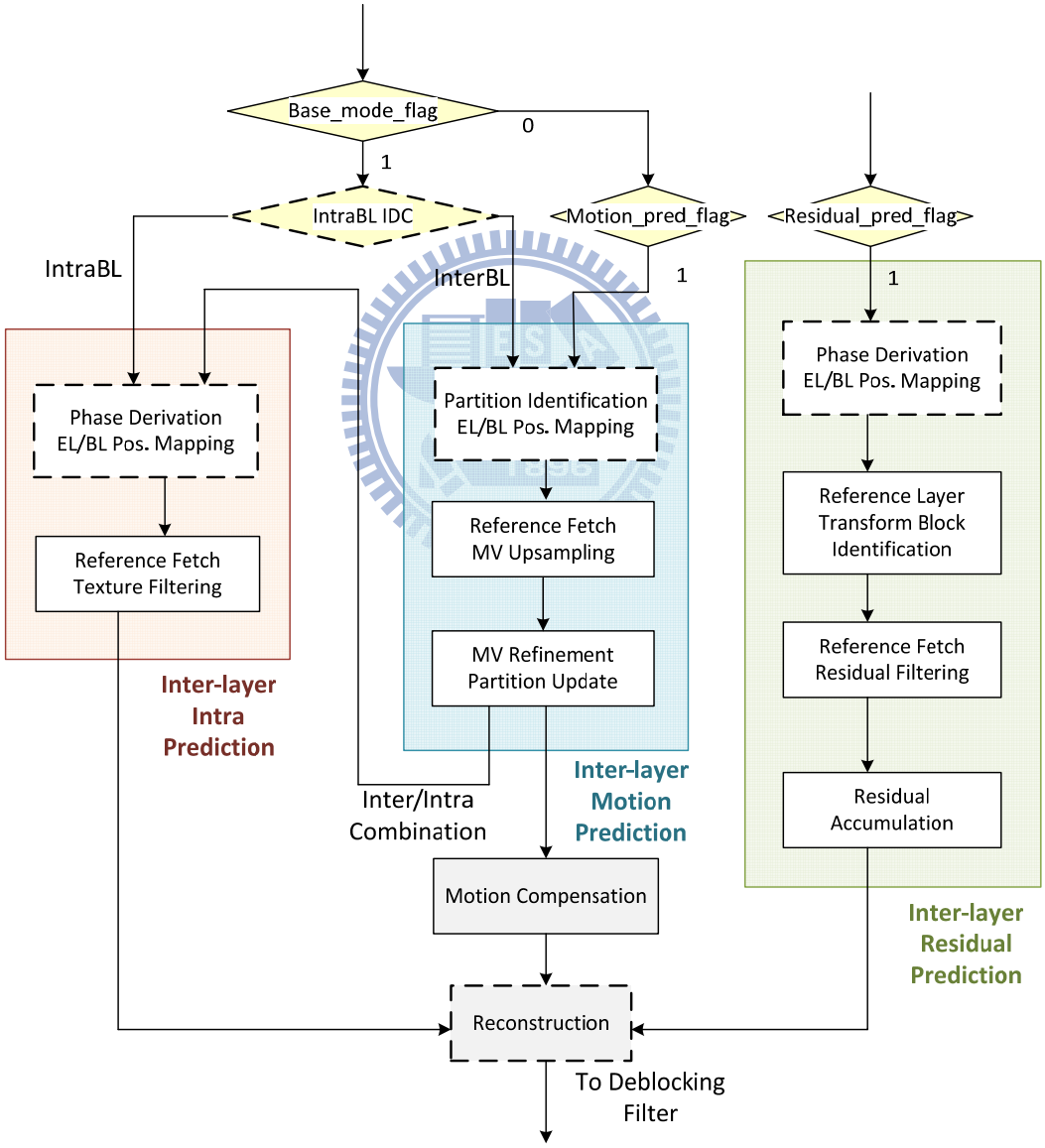


Fig. 2.6. Reconstruction flow of ILP

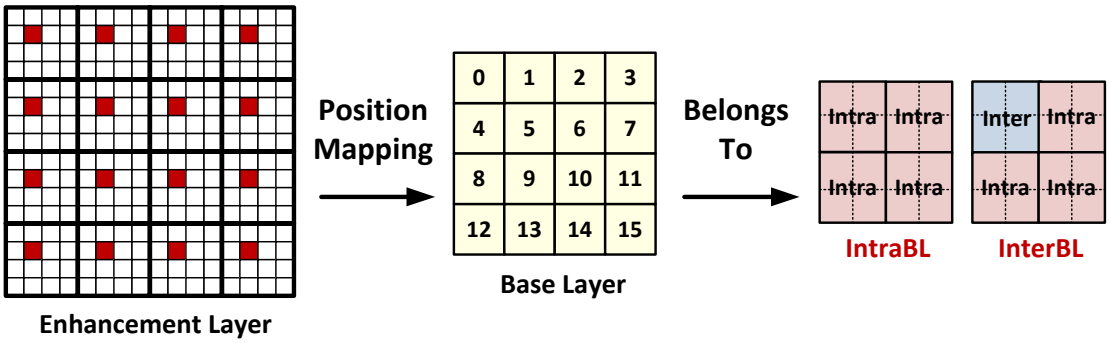


Fig. 2.7. EL/BL position mapping of IntraBL identification

2.3.2. Calculation of Corresponding Spatial Positions

Compared with dyadic spatial scalability, non-dyadic one suffers from the issue of indirect position mapping between two spatial layers. ESS adopts the scheme called "Calculation of Corresponding Spatial Positions (CCSP) [12]" to map corresponding samples between two successive layers. The concept of this derivation is to exploit inter-layer resolution ratios to determine the corresponding positions and upsampling parameters. The mathematical term of CCSP is shown as below

$$BL_m = \{[m \times D_m) + A_m] >> (S_m - 4)\} - d_m$$

$$\text{where } m \in \{x,y\} \text{ and } 0 < x < FrameWidth,\ 0 < y < FrameHeight \qquad (2.1)$$

m is the current sample absolute position in enhancement layer. S_m and D_m are parameters which refer to the resolution ratio between spatial layers. A_m and d_m are terms added to increase the arithmetic precision. With parameters mentioned above, BL_m can be finally derived where indicates the absolute position with 1/16th precision in base layer. This term is derived to infer the corresponding sample positions and interpolation phases for upsampling. The inferred parameters play significant roles in the overall ILP mechanism. In Fig. 2.6, blocks within dotted line are the components accompanied with CCSP process directly.

2.3.3. Inter-layer Intra Prediction

Inter-layer Intra Prediction (ILIP), one of the mechanisms of ILP, uses the up-sampled deblocked samples as the texture prediction. The up-sampling operation applies a separable 4-tap and bi-linear poly-phase interpolation filter for luma and chroma respectively. The numerator tap values for these dedicated filters are shown in Table. 2.2, and a five-bit rounding right shift operation will be applied for producing normalized results.

Table. 2.2. Poly-phase coefficients of upsampling filter

(a) Bi-linear filter;

phase	coefficients	
	C_0	C_1
0	32	0
1	30	2
2	28	4
3	26	6
4	24	8
5	22	10
6	20	12
7	18	14
8	16	16
9	14	18
10	12	20
11	10	22
12	8	24
13	6	26
14	4	28
15	2	30

(b) 4-tap filter

phase	coefficients			
	C_{-1}	C_0	C_1	C_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	-3	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

Corresponding positions of target reference samples in base layer are derived from CCSP, which is listed below

$$BLPOS_m = BL_m >> 4 \quad (2.2)$$

Where BL_m is the derived term in Eg. (2.1) In addition, the target interpolation phases are also derived according to BL_m

$$BLPHS_m = BL_m \,\&\, 15 \quad (2.3)$$

For an enhancement layer position m, its corresponding 4-tap poly-phase filtering equation is

$$P = Sp_{(BLPOS_m - 1)} \times C\text{-}1_{(BLPHS_m)} + Sp_{(BLPOS_m)} \times C0_{(BLPHS_m)} + Sp_{(BLPOS_m + 1)} \times C1_{(BLPHS_m)} + Sp_{(BLPOS_m + 2)} \times C2_{(BLPHS_m)} \quad (2.4)$$

where $Sp_{(x)}$ is the sample value in absolute position x and $Cn_{(y)}$ is the n^{th} coefficient to phase index y. SVC adopts the interpolation in Eq. (2.4) in horizontal and vertical dimensions to produce the texture prediction. The process fully takes advantage of the corresponding parameters derived from CCSP with concept of position mapping. From related experimental results, the poly-phase filter indeed holds good prediction performance with prediction generation.

2.3.4. Inter-layer Residual Prediction

The prediction generation of Inter-layer residual prediction (ILRP) is similar to ILIP. Being a little different, ILRP adopts bi-linear filter to interpolate both luma and chroma samples. Moreover, because discrete cosine transform (DCT) is based on 4×4 or 8×8 blocks, filtering will not take place in transform block boundaries in SVC standard. The block-wise interpolation equation is listed below

$$P = (Transformblock_{(BLPOS_m)} == Transformblock_{(BLPOS_m+1)}) \; ? \; (Sp_{(BLPOS_m)} \times C0_{(BLPHS_m)} + Sp_{(BLPOS_m + 1)} \times C1_{(BLPHS_m)}) \; : \; [(BLPHS_m < 8) \; ? \; (Sp_{(BLPOS_m)} << 5) : (Sp_{(BLPOS_m + 1)} << 5)] \quad (2.5)$$

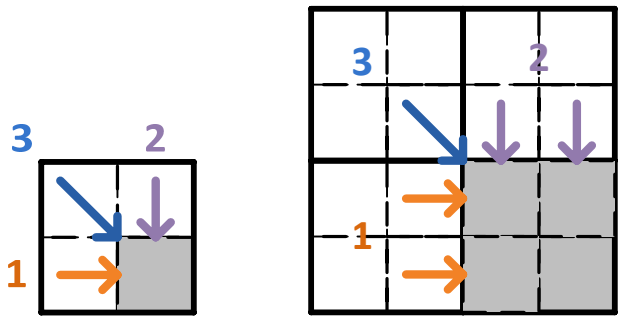
From Eq. (2.5), we can observe the filtering will be skipped if the input pairs of bi-linear filter are within different transform blocks.

2.3.5. Inter-layer Motion Prediction

Motion vector is another information which delivers the similarity between spatial layers. As a result, inter-layer motion prediction (ILMP) mechanism uses the scaled motion vectors to acquire the better solution in temporal locality. The upsampling of motion vectors can be divided to several steps as followings.

1) Find the corresponding motion vectors

ILMP generates all 4×4 blocks in current macroblock with up-sampled motion vectors. The spatial mapping of successive layers is also shown in Fig. 2.7. The sixteen corresponding positions are located at sixteen corresponding 4×4 blocks which should contains motion vectors in every block. However, if the corresponding block is located at Intra or IntraBL region, there is no motion vector available. Thus, SVC adopts hierarchical strategy to fill the corresponding motion vectors of un-available regions. Fig.2.8 shows the 4×4-based and 8×8-based mapping. Motion vector filling is processed first in 4×4 level. Single un-available block applies the neighboring motion vectors in priority of horizontal, vertical, and diagonal directions. If all of the motion vectors inside 8×8 are not available, 8×8-based processing is further applied. After the filling process, these base layer motion vectors at co-located blocks are going to be upsampled then.



(a) (b)

Fig.2.8. Filling strategy for un-available blocks: (a) 4×4 process; (b) 8×8 process

2) Motion vector upsampling

The motion vector upsampling equation is shown as below

$$MV_{EL} = (MV_{BL} \times Dmv + 32768) >> 16 \quad (2.5)$$

where MV_{BL} is the reference motion vector value in base layer, and Dmv is the spatial parameter which represents the ratio between layers. MV_{EL} is the target derived motion vector in the current macroblock.

3) Refinements of reference index and motion vector

Two level steps are needed to refine the scaled motion vectors and the reference indexes. Reference indexes are used to indicate the reference direction (forward or backward) in motion compensation which is in a 8×8 block set. However, upsampled motion vectors with reference indexes from base layer may cause the inconsistency. Thus, reference indexes are further reorganized in the reference list domain with the priority shown in Fig. 2.8 (a).

The second level refinement is the motion vector integration technique. SVC averages "similar" motion vectors of adjacent 4×4 blocks within the same reference indexes to generate a new motion vector as shown in Fig. 2.9. The definite threshold of the term "similar" is represented by Eq. (2.6).

$$| MVx0 - MVx1 | + | MVy0 - MVy1 | \leq 1 \quad (2.6)$$

With this process, block partitions are further integrated and become simpler.

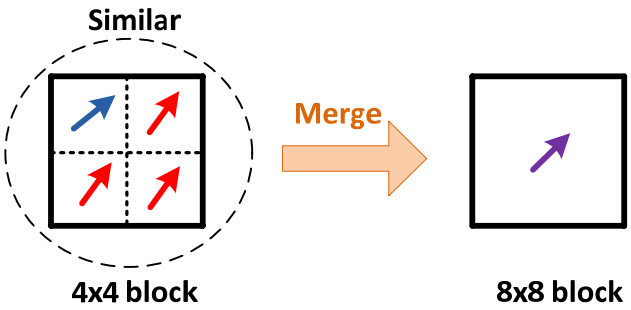


Fig. 2.9. Example of motion vector integration

2.4. Related Works

Since SVC has been standardized in 2007 [2], more and more researches and analysis about SVC architecture are published recently. Memory storage is an important issue which is caused by additional data dependency of SVC. For memory architecture analysis, [13] analyzed the spatial layer decoding flow in consideration of inter-layer prediction data. In this research, frame-base decoding flow is most efficient for spatial layer decoding with memory characteristics. [14] also proposed a memory architecture for SVC which focus on the size reduction of on-chip memory. For inter-layer prediction, the most different part from H.264/AVC, [15] proposed a cost-efficient residual prediction hardware architecture for encoder. However, it does not mention the detail implementation of residual prediction. For the whole chip integration, designs [3] ~ [7] implemented and optimized H.264/AVC decoder already by focusing on the power consumption, on-chip memory demand, or gate count issues. [16] is the first published work about the integration of a SVC decoder which will be compared with this work later.

Chapter 3. Framework of SVC Decoder

In this Chapter, the analysis and architecture of the proposed SVC decoder chip will be demonstrated as follows. We first consider the most appropriate decoding flow among three dimensions: spatial layer, quality layer and texture padding procedure. Each dimension is analyzed in consideration of memory bandwidth, decoding time or area cost. Afterward, the overall 4-stage pipeline architecture will be presented with design methodologies. There are architecture optimization and comparison to improve the area cost of this work. Then the external memory access system with the efficient data transaction will be introduced in the final section.

3.1. Design Specification

In Chapter 2, we have introduced the profiles with coding tools of H.264/AVC Scalable extension. To achieve high performance in video decoding, the proposed SVC decoder supports main features in Scalable High Profile. Because of the support to high throughput decoding for high definition applications, the limited resources are hardly able to cope with infinite scalabilities. With restrictions to the layers of scalabilities, specifications of the proposed SVC decoder chip are listed as follows.

(a) Supports bitstreams of H.264/AVC High Profile.

(b) Supports at most 3 spatial layers from QCIF to HD1080p.

(c) Supports at most 3 quality layers.

(d) Supports all GOP sizes under 8.

(e) Supports arbitrary resolution ratios between spatial layers with ESS.

Under the restrictions mentioned above, the proposed SVC decoder successfully

reconstructs the Scalable High profile bit-streams encoded by reference software JSVM 9.14 [17]. In addition, the critical specification of this work is to simultaneously support 3 spatial layers (CIF-SD480p-HD1080p) and 3 quality layers decoding at 60 frames per second, which is under the working frequency of 135MHz.

3.2. Decoding Flow Analysis

With the additional feature of multiple-domain scalabilities, the traditional decoding flow of H.264/AVC is no longer appropriate for scalable extension. Therefore, an efficient decoding flow must be investigated to deal with the additional complexity from SVC. Moreover, memory bandwidth is a typical issue for video chip design and significantly influenced by the processing procedure. In this section, some strategies of decoding flow will be analyzed in consideration of memory access. A most efficient decoding procedure will be utilized for the proposed SVC decoder. Finally, we will introduce the memory controller protocol which plays an important role in external data access.

3.2.1. Spatial Layer Decoding

Spatial scalability is the most distinct part between H.264/AVC and SVC. With utilization of inter-layer prediction, data dependency exists between spatial layers. Among the related researches, [13] has analyzed the spatial decoding flow based on memory access from H.264/AVC to SVC. From this research, frame-based spatial decoding flow has better characteristics than other strategies with appropriate memory partition

3.2.2. Quality Layer Decoding

In SVC, quality scalability is achieved by using coarse-grain scalability (CGS) or medium-grain scalability (MGS). Coefficients in enhancement layer are reconstructed by the sum coefficients from base layer and delta coefficients from entropy decoder.

With quality scalability, the reconstruction capability of SVC decoder is challenged by the increasing amount of quality layers. To deal with all quality layer coefficients, the decoding flow plays an important role in both processing time and memory access aspects.

1) Frame-based quality decoding

In traditional flow [2], quality layer decoding proceeds in frame-level approach as shown in Fig. 3.1. Enhancement layer frames are reconstructed after the previously decoded frame in base layer. However, this flow introduces significant external memory accesses. For one thing, reference data for prediction generation must be loaded if necessary (I/P slice in top spatial layer). In this case, external memory access for prediction reference data doubles with multiple quality layers. For the other, coefficients in base layer are also stored for being referenced in enhancement layer. This amount of quality coefficients is too large to store in internal memory. For example, memory space of 8160×384×2 bytes are totally needed for the frame size of HD1080p. Therefore, considerable external memory access occurs for processing every quality layer. Table. 3.1 lists the external memory bandwidth requirement in frame-based quality decoding flow. In Table. 3.1, the “others” item means the write out data which includes inter-layer data or frame pixels.

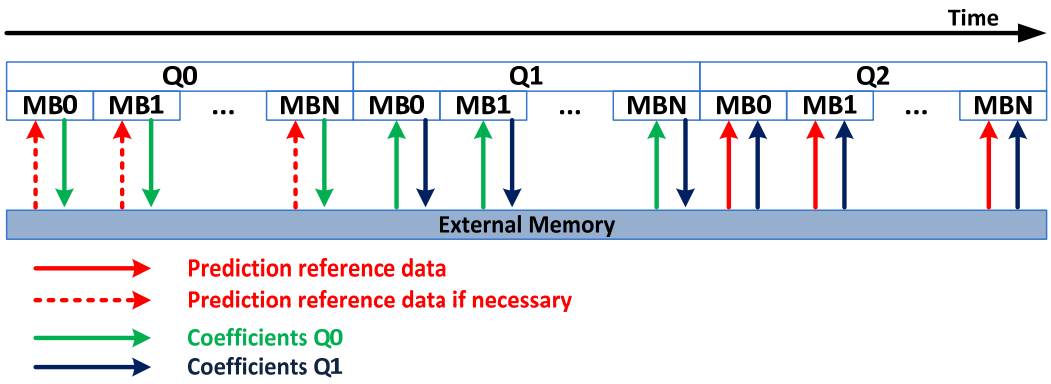


Fig. 3.1. Frame-based quality layer decoding flow

Table. 3.1. Memory bandwidth requirement in frame-based quality decoding flow

Item/Sequence	Blue-Sky	Tractor	Pedestrian-area
Prediction Reference	31.51MBps	33.14MBps	33.31MBps
Quality Coefficients	160.9MBps	160.9MBps	160.9MBps
Others	43.48MBps	43.53MBps	43.32MBps
Total	235.89MBps	237.57MBps	237.53MBps

* GOP : 8 / Frame-rate : 30fps / QP: 12-22-32 / Frame-size : CIF-480p

2) MB-based quality decoding

MB-based decoding is another approach for quality decoding. Coefficients from different quality layers are reconstructed within the same macroblock in successive order, as shown in Fig. 3.2. It is not necessary to put the quality coefficients out because they will be referenced within the same macroblock process. Instead, internal memory can handle the amount of coefficients (384x2 bytes). As a result, no external memory access for quality coefficients is required in MB-based quality layer decoding. The only data to be accessed from external memory is the reference data for prediction generation. The external memory bandwidth requirement of MB-based quality decoding flow is listed as Table. 3.2.

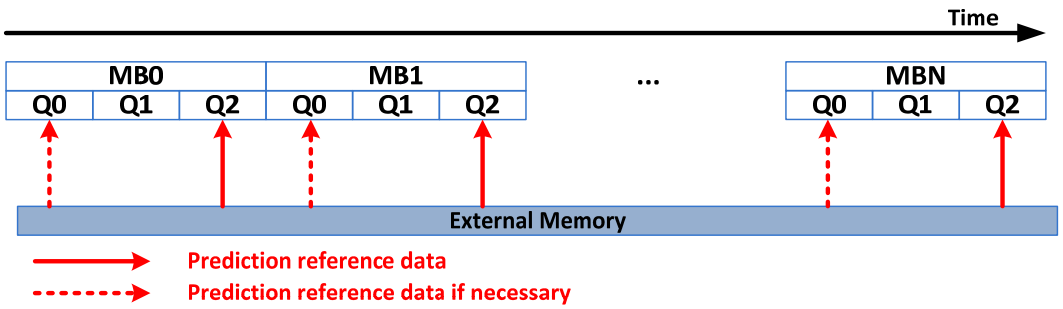


Fig. 3.2. MB-based quality layer decoding flow

Table. 3.2. Memory bandwidth requirement in MB-based quality decoding flow

Item/Sequence	**Blue-Sky**	**Tractor**	**Pedestrian-area**
Prediction Reference	31.51MBps	33.14MBps	33.31MBps
Quality Coefficients	0	0	0
Others	43.48MBps	43.53MBps	43.32MBps
Total	74.99MBps	76.67MBps	76.63MBps

* GOP : 8 / Frame-rate : 30fps / QP: 12-22-32 / Frame-size : CIF-480p

3) One-Pass quality decoding

Although the access of external memory time is saved, the long decoding time for quality layers is still a problem in MB-based decoding. The long latency comes from the entropy decoding for DCT coefficients and the reconstruction path for residuals which increases by the quality layers. Fortunately, because the entropy decoding process of each quality layer is independent with others, all of the coefficients in each quality layer can be parses separately. Thus, the parallel entropy decoding is utilized to achieve higher throughput. In addition, with the combined scalability concept, the residual reconstruction path can be merged into single macroblock processing [11]. Thus, at most 66% macroblock processing cycles can be saved for the 3-quality-layer application as shown in Fig. 3.3.

Besides, because quality layers are processed in parallel, the prediction generation can be parallel processed with residual reconstruction. That means only single-loop is required for prediction generation. Thus, prediction reference data (i.e. reference pixels for motion compensation, base layer texture, base layer residual, etc.) from external memory are only fetched once in single macroblock processing. As a result, memory bandwidth can be further reduced as listed in Table. 3.3.

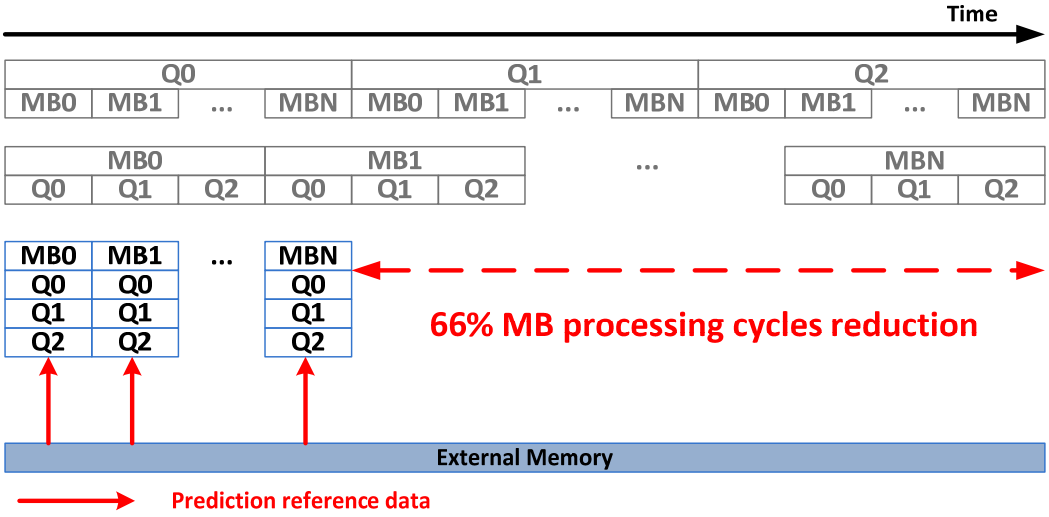


Fig. 3.3. Proposed one-pass quality layer decoding flow

Table. 3.3. Memory bandwidth requirement in One-Pass quality decoding flow

Item/Sequence	Blue-Sky	Tractor	Pedestrian-area
Prediction Reference	24.83MBps	25.63MBps	25.59MBps
Others	43.48MBps	43.53MBps	43.32MBps
Total	68.31MBps	69.16MBps	68.91MBps

* GOP : 8 / Frame-rate : 30fps / QP: 12-22-32 / Frame-size : CIF-480p

3.2.3. Texture Padding

Inter-layer intra prediction (ILIP) is one of the inter-layer prediction schemes as described in Chapter 2. The prediction in spatial enhancement layer is generated by upsampled texture in base layer. However, with the utilization of single-loop decoding flow, there might be un-available regions in the reference layer. That is because with this flow, it is not necessary to reconstruct pixels in Inter-coded macroblocks during spatial base layer processing. Thus, extensions for un-available regions in base layer are generated in case of invalid reference during upsampling. Fig. 3.4. shows the example of extensive padding for reference layer. In Fig. 3.4 (a). and Fig. 3.4. (b), the un-available region only needs to be constantly extended for both vertical and horizontal borders. In Fig. 3.4 (c). and Fig. 3.4. (d), diagonal down-right-like padding

is required for filling the referenced un-available region.

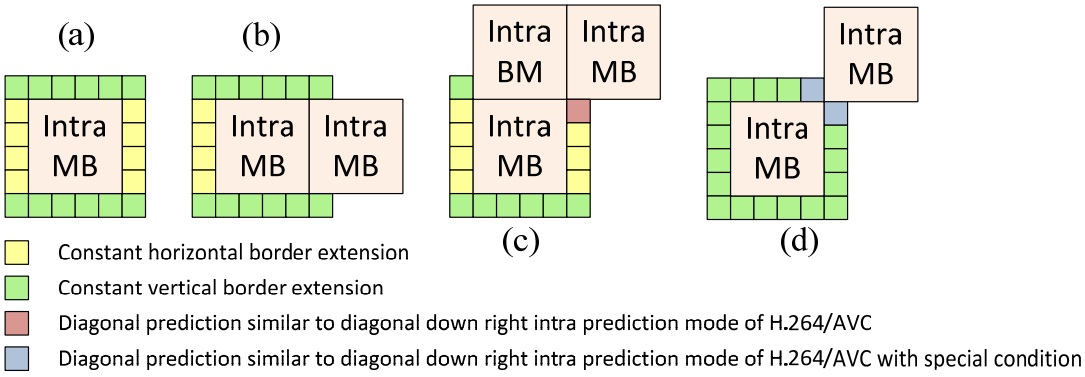


Fig. 3.4. Padding of intra macroblocks before upsampling [9]

In the traditional upsampling flow of ILIP [2], the padding action is processed in spatial enhancement level. The first step of upsampling in IntraBL macroblocks is to determine the reference co-located region in base layer. Once the reference region is determined, required reference texture from base layer would be fetched. For those regions cannot be reconstructed, they could be filled by padding process as shown in Fig. 3.4. After that, reference pixels with border extension will be upsampled by linear filter to be the prediction of enhancement layer. However, critical issue exists in the traditional upsampling flow. The most significant problem is that the padding complexity grows when the reference region gets larger or more arbitrarily organized. With the bit-stream where ILIP appears frequently, latency of padding may however be the burden of video decoding. As a result, we proposed the “BL-level” padding flow to solve problems mentioned above.

We move the padding procedure from enhancement layer to base layer, from IntraBL macroblocks to Inter macroblocks as shown in Fig. 3.5. In the proposed flow, macroblocks pre-pad in base layer to extend the reconstruction border and fills up the un-available Inter-coded regions. Fig. 3.6 shows the padding example within a base layer frame. With the pre-padding step, there will be no extension procedure in enhancement layer and time consumption can be obviously saved. Furthermore, the

padding procedure causes no additional time repaid in base layer decoding. For one thing, because Inter-coded macroblocks would not be reconstructed to pixel samples in spatial base layer with single-loop decoding concept, deblocking filter would also not be utilized in these macroblocks. Thus, conceptually, pre-padding for un-available region can be processed instead of deblocking during Inter-coded macroblocks. For another, deblocking filter component is commonly organized as an isolated pipeline data path in video decoder design. If we combine padding and deblocking in the same stage, the padding time can be hidden without additional penalty. From simulation results, 26% decoding time can be saved for IntraBL macroblocks.

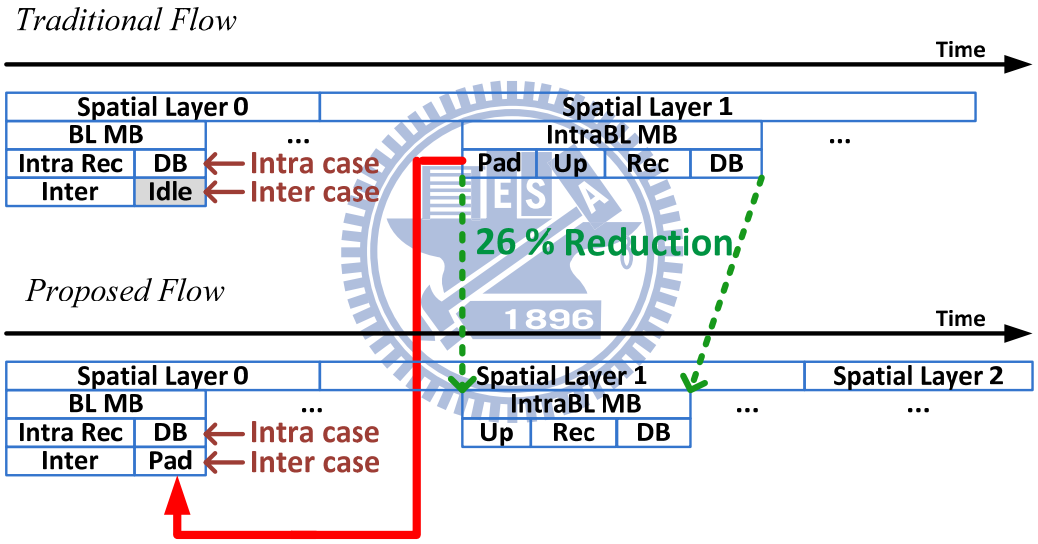
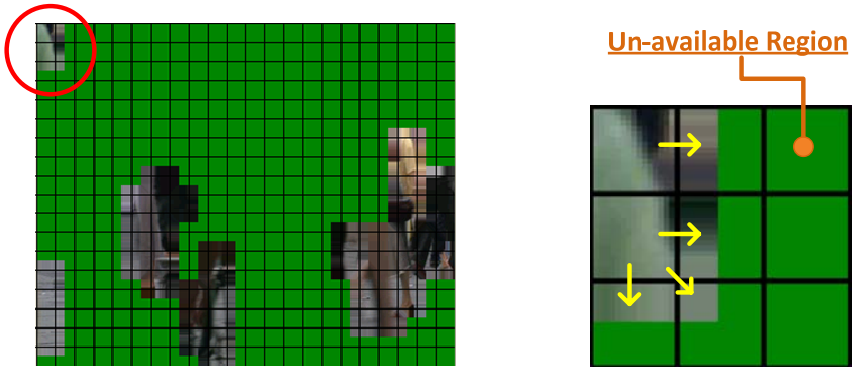


Fig. 3.5. Proposed BL-level padding



(a) (b)

Fig. 3.6. Example of padding in base layer: (a) CIF frame; (b) local enlarge

3.3. Proposed Framework

Based on the appropriate flow previously described, a novel framework of SVC decoder will be demonstrated in this section. An architecture overview of this work will be firstly introduced with the basic components of SVC decoder. After that, we will present the architecture optimization which lowers the complexity in decoding.

3.3.1. Overview of Architecture

To improve the decoding/encoding speed and lower the power consumption, partition the entire system into several pipeline stages is a common methodology for video CODEC chip design [18][19][20]. In this work, we adopt the 4-stage pipeline architecture to achieve high throughput decoding. Fig. 3.7 shows the overall architecture of the proposed SVC decoder and the organization of every stage is briefly demonstrated as follows.

The 1st stage is composed of parallel processing entropy decoding components and the syntax parser. As mentioned in previous section, at most 3 DCT coefficients from 3 quality layers are parsed in this stage. Fortunately, with the improvement of decoding throughput [21], only 2 set of CAVLD/CABAD decoder is required which did not cause too much gate count overhead. Besides, motion vector difference (MVD) is also produced by entropy decoding. The syntax parser delivers parsed parameters which are required for the following stages.

The 2nd stage is composed of the residual reconstruction path, the inter-layer predictor, the motion vector generator and other reconstruction elements. Coefficients from 1st stage1 are reconstructed to residual in this stage. The residual reconstruction path basically includes the pipeline chain with the process of inverse quantization, coefficients refinement, inverse transform, and residual accumulation. In this path, triple sets of this pipeline chain produce different quality residuals in parallel. The ILP

module generates inter-layer predictions by upsampling information from base layer which includes motion vectors, reconstructed pixels and residuals. If residual prediction mode is applied to current macroblock, ILP interpolator will be involved in the pipeline chain to obtain the final residual. Besides, MV Generator generates the motion vectors for bi-directional reference of current macroblock. With the derived motion vectors and partition sizes, reference pixel for motion compensation can be fetched at once.

Sample prediction of both Inter-coded and Intra-coded modes is generated in the 3rd stage. This stage includes interpolators which accomplish the 6-tap and bi-liner filtering for motion compensation and the prediction generator with all 4×4, 8×8, and 16×16 Intra modes. The produced prediction is then added with residuals from 2nd stage and forms the pre-deblocking samples. The final 4th stage is composed of deblocking filter and texture padding module. The deblocked data is passed to padding module to acquire the extension for inter-layer upsampling.

In this work, each pipeline stage is separated by pipeline ping-pong buffers. Ports of current and next stages are read/write in interleave way to avoid the access conflicts. To simply the control logic, we adopt synchronous approach to deliver the pipeline register which means the macroblock changes simultaneously for each stage. For external memory access, memory controller is utilized as the interface of decoder design and system bus.

Besides, under the target specification described in Section 3.1, averaging 227 cycles are mostly consumed for single macroblock decoding. It comes from the equation listed below.

$$227 = \left(\frac{135 \times 10^6}{\text{Frequency}}\right) / \left[\left(\frac{396 + 1350 + 8160}{\text{Total MBs in one frame}}\right) \times \frac{60}{\text{Frame rate}}\right] \quad (3.1)$$

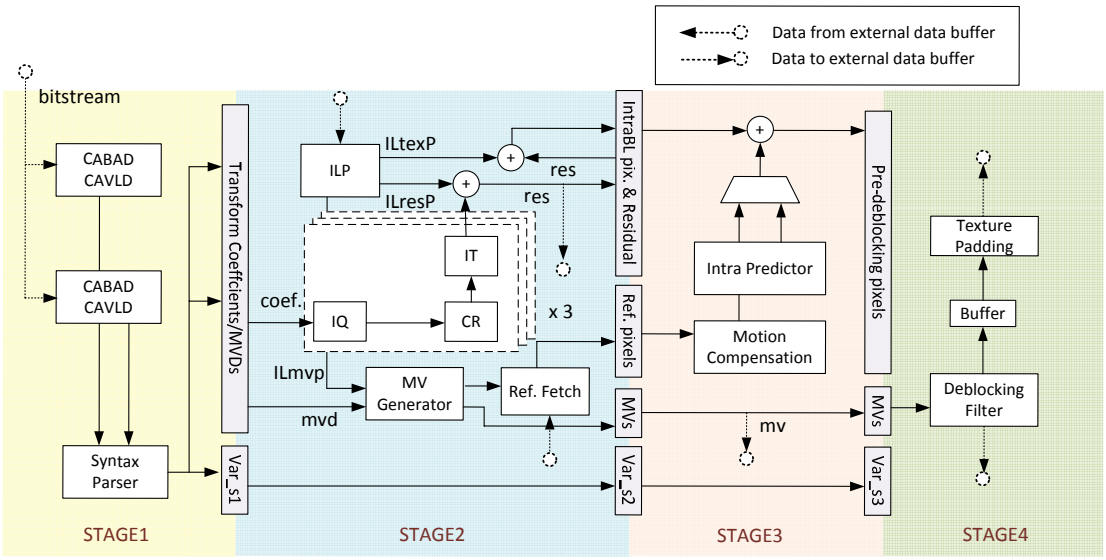


Fig. 3.7. Architecture of proposed SVC decoder

3.3.2. Optimization

1) Reconstruction path

In H.264/AVC decoding process, the code words with residual information will be reconstructed by entropy decoding, inverse quantization, and inverse transform. This reconstruction path is used to provide the dedicate prediction errors. However, the residual reconstruction path in SVC is different and more complex than H.264/AVC.

- Complexity issue in SVC

For one thing, SVC supports the quality scalability by involving the code words with transform coefficients of different QPs. Coefficients in enhancement layer are usually reconstructed by adding the sum of delta coefficients in the previous layers. This multi-level coefficients stratification makes the video quality scalable for different requirements. For the other, inter-layer residual accumulation is added in SVC. It further lowers the bit-rate by removing the redundancy in residual level. With the utilization of residual prediction, upsampled data must be added to the results of inverse transform. The two steps mentioned above are newly involved in SVC standard and the reconstruction flow is shown in Fig. 3.8.

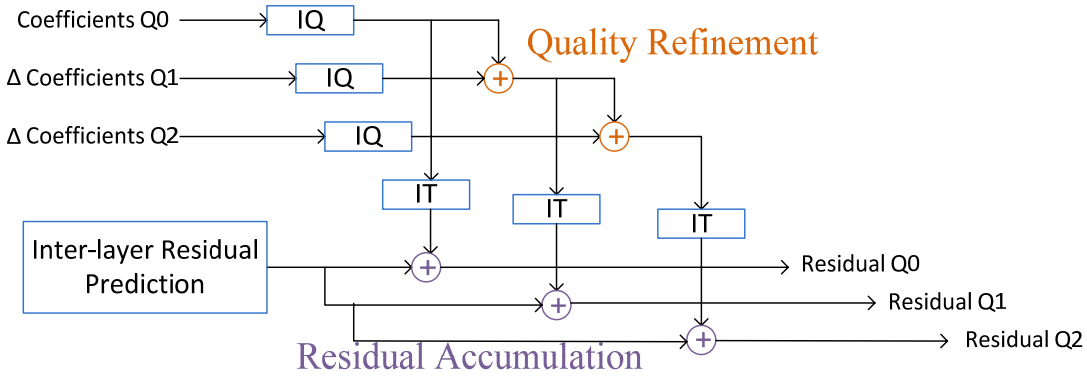


Fig. 3.8. Residual reconstruction flow with SNR scalability

- Shared interpolation components

Both inter-layer intra and inter-layer residual predictions utilize the poly-phase filter to generate the prediction samples. To further improve the efficiency of hardware utilization, the interpolation components of inter-layer intra and inter-layer residual prediction can be shared. First of all, we develop the 2-tap/4-tap switching basic interpolator, which can be reused for residual and texture upsampling (will be further described in Chapter 4). Thus, only one set of interpolator is required to process texture and residual upsampling.

In addition, the external memory buffer can be shared. It is used to restore the reference residual and texture from base layer. Inter-layer texture prediction needs the size of 19×19 luma pixel buffers and two 9×9 chroma pixel buffers to contain the maximum external data. Besides, inter-layer residual prediction needs 17×17 luma residual buffers and two 9×9 buffers to restore the reference residual. With the buffer reusing strategy, 47.2% buffer usage can be reduced.

Inter-layer intra and inter-layer residual prediction never exist during the same macroblock process except for InterBL macroblock type. Thus, for the InterBL type, processing intervals of texture prediction and residual prediction must be staggered to avoid the resource hazard from the shared interpolator and memory buffers.

- Pipeline chain of residual reconstruction

The residual construction flow is also commonly implemented by pipelining in video chip design. For SVC decoder, quality refinement and residual accumulation are added as shown in Fig. 3.8. Thus, these two steps are further involved in the pipeline chain. Fig. 3.9 shows the timing diagram of the residual reconstruction with pipelining. Each pipeline stage deals with a block of samples and then passes the result to next stage in every cycle. After the first 4 cycles, residuals from the same block are generated in successive cycles. The more samples within one block (more processing throughput), the less reconstruction time is needed. However, that is the trade-off result against the computation complexity of every stage.

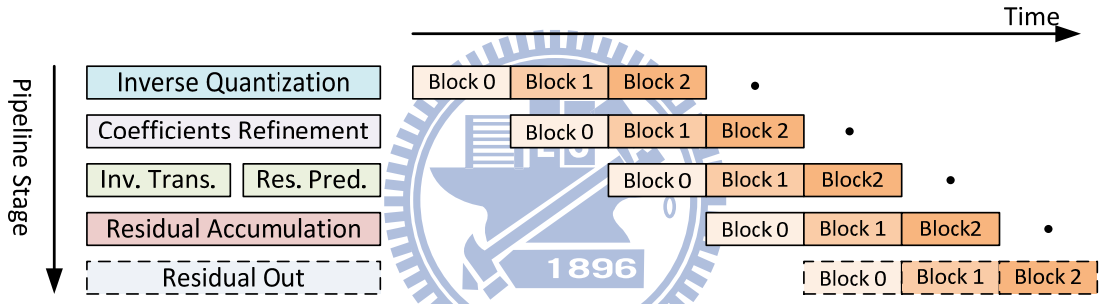


Fig. 3.9 Pipeline chain of residual reconstruction in SVC

Taking the numerous (at most 3 in this work) layers of SNR scalability into consideration, the scheduling and arrangement of residual reconstruction is a novel issue worth discussing. Therefore, two scheduling strategies will be analyzed here and one of them will be chosen for this work.

i. Serial-pipeline quality processing

In this strategy, quality coefficients from different layers are reconstructed in serial order within different timing intervals as shown in Fig. 3.10. Only one set of pipeline processing unit is required which is reused by every quality layer. A size of 384×2 bytes coefficients refinements buffer is required to restore the coefficients of quality base layer. Besides, with the shared interpolation component scheme,

inter-layer intra prediction is processed after the residual reconstruction flow with the same throughput. To meet the constraint of cycle budget (227 cycles), the processing parallelism is set to 8.

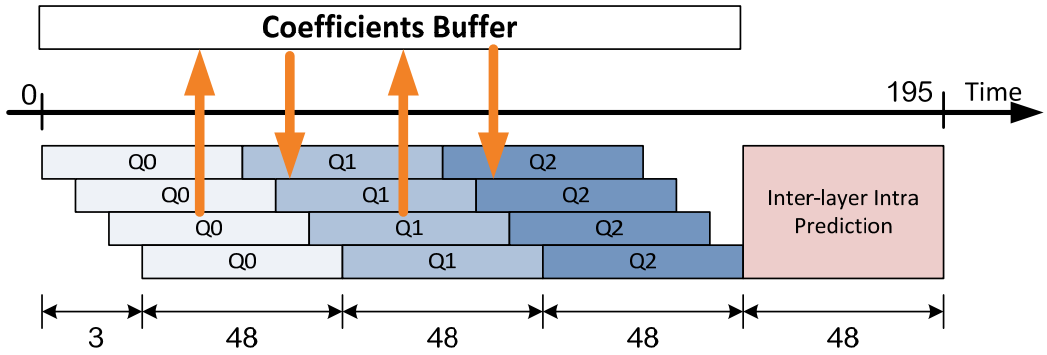


Fig. 3.10. Serial-pipeline chain quality processing

ii. Parallel-pipeline quality processing

Unlike the serial-pipeline strategy with one pipeline chain, parallel-pipeline strategy triples the pipeline chain to separately reconstruct the residual form different quality layers. However, quality enhancement layers need the coefficients from base layer during the coefficients refinement stage. The absolutely parallel reconstruction may cause hazards between layers because of the data dependency. Therefore, we delay the reconstruction procedure between layers with one cycle so that the coefficients can be passed across layers. Fig. 3.11 shows the scheduling of parallel-pipeline quality processing with the throughput of 4.

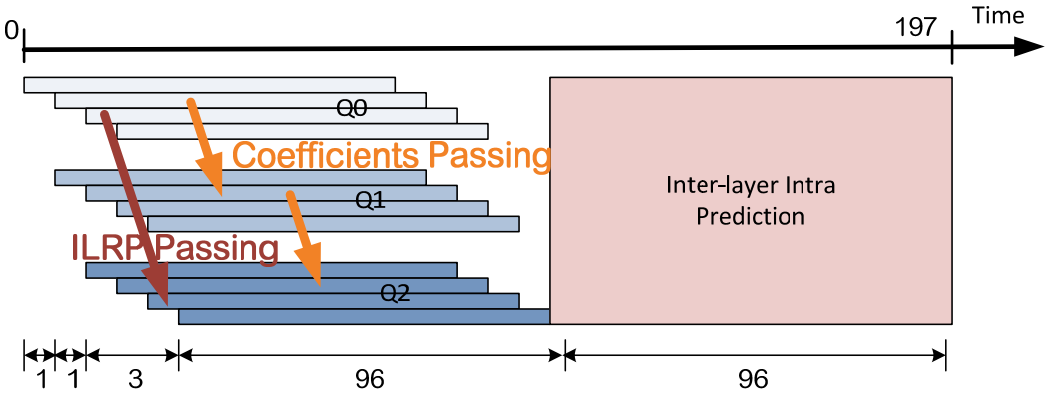


Fig. 3.11. Parallel-pipeline chain quality processing

iii. Comparison

The processing time consumption of these two strategies is very close and both of them are under the constraint of cycle budget. Thus, with the similar timing characteristics, complexity issue dominates the performance. Table. 3.4 lists the synthesis gate counts and memory requirements of major components with both strategies. From Table 3.4, although components such as inverse quantization are tripled for different quality layers, components of inter-layer prediction are not tripled in parallel-pipeline strategy. That is because with the combined scalability, all quality layers use the same contents as their prediction. Thus, only one set of interpolator is required for the parallel reconstruction of different quality layers. Compared to the high throughput of serial-pipeline chain, gate counts of interpolator are lowered with the smaller processing throughput in parallel pipeline chain. By simplifying the most complex part, total gate counts are significantly lowered by simplified inter-layer interpolator in spite of the increasing gate counts of other pipeline components. Furthermore, memory usage is saved in parallel-pipeline strategy with the coefficient passing technique. As a result, parallel-pipeline is more efficient strategy form comparison and would be adopted in this SVC decoder work.

Table. 3.4 Comparison of serial and parallel pipeline strategy

Item		**Serial-pipeline**	**Parallel-pipeline**
Inverse Quantization		9,327	4,724 × 3
Coefficients Refinement		314	155 × 3
Inverse Transform		23,124	12,137 × 3
Residual Accumulation		314	155 × 2
Residual/Texture Interpolation	Input Selector	5,901 × 20	5,901 × 16
	Basic Interpolator	Ver. 1,281 × 20	Ver. 1,281 × 16

		Hor. 1,816 × 8	Hor. 1,816 × 4
Total		191,247	173,379
Memory Requirement		768 Bytes	0

* Synthesized by UMC90 at 135MHz

2) Inter-layer Predictor

Inter-layer prediction (ILP) is an additional novel mechanism in SVC which spreads on three domains: inter-layer texture prediction, inter-layer residual prediction and inter-layer motion prediction. Prediction tools mentioned above significantly improve the coding efficiency but introduce much more coding complexity. It brings the overhead of hardware cost and power consumption. Not only for SVC encoder but also decoder suffers from the complexity of prediction generation. To light these burdens, we made efforts to simplify the architecture of ILP mechanism.

- Centralized accumulation-based CCSP engine

The first issue comes from "calculation of corresponding positions (CCSP)" where the mechanism has been mentioned in Chapter 2. From the reconstruction flow of ILP in Fig. 2.5, we can observe some functional blocks need CCSP to achieve the spatial scalability. However, the derivation of CCSP also introduces the complexity and decoding time overhead. First of all, the functional equation has listed in the Eq. (2.1) with arithmetic terms in addition, shift and multiplication. Generally, the multiplications are avoided for circuit designs which cost much more area than other arithmetic components. For the second, the ILP construction flow process CCSP in different forms repeatedly. For example, the InterBL macroblock type with ILRP totally process CCSP five times in the decoding flow which may cause considerable timing repaid. As a result, we proposed a centralized accumulation-based calculation CCSP engine to process to solve the problems mentioned above.

i. Use accumulators to replace multipliers

Despite there are many terms and complicated calculations in Eq. (2.1), [12] has introduced the opportunity to optimize the equation. Obviously, the terms of D_m, A_m, S_m, and d_m which are derived from inter-layer ratios, can be determined before coding the first macroblock in enhancement. Without spatial resolution changes, these terms will remain unchanged in following macroblock coding procedures. Thus, even though there seem many variables in CCSP derivation process, the difference between successive corresponding samples is actually a constant. This feature can be derived from

$$
\begin{aligned}
& BL_{m+1} - BL_m \\
= \; & \{\{[(m+1)\times D_m + A_m] >> (S_m - 4)\} - d_m\} - \\
& \{\{[m \times D_m + A_m] >> (S_m - 4)\} - d_m\} \\
= \; & [(m+1-m)\times D_m] >> (S_m - 4) \\
= \; & D_m >> (S_m - 4) \qquad (3.2)
\end{aligned}
$$

From Eq. (3.2), BL_{m+1} can be derived by adding a constant $D_m >> (S_m - 4)$ from BL_m. In other words, for the purposes of getting the next corresponding sample position, the only operation is adding a constant from current corresponding sample position. Thus, in hardware aspect, the needed multiplier in Eq. (3.2) can be replaced by a register plus an adder in the form of accumulator. Finally, the area cost of CCSP can be reduced by the substitution from multipliers to accumulators, which is a more efficient way in hardware architecture design.

ii. Centralized CCSP

In the traditional flow, CCSP of different inter-layer predictions types are separately processed as shown in Fig. 3.12 (a). However, calculation redundancy may

conceal in the traditional flow because the calculation of spatial parameters is similar for inter-layer predictions. The spatial parameters here are the interpolation phase, corresponding motion vectors… etc, which are related to the spatial relationship between two layers and originated from CCSP. In order to remove this redundancy and improve the decoding efficiency, a centralized CCSP concept is proposed as shown in Fig. 3.12 (b). Noticeably, the interpolation parameters (phase and BLPos) of inter-layer intra and inter-layer residual prediction are the same and also can be shared. In the proposed strategy, all spatial parameters from different prediction mechanisms are simultaneously derived by the centralized CCSP engine. The accumulation-based calculation is executed once instead of repeated operations. Some derived parameters are further restored, such as the interpolation phase or the corresponding macroblock partitioning. The buffering makes the prediction flow flexible because all of the inter-layer prediction modules can access the spatial parameters at any time without recalculation of CCSP. Some spatial parameters such as the corresponding motion vector positions and the range of reference texture can be integrated as the external memory requests. In summary, the proposed centralized accumulation-based CCSP engine not only reduces the hardware complexity but also further simplify the decoding flow of inter-layer prediction.

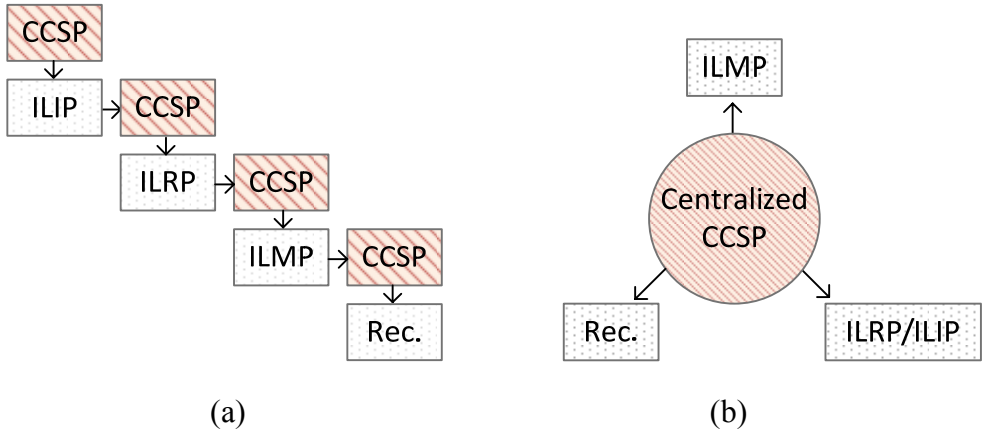


Fig. 3.12. Inter-layer prediction flows with (a) separated CCSP; (b) centralized CCSP

- Temporal Results Reuse

In Chapter 1, we have already introduced the equation of ILTP and ILRP, which are mainly composed of the upsampling interpolators to generate the prediction samples. To achieve high prediction throughputs, upsampling interpolators can be numerously established in parallel. However, complexity problem also exits in hardware architecture of upsampling interpolator. The complexity gap by the parallelism of interpolator has been previously described. It comes from the poly-phase basic interpolator and the input group selection module. Basic interpolators with various look-up table and multiplications bring out the intensity of hardware cost. The function of a basic interpolator has been listed in Eq. (2.4) and Eq. (2.5). Besides, every basic interpolators need to select its input groups from the external buffers. This action cause high complexity because of the large location range of input groups.

The target processing throughput is four pixels per cycle as previously described. Under this satiation, the requirement of reference data and basic interpolator for 4-tap filter is shown in Fig. 3.11.

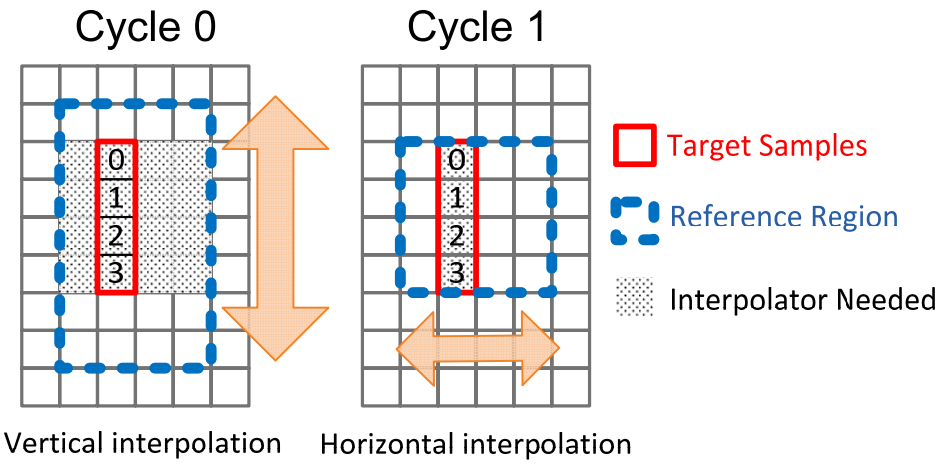


Fig. 3.13 4-tap poly-phase filtering process

Because of the limited critical path, the two dimensional interpolation is divided to a couple of successive cycles. For a group of four predicted samples, it requires 16(4×4) interpolators to generate the temporal result in vertical dimension. These temporal results are restored in temporal buffers as the inputs of the horizontal interpolation in next cycle. In the two dimensional procedure, total sixteen vertical basic interpolators and four horizontal basic interpolators are required for the dotted region in Fig. 3.13. This amount of basic interpolators may cause area overhead. To lower the gate counts, we adopt the temporal result reusing scheme, which reduces the usage amount of vertical basic interpolators.

We observed that if the target sample groups are in successive order toward specific (horizontal or vertical) direction, some part of temporal results can be reused in successive cycles. Fig. 3.14 shows the overlapped region of temporal result in horizontal interpolation order. In cycle n+1, the value of temporal results in those overlapped region stay unchanged compared with previous cycle. That means with value reusing, there are only 4 newly temporal results must be created in cycle n+1. In other words, the amount of vertical basic interpolators can be reduced to 4. The gate count reductions by temporal result reuse is 70.6% as listed on Table. 3.5.

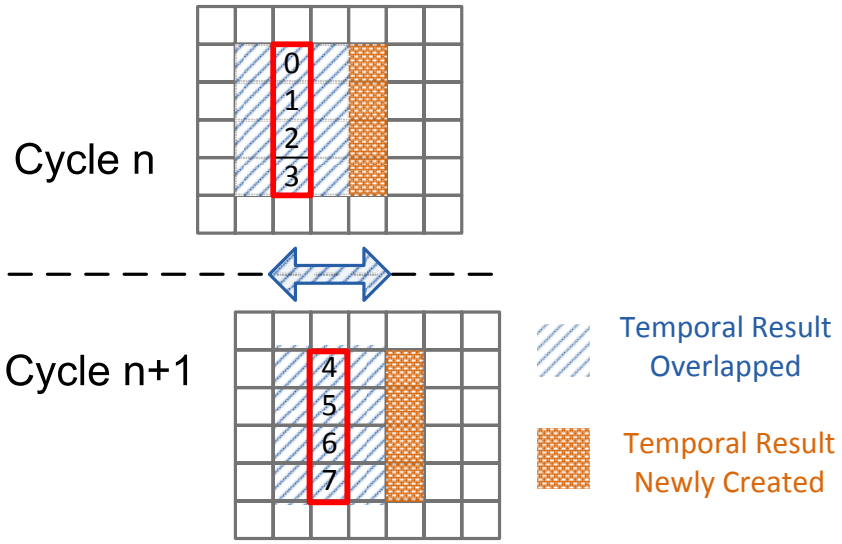


Fig. 3.14. Temporal results reusing scheme

Table. 3.5. Gate count savings from temporal result reusing

Item	Without Reuse	With Reuse	Reduction
Input Selector	94,416	23,604	75%
Vertical Interpolator	20,496	5,124	75%
Horizontal Interpolator	7,264	7,264	0%
Total	12,2176	35,992	70.6%

3.4. External Memory Access

External memory access is an important issue for video chips which dominates the performance of power consumption and coding throughput. In this section, the overview of memory system will be introduced at first. Then, the proposed memory request protocol will be presented to support the external memory access in SVC.

3.4.1. System Integration

Fig. 3.15 shows the system integration of external memory access. Designs within the 4-stage pipeline architecture can propose their request individually. In order to handle the data communication between design and DRAM, a memory controller module is established. The components inside the memory controller module are listed as follows:

Data/Address manager, the major component in memory controller, receives the requests and transforms them to corresponding addresses to DRAM. Also, it integrates the proper data in the data communication interface. Because of the large amount of external memory data in SVC, several processing units might propose their request simultaneously. Under this situation, an arbitrator is required to solve the request conflicts. Besides, FIFO buffer and busy signal are utilized to restore those

non-top priority requests and uncertain delays respectively. In this work, a packaged DDR400 DRAM SystemC model is utilized to co-simulate with the proposed design.

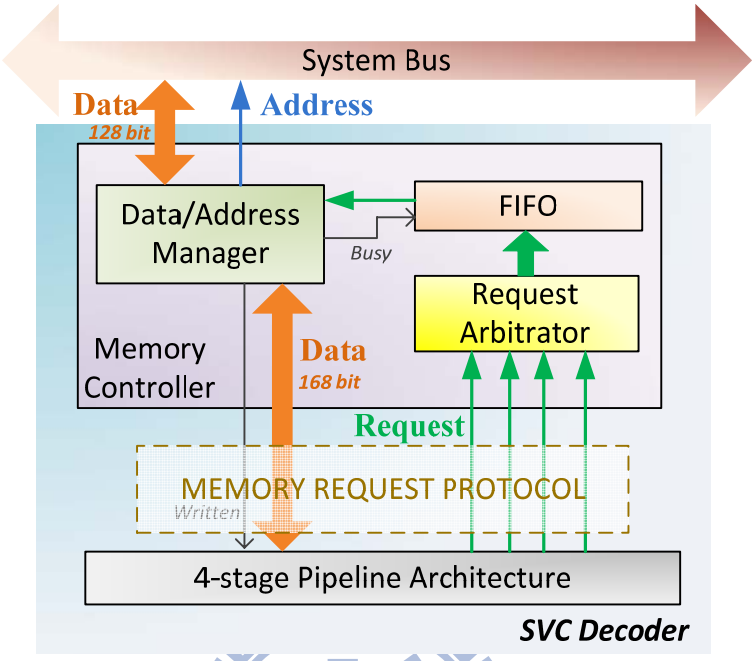


Fig. 3.15. System integration of external memory accessing

3.4.2. Memory Request Protocol

With the added data in spatial and quality scalability, SVC standard access the external memory more frequently than H.264/AVC. One group of the extra data is composed of residual data and motion vectors for inter-layer prediction, and the other is the reconstructed pixels from different quality layers for “key picture concept”. These extra data make the memory access more complicated in SVC. To deal with the significant and various external data access efficiently, fixed-length instruction-based request protocol for external memory access is utilized.

1) Request Format

The proposed protocol is composed of two parts: one is the MCM_RQST and the other is MCM_INST. The previous part is the enable signal from the 4-stage decoder design, which represents that the request of external memory access occurs. MCM_INST is the data with instruction format which can be parsed to identify the

feature of requests. Table. 3.6 shows the 45-bit length general format of MCM_INST. The first 10 bits are the global identifiers which lower the parsing complexity for memory controller with the fixed form among all request types. The following 35 bits are specified request content which depends on the request types.

Table. 3.6. General request format

Item	Bit Length	Function
Write	1	Write enable
Type	3	0 : MV data
		1 : Reconstructed data
		2 : Residual data
		3 : Bitstream (Read only)
		4 : Original picture (Encoder only)
Did	2	Dependency ID (0~2)
Frame_num	4	Frame number (0~8)
Content	35	Request dependent contents

Write is used to identify the read/write operation. *Type* represents the operation types with five different sources from external memory. *Did* and *Frame_num* indicate the target spatial layer and frame number of the accessed data, respectively. Except for the above global items, the rest bit-length is composed of *Content*, which is the request content depends on operation types. This fixed-length format simplifies the decoding complexity of memory controller.

2) Transaction

- Read operation

When a read request occurs, MCM_RQST is pulled up by the 4-stage pipeline decoder with the request instruction MCM_INST. By signaling of the enable symbol, MCM_INST will be then loaded into the FIFO of memory controller module. After

several cycles from the processing of DRAM latency, required data will be returned with a valid-bit. Fig. 3.16 (a) shows the timing diagram of read request in proposed protocol. The handshaking mechanism is globally controlled within the memory controller level. Thus, it is delivers efficient memory transaction without stalling the processing components for waiting the request acceptation.

- Write operation

In write operation, the write data from 4-stage pipeline design must hold until the pulling-up of written signal from external memory. Once the Written symbol is signaled, other MCM_RQSTs can be proposed immediately. Fig. 3.16 (b) shows the timing diagram of write request in proposed protocol.

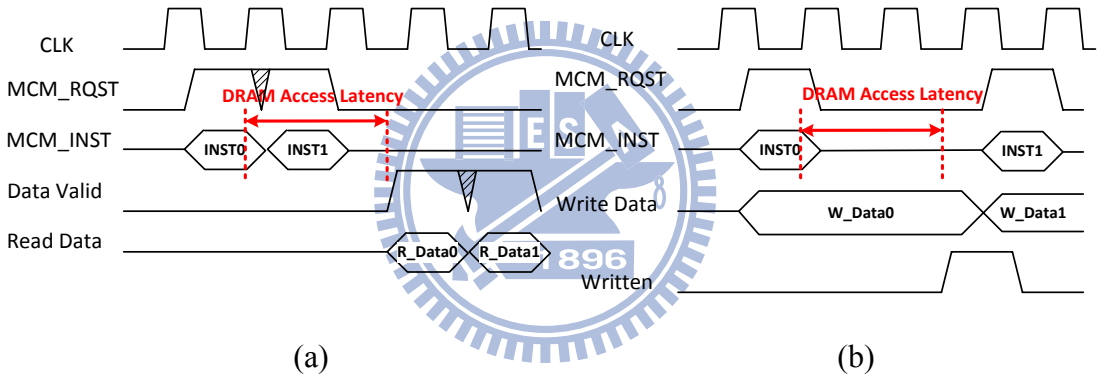


Fig. 3.16. Timing diagram: (a) read request; (b) write request

3.5. Summary

In this chapter, the decoding flow of SNR scalability is analyzed based on three strategies: frame-based, MB-based, and the proposed One-pass quality decoding. To summarize the simulation result, Fig. 3.17 shows comparison of the memory bandwidth and processing macroblocks among those decoding flows. The One-Pass quality decoding performs well from its parallel decoding strategy in addition to MB-based quality decoding. For architecture design, 4-stage synchronous pipeline architecture is utilized for this work. The residual reconstruction path in 2nd stage is

the most complicated part in SVC. Thus, we propose the parallel-pipeline reconstruction flow and temporal results reusing scheme to lower the gate count. The gate count saving is summarized as Fig. 3.18.

Besides, the BL-level padding flow is proposed to accelerate the texture padding process which has 26% improvement in IntraBL-coded macroblocks. To improve the area cost and the coding flexibility of inter-layer prediction, a centralized accumulation-based CCSP scheme is utilized. Finally, the external memory access in this work is introduced with the specified protocol.

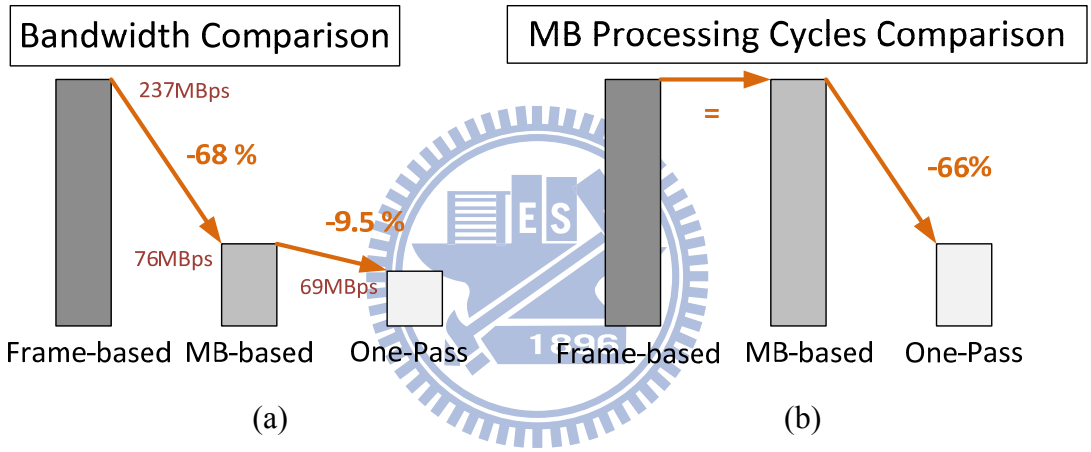


Fig. 3.17. Comparison of quality decoding flows in (a) memory bandwidth; (b) MB processing cycles

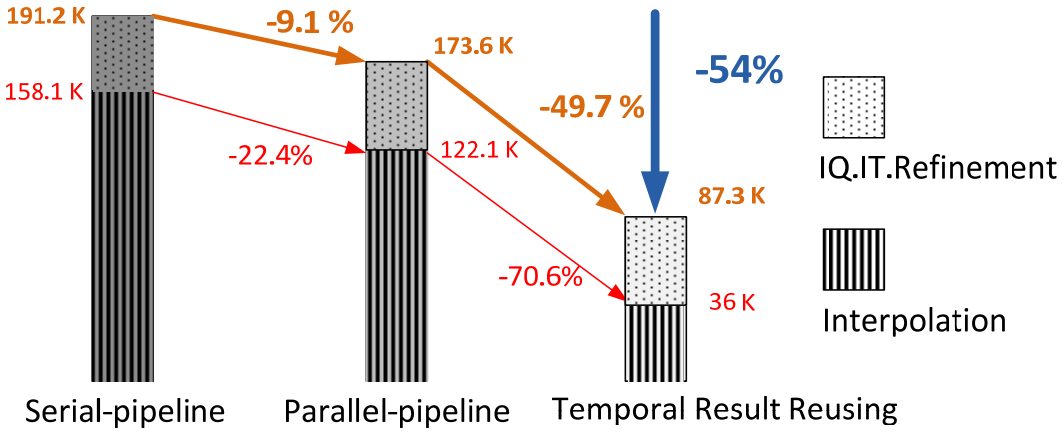


Fig. 3.18. Gate count savings of residual reconstruction path


E S A
1896

Chapter 4. Module Design

Detail implementation methodology and optimization strategy of SVC decoder modules are introduced in this Chapter. At first, the physical implementation of inter-layer content upsampler will be presented. Afterward, the intra-reconstruction component in this work will be mentioned with the proposed low-complexity plane generator. Components of entropy decoding, motion compensation, deblocking filter will be briefly introduced then. At last, there will be the research and refined architecture of the memory controller module arbitrator.

4.1. Inter-layer Prediction

Three components of inter-layer prediction are introduced in this Section: poly-phase interpolator, texture padding module, and MV upsampler.

4.1.1. Poly-phase Interpolator Design

The major complexity of a basic interpolator comes from the various combination of filtering coefficients. This variety introduces a large amount of arithmetic components and complex control circuits. To lighten hardware burden, we propose a hybrid interpolator architecture to implement both of 4-tap and bi-linear interpolations. Through our design, the adders can be shared by these two interpolation processes and consequently lowers the hardware cost. The details of our designs are describes as follows.

1) Simplification of coefficients table

To reduce the control complexity, the most important thing is to simplify the variety of coefficient combinations. From the coefficient table, the symmetries in the

filter coefficient table as shown in Table. 4.1 can be observed directly. For example, supposed that 4-tap filter is used and the interpolation phase is 13, circuits whose phase equal to three can be shared by only exchanging C_{-1} for C_2, and C_0 for C_1. Thus, half of contents in the table can be reduced by exchanging inputs with symmetric coefficient columns in the table.

Table.4.1. Symmetry of coefficients table

(a) Bi-linear filter

coefficients		phase		coefficients	
C1	C0			C0	C1
—	—	0		32	0
2	30	15	1	30	2
4	28	14	2	28	4
6	26	13	3	26	6
8	24	12	4	24	8
10	22	11	5	22	10
12	20	10	6	20	12
14	18	9	7	18	14
—	—	8		16	16

(b) 4-tap filter

coefficients				phase		coefficients			
C-1	C0	C1	C2			C-1	C0	C1	C2
—	—	—	—	0		0	32	0	0
-1	2	32	-1	15	1	-1	32	2	-1
-1	4	31	-2	14	2	-2	31	4	-1
-1	6	30	-3	13	3	-3	30	6	-1
-1	8	28	-3	12	4	-3	28	8	-1
-1	11	26	-4	11	5	-4	26	11	-1
-2	14	24	-4	10	6	-4	24	14	-2
-3	16	22	-3	9	7	-3	22	16	-3
—	—	—	—	8		-3	19	19	-3

2) Simple adder-tree architecture

Fig. 4.1 shows our proposed hybrid interpolation module. In Stage I, reference samples are rearranged according to interpolation phase and filtering mode. This step takes the advantage of coefficient table symmetry as previously described. In Stage II, scaling engine produces scaled elements and classifies them to three sets. The classifying strategy is to group minimum scaled elements in every set, thus the input selection for adders can be efficiently simplified. The proposed scaled element classification is listed in Table. 4.2 in detail. Finally, the data path in stage III is basically composed of simple two-level adder-tree architecture. This architecture makes the circuits simpler without complex control signals.

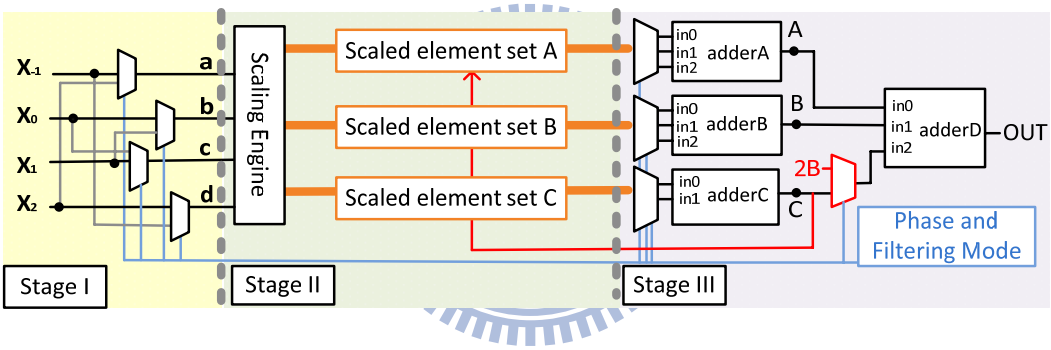


Fig. 4.1. Architecture of the proposed interpolator

Table. 4.2. Input selection and classification for adders

Mode	Phase	adder A			adder B			adder C		adder D		
		in0	in1	in2	in0	in1	in2	in0	in1	in0	in1	in2
0	0	16b	16b	0	0	0	0	0	0	A	B	C
0	1	16b	16b	0	2c	0	0	-a	-d	A	B	C
0	2	16b	16b	-b	0	4c	0	-2a	-d	A	B	C
0	3	16b	16b	-2b	2c	4c	-d	-a	-2a	A	B	C
0	4	16b	16b	-4b	0	8c	-d	-a	-2a	A	B	C
0	5	16b	16b	2b	2c	8c	c	-4a	-d	A	B	C
0	6	16b	16b	0	2c	8c	4c	-4a	-2d	A	B	C
0	7	16b	16b	0	2d	16c	-d	-a	-2a	A	B	C

0	8	C	2C	16B	0	b	c	-a	-d	A	B	2B
1	0	16b	16b	0	0	0	0	0	0	A	B	C
1	1	16b	16b	-2b	0	0	2c	0	0	A	B	C
1	2	16b	8b	4b	4c	0	0	0	0	A	B	C
1	3	16b	8b	2b	4c	0	2c	0	0	A	B	C
1	4	16b	8b	0	0	8c	0	0	0	A	B	C
1	5	16b	2b	4b	0	8c	2c	0	0	A	B	C
1	6	16b	0	4b	4c	8c	0	0	0	A	B	C
1	7	16b	2b	0	4c	8c	2c	0	0	A	B	C
1	8	8b	0	0	0	8c	0	0	0	A	B	C
		Set A			*Set B*			*Set C*				

The synthesis result and the comparison of proposed method and direct implementations is listed in Table. 4.3. The gate counts saving from switched method is 14.7% by combining bi-linear and 4-tap filter together. Additional 28.8% gate counts can be further saved by the proposed simple poly-phase interpolator design.

Table. 4.3. Synthesis results of horizontal basic interpolator

Method	**Gate counts**
4-tap (Direct)	2,119
Bi-linear (Direct)	873
Switched (Direct)	2,551
Proposed	1,816

* Synthesized by UMC90 at 135MHz

4.1.2. Texture Padding

In Chapter 3, we have mentioned the proposed BL-level flow for texture padding for inter-layer intra prediction. In this section, the detail module design of texture padding will be introduced. Fig. 4.2 shows the detail of texture padding. The dotted

region is the target region, and the texts in the right side indicate the padding function. ~A means region A is not reconstructed (Inter-coded region in base layer), so as ~B, ~C. The different types of neighboring macroblocks affect the operations of the target 8×8 block which is divided to three regions with different cases. The operation functions "ddr", "ver", "hor" mean the down-diagonal right, vertical, horizontal extrapolations respectively. Therefore, the target 8×8 block can be padded if the neighboring macroblock types and neighboring pixels (a, b, and c) are determined.

With the data dependency from neighboring pixels and macroblock types, the target region of macroblock-based texture padding is special where the padded region is not a real macroblock. It is a macroblock-size region which composed of 4 8×8 blocks at the up-left side of current macroblock. Fig. 4.2 also shows the target 4 8×8 blocks within the dotted line which can be simultaneously determined.

The determination function is presented as follows: If the current macroblock is reconstruct-able, the deblocked samples in block *X* will be regarded as neighboring pixels for other three blocks. Otherwise, the block *X* will be padded where the neighboring information is derived and ready for use. Also, other three blocks (A, B and C in Fig. 4.2) can be determined in the same way with block *X*. Fig. 4.3 (a) illustrates the padding status with the macroblock-based processing. The deblocked cross stripe at the macroblock boundaries are referenced for padding the four neighboring 8×8 blocks. This texture padding order works successively without data hazards from neighboring information.

Besides, extra buffers are needed to restore the neighboring samples as shown in Fig. 4.3 (b). Parts of deblocked data from current macroblock processing are picked into buffers to be referenced in texture padding. In summary, padding buffers can be classified to three types: current MB buffer, column buffer, and row buffer. Current MB buffer restores the deblocked samples for the texture padding with current

macroblock processing. Column buffer and row buffer is used to buffer the reference samples for the next and the next-row macroblock processing, respectively. Fig. 4.3 (a) also shows the usage of padding buffers with the cross stripe in different colors. The capacity requirement of padding buffers is listed in Table. 4.3.

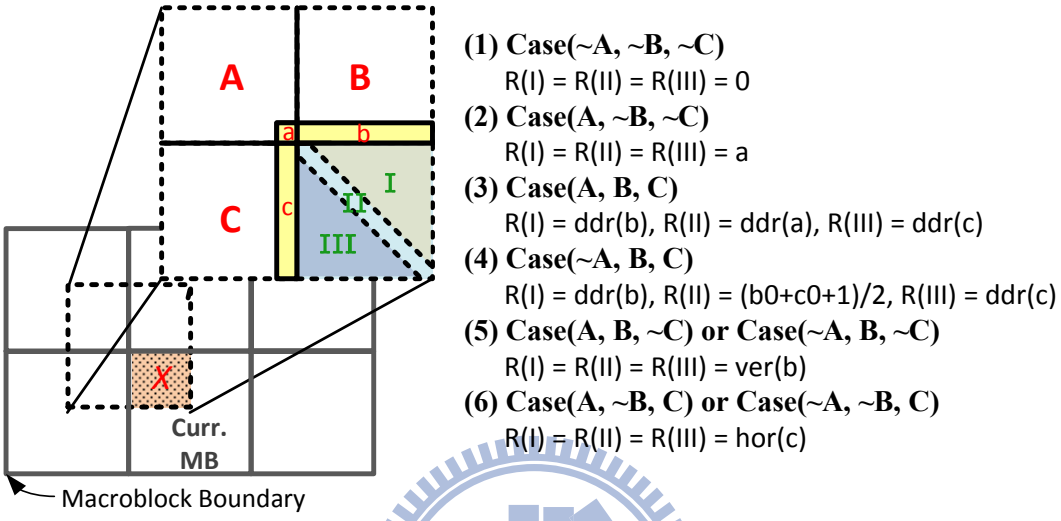
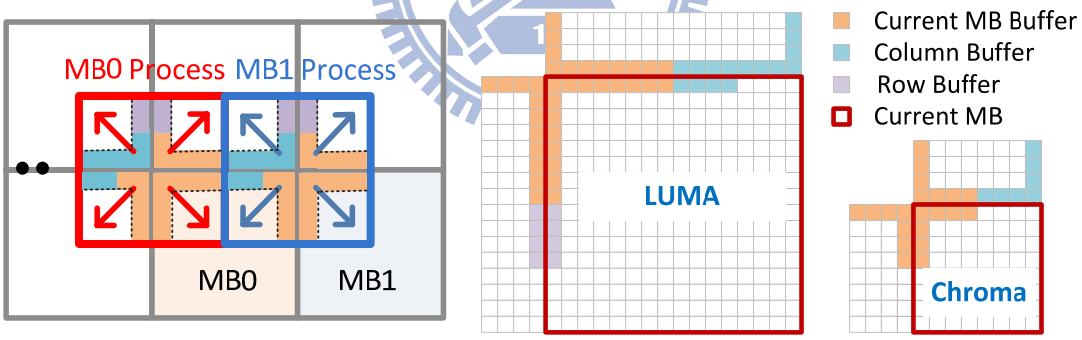


Fig. 4.2. Detail of texture padding



(a) (b)

Fig. 4.3. Texture Padding: (a) Process status; (b) Neighboring pixel buffering

Table. 4.3. Capacity requirement of padding buffers

Item	Current MB	Column	Row
Luma	37	15	8 × 120
Cb + Cr	21 + 21	7 + 7	0
Total	79	29	960

*Unit: Byte

4.1.3. MV Upsampler

The flow of motion vector upsampling is mentioned in Chapter 2 with its refinement actions. From Eq. (2.5), because of the multiplier term, motion vector from base layer is not a constant. Thus, multiplications cannot be avoided by adder-tree architecture or accumulator. In the proposed MV upsampler, two multipliers are established for the *MVx* and *MVy*. Thus, total 16 cycles are needed to scale the motion vectors in a reference list. This throughput is acceptable with minimum hardware cost in the proposed architecture. In addition, MV Merge component is required to integrate the motion vector refinements. One MV Merge module is utilized in this work to average the similar motion vectors and derive the new sub-partition type. The averaging and partitioning strategy is listed in Fig. 4.4. Motion vectors in a 4×4 block are required to further profile their similarity characteristics. Then the macroblock partitioning can be derived under the combinations among those similarity sets. Finally, three partition types lead to different adding combinations of motion vectors with reused adder units.

Furthermore, the MV upsampling is accelerated by the identification of direct_8x8_inference_flag. This term is an optional flag signaled from encoder, which represents the other three motion vectors within a 8×8 block are set equal to the corner one in B_Slice. This configuration make the macroblock partition size limited to beyond-8×8. Thus, the identification of direct_8x8_inference_flag can remove additional steps from general cases. Under this flag, only 8 motion vectors are required to be upsampled and the MV merge step is skipped because the motion vectors are already integrated.

Fig. 4.5. shows the timing schedule of proposed MV upsampler determined by the slice type and inference_8×8_flag. The processing time is saved by the adaptive scheduling with flag identification.

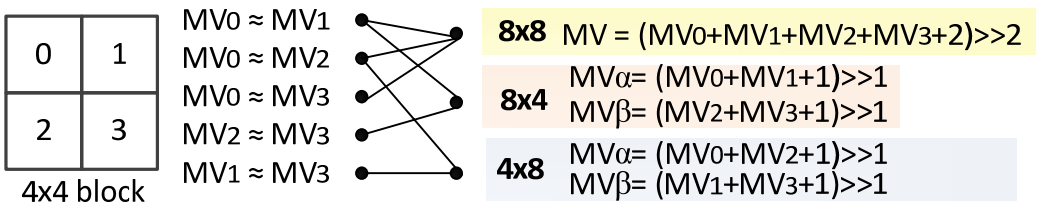


Fig. 4.4. MV and Sub-macroblock type derivation

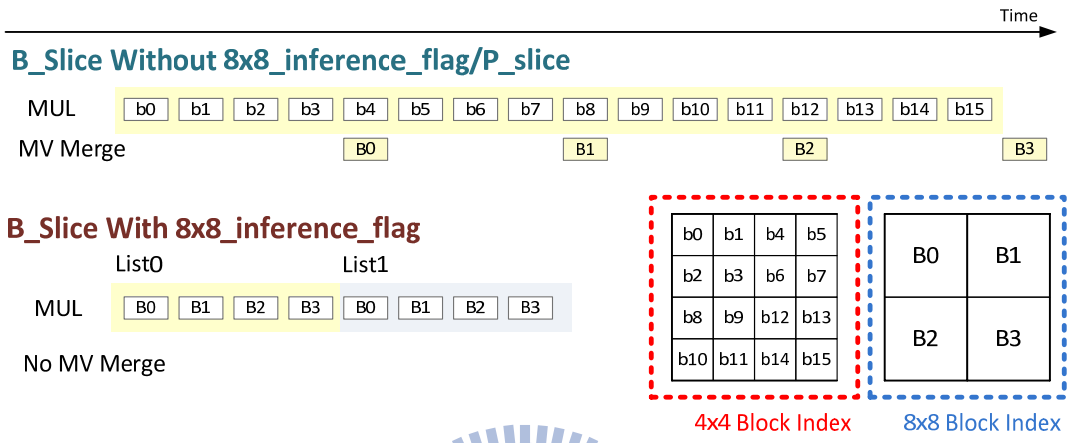


Fig. 4.5. Timing schedule of MV upsampler

4.2. Intra Prediction

Intra-frame prediction is a commonly seen prediction mechanism which widely used in image/video coding standards. In the Intra-coded macroblocks, prediction samples are generated from padding the spatial neighboring pixels. Because pixels in natural scenes are likely similar to other pixels around, Intra prediction achieve good compression performance.

4.2.1. Architecture

In this work, Intra prediction generator is arranged in the 3rd pipeline stage. Residual data within at most two qualities are read from previous stage and being reconstructed to sample pixels in this stage. The lower quality residual is used to reconstruct the lowest quality samples, which are regarded as the prediction data to the quality enhancement layer. The architecture of proposed Intra predictor is shown in Fig. 4.6.

To restore the needed data from neighboring macroblocks, two neighboring data SRAMs are established for luma and chroma respectively. Because the maximum frame size of this work is 1080p which contains 120 macroblocks in a frame row, the entries 0 to 119 are used to buffer the neighboring data from upper line. In addition, the entry 120 is for the left neighboring line data. The illustration of the neighboring data SRAM is shown in Fig. 4.7. 16 luma neighboring sample pixels and 4 neighboring prediction modes are grouped to 144 bits as a memory column. Besides, 8 cb and 8 cr samples are also grouped as chroma neighboring data.

With the required neighboring data, prediction of current macroblock can be generated in sequential order. The prediction data would be added to residuals to form the reconstructed pixels. During the reconstruction process, the minimum quality layer reconstructed data will be updated to neighboring buffer for the un-processed blocks. Besides, with the minimum reconstructed samples, maximum quality layer residuals can be formed by simple accumulation. The residuals with different quality layers are accessed in parallel in this work, thus reconstruction of two different quality layers can be processed within the same cycle.

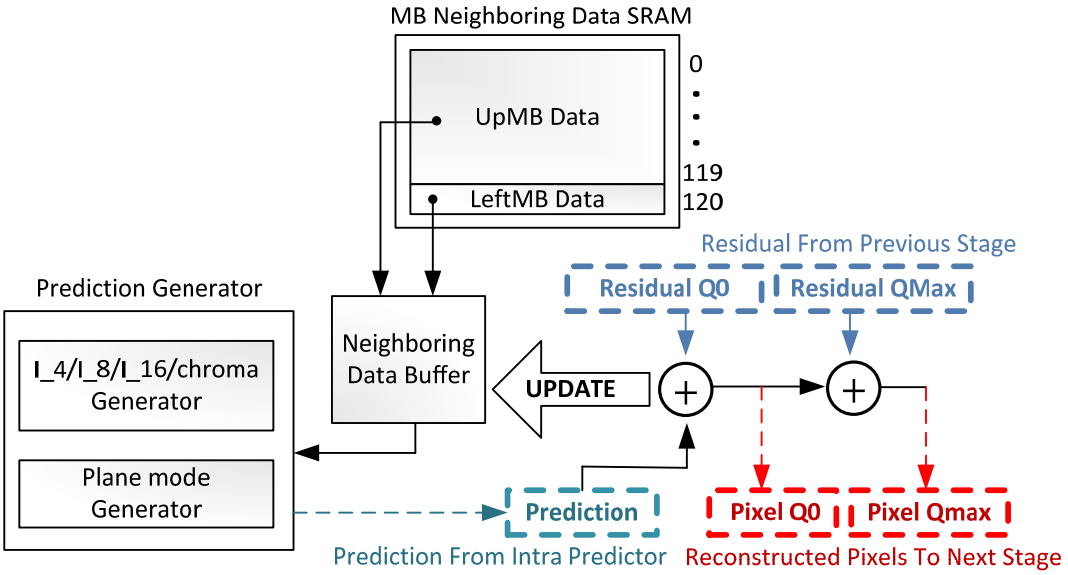


Fig. 4.6. Data path of Intra Prediction

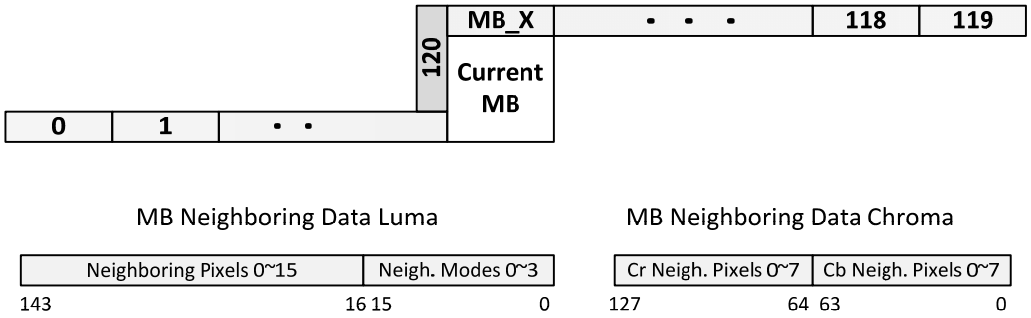


Fig. 4.7. Neighboring data SRAM

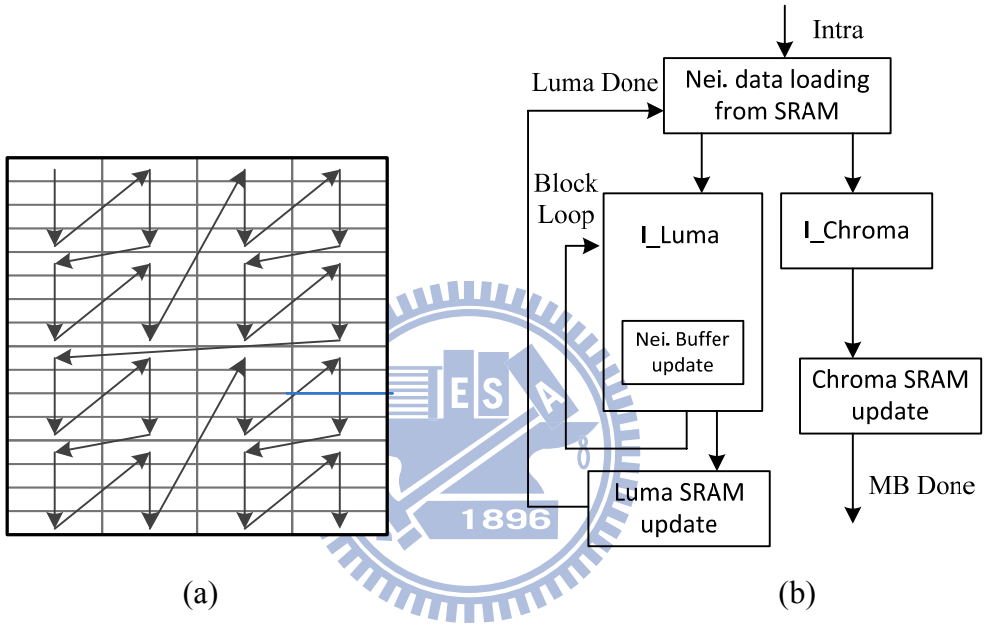


Fig. 4.8 Intra Prediction: (a) Processing order; (b) Processing flow

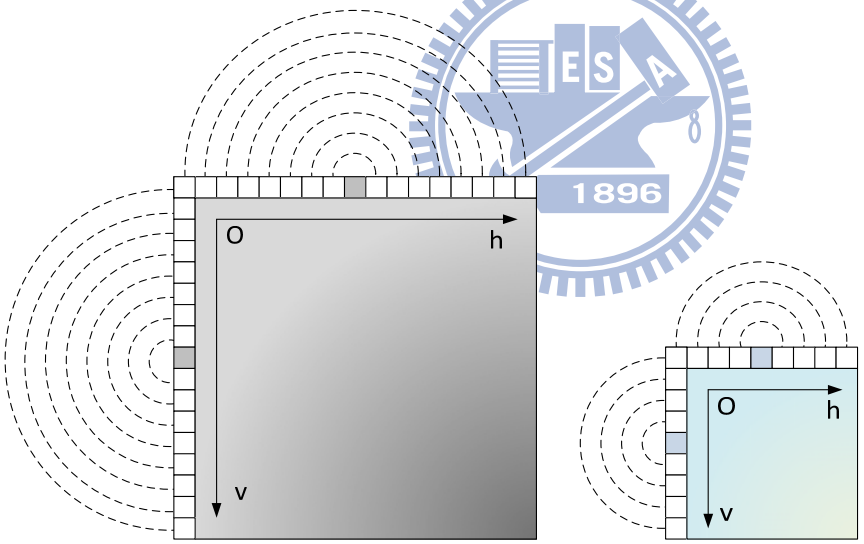
4.2.2. Processing Flow

Trading off by hardware cost and the cycle budget, this work takes 4 pixels parallelism as the processing throughput. Fig. 4.8. (a) shows the reconstruction order for the luma samples and the arrows indicate the next processing row. No matter what prediction modes, 4 samples within each 4×1 row are reconstructed in the same cycle. Consequently, there are total 64 luma rows and 32 chroma processing rows for an Intra-coded macroblock. Fig. 4.8 (b) shows the processing flow of the proposed Intra predictor. The reconstruction of luma samples and neighboring buffer update are firstly processed according to the block-based prediction modes. The state of luma

sample procedure is repeated according to the prediction mode, I_4×4, I_8×8, and I_16×16. After the luma loop, chroma reconstruction can be process then with the refreshed neighboring data from chroma neighboring data SRAM. The reconstruction of macroblock will be done in 121~145 cycles depends on the intra prediction modes.

4.2.3. Plane Mode Generator

Plane mode is the most complex mode among the Intra prediction modes. Some encoder design removes this mode because of the major problem, significant area cost. Fig. 4.9 shows the function of plane mode, which needs distinctive calculation compared with other prediction modes in Intra prediction. The Intra predictor supports all prediction modes in H.264/AVC in this work. To reduce the complexity overhead, we proposed an area efficient scheme to implement the plane mode generator.



$H = \sum h \times [\, p(7+i,-1) - p(7-i,-1) \,]$, for $i = 1 \sim 8$
$V = \sum v \times [\, p(-1,7+i) - p(-1,7-i) \,]$, for $i = 1 \sim 8$
$\alpha = 16 \times [\, p(-1,15) - p(15,-1) \,]$
$\beta = (5 \times H + 32) >> 6$
$\gamma = (5 \times V + 32) >> 6$
$Pred(h,v) = [\, \alpha + \beta \times (h-7) + \gamma \times (v-7) + 16 \,] >> 5$

(a)

$H = \sum h \times [\, p(4+i,-1) - p(4-i,-1) \,]$, for $i = 1 \sim 4$
$V = \sum v \times [\, p(-1,4+i) - p(-1,4-i) \,]$, for $i = 1 \sim 4$
$\alpha = 16 \times [\, p(-1,7) - p(7,-1) \,]$
$\beta = (17 \times H + 16) >> 5$
$\gamma = (17 \times V + 16) >> 5$
$Pred(h,v) = [\, \alpha + \beta \times (h-3) + \gamma \times (v-3) + 16 \,] >> 5$

(b)

Fig. 4.9. Intra plane mode prediction (a) intra 16×16; (b) chroma

1) Weight Derivation

At first, the horizontal and vertical plane "Pre_Weight" H and V can be generated by the accumulation buffer and scaling-addition combinations. Fig. 4.10. shows the derivation circuit. In0 and In1 are the input pair which selected for subtraction. The timing schedule and scaled term for weight derivation is listed in Table. 4.4. The cycle counter can be the index of the input pair and scaling coefficients. Acc_Buf is used to restore the temp relay pre-weight which will be accumulated later. During the 9th cycle, the plane weight α or β will be further derived to by shifting and adding by the final accumulation result. With the accumulation based weight derivation, 9 cycles and 5 cycles are needed to process one directional weight derivation for luma and chroma respectively.

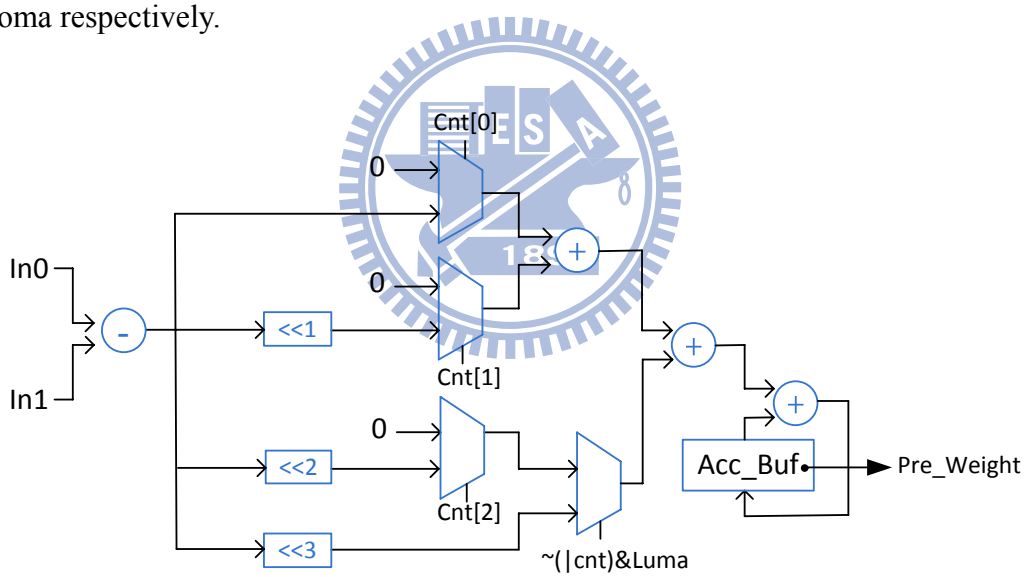


Fig. 4.10. Wight derivation circuit

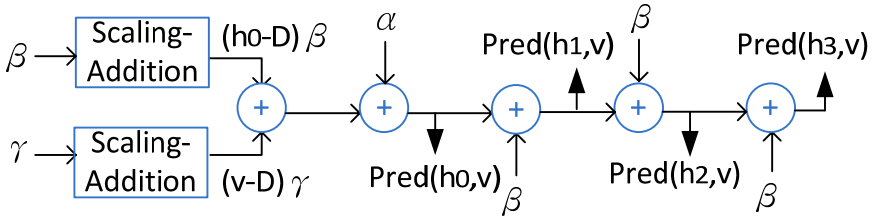


Fig. 4.11. Proposed plane sample generator

Table. 4.4. Timing scheduling (a) luma; (b) chroma

(a)

Cycles	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Cnt	0	1	2	3	4	5	6	7	(Acc_buf×
In0	p15	p8	p9	p10	p11	p12	p13	p14	5+16)
In1	pUL	p6	p5	p4	p3	p2	p1	p0	>>6
Scaled	8	1	2	3	4	5	6	7	

(b)

Cycles	**1**	**2**	**3**	**4**	**5**
Cnt	0	1	2	3	(Acc_buf
In0	p7	p4	p5	p6	×5+16)
In1	pUL	p2	p1	p0	>>6
Scaled	4	1	2	3	

2) Coefficients combination

After the previous step, calculated plane weights will be involved with position-related linear combination then. The coefficient weight for every position is listed in Table.4.4.

$$Pred(h,v) = [\ \alpha + \beta \times (h - D) + \gamma \times (v - D) + 16\ ] >> 5$$

$$D = 7 \text{ for luma, } D = 3 \text{ for chroma} \qquad (4.1)$$

Because 4 prediction samples which output per every successive cycle is in the same row in this work, the $\gamma \times (v - D)$ term can be reuse for these 4 samples (the same v). Moreover, These 4 samples are in successive order in horizontal direction. Once the term $\beta \times (h - D)$ of the left most sample is derived, other 3 samples can be derived by adding the scalings of constant β. With this characteristic, the coefficient index can be reduced to quarter in horizontal direction. Fig. 4.11 illustrates the proposed plane sample generator.

4.2.4. Synthesis Result

The synthesis result of Intra predictor is listed as Table. 4.5. From Table. 4.5, this work is competitive with [20] in the gate count and internal memory usage aspect.

Table. 4.5. Synthesis results of Intra predictor

Item	**[20]**	**Proposed**
Process	0.18um	0.09um
Frequency	120MHz	135MHz
Specification	H.264/AVC Decoder	SVC Decoder
Pixel Parallelism	4 pixels	4 pixels
Internal Memory	4.93KBytes	4.12KBytes
Gate counts	28,707	28,326
- *Neighboring Pixel Buffer*	N/A	7,490
- *I16 / Plane Mode*	N/A	3,556 / 2,889
- *I4*	N/A	2,167
- *I8*	N/A	4,345
- *IChroma*(Not including plane mode)	N/A	1,680
- *Control & Arithmetic*	N/A	8,108
- *Reconstruction*	N/A	980

Chapter 5. Implementation Result

In this chapter, implementation results of this work are summarized which includes the synthesized gate counts and memory requirement. The proposed Scalable High profile video decoder architecture is implemented in Verilog HDL with UMC 90nm 1P9M CMOS technology.

5.1. Design Flow

Fig. 5.1 shows the design flow in this work. After certainly defining the target system specification, the corresponding C-model is generated then. It is convenient to develop the coding algorithms by software-based approach. Once the algorithm is confirmed, the hardware architecture can be implemented in verilog HDL. The RTL verification of functional and timing behaviors starts by simulation with golden C-model. SystemC model is introduced to co-simulate with verilog models for higher level design. Comparisons from synthesis results direct the refinement loop to an appropriate architecture design.

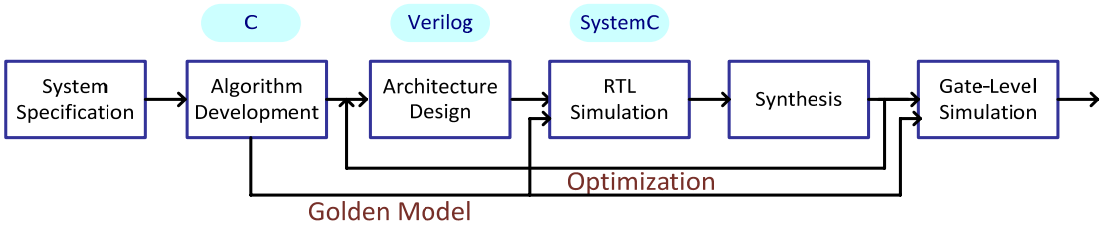


Fig. 5.1. Design Flow in this work

5.2. Gate Count

For 135 MHz synthesis frequency (clock period is set to 7.4 ns), the total gate count of this work is about 565 k. The gate count of each component is listed in

Table. 5.1.

Table. 5.1. List of gate count for proposed SVC decoder

Module	Gate counts
Entropy Decoder + Syntax Parser	213,638
Motion Compensation	107,080
Deblocking Filter	24,573
Inter-layer Prediction	87,778
- *Centralized CCSP*	14,312
- *Texture/Residual Upsample*	43,032
- *MV Upsample*	6,703
- *External Data Buffer*	23,131
Residual Reconstruction	56,184
- *Inverse DCT & Hadamard Transform*	36,117
- *Inverse Quantization*	15,972
- *Reconstruction + Control*	4,095
Intra Prediction	28,326
- *Prediction Generator + Control*	20,836
- *Neighboring Pixel Buffer*	7,490
Texture Padding	9,870
- *Padding Unit*	4,398
- *Neighboring Pixel Buffer*	5,472
Memory Controller	12,678
System Control	2,001
Total	**541,527**

* Synthesized by UMC90 at 135MHz

5.3. Memory Organization

With the numerous quality and spatial scalabilities provided by SVC, memory demand is also significantly increasing among other standards. To balance the on-chip

memory demand and processing throughput, partitioning the data stream into internal (high speed) and external (large capacity) storage elements is needed. The requirement and organization of external and internal memory usage are presented in this sub-section.

1) External memory

The detail of external memory data is listed in Table. 5.2. With the MB-based spatial layer decoding flow, inter-layer prediction data is placed into the external memory. Because the inter-layer prediction data is not referenced across frame boundary (see Fig. 2.5), only one frame size is required. Noticeably, motion vector is defined as a 42-bit data with two reference lists in this work. Thus, 84 bytes are required to restore motion vectors in one macroblock. In the target (top enhancement) spatial layer, deblocked pixel samples are wrote to external memory to be referenced or displayed. Two quality layers in key pictures are restored according to key picture concept. Besides, in the lowest spatial layer, column motion vectors in list1 frames are restored for B_Direct mode. The total external memory requirement in this work is 44178.75 Kbytes (about 43Mbytes).

Table. 5.2. List of DRAM requirement for proposed SVC decoder

Module	**DRAM Requirement**
Inter-layer Prediction	7171.875
- *Motion Vector*	669.375
- *Residual*	3442.5
- *Texture*	3060
MC Reference/Display	33660
Column Motion Vector	3346.875
Total	**44178.75**

Unit: Kbytes

2) Internal memory

Table 5.3 lists the internal memory requirement among components and pipeline stages. For inter-layer intra and residual prediction, the transform size and reconstruct-able symbol of corresponding macroblocks in base layer are restored in SRAM. For deblocking filter, motion compensation, and Intra prediction, SRAMs are used to buffering the neighboring data from left or the upper line macroblocks. However, because two quality reconstruction pixels need to be deblocked in the key picture, SRAM requirement is also doubled. Similar situations exist in the pipeline stage buffer, which pass coefficients, residuals, and reconstructed pixels with two qualities to next stage. In summary, the total internal memory requirement in this is 39.66 Kbytes.

Table. 5.3. List of SRAM requirement for proposed SVC decoder

Module	**SRAM Requirement**	
	Single Port	**Dual/Two Port**
Entropy Decoder + Syntax Parser	2.039	3.411
Motion Compensation	2.461	
Deblocking Filter	7.461	1.218
- *Neighboring Pixels (Q0+Qmax)*	7.374	
- *Others*	0.087	1.218
Inter-layer Prediction	4	
Intra Prediction	4.116	
- *Luma Neighboring Data*	2.179	
- *Chroma Neighboring Pixels*	1.937	
Texture Padding	0.937	
Pipeline Ping-pong Buffer	12.893	1.125
- *Stage1*	4.269	
- *Coefficients*	4.113	
- *MVDs*	0.156	

- *Stage2*	6.968	1.125
- *Residuals (Q0+Qmax)*	0.562	1.125
- *MC Reference Pixels*	6.250	
- *MVs*	0.156	
- *Stage3*	1.656	
- *Reconstructed Pixels*	1.50	
- *MVs*	0.156	
Total	**33.907**	**5.754**
Total (Single port + Two/Dual port)		**39.661**

* Unit: Kbytes

5.4. Comparison

The comparison of this and other state-of-art video decoders are listed in Table. 5.4. Because there is only one published design of H.264/AVC scalable extension so far [16], H.264/AVC HD decoders [4] [5] are listed in the comparison table. Generally, the gate count cost for SVC decoder is larger than H.264/AVC due to the additional scalabilities. It majorly comes from the inter-layer prediction which introduces high arithmetic complexity and numerous external data buffers. Also, SVC applications have more external memory requirements as well. With the equal standard, Scalable High profile, although [16] has better capability in decoding single scalability, this work can provide superior Max Throughput for multiple scalabilities. Max Throughput represents the processing competence for combined spatial and quality scalability as defined in Table. 5.4. To normalize the performance, Gate Efficiency and SRAM Efficiency are utilized which mean the Max throughput per kilo gates and Max Throughput per kilo bytes, respectively. This work has better performance in Gate Efficiency. However, the relative high internal memory requirement causes the drop of SRAM Efficiency.

Table. 5.4. Comparison with other state-of-art video decoders

	[4]	[5]	[16]	**Proposed**
Technology	0.18 um	0.18 um	0.09 um	0.09 um
Max Clock Rate	100MHz	120MHz	210MHz	135MHz
Profile	MPEG-2 SP@ML H.264 BL@L4	H.264 Baseline/Main	SVC High MVC High	SVC High @ L5
Max Spec. **(H.264)**	1920×1088 @ 30fps	1920×1088 @30fps	4096×2160 @24fps	1920×1088 @60fps
Max Spec. **(SVC)**	N/A	N/A	SL: 1920×1088 + 1280×720 @ 30fps QL: 1920×1088 w/ 4 QLs @30fps	1920×1088 + 720×480 + 352×288 w/ 3 QLs @60fps
Gate Count	303.78 K	160 K	414.28 K	541.52 K
Internal Memory	22.75 Kbtyes	4.5 Kbtyes	8.99 Kbtyes	39.66 Kbtyes
External Memory	8MB DRAM	N/A	N/A	43MB DRAM
Max Throughput	244800 MB/s	244800 MB/s	979200 MB/s	1783080 MB/s
Gate Efficiency	805.84 MB/Kgates-s	1530 MB/Kgates-s	2363.62 MB/Kgates-s	3292.68 MB/Kgates-s
SRAM **Efficiency**	10760 MB/Kbytes-s	54400 MB/Kbytes-s	108921 MB/Kbytes-s	44959 MB/Kbytes-s

** Max Throughput = Frame rate × Processing MBs in (Spatial + Quality) layers*

Chapter 6. Conclusion and Future Work

6.1. Conclusion

In this thesis, a complete design methodology for SVC decoder is presented from decoding flow analysis to module implementation. From simulation results, the proposed One-pass quality decoding further reduces 71% memory bandwidth and at most 66% macroblock processing time. For the padding issue before texture upsampling, the proposed BL-level scheme can save 26% time consumption for the IntraBL macroblocks. This work also analyzes and optimizes the architecture design of SVC decoder. For the residual reconstruction path, parallel-pipeline architecture saves 9.1% gate count cost and 100% memory cost compared with the other one. Temporal result reusing scheme reduces 75% required basic vertical interpolations, which lowers 70.6% gate counts in interpolator design. The parallel-pipeline architecture and temporal result reusing scheme can be combined and lead to 54% gate count savings in residual reconstruction path. This thesis also introduced the architecture design of external memory access and inter-layer prediction. For inter-layer module design, the proposed basic interpolator reduces 28.8% hardware complexity. The gate counts of MV upsampler and texture padder are 6.7k and 4.4k respectively.

Finally, the proposed a Scalable high profile decoder can successfully support 3 spatial layers and 3 quality layers simultaneously. It meets the specification which combines CIF, SD480p, and HD1080p with 3 quality layers at 60 frames per second. The total gate counts and internal memory usage in this work are 541.52k and 39.66 Kbytes, respectively.

6.2. Future Work

The power consumption issue is not mentioned in this work. Although the simplified complexity basically helps the power reduction, specific power saving strategies may further help the power performance in this work. In addition, because this work is based on the result of gate-level simulation, physical layout and chip testing are not included. For further verification or power testing, the back-end design flow is required in the future.

Reference

[1] Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264/ISO/ IEC 14 496-10 AVC," JVT-G050, 2003.

[2] Joint Draft 11 of SVC Amendment, Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, Oct. 2007.

[3] D.-Zhou et al., "A 1080p@60fps multi-standard video decoder chip designed for power and cost efficiency in a system perspective," in *Symposium on VLSI Circuit*, pp. 262-263, June. 2009.

[4] T.-M. Liu et al., "A 125 μW, Fully Scalable MPEG-2 and H.264/AVC Video Decoder for Mobile Applications," in *IEEE Solid-State Circuits Conference*, Dig. Tech, pp. 402-403, 2006, Feb. 2006.

[5] C.-C. Lin et al., "A 160kGate 4.5kB SRAM H.264 Video Decoder for HDTV Applications," in *IEEE Solid-State Circuits Conference*, Dig. Tech, pp. 406-407, Feb. 2006.

[6] C.-D. Chien et al., "A 252kgate/71mW Multi-Standard Multi-Channel Video Decoder for High Definition Video Applications," in *IEEE Solid-State Circuits Conference*, Dig. Tech, pp. 282-283, Feb. 2007.

[7] Sze. V. et al., "A 0.7-V 1.8-mW H.264/AVC 720p Video Decoder." in *IEEE Journal of Solid-State Circuits*, vol.4, no.11, pp. 2943-2956, Nov. 2009.

[8] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," in *IEEE Transactions on*

Circuits and Systems for Video Technology, vol. 17, no. 9, pp. 1103-1120, Sep. 2007.

[9] H. Schwarz et al., "Constrained Inter-Layer Prediction for Single-Loop Decoding in Spatial Scalability, " in *Proceedings of IEEE International Conference on Image Processing*, vol. 2, pp. 870-873, Sep. 2005.

[10] H. Schwarz, D. Marpe and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103-1120, Sep. 2007.

[11] H. Schwarz et al., "Combined Scalability Support for the Scalable Extension of H.264/AVC," in *Proceedings of IEEE International Conference on Multimedia & Expo*, pp. 446-449, Jul. 2005.

[12] C. A. Segall and G. J. Sullivan, "Spatial Scalability Within the H.264/AVC Scalable Video Coding Extension," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1121-1135, Sep. 2007.

[13] P.-Y. Hsu, G.-L. Li and T.-S. Chang, "Memory Analysis for H.264/AVC Scalable Extension Decoder, " in *Proceeding of APSIPA Annual Summit and Conference*, pp. 299-302, Oct. 2009.

[14] Narvekar, N.D. et al., "An H.264/SVC memory architecture supporting spatial and course-grained quality scalabilities," in *IEEE Conference on Image Processing,* pp. 2661-2664, Nov. 2009.

[15] Y.-H. Chen, et al., "A cost-efficient residual prediction VLSI architecture for H.264/AVC scalable extension," in *Proc. of Picture Coding Symposium*, 2007

[16] T.-D. Chuang, et al., "A 59.5mW Scalable/Multi-view Video Decoder Chip for

Quad/3D Full HDTV and Video Streaming Applications," in *IEEE Solid-State Circuits Conference*, Dig. Tech, pp. 330-331, Feb. 2010.

[17] *SVC Reference Software,* JSVM 9.14.

[18] T.-W. Chen, et al., "Architecture Design of H.264/AVC Decoder with Hybrid Task Pipelining for High Definition Videos," in *Proceeding of IEEE International Symposium on Circuits and Systems*, pp.2931–2934, May. 2005.

[19] T.-A. Lin, et al., "An H.264/AVC Decoder with 4x4-block Level Pipeline," in *Proceeding of IEEE International Symposium on Circuits and Systems*, pp.1810-1813, May 2005.

[20] T.-C. Chen, C.-J. Lian, L.-G. Chen, "Hardware architecture design of an H.264/AVC video codec," in *Asia and South Pacific Design Automation Conference*, March. 2006.

[21] Y.-H. Liao, G.-L. Li and T.-S. Chang, "A high throughput VLSI design with hybrid memory architecture for H.264/AVC CABAC decoder," in *Proceeding of IEEE International Symposium on Circuits and Systems*, pp.2007- 2010, May. 2010.