國 立 交 通 大 學

電子工程學系電子研究所

碩 士 論 文

H.264/AVC可調性視訊編碼之分析與框內編碼器設計

Analysis and Intra Encoder Design for H.264/AVC Scalable Extension

研 究 生： 陳之悠

指導教授： 張添烜 教授

中華民國 九十八 年 八 月

H.264/AVC可調性視訊編碼之分析與框內編碼器設計

Analysis and Intra Encoder Design for H.264/AVC Scalable Extension

研 究 生： 陳之悠　　Student：Tzu-Yu Chen

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

國 立 交 通 大 學

電子工程學系 電子研究所

碩 士 論 文

A Thesis
Submitted to Institute of Electronics
College of Electrical and Computer Engineering
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in
Electronics Engineering
August 2009
Hsinchu, Taiwan

中華民國 九十八 年 八 月

H.264/AVC 可調性視訊編碼之分析與框內編碼器設計

研究生：陳之悠　　指導教授：張添烜博士

國立交通大學

電子研究所

摘　要

幾十年來，各式的視訊編碼規格被開發用以滿足人類對視訊的需求：高解析度、高畫質、高播放速率等。和之前的標準相比，H.264/AVC 可在相同的資料量下提供更高的視訊品質。而作為 H.264/AVC 的延伸，可調性視訊編碼被設計用來以單一編碼端滿足多重解碼端的資料需求，接收端可依本身的頻寬及系統處理能力，自固定的資料流擷取相對應的部分予以重建視訊。

由於在 H.264/AVC 的基礎上建構，可調性視訊編碼的複雜度遠大於前者。因此在維持或少量增加複雜度及系統規格的前提下，設計一個可支援高效能的可調性視訊編碼器，是項極富挑戰性的議題。本篇論文藉由一項關於可調性視訊編碼器的記憶體分析、一種框內編碼的快速演算法，以及一個可調性視訊框內編碼器的實作，解決了此一問題。

首先，一項關於可調性視訊編碼的記憶體分析將被整理歸納，藉由選用最適合的編碼流程，可得到最低的記憶體使用量及頻寬。而在導入一些改善方法的幫助下，以增加 8.3%內部記憶體的容量，再省下超過一半的外部記憶體頻寬。

其次，本篇論文提出一種框內編碼的快速演算法，利用整數離散餘弦轉換或哈達瑪轉換之結果，推測區塊的紋路，並藉此以選擇較少的候選模式做決策。在可接受的效能損失下，有效縮短約一半的編碼時間。

最後，一個結合快速演算法，可提供三種畫質的可調性視訊框內編碼器被實作出來。以巨圖塊層次的管線化架構進行編碼，避免了閒置時間並增加資料輸出量；此外在整體系統排程的規劃下，利用各元件的共用性，有效節省硬體資源。在 135MHz 的工作時脈之下，此編碼器每秒可處理超過 59 萬 4 千個巨圖塊，相當於 60 張 HD 1080p、SD 480p 以及 CIF 之總和。

簡而言之，本篇論文對於可調性視訊編碼之貢獻有三：其一是關於可調性視訊編碼之系統性的記憶體分析，其二是框內編碼的快速演算法，最後是一個可調性視訊框內編碼器的實作，藉低複雜度的計算及可接受的硬體成本，滿足在多重解析度下三種畫質的需求。

Analysis and Intra Encoder Design for H.264/AVC Scalable Extension

Student: Tzu-Yu Chen Advisor: Dr. Tian-Sheuan Chang

Institute of Electronics

National Chiao-Tung University

Abstract

For recent decades, many video coding standards have been developed to satisfy the demands of video: high resolution, high quality, and high frame rate. Compared to previous standards, the state-of-art coding standard, H.264/AVC, can provide higher video quality by the same data transfer rate. As an extension of H.264/AVC, scalable video coding (SVC) is designed to satisfy multiple demands from data receivers by a single encoder. A receiver extracts the fixed bit-stream according to the bandwidth and processing ability itself, and reconstructs the video.

Constructing on the fundamental of H.264/AVC, the complexity of SVC is much higher than former. Hence, it is full of challenge to design an SVC encoder to support high performance, with the same or slightly increased complexity and system requirement. In this thesis, that issue can be solved through a memory analysis on SVC encoder, a fast intra prediction algorithm, and the implementation of SVC fast intra encoder.

At first, the memory analysis of SVC encoder will be presented. Through choosing the most suitable encoding method, the system would enjoy the minimal

memory usage and bandwidth. Furthermore, by the aid of some improvement method, over the half of the external memory bandwidth can be saved with 8.3% increase in internal memory storage.

Then, this thesis proposed a fast intra prediction algorithm. The texture of a block can be predicted by observing the result of integer discrete cosine transform or Hadamard transform. Thus, the total encoding time can be shortened to the half effectively, by selecting fewer mode candidates in mode decision, with acceptable performance loss.

Last, an SVC encoder with 3 quality layers, combined with fast intra prediction algorithm is implemented. It operates under the pipeline architecture in macro-block level to avoid time idling and to increase data throughput. Besides, by the scheduling of the system, components are reused for effective hardware resource saving. This encoder processes more than 594k macro-blocks per second, which is equivalent to the summation of 60 HD 1080p, SD 480p, and CIF frames under the 135 MHz working frequency.

In brief, to SVC, the contributions of this thesis can be divided by three parts: the first is the analysis for SVC encoder in system level; the second is a fast intra prediction algorithm; the third and the last is the implementation of an SVC intra encoder. With the low complexity computation and acceptable hardware cost, it satisfies the demands for three video qualities in multiple resolutions.

誌　　謝

在此首先要感謝我的指導教授－張添烜博士。在研究所的生涯中，無論是研究或是求職，當我遇到問題時，總能給我很多建議與想法。老師的充分授權與信任尊重，掌握住研究的大方向與大原則，讓我得以在這浩瀚的領域中自由揮灑、激盪創意，也因此才能有本篇論文的產生。對老師的恩情與包容，我的感激並非字裡行間可以道盡。

同時也要感謝我的口試委員們，清華大學電機工程系陳永昌教授及交通大學資訊工程系彭文孝教授，感謝兩位能從百忙中專程抽空過來指導，教授們寶貴的意見與指教將使本篇論文更臻完備。

接著我要謝謝實驗室的同仁們。感謝李國龍學長，從我申請上研究所之後就進行前期的指導，引領我進入視訊處理這門學問的殿堂，並教導我如何做研究及找資料。感謝林佑昆學長和曾宇晟學長，在研究上以及軟體工具使用上給了我相當大的幫助。感謝蔡宗憲學長、詹景竹學長、戴瑋呈學長以及張瑋城學長，在我入學前的寒假開設訓練課程，教導我硬體設計的程式撰寫，至今都受用無窮。再來要感謝我的工作夥伴們，黃筱珊同學、許博淵同學、沈孟維同學，以及元歆、宥辰、孟勳等三位學弟，和你們共事的過程中我學到了很多，也成長了很多，日後回首這段日子，將會是個難忘的經驗。此外要感謝其他實驗室成員，張彥中學長、王國振學長、蔡政君同學，還有博雄、瑩蓉、奕均等學弟妹，你們的鼓勵與砥礪是我這兩年多的生活中不可或缺的一環。

最後要感謝我的家人們，我的父親、母親、弟弟，還有女友，你們默默不間斷的支持與關心，讓「之悠」沒有後顧「之憂」，是我能夠順利完成學業的最大動力。

碩士班的歲月，或許是帶點苦澀的短暫。因為大家的包涵，讓我可以一直帶著希望走到這一步。因此在這裡謹以這篇論文獻給所有愛我以及支持我的人。

Content

List of Figures

(Chapter 2)

(Chapter 3)

(Chapter 4)

(Chapter 5)

List of Tables

(Chapter 3)

(Chapter 4)

(Chapter 5)

Chapter 1 Introduction

With the prosperity of portable devices, digital televisions, and internet video, the applications of digital video become diversified. To integrate the processing of those 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), is used to offer flexible scalabilities in temporal, spatial and quality domains by a single bit-stream [1]. Receivers can partially extract the bit-stream according to their requirements.

1.1 Motivation

For video application, real-time processing seems to be the basic requirement today, and by the popularity of high-definition television (HDTV), over-30 Hz frame rate, which can offer more fluent visual entertainment, would be the trend in the future.

Most of the operations of video encoding are repeated, and can be processed in parallel under the pipeline architecture, therefore it is commonly implemented with application-specific integrated circuits (ASIC).

Due to the mass data access, memory analysis in SVC is often studied in many previous researches [2], [3]. However they either displayed an overall statistical data or just focused on a part of the system, none of them considered memory requirement from system level in detail. Hence, before the hardware implementation of SVC, it is necessary to analyze the overall memory usage and bandwidth of the system.

To overcome the limitation of average processing time, how to select prediction

mode smart is an important issue. Thus, the development of a fast algorithm would provide a key to solve this problem.

Besides, in architecture design, in order to support such high throughput rate, hardware utilization would also be a critical concern. To dispose resource reasonably under the tight system scheduling is really a big challenge.

1.2 Thesis Organization

This thesis is organized with five parts: Chapter 1 introduces the motivation of this work, and Chapter 2 gives a briefly overview on SVC standard. In Chapter 3, the memory analysis on SVC encoder is proposed, with the improvement in memory storage and bandwidth. Then in Chapter 4, a fast intra prediction algorithm “TraDED” is presented, for achieving the higher system requirement. Chapter 5 shows the hardware design and implementation results. Finally, the conclusion will be given in Chapter 6.

Chapter 2 Overview of SVC Standard

SVC, the next-generation video coding standard, is inherited from H.264/AVC [4]. Unlike the earlier standards listed in Fig. 1, SVC focuses on the integration of scalability instead of improving coding efficiency. To satisfying all different requirements, it tries to combine the scalability on temporal, spatial, and quality domain into a single encoding. The word "scalable" means users can have flexible extraction on the fixed bit-stream.

Fig. 2 gives an example of SVC: in the encoder side, the sequence with the highest resolution, highest frame, and highest quality is prepared. Encoder uses this sequence to produce several packets and composes a single bit-stream. Decoders extract the fixed bit-stream according to their requirements and ability. For instance, a HDTV has a wider bandwidth and a specific processor for video decoding, so it might receive all packets and reconstruct the sequence with the highest resolution, highest frame rate, and highest quality. For a smart phone, due to the restriction on the screen size, it would focus on receiving the packets for CIF size and reconstructs the sequence with the finest quality in 30 Hz frame rate.

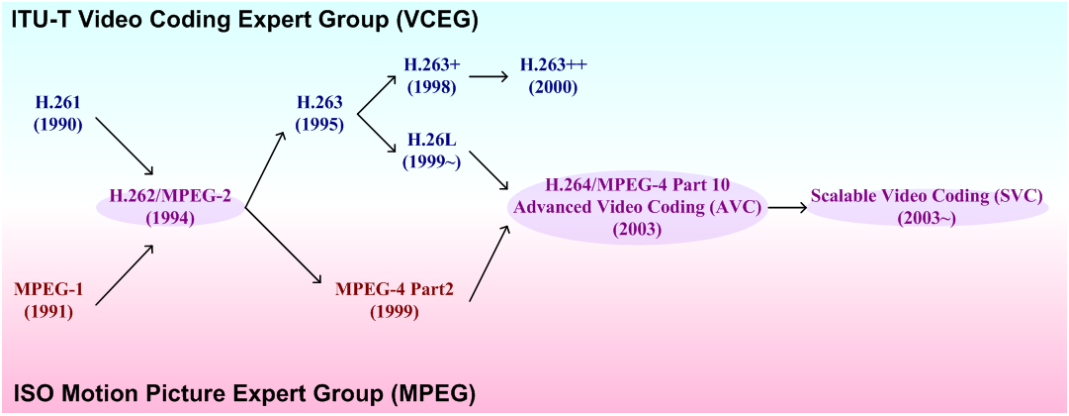


Fig. 1 The history of video coding standards

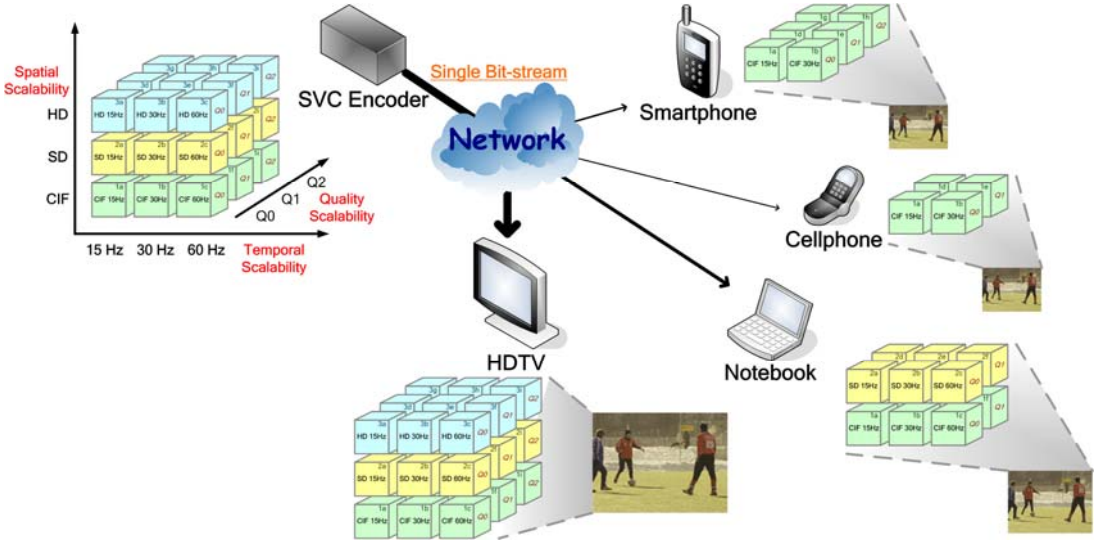


Fig. 2 The concept of SVC

2.1 Fundamentals of SVC

Fig. 3 shows the architecture of an SVC encoder with two spatial layers and two quality layers. The orange block in figure indicates a typical H.264/AVC encoder. At the beginning, the resolution of original input sequence would be down-sampled. Encoder processes these down-sampled data and put them to the bit-stream, like the way how H.264/AVC does. Quality enhancement coding deals with the transformed coefficients and produces the bit-stream to increase visual quality. For the encoding of original input sequence, prediction information, like motion vectors, residuals, and texture, can be obtained from base layer. In temporal domain, hierarchical coding structure [5], [6], shown on Fig. 4, is adopted. Under this structure, P or B frames are selected according to the coding requirement (speed or efficiency).

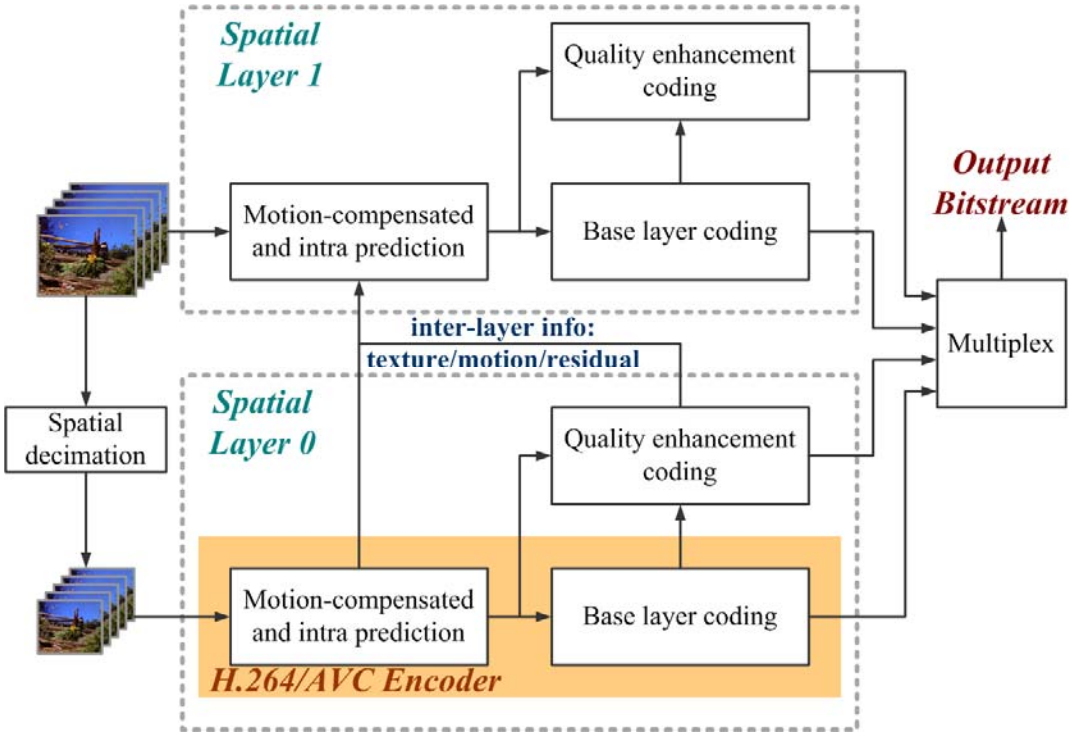
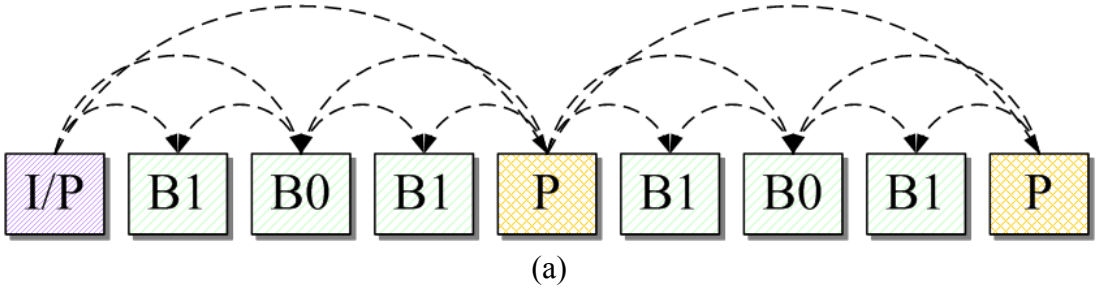
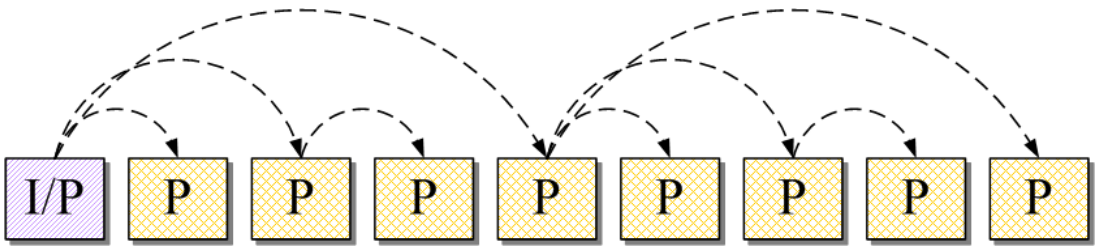


Fig. 3 Architecture of an SVC encoder



(a)



(b)

Fig. 4 Hierarchical coding structure for temporal scalability: (a) hierarchical-B structure; (b) low-delay coding structure

2.2 Components of SVC Encoder

Since SVC is constructed based on H.264/AVC, compensated-transform coding is adopted. In a typical H.264/AVC encoder (shown on Fig. 5), intra prediction (Fig. 6) and inter prediction (Fig. 7) would provide reference from spatial and temporal domain respectively. Then, the prediction residual with the lowest cost would be processed by transform, which breaks the correlation among data. Quantization erases unimportant information in transform coefficients, and entropy coding deals with the quantized (scaled) transform coefficients, put them into the bit-stream. On the other hand, an encoder reconstructs data by inverse quantization, inverse transform, and compensation. Finally, a deblocking filter is used to cancel artifacts.

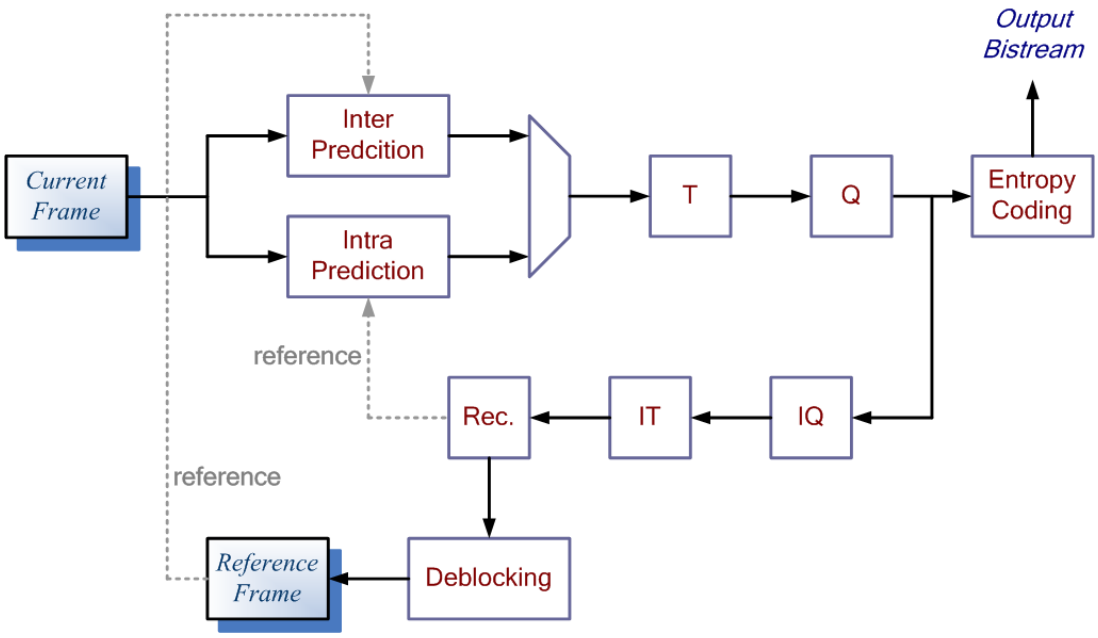


Fig. 5 Architecture of an H.264/AVC encoder

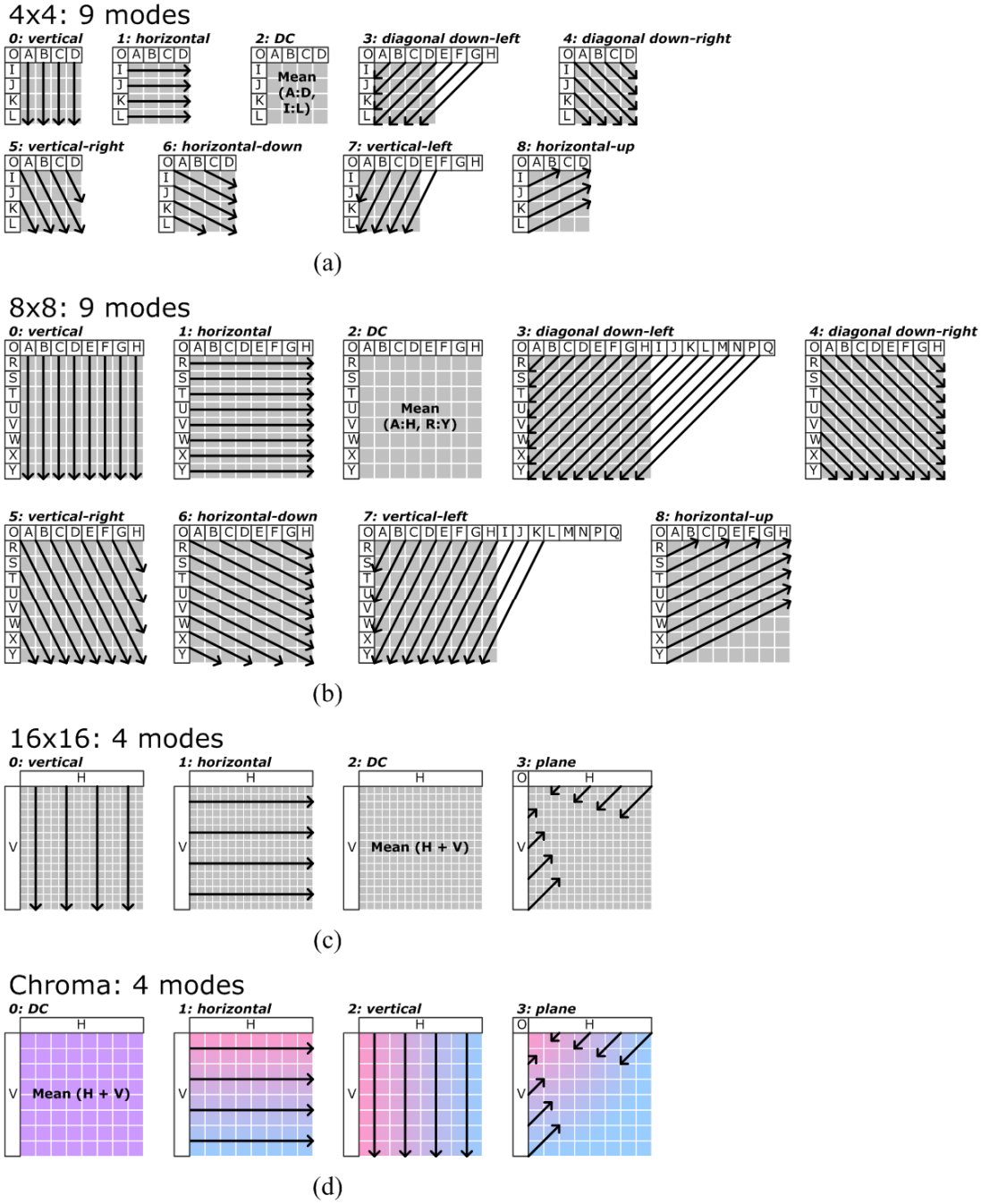


Fig. 6 Modes of intra prediction: (a) 4x4 block; (b) 8x8 block; (c) 16x16 block; (d) chroma block

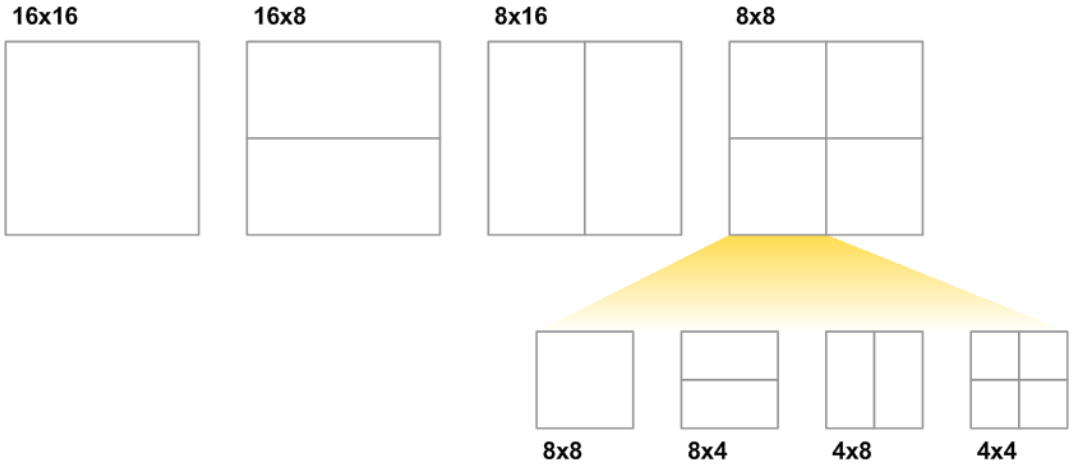


Fig. 7 Block partition of inter prediction

The concept of spatial scalability was firstly introduced in H.264/MPEG-2 Video [7]. In SVC, it is achieved by inter-layer-prediction. An enhancement layer (EL) uses motion vectors, residues, and texture information from base layer (BL). The prediction modes of inter-layer prediction are summarized in Fig. 8.

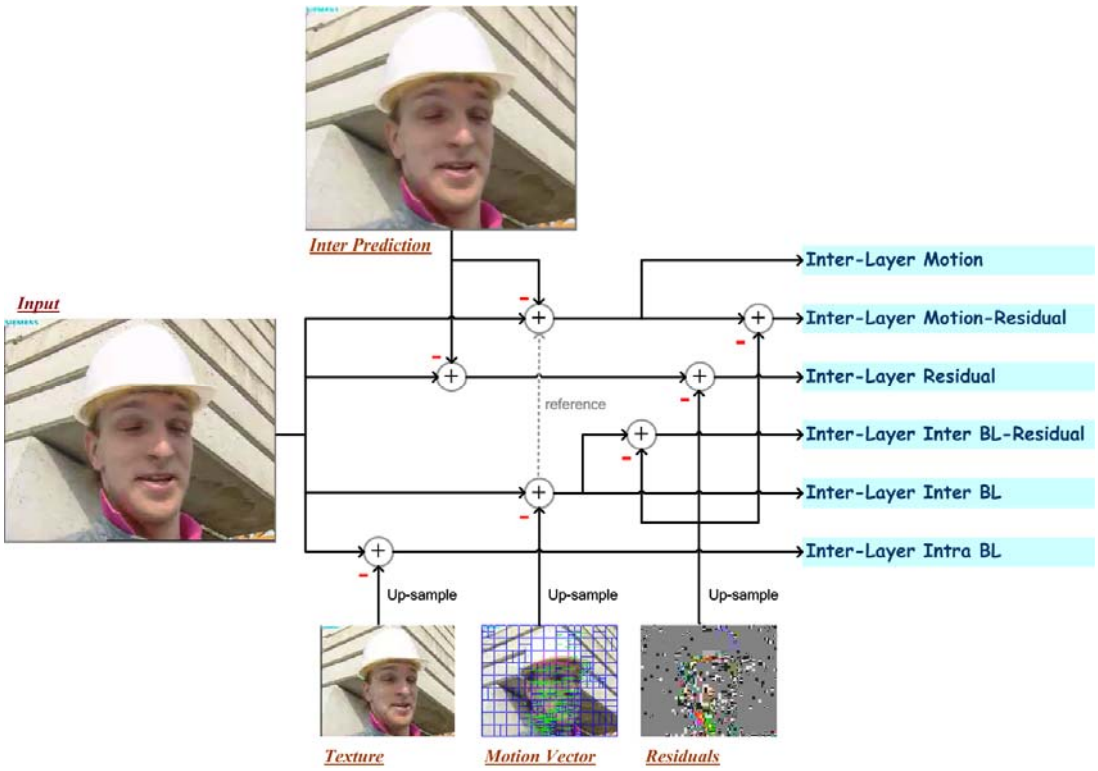


Fig. 8 Prediction modes of inter-layer prediction

Quality scalability can be treated as a special case of spatial scalability with the same resolution between BL and EL [1]. Quality enhancement coding refines visual quality of reconstructed picture by selecting smaller quantization parameter (QP) in re-quantization process to the rest of transformed coefficients (shown on Fig. 9). Therefore, some un-encoded messages can be obtained in EL.

Three coding strategies can be selected in quality enhancement coding: coarse-grain scalability (CGS), fine-grain scalability (FGS), and medium-grain scalability (MGS). The main difference between CGS and FGS is that: for entropy coding of EL, FGS processes coefficients by frame-level, instead of by MB-level in CGS. Thus, the quality improvement in FGS coding is continuous, but the drawback is it needs a large space for store coefficients, and has to deal with drift problem. MGS is a variation of CGS, which increases the number of rate-points. And as the same as FGS, it also has the drift problem between encoder and decoder.

The brief overview of SVC is described above, for more detail of SVC, please refer [1], [8], and [9].

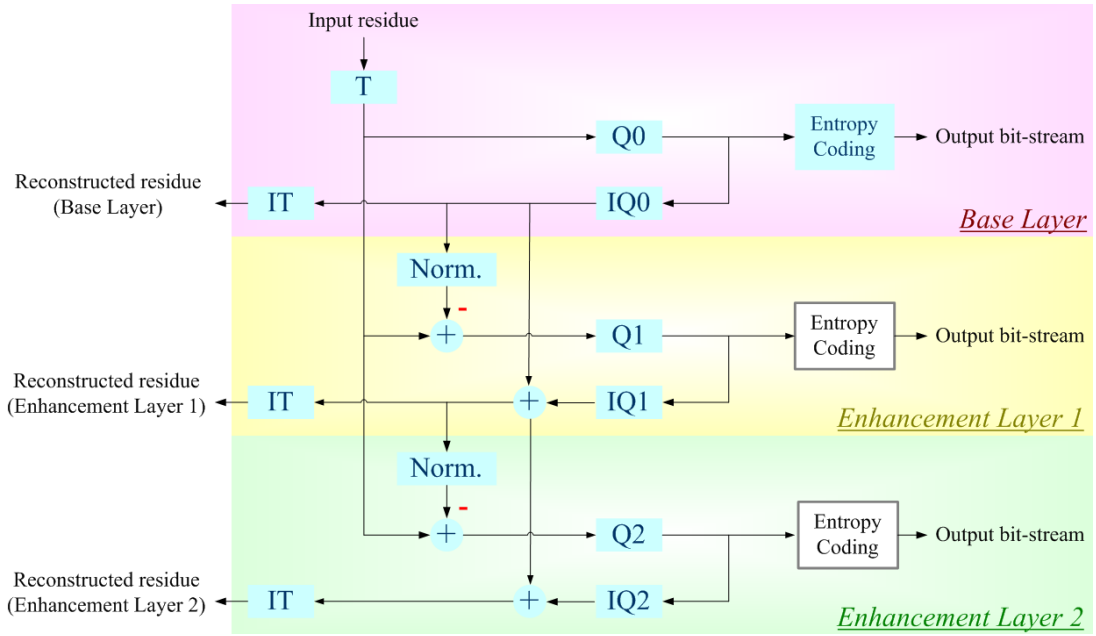


Fig. 9 Block diagram of quality enhancement coding in SVC

Chapter 3 Memory Analysis for SVC Encoder

From previous overview, it is obvious to see that the processing of SVC is complicated. Especially for hardware design, mass data access leads low memory bandwidth to be a main challenge. Based on this concern, “frame-parallel encoding scheme [10]” was proposed to save external memory bandwidth. Under this encoding scheme, by the aid of double hardware cost, two B frames can be encoded simultaneously. The most significant benefit for memory access saving is that the search window can be shared by two frames.

For instance, with temporal scalability equals to 3, three encoding schemes meet the requirement. Fig. 10(a) is traditional hierarchical-B structure with GOP size of 4; Fig. 10(b) and (c) are two examples for frame-parallel encoding scheme. In Fig. 10(b), the search window data from I/P and P frame can be shared by two B1 frames. The blue and red arrows indicate this reference. Fig. 10 also infers that if the GOP size increases, more data can be shared.

Since memory resource distribution is one of the important topics in SVC, therefore, before the implementation on SVC, the memory issue on SVC would be discussed in this chapter. The research on this topic is also published in [11].

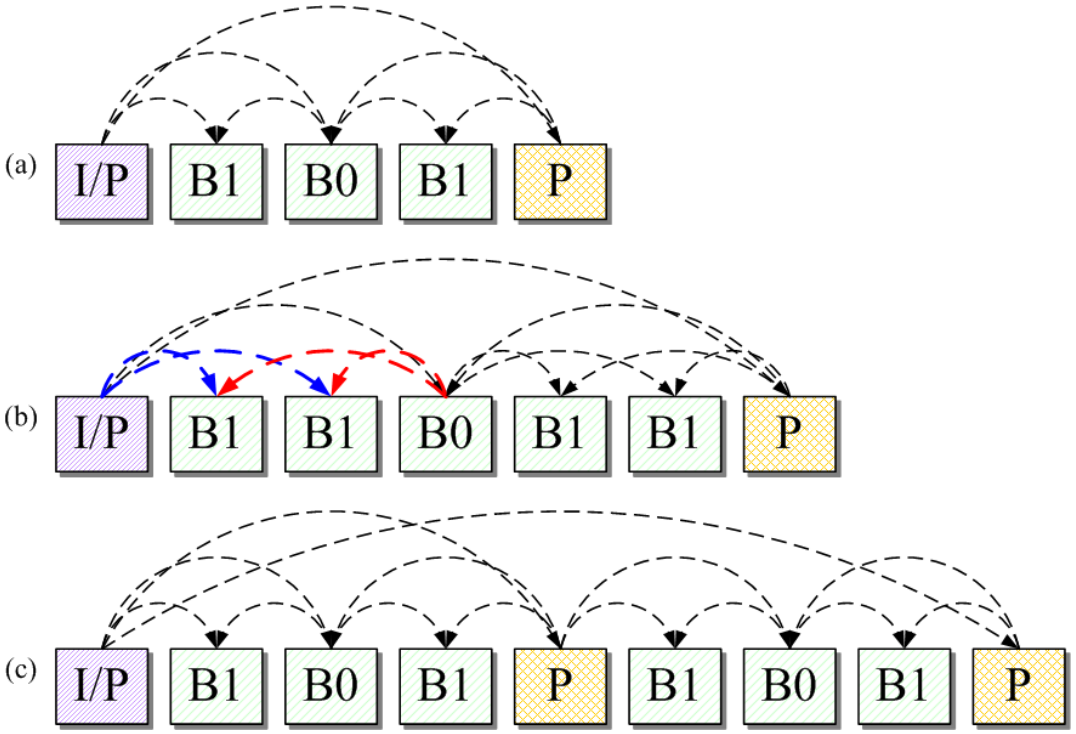


Fig. 10 Three encoding schemes with different GOP size: (a) GOP size = 4; (b) GOP size = 6; (c) GOP size = 8

3.1 Coding Flow

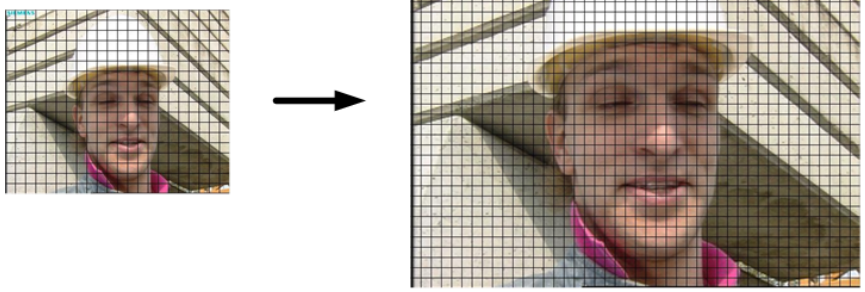
In the following analysis, for the maximum memory reuse, GOP size of 8 is set (Fig. 10(c)). For convenience, a set of frame (SOF) is defined as the 3 spatial input layers with the same temporal order of the input sequence.

3.1.1 Frame-Level Method

There are three encoding methods in frame data processing. The first one is called "frame-level" method, shown on Fig. 11. In this coding method, the spatial enhancement layer would be encoded after the entire reference layer is encoded. To examine the internal storage under this coding method, all variable terms are listed in Table 1. This table indicates that: the search window size would be the maximum of the P frame search window size and the total B frame search window size; the storage

size for motion vector, pre-deblocking (PDB) data, mb type, sub-mb type, and other neighboring information, would all be derived as the terms of the width of the largest frame in the unit of macro-block.

(a)



(b)

```
while(spatial_frame_index < n){
  while(CurrMbAddr < PicSizeInMbs){
    encode(CurrMbData);
    CurrMbAddr++;
  }
  spatial_frame_index++;
}
```

Note:
spatial_frame_index: *indicate which spatial layer is encoded currently*
n: *numbers of spatial layer*
CurrMbAddr: *current MB address*
PicSizeInMbs: *picture size in the unit of MBs*
CurrMbData: *current MB data*

Fig. 11 Frame-level method: (a) graphical illustration; (b) pseudo code

Table 1 Internal memory usage under the frame-level coding flow

Name	Bytes
Search window	max(3*SW_{wb}*SW_{hb}, SW_{wp}*SW_{hp})
Motion vector	144*($W_{mb,\ max}$ + 2)
PDB coefficient (row and column data)	256*m*($W_{mb,\ max}$ + 1)
mb_type and sub_mb_type	5.5*($W_{mb,\ max}$ + 2)
Other neighboring info*	5.5*$W_{mb,\ max}$ + 9

***SW_{wb}**: the width of search window for B frame*

***SW_{hb}**: the height of search window for B frame*

***SW_{wp}**: the width of search window for P frame*

***SW_{hp}**: the height of search window for P frame;*

***$W_{mb,\ max}$**: the width of the largest frame in the unit of macro-block*

***m**: numbers of quality layer*

**includes coded block pattern, non-zero transform coefficient flags for deblocking process, and neighboring intra prediction modes*

In the above case, PDB data are assumed to store in internal memory and they dominate the internal memory usage. For high resolution application, that term would be a serious burden. If PDB data are put in external memory, the corresponding storage size in internal memory would be a constant, the additional cost from PDB data, and other terms in external bandwidth can be seen on Table 2.

Table 2 External memory access per SOF under frame-level coding method

Name	Bytes
Input pixel	$1.5 * \sum_{i=0}^{n-1} W_i * H_i$
Reconstructed pixel	$1.5 * m * \sum_{i=0}^{n-1} W_i * H_i$
Search window (P frame)*	$\frac{1}{8} * (3 * \sum_{i=0}^{n-1} W_i * H_i - 32 * \sum_{i=0}^{n-1} W_i)$
Search window (B frame)*	$\frac{9}{8} * (2 * \sum_{i=0}^{n-1} W_i * H_i - 16 * \sum_{i=0}^{n-1} W_i)$
Chroma inter prediction	$\frac{81}{128} * \sum_{i=0}^{n-1} W_i * H_i$
FGS coefficient	$6 * (m-1) * \sum_{i=0}^{n-1} W_i * H_i$
Output bit-stream**	$0.15 * m * \sum_{i=0}^{n-1} W_i * H_i$
Inter-layer data	$2{,}186 * \sum_{i=0}^{n-1} W_{mb,i} * H_{mb,i}$
Pre-deblocking Coefficient***	$256 * m * \sum_{i=0}^{n-1} W_{mb,i} * (H_{mb,i} - 1)$

n: *numbers of spatial layer*

m: *numbers of quality layer*

Wi: *the frame width of the i-th spatial layer*

Hi*: the frame height of the i-th spatial layer*

Wmb,i: *the frame width of the i-th spatial layer in the unit of macro-block*

Hmb,i: *the frame height of the i-th spatial layer in the unit of macro-block*

**: the search range in P frame and B frame is set to 128 * 48 and 64 * 32 respective*

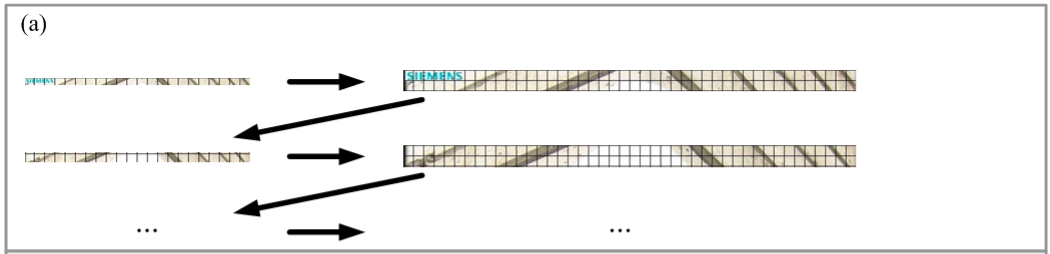
For the frame height which is not fully divided by 16, these formulas may need extra amendments

***: the output bit-stream of each quality layer is conservatively estimated as 10% of the original input*

3.1.2 Row-Level Method

The second method is called “row-level” method, and is exposed on Fig. 12. In this coding method, the corresponding rows of macro-block would be encoded after a row of macro-block in reference layer. If the PDB data are put in internal memory, the variable terms of internal memory usage would be shown on Table 3. Compare with Table 1, the formula for search window is the same, but for other formulas, the term “$W_{mb,\ max}$” in Table 1 is replaced by a summation term “$\sum_{i=0}^{n-1} W_{mb,i}$” due to resolution switching.

On the other hand, the external memory bandwidth usage under this coding flow is as the same as the frame-level method, and can be referenced in Table 2.



(b)

```
while(spatial_frame_index < n){
  while(CurrMbAddr < PicWidthInMbs*(ver_upsamp_r^spatial_frame_index)){
    encode(CurrMbData);
    CurrMbAddr++;
  }
  spatial_frame_index++;
}
```

Note:
spatial_frame_index: *indicate which spatial layer is encoded currently*
n: *numbers of spatial layer*
CurrMbAddr: *current MB address*
PicSizeInMbs: *picture size in the unit of MBs*
CurrMbData: *current MB data*
PicWidthInMbs: *picture width in the unit of MBs*
ver_upsamp_r: *vertical up-sampling ratio*

Fig. 12 Row-level method: (a) graphical illustration; (b) pseudo code

Table 3 Internal memory usage under the row-level coding flow

Name	Bytes
Search window	$\max(3*SW_{wb}*SW_{hb}, SW_{wp}*SW_{hp})$
Motion vector	$144*\sum_{i=0}^{n-1}(W_{mb,i}+2)$
PDB coefficient (row data)	$256*m*\sum_{i=0}^{n-1}(W_{mb,i}+1)$
mb_type and sub_mb_type	$5.5*\sum_{i=0}^{n-1}(W_{mb,i}+2)$
Other neighboring info*	$\sum_{i=0}^{n-1}(5.5*W_{mb,i}+9)$

***SW_{wb}**: the width of search window for B frame*

***SW_{hb}**: the height of search window for B frame*

***SW_{wp}**: the width of search window for P frame*

***SW_{hp}**: the height of search window for P frame*

***$W_{mb,\ max}$**: the width of the largest frame in the unit of macro-block*

***m**: numbers of quality layer*

**includes coded block pattern, non-zero transform coefficient flags for deblocking process, and neighboring intra prediction modes*

3.1.3 MB-Level Method

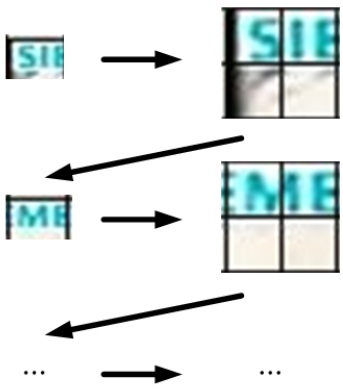
The third one, also the last one, is called “MB-level” method. In this coding method, a macro-block in reference block would be encoded at first, then comes the encoding of corresponding macro-blocks in enhancement layer. This coding flow is shown on Fig. 13. On internal memory storage, this method is similar with row-level method (please refer Table 3), but needs some extra amendments for additional space for column data.

In external memory access, compared to other methods, most of terms remain the same, except search window data. Due to the lack of data reuse, this term grows significantly, and can be derived on Table 4.

(b)

```
while(spatial_frame_index < n){
  while(CurrMbAddr < (ver_upsamp_r*hor_upsamp_r)^spatial_frame_index){
    encode(CurrMbData);
    CurrMbAddr++;
  }
  spatial_frame_index++;
}
```

(a)



Note:
spatial_frame_index: *indicate which spatial layer is encoded currently*
n: *numbers of spatial layer*
CurrMbAddr: *current MB address*
PicSizeInMbs: *picture size in the unit of MBs*
CurrMbData: *current MB data*
PicWidthInMbs: *picture width in the unit of MBs*
ver_upsamp_r: *vertical up-sampling ratio*
hor_upsamp_r: *horizontal up-sampling ratio*

Fig. 13 MB-level method: (a) graphical illustration; (b) pseudo code

Table 4 External memory access of search window data per SOF under the MB-level coding flow

Name	Bytes
Layer 0 in P frame	$\frac{1}{8}*(24*W_0*H_0-2*(128*W_0+384*H_0)+8192)$
Layer 1 in P frame**	$\frac{1}{8}*(40*W_0*H_0-2*(144*W_0+384*H_0)+5120)$
Layer 2 in P frame**	$\frac{1}{8}*(102*W_0*H_0-2*(176*W_0+624*H_0)+3584)$
Layer 3 in P frame**	$\frac{1}{8}*(346*W_0*H_0-2*(240*W_0+1344*H_0)+3584)$
Layer 4 and above in P frame**	$\frac{1}{8}*((21+3*2^i)*2^i*W_0*H_0-2*(16*((7+2^i)*W_0+168*2^i*H_0))+3584)$
Layer 0 in B frame	$\frac{9}{8}*(8*W_0*H_0-2*(32*W_0+64*H_0)+1024)$
Layer 1 in B frame**	$\frac{9}{8}*(16*W_0*H_0-2*(40*W_0+88*H_0)+768)$
Layer 2 in B frame**	$\frac{9}{8}*(53*W_0*H_0-2*(56*W_0+192*H_0)+768)$
Layer 3 and above in B frame**	$\frac{9}{8}*((6+2*2^j)*2^j*W_0*H_0-2*(16*((1.5+0.5*2^j)*W_0+48*2^j*H_0))+768)$

i: *the i-th spatial layer (i > 3)*

j: *the j-th spatial layer (j > 2)*

**: the search range in P frame and B frame is set to 128 * 48 and 64 * 32 respectively*

***: up-sampling ratio is set to 2 for horizontal direction and vertical direction*

For un-multiple up/down-sampling, these formulas may need extra amendments

3.2 Improvements

From previous analysis, there are three factors that dominate external memory bandwidth: reconstructed data, FGS coefficients, and PDB data. In the following section, some improvements are used or proposed to reduce the access on those terms.

3.2.1 Key Picture Concept

In previous deduction, memory usage is counted conservatively by using multi-loop reference, but if the “key picture concept [12] (shown on Fig. 14)” is adopted, I frames and P frames are regarded as “key pictures”, which use base layer as the reference. B frames are classified as “non-key pictures”, and the highest quality layers are used in their encoding.

This idea is extended in memory bandwidth reduction: since not being referenced, the frames in first quality enhancement layer and the B1 frames in the second quality enhancement layer in Fig. 14 will not be fully reconstructed. Also, all relative process after reconstruction (like deblocking) of those frames can be reduced. The only cost is that an extra space for a row of temporary reconstructed data is preserved in internal memory, for the use of intra prediction.

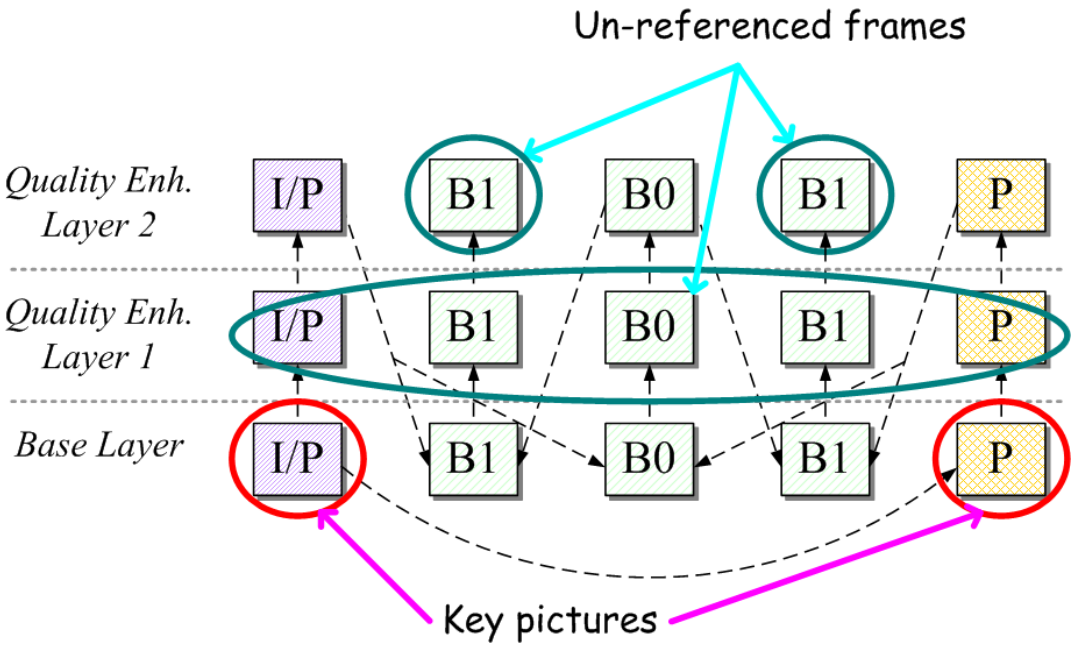


Fig. 14 Key picture concept

3.2.2 Shorten the Bit Length of FGS Coefficients

For frame-based FGS entropy coding, instead of internal memory, it is more efficient to store all FGS coefficients in external memory, and it consumes lots of external bandwidth. If they can be expressed in another way, the bandwidth utilization may be further reduced. Through a primary simulation, shown on Table 5, it is obvious to see that FGS coefficients are highly concentrated in a small range. The best case "Akiyo" and the worst case "Football" are plotted in Fig. 15 and Fig. 16. For the sequences with higher resolution, the correlation among neighboring pixels is much higher, and it is reasonable to say that the distribution would be more concentrated. By above information, it is worth to try to shorten the bit length of FGS coefficients from 2 byte to 4 bits, and that reduces 75 % of bandwidth in that term. (Note: this enhancement can be applied with the restricted QP step between two quality layers. In general, for a less-than-18 QP step, this enhancement works.)

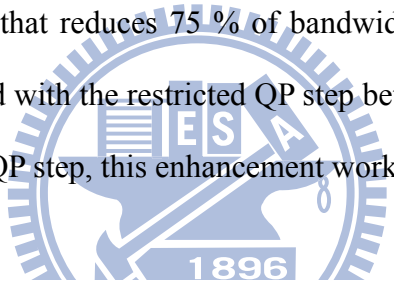


Table 5 Distribution of FGS coefficients in 5 CIF test sequences (first 100 frames)

Percentage of coefficient	**Coefficient Range**		
	-1 and 0	*-2 and 1*	*-8 and 7*
Akiyo	84.269	97.374	100.000
Football	62.203	91.595	100.000
Foreman	77.358	95.703	100.000
Mobile	63.178	91.742	100.000
Stefan	71.689	94.065	100.000

Note: the QP step between each layer is 12 (BL: 34; EL1: 22; EL2: 10)

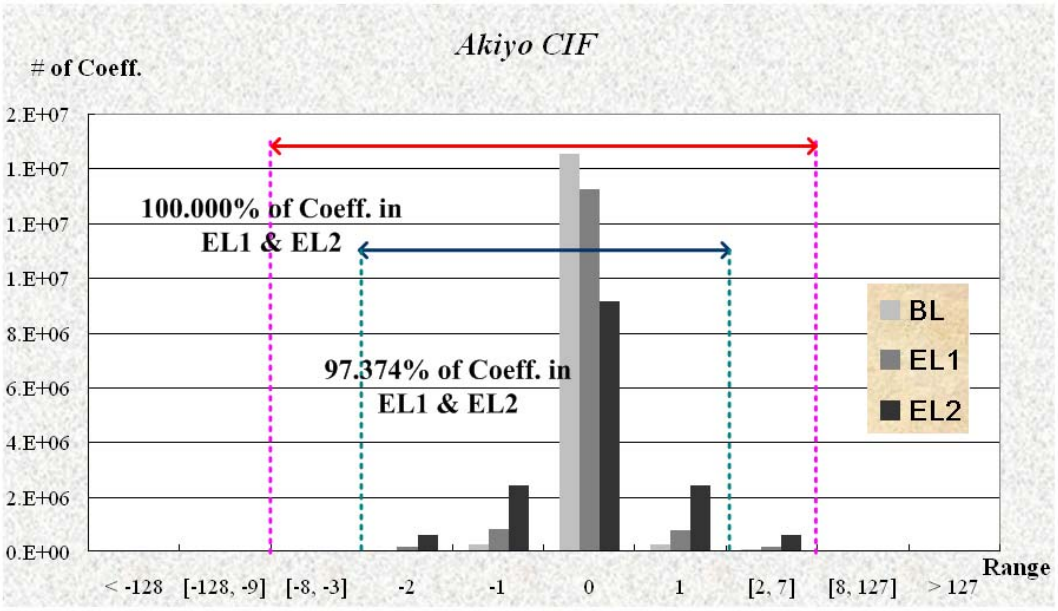


Fig. 15 The distribution of FGS coefficients in the test sequence "*Akiyo*"

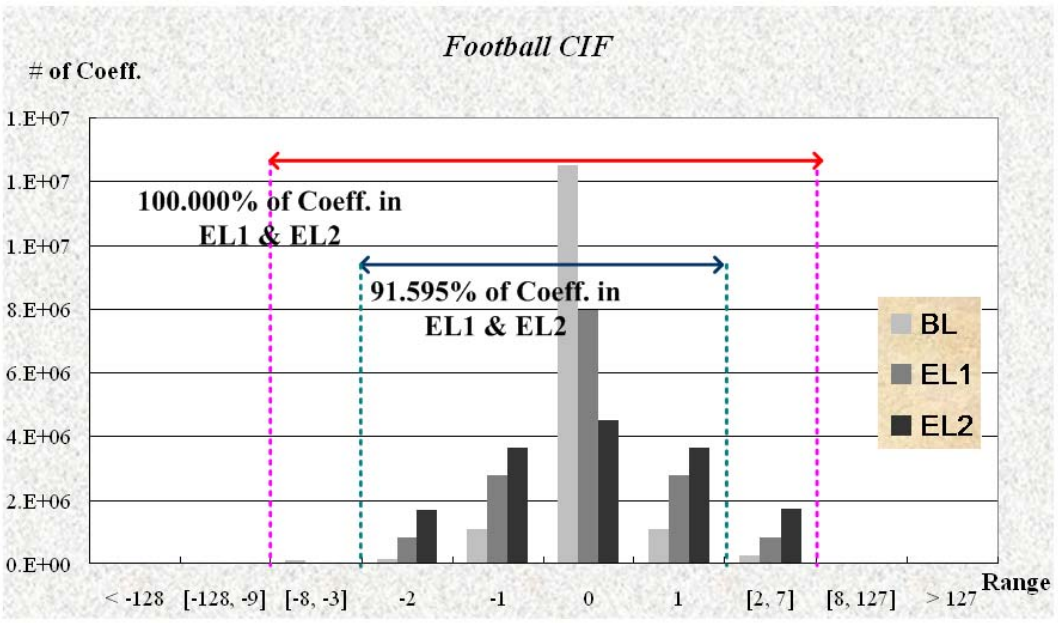


Fig. 16 The distribution of FGS coefficients in the test sequence "*Football*"

3.3 Experimental Results

The encoding process of following description is discussed in this section (also can be seen in Fig. 17): three spatial layers (CIF, SD 480p, and HD 1080p), three quality layers, and three temporal layer (GOP size of 8), with YUV 420 format in 30 Hz frame rate.

All previous analysis and improvements are applied in that case, and the result is summarized in Table 6. For the given condition, frame-level method, listed on row (1), would be the best one. When GOP size is equal to 8, the proposed coding flow is shown on Fig. 18, where the number in frame block represents the coding order.

Table 6 shows the comparison of different coding flow and the later improvements. In case (A), PDB data stored in internal memory, with improvements, more than 53 % of external bandwidth can be save, and furthermore, over 30 % of internal memory is reduced. On the other hand, in case (B), PDB data stored in external memory, about 56 % of external memory bandwidth is decreased, along with 8.3 % increase in internal memory size. By the aids of such improvements, the proposed method takes 38.38 % and 40.78 % of external bandwidth for case (A) and (B) respectively, with a 64-bit bus connected to a 266 MHz DDR DRAM, as shown on Fig. 19. Thus, this design is much easier to be implemented by the current chip technology.

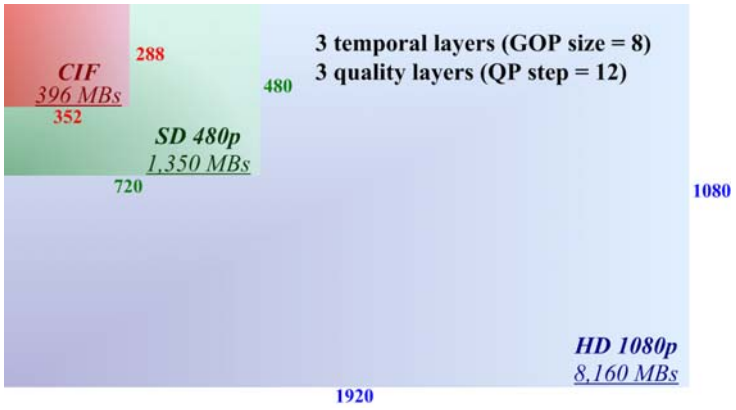


Fig. 17 Proposed input spec

Table 6 Memory performance of each coding flow under the given specification

Memory Usage	(A) PDB Data Stored in Internal Memory		(B) PDB Data Stored in External Memory	
	Internal Storage (KB)	***External Access per SOF (MB)***	***Internal Storage (KB)***	***External Access per SOF (MB)***
Frame-level (1)	133.49	55.78	44.24	62.90
Row-level (2)	200.55	55.78	55.05	62.90
MB-level (3)	203.55	60.64	59.55	67.76
Improved frame-level (4)	92.52	25.90	47.89	27.59
Change (1) vs. (4)	-30.7 %	-53.6 %	+8.3 %	-56.1 %

Note: Improve frame-level: frame-level method with improvement in Section 3.2

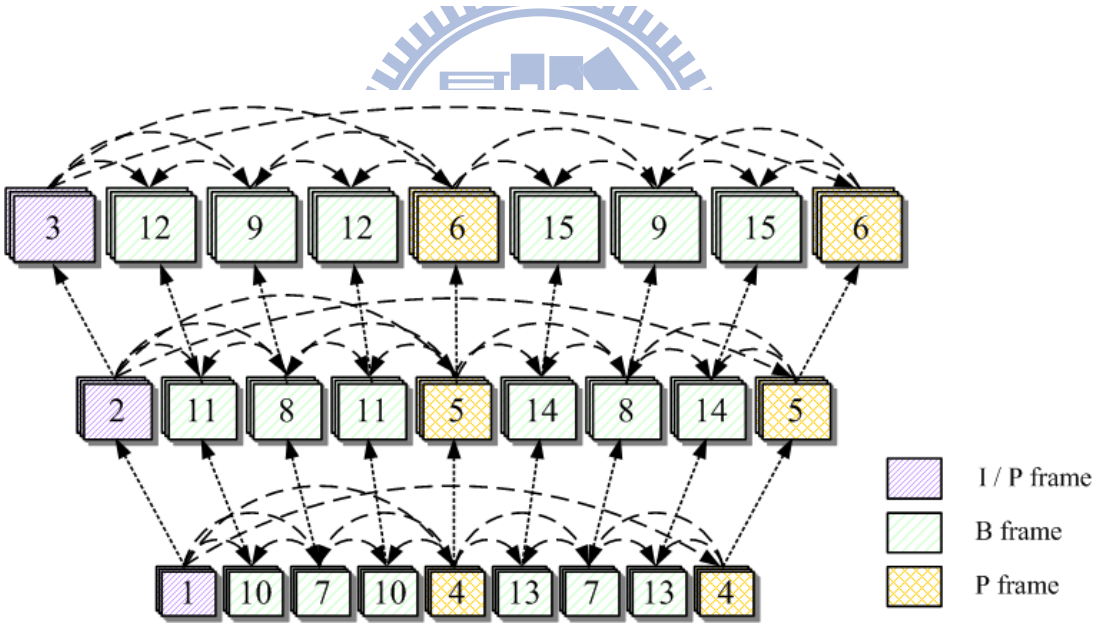


Fig. 18 Proposed coding flow

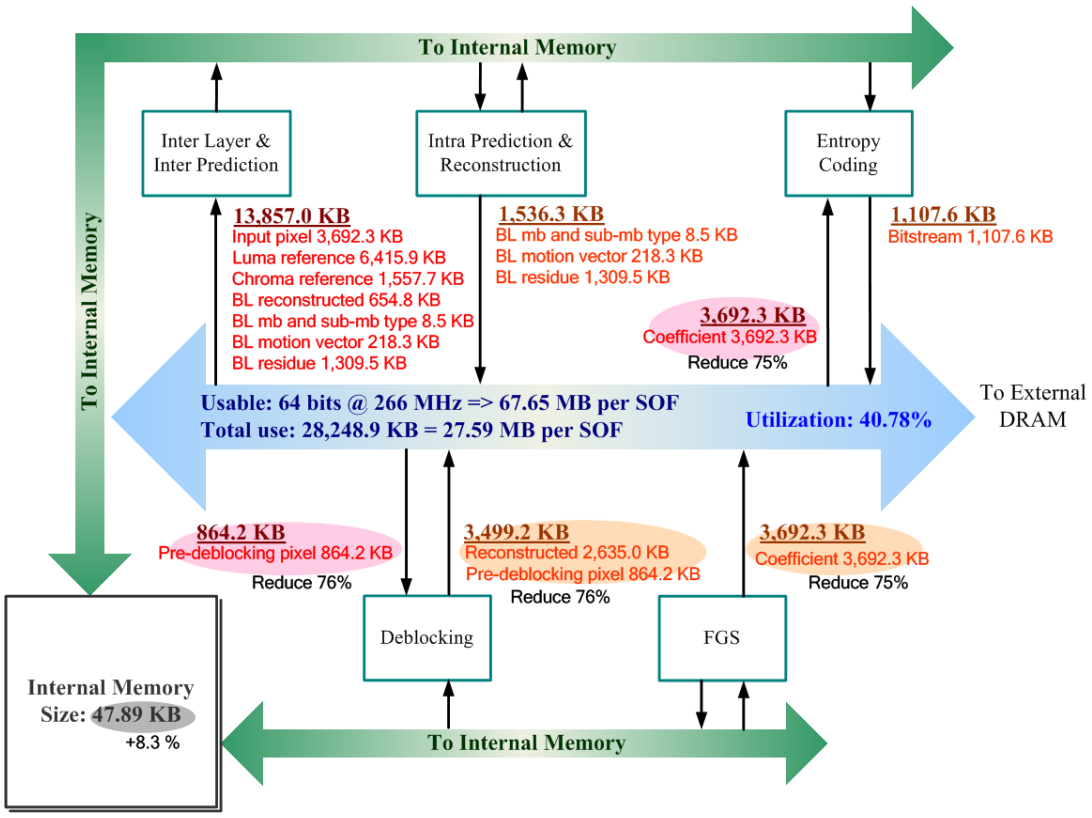


Fig. 19 The overall internal storage and external bandwidth of the proposed architecture in case (B) in Table 6

3.4 Summary

In this chapter, the memory performance of SVC is studied. In spatial domain, frame-level encoding enjoys the minimal internal memory and the lowest external memory bandwidth. With other improvements in quality and temporal domain, the external memory bandwidth can be further reduced over 53 %.

Chapter 4 Fast Intra Prediction Algorithm in SVC

Differ from inter prediction, intra prediction provides reference data from another way, spatial domain, combination of reconstructed pixels from the neighboring is used as the prediction result. Traditionally, intra prediction uses exhaustive search, which means, all available modes in each kind of block size would be examined one by one. However, in hardware design, full search may be not time-efficient. Thus, many fast algorithms have been developed to remain acceptable performance with the fewest number of mode candidates. In some of them, thresholds are defined for SAD cost [13] or for RDO [14] as early termination. In three-step fast algorithms [15] and "FIPMDA [16]", mode candidates are selected from the decision of previous step. Edge direction histogram [17] and dominant edge strength [18] also provides solutions in mode decision. Besides, intra prediction can be combined with transform process, transform results are taken for mode decision [19] or block size selection [20].

In this chapter, a fast intra prediction algorithm is proposed. Based on the algorithm in [21], edge detection in transform domain is modified for achieving the higher system requirement. The main difference is that "DC-dominant" condition is adopted in proposed algorithm, and the encoding for intra 8x8 prediction is considered in this work.

4.1 Fast Intra Prediction Algorithm with Transform Domain Edge Detection (TraDED)

In the following, the mode decision for intra 4x4, intra 8x8, intra 16x16, and intra chroma blocks are described separately.

4.1.1 Intra 4x4 Prediction

Consider the basis patterns of 4x4 DCT in Fig. 20(a), patterns can be classified into six categories (see Fig. 20(b)): DC, vertical (V), horizontal (H), diagonal (D), diagonal-vertical (DV), and diagonal-horizontal (DH). The intensities of those categories can be calculated as the sum of absolute values of corresponding coefficients:

$I_{DC} = |t00|$

$I_V = |t01| + |t02| + |t03|$

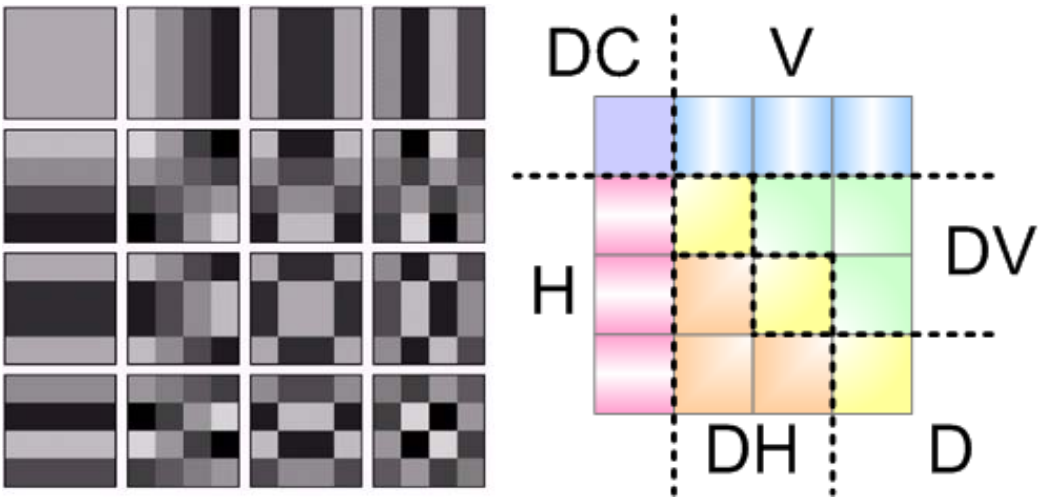
$I_H = |t10| + |t20| + |t30|$

$I_D = |t11| + |t22| + |t33|$

$I_{DV} = |t12| + |t13| + |t23|$

$I_{DH} = |t21| + |t31| + |t32|$

$I_{AC} = I_V + I_H + I_D + I_{DV} + I_{DH}$



(a) (b)

Fig. 20 4x4 DCT: (a) the basis patterns, and (b) categories of coefficients

Mode decision is made from the magnitudes of these intensities. For DC mode, if the block is on the boundary, in case that there are no mode candidates, DC mode must be turned on. Else, if the ratio (I_{DC} / I_{AC}) is smaller than a given threshold, which means the DC intensity is weak, the DC mode would be turned off.

On the other hand, in AC mode decision, if the ratio (I_{DC} / I_{AC}) is larger than another given threshold, DC intensity dominants the block edge direction, most of AC modes would be turn off, except vertical or horizontal mode. Else, AC candidate modes are chosen by the comparison of intensities I_V, I_H, I_D, I_{DV}, and I_{DH}.

In this case, if I_V is the maximum in $\{I_V, I_H, I_D\}$, and the ratio (I_V / I_H) is larger than 2, the vertical intensity is obvious. Hence, vertical mode (mode 0) and two adjacent modes (mode 5 and 7) would be chosen as the mode candidates. Similarly, if I_H is the maximum and the ratio (I_V / I_H) is less than 0.5, horizontal mode (mode 1) and two adjacent modes (mode 6 and 8) are selected. Else, neither vertical nor horizontal edges are regard as the major direction. Diagonal modes (mode 3 and 4) would be considered as the mode candidates. Then, depends on the relation between (I_V + I_{DV}) and (I_H + I_{DH}), different modes would be taken.

Besides, the most probable mode (mode N) would be selected for intra prediction if the number of AC mode candidates are smaller than 4. That not only increases the coding efficiency, but also makes the total number of mode candidates be restricted in less than 5.

The mode decision of 4x4 intra prediction is summarized in Fig. 21. The qualification of DC mode and AC modes are decided separately. Generally, the threshold value *"th4DCdom"* is set larger than threshold *"th4DCoff"*. (Note: in the following simulation, *"th4DCdom"* is set to 16, and *"th4DCoff"* is set to 2.)

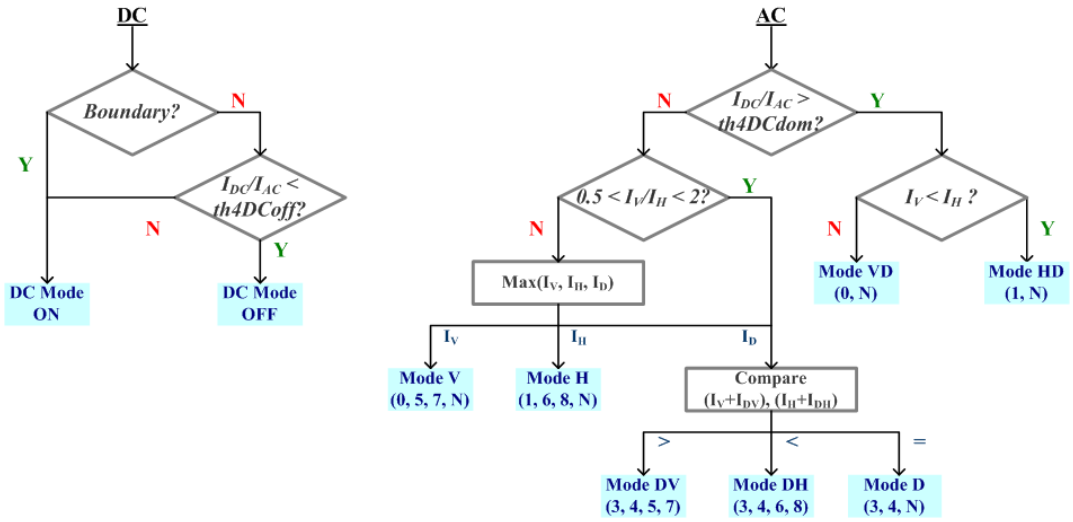


Fig. 21 Flow chart of intra 4x4 mode decision

4.1.2 Intra 8x8 Prediction

Same as intra 4x4 prediction, there are 9 modes for intra 8x8 prediction. Hence, the decision algorithm of intra 4x4 can be adopted in intra 8x8 with slightly modification. Consider the basis patterns of 8x8 DCT, shown on Fig. 22(a), six categories can be classified in similar way (please refer Fig. 22 (b)). Differ from intra 4x4 mode, intra 8x8 mode is usually selected by the macro-blocks with less variation among pixels, high-frequency-related patterns can be ignored during intensity calculation. The calculations of intensities are listed in following:

$$I_{DC} = |t00|$$

$$I_V = |t01| + |t02| + |t03| + |t04| + |t05|$$

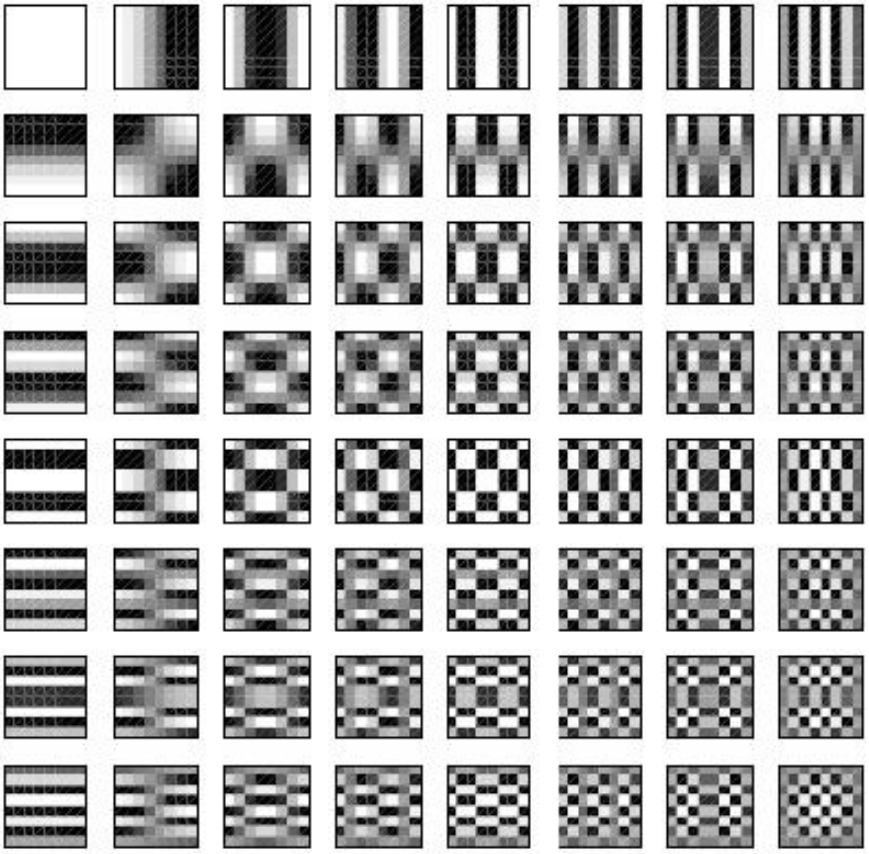
$$I_H = |t10| + |t20| + |t30| + |t40| + |t50|$$

$$I_D = |t11| + |t22| + |t33| + |t44| + |t45|$$

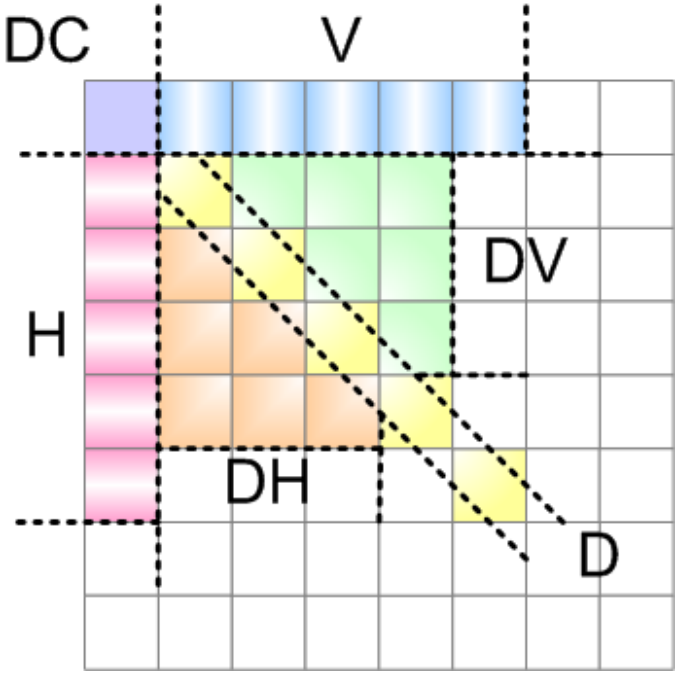
$$I_{DV} = |t12| + |t13| + |t14| + |t23| + |t24| + |t34|$$

$$I_{DH} = |t21| + |t31| + |t32| + |t41| + |t42| + |t43|$$

$$I_{AC} = \sum_{i=0}^{7} \sum_{j=0}^{7} |tij| \quad , for\ i, j \neq 0$$



(a)



(b)

Fig. 22 8x8 DCT: (a) the basis patterns; (b) categories of coefficients

Mode decision in intra 8x8 prediction (please refer Fig. 23 for flow chart) is almost as the same as the mode decision in intra 4x4 prediction. The only discrepancy between them is that the DC dominant threshold (*th8DCdom*) and DC off threshold (*th8DCoff*) need to be reset. (Note: in the following simulation, *"th8DCdom"* is set to 64, and *"th8DCoff"* is set to 2.)

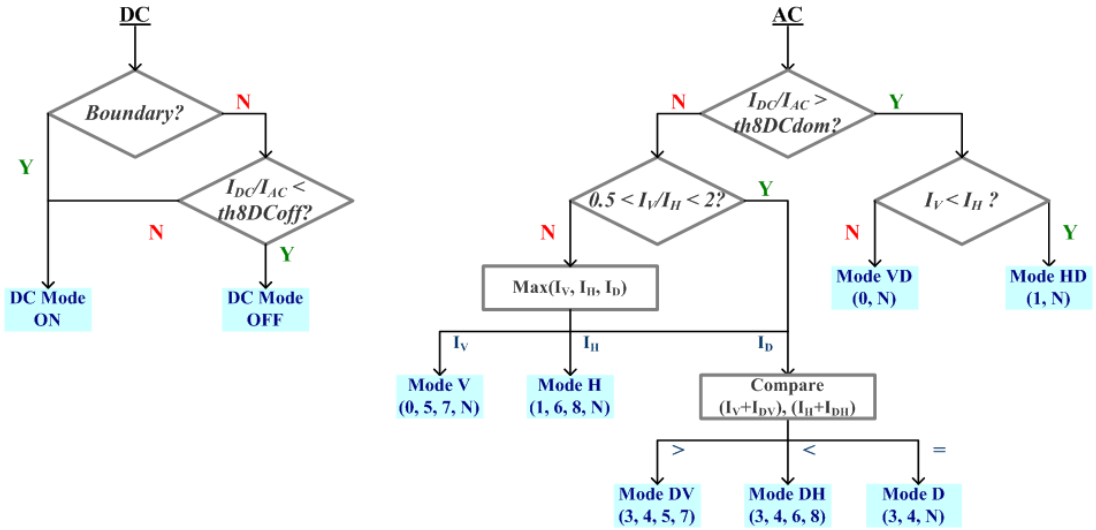


Fig. 23 Flow chart of intra 8x8 mode decision

4.1.3 Intra 16x16 Prediction

In intra 16x16 prediction, the DC coefficient of each 4x4 block is collected, then they will be processed by a 4x4 Hadamard transform. In the prediction of 16x16 block size, pixels tend to be highly correlated with each other, and that means DC coefficients would be much larger than AC coefficients. Hence, in edge detection, DC coefficients in Hadamard transform can be taken for simplicity.

The basis patterns of 4x4 DHT is shown on Fig. 24(a). To combine with 4 available modes, 4 categories are divided correspondingly, shown on Fig. 24(b). The intensities are calculated as following:

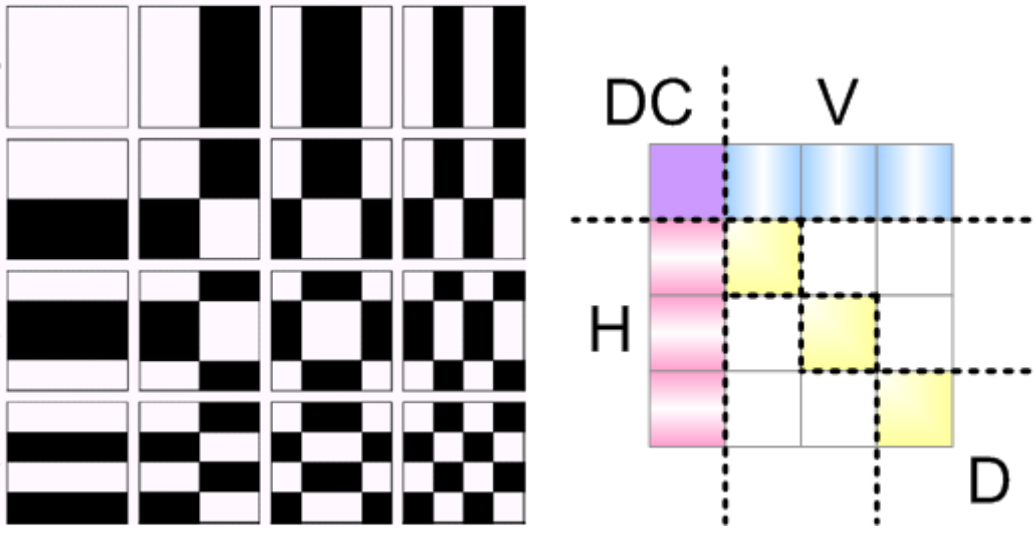
$$I_{DC} = |t00|$$

$$I_V = |t01| + |t02| + |t03|$$

$$I_H = |t10| + |t20| + |t30|$$

$$I_D = |t11| + |t22| + |t33|$$

$$I_{AC} = \sum_{i=0}^{3} \sum_{j=0}^{3} |tij| \quad , for\ i, j \neq 0$$



(a) (b)

Fig. 24 4x4 DHT: (a) the basis patterns; (b) categories of coefficients

Mode candidates are generated by these intensities. If the current block is not at the boundary, and the ratio (I_{DC} / I_{AC}) is smaller then a given threshold *"th16DCoff"*, DC mode would be turned off. Else, DC mode is always on. Another mode candidate is assigned to whom the maximum intensity among $\{I_V, I_H, I_D\}$ is related to. The mode decision of intra 16x16 prediction is summarized in Fig. 25. For the worst case, 2 modes would be chosen. (Note: in the following simulation, *"th16DCoff"* is set to 2.)

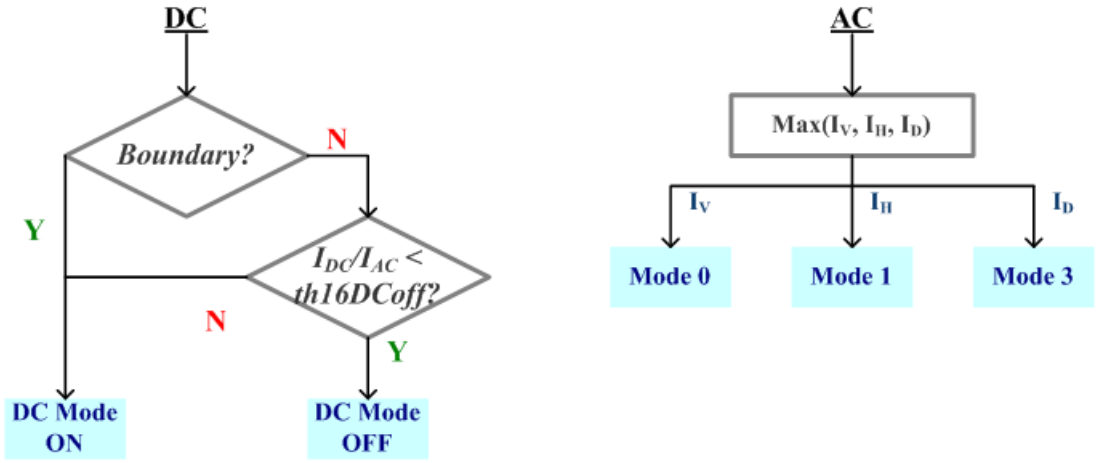


Fig. 25 Flow chart of intra 16x16 mode decision

4.1.4 Intra Chroma Prediction

Like intra 16x16 prediction, the DC coefficients in chroma parts have secondary processing, Hadamard 2x2 transform. For calculation simplicity, intensities are counted as the sum of two components (Cb and Cr) with the corresponding coefficient shown on Fig. 26, and listed below:

$$I_{DC} = |t00|_{Cb} + |t00|_{Cr}$$

$$I_V = |t01|_{Cb} + |t01|_{Cr}$$

$$I_H = |t10|_{Cb} + |t10|_{Cr}$$

$$I_D = |t11|_{Cb} + |t11|_{Cr}$$

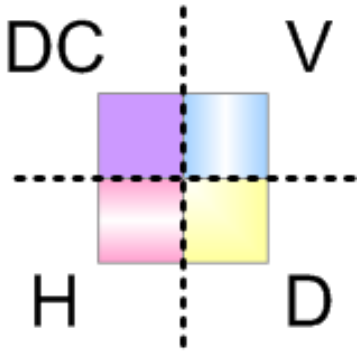


Fig. 26 Categories of coefficients in DHT 2x2

Fig. 27 indicates flow chart of mode decision in intra chroma prediction: DC mode is always on to ensure there are 2 mode candidates in non-boundary macro-blocks. Other intensities are compared together, and another mode candidate is decided by the largest intensity (I_H for mode 1, I_V for mode 2, and I_D for mode 3).

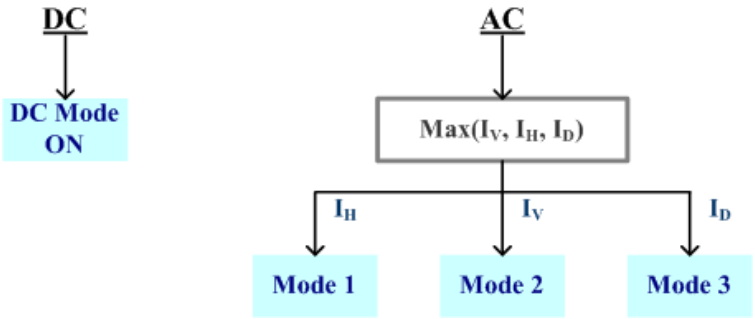


Fig. 27 Flow chart of intra chroma mode decision

4.2 Simulation Results

In the following simulation, 100 I-frames in 8 sequences with different resolutions are under tested. The comparison of bit-rate (BR) and PSNR between original result in JM 12.4 [22] (with the low-complexity mode) and proposed algorithm is listed in Table 7, the rate-distortion (RD) curves of them are plotted in Fig. 28 to Fig. 31, and the average mode candidates for each sequence is summarized in Table 8.

Table 7 Comparison between JM 12.4 and proposed algorithm for 100 I-frames

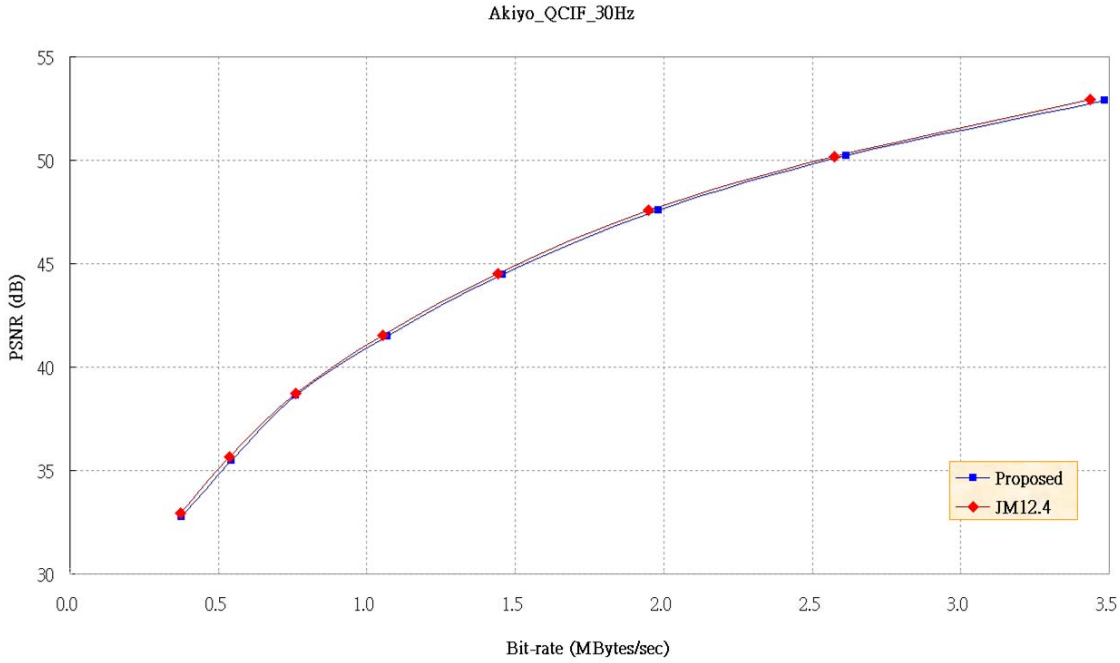
			QP							
			8	**12**	**16**	**20**	**24**	**28**	**32**	**36**
Akiyo (QCIF@30Hz)	**JM 12.4**	**BR (MBytes/s)**	3.436	2.575	1.949	1.440	1.054	0.761	0.536	0.374
		PSNR (dB)	52.933	50.194	47.605	44.524	41.536	38.715	35.678	32.949
	TraDED	**BR (MBytes/s)**	3.484	2.613	1.981	1.458	1.070	0.762	0.543	0.375
		PSNR (dB)	52.913	50.238	47.583	44.490	41.498	38.627	35.494	32.759
		ΔBR (%)	1.424	1.493	1.642	1.216	1.500	0.179	1.192	0.295
		ΔPSNR (dB)	-0.020	0.044	-0.022	-0.034	-0.038	-0.088	-0.184	-0.190
Foreman (QCIF@15Hz)	**JM 12.4**	**BR (MBytes/s)**	8.894	6.910	5.244	3.773	2.648	1.843	1.251	0.841
		PSNR (dB)	52.851	49.642	46.418	43.010	39.864	37.031	34.098	31.421
	TraDED	**BR (MBytes/s)**	8.917	6.951	5.293	3.794	2.692	1.880	1.283	0.868
		PSNR (dB)	52.872	49.705	46.447	43.001	39.920	37.082	34.130	31.341
		ΔBR (%)	0.256	0.584	0.935	0.574	1.649	2.024	2.533	3.203
		ΔPSNR (dB)	0.021	0.063	0.029	-0.009	0.056	0.051	0.032	-0.080
Mobile (CIF@30Hz)	**JM 12.4**	**BR (MBytes/s)**	28.211	23.462	19.294	15.346	11.897	9.014	6.546	4.521
		PSNR (dB)	52.777	49.444	46.080	42.319	38.656	35.222	31.666	28.419
	TraDED	**BR (MBytes/s)**	28.356	23.609	19.442	15.441	12.041	9.150	6.686	4.649
		PSNR (dB)	52.804	49.523	46.128	42.327	38.761	35.322	31.761	28.448
		ΔBR (%)	0.514	0.629	0.768	0.620	1.208	1.508	2.139	2.819
		ΔPSNR (dB)	0.027	0.079	0.048	0.008	0.105	0.100	0.095	0.029

Note: entropy coding method: context adaptive variable length coding (CAVLC)

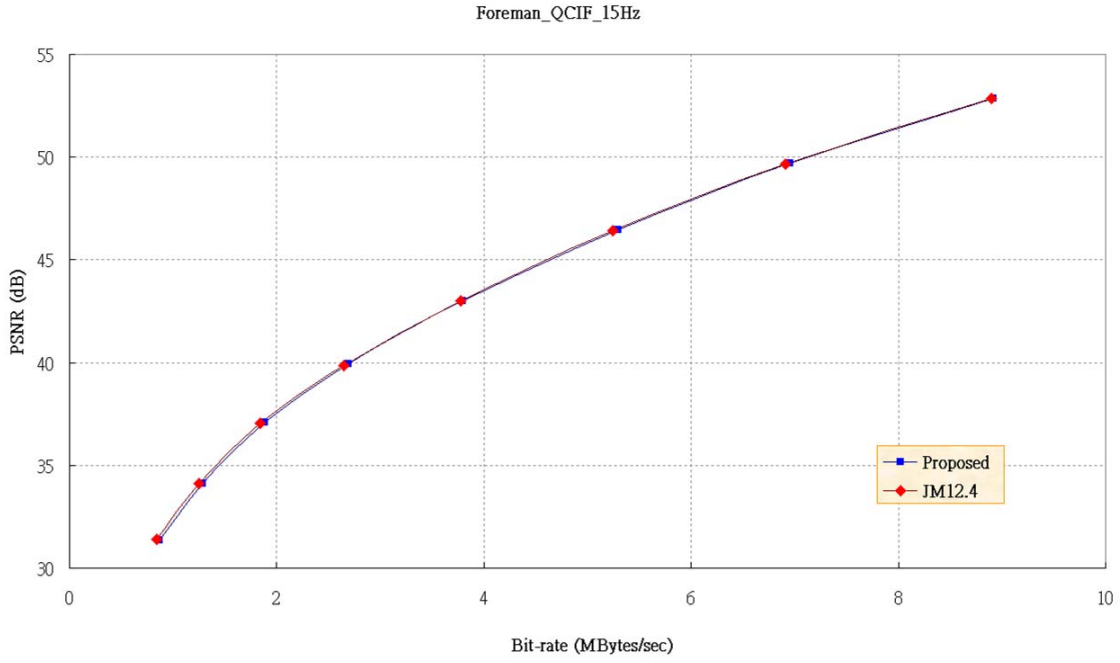
Table 7 (Cont.)

			QP							
			8	**12**	**16**	**20**	**24**	**28**	**32**	**36**
Stefan (CIF@30Hz)	**JM 12.4**	**BR (MBytes/s)**	21.831	17.495	13.838	10.700	8.156	6.054	4.290	2.930
		PSNR (dB)	52.797	49.612	46.458	43.017	39.621	36.376	32.979	29.872
	TraDED	**BR (MBytes/s)**	21.950	17.612	13.981	10.796	8.283	6.129	4.332	2.976
		PSNR (dB)	52.819	49.667	46.490	43.017	39.710	36.453	33.030	29.834
		ΔBR (%)	0.546	0.673	1.033	0.894	1.562	1.231	0.987	1.555
		ΔPSNR (dB)	0.022	0.055	0.032	0.000	0.089	0.077	0.051	-0.038
Shields (720p@30Hz)	**JM 12.4**	**BR (MBytes/s)**	199.10	156.02	114.67	76.75	49.00	31.67	20.48	13.07
		PSNR (dB)	52.691	49.300	45.803	42.167	39.087	36.572	34.048	31.710
	TraDED	**BR (MBytes/s)**	200.50	157.37	115.97	77.25	49.95	32,56	21.28	13.44
		PSNR (dB)	52.742	49.387	45.865	42.139	39.062	36.533	33.921	31.304
		ΔBR (%)	0.702	0.871	1.126	0.654	1.932	2.802	3.927	2.804
		ΔPSNR (dB)	0.051	0.087	0.062	-0.028	-0.025	-0.039	-0.127	-0.406
Stockholm (720p@30Hz)	**JM 12.4**	**BR (MBytes/s)**	195.23	153.09	113.84	76.35	47.20	28.33	17.08	10.71
		PSNR (dB)	52.667	49.255	45.713	41.910	38.650	36.107	33.791	31.739
	TraDED	**BR (MBytes/s)**	196.80	154.52	115.41	77.00	47.79	29.09	17.79	10.98
		PSNR (dB)	52.731	49.362	45.787	41.865	38.586	36.035	33.587	31.278
		ΔBR (%)	0.802	0.939	1.379	0.847	1.248	2.678	4.148	2.586
		ΔPSNR (dB)	0.064	0.107	0.074	-0.045	-0.064	-0.072	-0.204	-0.461
Sunflower (1080p@30Hz)	**JM 12.4**	**BR (MBytes/s)**	233.65	156.92	97.35	56.67	36.87	25.14	17.78	12.55
		PSNR (dB)	52.842	50.029	47.436	45.278	43.643	42.015	39.918	37.738
	TraDED	**BR (MBytes/s)**	234.03	157.66	99.01	57.60	38.36	26.29	18.32	12.69
		PSNR (dB)	52.826	50.034	47.453	45.266	43.565	41.823	39.591	37.285
		ΔBR (%)	0.161	0.471	1.709	1.643	4.059	4.582	3.061	1.179
		ΔPSNR (dB)	-0.016	0.005	0.017	-0.012	-0.078	-0.192	-0.327	-0.453
Tractor (1080p@30Hz)	**JM 12.4**	**BR (MBytes/s)**	320.80	237.40	167.33	110.90	72.58	46.88	30.29	20.01
		PSNR (dB)	52.964	49.846	46.814	43.791	41.189	38.915	36.574	34.392
	TraDED	**BR (MBytes/s)**	321.81	239.02	169.05	111.43	73.37	47.66	31.20	19.80
		PSNR (dB)	52.977	49.901	46.854	43.794	41.209	38.831	36.162	33.469
		ΔBR (%)	0.316	0.681	1.028	0.477	1.082	1.654	3.014	-1.050
		ΔPSNR (dB)	0.013	0.055	0.040	0.003	0.020	-0.084	-0.412	-0.923
Average: ΔBR = 1.452 %; ΔPSNR = -0.046 dB										

Note: entropy coding method: context adaptive variable length coding (CAVLC)

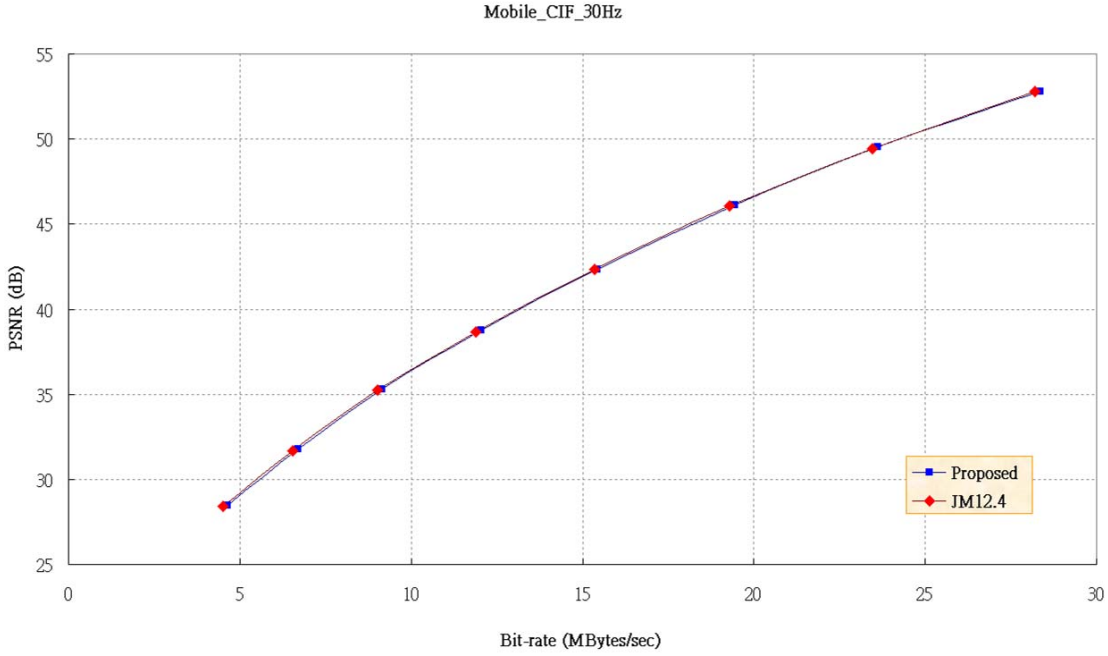


(a)

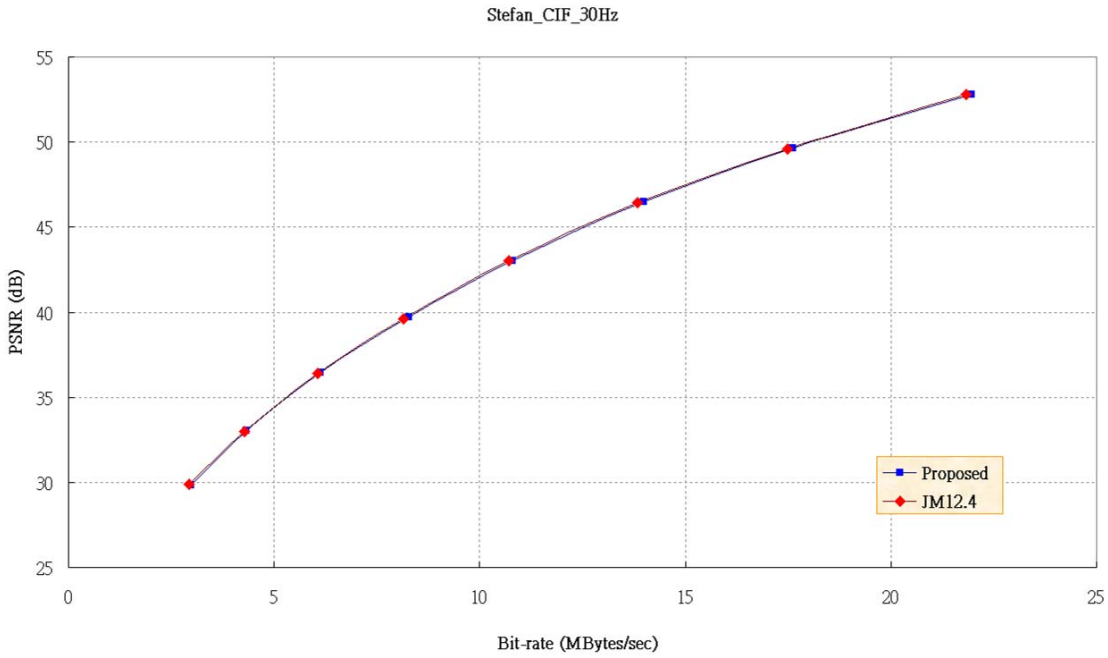


(b)

Fig. 28 RD curves of JM 12.4 and proposed algorithm for QCIF sequences: (a) Akiyo; (b) Foreman

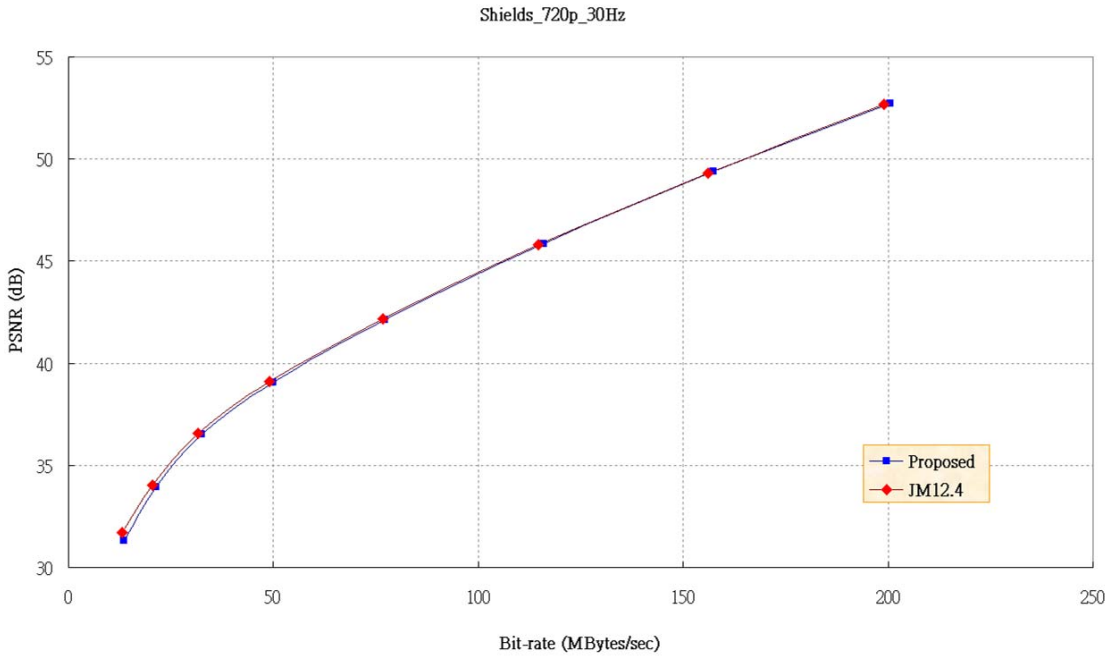


(a)

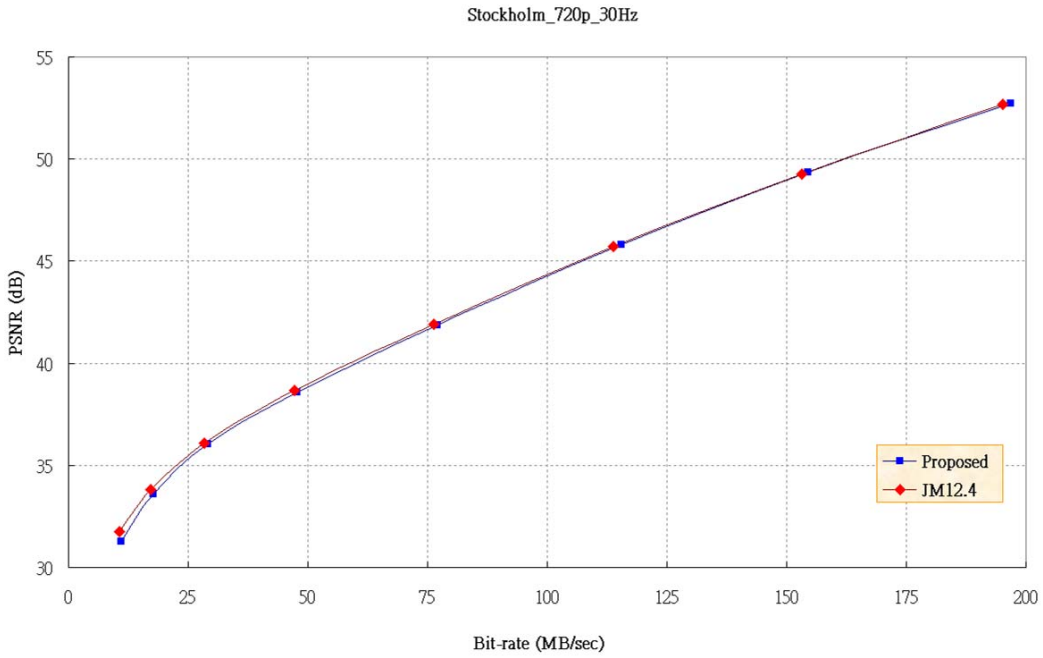


(b)

Fig. 29 RD curves of JM 12.4 and proposed algorithm for CIF sequences: (a) Mobile; (b) Stefan

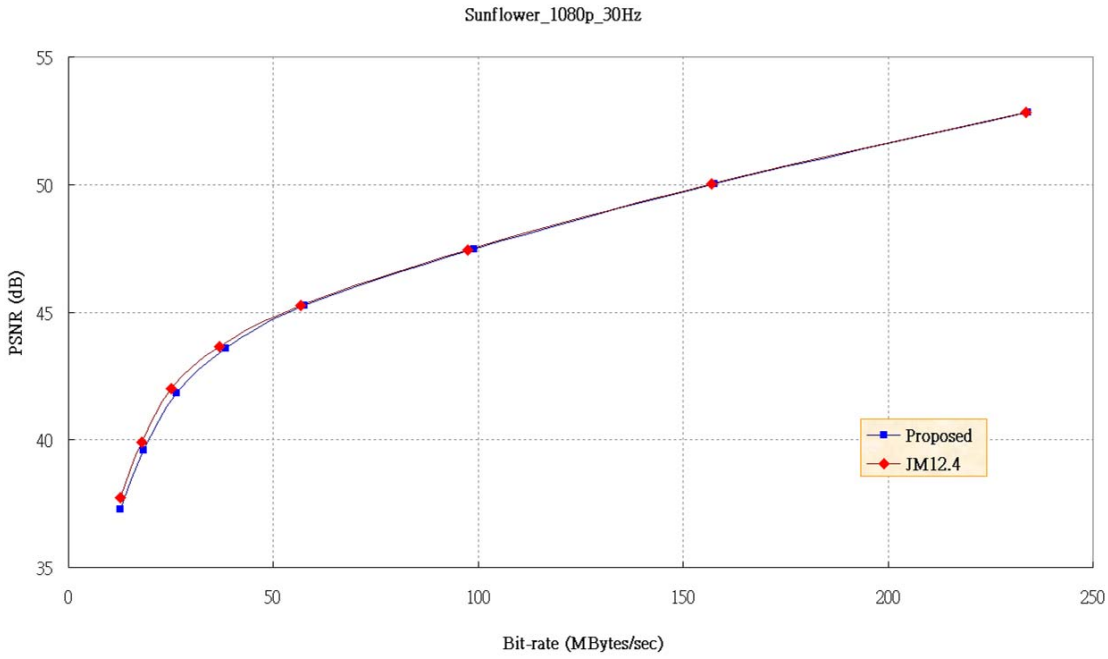


(a)

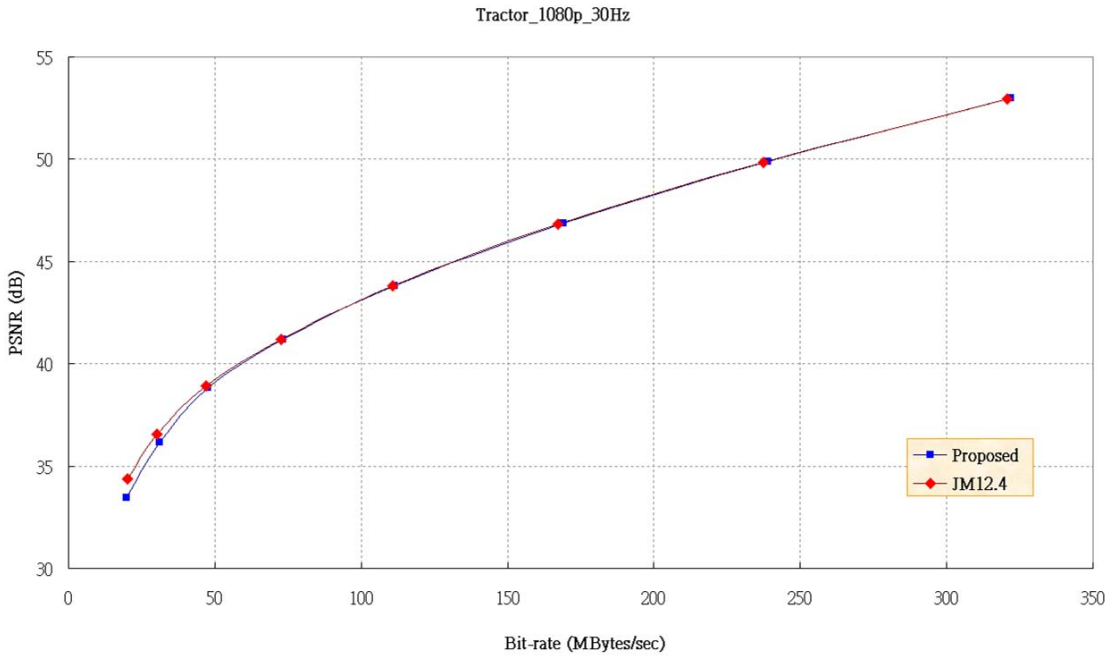


(b)

Fig. 30 RD curves of JM 12.4 and proposed algorithm for HD 720p sequences: (a) Shields; (b) Stockholm



(a)



(b)

Fig. 31 RD curves of JM 12.4 and proposed algorithm for HD 1080p sequences: (a) Sunflower; (b) Tractor

Table 8 Average mode candidates in encoding

Algo. / Seq.	JM 12.4 (Full search)				Proposed			
	4x4	8x8	16x16	Chroma	4x4	8x8	16x16	Chroma
Akiyo	8.72	8.45	3.61	3.61	3.60	3.96	1.67	1.92
Foreman	8.72	8.45	3.61	3.61	3.74	4.02	1.83	1.94
Mobile	8.86	8.72	3.80	3.80	3.70	3.83	1.62	1.95
Stefan	8.86	8.72	3.80	3.80	3.48	3.86	1.83	1.96
Shields	8.92	8.85	3.89	3.89	4.27	4.10	1.67	1.98
Stockholm	8.92	8.85	3.89	3.89	4.16	4.08	1.82	1.99
Sunflower	8.97	8.94	3.95	3.95	4.17	4.49	1.92	1.99
Tractor	8.97	8.94	3.95	3.95	4.20	4.24	1.89	2.00
Average	**8.87**	**8.74**	**3.81**	**3.81**	**3.92**	**4.07**	**1.78**	**1.97**
Savings (%)					**55.85**	**53.40**	**53.28**	**48.43**

From the above results, in high bit-rate points, the performance of "TraDED" pretty approaches the performance of full search algorithm in JM 12.4. However, for low bit-rate points, especially with high resolution, the performance of proposed algorithm is not as good as expected. The main reason is that in intra 16x16 mode, fierce hardware-oriented discard of single coefficient is adopted in proposed design, which causes the reduction of bit-rate and quality. (Note: in JM, if the total absolute value of non-zero coefficients is less than a threshold value, all coefficients would be set to zero for bit-rate reduction. For hardware design, it may take extra re-computation cycles or internal storage to deal with the condition for large block size. Therefore, in the proposed design, the threshold for non-zero coefficient discard is set to another value to a fixed block size).

It is worth to note that, in real SVC applications, high quality can be achieved by quality enhancement coding, and large resolution can be done by inter-layer prediction. Hence, considering the results of the sequences with smaller frame size (QCIF and CIF) with high QP, from 24 to 36, the average bit-rate increment would be

1.6 %, with slight quality gain (0.004 dB).

The comparison result with [21] is listed in Table 9 with four different QP values (from 8 to 32). Although based on the different simulation environments, this result still infers that: for low bit-rate points, "TraDED" has larger performance loss compared to JM, which is mainly caused by reason described previously. But for high bit-rate points, "TraDED" outperforms.

Table 9 Comparison between [21]* and "TraDED"** for ΔPSNR and ΔBR

Seq. \ QP		8		16		24		32	
		[21]	TraDED	[21]	TraDED	[21]	TraDED	[21]	TraDED
Akiyo	ΔPSNR	-0.155	-0.020	-0.073	-0.022	-0.042	-0.038	-0.030	-0.184
(QCIF)	ΔBR	0.272	1.424	0.762	1.642	0.751	1.500	1.020	1.192
Foreman	ΔPSNR	-0.154	0.021	-0.071	0.029	-0.041	0.056	-0.032	0.032
(QCIF)	ΔBR	0.150	0.256	0.430	0.935	0.739	1.649	0.879	2.533
Mobile	ΔPSNR	-0.203	0.027	-0.093	0.048	-0.073	0.105	-0.080	0.095
(CIF)	ΔBR	0.361	0.514	0.578	0.768	0.743	1.208	0.892	2.139
Stefan	ΔPSNR	-0.182	0.022	-0.072	0.032	-0.068	0.089	-0.077	0.051
(CIF)	ΔBR	0.301	0.546	0.613	1.033	0.805	1.562	1.100	0.987
Shields	ΔPSNR	-0.222	0.051	-0.096	0.062	-0.057	-0.025	-0.062	-0.127
(720p)	ΔBR	0.290	0.702	0.501	1.126	0.901	1.932	1.228	3.927
Stockholm	ΔPSNR	-0.234	0.064	-0.115	0.074	-0.052	-0.064	-0.040	-0.204
(720p)	ΔBR	0.203	0.802	0.357	1.379	0.946	1.248	1.370	4.148
Tractor	ΔPSNR	-0.156	0.013	-0.057	0.040	-0.035	0.020	-0.016	-0.412
(1080p)	ΔBR	0.044	0.316	0.540	1.028	1.131	1.082	1.311	3.014
Average	ΔPSNR	**-0.187**	**0.025**	**-0.082**	**0.038**	**-0.053**	**0.020**	**-0.048**	**-0.107**
	ΔBR	**0.232**	**0.651**	**0.540**	**1.130**	**0.859**	**1.455**	**1.114**	**2.563**

Note: measurement unit: ΔPSNR (dB), ΔBR (%)

**: implemented in JM 12.0, intra 8x8 mode is not included, and compared with JM 12.0 in RDO mode*

***: implemented in self-developed program, intra 8x8 mode is included, and compare with JM 12.4 in low complexity mode*

On the other hand, the comparison of average mode candidates between [21] and proposed algorithm is summarized in Table 10. For intra 4x4 and 16x16 prediction, thank the help of "DC dominant" condition, which selects only 3 mode candidates, "TraDED" further reduces about 15 % of mode candidates corresponding to the former.

Table 10 Comparison between [21] and "TraDED" for average mode candidates

		[21]			TraDED			
		4x4	16x16	Chroma	4x4	8x8	16x16	Chroma
QCIF	*Coastguard*	4.11	2.00	2.16	3.87	3.62	1.69	1.89
	Container	4.33	2.00	2.09	3.52	3.75	1.73	1.91
	Foreman	4.67	2.00	2.18	3.74	4.02	1.83	1.96
	News	4.40	2.00	2.12	3.82	3.81	1.55	1.95
	Silent	4.66	2.00	2.08	4.10	4.05	1.80	1.95
	Average	**4.43**	**2.00**	**2.13**	**3.81**	**3.85**	**1.72**	**1.93**
CIF	*Mobile*	4.43	2.00	2.16	3.70	3.83	1.62	1.95
	Paris	4.33	2.00	2.13	3.85	3.80	1.66	1.94
	Stefan	4.57	2.00	2.12	3.48	3.86	1.83	1.96
	Tempete	4.37	2.00	2.17	3.83	3.70	1.41	1.95
	Average	**4.43**	**2.00**	**2.15**	**3.72**	**3.80**	**1.63**	**1.95**
Maximum Number		6	2	3	5	5	2	2

4.3 Summary

In this chapter, a fast algorithm for intra prediction "TraDED" is proposed. By calculating the edge intensities of transformed basis patterns, suitable mode candidates can be selected. With less than 0.05 dB of quality loss and 1.5 % of rate increment averagely, over 53 % of intra luma modes (4x4, 8x8, and 16x16) can be skipped during intra prediction.

Chapter 5 Hardware Design for SVC Intra Encoder

The implementation of an SVC intra encoder, with the adoption of previous analysis and fast algorithm, is introduced in this chapter. First of all, the analysis for hardware complexity will be taken, and parallelism and encoding schedule will also be studied. Then design modules in this work will be introduced according to coding flow. Finally, the implementation results and the comparison with previous work will be discussed.

5.1 System Analysis

The desired system specification is described as follows: an SVC encoder works under 135 MHz clock frequency with 3 quality layers, 3 spatial layers (CIF, SD 480p, and HD 1080p), and frame rate is set to 60 fps. To fulfill such a tough target like this, "frame-parallel" encoding is consequently needed. Since in an intra encoder, there is no data dependency among frames, which means several frames can be processed simultaneously. Thus, frame-parallel coding can be easily adopted in proposed design by doubling the hardware resource.

Another way to reduce design difficulty is pixel parallelism. In above assumption, the total number of processed macro-blocks per SOF is

396 + 1,350 + 8,160 = 9,906 (refer Fig. 17)

For 135 MHz working frequency, the available operation time per MB would be

*135M / ((60 / 2) * 9,906) = 454 cycles*

Then just considering prediction data under fast algorithm, the total pixel count per MB would be

*16 * 16 * (5 + 5 + 2) + 8 * 8 * 2 = 3,328*

Finally, the best parallelism can be calculated by

3,328 / 454 = 7.33 => 8 pixels (for double hardware resource)

To reduce design complexity, two equivalent components can be replaced by the two with different functions. Therefore, the parallelism of this work is set to 16 pixels / 8 pixels (16 pixels for some components and 8 pixels for others). This change decreases the hardware complexity of the system since functionalities of components can be integrated and specified.

Based on previous parallelism analysis, a two-stage (intra stage and quality enhancement stage) MB-based pipeline architecture is constructed, with three main phases: “Intra-T” (doubled hardware with 16-pixel throughput), “QQ-Rec” (single hardware with 8-pixel throughput), and “Quality Refine” (single hardware with 8-pixel throughput). The architecture is shown on Fig. 32. Note that the hardware in “QQ-Rec” phase can be shared by other two phases in order to increase utilization.

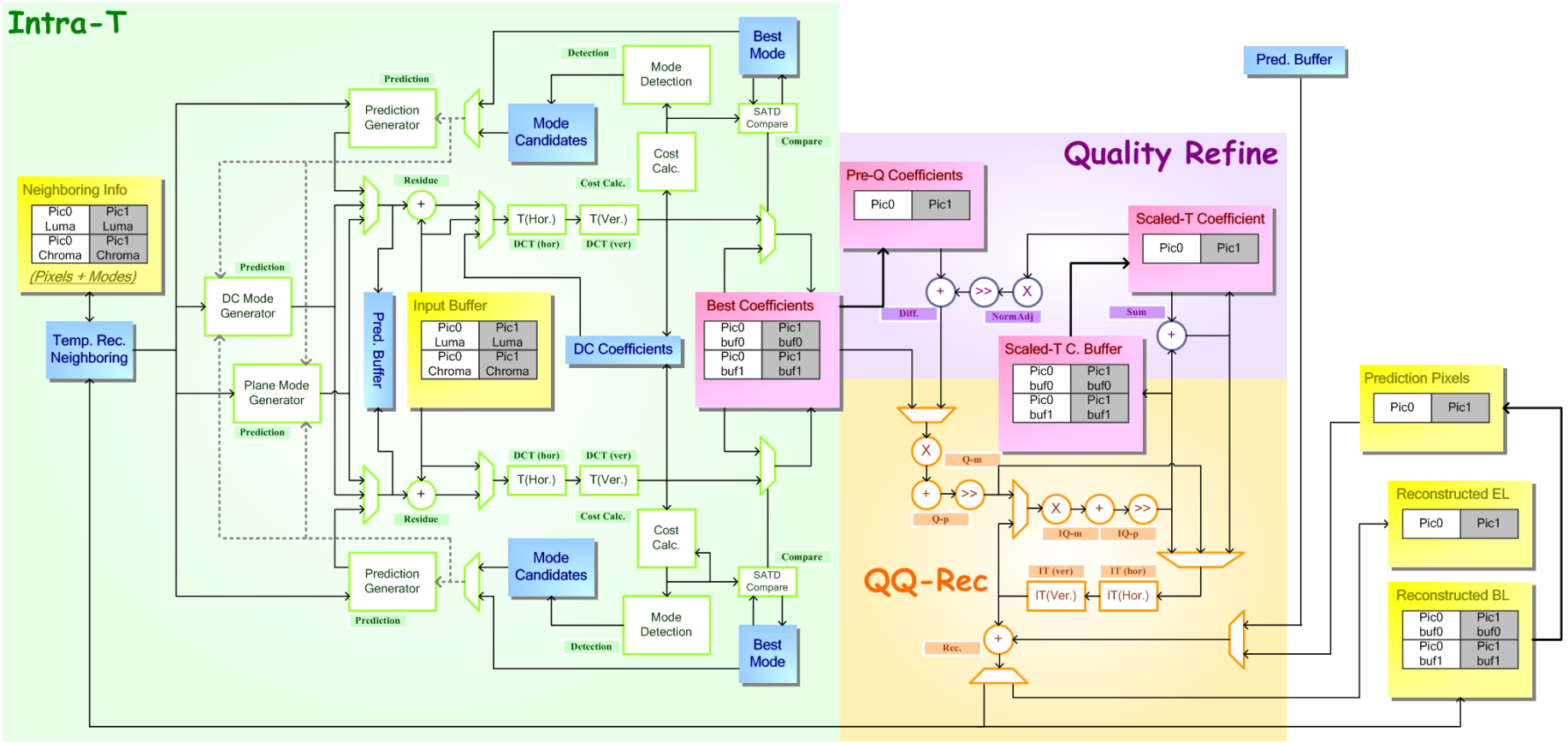


Fig. 32 Proposed architecture of an SVC encoder with fast algorithm

5.2 Encoding Schedule

From the result in previous section, the overall scheduling of this encoder is illustrated in Fig. 33. The available encoding cycle for a macro-block is set to 454. In intra 4x4 and 8x8 prediction modes, the processing of two macro-blocks with the same address is interlaced (refer Fig. 34 for detail) under frame-parallel coding scheme. In quality enhancement stage, two pictures are processed separately. Besides, for the future integration with inter and inter-layer prediction, the processing cycles is reserved in this encoding scheduling.

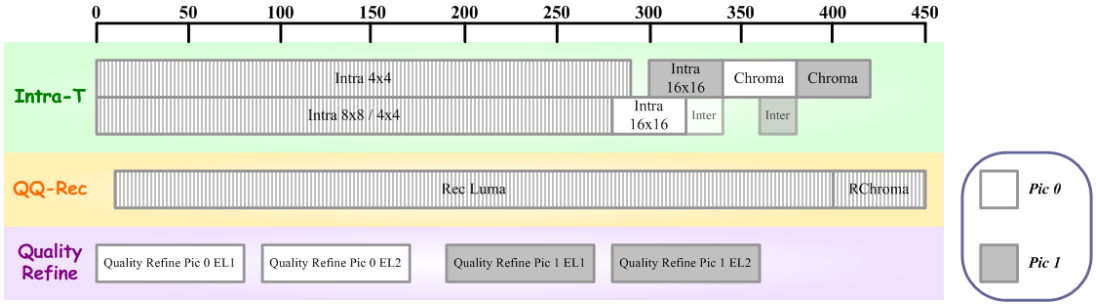


Fig. 33 Overall scheduling of proposed encoder

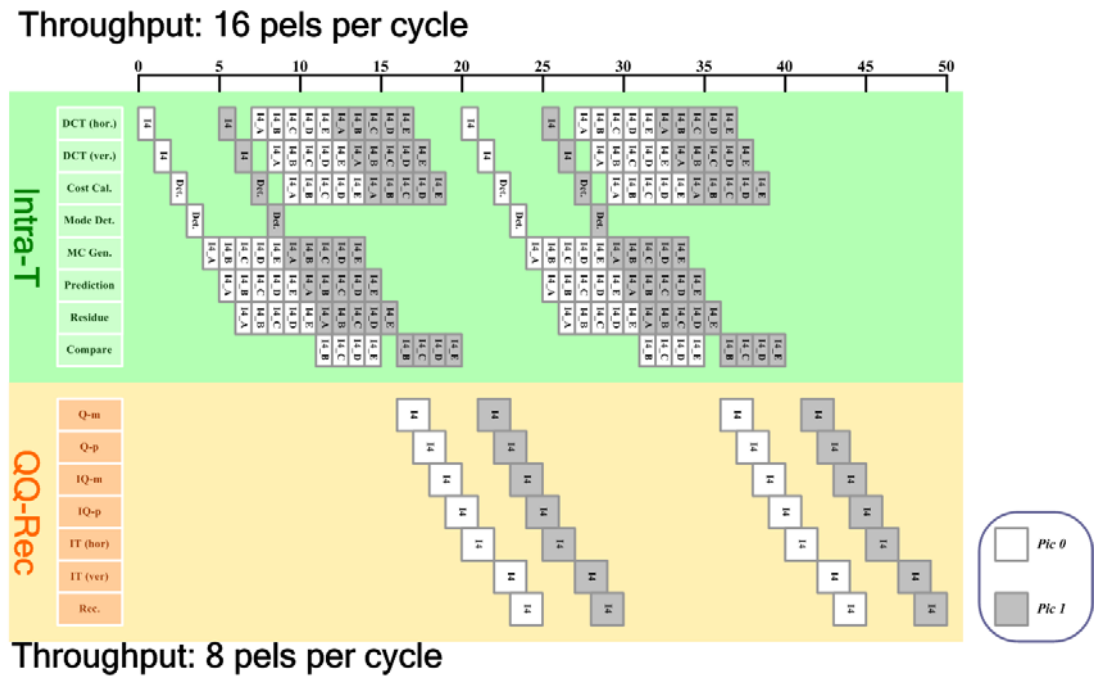


Fig. 34 Scheduling of intra 4x4 blocks

5.3 Specific Components

In the following, components in intra encoder will be introduced one after another according to the order of encoding flow.

5.3.1 Transform Modules

Since “TraDED” algorithm mentioned in Chapter 4 is adopted in proposed design, a transform module not only changes the domain of residues, but also plays a role in basis construction for mode candidate decision.

There are two transform modules in proposed encoder, one (Tran4DC) for 4x4 DCT and DHT processing, and another one (Tran48) deals with 4x4 and 8x8 DCT. In module “Tran4DC”, shown on Fig. 35, it contains eight 1-D 4x4 transform units, four for each direction (horizontal and vertical). In each transform unit, consider the similarity of the operations with 4x4 DCT, DHT is implemented by adding several multiplexers on traditional 4x4 DCT butterfly architecture [23], shown on Fig. 36.

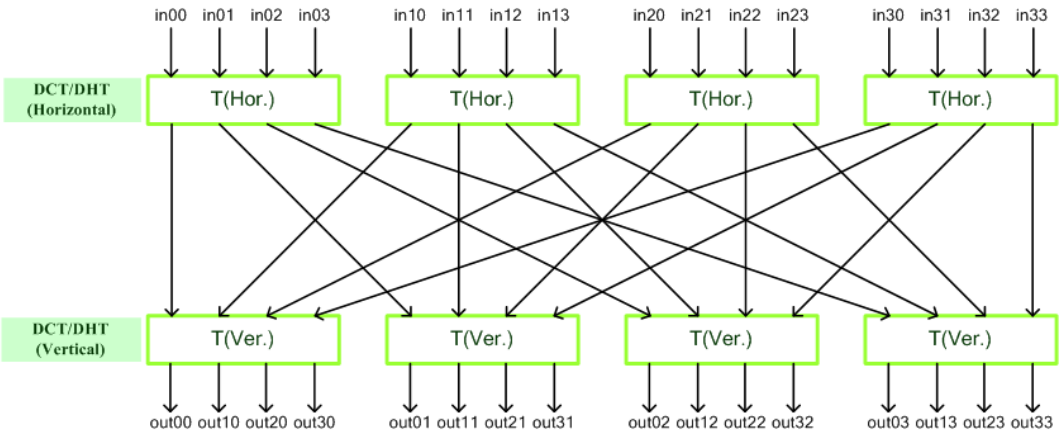


Fig. 35 Architecture of module “Tran4DC”

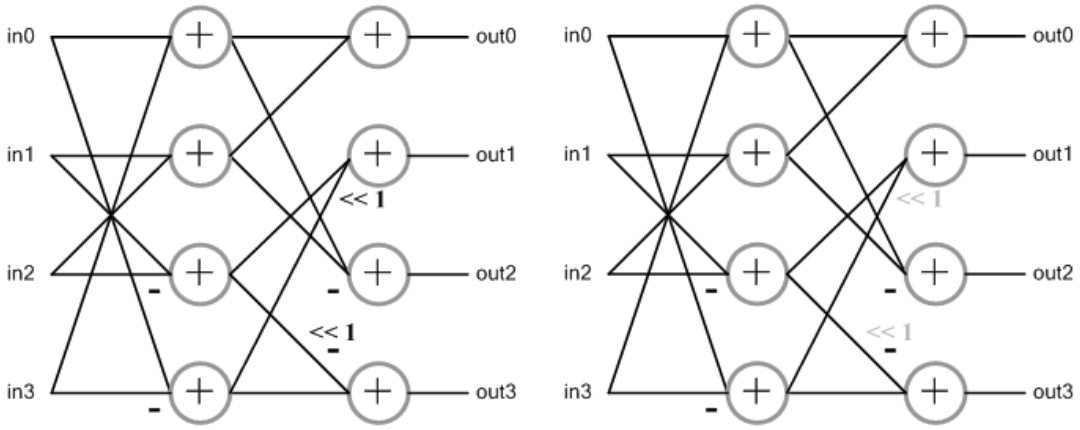


Fig. 36 Data path for 4x4 (a) DCT / (b) DHT

On the other hand, in module “Tran48”, shown on Fig. 37, four 1-D 8x8 transform units are included, two for horizontal and two for vertical operation, connected to an 8x8 block-size coefficient registers. Due to the consideration of encoding scheduling and hardware utilization, by using multiplexers, this transform unit can also support 1-D 4x4 transform for two rows, and the detail data path is shown on Fig. 38.

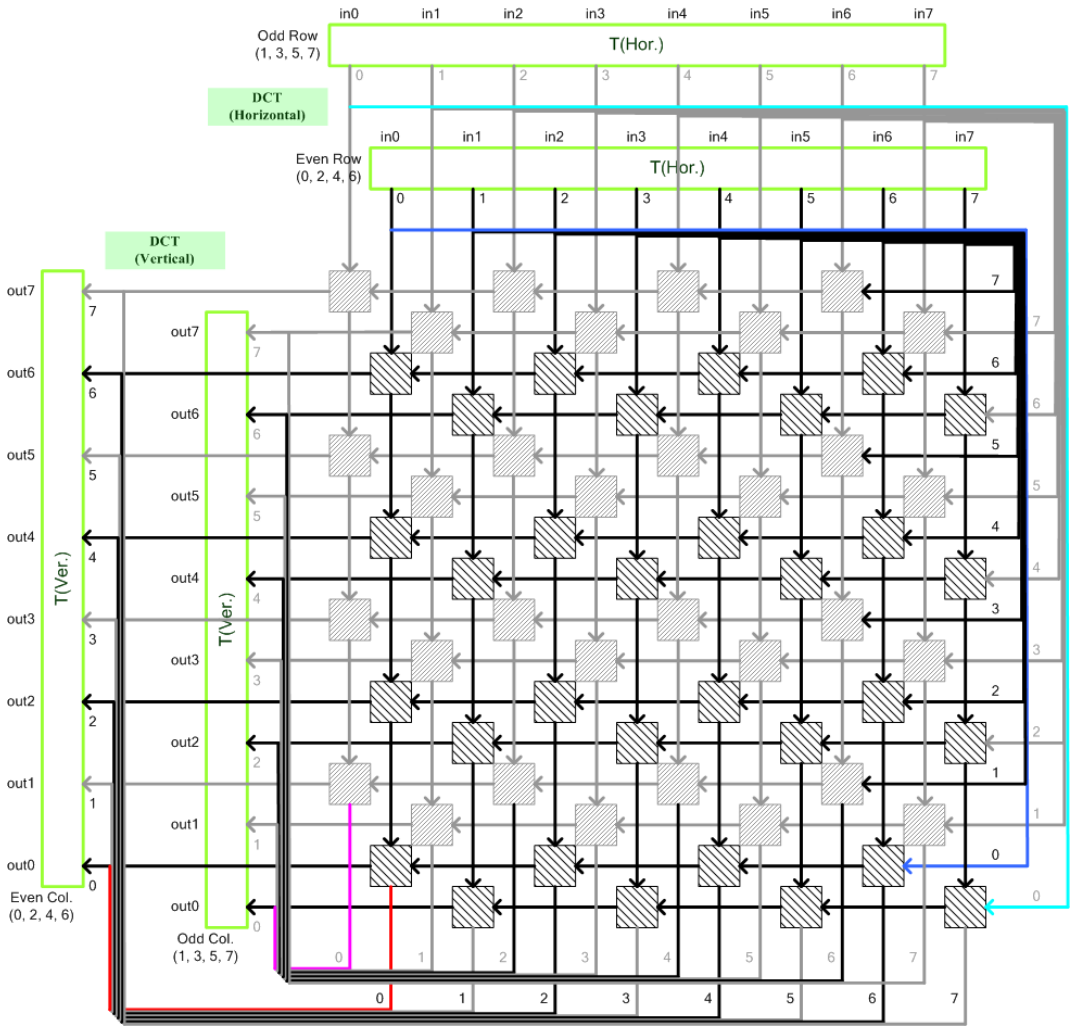
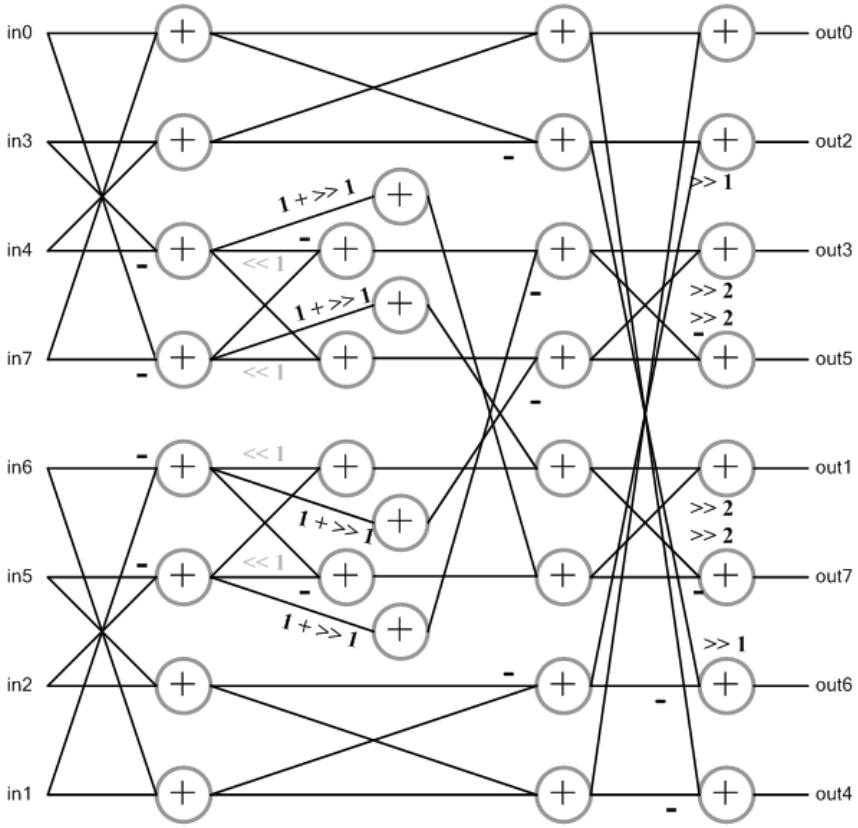
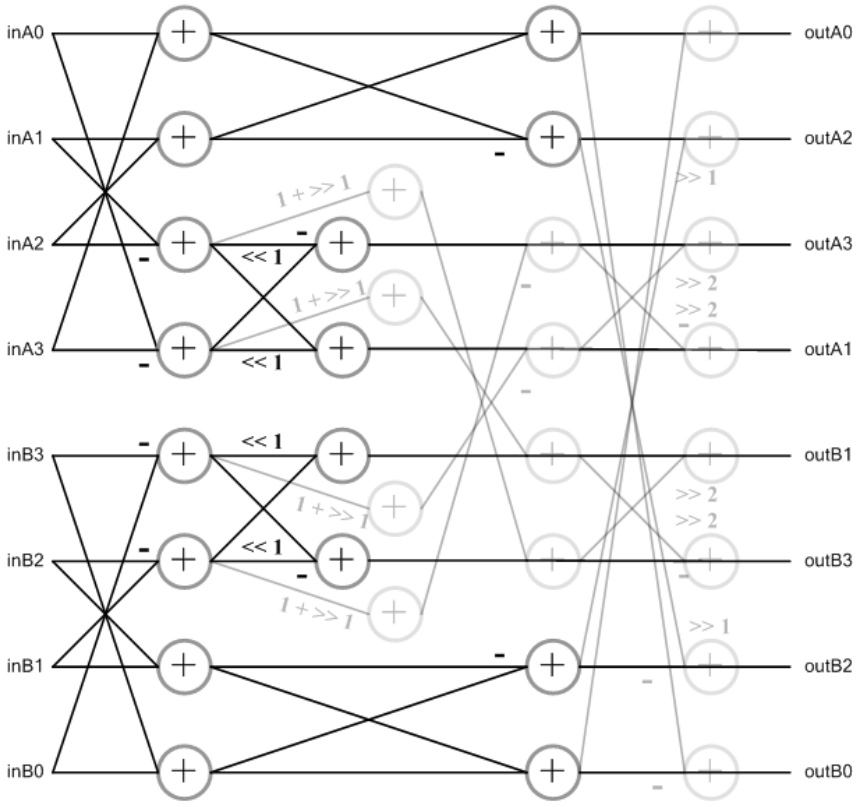


Fig. 37 Architecture of module “Tran48”



(a)



(b)

Fig. 38 Data path for (a) 8x8 / (b) 4x4 DCT

5.3.2 Cost Calculation Modules

Connected to transform modules, two cost calculation modules “CostCal416C” and “CostCal4816” receive the transformed data and produce the edge intensities and SATD cost. The former deals with 4x4, 16x16 or chroma block, and the architecture is shown on Fig. 39. Intensity I_{AC} shares the processing element with SATD calculation, and other intensities are obtained separately.

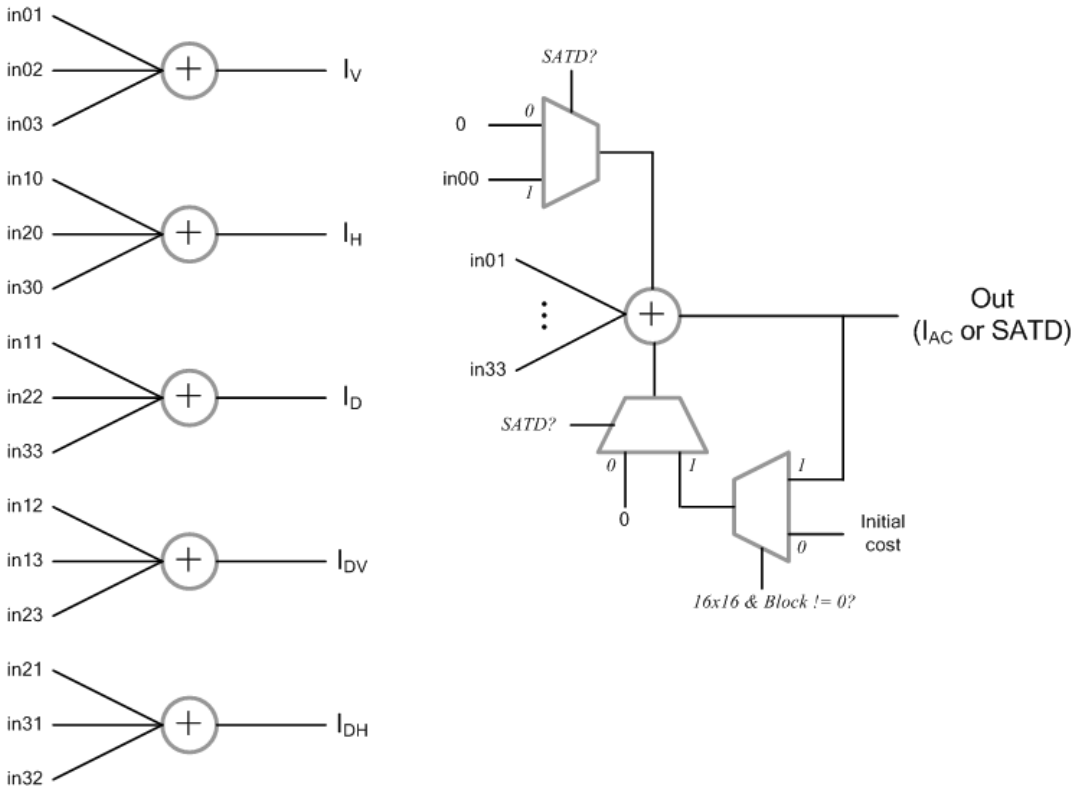


Fig. 39 Architecture of module “CostCal416C”

Another module “CostCal4816” (shown on Fig. 40) can process 4x4, 8x8, or 16x16 blocks, the processing unit for I_{AC} / SATD calculation is similar with the one in “CostCal416C”. For other intensities, the computational units are assigned by the case summarized in Table 11. Note that in the 8x8 mode, temporary intensity values would be stored for accumulation.

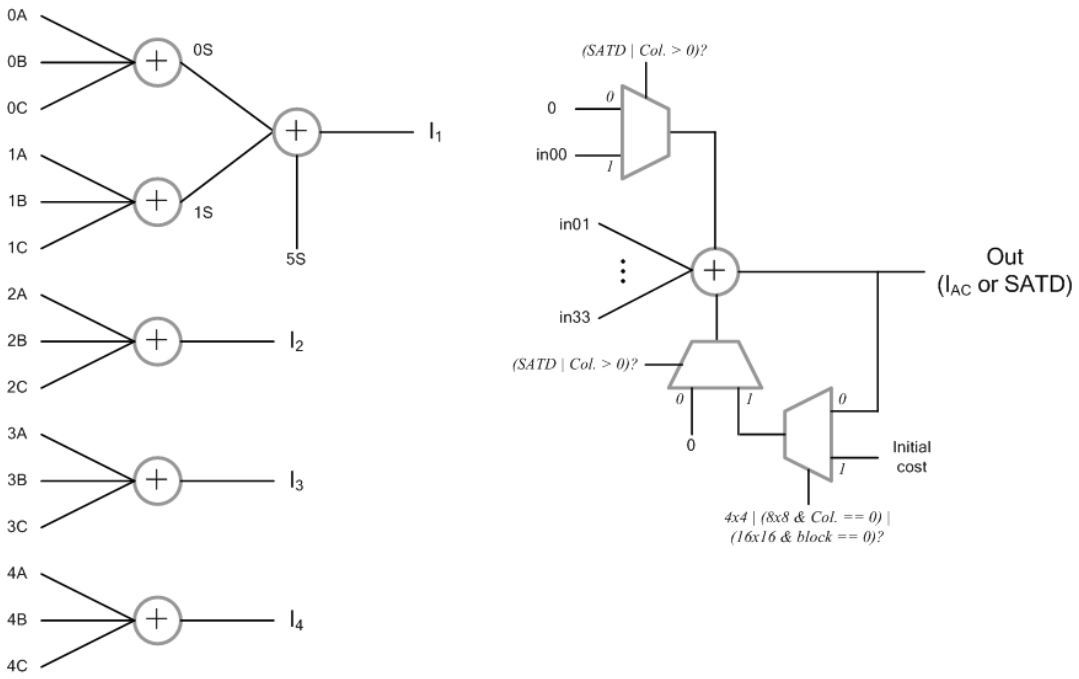


Fig. 40 Architecture of module "CostCal4816"

Table 11 Input assignment of computational units for intensity generation

Input / ***#CU***		***8x8***								***4x4***		***16x16***	
		Col 0, 1		***Col. 2, 3***		***Col. 4, 5***		***Col. 6, 7***					
5S		Z		I_{DH}		I_{DV}		Z		X		X	
0	***A***	1,0		3,2		1,4		Z		0,1		0,1	
	B	2,0		4,2		2,4		Z		0,2	I_V	0,2	I_V
	C	3,0	I_H	4,3	I_{DH}	3,4	I_{DV}	Z		0,3		0,3	
1	***A***	4,0		Z		Z		Z		1,0		1,0	
	B	5,0		Z		Z		Z		2,0	I_H	2,0	I_H
	C	Z		Z		Z		Z		3,0		3,0	
2	***A***	0,1		0,2		0,4		Z		1,1		1,1	
	B	Z	I_V	0,3	I_V	0,5	I_V	Z		2,2	I_D	2,2	I_D
	C	Z		I_V		I_V		Z		3,3		3,3	
3	***A***	1,1		2,2		4,4		Z		1,2		X	
	B	Z	I_D	3,3	I_D	5,5	I_D	Z		1,3	I_{DV}	X	
	C	Z		I_D		I_D		Z		2,3		X	
4	***A***	2,1		1,2		Z		Z		2,1		X	
	B	3,1	I_{DH}	1,3	I_{DV}	Z		Z		3,1	I_{DH}	X	
	C	4,1		2,3		Z		Z		3,2		X	

i, j *= the {vertical, horizontal} position in transformed data;* ***Z*** *= zero;* ***X*** *= don't care*

5.3.3 Mode Decision Modules

Two mode decision modules read intensities information from cost calculation modules. The concept of fast intra prediction algorithm “TraDED”, mentioned in Chapter 4 (please see Fig. 21, Fig. 23, Fig. 25, and Fig. 27 for the detail.), is implemented in these two modules. Module ”MoDe416C” processes intra 4x4, intra16x16 and intra chroma prediction, module “MoDe48” deals with intra 4x4 and intra 8x8 prediction.

The output length of a mode decision module is 9 bits, where each of them represents the enable signal for the related mode. This 9-bit length data will be decoded by mode candidate generator and output one mode candidate for each cycle. One advantage of this design is that mode enable signals can be easily combined with other control signals by a simply logic operation (“OR” or “AND”).

5.3.4 Intra Prediction Generator Modules

There are four modules to generate intra prediction info: one for DC mode (“PG_DC”), one for plane (“PG_Plane”) mode, and two for other modes (“PG416C” and “PG4816”). Since it is not resource-efficient to complete DC or plane mode prediction within one cycle, these modes need a specific computational unit for related processing.

In module “PG416C”, shown on Fig. 41, intra 4x4, intra 16x16 and intra chroma modes are generated. Seven T-type adders (3-input with 1-bit 1 increment, Fig. 42(a)) and three S-type (2-input with 1-bit 1 increment, Fig. 42(b)) adders are contained. By the neighboring position provided in Fig. 43, the input assignment for each adder is organized in Table 12, and the output selection rule is summarized in Table 13.

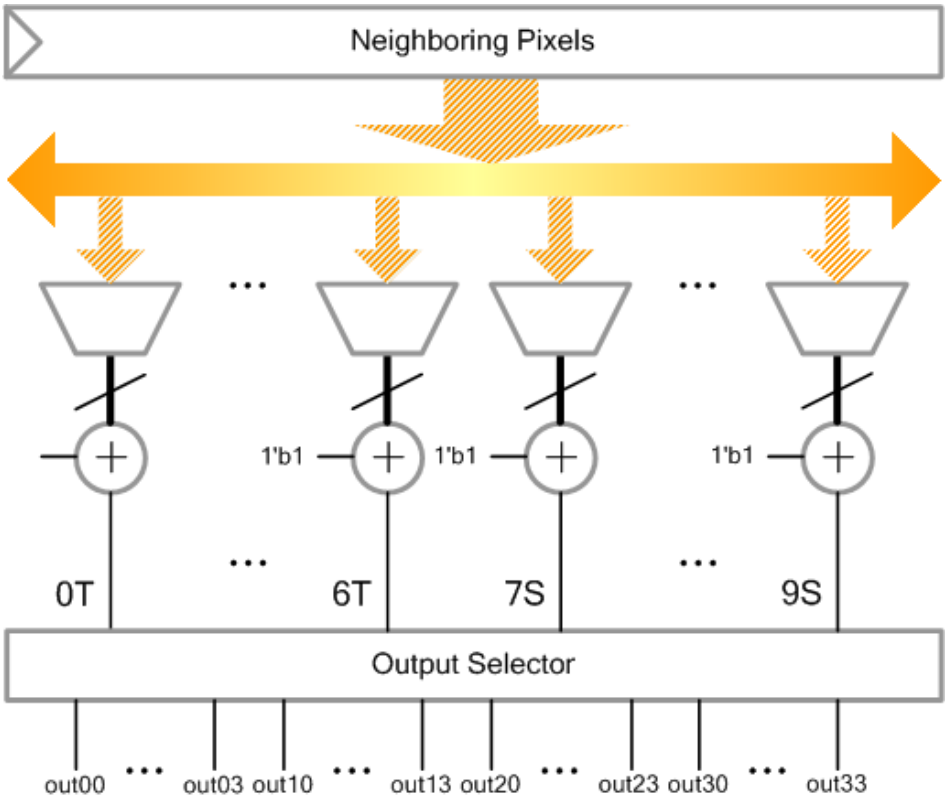
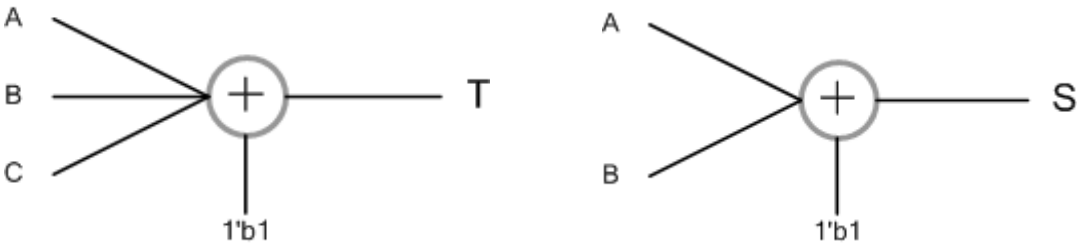


Fig. 41 Architecture of module “PG416C”



(a) (b)

Fig. 42 (a) T-type adder; (b) S-type Adder

Fig. 43 Neighboring position in module “PG416C”

Table 12 Input assignment for each adder in module “PG416C”

Mode / #CU		3	4	5	6	7	8
0	A	A	A	A	A	A	K
	B	C	C	C	C	C	L
	C	{B, 1’b1}	{B, 1’b1}	{B, 1’b1}	{B, 1’b1}	{B, 1’b1}	{L, 1’b1}
1	A	B	B	B	O	B	
	B	D	D	D	A	D	
	C	{C, 1’b1}	{C, 1’b1}	{C, 1’b1}	9’b0	{C, 1’b1}	
2	A	C	O	O	O	C	
	B	E	B	B	B	E	
	C	{D, 1’b1}	{A, 1’b1}	{A, 1’b1}	{A, 1’b1}	{D, 1’b1}	
3	A	D	I	I	I	D	
	B	F	A	A	A	F	
	C	{E, 1’b1}	{O, 1’b1}	{O, 1’b1}	{O, 1’b1}	{E, 1’b1}	
4	A	E	O	O	O	E	
	B	G	J	J	J	G	
	C	{F, 1’b1}	{I, 1’b1}	{I, 1’b1}	{I, 1’b1}	{F, 1’b1}	
5	A	F	I	I	I	F	I
	B	H	K	K	K	E	K
	C	{G, 1’b1}	{J, 1’b1}	{J, 1’b1}	{J, 1’b1}	9’b0	{J, 1’b1}
6	A	G	J	A	J	A	J
	B	H	L	B	L	B	L
	C	{H, 1’b1}	{K, 1’b1}	9’b0	{K, 1’b1}	9’b0	{K, 1’b1}
7	A			O	K	D	K
	B			A	L	E	L
8	A			B	I	B	I
	B			C	J	C	J
9	A			C	J	C	J
	B			D	K	D	K

Note: vertical and horizontal modes are bypassed, and DC mode is not processed by this component.

Table 13 Output selection rule for module ”PG416C”

Row / Mode	0	1	2	3
3	{0, 1, 2, 3}	{1, 2, 3, 4}	{2, 3, 4, 5}	{3, 4, 5, 6}
4	{3, 2, 0, 1}	{4, 3, 2, 0}	{5, 4, 3, 2}	{6, 5, 4, 3}
5	{7, 6, 8, 9}	{3, 2, 0, 1}	{4, 7, 6, 8}	{5, 3, 2, 0}
6	{1, 3, 2, 0}	{8, 4, 1, 3}	{9, 5, 8, 4}	{7, 6, 9, 5}
7	{6, 8, 9, 7}	{0, 1, 2, 3}	{8, 9, 7, 5}	{1, 2, 3, 4}
8	{8, 5, 9, 6}	{9, 6, 7, 0}	{7, 0, ***L, L***}	{***L, L, L, L***}

Note: the number means the output of adder, with 2-LSB truncation

***L**: neighboring position in Fig. 43*

N: truncate only 1 LSB

More complicated architecture is in module “PD4816”, shown on Fig. 44. Composed of 12 T-type (from *0T* to *7T* and from *bT* to *fT*) adders and 3 S-type (*8S, 9S,* and *aS*) adders, it not only supports 4x4, 8x8, and intra 16x16 prediction data generating, but also deals with neighboring data pre-filtering in intra 8x8 prediction. For this module, according to the neighboring position in Fig. 45, input assignment is listed in Table 14, and output selection is summarized in Table 15 and Table 16.

To reduce the complexity of multiplexers in prediction generator, the input assignments of module “PG416C” and “PG4816” are designed based on regarding the common points among modes and rows. Furthermore, the total number of case for an adder is balanced to reduce the length of critical path.

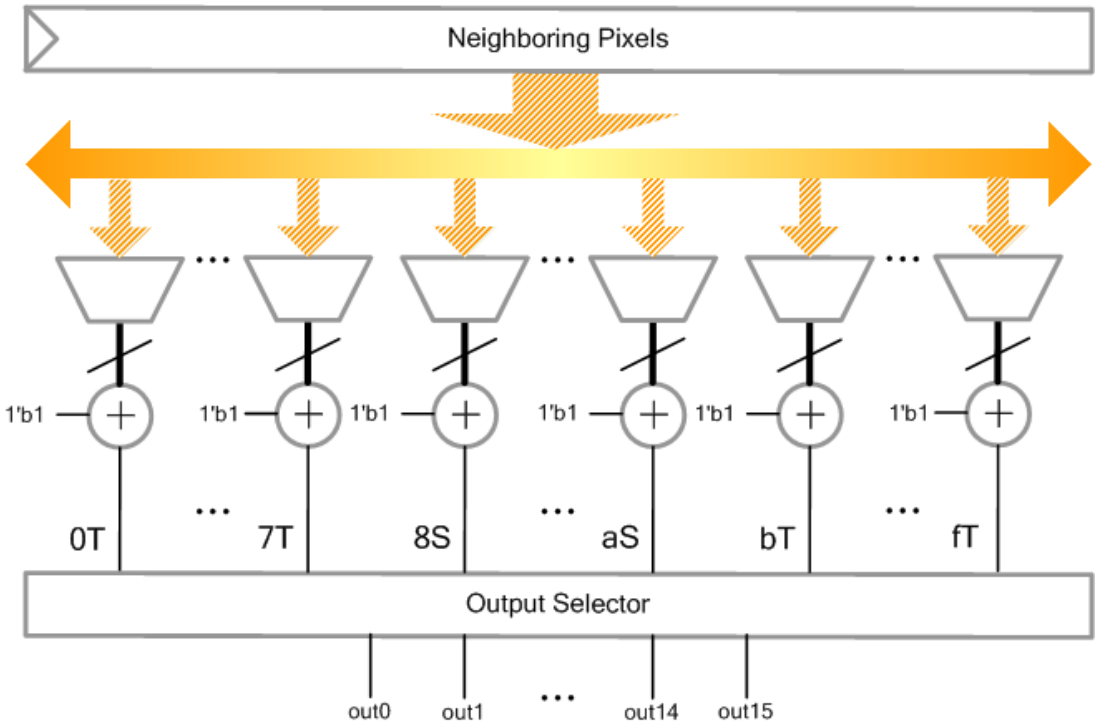


Fig. 44 Architecture of module “PG4816”

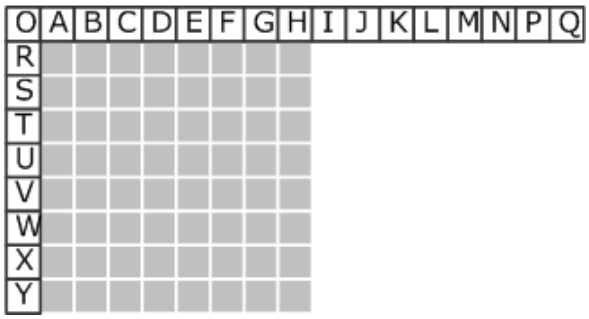


Fig. 45 Neighboring position in module “PG4816”

Table 14 Input assignment for each adder in module “PG4816”

Mode \ #CU		0			1			2			3		
		A	***B***	***C****	***A***	***B***	***C****	***A***	***B***	***C****	***A***	***B***	***C****
4x4	***3***	A	C	B	B	D	C	C	E	D	D	F	E
	4	A	C	B	B	D	C						
	5	A	C	B	B	D	C						
	6	A	C	B	T	U	0						
	7	A	C	B	B	D	C	C	E	D	D	F	E
	8				T	U	0				T	U	U
8x8	***F_0***	A	C	B	B	D	C	C	E	D	V	X	W
	F_1	I	K	J	I	L	K	L	N	M	D	F	E
	3_0	A	C	B	B	D	C	C	E	D	D	F	E
	3_1				I	L	K	C	E	D	D	F	E
	3_2				I	L	K	L	N	M			
	3_3				I	L	K	L	N	M			
	4_0	A	C	B	B	D	C	C	E	D	D	F	E
	4_1	A	C	B	B	D	C	C	E	D	D	F	E
	4_2	A	C	B	B	D	C						
	4_3										V	X	W
	5_0	A	C	B	B	D	C	C	E	D	D	F	E
	5_1	A	C	B	B	D	C	C	E	D	D	F	E
	5_2	A	C	B	B	D	C	C	E	D	D	F	E
	5_3	A	C	B	B	D	C	C	E	D	V	X	W
	6_0	A	C	B	B	D	C	C	E	D	D	F	E
	6_1	A	C	B	T	U	0						
	6_2	V	W	0	T	U	0						
	6_3	V	W	0	T	U	0				V	X	W
	7_0	A	C	B	B	D	C	C	E	D	D	F	E
	7_1	I	K	J	B	D	C	C	E	D	D	F	E
	7_2	I	K	J	I	L	K	C	E	D	D	F	E
	7_3	I	K	J	I	L	K	K	L	0	D	F	E
	8_0	V	W	0	T	U	0				V	X	W
	8_1	V	W	0	T	U	0				V	X	W
	8_2	V	W	0							V	X	W
	8_3												

**C = {n, 1’b1} for non-zero assignment*

Table 14 (Cont.)

Mode \ #CU		4			5			6			7		
		A	***B***	***C****	***A***	***B***	***C****	***A***	***B***	***C****	***A***	***B***	***C****
4x4	***3***	E	G	F	F	H	G	G	H	H			
	4				R	T	S	O	B	A	R	A	O
	5				R	T	S	O	B	A	R	A	O
	6				R	T	S	O	B	A	R	A	O
	7	E	G	F									
	8				R	T	S						
8x8	***F_0***	T	V	U	R	T	S	O	B	A	R	A	O
	F_1	E	G	F	F	H	G	G	I	H	H	J	I
	3_0	E	G	F	F	H	G	G	I	H	H	J	I
	3_1	E	G	F	F	H	G	G	I	H	H	J	I
	3_2	E	G	F	F	H	G	G	I	H	H	J	I
	3_3							G	I	H	H	J	I
	4_0	E	G	F	F	H	G	O	B	A	R	A	O
	4_1				R	T	S	O	B	A	R	A	O
	4_2	T	V	U	R	T	S	O	B	A	R	A	O
	4_3	T	V	U	R	T	S	O	B	A	R	A	O
	5_0	E	G	F	F	H	G	O	B	A	R	A	O
	5_1	E	G	F	R	T	S	O	B	A	R	A	O
	5_2	T	V	U	R	T	S	O	B	A	R	A	O
	5_3	T	V	U	R	T	S	O	B	A	R	A	O
	6_0	E	G	F				O	B	A	R	A	O
	6_1				R	T	S	O	B	A	R	A	O
	6_2	T	V	U	R	T	S						
	6_3	T	V	U									
	7_0	E	G	F	F	H	G	G	I	H	H	J	I
	7_1	E	G	F	F	H	G	G	I	H	H	J	I
	7_2	E	G	F	F	H	G	G	I	H	H	J	I
	7_3	E	G	F	F	H	G	G	I	H	H	J	I
	8_0	T	V	U	R	T	S						
	8_1	T	V	U									
	8_2												
	8_3												

**C = {n, 1'b1} for non-zero assignment*

Table 14 (Cont.)

Mode \ #CU		8			9			a			b		
		A	B	C	A	B	C	A	B	C	A	B	C*
4x4	3												
	4												
	5	Z	A		A	B		B	C		C	D	0
	6	S	T		O	R		R	S				
	7				A	B		B	C		C	D	0
	8	S	T					R	S				
8x8	F_0										X	Y	Y
	F_1										K	M	L
	3_0												
	3_1										K	M	L
	3_2										K	M	L
	3_3										K	M	L
	4_0												
	4_1												
	4_2												
	4_3												
	5_0	Z	A		A	B		B	C		C	D	0
	5_1	Z	A		A	B		B	C		C	D	0
	5_2	Z	A		A	B		B	C		C	D	0
	5_3	Z	A		A	B		B	C		C	D	0
	6_0				O	R		R	S				
	6_1	S	T		O	R		R	S				
	6_2	S	T		U	V		R	S				
	6_3	X	Y		U	V		W	X				
	7_0	H	I		A	B		B	C		C	D	0
	7_1	H	I		I	J		B	C		C	D	0
	7_2	H	I		I	J		J	K		C	D	0
	7_3	H	I		I	J		J	K		K	M	L
	8_0	S	T		U	V		R	S				
	8_1	X	Y		U	V		W	X		X	Y	Y
	8_2	X	Y					W	X		X	Y	Y
	8_3	X	Y								X	Y	Y

**C = {n, 1'b1} for non-zero assignment*

Table 14 (Cont.)

Mode \ #CU		c			d			e			f		
		A	B	C*	A	B	C*	A	B	C*	A	B	C*
4x4	3												
	4							S	U	T	O	S	R
	5										O	S	R
	6							S	U	T	O	S	R
	7	D	E	0	E	F	0						
	8							S	U	T			
8x8	F_0	W	Y	X	U	W	V	S	U	T	O	S	R
	F_1	P	Q	Q	N	Q	P	M	P	N			
	3_0										I	K	J
	3_1										I	K	J
	3_2							M	P	N	I	K	J
	3_3	P	Q	Q	N	Q	P	M	P	N	I	K	J
	4_0										O	S	R
	4_1							S	U	T	O	S	R
	4_2				U	W	V	S	U	T	O	S	R
	4_3	W	Y	X	U	W	V	S	U	T	O	S	R
	5_0	D	E	0	E	F	0	F	G	0	G	H	0
	5_1	D	E	0	E	F	0	F	G	0	O	S	R
	5_2	D	E	0	E	F	0	S	U	T	O	S	R
	5_3	D	E	0	U	W	V	S	U	T	O	S	R
	6_0										O	S	R
	6_1							S	U	T	O	S	R
	6_2				U	W	V	S	U	T	O	S	R
	6_3	W	Y	X	U	W	V	S	U	T			
	7_0	D	E	0	E	F	0	F	G	0	G	H	0
	7_1	D	E	0	E	F	0	F	G	0	G	H	0
	7_2	D	E	0	E	F	0	F	G	0	G	H	0
	7_3	D	E	0	E	F	0	F	G	0	G	H	0
	8_0				U	W	V	S	U	T			
	8_1	W	Y	X	U	W	V						
	8_2	W	Y	X									
	8_3												

**C = {n, 1'b1} for non-zero assignment*

Table 15 Output selection rule of intra 8x8 mode for module ”PG4816”

Row / Mode	Even (0, 2, 4, 6)	Odd (1, 3, 5, 7)
F_0	{f, 5, e, 4, d, 3, c, b}	{7, 6, 0, 1, 2}
F_1	{3, 4, 5, 6}	{7, 0, 1, b, 2, e, d, c}
3_0	{0, 1, 2, 3, 4, 5, 6, 7}	{1, 2, 3, 4, 5, 6, 7, f}
3_1	{2, 3, 4, 5, 6, 7, f, 1}	{3, 4, 5, 6, 7, f, 1, b}
3_2	{4, 5, 6, 7, f, 1, b, 2}	{5, 6, 7, f, 1, b, 2, e}
3_3	{6, 7, f, 1, b, 2, e, d}	{7, f, 1, b, 2, e, d, c}
4_0	{7, 6, 0, 1, 2, 3, 4, 5}	{f, 7, 6, 0, 1, 2, 3, 4}
4_1	{5, f, 7, 6, 0, 1, 2, 3}	{e, 5, f, 7, 6, 0, 1, 2}
4_2	{4, e, 5, f, 7, 6, 0, 1}	{d, 4, e, 5, f, 7, 6, 0}
4_3	{3, d, 4, e, 5, f, 7, 6}	{c, 3, d, 4, e, 5, f, 7}
5_0	{8, 9, a, b, c, d, e, f}	{7, 6, 0, 1, 2, 3, 4, 5}
5_1	{f, 8, 9, a, b, c, d, e}	{5, 7, 6, 0, 1, 2, 3, 4}
5_2	{e, f, 8, 9, a, b, c, d}	{4, 5, 7, 6, 0, 1, 2, 3}
5_3	{d, e, f, 8, 9, a, b, c}	{3, 4, 5, 7, 6, 0, 1, 2}
6_0	{9, 7, 6, 0, 1, 2, 3, 4}	{a, f, 9, 7, 6, 0, 1, 2}
6_1	{8, 5, a, f, 9, 7, 6, 0}	{1, e, 8, 5, a, f, 9, 7}
6_2	{9, 4, 1, e, 8, 5, a, f}	{0, d, 9, 4, 1, e, 8, 5}
6_3	{a, 3, 0, d, 9, 4, 1, e}	{8, c, a, 3, 0, d, 9, 4}
7_0	{9, a, b, c, d, e, f, 8}	{0, 1, 2, 3, 4, 5, 6, 7}
7_1	{a, b, c, d, e, f, 8, 9}	{1, 2, 3, 4, 5, 6, 7, 0}
7_2	{b, c, d, e, f, 8, 9, a}	{2, 3, 4, 5, 6, 7, 0, 1}
7_3	{c, d, e, f, 8, 9, a, 2}	{3, 4, 5, 6, 7, 0, 1, b}
8_0	{a, 5, 8, e, 1, 4, 9, d}	{8, e, 1, 4, 9, d, 0, 3}
8_1	{1, 4, 9, d, 0, 3, a, c}	{9, d, 0, 3, a, c, 8, b}
8_2	{0, 3, a, c, 8, b, ***Y, Y***}	{a, c, 8, b, ***Y, Y, Y, Y***}
8_3	{8, b, ***Y, Y, Y, Y, Y, Y***}	{***Y, Y, Y, Y, Y, Y, Y, Y***}

Note: the number means the output of adder, with 2-LSB truncation

Y*: neighboring position in Fig. 45*

N: truncate only 1 LSB

Table 16 Output selection rule of intra 4x4 mode for module ”PG4816”

Mode \ Row	0	1	2	3
3	{0, 1, 2, 3}	{1, 2, 3, 4}	{2, 3, 4, 5}	{3, 4, 5, 6}
4	{7, 6, 0, 1}	{f, 7, 6, 0}	{5, f, 7, 6}	{e, 5, f, 7}
5	{8, 9, a, b}	{7, 6, 0, 1}	{f, 8, 9, a}	{5, 7, 6, 0}
6	{9, 7, 6, 0}	{a, f, 9, 7}	{8, 5, a, f}	{1, e, 8, 5 }
7	{9, a, b, c}	{0, 1, 2, 3}	{a, b, c, d}	{1, 2, 3, 4}
8	{a, 5, 8, e}	{8, e, 1, 3}	{1, 3, ***U, U***}	{***U, U, U, U***}

Note: the number means the output of adder, with 2-LSB truncation

***U**: neighboring position in Fig. 45*

N: truncate only 1 LSB

For DC mode, module “PG_DC” calculates the mean values of neighboring pixels. Considering the utilization of DC mode is not frequent (about 20%), it is appropriate to integrate the DC calculation of different block size. The architecture of this module is shown on Fig. 46. Note that *s0* is the DC value of input pixels.

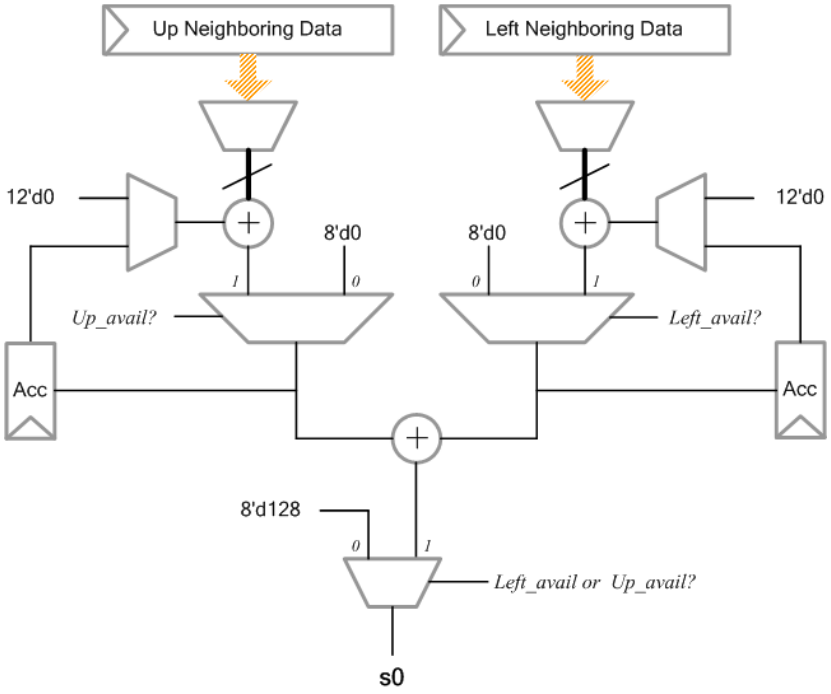
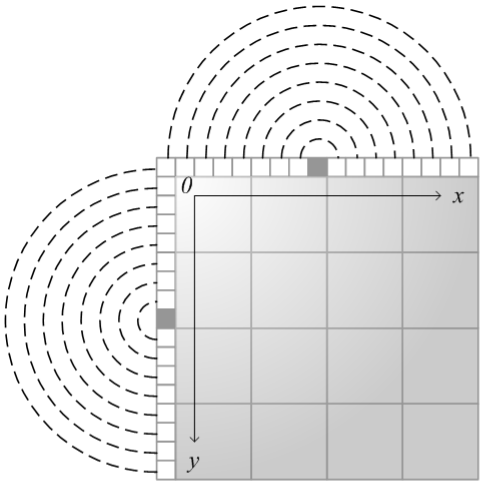


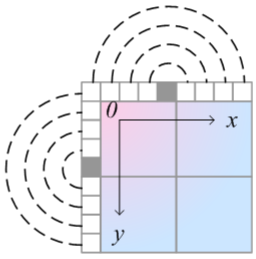
Fig. 46 Architecture of module “PG_DC”

Similar to DC mode, the calculation of plane mode is also complex, and rarely used, it is better to implement it with only one specific processing unit to satisfy all requirements of encoding. To consider the operation of plane mode shown on Fig. 47, the parameters H, V, α, β, and γ can be calculated before prediction processing (by "PG_pls"). When calling plane module "PG_plane" for data, the prediction values would be generated by these pre-calculated parameters.



$H = \Sigma\ x * [p(7 + i, -1) - p(7 - i, -1)]$, for $i = 1 \sim 8$
$V = \Sigma\ y * [p(-1, 7 + i) - p(-1, 7 - i)]$, for $i = 1 \sim 8$
$\alpha = 16 * [p(-1, 15) - p(15, -1)]$
$\beta = (5 * H + 32) >> 6$
$\delta = (5 * V + 32) >> 6$
$Pred(x, y) = [\alpha + \beta * (x - 7) + \delta * (y - 7) + 16] >> 5$

(a)



$H = \Sigma\ x * [p(4 + i, -1) - p(4 - i, -1)]$, for $i = 1 \sim 4$
$V = \Sigma\ y * [p(-1,4 + i) - p(-1, 4 - i)]$, for $i = 1 \sim 4$
$\alpha = 16 * [p(-1, 7) - p(7, -1)]$
$\beta = (17 * H + 16) >> 5$
$\delta = (17 * V + 16) >> 5$
$Pred(x, y) = [\alpha + \beta * (x - 3) + \delta * (y - 3) + 16] >> 5$

(b)

Fig. 47 Intra plane mode prediction: (a) intra 16x16; (b) chroma

5.3.5 Quantization / Inverse Quantization Module

In forward and inverse quantization, in order to save resource, the parallelism is set to 8-pixel instead of 16-pixel. The architecture of a quantization unit (note: inverse quantization has the same structure) is shown on Fig. 48. QP-dependent multipliers are implemented by look-up tables listed in Table 17, Table 18 (for forward), Table 19, and Table 20 (for inverse quantization).

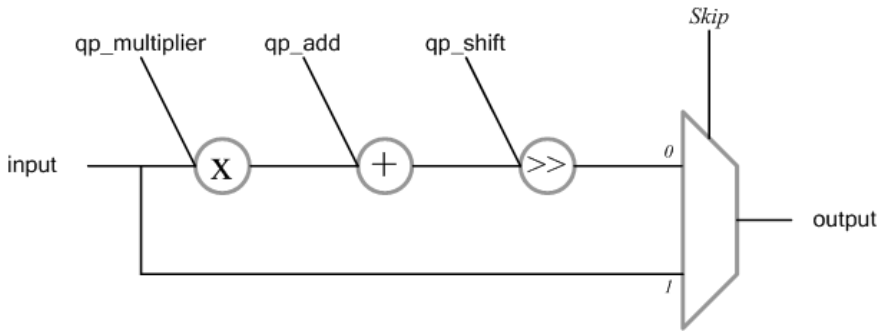


Fig. 48 Architecture of a quantization unit

Table 17 Quantization factors for a 4x4 block

Point* / ***QP % 6***	(0, 0), (0, 2), (2, 0), (2, 2)	(1, 1), (1, 3), (3, 1), (3, 3)	Other
0	13107	5243	8066
1	11916	4660	7490
2	10082	4194	6554
3	9362	3647	5825
4	8192	3355	5243
5	7282	2893	4559

**: position (i, j) indicates {vertical, horizontal) location of a 4x4 block*

Table 18 Quantization factors for an 8x8 block

Point* / ***QP % 6***	i % 4 = 0 j % 4 = 0	i % 2 = 1 j % 2 = 1	i % 4 = 2 j % 4 = 2	(i % 2 , j % 2) = (0,1) or (1, 0)	(i % 4, j % 4) = (0, 2) or (2, 0)	Other
0	13107	11428	20972	12222	16777	15481
1	11916	10826	19174	11058	14980	14290
2	10082	8943	15978	9675	12710	11985
3	9362	8228	14913	8931	11984	11259
4	8192	7346	13159	7740	10486	9777
5	7282	6428	11570	6830	9118	8640

**: position (i, j) indicates {vertical, horizontal) location of an 8x8 block*

Table 19 Inverse quantization factors for a 4x4 block

Point* / ***QP % 6***	(0, 0), (0, 2), (2, 0), (2, 2)	(1, 1), (1, 3), (3, 1), (3, 3)	Other
0	10	16	13
1	11	18	14
2	13	20	16
3	14	23	18
4	16	25	20
5	18	29	23

**: position (i, j) indicates {vertical, horizontal) location of a 4x4 block*

Table 20 Inverse quantization factors for an 8x8 block

Point* / ***QP % 6***	i % 4 = 0 j % 4 = 0	i % 2 = 1 j % 2 = 1	i % 4 = 2 j % 4 = 2	(i % 2 , j % 2) = (0,1) or (1, 0)	(i % 4, j % 4) = (0, 2) or (2, 0)	Other
0	20	18	32	19	25	24
1	22	19	35	21	28	26
2	26	23	42	24	33	31
3	28	25	45	26	35	33
4	32	28	51	30	40	38
5	36	32	58	34	46	43

**: position (i, j) indicates {vertical, horizontal) location of an 8x8 block*

5.3.6 Reconstruction Path

The reconstruction path includes inverse transform module and a set of adders. The inverse transform module “ITran”, shown on Fig. 49, the integration of 2 transform units and transposition registers supports all kinds of inverse transform with 8-pixel throughput. In a transform unit, the detail data paths for inverse transform of 8x8, 4x4 and Hadamard transform are drew in Fig. 50, Fig. 51, and Fig. 52 respectively.

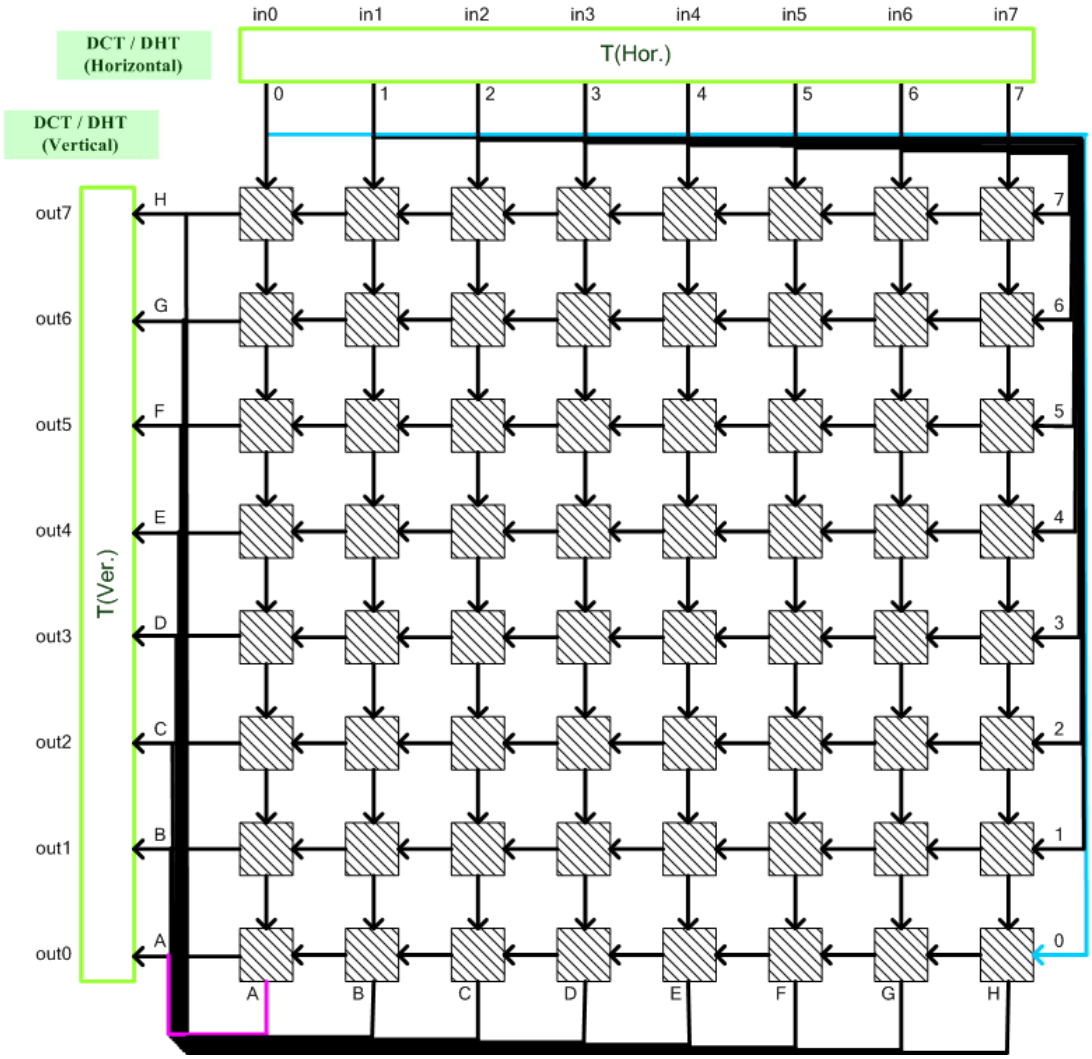


Fig. 49 Architecture of module “ITran”

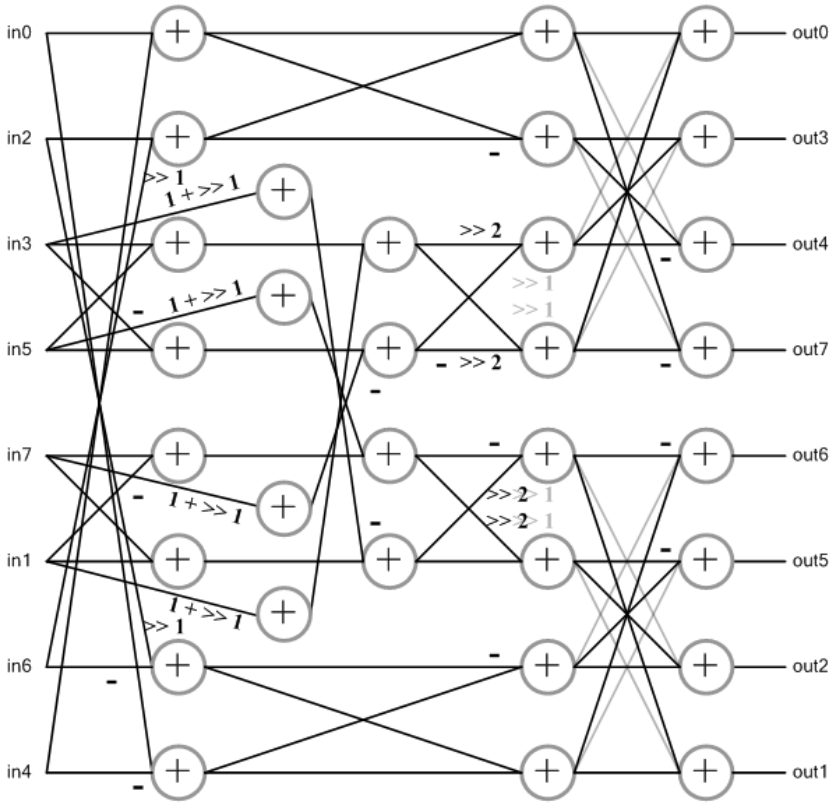


Fig. 50 Data path for 8x8 inverse transform

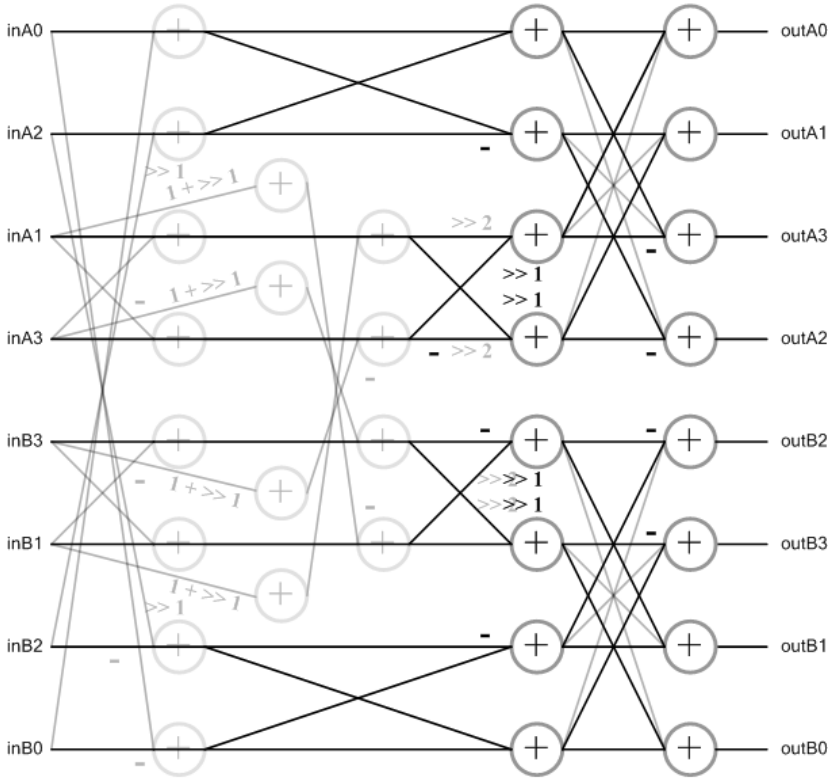


Fig. 51 Data path for 4x4 inverse transform

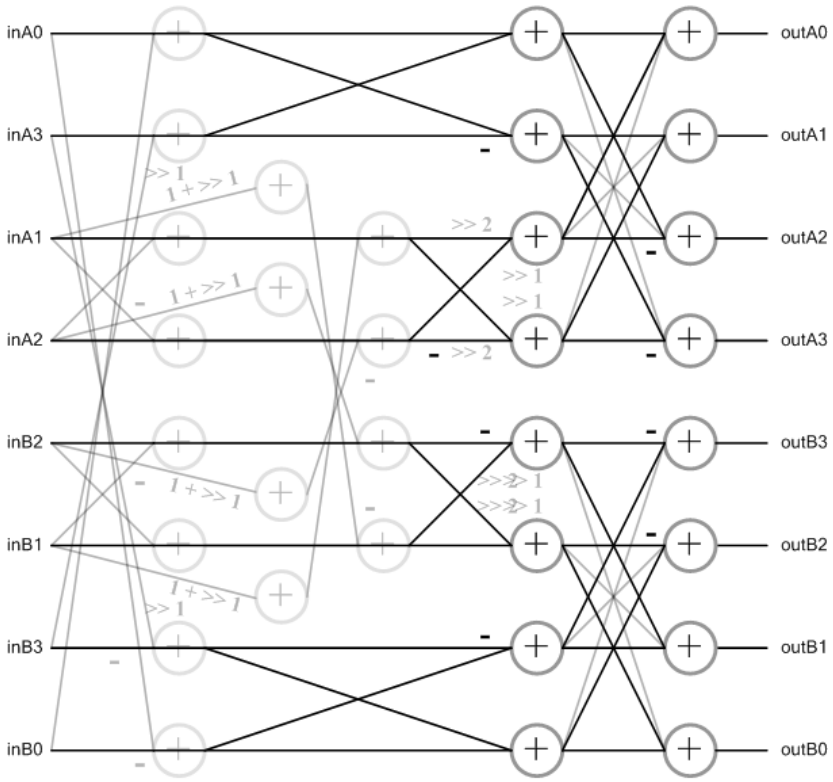


Fig. 52 Data path for Hadamard transform

5.3.7 Quality Enhancement Module

The architecture of quality enhancement coding is introduced in Chapter 2. In hardware design, consider the most of operations in this coding can be processed by quantization module. Therefore, normalization and coefficient subtraction are only two units need to be implemented. Normalization process is similar with quantization, so 8-pixel parallelism and table looking-up method is also used in this unit. The normalization matrices for 4x4 and 8x8 blocks are organized in Table 21 and Table 22.

Table 21 Normalization matrix for a 4x4 block

Point*	(0, 0), (0, 2), (2, 0), (2, 2)	(1, 1), (1, 3), (3, 1), (3, 3)	Other
Value	16	25	20

**: position (i, j) indicates {vertical, horizontal) location of a 4x4 block*

Table 22 Normalization matrix for an 8x8 block

Point*	i % 4 = 0 j % 4 = 0	i % 2 = 1 j % 2 = 1	i % 4 = 2 j % 4 = 2	(i % 2 , j % 2) = (0,1) or (1, 0)	(i % 4, j % 4) = (0, 2) or (2, 0)	Other
Value	64	81	25	72	40	45

**: position (i, j) indicates {vertical, horizontal) location of an 8x8 block*

5.3.8 Memory Organization

The on-chip memory of this work will be introduced in this sub-section. For current macro-block input, since “frame-parallel” encoding scheme is adopted, doubled space would be preserved for current MBs in two frames. Fig. 53 illustrates the memory organization of current source. A dual-port random access memory (DPRAM) with 64 bits width and 96 entries is used for this work.

The convenience of data access is regarded in the addressing method. For a 4x4 block access, the LSB of address would be fixed, for two 8-pixel rows access in intra 8x8 operation, the second LSB is fixed. Besides, luma and chroma data can be classified by the MSB of address. This concept is also adopted in the addressing of other memory blocks which summarized in Table 23.

For the rush of data access in SVC encoding, compared to a H.264/AVC encoder, the memory requirement for an SVC encoder is much higher. Especially for quality enhancement coding, different kinds of coefficients are stored and prepared.

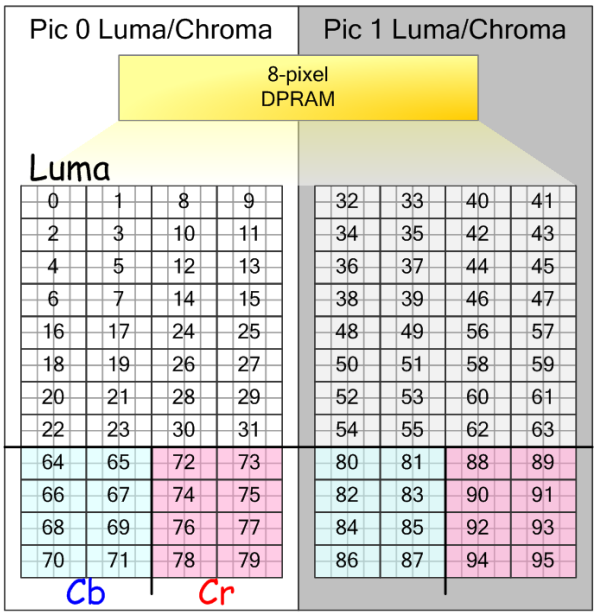


Fig. 53 Memory organization of current MB

Table 23 Internal memory storage for proposed design

Name	*Type*	*Entry*	*Width*
Intra-T phase			
Current MB	Dual port	96 (2 MBs)	64
Neighboring pixels	Single port	960 (2 rows)	64
Neighboring modes	Single port	240 (2 rows)	16
Best mode coefficients	Dual port	128 (2 buffers)	152
QQ-Rec phase			
Transform coefficients for BL	Single port	128 (2 buffers)	128
Scaled transform coefficients for BL	Single port	128 (2 buffers)	136
Reconstructed pixels for BL	Single port	128 (2 buffers)	64
Reconstructed pixels for EL	Single port	96 (2 MBs)	64
Quality Refine phase			
Pre-quantized coefficients	Single port	96 (2 MBs)	136
Scaled transform coefficients for EL	Single port	96 (2 MBs)	136

5.4 Implementation Results

The proposed variable-pixel parallel SVC intra encoder with "TraDED" fast algorithm is designed by Verilog-HDL and implemented by using UMC 90nm 1P9M CMOS technology. The critical path of this design is in 8x8 transform due to the complex architecture.

5.4.1 Design Flow

The standard AISC design flow is adopted in this working progress, shown on Fig. 54. At first, the SVC standard had been studied, and constructed in C model. Then, the fast algorithm is developed to reduce total computational complexity. Architecture is designed with RTL coding and revised according to the timing results of synthesis.

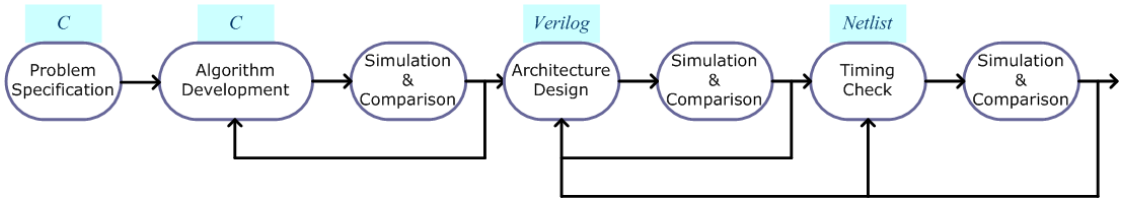


Fig. 54 Design flow of this work

5.4.2 Gate Count

For 135 MHz synthesis frequency (clock period is set to 7.4 ns), the total gate count of this encoder excluded internal memory is about 215 k. The gate count of each component is listed in Table 24.

Compared to previous work [24], the gate count increment of this work is mainly caused by doubled throughput (from 8 pixels to 16 pixels). And for specific components, in transform, the integration of 4x4 transform and 8x8 transform takes much gate space. In cost calculation and mode decision, since “TraDED” is used in this design, extra computations for edge intensities are necessary. In prediction, plane mode generator occupies about 40 % of gate count in this area, which is not supported in previous work.

Table 24 List of gate count for [24] and proposed design

Module	[24]*	This work
Transform	19,868	42,783
- for 4x4 and DC	N/A	15,553
- for 4x4 and 8x8	N/A	27,230
Cost calculation and mode decision	12,923	24,779
- for 4x4, 16x16, and chroma	N/A	11,028
- for 4x4, 8x8, and 16x16	N/A	13,751
Prediction and residual generator	6,646	31,362
- for 4x4, 16x16, and chroma modes	N/A	3,644
- for 4x4, 8x8, and 16x16	N/A	8,713
- DC mode	N/A	1,075
- plane mode	Not Support	12,582
- residual	N/A	5,348
Quantization and reconstructed path	70,104	86,344
- forward / inverse quantization	31,908	56,535
- inverse transform and reconstruction	38,196	29,809
Quality enhancement	Not Support	9,024
Global control and storage	24,534	21,137
Total	**134,075**	**215,429**

**: synthesizing in 145 MHz, UMC 0.13μm technology*

5.4.3 Comparison

Since there is no SVC intra encoder has been published, this design is compared with H.264/AVC intra encoder. Table 25 shows the comparison between [24] and this work. In order to support the higher standard, SVC, this work raises the number of pixel parallelism, also spends much more hardware costs, especially in internal memory usage, the process of quality enhancement coding takes much storage space. Generally speaking, the hardware efficiency of this work is still higher than the previous work [24].

Table 25 Comparison between [24] and this work

Design feature	[24]*	**This work**
CMOS technology	UMC 0.13μm	UMC 90nm
System pipeline	MB-based	MB-based
Pixel parallelism	8 pixels	16 pixels / 8 pixels
Max operation frequency	145 MHz	135MHz
*Gate count***	134 k	215 k
On-chip memory usage	(dual) 48 x 64 (x 1) (single) 16 x 112 (x 2) 8 x 96 (x 2)	(dual) 96 x 64 (x 1) 128 x 152 (x 1) (single) 960 x 64 (x 1) 240 x 16 (x 1) 128 x 128 (x 1) 128 x 136 (x 1) 128 x 64 (x 1) 96 x 64 (x 1) 96 x 136 (x 2)
Standard	H.264/AVC	SVC
Number of quality layers	1	3
Frame rate	30 fps	60 fps
Max target resolution	HD 1080p (1920 x 1080)	HD 1080p (1920 x 1080) + SD 480p (720 x 480) + CIF (352 x 288)
Throughput (MB / sec)	244,800	594,360
Hardware efficiency (throughput / gate count)	1.826	2.759
Processing cycles/MB	600	454
Cost function	Enhanced DCT-based SATD	DCT-based SATD
Mode decision method	Modified 3-step	TraDED
Plane mode	N	Y

**: intra part only*

***: for more detail information, please refer Table 24*

5.5 Summary

In this chapter, the implementation of an SVC intra encoder is presented. Based on the fast algorithm proposed in Chapter 4, this encoder can support SVC standard with target spec: 3 quality layers in 3 resolutions (CIF, SD 480p, and HD 1080p) with 60 Hz frame rate. The gate count of this work is about 215 k, with 135 MHz working frequency, implemented in UMC 90 nm 1P9M CMOS technology.

Chapter 6 Conclusion and Future Work

In summary, the contribution of this thesis can be divided by three parts: in Chapter 3, the memory performance of SVC in system level has been analyzed. By choosing the frame-level encoding method, the system can have the minimal internal storage and the lowest external bandwidth. Moreover, by the help of improvements in other domain: key-picture concept, and shorten the bit-length of FGS coefficients, the external memory bandwidth can be further decreased by over 53%, with only 8.3% increment of internal memory size.

A fast algorithm “TraDED” for intra prediction is proposed in Chapter 4. By observing the transformed patterns, the intensities of texture can be calculated and this result would provide suitable modes as the prediction candidates. Compared to the reference software, “TraDED” skips over 50% of candidate modes in average, with less than 0.05 dB of quality loss and within 1.5 % of bit-rate increment.

The implementation of an intra encoder for SVC with three quality layer is introduced in Chapter 5. Through the application of “TraDED” algorithm and appropriate resource allocation, this encoder can support 60 frames per second throughput in CIF, SD 480p and HD 1080p.

Lastly, for the future work, the proposed design can be further integrated with deblocking filter and entropy coding. For more powerful coding efficiency, inter prediction and inter-layer prediction modules are also essential. Hope this research result can bring some benefits in video applications and promote the visual entertainment for human as well.

Reference

[1] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 17, no. 9, pp. 1103-1120, Sep. 2007.

[2] C. -T. Huang, C. -Y. Chen, Y. -H. Chen, and L. -G. Chen, "Memory analysis of VLSI architecture for 5/3 and 1/3 motion-compensated temporal filtering," *in proceeding of IEEE International Conference on Acoustic Speech, and Signal Processing,* vol. 5, pp. v/93-v/96, Mar. 2005.

[3] Y. -H. Chen, T. -D. Chuang, Y. -J. Chen, and L. -G. Chen, "Bandwidth-efficient encoder framework for H.264/AVC scalable extension," *in proceeding of IEEE International Symposium on Multimedia Workshop,* pp. 401-406, Dec. 2007.

[4] *Advanced Video Coding for Generic Audiovisual Services,* ITU-T Recommendation H.264 and ISO/IEC 14496-10 (MPEG-4 AVC), Joint Video Team, May 2003.

[5] H. Schwarz, D. Marpe, and T. Wiegand, "*Hierarchical B Pictures,"* Joint Video Team, Doc. JVT-P014, Jul. 2005.

[6] H. Schwarz, D. Marpe, and T. Wiegand, "Analysis of hierarchical B-pictures and MCTF," *in proceeding of IEEE International Conference on Mutimedia and Expo,* pp. 1929-1932, Jul. 2006.

[7] *Generic Coding of Moving Pictures and Associate Audio Information – Part 2: Video,* ITU-T Recommendation H.262 and ISO/IEC 13818-2 (MPEG-2 Video), Nov. 1994.

[8] C. A. Segall and G. J. Sullivan, "Spatial scalability within the H.264/AVC scalable

video coding extension," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 17, no. 9, pp. 1121-1135, Sep. 2007.

[9] M. Wien, H. Schwarz, and T. Oelbaum, "Performance analysis of SVC," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 17, no. 9, pp. 1194-1203, Sep. 2007.

[10] Y. -H. Chen, T. -D. Chuang, Y. -H. Chen, C. -H. Tsai, and L. -G. Chen, "Frame-parallel design strategy for high definition B-frame H.264/AVC encoder," *in proceeding of IEEE International Symposium on Circuits and Systems,* pp. 29-32, May 2008.

[11] T. -Y. Chen, G. -L. Li, and T. -S. Chang, "Memory analysis for H.264/AVC scalable extension encoder," *in proceeding of IEEE International Symposium on Circuits and Systems,* pp. 361-364, May 2009.

[12] H. Schwarz, T. Hinz, H. Kirchihoffer, D. Marpe, and T. Wiegard, *"Technical Description of the HHI Proposal for SVC CE1,"* ISO/IEC, JTC 1/SC 29/WG 11 Doc. M11244, Oct. 2004.

[13] B. Meng, O. C. Au, C. -W. Wong; H. -K. Lam, "Efficient intra-prediction algorithm in H.264, " *in proceeding of IEEE International Conference on Signal Processing,* vol. 3, pp. 837-840, Sep. 2003.

[14] F. Fu, X. Lin, and L. Xu, "Fast intra prediction algorithm in H.264-AVC," *in proceeding of IEEE International Conference on Signal Processing,* vol. 2, pp. 1191-1194, Aug. 2004.

[15] C. -C. Cheng and T. -S. Chang, "Fast three step intra prediction algorithm for 4x4 blocks in H.264," *in proceeding of IEEE International Symposium on Circuits and Systems,* vol. 2, pp. 1509-1512, May 2005.

[16] C. -H. Hsia, J. -S. Chiang, Y. -H. Wang, and T. -Y. Teng, "Fast intra prediction mode decision algorithm for H.264/AVC video coding standard," *in proceeding of IEEE International Conference on Intelligent Information Hiding and Multimedia Signal Processing,* vol. 2, pp. 535-538, Nov. 2007.

[17] F. Pan, X. Lin, S. Rahardja, K. P. Lim, Z. G. Li, D. Wu, and S. Wu, "Fast mode decision algorithm for intraprediction in H.264/AVC video coding," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 15, no. 7, pp. 813-822, Jul. 2005.

[18] J. -C. Wang, J. -F. Wang, J. -F. Yang, and J. -T. Chen, "A fast mode decision algorithm and its VLSI design for H.264/AVC intra-prediction," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 17, no. 10, pp. 1412-1422, Oct. 2007.

[19] Y. -C. Wei and J. -F. Yang, "Transformed-domain intra mode decision in H.264/AVC encoder," *in proceeding of IEEE Region 10 Conference,* pp. 1-4, Nov. 2006.

[20] T. Zhang, G. Tian, and S. Goto, "A frequency-based fast block type decision algorithm for intra prediction in H.264/AVC high profile," *in proceeding of IEEE Asia Pacific Conference on Circuits and Systems,* pp 1292-1295, Nov. 2008.

[21] H. -Y. Lin, K. -H. Wu, B. -D. Liu, and J. -F. Yang, "An efficient VLSI architecture for transform-based intra prediction in H.264/AVC," *submitted to IEEE Transactions on Circuits and Systems for Video Technology.*

[22] H.264/MPEG-4 AVC *reference software*, JM 12.4

[23] H. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, "Low-complexity transform and quantization with 16-bit arithmetic for H.26L," *in proceeding of*

IEEE International Conference on Image Processing, vol. 2, pp. 489-492, Sep. 2002.

[24] Y. -K. Lin, D. -W. Li, C. -C. Lin, T. -Y. Kuo, S. -J. Wu, W. -C. Tai, W. -C. Chang, and T. -S. Chang, "A 242mW 10mm^2 1080p H.264/AVC high-profile encoder chip," *IEEE International Solid-State Circuits Conference Digest of Technical Papers,* pp. 314-615, Feb. 2008.

Biographical Notes

姓名：陳之悠

學歷：

國立科學工業園區實驗高級中學　(2000/09 – 2003/06)

國立交通大學電機資訊學士班　(2003/09 – 2007/06)

國立交通大學電子研究所系統組　(2007/09 – 2009/08)

著作：

Tzu-Yu Chen, Gwo-Long Li, and Tian-Sheuan Chang, “Memory analysis for H.264/AVC scalable extension encoder, ” *in proceeding of IEEE International Symposium on Circuit and System*, pp. 361-364, May 2009.