Low Complexity Hardware Oriented H.264/AVC Motion Estimation Algorithm and Related Low Power and Low Cost Architecture Design

HUANG, Yiqing

Graduate School of Information, Production and Systems

Waseda University

February 2010

Abstract

The ever increasing bit-rate on network applications such as broadcasting digital television makes storage capacity larger than ever before. Especially, the advent of Super Hi-Vision (SHV) which has feature of high resolution further intensifies the tough situation. Since limitation exists in network bandwidth and disk storage, the video compression technique is becoming more important than before. As the latest video coding standard, H.264/AVC can provide superior performance to previous standards. However, it also consists of huge complexity. When ASIC (Application Specific Integration Circuits) based real-time hardware system is considered, the intensive complexity in H.264/AVC will cause problems in hardware cost and power consumption. Therefore, to solve the problem, this dissertation focuses on two key issues which are low complexity hardware oriented algorithm and its related architecture.

In H.264/AVC based system, motion estimation (ME) which is the major part of inter prediction is the most significant component. It consists of integer ME (IME) and fractional ME (FME) and occupies almost 90% computation, which makes it a must to divide IME and FME into two separate stages in real-time hardwired encoder. Besides motion estimation part, hardware engine of intra prediction is another time consuming part because of its abundant prediction modes. Moreover, the rate distortion based mode decision part which makes a final judgment of inter and intra modes also consumes lot of computation in the final stage of whole encoding system. Many software based fast algorithms have already been proposed to release complexity of H.264/AVC based system. However, most of these algorithms can not be efficiently realized in hardware because of constraints in hardware design. In hardware, factors such as predictable data flow, regular access of memory and full hardware utilization are important to the whole system's performance. Without considering these factors, hardware cost, throughput and power consumption will increase greatly. So, hardware oriented low complexity algorithm and related low cost and low power hardware architecture are important issues to H.264/AVC based real-time encoder design.

Based on analysis of existing works and current problem, this dissertation mainly targets on low cost and low power H.264/AVC real-time hardwired encoder. In detail, it focuses on IME, FME, intra and mode decision, which are four computation intensive parts in H.264/AVC based system. Firstly, low complexity algorithm which follows hardware data flow is proposed. Secondly, based on proposed algorithm, flexible and highly parallel architectures are given out. Moreover, architecture and circuit optimizations are proposed to further reduce the hardware cost and power consumption.

The whole dissertation consists of 6 chapters as follows.

In the first chapter, introduction in video compression field is given out. The development and feature of video coding standards and emphasis of this dissertation are described in detail.

In the second chapter, hardware oriented low complexity motion estimation algorithms are given out. The complexity reduction is achieved in MRF, search range and matching pattern of H.264/AVC based system. Firstly, for MRF technique, gradient and block matching information are used for fast MRF algorithms. The proposed algorithms release the MRF complexity according to macroblock (MB) features in spatial and temporal domains. Secondly, based on the statistical analysis, it is shown that motion feature is conformity across several frames and search range can be adaptive adjusted according to the motion feature of MB. So, two proposals of search range adjustment is given out in this dissertation. For MB with extreme small motion, search range is restricted into 1/8 of original value. For MB with other cases, the search range is adjusted recursively according to the motion feature of MB on previous frame. Thirdly, since pixel difference can reflect spatial feature of current MB, it is used to classify matching pattern of ME process. An pixel difference based adaptive sub-sampling scheme is proposed, which uses three hardware oriented patterns for MB with different spatial features. By combining all the proposed schemes, the overall algorithm can achieve up to 95.72% complexity reduction with average 0.072dB PSNR loss and 0.902% bit-rate increase based on hardware data flow.

In the third chapter, two flexible IME architectures for adaptive sub-sampling algorithm, namely adaptive propagate partial SAD (APPSAD) and reconfigurable SAD Tree (RSADT), are proposed. By using configurable SAD, the proposed RSADT architecture achieves data organizations in both architectural and memory level, which speeds up processing time and saving power consumption. For APPSAD, the original processing element (PE) is expanded into four different types. According to different matching patterns, only the related type of PE is enabled and power consumption of other types of PE can be saved. Moreover, circuit optimization is applied on both APPSAD and RSADT are optimized. The propagation chain, original PE and adder trees are simplified, with no redundant registers and adders. So, hardware cost and power consumption are further reduced. With TSMC 0.18um CMOS library, it is shown that the proposed architectures can achieve 61.71% saving of processing cycles and up to 39.8% power reduction of existing works.

In the fourth chapter, two low design effort SHV engines for FME and intra prediction are proposed. Firstly, for FME engine, two optimizations in the algorithm level, namely inter mode pre-filtering and one-pass algorithm are proposed. For inter mode pre-filtering, it analyze the motion cost of sub-blocks in IME stage and only focuses on two modes which have smaller cost than others. As for one-pass algorithm, it firstly decides the sliding window based on integer motion cost of neighboring positions. Then, only half and quarter pixel within the sliding window are processed simultaneously, which saves hardware cost and processing time. In the hardware level, with quarter sub-sampling technique in FME stage, a 16-Pel interpolation structure is proposed, which speeds up 4 times of original 4-Pel design while keep almost the same hardware amount. With MB and frame level parallel processing flow, compared with representative design which requires 2.16GHz for 4k×4k@60fps, the proposed FME engine can accomplish real-time processing with only 145MHz. For intra engine, the predictor generation is the most time consuming part. From the analysis of data dependency issue of intra prediction, it is observed that the maximum parallel processing scale is two sub-block instead of original one sub-block way. In this dissertation, one lossless two sub-block parallel data flow are proposed, which saves 37.5% processing time of original one sub-block way. Also, in the original intra predictor generation engine, lots of repetitive computation exists among different modes. In the proposed fully utilized intra predictor generation architecture, no repetitive generation of predictors exists and it is applicable for all intra prediction modes. With proposed architecture, the whole predictor generation process can be finished within only 22.5% cycles of original design. By combining parallel data flow and fully utilized architecture, the proposed intra predictor generation engine is capable of handling 4k×2k@60fps specification.

In the fifth chapter, high complexity problem in H.264/AVC mode decision is discussed. By utilizing spatial and temporal information, complexity reduction is achieved in two stages. Firstly, gradients of current MB and motion vector of encoded MB on both current and previous frames are utilized for pre-stage skip mode check. Secondly, during the motion stage, it is observed that information of motion vector predictor (MVP), block overlapping status and rate distortion cost can indicate the accuracy of matching process. In detail, the MVP represents the accuracy of predicted start point. The block overlapping status of different inter modes indicates the motion trend of object. As for rate distortion cost, it is an objective measurement of matching result. Thus, such information is used for early decision of whole encoding process in the proposed mode decision algorithm. Compared with existing works, the proposed algorithm can achieve up to 53.4% speed-up ratio with trivial quality loss.

In the sixth chapter, the whole dissertation is concluded and future trend in video compression fields is also briefly discussed. In this dissertation, it focuses on IME, FME, intra and mode decision which are four most important parts in H.264/AVC real-time encoding system. Hardware oriented low complexity algorithm and low cost, low power hardware architectures are proposed. By combining hardware oriented algorithms with proposed architectures, compared with recent 4-stage real-time encoder design, about 90.68% power in IME part can be reduced. As for SHV targeted FME and intra engines, about 93.31% and 67.24% estimated power reduction in hardware design.

Contents

List of Figures

List of Tables

LIST OF TABLES

Acronyms

AMPD: advanced mode pre-decision

APPSAD: adaptive propagate partial SAD

B.MB: bottom macroblock

BDBR: bj$o$ntegaard bit-rate

BDPSNR: bj$o$ntegaard peak source to noise ratio

BIP: best integer point

BISP: best integer search position

BitR: bit-rate

BL.MB: bottom-left macroblock

BMMB: big mode macroblock

BR.MB: bottom-right macroblock

CMO: cross mode overlapping

CMPR32: 3-2 compressor

CMPR42: 4-2 compressor

Co.MB: co-located macroblock

CRS: cross reuse structure

CSAD: configurable Sum of absolute difference

Cur.MB: current macroblock

DB: de-blocking

dynamic_SR: dynamic search range scheme

EC: entropy coding

fm: full mode

FME: fractional motion estimation

HD: high definition

Homo: homogeneous

HW_utiliz: hardware utilization

I4MB: intra 4×4 prediction modes

I16MB: intra 16×16 prediction modes

IBO: inner block overlapping

ICI: immediate carry-in

ICO: immediate carry-out

IMC: integer motion cost

IME: integer motion estimation

IMV: integer motion vector

IP: intra prediction

L.MB: left macroblock

LU.MB: left-up macroblock

MAFD: mean of absolute frame difference

MB: macroblock

MCDOP: motion cost oriented one-pass

ME: motion estimation

MET: motion estimation time

Min_Freq: minimum required frequency

mr: mode reduction

MP: matching pattern

MRF: multiple reference frame

MRMPF: mode reduction based mode pre-filtering

MSU: modified snake scan unit

MV: motion vector

MVP: motion vector predictor

NMB: normal macroblock

Non Homo: nonhomogeneous

PA: pixel assemble

PD: pixel difference

PDA: pixel difference analysis

PE: processing element

PE_CONV: conventional processing element

PPSAD: propagate partial sad

pro_SR: proposed search range scheme

PSNR: peak source to noise ratio

PU: processing unit

PUH: processing unit for half pixel refinement

PUQ: processing unit for quarter pixel refinement

QP: quantization parameter

R.MB: right macroblock

RD: rate distortion

RSA: reference shift array

RSADT: reconfigurable SAD Tree

RU.MB: right-up macroblock

SA: similarity analysis

SAD: sum of absolute difference

SAD8x8_BL: bottom-left 8x8 SAD

SAD8x8_BR: bottom-right 8x8 SAD

SAD8x8_LU: left-up 8x8 SAD

SAD8x8_RU: right-up 8x8 SAD

SHV: super hi-vision

SR: search range

SU: snake scan unit

U.MB: upper macroblock

UBP: unified pixel block

VBS: variable block size

Chapter 1

Introduction

1.1 Background and purpose of this dissertation

Sixteen years ago, the advent of MPEG-2 standard enriches our life with worldwide digital television system. From that time, MPEG-2 has become a key technique which is widely used in transmission of High Definition (HD) TV signals over satellite, cable, and the storage of high-quality SD video signals onto DVDs. However, the increasing demand for more service over network, or desire for vivid and impressive daily life makes bit rates on network roar dramatically. Nowadays, high bit rate connections are almost everywhere around us. The ever increasing tough situation on network transmission continuously pushes video compression technique forward.

Currently, the latest video coding standard is H.264/AVC which firstly comes to existence in 2003 [1]. Compared with previous standards, the performance improvement of H.264/AVC is quite significant [2]. Figure. 1.1 demonstrates the development of video coding standards. Compared with MPEG-4 [3], H.263 [4], and MPEG-2 [5], the H.264/AVC standard can achieve 39%, 49% and 64% bit-rate reduction, respectively. In the near future, H.265 may come into existence and the performance improvement of new standard is always a heated topics. Figure. 1.2 gives out the whole block diagram of H.264/AVC based hybrid encoding system. The bold italic font marked on the diagram represent the new techniques introduced by H.264/AVC standard. For example, in H.264/AVC, it adopts techniques such as variable block size (VBS), multiple reference frame (MRF),

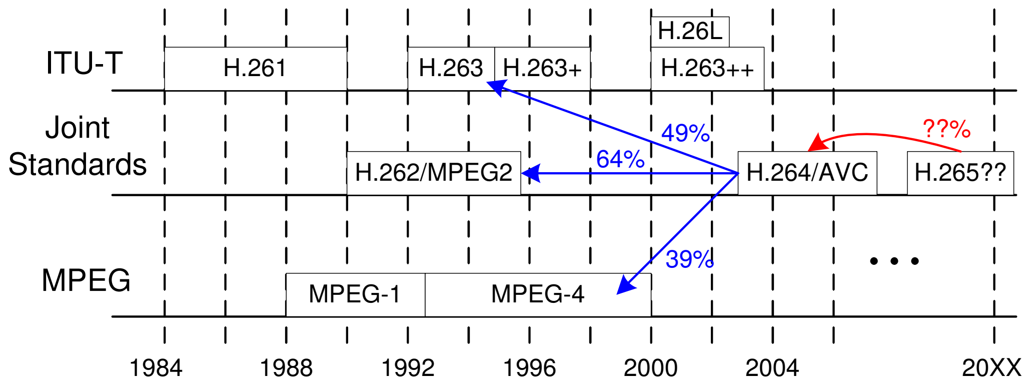


Figure 1.1: Overview of video coding standards

intra prediction (IP), context adaptive entropy coding (EC), in-loop de-blocking (DB) and so on. These techniques mainly fall into three categories. Firstly, H.264/AVC introduces techniques which target at higher prediction accuracy. The ME and IP parts fully exploit the temporal and spatial redundancy. Besides skip modes, there are seven inter modes with different block sizes in inter prediction. Considering the MRF technique, the efficiency of condensing temporal information is very high. As for IP modes, there exists nine intra 4×4 modes and four intra 16×16 modes. All these inter and intra modes are involved in a rate distortion based encoding process, which ensures the best outcome result over available resources. Secondly, H.264/AVC introduces techniques which focus on image enhancement. To remove the visible artifacts of block based hybrid compression scheme, it uses an adaptive in-loop de-blocking filter, where the strength of filtering is controlled by the values of several syntax elements. Also, the interpolation of half and quarter pixel for fractional motion estimation is an efficient way to compensate the inevitable aliasing problem, which also leads to better image quality. Thirdly, H.264/AVC introduces new mathematics model which greatly improves the compression capability. The powerful entropy coding method, namely CABAC, provides a good solution to the ever increasing bit rates.

Although, there are many appealing points in H.264/AVC standard, the shortcoming of this standard is also quite obvious. Compared with previous standards, the complexity problem of H.264/AVC become a 'hot potato' and many researchers focus on this topic for several years. The computation complexity of each part is also marked on Fig. 1.2.

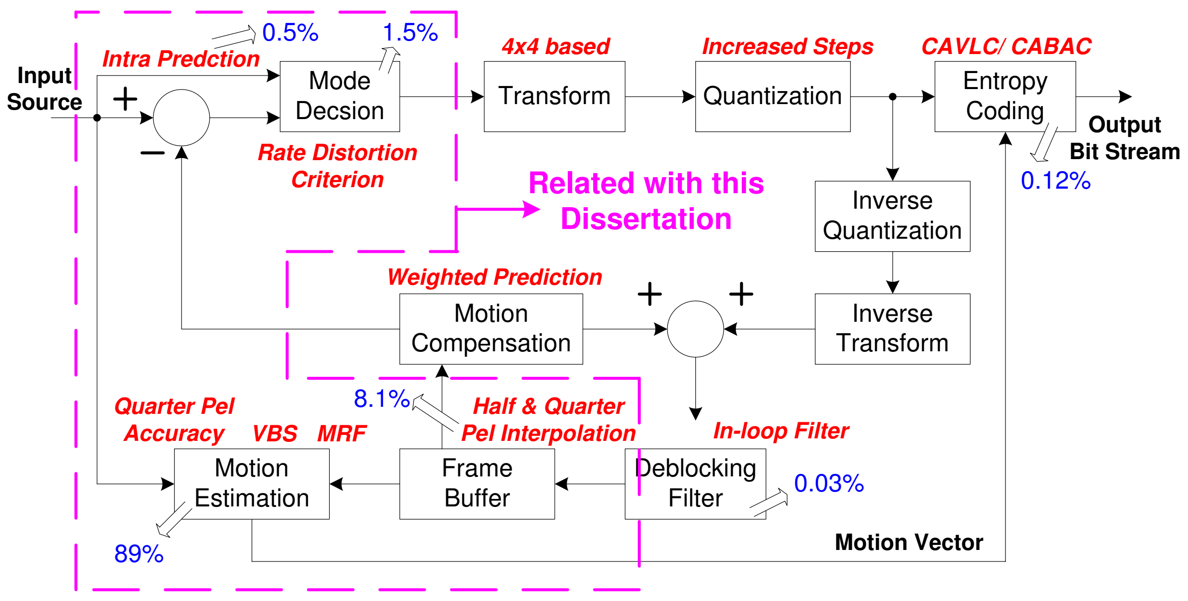


Figure 1.2: Block diagram of H.264/AVC video coding system

Besides IP, mode decision and interpolation, the ME part is the most significant one, which occupies almost 90% computation. In order to reduce computation complexity while keep video quality, many software algorithms are proposed to speedup ME process. However, when hardware is considered, the efficiency of software level algorithm is greatly decreased. The high throughput of ME part makes pipeline stage a must, which deteriorates the efficiency of many fast algorithms. Also, important issues in hardware field are quite different from software region which has abundant power resource and computation capability as long as the computer is strong enough. In hardware fields such as ASIC design, issues like hardware cost, parallel processing, power dissipation, data reuse, memory size and hardware utilization are of great importance. Therefore, there exist a gap between software algorithm and hardware design. The purpose of this dissertation is to fill in this gap and propose hardware friendly fast algorithm together with some low hardware cost and low power architectures. The related research topics in this thesis are marked with broken lines in Fig. 1.2.

1.2 Scope of this dissertation

This dissertation focuses on hardware friendly low complexity fast motion estimation algorithm and related low cost architecture. To attain this goal, this work focuses on three areas of research:

1. hardware friendly algorithm
2. low cost hardware architecture
3. fast mode decision scheme

To cover these three areas, the dissertation consists of six chapters as shown in Fig. 1.3.

Chapter 2 describes the origin of video quality loss in sampling based digital signal system. Based on theoretical and statistical analysis, several hardware friendly complexity reduction schemes are proposed. The proposed algorithm is based on hardware data flow and it releases the complexity in MRF technique, redundant search points and full pixel matching pattern. Experimental results show that, the proposed hardware friendly algorithm can achieve up to 95.26% complexity reduction and is orthogonal to existing software oriented fast algorithms. Moreover, all the proposed schemes can be easily implemented in pipeline stage based real-time encoding system.

In chapter 3, two HDTV targeted flexible hardware architectures are given out. The proposed structures adopts adaptive sub-sampling algorithm which can not be efficiently realized on existing SAD Tree and propagate partial SAD (PPSAD) architectures. In the proposed architectures, architectural level and memory level data organization is adopted, which enables full data reuse, hardware utilization and lower power consumption feature for adaptive sub-sampling algorithm. Compared with original design, the proposed reconfigurable SAD Tree and adaptive PPSAD architectures can achieve 38.8% and 39.8% reduction of power dissipation.

In chapter 4, the dissertation focuses on the high throughput issue of Super Hi-Vision (SHV) application. With the advent of SHV concept, the hardware implementation of SHV based real-time encoding system has become a hot topic. From the analysis of

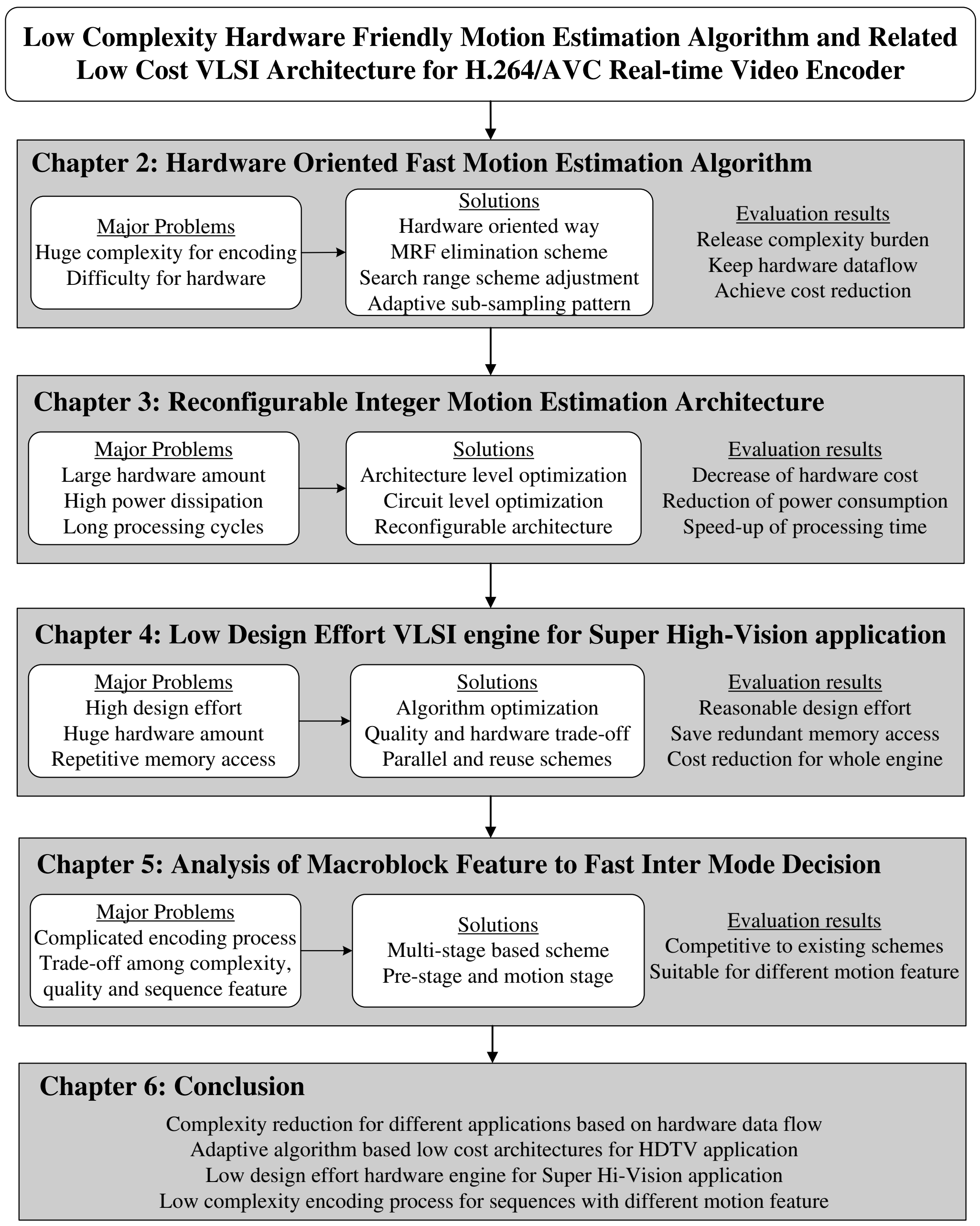

Low Complexity Hardware Friendly Motion Estimation Algorithm and Related Low Cost VLSI Architecture for H.264/AVC Real-time Video Encoder
Chapter 2: Hardware Oriented Fast Motion Estimation Algorithm
Major Problems
Huge complexity for encoding
Difficulty for hardware
Solutions
Hardware oriented way
MRF elimination scheme
Search range scheme adjustment
Adaptive sub-sampling pattern
Evaluation results
Release complexity burden
Keep hardware dataflow
Achieve cost reduction
Chapter 3: Reconfigurable Integer Motion Estimation Architecture
Major Problems
Large hardware amount
High power dissipation
Long processing cycles
Solutions
Architecture level optimization
Circuit level optimization
Reconfigurable architecture
Evaluation results
Decrease of hardware cost
Reduction of power consumption
Speed-up of processing time
Chapter 4: Low Design Effort VLSI engine for Super High-Vision application
Major Problems
High design effort
Huge hardware amount
Repetitive memory access
Solutions
Algorithm optimization
Quality and hardware trade-off
Parallel and reuse schemes
Evaluation results
Reasonable design effort
Save redundant memory access
Cost reduction for whole engine
Chapter 5: Analysis of Macroblock Feature to Fast Inter Mode Decision
Major Problems
Complicated encoding process
Trade-off among complexity, quality and sequence feature
Solutions
Multi-stage based scheme
Pre-stage and motion stage
Evaluation results
Competitive to existing schemes
Suitable for different motion feature
Chapter 6: Conclusion
Complexity reduction for different applications based on hardware data flow
Adaptive algorithm based low cost architectures for HDTV application
Low design effort hardware engine for Super Hi-Vision application
Low complexity encoding process for sequences with different motion feature

Figure 1.3: Overview of this dissertation

existing works, the simple extension of these works to SHV will cause high design effort, large hardware resource and redundant memory access. In the propose architecture, algorithm level optimization and hardware level parallel processing are both adopted to satisfy the throughput issue. With only 145MHz work speed, one SHV 4k×4k@60fps targeted fractional motion estimation engine is given out. As for intra prediction, the predictor generation part is the most significant component towards high throughput application. In this dissertation, one highly parallel intra predictor generation structure is given out. Based on parallel processing flow and dedicated fully reuse structure, about 77.5% processing time is saved compared to original design.

For H.264/AVC based real-time system, mode decision is another important part considering the complexity of whole encoding system. The trade-off among video quality, complexity reduction and image feature is always a tough research topic. In chapter 5, one novel inter mode decision algorithm is introduced. The propose scheme achieves complexity reduction in a multi-stage way, which makes it suitable for image with different motion features. Compared with existing works, the proposal is superior to other schemes among various types of sequences.

Chapter 6 summarizes the whole research activities and gives out a brief view of future research direction which will further push my current research towards higher level and wider application fields

Chapter 2

Hardware oriented fast H.264/AVC motion estimation algorithm

2.1 Introduction

As mentioned in previous chapter, the H.264/AVC standard is superior to previous ones in terms of image quality and compression capability. However, it is also computation intensive due to many dedicated techniques. Literature [6] gives out complexity distribution of each part. The motion estimation (ME) part which occupies almost 90% computation turns out to be the most significant part. As shown in Fig. 2.1, the overwhelming complexity in ME mainly comes from five aspects. They are search pattern, search range, sampling pattern, VBS, and reference frame number. During ME of current MB, the VBS technique will divide one 16×16 block into 16×8, 8×16, and 8×8 modes. When 8×8 mode is selected, it can be further divided into 8×4, 4×8 and 4×4 modes. Motion estimation is executed on each mode. For sampling pattern, as shown in Fig. 2.1, when quarter sub-sampling is used, only 1/4 of original pixels are used for block matching process. So, 75% calculation in block matching process can be saved. However, the direct sub-sampling will cause quality degradation. The relationship of reference frame number to complexity is linear. When 5 reference frames are used, the complexity will increase 5 times compared with 1 reference frame under the same conditions. The setting of search range determines the number of candidate search points, which also affects complexity a

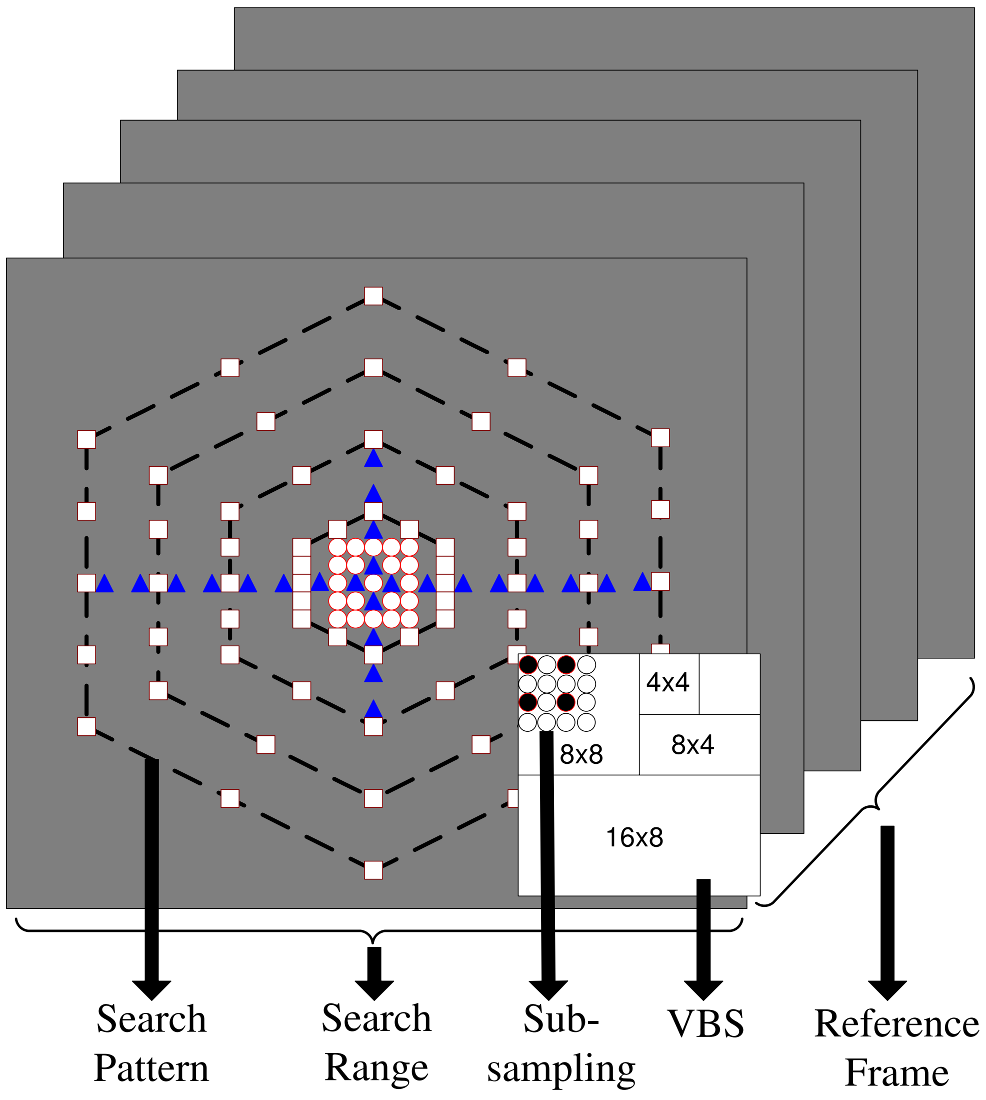


Figure 2.1: Complexity in H.264 motion estimation

lot. In terms of search pattern, many existing patterns such as diamond search [7] [8], four-step search [9], three-step search [10], predictive zonal search [11] [12] and hexagon pattern [13] have already been proposed to reduce search points. Fig. 2.1 is an example of hexagon search pattern.

In order to reduce computation complexity while keep video quality, many works have been done [6, 13, 14, 15, 16]. Literature [14] proposes a fast motion estimation algorithm which is based on analysis of motion vectors (MVs) in previous frames. In literature [15], it uses the MVs of previous frames and up-layer blocks to reduce computation complexity of search points and reference frames. In case of [16], the proposed algorithm first builds up three error surface by using initial 3 block modes (16×16, 8×8, 4×4). The decision of whether to test other modes or finer sub-block partition is based on the error surface analysis. The work of literature [6] uses four heuristic criterions to early terminate the ME process. These algorithms can achieve 30% to 90% reduction in ME time. As for search pattern based fast motion estimation, the UMHexagon search [13] can achieve ME time reduction up to 90%.

In hardware field, as mentioned by many works [17, 18, 19], it is a must to divide motion estimation engine into two stages due to the huge throughput in every clock cycle. As shown in Fig. 2.2, the integer motion estimation (IME) engine is arranged in the first stage while fractional motion estimation (FME) is in the second stage. Therefore, early termination on FME stage like [6] does not work because the IME which occupies 52% computation has already finished its work before handling best MVs to FME stage. As for motion vector based fast algorithms [14][15], they are not favorable for hardware because the storage of all MVs in previous frames is a great burden on system's hardware cost. For instance, with 24×24 search window size, 10 bits are required for storage of one MB's MV in [14]. When 5 reference frames are adopted, even in the CIF format, the extra SRAM will be 19.8k bits. With the increase of image size (HDTV for example), the related extra memory will cause a serious burden on the system. For [16], since the rate distortion cost is only available in the last stage based on the hardwired video coding system, it is impossible to apply this algorithm in real-time encoding process. In terms of search pattern based fast algorithm [13], the irregular access of memory and unpredictable data flow make this algorithm difficult for hardware implementation. So, the existing software oriented algorithms are either impractical or inefficient for hardware design. For hardwired video encoding system, the widely adopted search scheme is full search algorithm which has best video quality, regular memory access and fixed processing control [20].

In this chapter, several hardware friendly fast motion estimation schemes are given out, which achieves complexity reduction while maintains full search data flow unchanged. Firstly, for MRF technique, two low complexity schemes are introduced. Based on mathematics analysis, the aliasing problem in image processing field is discussed. Image with

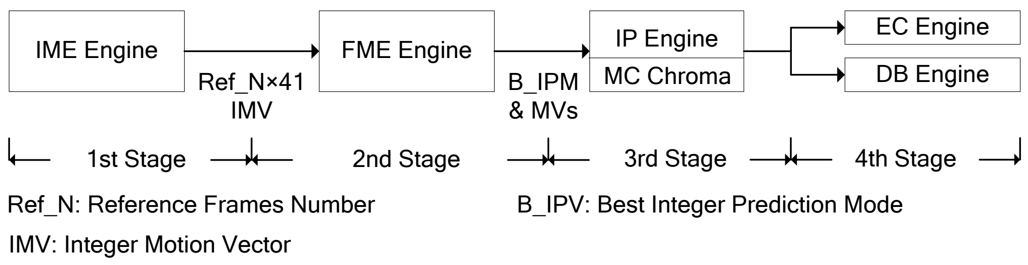


Figure 2.2: 4-stage pipeline based video coding system

high frequency feature is regarded as aliasing sensitive one and MRF technique is applied on such image. In this dissertation, I use Sobel edge detector to classify MB with different frequency feature. Also, simulation shows that for image which consists of abundant stationary parts, MRF can be eliminated. In this dissertation, similarity analysis is executed on central nine positions during block matching on first frame. The MRF technique on stationary MB is disabled to achieve further reduction of complexity. Secondly, in terms of search range, two adaptive search range adjustment schemes are given out. For small motion MB, search range is restricted in a local centering field and redundant search points are removed consequently. For ordinary motion MB, one recursive 6-ring search range adjustment scheme is introduced to achieve complexity reduction for such MB. Furthermore, in the aspect of matching pattern, one adaptive sub-sampling scheme is given out to release complexity and compensate quality loss of direct sub-sampling technique. The detail of each scheme is shown in the remaining parts of this chapter.

2.2 Hardware oriented multiple reference frame elimination

In this section, the aliasing problem in conventional video encoding system is analyzed. After that, two complexity reduction schemes for MRF technique are given out.

2.2.1 Aliasing problem and impact of edge detection

In [21], it has already proved that aliasing is the main reason that deteriorates video quality. The adoptions of MRF and sub-pel interpolation in H.264 are actually to compensate for the aliasing problem. Here, I will analyze the aliasing problem in spatial and frequency domains and then give out influence of edge gradient on frequency spectrum.

In order to ease the analysis, only one dimension signal is analyzed and the spatial sampling interval is assumed to be $I = 1$. Let $l(x)$ be spatial continuous signal. The $l_t(x)$ and $l_{t-1}(x)$ are signals at time instance t and $t-1$. Their spatial Fourier transforms are shown in Eq. 2.1. The d_x is the distance between $l_t(x)$ and $l_{t-1}(x)$. It is shown that $L_{t-1}(j\omega_x)$ and $L_t(j\omega_x)$ are the same except their phase difference.

Let $s_t(x_n)$ and $s_{t-1}(x_n)$ be sampling results of space continuous signals $l_t(x)$ and $l_{t-1}(x)$ and their Fourier transform is shown in Eq. 2.2. Equation. 2.2 shows that aliasing problem can be avoided if Eq. 2.3 which represents the band limit low pass filter in the image sensor system is satisfied.

$$l_t(x) \Leftrightarrow L_t(j\omega_x) = L_{t-1}(j\omega_x) \cdot e^{-jd_x\omega_x} \tag{2.1}$$

$$S_t(j\omega_x) = S_{t-1}(j\omega_x) \cdot e^{-jd_x(\omega_x - k2\pi)} = \sum_{k=-\infty}^{+\infty} L_{t-1}(j\omega_x - jk2\pi) \cdot e^{-jd_x(\omega_x - k2\pi)} \tag{2.2}$$

$$L_{t-1}(j\omega_x) = 0, \qquad |\omega_x| \geq \pi \tag{2.3}$$

However, due to the nonexistence of idea low pass filter, the aliasing problem occurs inevitably in video coding system, as shown in Fig. 2.3. Another important result which is derived from Eq. 2.2 and Eq. 2.3 is that the image rich of high frequency signals is vulnerable to be affected by aliasing problem.

Figure. 2.4 is the rate distortion (RD) curves of two qcif sequences. It is shown that 'mobile_qcif' is more sensitive to MRF than 'football_qcif'. The quality degradation of 'mobile_qcif' with 1 and 5 reference frames is up to 1.5 dB, which is unacceptable for video coding system. In fact, from the features of sequences, it is shown that many textures are contained in 'mobile_qcif' and sharp edges in the spatial domain will generate rich

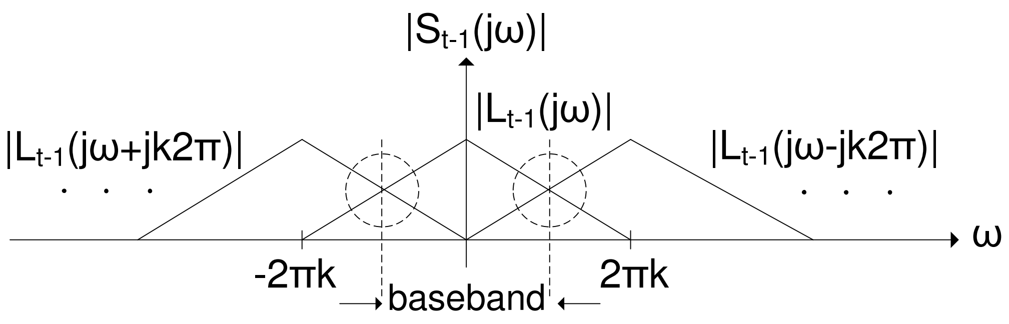


Figure 2.3: Aliasing in Hybrid Video Coding

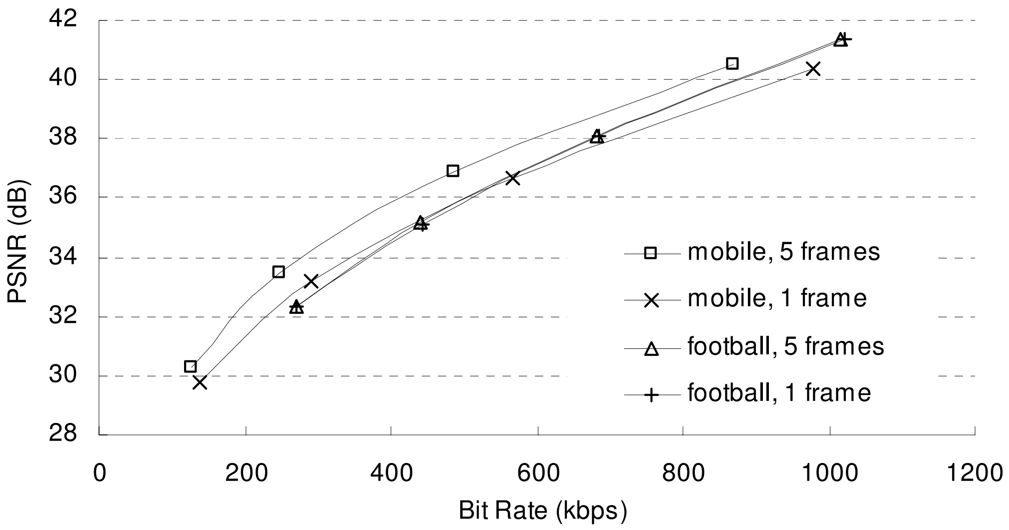
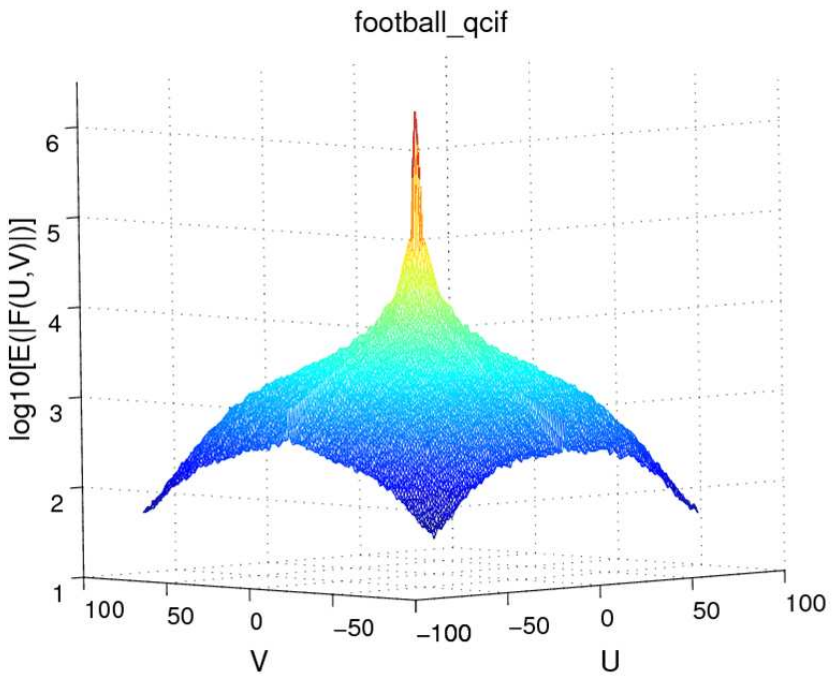


Figure 2.4: RD Curves of QCIF 'football' and 'mobile'

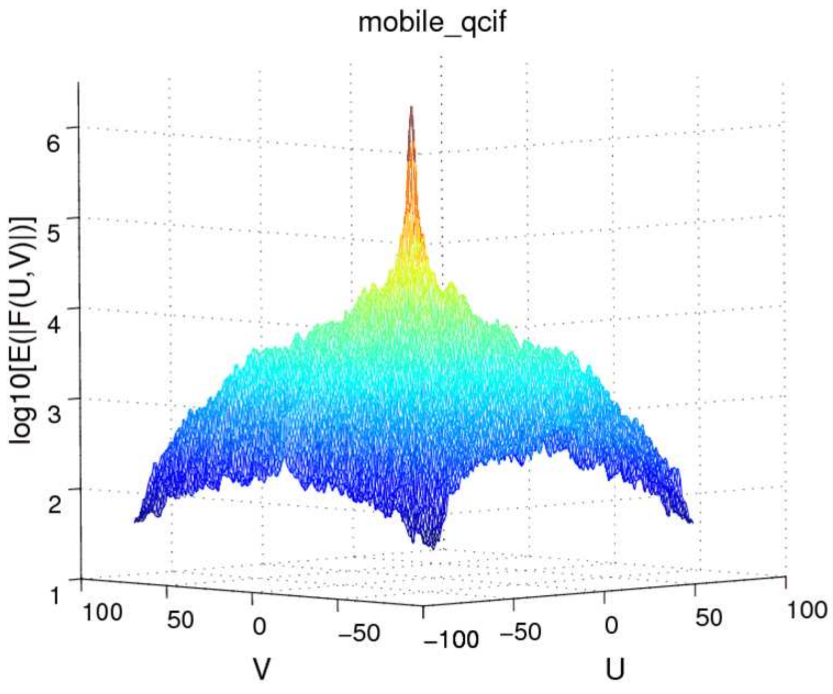
high frequency signals after Fourier transform. The abundant high frequency ingredient in 'mobile_qcif' is the main reason of the occurrence of aliasing.

Figure. 2.5 is the 2-D Fourier spectrum amplitude of two sequences. Hamming window is used to compensate the spectrum leakage. The spectrum analysis obviously shows that high frequency signal in 'mobile_qcif' is much more abundant than 'football_qcif'. Thus, from the above theoretical analysis, it is proved that aliasing is the main reason of video quality degradation and Fourier spectrum can reflect the importance of MRF for video sequence.

The intuitive way of adjusting reference frame number is through analysis of Fourier spectrum. However, such kind of decision criterion is impractical because the compu-



(a) 'football_qcif' frequency



(b) 'mobile_qcif' frequency

Figure 2.5: 2-D Fourier Spectrum Amplitude of 'football_qcif' and 'mobile_qcif'

tation complexity will increase dramatically. In fact, the signal's frequency spectrum is coordinate with its gradient amplitude. Edge information in MB will reflect the spread of

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Figure 2.6: Convolution mask of Sobel operator

frequency spectrum in that MB and gradient analysis is feasible to be used as a decision criterion. In the edge detection based reference frame elimination scheme, I use result of gradient amplitude of each MB to restrict number of reference frames.

2.2.2 Gradient based multiple reference frame elimination

In edge detection field, there exist many operators. Among all of them, Sobel operator is widely used to get 2-D spatial gradient by emphasizing the edges which represent high spatial frequency. So, I use Sobel operator in the proposed fast algorithm. In fact, the Sobel operator is already applied in many mode decision algorithms [22][23][24] and its merit is proved by these algorithms. The convolution mask of Sobel edge detector is described in Fig. 2.6. In luminance picture, if $P(m,n)$ denotes the pixel value at (m,n) position, as shown in Eq. 2.4 and Eq. 2.5, its gradients in x-direction and y-direction are $G_x(m,n)$ and $G_y(m,n)$. $G(m,n)$, which is the gradient of $P(m,n)$ is calculated by addition of $G_x(m,n)$ and $G_y(m,n)$, as shown in Eq. 2.6.

$$\begin{aligned} G_x(m,n) = |&P(m-1,n-1) + 2P(m-1,n) \\ &+P(m-1,n+1) - P(m+1,n-1) \\ &-2P(m+1,n) - P(m+1,n+1)| \end{aligned} \tag{2.4}$$

$$\begin{aligned} G_y(m,n) = |P(m-1,n-1) + 2P(m,n-1) \\ +P(m+1,n-1) - P(m-1,n+1) \\ -2P(m,n+1) - P(m+1,n+1)| \end{aligned} \tag{2.5}$$

$$G(m,n) = G_x(m,n) + G_y(m,n) \tag{2.6}$$

$$\begin{cases} G(m,n) < THR_G, & Homo \\ otherwise, & Non\ Homo \end{cases} \tag{2.7}$$

Figure. 2.7 is the MB partition in H.264 VBSME algorithm. The edge detection criterion is applied base on the partition enclosed in dashed lines. The minimum block size in gradient analysis is 8×8. The block is regarded as homogeneous block if the analysis result is within certain threshold. Figure. 2.8 is the flow chart of proposed edge gradient analysis procedure. Firstly, gradient analysis is executed on each 8×8 block in the MB. One 8×8 block is judged as homogeneous (Homo) block if it satisfies Eq. 2.7. Based on the result of four 8×8 block, one 16×8 block is judged as homo block if its sub 8×8 blocks are all homo blocks. For example, B16 × 8_0 is homo block if B8 × 8_00 and B8 × 8_01 are all homo blocks. The 16×16 block is regarded as homo block only if all of its four sub 8×8 blocks are homo blocks. Otherwise, it is treated as nonhomogeneous (Non Homo) 16×16 block. Here, the setting of THR_G is a critical factor that affects both computation complexity and video quality. If the THR_G is set too high, the video quality will degrade greatly although complexity reduction can be achieved to some extent. On the other hand, too low THR_G can not release the intensive computation of MRF algorithm. In the following part, I will analyze the setting of THR_G in detail through experimental result. The edge gradient analysis is executed at the same time of loading pixels of current MB. It is finished before IME starts and will decides the reference frame number for the following block matching process.

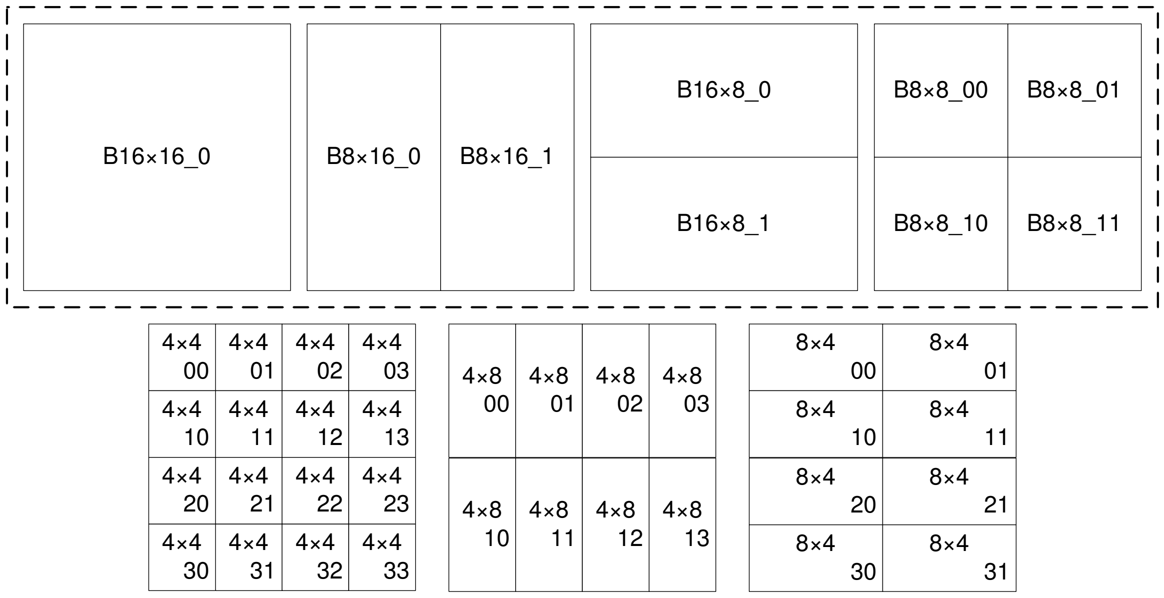


Figure 2.7: MB partition in VBSME algorithm

2.2.3 Quantization parameter based threshold adjustment

From the theoretical point of view, the threshold setting is always a trade-off between quality and complexity. The prediction error e in block matching process can be assumed as a jointly Gaussian source with zero mean and variance σ^2. According to [25], the distortion of quantization D is approximated as $QP^2/3$, where QP is the quantization parameter. So, the rate distortion function [26] can be represented as Eq. 5.12, where $R(D)$ is the related transmission bit-rate for distortion D. The σ^2 represents maximum distortion based on Gaussian model. When distortion D equals to zero, it indicates that original signal is reconstructed without any loss in image detail. All the information of image (including textures and noise) is exacted the same as original source image. Maximum transmission bit-rate is required for keeping the related information. In fact, such case is one ultimate state which will never happen in real video encoding system, like H.264/AVC. The reason is that the transform and quantization will cause some loss in image detail, which makes distortion between original source image and reconstructed one occur inevitably. On the other hand, when D is larger than σ^2, the related transmission bit-rate for D will become zero. This conclusion is in accordance with QP setting in H.264 encoding system. With the increase of QP, the smoothness of reconstructed frames

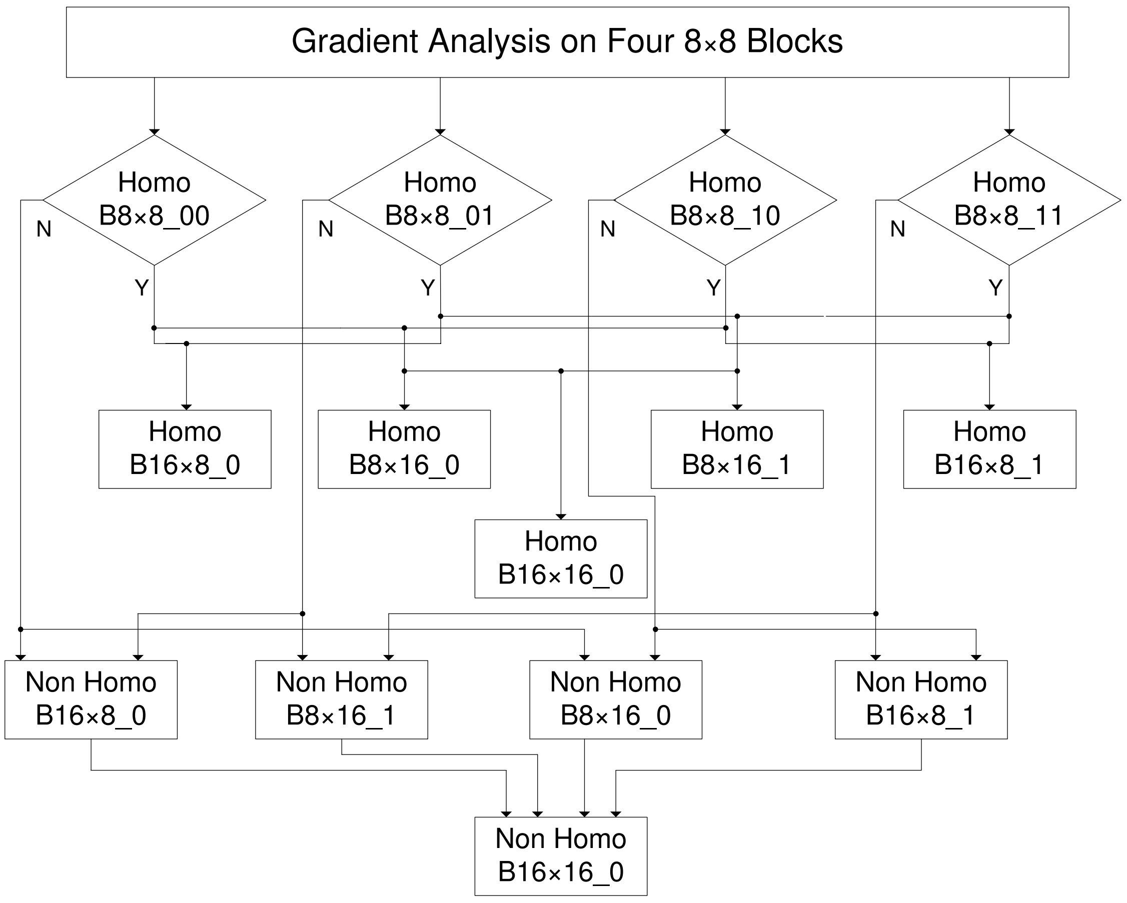


Figure 2.8: Edge gradient analysis flow chart

is increased, which results in decline of image's details. The related residue value is also decreased. It means that quality degradation for edge abundant image is quite obvious under big QP. In the extreme case, all the details are removed by one very large QP and the residue information is vanished, which indicates that no transmission bit-rate is required. Thus, from theoretical analysis of [25] and [26], the threshold can be simply regarded as linear relationship with QP value.

$$R(D) = \begin{cases} \frac{1}{2} log \frac{\sigma^2}{D}, & 0 \le D \le \sigma^2 \\ 0, & D > \sigma^2 \end{cases} \tag{2.8}$$

From the statistical point of view, exhaustive experiments are executed to get optimum threshold value. I apply edge gradient based reference frame elimination scheme on typical sequences. Since the setting of QP value will affect video quality which is represented by PSNR and bit-rate (BitR) variation, I define the $\Delta PSNR$ and $\Delta BitR$ as two tolerance constraints under different QPs, as shown in Eq. 2.9. The $PSNR_{pro}$ and $BitR_{pro}$ represent the result based on proposed algorithm while $PSNR_{jm}$ and $BitR_{jm}$ are the result based on original JM full search algorithm. Equation. 2.9 can clearly show the PSNR and BitR difference of each point on RD curves.

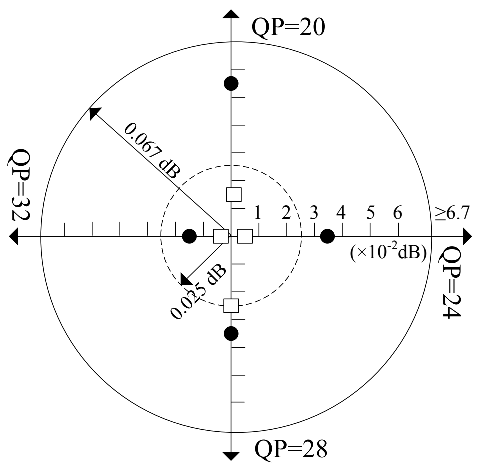
$$\begin{cases} \Delta PSNR = |PSNR_{pro} - PSNR_{jm}| \\ \Delta BitR = |10logBitR_{pro} - 10logBitR_{jm}| \end{cases} \tag{2.9}$$

Several THR_G value is applied on typical MRF sensitive sequences to test the impact of THR_G. The sequences used are 'foreman_qcif' and 'mobile_qcif' which are both MRF sensitive sequences. Five reference frames are enabled and 200 frames are encoded under baseline profile. Table. 2.1 is the experimental result with THR_G ranging from 160 to 360. On the whole, it is shown that for the same QP, the video quality degrades with the increase of THR_G. Specifically, for 'foreman_qcif', if 0.067 dB is set as maximum tolerance constraint of PSNR loss and 0.025dB as maximum tolerance constraint of bit-rate gain, then it is shown that large THR_G is only suitable for big QP value. The data with asterisk represent the violation data against constraint. Figure. 2.9 is the tolerance graph of 'foreman_qcif' based on Table. 2.1. It depicts the relationship of THR_G, $\Delta PSNR$ and $\Delta BitR$. Each black circle on the axle represents the $\Delta PSNR$ under certain QP. Each white square represents the corresponding $\Delta BitR$. The solid circle line is the PSNR tolerance constraint while the broken circle line is the BitR constraint. Based on Table. 2.1 and Fig. 2.9, it is shown that when THR_G is set linearly with QP, maximum ME time can be achieved while video quality loss is under constraint. Different sequences have different tolerance degree. However, the linear relationship between THR_G and QP is the same for MRF sensitive sequences. For example, as shown in Table. 2.1, if I set 0.03dB for PSNR constraint and 0.015dB for BitR constraint of 'mobile_qcif', then it is also possible to get the linear relationship between QP and its THR_G. In fact, the increase of QP means that the reference frame will be more smooth so that the ratio of homo block is increased, which makes it reasonable to change THR_G according to QP. Therefore, I set the THR_G of edge gradient based reference frame number adjustment scheme as $10QP$ to achieve much ME time reduction while keep good video quality.

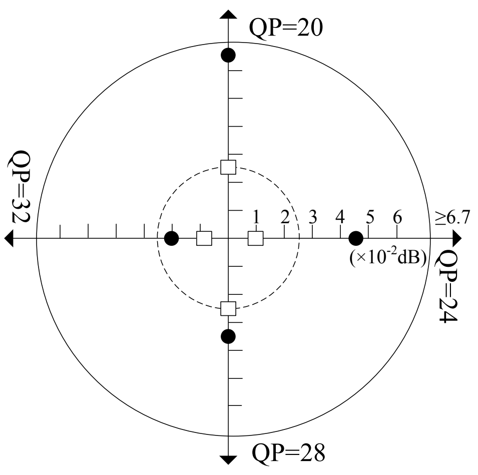
Table 2.1: Impact of THR_G on sequences

S1, S2		$\Delta PSNR$ $\times 10^{-2}$(dB)		$\Delta BitR$(dB) $\times 10^{-2}$(dB)		MET_R (%)	
THR_G	QP	S1	S2	S1	S2	S1	S2
160	20	5.5	1.9	1.5	0.6	56.10	56.55
	24	3.5	0.8	0.5	0.2	52.25	54.87
	28	3.5	1.2	2.5	0.6	48.17	53.17
	32	1.5	1.6	0.3	0.4	44.41	50.54
200	20	6.3	1.8	2.5	1.4	57.32	57.50
	24	4.6	2.4	1.1	0.7	54.36	55.61
	28	3.5	1.7	2.5	0.4	50.45	53.18
	32	2.0	0.8	0.9	1.9	45.83	51.23
240	20	$*$**7.0**	$*$**3.2**	$*$**2.6**	$*$**1.6**	60.18	58.25
	24	5.2	2.7	0.2	0.7	56.89	56.73
	28	5.0	1.9	1.4	0.5	53.15	55.07
	32	2.8	0.6	2.0	0.3	49.44	52.34
280	20	$*$**7.4**	$*$**3.8**	$*$**4.7**	$*$**3.1**	62.62	59.11
	24	$*$**6.8**	$*$**3.8**	$*$**2.6**	1.2	59.97	57.10
	28	6.7	2.4	0.5	0.4	56.03	55.63
	32	1.1	0.1	2.0	0.1	52.74	53.54
320	20	$*$**8.4**	$*$**4.5**	$*$**4.8**	$*$**6.7**	64.39	60.45
	24	$*$**7.4**	$*$**5.5**	2.4	$*$**4.8**	62.03	58.85
	28	$*$**6.8**	$*$**3.8**	0.6	$*$**3.5**	58.43	57.20
	32	2.0	1.6	2.2	1.4	54.81	55.43
360	20	$*$**8.7**	$*$**6.1**	$*$**6.3**	0.117	65.62	62.49
	24	$*$**7.8**	$*$**7.7**	$*$**3.8**	0.109	63.54	60.74
	28	$*$**9.0**	$*$**6.9**	1.7	$*$**6.6**	59.96	59.19
	32	$*$**7.5**	$*$**6.6**	1.1	1.2	56.19	57.15

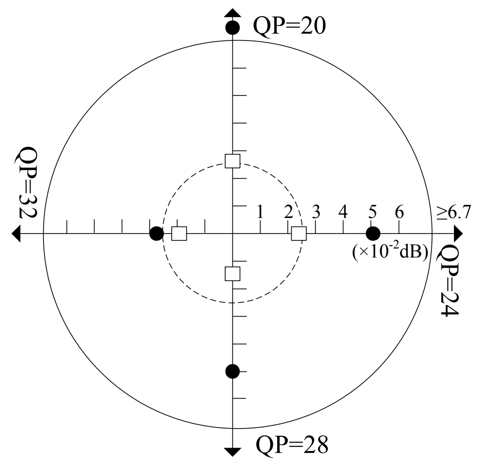
S1: foreman_qcif, S2: mobile_qcif



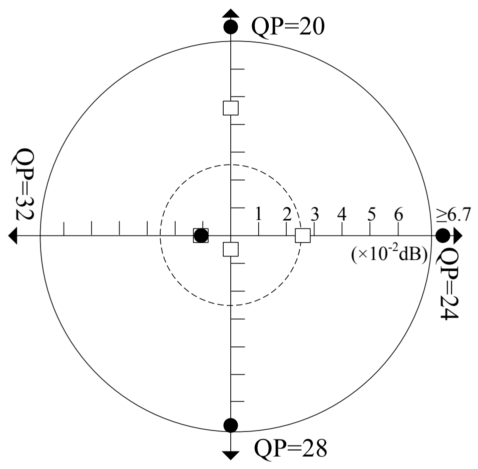
(a) THR_G=160



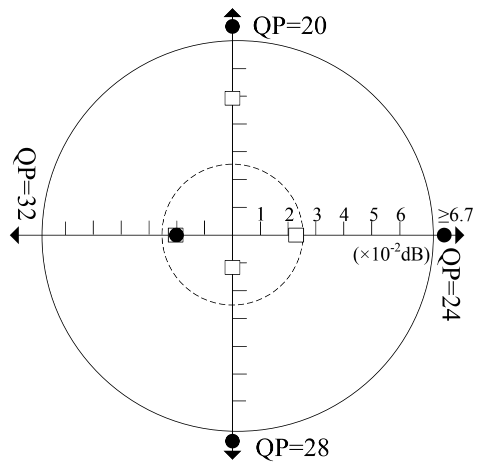
(b) THR_G=200



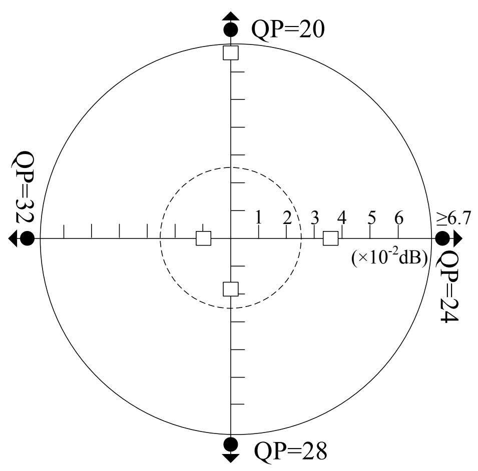
(c) THR_G=240



(d) THR_G=280



(e) THR_G=320



(f) THR_G=360

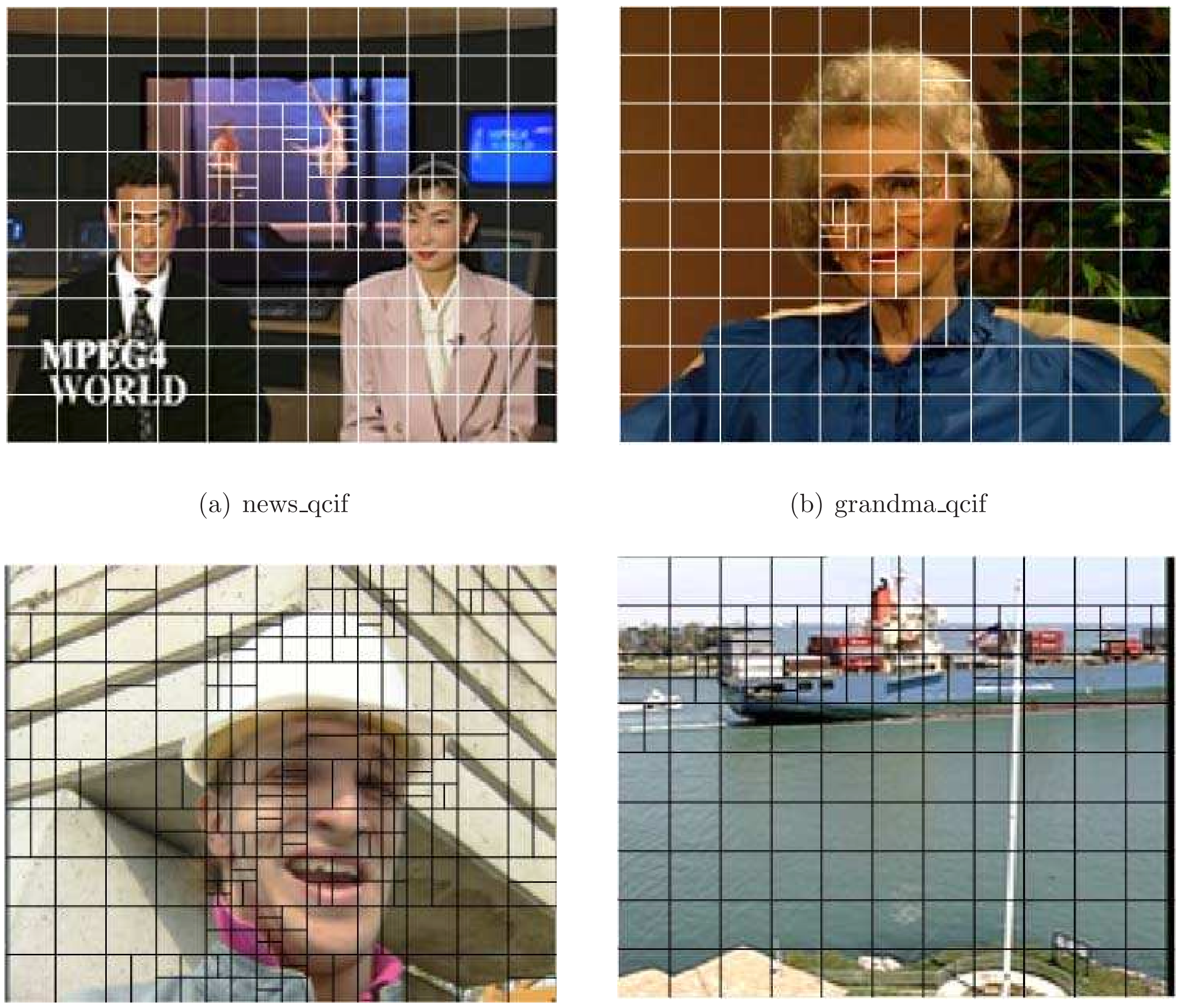
Figure 2.9: Tolerance graph of ‘foreman_qcif’

2.2.4 Similarity-analysis based multiple reference frame elimination

In H.264/AVC based real-time encoding systems, the widely adopted ME algorithm is a full search algorithm that provides regular access to memory, predictable control, and the optimal video quality [20]. In full search algorithm, the sum of the absolute difference (SAD) is selected as a criterion to determine the best position on the reference frame plane. It is obvious that considerable computational resources are wasted because only the MV that has minimum cost is stored while other MVs are discarded at the end of the search process. This wasteful situation becomes more significant if the MRF algorithm is introduced. In fact, since many static parts exist in each sequence, the computation of all search positions is not always necessary. In this section, statistical analysis of typical sequences will be given out and one similarity analysis (SA) based multiple reference frame elimination scheme is proposed.

To simplify the statistical analysis, I select 'foreman_qcif', 'news_qcif', 'grandma_qcif', and 'container_qcif' as four typical sequences and extract the final coding mode for a certain frame. Figure. 2.10 shows the tracing result. The different sizes of black and white boxes overlaid on the images represent different block modes that are chosen after rate distortion (RD) optimization. It is shown that, if a large region has a similar trend of motion, it is more likely to be coded with a large block size. In detail, for sequences such as 'container_qcif' and 'grandma_qcif', there are many temporal stationary background parts which are mostly coded by a large blocks. Rapid moving parts such as the dancer in 'news_qcif' and the facial expression in 'grandma_qcif' are coded in small blocks. Although the lady's suit in 'news_qcif' contains a large amount of edge information, it is also coded by large blocks because it is treated as stationary background. In the case of 'foreman_qcif', even though many background MBs exist in the sequence, many MBs are still coded with small blocks because of the facial expression and the dithering of the vidicon.

In JM software, the hardware-friendly full search algorithm is executed on different search positions. It adopts the spiral searching method, which searches from the center



(a) news_qcif

(b) grandma_qcif

(c) foreman_qcif

(d) container_qcif

Figure 2.10: Coding block sizes of QCIF sequences

to the outside positions. Figure. 2.11 shows an example of spiral-order graph for the first 49 positions. The number in the circle represents the searching order. Position 0 is the motion vector predictor (MVP) point, which is calculated on the basis of neighboring blocks. The block matching process of ME starts from this position. The position that has the minimum cost (MV cost + SAD) is regarded as the best integer search position (BISP) and its corresponding MV is stored. From the previous analysis, it is known that for sequences with a large stationary part, the probability of selecting a big coding mode is very high. Therefore, if an MB with a stationary feature can be detected at an early stage, the ME computation can be reduced because splitting of the MB into small modes

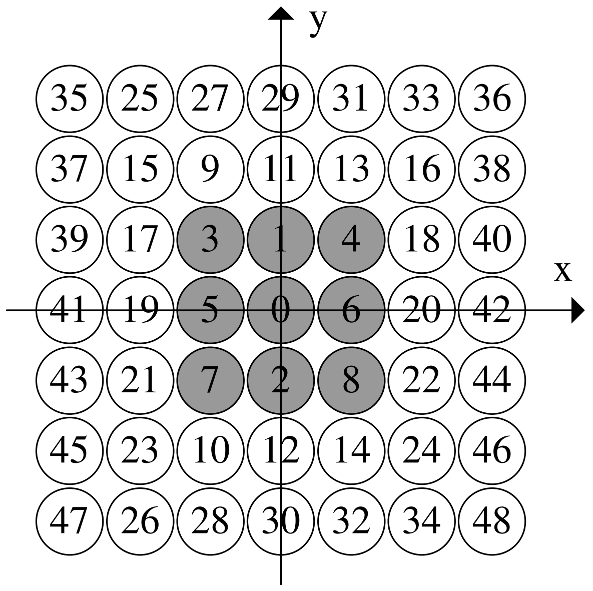
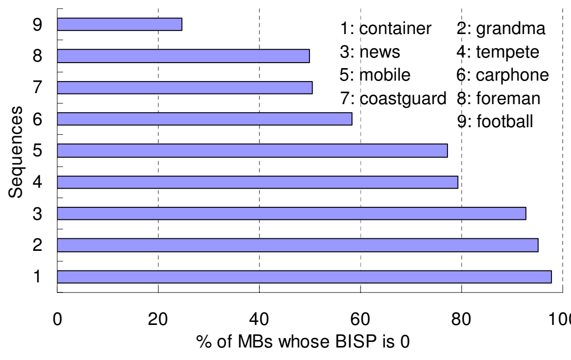


Figure 2.11: Spiral search order

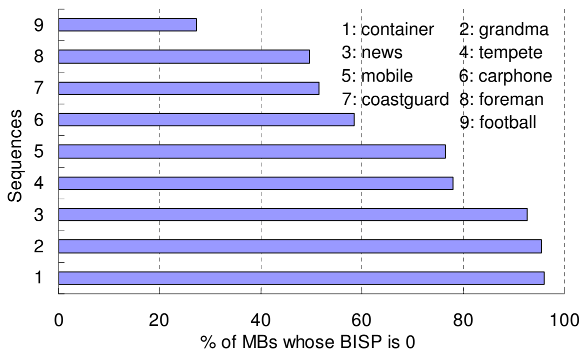
and the MRF technique are both unnecessary for such an MB.

Figure. 2.12 shows the ratio of MBs whose BISP fall into the MVP position (call such MBs as MVP_MBs). Here, I only show histograms for the P16×16 mode and for the first block (block_0) of the P16×8 and P8×16 modes. Since there is considerable similarity among the MRFs, only distribution of MVP_MBs in the previous reference frame is given out. The simulation conditions are listed in Table. 2.2. First, note that the distribution in Fig. 2.12(a) is very similar to those in Fig. 2.12(b) and Fig. 2.12(c). This similarity also occurs among Fig. 2.12(d), Fig. 2.12(e), and Fig. 2.12(f), which means that the features of the sequence are similarly among the P16×16, P16×8, and P8×16 modes. To reduce the computation complexity, I only focus on the P16×16 mode of the first reference frame in my algorithm. Second, for sequences such as 'container_qcif/cif', 'grandma_qcif', and 'news_qcif/cif', many MBs have their best position in the MVP point, which means that the initial MVP is of high accuracy. On the other hand, for sequences such as 'football_qcif/cif' and 'canoa_cif', the percentage of MVP_MBs is low. Thus, the accuracy of the MVP can reflect the characteristics of MBs in different sequences.

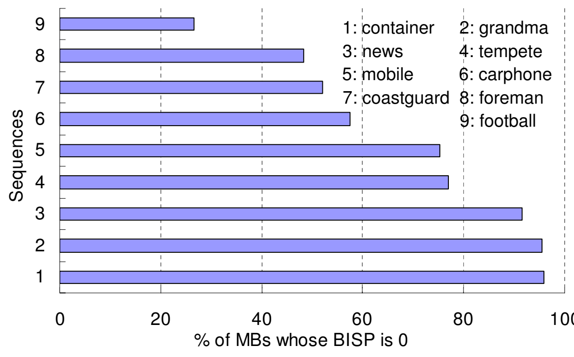
Moreover, even though the current MB selects position 0 as the BISP on the previous reference frame, the final best mode may vary among the SKIP mode, and the P16×16,



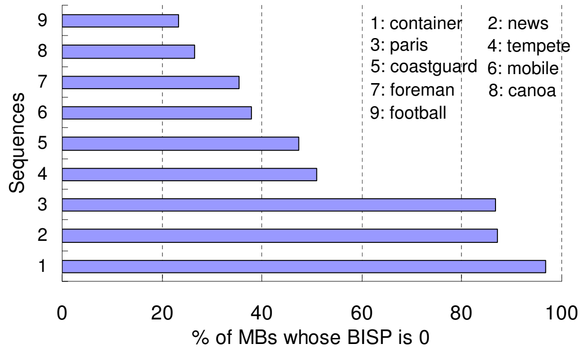
(a) P16×16 mode, QCIF



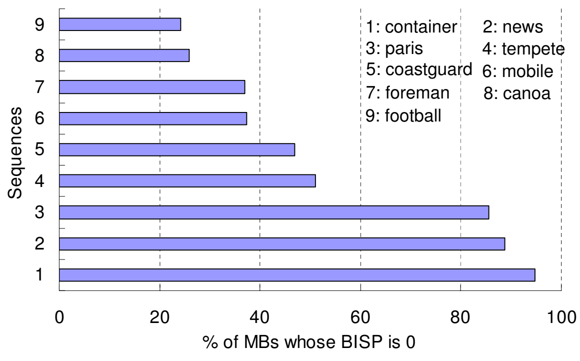
(b) P16×8 mode (block_0), QCIF



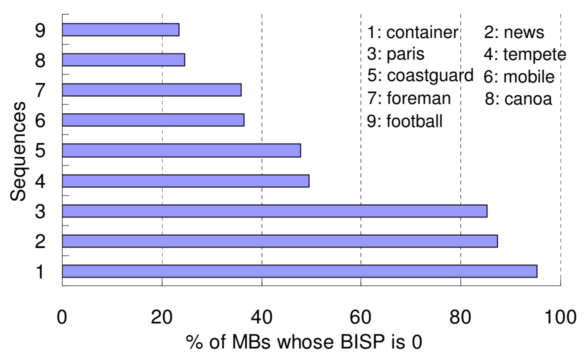
(c) P8×16 mode (block_0), QCIF



(d) P16×16 mode, CIF



(e) P16×8 mode (block_0), CIF



(f) P8×16 mode (block_0), CIF

Figure 2.12: Number of MBs with BISP in MVP

P16×8, P8×16, and P8×8 modes due to more accurate matching under other modes. Here, the inter search modes below 8×8 are also included in P8×8 mode when determining the final best mode in the H.264/AVC standard. Therefore, for MVP_MB, the final best macroblock mode is traced for different quantization parameters (QPs). The experimental result is shown in Fig. 2.13. Here, I list the results of 3 QCIF and 3 CIF sequences as

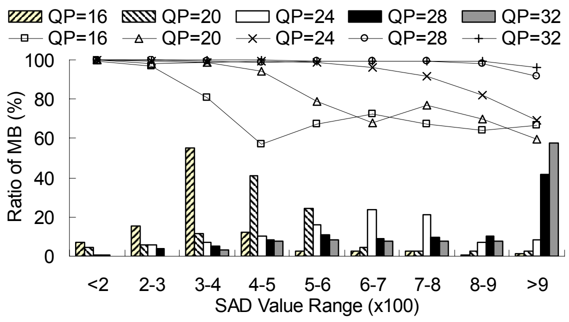
Table 2.2: Simulation conditions for BISP on previous frame

Sequences	QCIF & CIF	QP	20
Search Range	$\pm$ 16 & $\pm$ 24	Frames Encoded	200
etc	no B Slice, CAVLC, 5 Reference Frames RDO is ON, GOP is IPPP		

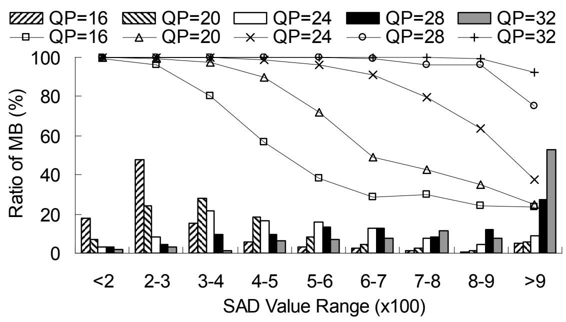
an example. The x axis represents the SAD value range in the MVP position while the y axis is the percentage of MB quantities. Specifically, the histogram reflects the percentage of MVP_MB quantities under different QP and SAD values. For sequences with many stationary parts such as 'container_qcif/cif' and 'news_qcif/cif', many MBs select the MVP as the best search position when QP is small. With increasing SAD value at position 0, the ratio of MVP_MBs decreases; on the other hand, for big QP, this ratio increases rapidly with the SAD value. In the case of sequences with a large amount of motion such as 'football_qcif/cif', the initial MVP is inaccurate and most MVP_MBs have a large SAD value. The curves overlaid on the histogram represent the ratio of MBs whose final coding mode is big mode (SKIP mode, P16×16, P16×8, or P8×16), which means that the MBs are coded in the big mode with less MB splitting. From Fig. 2.13, it is shown that the ratio of MVP_MBs whose final mode is the big mode decrease rapidly in the case of small QP such as 16 and 20. In case of a big QP, this ratio decreases slowly. In fact, for a big QP, after the quantization and reconstruction of reference frames, the reference pixels become more homogeneous with a considerable loss of high-frequency components, which leads to big coding modes after RD.

$$SA\ on\ ref_1\ (SP0\ to\ SP8, P16\times16\ Mode)$$
$$\begin{cases} BISP = 0\ \&\ SAD_{8\times8} \le THR_{SAD}, BMMB \\ otherwise, NMB \end{cases} \tag{2.10}$$

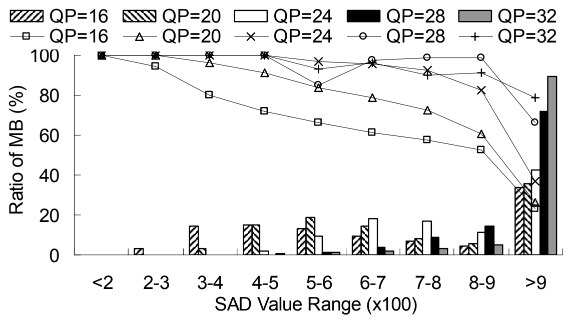
On the basis of the above analysis, the ME and mode decision process can be sped-up for sequences with many stationary parts. I use a threshold (THR_{SAD}) to indicate the degree of similarity of IME in the first reference frame (ref_1) and use it to guide the result of mode decision. To reduce the extra computation that is introduced into the ME process, I only focus on the P16×16 mode in my algorithm. The SA-based big-mode MB



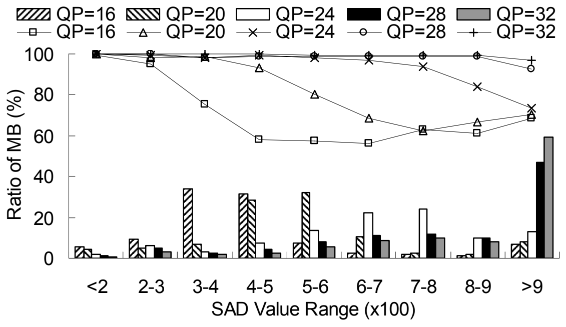
(a) container_qcif



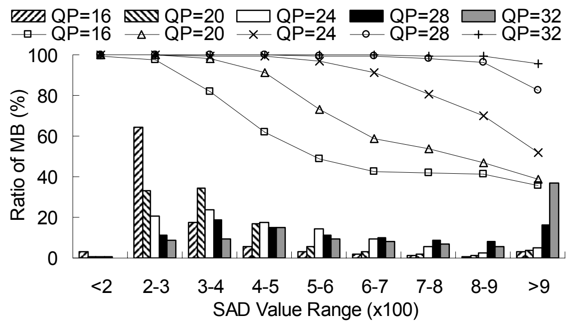
(b) news_qcif



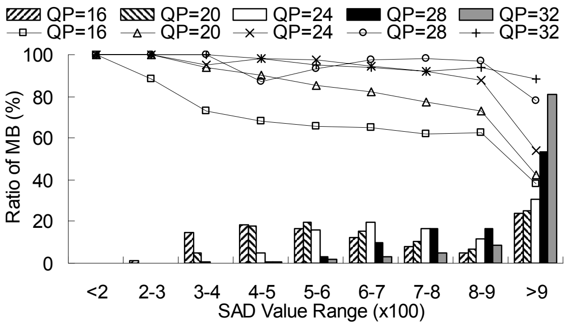
(c) football_qcif



(d) container_cif



(e) news_cif



(f) football_cif

Figure 2.13: Distribution of final best mode

(BMMB) detection scheme is shown in Eq. 2.10. It means that during the IME process, the SA is performed on the 9 central positions of ref_1 (the gray circles in Fig. 2.11). The MB is defined as a BMMB if its BISP at these 9 positions is 0 and all four of its 8×8 sized SAD ($SAD_{8\times8}$) values are within THR_{SAD}; otherwise it is treated as a normal MB (NMB). For a BMMB, the IME process is early terminated after IME of P16×16, P16×8, and P8×16 modes for the 9 central positions of the previous frame; and only big modes are enabled during mode decision stage. On the basis of experimental results, the threshold is defined according to the QP value. In detail, when QP is less than 24, THR_{SAD} is set as 6×QP, otherwise it is set as 7×QP.

2.3 Hardware oriented search range adjustment

In the H.264/AVC motion estimation, search range is another important factor which influence the computation complexity greatly. For example, when search range (SR) is decided, the number of search points can be calculated based on Eq. 2.11, where SP_{num} is the number of search point and SR is the dedicated search range. So, when SR equals 24, the SP_{num} will become 2401 which is a quite large number for hardware engine. Therefore, hardware oriented search range adjustment scheme is needed.

$$SP_{num} = (2SR + 1) \times (2SR + 1) \tag{2.11}$$

2.3.1 Motion feature based search range adjustment

In H.264/AVC based encoding system, different sequences have different features; a large SR is not necessary for all sequences. Figure. 2.14 shows two RD curves under different SR. It is shown that changing the SR does not cause significant video quality loss in 'foreman_qcif'. On the other hand, the quality degradation in the case of 'football_qcif' is very obvious, which means that a big SR is necessary for 'football_qcif'.

For MB with different motion (small or large motion), complexity reduction can be achieved based on the motion feature analysis. Since different type of sequences may have different best integer search point (BISP) distributions, I trace BISP result on each reference frames under 16 × 16 mode, as shown in Table. 2.4. The simulation conditions are shown in Table. 2.3.

Firstly, it is shown that BISP distribution of the same sequence among different reference frames demonstrates the same motion feature. For example, in 'container_qcif', the BISP distribution in first reference frame shows that many BISPs are located within centering 25 positions, the situation of which is almost the same with BISP distributions

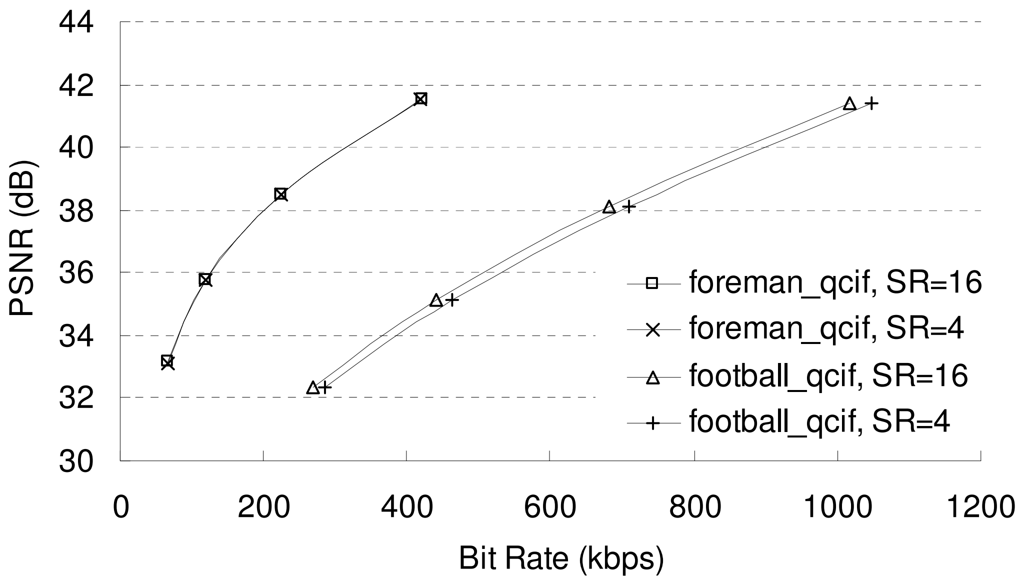


Figure 2.14: Impact of search range to video quality

in other four reference frames. So the BISP distribution in first reference frame can represent the motion feature of this MB and I only focus on the first reference frame in my search range adjustment scheme.

Secondly, Table. 2.4 show that the BISP distribution of 'football_qcif' is different from other 4 sequences. The BISPs located between 169th and 1088th position are much more than other 4 sequences, which shows its large motion trend. It also implies that the initial motion vector predictor (MVP) is far from accurate for 'football_qcif'.

Thirdly, for sequences except 'football_qcif', large proportion of BISPs are located within the inner 25 position, which shows the small motion trend. Comparing 'foreman_qcif' and 'carphone_qcif' with 'container_qcif' and 'news_qcif', the proportion of BISPs that are located in position 0 is much smaller in 'foreman_qcif' and 'carphone_qcif'. It means that there are many static background MBs in 'container_qcif' and 'news_qcif' and MVP is of high accuracy for motion estimation in these sequences.

Therefore, from the statistic analysis of typical sequences, it is shown that the BISP location can reflect the motion feature of the MB. For MB with big motion, large search range is necessary to keep the overall best search point within available search range. On the other hand, for MB which shows static or small motion feature, many redundant

Table 2.3: Simulation conditions for BISP on five reference frames

QP	24	Sequences	qcif
Search Range	$\pm$ 16	Frames Encoded	100
etc	no B Slice, CAVLC, 5 Reference Frames RDO is ON, GOP is IPPP		

search points exist in reference frames, which occupy much computation during motion estimation.

There exist many MV oriented search range adjustment algorithms [15][14]. However, implementing these algorithms will cause huge storage for MVs of former frames, which deteriorates the efficiency of system. Instead of analyzing MV, I use the motion feature which is extracted from IME process to guide search range adjustment process. Since small blocks contain less texture and are prone to be trapped into local optimum position, I just do motion feature analysis on 16×16 mode. Based on the above analysis, the proposed motion feature analysis based search range adjustment scheme concludes 2 key steps:

1. In the first reference frame, VBSME in full search range is executed. The initial search range (SR_{JM}) for QCIF, CIF and HDTV720p sequences are ± 16, ± 24 and ± 64 in both width and height.

2. After IME with first reference frame on 16×16 mode, check BISP. If BISP is smaller than $\pm 1/8$ SR_{JM}, which means that MVP's accuracy is high, the search range is adjusted to $\pm 1/8$ original search range in both width and height for rest ME process. With the decrease of search range on rest reference frames, much computation is saved.

2.3.2 Recursive 6-ring search range adjustment

On the basis of statistical analysis of previous sub-section, the search range can be further adjusted for MBs of different motion feature on the first reference plane. In my algorithm, I divide the SR into 6 rings, as shown in Fig. 2.15. The scale of search range in the x direction and y direction is $SR_{jm}/16$, where SR_{jm} is the general SR in JM software (16 for QCIF and 24 for CIF). Since small inter modes tend to fall into local minimum, I only apply SR adjustment scheme based on BISP in the 16×16 mode. The proposed scheme

Table 2.4: BISP Distribution on 1st to 5th Reference Frame

IME on 1st Reference Frame

BISP	Seq 1	Seq 2	Seq 3	Seq 4	Seq 5
0	4961	5556	9434	9263	2040
1∼8	3749	3095	159	433	2996
9∼24	564	436	38	34	1090
25∼48	204	235	64	17	706
49∼80	100	107	7	10	454
81∼120	53	74	7	6	388
121∼168	43	71	5	4	321
169∼1088	127	227	87	34	1806

IME on 2nd Reference Frame

BISP	Seq 1	Seq 2	Seq 3	Seq 4	Seq 5
0	2827	4008	9183	8511	654
1∼8	4624	3453	261	950	632
9∼24	1375	1051	50	106	691
25∼48	439	499	85	35	857
49∼80	175	191	13	16	748
81∼120	79	135	9	14	646
121∼168	43	82	8	11	798
169∼1088	140	283	93	59	4674

IME on 3rd Reference Frame

BISP	Seq 1	Seq 2	Seq 3	Seq 4	Seq 5
0	1972	3510	8826	7916	471
1∼8	3612	3151	473	1191	369
9∼24	1997	1247	63	269	390
25∼48	985	697	102	65	471
49∼80	436	342	12	27	510
81∼120	236	192	18	25	551
121∼168	115	116	10	19	599
169∼1088	250	348	99	91	6242

IME on 4th Reference Frame

BISP	Seq 1	Seq 2	Seq 3	Seq 4	Seq 5
0	1345	3141	8102	7563	391
1∼8	2994	2942	1036	1118	281
9∼24	1865	1321	88	459	297
25∼48	1341	666	127	140	381
49∼80	799	446	17	51	375
81∼120	405	303	19	34	419
121∼168	277	213	13	16	485
169∼1088	478	472	102	123	6875

IME on 5th Reference Frame

BISP	Seq 1	Seq 2	Seq 3	Seq 4	Seq 5
0	1073	2917	6936	7389	335
1∼8	2547	2604	2014	977	238
9∼24	1734	1412	127	469	232
25∼48	1233	753	147	281	312
49∼80	976	373	21	73	356
81∼120	600	358	26	49	354
121∼168	397	275	18	14	427
169∼1088	845	713	116	153	7151

Seq 1: foreman, Seq 2: carphone, Seq 3: container

Seq 4: news, Seq 5: football

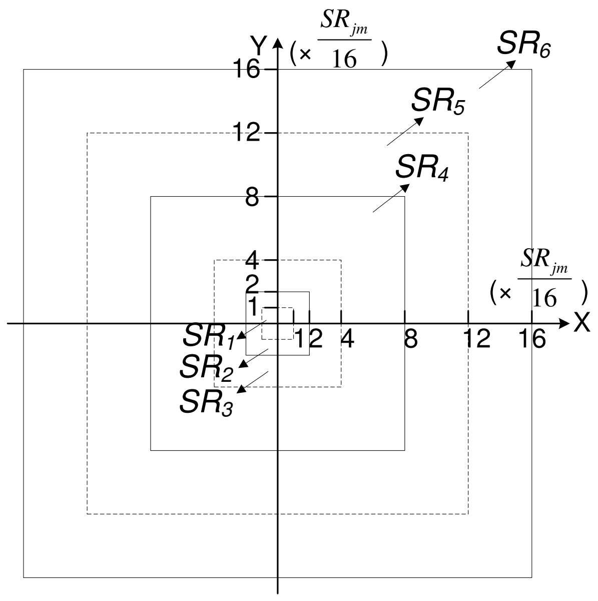


Figure 2.15: 6-Ring search range adjustment

is given in Eq. 2.12. After IME on the m_{th} reference frame (ref_m), the BISP of this frame ($BISP(m)$) in the 16×16 mode is analyzed. If it is between the values of SP_{num} for SR_i and SR_{i+1}, then the SR in the $(m+1)_{th}$ reference frame ($SR(m+1)$) is changed to SR_{i+2}. If the $BISP(m)$ value surpasses the SP_{num} of SR_5 in Fig. 2.15, then original JM SR is used for next ME process. The proposed search range scheme adaptively shrinks the SR for small-motion MBs. For normal or big motion MB, large SR value is still available to keep the best MV.

$$\begin{gathered} After\ IME\ on\ ref_m,\ m \in [1,4] \\ \begin{cases} (2SR_i+1)^2 \le BISP(m) < (2SR_{i+1}+1)^2, \\ \qquad SR(m+1) = SR_{i+2},\ i \in [0,4] \\ BISP(m) > (2SR_5+1)^2, \\ \qquad SR(m+1) = SR_{jm} \end{cases} \end{gathered} \tag{2.12}$$

2.4 Pixel difference based adaptive sub-sampling

In hardware application, sub-sampling is widely used to release computation complexity and achieve compact hardware architecture. The concept of sub-sampling in ME is to use part of pixels to represent the whole MB so that computation reduction can be achieved. In [27], it also adopts direct half sub-sampling technique and 50% computation is saved. However, the sub-sampling will also introduce video quality loss because the further sampling on the pixels will intensify the aliasing problem [21] caused by video sensor.

When sub-sampling is applied on $s_t(x_n)$, the related Fourier transform $S_t(j\omega_x)$ will be derived to $Y_t(j\omega_x)$, as shown in Eq. 2.13, where $S_t(j\omega_x)$ is the Fourier transform of $s_t(x_n)$. It means that the original ideal cut-off frequency of band limit low pass filter is extended, which will result in further entangling of frequency components. The inevitable aliasing problem becomes even worse. So, the direct sub-sampling in both horizontal and vertical direction (quarter sub-sampling) is a very risky decision. Another conclusion that can be obtained from Eq. 2.13 is that the degree of aliasing problem may vary greatly based on different ω_x. Since the IME engine handles the image MB by MB, the frequency feature of MB will determine the result of direct sub-sampling.

$$Y_t(j\omega_x) = \frac{1}{2}[S_t(j\frac{\omega_x}{2}) + S_t(j(\frac{\omega_x}{2} - \pi))] \tag{2.13}$$

The difference among the pixels is a direct reflection of the spread of frequency spectrum. With the frequency feature of current MB, the block matching process which is usually based on SAD (sum of absolute difference) calculation in hardware [28] can be simplified. For example, if the pixels' values within one MB are close to each other, then this MB is a homogeneous MB and half or quarter sub-sampling technique can be adopted to achieve computation reduction. On the other hand, if big pixel difference occurs within one MB, then much high frequency ingredient exists in this MB. So full pixel pattern has to be used for block matching in order to ensure precise estimation.

Figure. 2.16 is the video quality comparisons based on JM software [29]. It is shown that, the direct quarter sub-sampling (ds) on 'foreman_qcif' will averagely cause 0.3 dB video quality loss compared with hardware friendly full search algorithm. In the case of 'container_qcif', the quality degradation is negligible. Through observing the feature of these two sequences, it is also obvious that MBs in 'container_qcif' are much more homogeneous. In detail, the pixels in MBs of 'container_qcif' are very similar to each other so that sub-sampling will not cause great influence on the block matching process

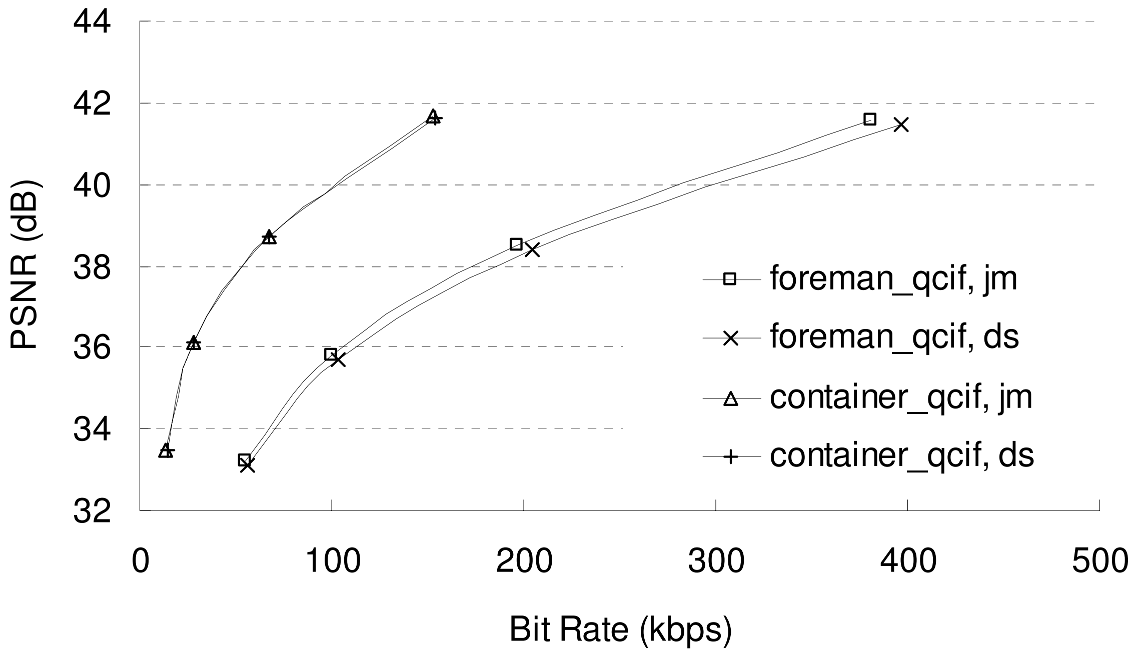


Figure 2.16: Impact of direct sub-sampling

of these MBs. Thus, classifying the MBs into sub-sampling allergic MB and sub-sampling insensitive MB is of great importance.

In this dissertation, I use pixel difference analysis to obtain the feature of MB in a hardware friendly way. Figure. 2.17 shows three hardware friendly sub-sampling patterns. Pattern 1 is the quarter sub-sampling pattern which uses one pixel (black point) to represent its neighboring three pixels. Pattern 2 and pattern 3 are horizontal and vertical half sub-sampling patterns. Each pixel in these two patterns is selected to represent its horizontal or vertical neighboring pixel respectively. The pixel difference analysis method is shown in Eq. 2.14 and Eq. 2.15, where $P(i,j)$ is the pixel value in position (i,j). It means that during load of current MB, the horizontal pixel difference $(PD_h(i,j))$ and vertical pixel difference $(PD_v(i,j))$ of this MB are examined. Since only horizontal, vertical, and diagonal neighbors of the pixel are used to get the horizontal and vertical pixel difference, the extra computation is small. Figure. 2.18 is the flow chart of proposed adaptive sub-sampling method. If $PD_h(i,j)$ of each position within one MB is smaller than threshold THR_{PD}, then horizontal half sub-sampling on this MB (pattern 2 in Fig. 2.17) is applied. In this way, pixel information in vertical direction is preserved. On the other

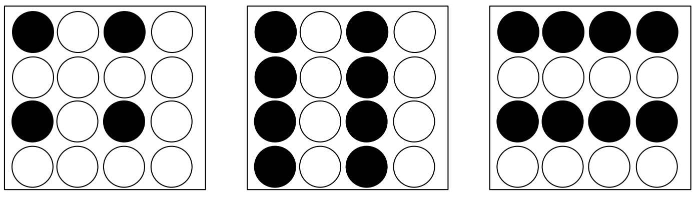


Figure 2.17: Three sub-sampling patterns

hand, horizontal pixel information is kept by using vertical half sub-sampling (pattern 3 in Fig. 2.17) if each $PD_v(i, j)$ is within threshold THR_{PD}. When both $PD_h(i, j)$ and $PD_v(i, j)$ of each position are within THR_{PD}, then quarter sub-sampling (pattern 1 in Fig. 2.17) is applied on this MB. From exhaustive experiments, I finally set THR_{PD} as $4 \times QP$ (quantization parameter) to achieve much computation reduction according to different QP values.

$$\begin{aligned} PD_h(i,j) = &|P(i,j) + P(i,j+1) \\ &- P(i+1,j) - P(i+1,j+1)| \end{aligned} \tag{2.14}$$

$$\begin{aligned} PD_v(i,j) = &|P(i,j) + P(i+1,j) \\ &- P(i,j+1) - P(i+1,j+1)| \end{aligned} \tag{2.15}$$

2.5 Experiments, comparison and analysis

In order to verify the effectiveness of proposed fast ME algorithm, I combine all the schemes together and apply my algorithm on 8 QCIF, 8 CIF and 4 HDTV720p format sequences by using JM 11.0 software. The QP values are 20, 24, 28, and 32. Since my algorithm targets at complexity reduction for hardware, the comparison is first based on algorithm adopted in existing hardware engine. For H.264/AVC hardwired engine, factors such as data reuse, hardware utilization and predictable control are critical ones to the design. The most widely adopted algorithm for motion estimation engine is full search algorithm [17][18]. So, I first implement my schemes in JM full search algorithm. The simulation conditions are shown as follows.

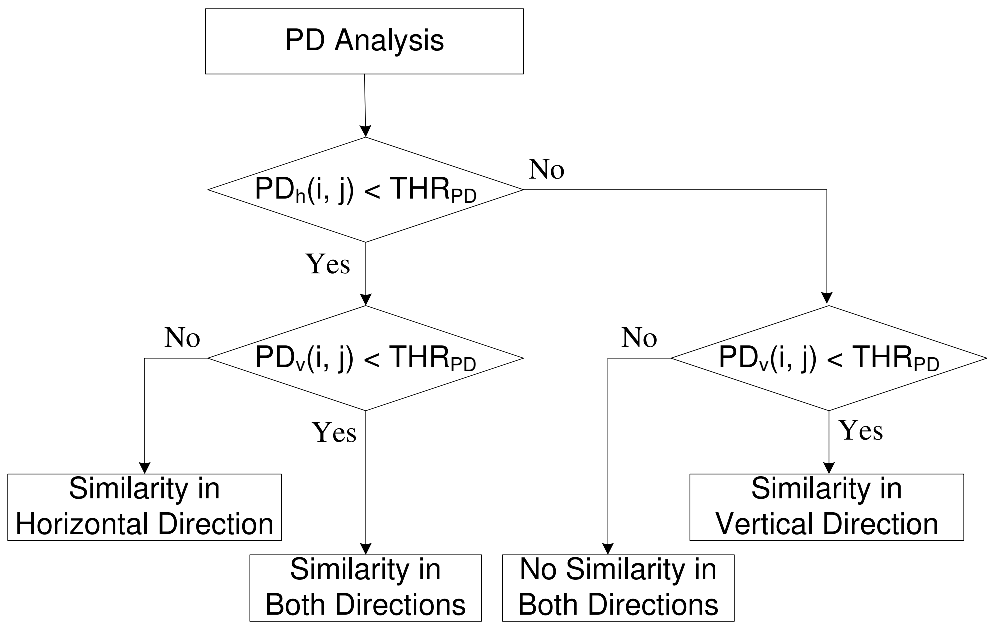

PD Analysis
PDh(i, j) < THRPD
No
Yes
No
PDv(i, j) < THRPD
Yes
No
PDv(i, j) < THRPD
Yes
Similarity in Horizontal Direction
Similarity in Both Directions
No Similarity in Both Directions
Similarity in Vertical Direction

Figure 2.18: Flow chart of adaptive sub-sampling

. GOP is IPPP

. Encoding 200 frames for QCIF/CIF, 100 frames for HDTV720p

. Reference Frame Number is 5

. Search Range is ±16, ±24 and ±64 for QCIF, CIF and HDTV720p

. RDO is On

In my proposed algorithm, the ME's sub-sampling pattern is determined by pixel difference analysis. Table. 2.5 shows the ratio of MBs that are classified as homogeneous MBs and adaptive sub-sampling is applied on these MBs. Here, I only give out pixel difference analysis result of MBs whose PD_v and PD_h are both within threshold as an example. It means that the quarter sub-sampling will be applied on these MBs. Table. 2.5 shows that for high frequency abundant sequences such as 'mobile_cif' and 'tempete_qcif', the homogeneous MB ratio is not high (averagely 10.50%). However, for sequences like 'grandma_qcif', 'container_cif','city_720p' and 'crew_720p', many homogeneous MBs exist (averagely 74.98%), so that adaptive sub-sampling scheme contributes much to these sequences.

Secondly, the MRF elimination ratio is shown in Table. 2.6. It is shown that for

Table 2.5: Homo MB Ratio (%) for 1/4 Subsampling

QP	20	24	28	32	Average
foreman_qcif	31.28	36.99	48.38	66.86	45.88
mobile_qcif	5.58	8.97	16.33	25.36	11.72
container_qcif	38.80	40.23	41.42	42.30	40.69
grandma_qcif	64.42	70.84	76.89	82.56	56.38
news_qcif	33.35	38.82	44.06	48.28	41.13
tempete_qcif	6.65	9.38	13.40	18.97	12.10
coastguard_qcif	29.02	38.42	45.18	51.44	41.02
carphone_qcif	44.53	49.63	54.02	62.06	52.56
stefan_cif	25.15	28.75	33.22	39.43	31.64
mobile_cif	5.63	7.55	10.07	12.40	8.91
football_cif	63.94	69.02	74.68	82.55	72.55
container_cif	52.94	55.67	56.49	57.77	55.72
news_cif	57.99	63.47	68.87	73.12	65.86
tempete_cif	22.20	29.77	38.80	49.40	35.04
coastguard_cif	38.68	49.36	58.75	66.03	53.20
paris_cif	23.92	26.14	28.71	31.34	27.53
parkrun_720p	24.24	32.93	39.37	44.68	49.66
mobcal_720p	35.98	44.59	54.27	63.32	49.54
city_720p	61.89	73.85	82.04	87.90	76.42
harbor_720p	50.58	64.33	76.16	85.42	69.12

sequences with large proportion of static part such as 'container_qcif/cif', 'grandma_qcif' and 'news_qcif/cif' (averagely 55.08%), much complexity can be eliminated by our MRF elimination algorithm. In case of 'mobile_qcif/cif' and 'tempete_qcif/cif', the ratio is very small (averagely 7.30%) because the motion on the edge abundant background deteriorates our algorithm greatly.

Thirdly, I also test the search range reduction scheme on different sequences individually. The experimental results are shown in Table. 2.7. Here, I only list the ratio of small

Table 2.6: Ratio (%) of MB with MRF Elimination

QP	20	24	28	32	Average
foreman_qcif	12.25	16.63	18.95	23.26	17.77
mobile_qcif	3.13	4.66	5.64	6.43	4.97
container_qcif	50.01	51.14	50.91	50.53	50.65
grandma_qcif	57.20	61.07	56.15	58.74	58.29
news_qcif	54.67	56.57	42.62	35.41	47.32
tempete_qcif	4.62	5.42	5.64	6.51	5.55
coastguard_qcif	14.43	21.62	28.53	33.53	24.53
carphone_qcif	33.43	38.85	40.04	40.71	38.26
stefan_cif	20.63	22.03	22.58	23.67	22.23
mobile_cif	3.35	3.94	4.30	4.90	4.12
football_cif	40.88	47.77	52.56	56.28	49.37
container_cif	55.93	58.25	56.78	56.45	56.85
news_cif	64.54	67.05	60.89	56.80	62.32
tempete_cif	11.59	13.48	15.37	17.92	14.59
coastguard_cif	14.43	21.62	28.53	33.53	24.53
paris_cif	34.57	34.31	27.91	26.48	30.82
parkrun_720p	11.02	17.64	23.86	29.15	20.42
mobcal_720p	21.93	24.94	28.47	31.60	26.74
city_720p	30.73	38.38	45.57	58.90	43.40
harbor_720p	11.76	17.64	24.83	32.79	21.76

motion MB, which means that MB which adopts recursive search range adjustment is not included. It is shown that, for most small motion sequences such as 'news_qcif/cif', 'mobile_qcif', 'coastguard_qcif/cif', 'paris_cif' and 'harbor_720p', about 97.71% MBs adopt search range adjustment through our motion feature analysis. For sequences such as 'foreman_qcif' and 'carphone_qcif', the ratio decreases slightly (averagely 92.11%), because the motion in these sequences is a little more severe than former sequences. In case of 'football_cif' and 'stefan_cif', since there are many large motion MBs, our motion

Table 2.7: Ratio (%) of MB with Small Range Constraint

QP	20	24	28	32	Average
foreman_qcif	94.51	93.92	92.94	91.31	93.17
mobile_qcif	99.41	99.27	99.26	98.44	99.10
container_qcif	98.21	96.02	93.55	91.48	94.82
grandma_qcif	96.73	97.09	96.67	96.19	96.67
news_qcif	98.49	96.96	97.87	96.42	97.44
tempete_qcif	96.33	96.10	96.11	95.93	96.12
coastguard_qcif	99.17	98.77	97.19	93.95	97.27
carphone_qcif	91.22	91.89	91.37	89.66	91.04
stefan_cif	80.46	82.80	84.39	85.37	83.26
mobile_cif	96.18	96.19	96.25	96.44	96.27
football_cif	67.24	67.93	67.29	64.75	66.80
container_cif	98.32	96.78	95.04	94.01	96.04
news_cif	98.49	96.96	97.87	96.42	97.44
tempete_cif	96.33	96.10	96.11	95.93	96.12
coastguard_cif	99.17	98.77	97.19	93.95	97.27
paris_cif	98.36	97.89	97.78	96.61	97.66
parkrun_720p	95.36	96.23	97.07	97.67	96.58
mobcal_720p	93.69	95.78	96.58	96.70	95.69
city_720p	97.03	97.05	96.76	96.06	96.73
harbor_720p	97.92	97.95	97.85	97.54	97.82

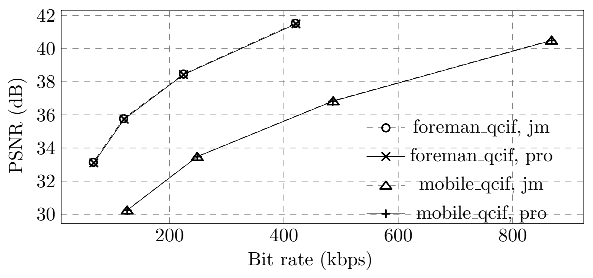
feature analysis scheme which is based on the accuracy of MVP reflects the necessity of large search range for these sequences. So the average ratio of MBs that adopt search range adjustment in these sequences is only about 75.03%.

The overall fast ME algorithm which combines adaptive sub-sampling, MRF elimination and search range adjustment schemes are tested. Figure. 2.19 and Fig. 2.20 are the rate distortion (RD) curve comparisons between proposed algorithm and JM full search algorithm which has the best video quality. Since the difference between two RD curves is very trivial, I use BDBR (Bjøntegaard Delta BitRate) and BDPSNR (Bjøntegaard PSNR) [30] which are respectively average difference of bit-rate and PSNR between curves of original algorithm and proposed algorithm, to evaluate video quality. The sign (+) in BDBR represents bit rate gain, and (−) sign in BDPSNR indicate the quality degradation. The BDBR and BDPSNR of each sequence are listed in Table. 2.8. It is shown that the maximum BDBR and BDPSNR differences among all sequences appear in 'stefan_cif'. About +1.561% BDBR and −0.224dB BDPSNR can be observed in 'stefan_cif'. Averagely, the quality degradation and bit-rate increase are very trivial compared with original full search algorithm.

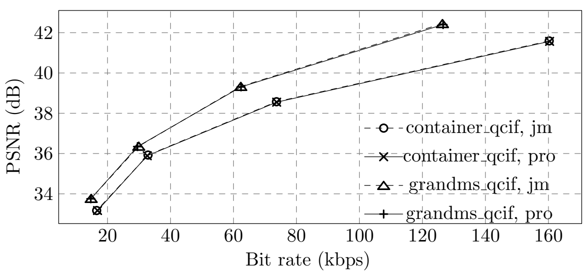
The ME time reduction (MET_R) under each QP is calculated based on Eq. 2.16, where MET_{JM} and MET_{pro} represent the ME time of original JM full search algorithm and proposed algorithm respectively. The experimental result is also shown in Table. 2.9. By using proposed fast ME algorithm, 83.69% to 95.72% ME time can be reduced compared with full search algorithm. Averagely, the proposed hardware oriented algorithm can achieve 88.53% reduction of ME time among all these sequences.

$$MET_R = \frac{MET_{JM} - MET_{pro}}{MET_{JM}} \times 100 \tag{2.16}$$

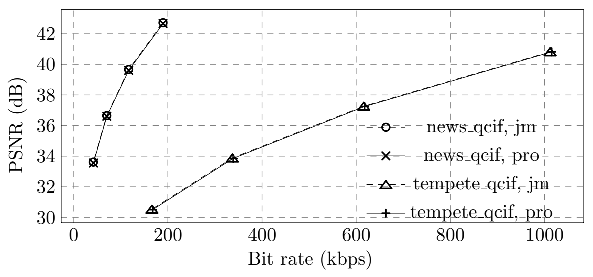
Furthermore, the proposed schemes are also orthogonal to other software oriented fast algorithms and can be combined together to achieve further complexity reduction. Instead of making comparisons with various software oriented algorithms, which are either impractical or inefficient for hardware flow, I only focus on UMHexagon search [13] which is famous among software oriented algorithms and already adopted by JM software. The UMHexagon method [13] is superior in speeding up the ME process and can achieve almost the same video quality as full search algorithm. Here, the proposed algorithm is embedded into UMHexagon search to show the impact of my algorithm. The pixel difference analysis will determine the number of block matching pixels for each search. The MRF elimination algorithm works together with UMHexagon's early termination. For search range adjustment scheme, I keep my algorithm together with dynamic search range algorithm in UMHexagon search, as shown in Eq. 2.17. Specifically, after IME on the first reference frame, if the x and y coordinates of 16×16's motion vector are both within



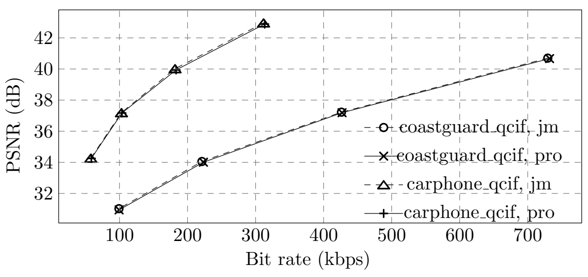
(a) foreman_qcif and mobile_qcif



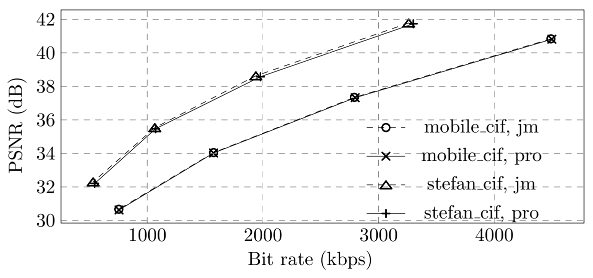
(b) container_qcif and grandma_qcif



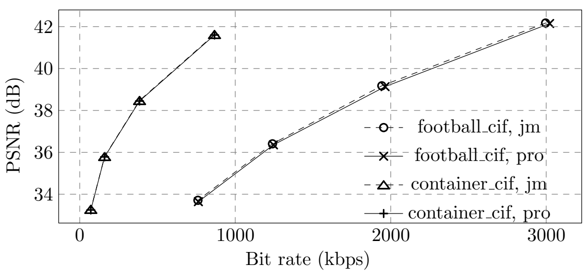
(c) news_qcif and tempete_qcif



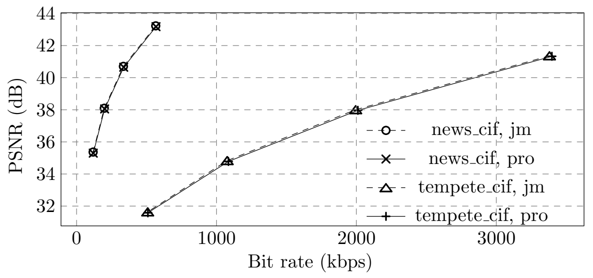
(d) coastguard_qcif and carphone_qcif



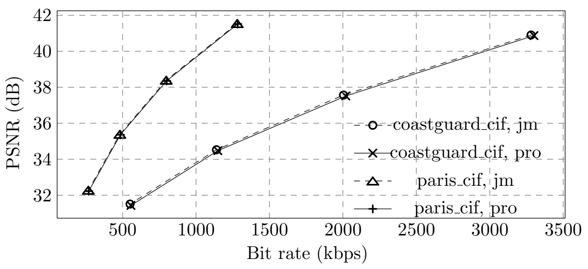
(e) mobile_cif and stefan_cif



(f) football_cif and container_cif



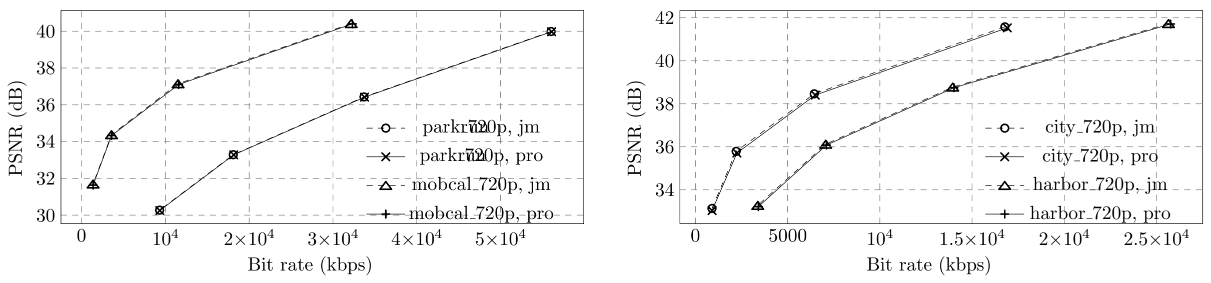
(g) news_cif and tempete_cif



(h) coastguard_cif and paris_cif

Figure 2.19: Comparison of QCIF and CIF RD Curves

$\pm 1/8\ SR_{JM}$, then current MB is defined as a small motion MB; and the search range (pro_SR) adjustment scheme on the following ME process (the SR of rest ME process is set as $\pm 1/8\ SR_{JM}$); otherwise, the original dynamic search range ($dynamic\_SR$) scheme



(a) parkrun_720p and mobcal_720p

(b) city_720p and harbor_720p

Figure 2.20: Comparison of 720p RD Curves

is used. Since dynamic search range exists in UMHexagon search, the recursive search range adjustment scheme is disabled. Based on the same simulation conditions described above, the video quality comparison between proposed algorithm and UMHexagon is given out in Table. 2.10. The speedup ratio γ is defined as MET_{UMHS}/MET_{pro} , where MET_{UMHS} is the ME time consumed by UMHexagon search. Table. 2.11 is the speedup ratio under four QPs. It is shown that the proposed algorithm keeps almost the same video quality (worst case BDBR and BDPSNR is +1.554% and −0.114 dB in football_cif and carphone_qcif) as UMHexagon search while can achieve speedup ratio up to 2.73 of the fast algorithm among all these sequences.

$$\begin{cases} SR = pro\_SR, & small\, motion\, MB \\ SR = dynamic\_SR, & otherwise \end{cases} \tag{2.17}$$

As for hardwired video coding system, the pixel difference analysis on current MB only acts as a pre-process before IME, the adaptive sub-sampling scheme is a hardware friendly proposal, which helps to save clock cycles and power in the architecture level. For MRF elimination scheme and search range adjustment, since I do not rely on the relationship of MVs in former reference frames, they are also hardware oriented schemes which target at reducing clock cycle and saving power in the system level. In fact, the proposed fast ME algorithm keeps the original full search data flow and the existing architectures such as propagate partial SAD [17] [28] and SAD Tree [17] can realize my algorithm with some

Table 2.8: Quality Comparison with Full Search

	BDBR (%)	BDPSNR (dB)
foreman_qcif	+1.023	-0.092
mobile_qcif	+0.369	-0.020
grandma_qcif	+0.115	-0.004
container_qcif	+0.669	-0.025
news_qcif	+0.134	-0.068
tempete_qcif	+0.855	-0.048
coastguard_qcif	+1.024	-0.080
carphone_qcif	+1.001	-0.143
stefan_cif	+1.561	-0.224
mobile_cif	+0.577	-0.032
football_cif	+1.276	-0.121
container_cif	+0.623	-0.021
news_cif	+1.264	-0.108
tempete_cif	+1.118	-0.087
coastguard_cif	+1.195	-0.102
paris_cif	+1.011	-0.065
parkrun_720p	+0.438	-0.023
mobcal_720p	+1.235	-0.060
city_720p	+1.437	-0.006
harbor_720p	+1.134	-0.117

optimization in the control module.

$$\begin{aligned} PE_{idle}\_ratio =& 1 - 0.5 \times R(hss + vss) \\ & - 0.25 \times R(qss) - 1.0 \times R(nss) \end{aligned} \tag{2.18}$$

$$\begin{aligned} clk\_sav\_MRF =& MB\_num \times R(MRF\_skip) \\ & \times (Ref\_num - 1) \times SP\_num \end{aligned} \tag{2.19}$$

Table 2.9: ME Time Reduction Ratio with Full Search(%)

QP	20	24	28	32	Average
foreman_qcif	87.37	87.74	87.52	87.52	87.49
mobile_qcif	87.48	87.32	87.21	86.68	87.23
grandma_qcif	92.65	92.32	91.97	91.59	92.36
container_qcif	92.65	92.31	91.97	91.59	92.36
news_qcif	90.56	90.37	88.82	87.64	89.35
tempete_qcif	87.37	87.04	86.94	86.76	87.03
coastguard_qcif	91.31	91.70	91.37	90.92	91.33
carphone_qcif	88.00	88.19	87.56	87.47	87.81
stefan_cif	85.22	85.57	85.78	86.37	85.62
mobile_cif	90.40	90.23	89.93	89.59	90.12
football_cif	83.69	85.39	86.55	87.30	84.85
container_cif	95.59	95.06	94.19	93.15	94.80
news_cif	94.01	93.73	92.84	92.05	93.16
tempete_cif	90.18	90.45	90.68	90.76	90.52
coastguard_cif	93.77	94.06	94.09	93.69	93.90
paris_cif	91.71	91.50	90.91	90.51	91.16
parkrun_720p	93.08	93.73	94.23	94.54	93.89
mobcal_720p	93.00	93.75	94.34	94.56	93.91
city_720p	94.81	95.16	95.43	95.56	95.24
harbor_720p	95.18	95.49	95.66	95.72	95.51

$$\begin{aligned} clk\_sav\_SR =& MB\_num \times R(SR\_adj) \times (Ref\_num - 1) \\ & \times [SP\_num - (\frac{SR_{JM}}{4} + 1)^2] \end{aligned} \tag{2.20}$$

$$clk\_sav\_rat = \frac{clk\_sav\_MRF + clk\_sav\_SR}{clk\_ori} \tag{2.21}$$

Here, I pick SAD Tree architecture as a case study. Firstly, when adaptive sub-

Table 2.10: Quality Comparison with UMHexagon Search

	BDBR (%)	BDPSNR (dB)
foreman_qcif	+1.046	-0.093
mobile_qcif	+0.415	-0.022
grandma_qcif	+0.948	-0.039
container_qcif	+0.855	-0.030
news_qcif	+0.911	-0.068
tempete_qcif	+0.762	-0.041
coastguard_qcif	+1.004	-0.075
carphone_qcif	+1.335	-0.114
stefan_cif	+1.531	-0.080
mobile_cif	+0.902	-0.052
football_cif	+1.554	-0.096
container_cif	+0.554	-0.080
news_cif	+0.976	-0.099
tempete_cif	+1.141	-0.088
coastguard_cif	+1.261	-0.076
paris_cif	+1.134	-0.068
parkrun_720p	+0.438	-0.024
mobcal_720p	+1.210	-0.061
city_720p	+1.209	-0.103
harbor_720p	+1.088	-0.045

sampling is applied on SAD Tree architecture, the original data flow of SAD Tree can be kept unchanged with modification only in the control module. So, the processing element (PE) can be set idled in different sub-sampling cases. In the original data flow, the whole 256 PEs in the architecture are busy every clock cycle. In my case, the PE' idle ratio (PE_{idle}_ratio) within each frame can be calculated based on Eq. 2.18, where $R(hss+vss)$ represents the sum of horizontal only sub-sampled MB ratio and vertical only sub-sampled MB ratio; $R(qss)$ is the quarter sub-sampled MB ratio; $R(nss)$ is the ratio of MB with

Table 2.11: Speed-up of UMHexagon Search

QP	20	24	28	32	Average
foreman_qcif	1.60	1.72	1.82	1.92	1.77
mobile_qcif	1.48	1.52	1.62	1.65	1.57
grandma_qcif	2.13	2.43	2.38	2.51	2.36
container_qcif	2.04	2.06	1.91	2.06	2.02
news_qcif	1.92	1.95	1.80	1.83	1.87
tempete_qcif	1.47	1.58	1.62	1.65	1.58
coastguard_qcif	1.80	2.04	2.16	2.12	2.03
carphone_qcif	1.72	1.87	1.99	2.13	1.93
stefan_cif	1.33	1.39	1.42	1.46	1.40
mobile_cif	1.54	1.62	1.67	1.67	1.63
football_cif	2.11	2.32	2.48	2.73	2.41
container_cif	2.37	2.34	2.27	2.19	2.29
news_cif	2.42	2.47	2.40	2.31	2.40
tempete_cif	1.53	1.64	1.74	1.87	1.69
coastguard_cif	1.72	1.97	2.18	2.35	2.06
paris_cif	1.68	1.73	1.70	1.72	1.71
parkrun_720p	1.50	1.62	1.75	1.88	1.69
mobcal_720p	1.36	1.42	1.51	1.61	1.48
city_720p	1.55	1.72	1.90	2.14	1.83
harbor_720p	1.41	1.50	1.62	1.80	1.58

no sub-sampling. For horizontal or vertical sub-sampling, 50% PEs can be set idle. In case of quarter sub-sampling, only 25% PEs are kept active. Therefore, many PEs can be set idle during the ME process. Figure. 2.21 is an example of 'container_qcif' under 100 encoding frames, it is assumed that QP is 28 and search range is fixed at 16. It is obvious that much calculation in PEs can be saved in the architecture level and average 46.02% PEs are set as idle during encoding of 100 frames. Secondly, for MRF elimination and SR adjustment scheme, the control module can set the whole IME engine to idle state

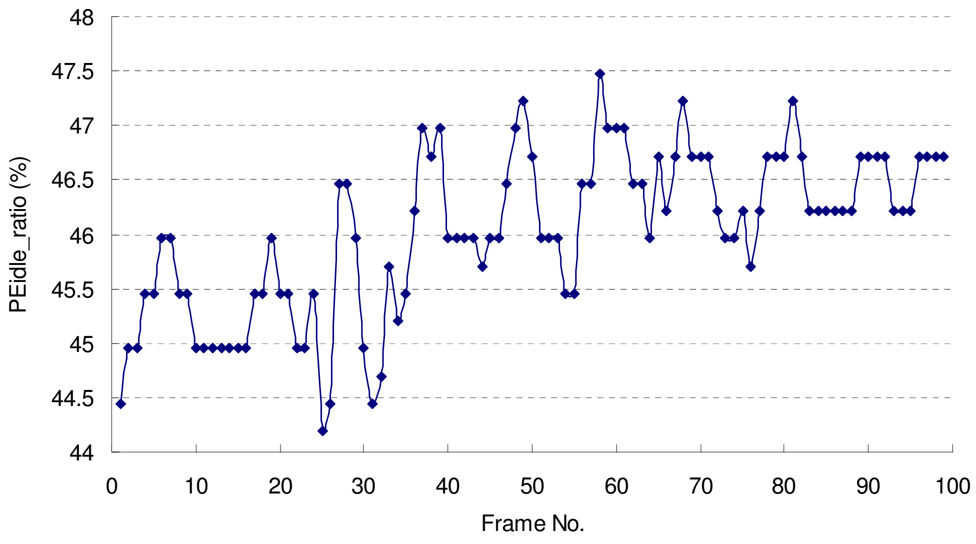


Figure 2.21: PE idle ratio

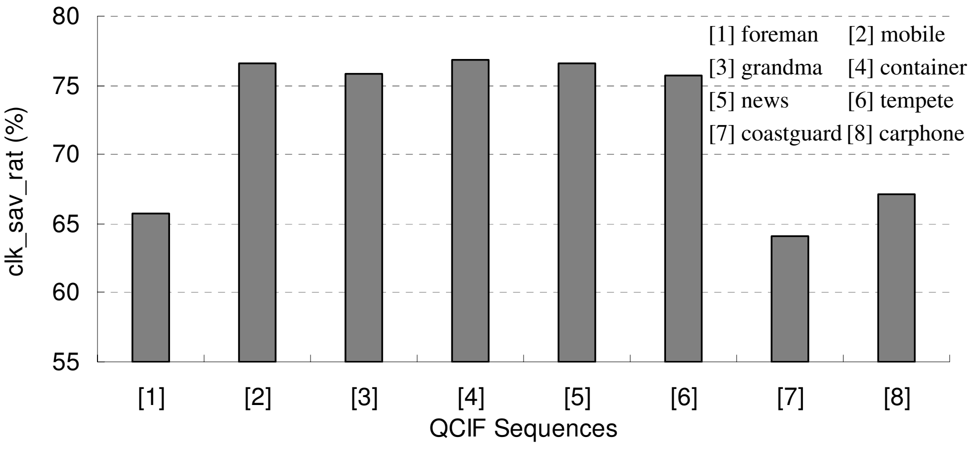


Figure 2.22: Clock cycle saving ratio

when early termination occurs or shorten the processing clock cycles for other reference frames based on motion feature analysis. The clock cycle saving of MRF elimination (clk_sav_MRF) and SR adjustment (clk_sav_SR) schemes is expressed in Eq. 2.19 and Eq. 2.20, where $R(MRF\_skip)$ and $R(SR\_adj)$ are the MRF skipped MB ratio and search range adjusted MB ratio respectively. The MB_num and SP_num represent the number of MB within one frame and search points within the search window. Ref_num is the reference frame number and SR_{JM} is the original JM search range(16/24 for QCIF/CIF, 64 for HDTV720p). Figure. 2.22 gives out percentage of clock cycle saving (clk_sav_rat)

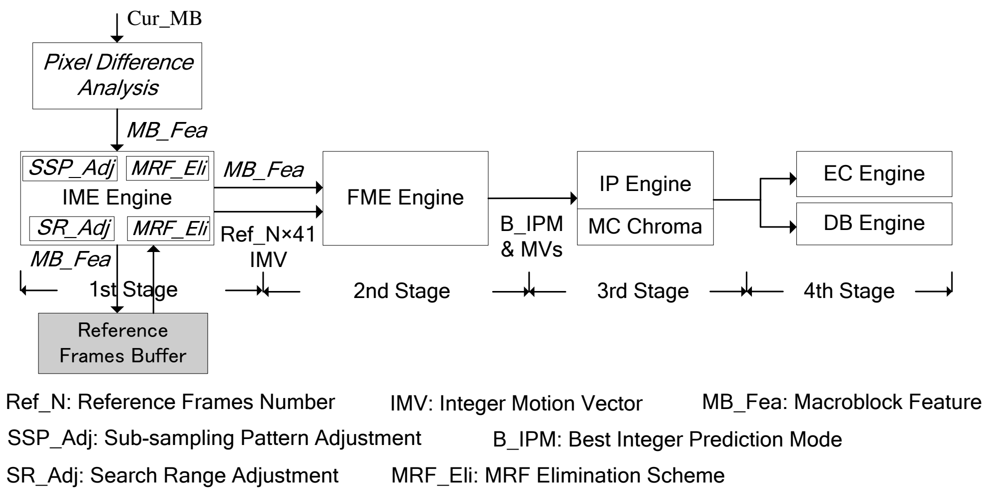

Cur_MB
Pixel Difference Analysis
MB_Fea
SSP_Adj
MRF_Eli
IME Engine
SR_Adj
MRF_Eli
MB_Fea
MB_Fea
Ref_N×41 IMV
FME Engine
B_IPM & MVs
IP Engine
MC Chroma
EC Engine
DB Engine
1st Stage
2nd Stage
3rd Stage
4th Stage
Reference Frames Buffer
Ref_N: Reference Frames Number
IMV: Integer Motion Vector
MB_Fea: Macroblock Feature
SSP_Adj: Sub-sampling Pattern Adjustment
B_IPM: Best Integer Prediction Mode
SR_Adj: Search Range Adjustment
MRF_Eli: MRF Elimination Scheme

Figure 2.23: 4-Stage encoding system with proposed algorithm

based on Eq. 2.21. To simplify the situation, MB with recursive search range adjustment is not included. The *clk_ori* represents the original clock cycles caused by SAD Tree architecture. I use 5 reference frames, 16×16 search window and 100 frames are encoded under QP 28 for case study. Averagely, 72.32% clock cycles can be saved by proposed schemes among these QCIF format sequences.

For memory access, since the proposed algorithm does not disturb the data flow of original full search algorithm, the same memory access scheme in existing IME engine is kept unchanged. The merit is that my schemes also help to save memory access. For example, in case of 1-set SAD Tree architecture [17], it will load 17 pixels (16 pixels for block matching and 1 pixel for column shift in snake scan method [17]) within each clock cycle. With saving in clock cycles by proposed algorithm, the corresponding memory access is also saved. In this dissertation, I focus on the hardware oriented algorithm and do not implement proposed algorithm directly into existing structures. The reason is that my schemes only incur some optimization in control logic and one pixel analysis module. So the extra modification and hardware to existing efficient engines such as PPSAD and SAD Tree [17] are very trivial. In detail, for each processing element (PE) in IME architecture, the three sub-sampling patterns only require one extra 'enable/disable' signal which is managed by system control. The block overlapping analysis only checks integer motion vectors (IMVs) of 16×16 to 8×8 modes and determines whether to end whole IME process for current MB. The information of these IMVs is easily obtained at the end of IME on 1st frame, so the complexity of block matching on following frames can be saved without complicated decision procedure. As for search range adjustment, it also depends on the IMV's information of 16×16 mode at the end of 1st frame's search. The only thing for system control is to set an early stop for block matching on the search window based on our motion feature analysis result.

Figure. 2.23 gives out optimized 4-stage based real-time hardwired encoder and my schemes are marked with italic font. It is shown that the IME engine is separately arranged in a single stage and the FME part is in another stage. For intra prediction (IP), entropy coding (EC) and deblocking filter (DB) engines, they are arranged in 3rd and 4th stages respectively. Based on the pipeline stage, any fast algorithms that use information in the 2nd to 4th stage such as (rate distortion cost [16], information after FME [6] [14] [15]) are impractical because the IME already finishes all its work when such information is available. In case of my schemes, they all work in the IME stage, which is compatible to the existing pipeline stage based design.

2.6 Conclusion remarks

In this chapter, one hardware oriented fast motion estimation algorithm is proposed. The algorithm targets at complexity reduction in three aspects. Firstly, the aliasing problem which is the main reason of video quality degradation is analyzed. By adopting edge detection technique, the complexity incurred by MRF technique is released. Also, one similarity analysis based MRF elimination scheme is also introduced for further reduction of complexity for MB with stationary feature. Secondly, motion feature of current MB is extracted during block matching process. Redundant search points for small motion MB is eliminated by restricting small motion MB's search area within a small centering region. Moreover, an recursive search range adjustment scheme is employed for MBs with different motion feature. Thirdly, by executing pixel difference analysis which is arranged before IME engine, an adaptive sub-sampling scheme is introduced for complexity reduction of full pixel pattern. Altogether, by combining all these schemes, the proposed algorithm can achieve 83.69% to 95.72% ME time reduction with trivial video quality loss compared with full search algorithm. Averagely, about 88.53% ME time is reduced among different sequences. Furthermore, the proposed fast ME algorithm is orthogonal to existing software oriented fast motion estimation algorithms, which can achieve speeds-up ratio of conventional UMHexagon search up to 2.73. Since the proposed algorithm operates in a hardware friendly way, it can be easily implemented in the 4-stage pipeline based real-time video encoding system.

Chapter 3

Flexible integer motion estimation architecture

3.1 Introduction

In the previous chapter, several schemes for hardware oriented algorithms are introduced. With these schemes, the complexity reduction can be achieved based on hardware data flow. Also, the related clock cycle saving ratio based on MRF elimination and search range adjustment schemes are analyzed. It is obvious that, with some control modules, the proposed MRF and search range schemes can be easily applied to existing architectures, such as SAD Tree and propagate partial SAD architectures. The control part for these schemes are belong to the system level adjustment.

In terms of adaptive sub-sampling, it can not be efficiently applied on existing fixed architectures. Here, Eq. (3.1) is introduced for MB classification. In detail, if all the $PD_h(i,j)$ of current MB is within a pre-defined threshold (THR_{PD}), this MB is called horizontal homogeneous MB (H_{homo}). The concept for V_{homo} can be traced with analogy. For MB which has all its $PD_h(i,j)$ and $PD_v(i,j)$ are within THR_{PD}, it is called strong homogeneous MB (S_{homo}). Otherwise, it is a none homogeneous MB (N_{homo}). Three hardware friendly sub-sampling patterns (pattern 1 to 3), as shown in Fig. 3.1, are used to reduce complexity of IME according to H_{homo}, V_{homo} and S_{homo} cases. The pattern 4 is full pixel pattern which is used for N_{homo} MB. The pixel difference analysis (PDA)

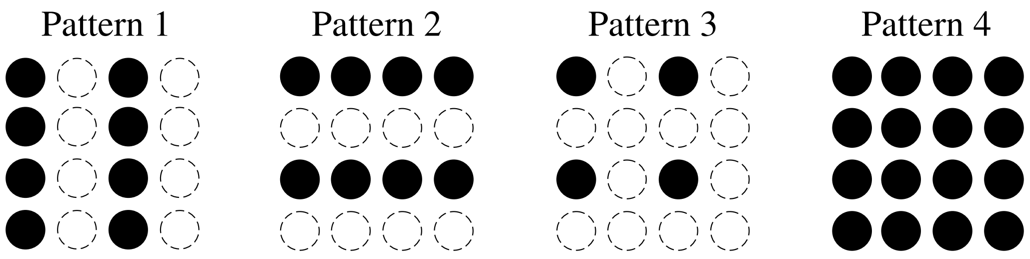


Figure 3.1: Sub-sampling patterns and full pixel pattern

is executed during loading of current MB. The THR_{PD} is set as 4×QP (quantization parameter) based on empirical and exhaustive experiments. As shown in Table 3.1, the PDA based adaptive sub-sampling will have better video quality than direct sub-sampling scheme. Another merit of adaptive sub-sampling is that it is friendly to power aware system for different customer's demand. In fact, the direct half sub-sampling [31] and quarter sub-sampling [19] are sub-classes of adaptive algorithm.

$$\begin{cases} H_{homo} : PD_h(i,j) < THR_{PD} \\ V_{homo} : PD_v(i,j) < THR_{PD} \\ S_{homo} : (PD_h(i,j) < THR_{PD}) \ \& \ (PD_v(i,j) < THR_{PD}) \\ N_{homo} : otherwise \\ i \in [1,15];\ j \in [1,15] \end{cases} \tag{3.1}$$

However, direct application of adaptive algorithm on existing fixed architecture will cause poor data reuse and hardware utilization. Moreover, repetitive of pixels loaded from SRAM will degrade the efficiency of adaptive sub-sampling scheme, especially when large image size such as HDTV application is incurred. Fig. 3.2 gives out a demonstration of data reuse problem. Assume that current MB is a H_{homo} MB and pattern 1 is adopted for current MB's IME. In the original SAD Tree structure, it loads 16 pixels in each cycle for SAD calculation and 1 extra pixel for column shift in snake scan method [17]. Based on the original data flow, only 50% pixels are useful for SAD calculation. As for hardware utilization, also 50% processing elements (PEs) can not be fully utilized. In case of S_{homo}, the waste of hardware resource rises up to 75%. So, flexible architectures are required for adaptive scheme.

Table 3.1: Quality analysis of adaptive sub-sampling

direct_ss	BDBR (%)	BDPSNR (dB)
crew_720p	+0.65	-0.025
city_720p	+1.23	-0.057
stockholm_720p	+2.19	-0.208
knightshields_720p	+2.38	-0.773
harbour_720p	+1.35	-0.077
parkrun_720p	+2.78	-0.770
adapt_ss	BDBR (%)	BDPSNR (dB)
crew_720p	**+0.22**	**-0.011**
city_720p	**+0.12**	**-0.006**
stockholm_720p	**+0.25**	**-0.014**
knightshields_720p	**+0.16**	**-0.008**
harbour_720p	**+0.14**	**-0.010**
parkrun_720p	**+0.39**	**-0.025**

direct_ss: direct quarter sub-sampling

adapt_ss: PDA based adaptive sub-sampling

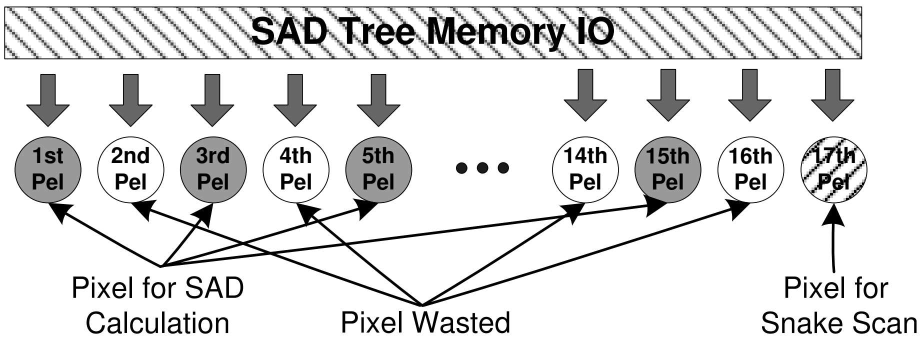


Figure 3.2: Data reuse problem in SAD Tree structure

The proposed flexible IME architectures are based on original SAD Tree and propagate partial SAD (PPSAD) structures. Firstly, with memory level and architecture level pixel organization, problems in data reuse and hardware utilization are well solved. Secondly, with configurable SAD and interactive data loading scheme, the processing cycle

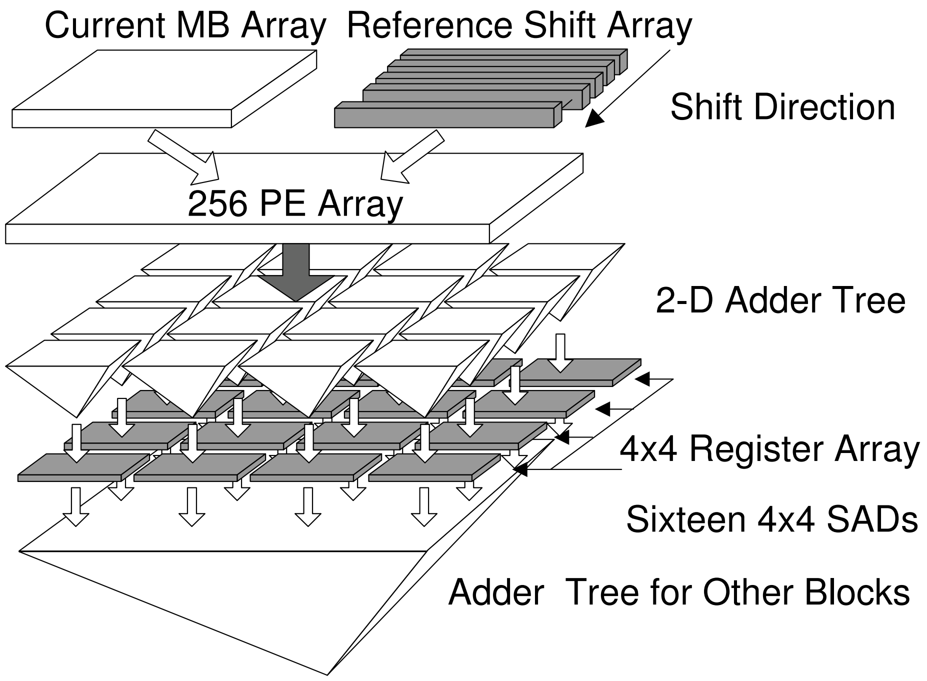


Figure 3.3: Original SAD Tree structure

and power dissipation of previous designs are greatly reduced. Moreover, circuit level optimization is applied in the proposed architecture which further saves hardware cost and power dissipation. The details are in the following sections.

3.2 Reconfigurable SAD tree architecture

3.2.1 System architecture

The proposed reconfigurable SAD Tree (RSADT) architecture is shown in Fig. 3.4. The left up part is the PDA part. It provides the pattern selection signal for proposed structure. During loading of current MB pixels (Pels), with 4 shift registers (shift_reg), 4 absolute difference operations (abs_diff_opt) and 2 adders, the VPD and HPD can be obtained. Therefore, the sub-sampling pattern is decided before IME starts to work. The extra calculation which is introduce to the system will not degrade system performance because the loading of current MB pixel occurs only once during the whole IME process.

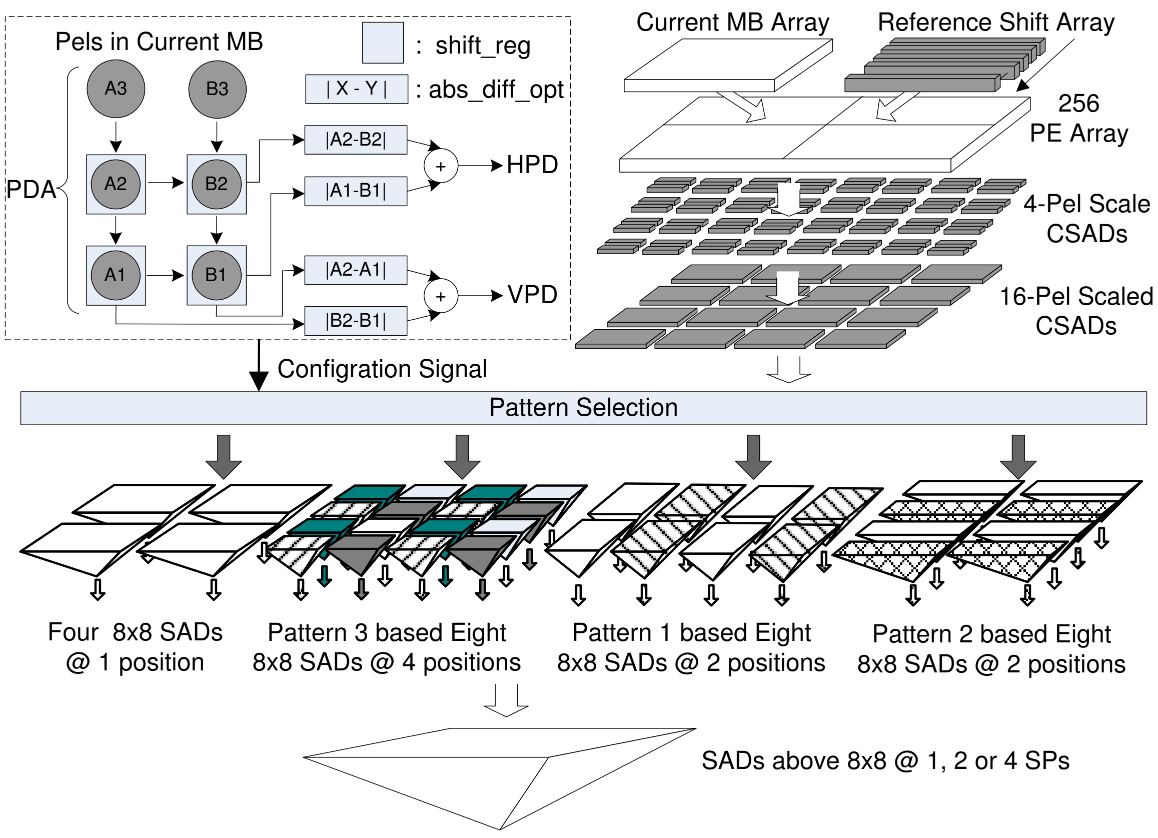


Figure 3.4: Proposed reconfigurable SAD tree architecture

In the proposed architecture, three major modifications are applied compared with SADT [17], as shown in Fig. 3.3. Firstly, instead of pipelining at partial 4×4 or 8×8 SAD scale, the proposed structure pipelines at Pel scale, that is 4-Pel scale and 16-Pel scale. The purpose of this adoption is to achieve full data reuse for the adaptive algorithm. Secondly, two pipeline stages are inserted in the RSADT (4-Pel and 16-Pel). Compared with one pipeline stage in SADT, the whole system clock speed is enhanced. Thirdly, based on 4-Pel SADs, an architecture level pixel organization scheme is introduced to form 4-Pel scaled configurable SAD (CSAD). So, the data reuse and hardware utilization problems are solved. Based on these 4-Pel CSAD and memory level pixel organization, the processing cycles can be shortened for MB with different homogeneity. Furthermore, a cross reuse structure for 16-Pel scaled CSAD generation is proposed to realize adaptive scheme efficiently. In the following section, description in detail will be given out.

3.2.2 Architecture level data organization and circuit modification

For architecture design, the data organization is always a critical problem to the whole system performance. The organization can happen in the memory level or in the architecture level. In the proposed architecture, I apply data organization in both levels.

Firstly, an architecture level data organization is proposed for RSADT architecture. Figure. 4.5(a) shows one 8×8 residue block. The original SADT pipelines at 4×4 SAD ($SAD_{4\times4}$) and generates 8×8 SAD by accumulating four $SAD_{4\times4}$. Equation. (3.2) and Eq. (3.3) are the expressions of left-up $SAD_{4\times4}$ and the whole $SAD_{8\times8}$ of Fig. 4.5(a), where G represents a group of marks which indicates the specific pixels for one $SAD_{4\times4}$. The related $SAD_{4\times4}$ for three sub-sampling patterns will change to Eq. (3.4) accordingly. Also, the related 8×8 SAD can be derived as Eq. (3.5). From Fig. 4.5(a) and Eq. (3.2) to Eq. (3.5), it is obvious that, when pattern 1 is adopted, the accumulated SAD of position Bk and Dk ($k = 1 \sim 16$) are $SAD_{8\times8}\_H$ of neighboring search point (SP2). Based on the same principle, SAD_{8x8}_V at SP1 and SP3 are available simultaneously when pattern 2 is used. In case of pattern 3, the SAD_{8x8}_S at SP1 to SP4 can be obtained at one time. So, I reorganize the SAD value as Fig. 4.5(b). It is clear that every two rows represent one $SAD_{4\times4}$ at SP1. On the other hand, when these SADs are accumulated vertically, it can form three types of outputs, that is four SAD_{8x8}_S at SP1 to SP4, two SAD_{8x8}_H at SP1 and SP2, or two SAD_{8x8}_V at SP1 and SP3. Then, one pipeline stage is inserted and sixteen 4-Pel scaled configurable SAD (CSAD) is formed, as shown in Fig. 3.6. Equation. (3.6) gives out the formation of all 4-Pel scaled CSADs.

$$\begin{aligned} SAD_{4x4} &= \Sigma_{k\in G} Ak + Bk + Ck + Dk \\ G &= \{1, 2, 5, 6\} \end{aligned} \tag{3.2}$$

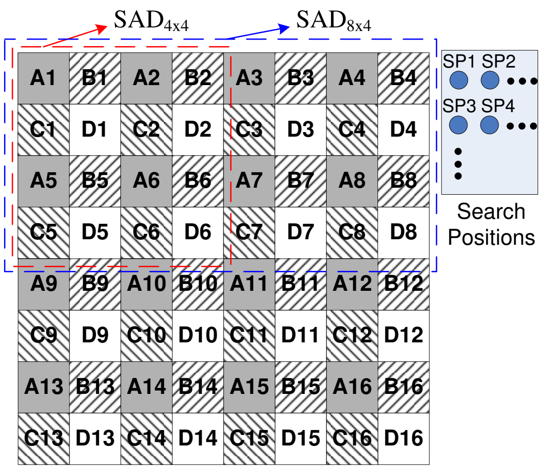
$$SAD_{8x8} = \Sigma_{k=1}^{k=16} Ak + Bk + Ck + Dk \tag{3.3}$$

$$\begin{cases} H_{homo} : SAD_{4x4}\_H = \Sigma_{k\in G} Ak + Ck \\ V_{homo} : SAD_{4x4}\_V = \Sigma_{k\in G} Ak + Bk \\ S_{homo} : SAD_{4x4}\_S = \Sigma_{k\in G} Ak \end{cases} \tag{3.4}$$

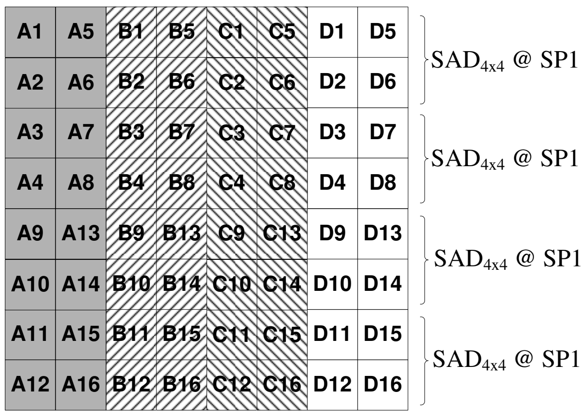
$$\begin{cases} H_{homo} : SAD_{8x8}\_H = \Sigma_{k=1}^{k=16} Ak + Ck \\ V_{homo} : SAD_{8x8}\_V = \Sigma_{k=1}^{k=16} Ak + Bk \\ S_{homo} : SAD_{8x8}\_S = \Sigma_{k=1}^{k=16} Ak \end{cases} \tag{3.5}$$

As shown in Fig. 3.6, when four horizontal CSAD values are added together, it will form one 16-Pel scaled $SAD_{4\times4}$ at SP1. Similarly, four 16-Pel scaled SAD_{8x8}_S can be obtained vertically. The decision of SAD generation is based on PDA. As shown in Fig. 3.4, all the sixteen 16-Pel scaled CSADs are pipelined. Based on these 16-Pel CSADs, adaptive output result is available, that is $SAD_{4\times4}$ at one SP, SAD_{8x8}_H or SAD_{8x8}_V at two SPs, or SAD_{8x8}_S at four SPs. So, the processing capability is doubled or quadrupled for MB with different homogeneity.

Secondly, since the adaptive scheme is applied in hardware, the original two dimensional reference shift array (RSA) must be modified. Figure. 3.8 is the original RSA structure which contains 272 SUs in 16 rows and 17 columns. The basic module of RSA is the snake scan unit (SU), as shown in Fig. 3.7. Assume that current SU is in ith row and j column (SU[i,j]). To enable snake scan, the related upper, lower and right data inputs are shown in Eq. (3.7), where SU[i,j]_O is the output of SU[i,j]. The modified SU (MSU) is given out in Fig. 3.7. The data input number is doubled and the relation with other MSU is shown in Eq. (3.8). With MSU module, the original RSA structure is changed to Fig.3.9.



(a) Original organization



(b) Optimized organization

Figure 3.5: Pixel data organization

$$\begin{cases} CSAD1 = A1 + A2 + A5 + A6 \\ CSAD2 = B1 + B2 + B5 + B6 \\ CSAD3 = C1 + C2 + C5 + C6 \\ CSAD4 = D1 + D2 + D5 + D6 \\ CSAD5 = A3 + A4 + A7 + A8 \\ CSAD6 = B3 + B4 + B7 + B8 \\ CSAD7 = C3 + C4 + C7 + C8 \\ CSAD8 = D3 + D4 + D7 + D8 \\ CSAD9 = A9 + A10 + A13 + A14 \\ CSAD10 = B9 + B10 + B13 + B14 \\ CSAD11 = C9 + C10 + C13 + C14 \\ CSAD12 = D9 + D10 + D13 + D14 \\ CSAD13 = A11 + A12 + A15 + A16 \\ CSAD14 = B11 + B12 + B15 + B16 \\ CSAD15 = C11 + C12 + C15 + C16 \\ CSAD16 = D11 + D12 + D15 + D16 \end{cases} \tag{3.6}$$

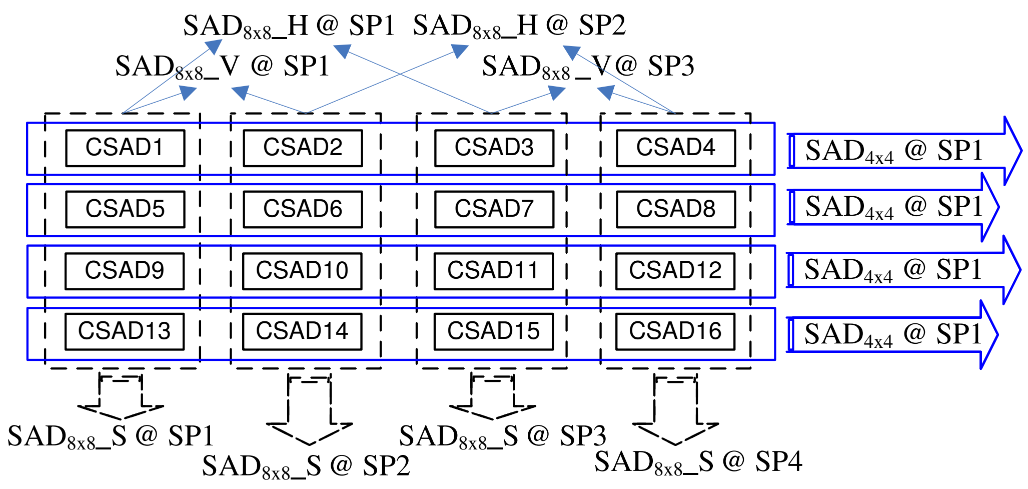


Figure 3.6: 4-Pel scaled CSAD

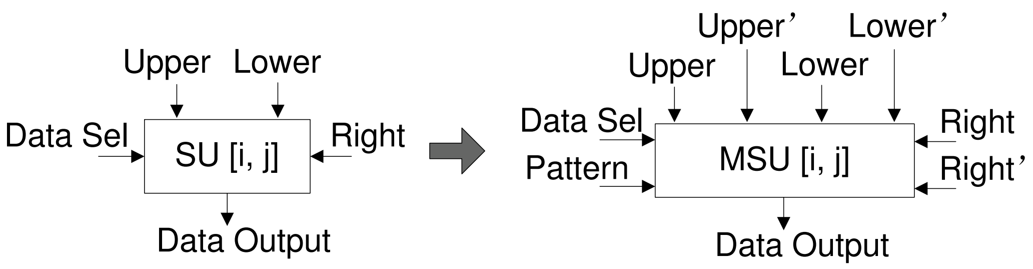


Figure 3.7: Modification in SU

$$
\begin{aligned}
Upper &= SU[i-1, j]\_O \\
Lower &= SU[i+1, j]\_O \\
Right &= SU[i, j+1]\_O
\end{aligned}
\tag{3.7}
$$

$$
\begin{aligned}
Upper = MSU[i-1, j]\_O, Upper' &= MSU[i-2, j]\_O \\
Lower = MSU[i+1, j]\_O, Lower' &= MSU[i+2, j]\_O \\
Right = MSU[i, j+1]\_O, Right' &= MSU[i, j+2]\_O
\end{aligned}
\tag{3.8}
$$

3.2.3 Memory level pixel organization

The memory pixel organization also has to be modified to enable full data reuse. The original memory organization of SADT architecture is shown in Fig. 3.10(a). Here, I only show search window of 32 × 32 (last 15 columns of data is added for the block matching

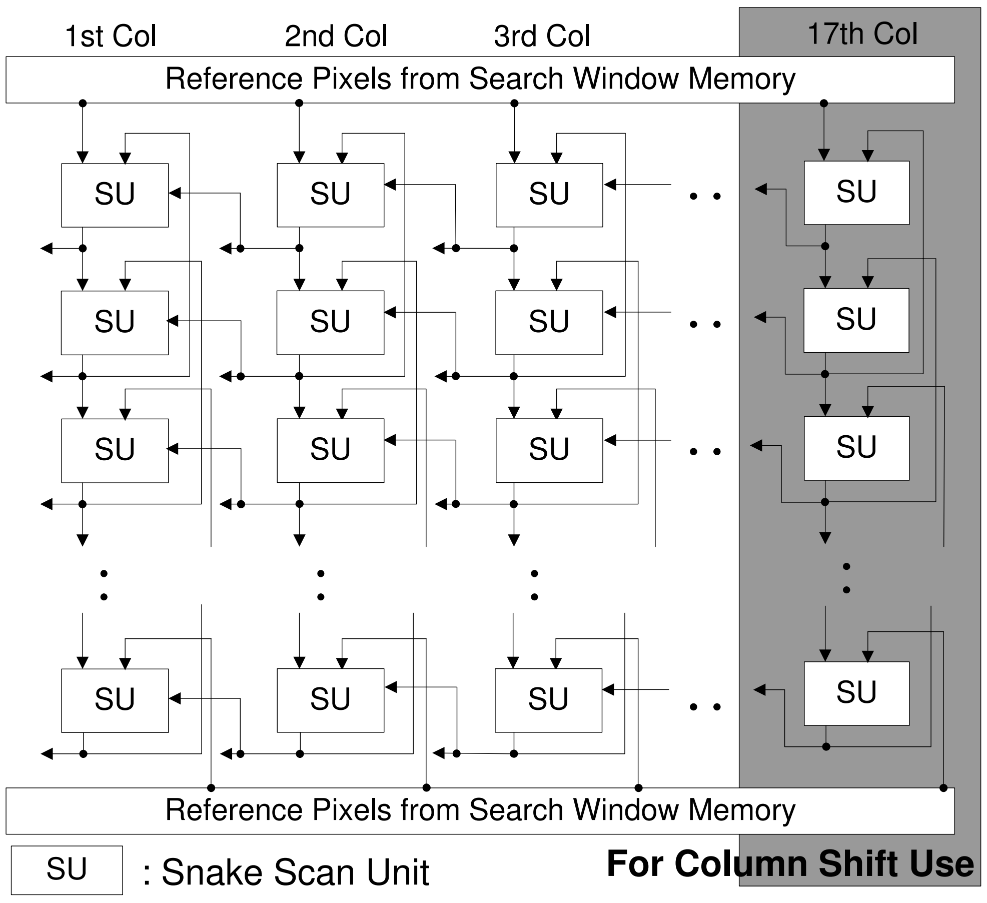

1st Col
2nd Col
3rd Col
17th Col
Reference Pixels from Search Window Memory
SU
Reference Pixels from Search Window Memory
SU
: Snake Scan Unit
For Column Shift Use

Figure 3.8: Original reference shift array

of positions on the 32th column) as an example. For the SADT architecture, by using memory mapping algorithm [28], the data loaded from search window memory is fully utilized. In each clock cycle, 17 pixels are loaded from the memory and transferred to the RSA. As shown in Fig. 3.8, 16 pixels are used for block matching calculation and 1 pixel is prepared for column shift in the snake scan method.

Since adaptive algorithm is applied on proposed RSADT architecture, the original memory pixel organization should be optimized to keep full data reuse. Figure. 3.10(b) demonstrates the proposed scheme. The whole reference pixels are classified into columns with odd rows and even rows. Then, they are arranged into two memory groups (A and B), which output pixel row to the modified RSA. The adoption of group division is mainly for the adaptive patterns. For example, in case of pattern 2 and pattern 3, two succeeding pixel rows are required for the modified RSA. To enable data reuse in pattern

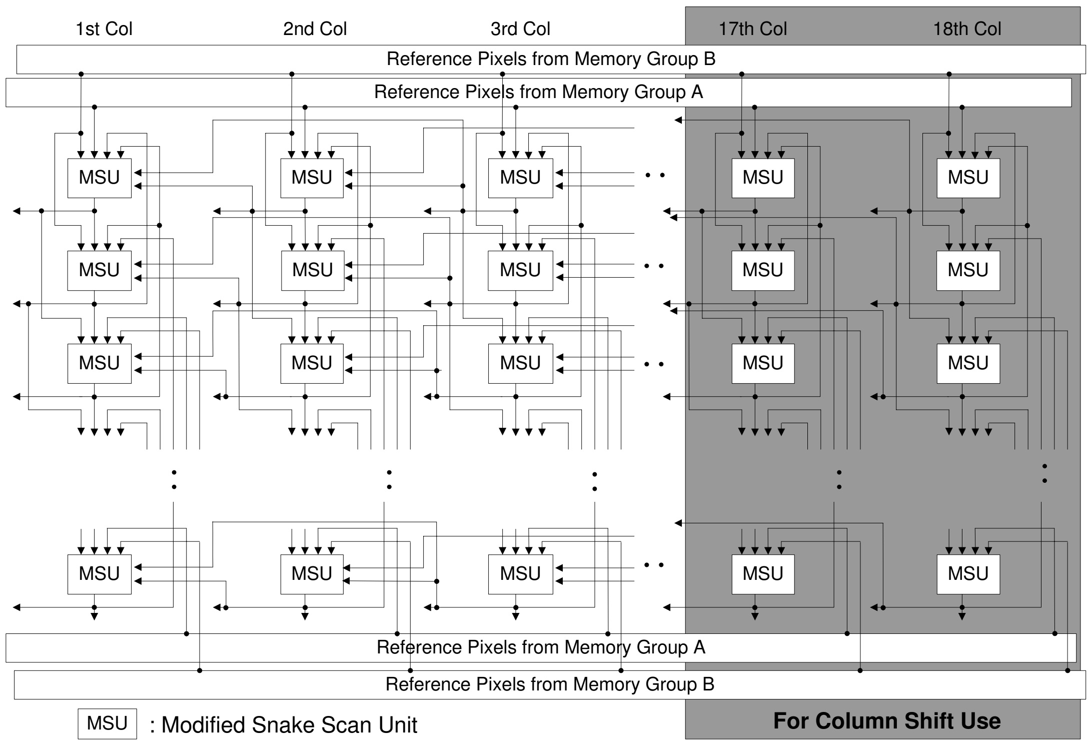
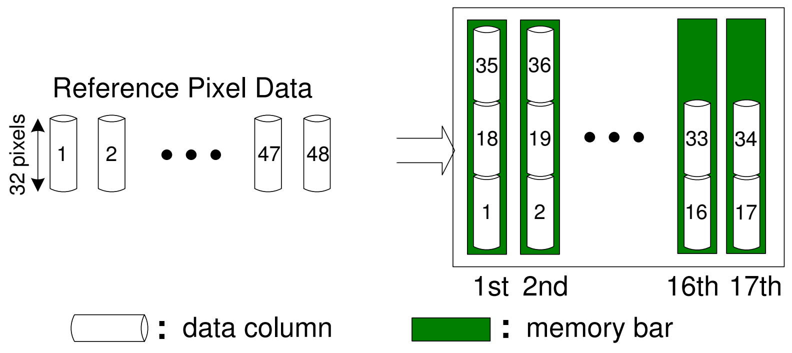

1st Col
2nd Col
3rd Col
17th Col
18th Col
Reference Pixels from Memory Group B
Reference Pixels from Memory Group A
MSU
Reference Pixels from Memory Group A
Reference Pixels from Memory Group B
MSU : Modified Snake Scan Unit
For Column Shift Use

Figure 3.9: Modified reference shift array

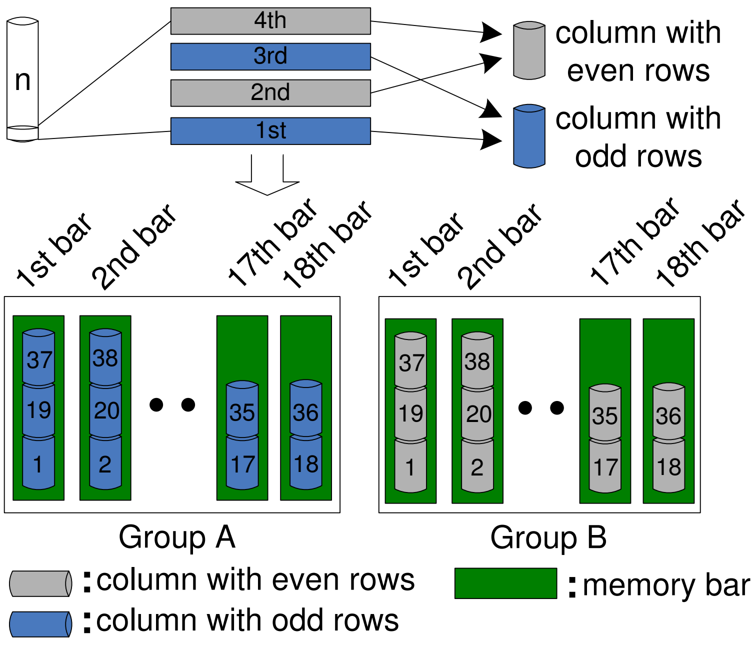
1, one extra pixel column (18th column) is added for column shift in MSU. Finally, the memory overlapping algorithm [28] is used for both memory groups.

3.2.4 Cross reuse structure for CSAD generation

Thirdly, the 4-Pel scaled CSAD is fully utilized and one cross reuse structure (CRS) for 16-Pel scaled CSAD generation is proposed. Figure. 3.11 is the proposed CRS structure. They are the same circuits with different configurations. In the intuitive implementation of Fig. 3.6, 8 adders and 4 big multiplexors are required to generate four 16-Pel scaled CSAD of one 8×8 block. For HDTV application, when 8 parallel IME engine is adopted, there will be 256 adders and 128 multiplexors. With the increase of synthesis clock speed, the hardware cost of these adders and multiplexors will be dilated greatly. In my design, I fully utilize the 4-Pel scaled CSAD and only four adders are need for generating all the 16-Pel scaled CSAD. As shown in Fig. 3.11, based on the control signal (Ctrl) from PDA module, the CRS can be used to get unsub-sampled 16-Pel scaled CSADs like



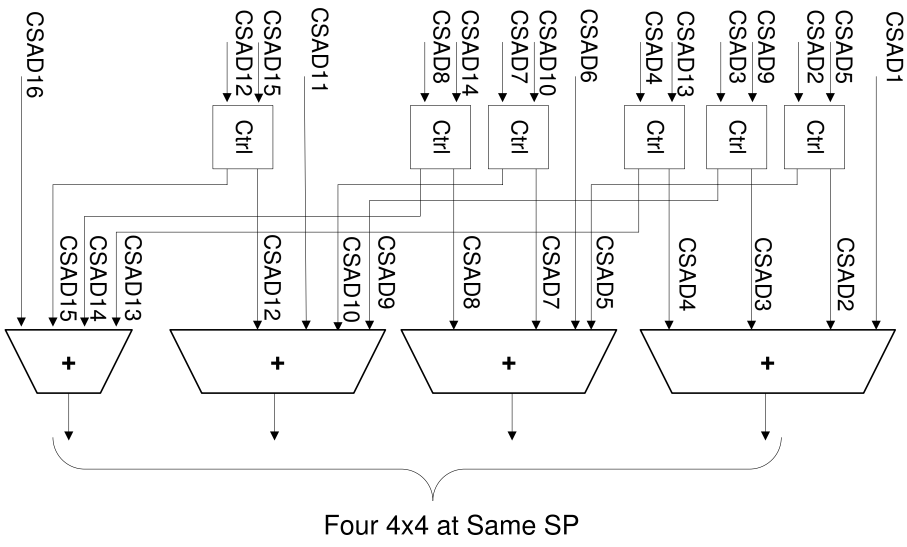
(a) Original pixel organization



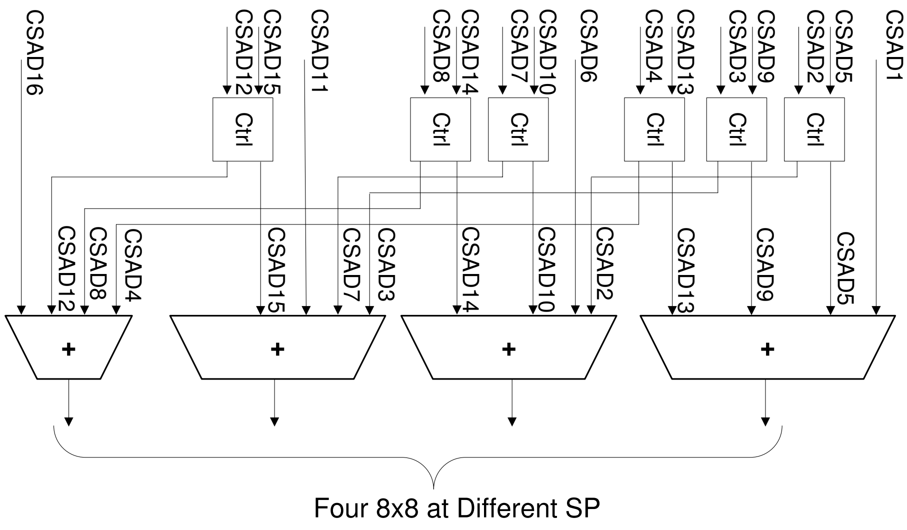
(b) Our proposed organization

Figure 3.10: Memory level pixel organization

Fig. 3.11(a) or sub-sampled CSADs like Fig. 3.11(b). Thus, the 4-Pel CSAD to 16-Pel CSAD generation process is fulfilled efficiently with our cross reuse structure.



(a) No sub-sampling case



(b) sub-sampling case

Figure 3.11: Cross reuse structure for CSAD generation

3.3 Adaptive propagate partial SAD architecture

3.3.1 System architecture

Based on the same adaptive algorithm, one adaptive propagate partial SAD architecture (APPSAD) is also proposed. Figure. 3.12 is the proposed APPSAD. Compared with fixed PPSAD architecture in [17]. Three major optimizations are applied in the architecture level.

Firstly, since the proposed architecture is target for HDTV application, the contribution of small inter mode is very trivial. So, I use mode reduction technique in the APPSAD architecture, which means that inter mode below 8×8 is discarded. Due to this adoption, the hardware costs related with small inter mode is removed.

Secondly, in the previous PPSAD architecture, 64 PEs are grouped together and used to accumulate one 8×8 SADs. In APPSAD architecture, the original fixed structure is modified for adaptive algorithm. Figure. 3.13(a) is the intuitive implementation of adaptive algorithm on previous architecture. PEs with black color are activated for all the patterns just like conventional PEs (name it PE_CONV) while PEs represented with triangle, square or grey circle are pattern dependent ones, which will be activated or deactivated according to different sub-sampling patterns. Besides, since the number of partial SADs will vary based on different sub-sampling patterns, some multiplexors are added into the architecture to enable adaptive feature. For example, when vertical sub-sampling is adopted, all the PEs on even lines of Fig. 3.13(a) are bypassed by configuring all the multiplexors. It is obvious that in the intuitive way, many multiplexors are required, which will intensify the complexity in control logic. The hardware size will also be dilated consequently due to this operation. In APPSAD structure, as shown in Fig. 3.12, I group all the PEs according to their types. For example, all the PE_CONV within one 8×8 block are grouped together and only one multiplexor is used to realize control logic of adaptive algorithm. Thus, 75% number of multiplexors are removed.

Thirdly, besides conventional PE (PE_CONV), extra three different PEs exist in our design, namely PE_DHS, PE_DVS and PE_FPS, which is represented with square, triangle and grey circle in Fig. 3.13(a), respectively. Each of these elements are activated or deactivated at different sub-sampling patterns. In detail, PE_DHS is disabled in horizontal sub-sampling case and PE_DVS is deactivated in vertical sub-sampling case. As for PE_FPS, it is only enabled for full pixel block matching situation (N_{homo} case). For PE_CONV, their status are always 'ON' no matter what sampling pattern the system selects. In the detail design, I simply send one disable signal to the PE to be deactivated and the output result of such PE will turn to constant zero. So, under different sub-

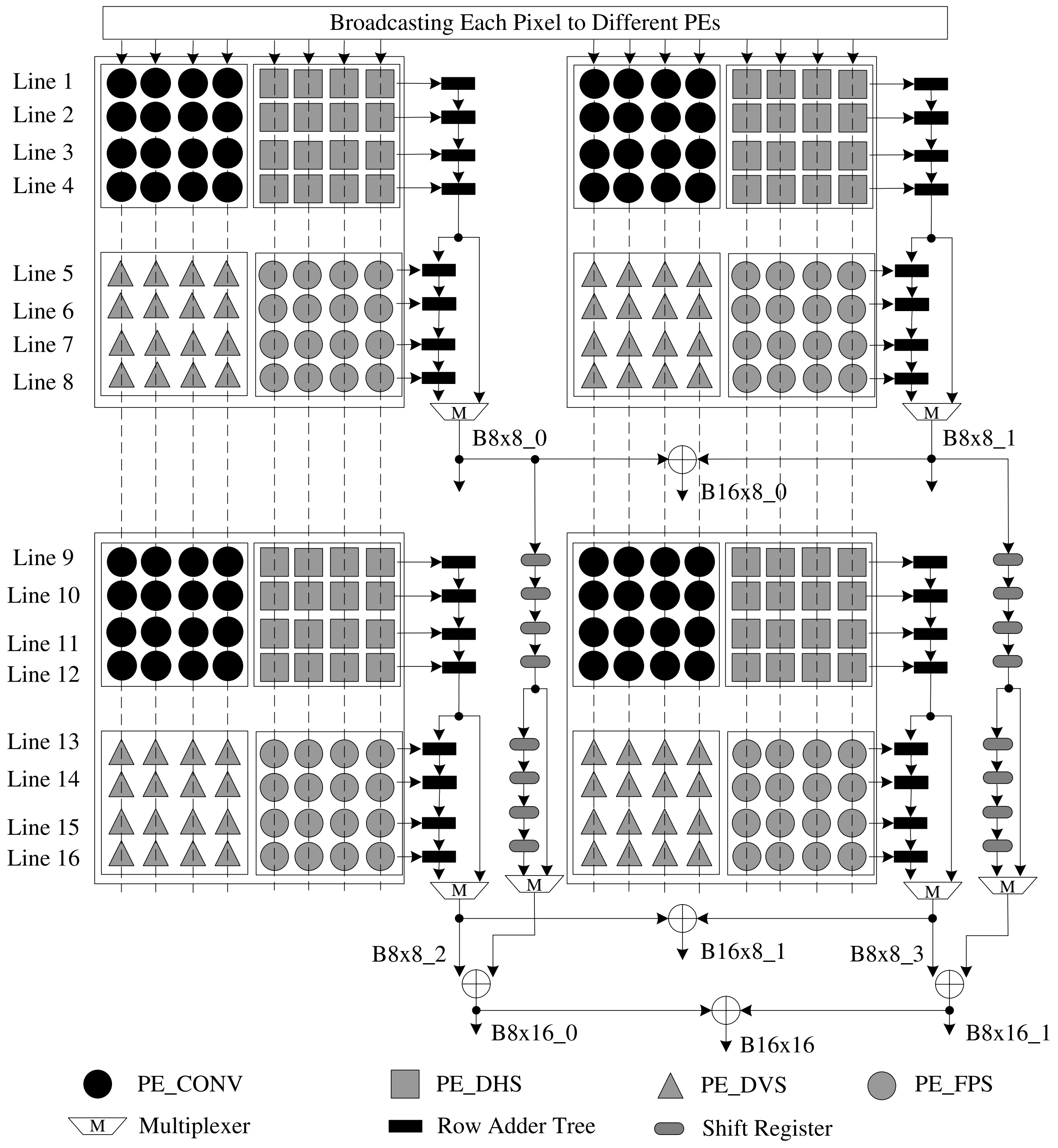


Figure 3.12: Adaptive propagate partial SAD architecture

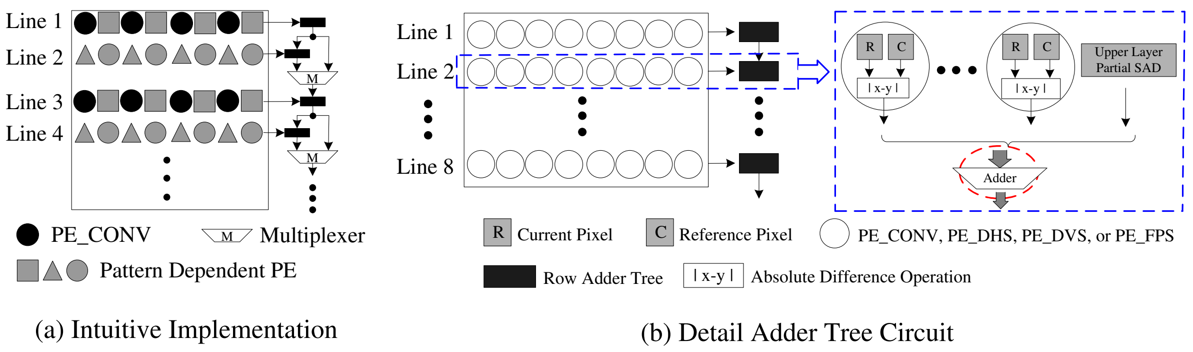


Figure 3.13: 8x8 PE array in PPSAD architecture

sampling patterns, the architecture can enable corresponding PEs for the block matching process. Many absolute difference calculations are saved and power dissipation is reduced in the architecture level consequently.

3.3.2 Memory organization

In the memory level, the original memory structure also needs to be modified to improve data reuse. In [20], it uses a memory overlapping algorithm to fully utilize pixel data loaded from memory. In my design, I divide memory pixels into four types, namely even-row, odd-row, even-column, odd-column, as shown in the left part of Fig. 3.14. All the square pixels in Fig. 3.14 are odd-row-odd-column pixel (P_{oo}); the triangle represents even-row-even-column pixels (P_{ee}); for circle and diamond symbols, they are odd-row-even-column pixel (P_{oe}) and even-row-odd-column (P_{eo}) respectively.

In the second step, all pixels are grouped together according to their types. Since there four patterns (including full pixel pattern) in my design, two memory groups are needed to store them. For instance, in case of N_{homo} and V_{homo}, the required pixel number (16 pixels per clock) for APPSAD architecture is two times of H_{homo} and S_{homo} cases (8 pixels per clock). So, as shown in Fig. 3.14, two memory groups, namely Mem_GA and Mem_GB, are used. Each group contains several one-pixel width memory bars.

All the P_{oo} (Part_OO) and P_{eo} (Part_EO) are stored in Mem_GA while the other two type pixels (Part_OE and Part_EE) are stored in Mem_GB. To improve the IO bandwidth utilization and erase bubble clock cycles of PPSAD based architecture [17], I further

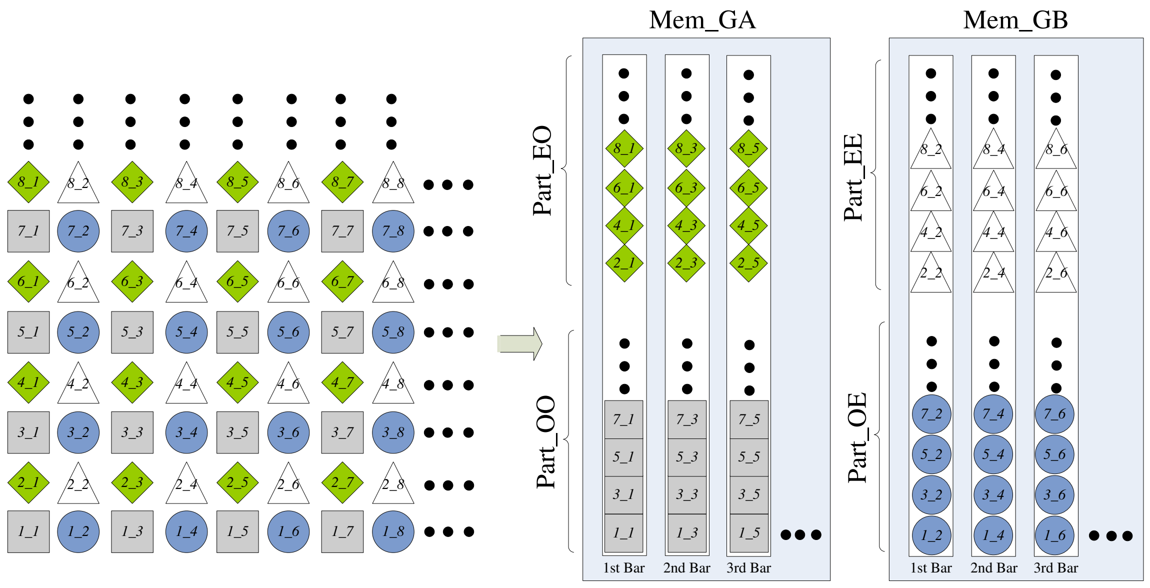


Figure 3.14: Pixel classification and memory organization

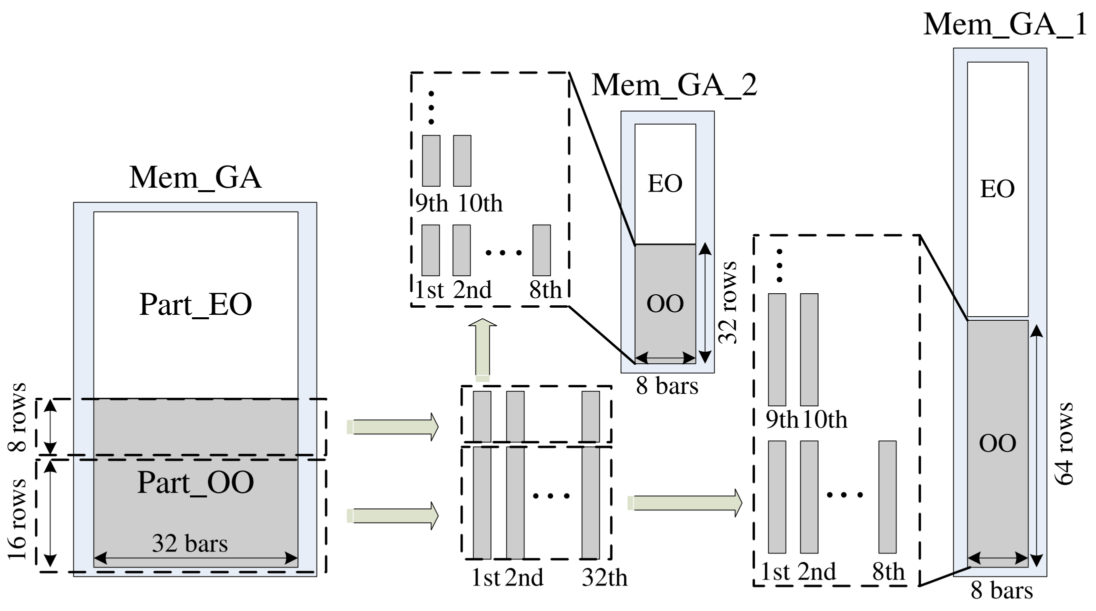


Figure 3.15: Memory separation and overlapping

separate each group into 2 sub-group, namely Mem_GA_1, Mem_GA_2, Mem_GB_1 and Mem_GB_2, and apply memory mapping algorithm [20]. Fig. 3.15 gives out an example. Assume that search range size is 48 in width (W=48) and 32 (H=32) in height. Last fifteen rows and columns are added for block matching on the boundary parts. One row and column is added for hardware implementation. So, the search window size is (W+16)×(H+16). Based on our pixel classification, the size of each part in Fig. 3.14, for example Part_OO, is 32×24. As shown in Fig. 3.15, I separate the last 8 rows of each part and apply memory overlapping algorithm [20] on both Mem_GA_1 and Mem_GA_2. So, the clock bubble in PPSAD based architecture is removed. Each memory group contains eight memory bars, which makes 100% IO bandwidth utilization for different sub-sampling patterns. In this dissertation, I only focus on the IME's on-chip memory and do not deal with pixel organization of off-chip frame memory. The original Level C or Level D [32] off-chip to on-chip data reuse scheme and their corresponding scan order still can be used for the whole encoder system. So, the required off-chip to on-chip memory bandwidth is the same with Level C or Level D scheme. The proposed pixel classification can be done in the encoder's system level, which is not ascribed to the IME engine's job.

Thirdly, the data flow of our architecture is different from previous design. Figure. 3.16 is my memory data loading flow for four types of patterns. To simplify the explanation, Mem_GA_1 and Mem_GA_2, Mem_GB_1 and Mem_GB_1 are merged together in my description.

As shown in Fig. 3.16, there are two stages in H_{homo} case. In the 1st Stage, the Part_Sel signal chooses data from Mem_GA, which means that only Part_OO and Part_EO are the candidate Parts. The pixel data are loaded interactively from these two parts and Mem_GB is set to idle state, which saves power of Mem_GB part. Based on our data organization style, the memory address control is also simplified. The difference of succeeding two addresses is only the height of Part_OO. For example, assume that there are h addresses in each bar of Part_OO. In $2n$th cycle, one pixel row at address m of Part_OO is loaded, in the next cycle ($(2n+1)$th cycle), another pixel row from Part_EO is required for the APPSAD structure based on pattern 2 of Fig. 2.17. The address of this pixel row will be $(m+h)$. The address generation of N_{homo} case and H_{homo}'s 2nd Stage

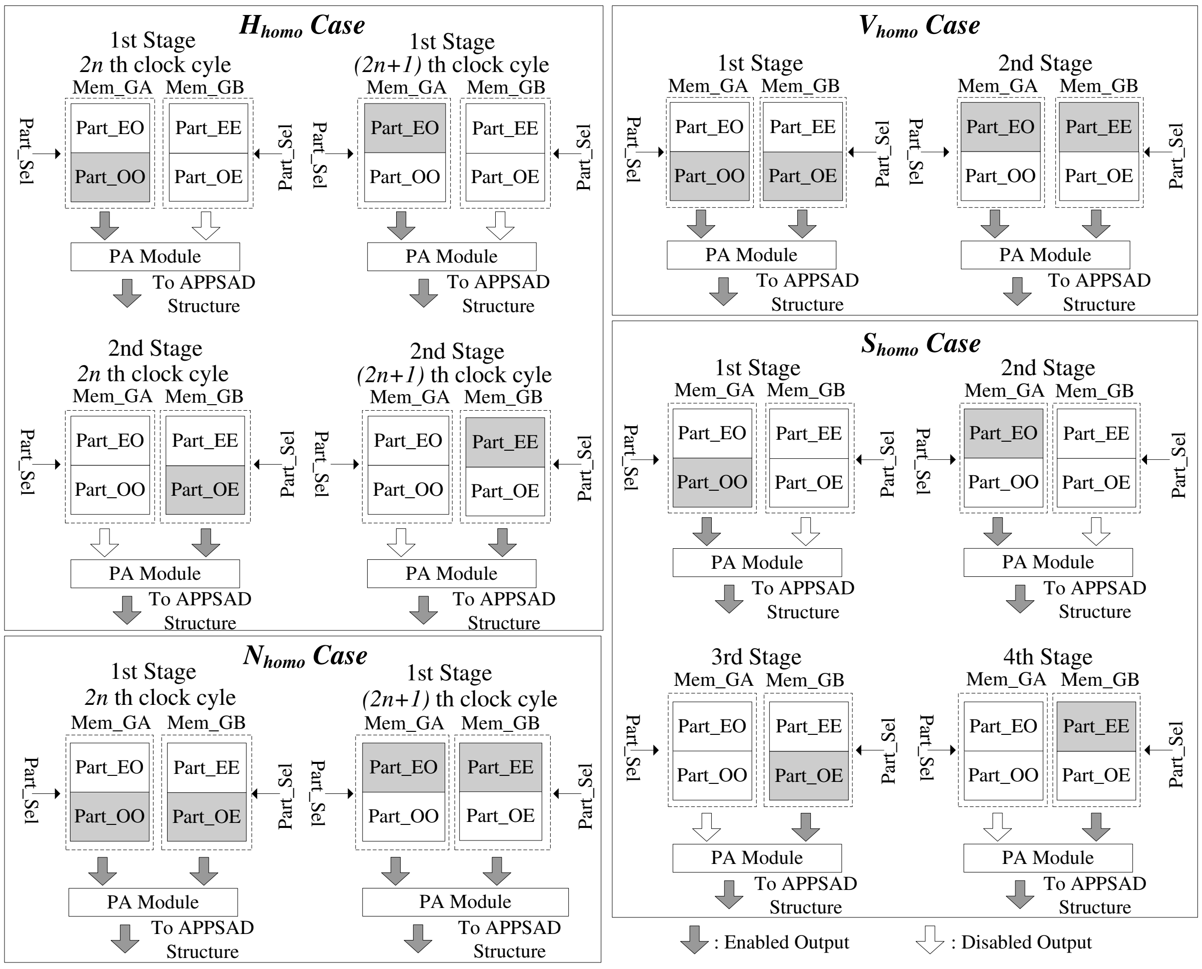


Figure 3.16: Data flow of APPSAD architecture

can be traced by analogy. When all the pixels in Part_OO and Part_EO of H_{homo} case are loaded, it turns to 2nd Stage, during which only Part_EE and Part_OE are candidate parts and power dissipation for Mem_GA can be saved.

For V_{homo} case, it also consists of two stages. In each clock cycle, both memory groups are activated because the required number of pixels for APPSAD structure is doubled (16 pixels) according to pattern 3 of Fig. 2.17. Specifically, in 1st Stage, only Part_OO and Part_OE are candidate parts. Two pixel rows are loaded simultaneously from low address to high address cycle by cycle. When all the pixels are loaded, it turns to 2nd Stage, which only requires pixels from Part_EO and Part_EE. The pixel assemble module (PA Module) combines the two rows together and outputs the assembled 16 pixels to the APPSAD architecture.

For S_{homo} case, since sub-sampling is adopted both horizontally and vertically, the pixels of different types are loaded one part by one part. So, there are four stages in all. In each stage, the pixel row of specific part is loaded from low address to high address cycle by cycle.

As for N_{homo} case, only 1 stage exists based on full pixel pattern in Fig. 2.17. As shown in Fig. 3.16, in the $2n$th clock, the pixel rows from Part_OO and Part_OE are loaded simultaneously. In the succeeding $(2n + 1)$th cycle, two rows from Part_EO and Part_EE are loaded. The whole process continues until all the pixels in the memory are loaded.

Furthermore, for H_{homo} and S_{homo} cases, the required number of pixels in each clock cycle is 8, which is half of the V_{homo} and N_{homo} cases. So, only one memory group is enabled within each stage and the power consumption of another group can be saved. Therefore, the proposed pixel organization can keep high data reuse while achieve lower memory power dissipation.

3.3.3 Compressor tree in standard cell library

The proposed APPSAD architecture can realize adaptive sub-sampling algorithm by introducing some multiplexors and optimizing previous PE array. The hardware size will also be dilated compared with original PPSAD structure. In this dissertation, by using 4-2 and 3-2 compressors to manually build up compact architecture, circuit optimization for APPSAD structure is accomplished.

In conventional standard cell library such as TSMC 0.18um, compressor tree is widely used to achieve optimum result during the compiling stage. The criterion for selecting compressor tree is flexible. The synthesis tool will follow some constraints such as timing and area criterions and generate net-list which is close to user's requirements. So, redundant adders exist inevitably in the final net-list. The cost of all these adders will dilate hardware cost with the increase of synthesis frequency. In my proposal, I use compressor tree to manually build PE array in APPSAD structure, which removes all the unnecessary adders. Figure. 3.17 gives out two kinds of compressors used in APPSAD structure. The left one is 3-2 compressor (CMPR32) and the right one is 4-2 compressor (CMPR42). The

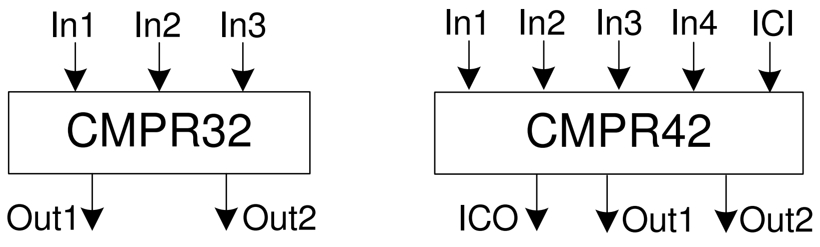


Figure 3.17: Compressors in standard cell library

ICI and ICO in CMPR42 compressor is the immediate carry-in flag (ICI) from previous compressor and the immediate carry-out (ICO) flag to the next one. The logic equations of CMPR42 and CMPR32 are shown in Eq. 3.9 and Eq. 3.10. Figure. 3.17 is an example of 1 bit-width library cell and both compressors can be extended into multiple bit-width ones based on combination of 1 bit-width cell. An example of compressing 4-bit width input data by connecting four 1-bit width CMPR42 is shown in Fig. 3.18.

$$\begin{cases} IS = In1 \oplus In2 \oplus In3 \\ ICO = (In1 \cdot In2) + (In1 \cdot In3) + (In2 \cdot In3) \\ Out1 = IS \oplus In4 \oplus ICI \\ Out2 = (IS \cdot In4) + (IS \cdot ICI) + (In4 \cdot ICI) \end{cases} \tag{3.9}$$

$$\begin{cases} Out1 = In1 \oplus In2 \oplus In3 \\ Out2 = (In1 \cdot In2) + (In1 \cdot In3) + (In2 \cdot In3) \end{cases} \tag{3.10}$$

3.3.4 Circuit optimization for single processing element

The processing elements (PEs) in APPSAD architecture will execute absolute difference (abd) operation between current pixels and reference ones. Each PE is responsible for one pixel location, which is one abd operation between two 8-bit width inputs. The intuitive PE circuit is shown in Fig. 3.19(a). It is obvious that one adder is required to generate final abd result by adding MSB (most significant bit) to the difference value. Since there are 256 PEs in one APPSAD architecture, the hardware cost of these adders is not negligible, especially when parallel processing and high speed requirement are considered

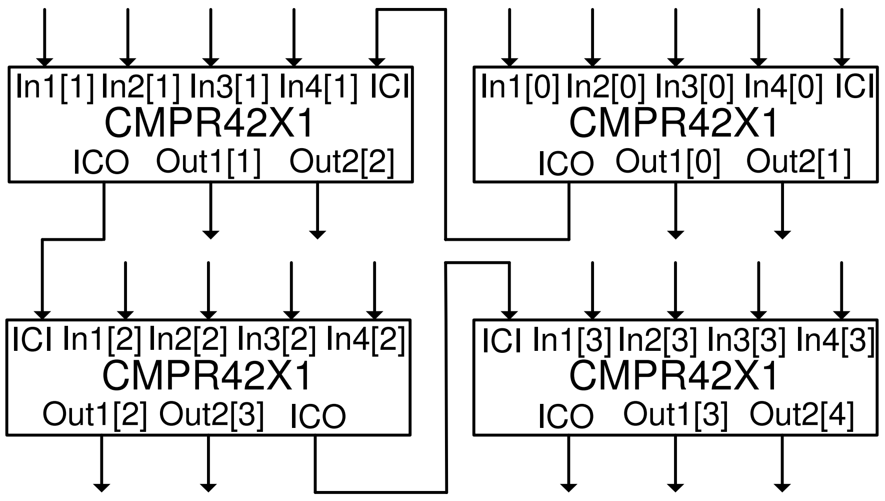
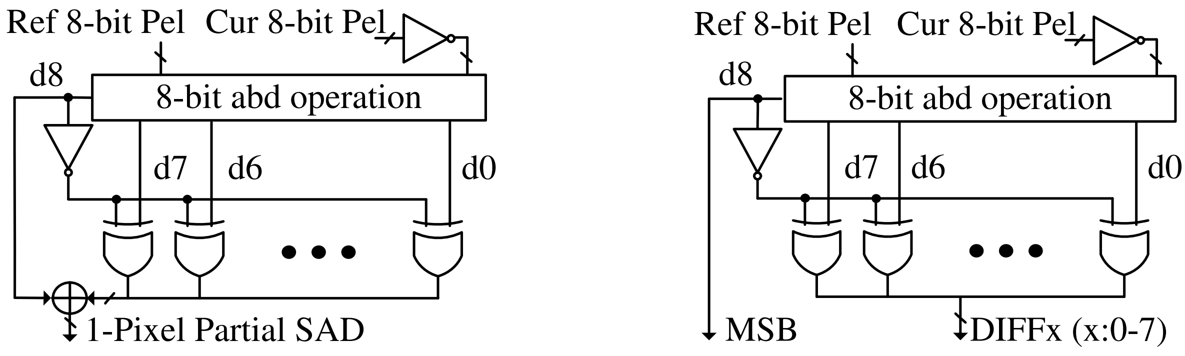

In1[1] In2[1] In3[1] In4[1] ICI
CMPR42X1
ICO Out1[1] Out2[2]
In1[0] In2[0] In3[0] In4[0] ICI
CMPR42X1
ICO Out1[0] Out2[1]
ICI In1[2] In2[2] In3[2] In4[2]
CMPR42X1
Out1[2] Out2[3] ICO
ICI In1[3] In2[3] In3[3] In4[3]
CMPR42X1
ICO Out1[3] Out2[4]

Figure 3.18: CMPR42X1 with Multiple-bits Wide Input



(a) Intuitive implementation (b) Optimized PE circuit

Figure 3.19: Optimization of processing element

(for example, HDTV application). In the optimized circuit, as shown in Fig. 3.19(b), the MSB and difference value are not added up. Thus, the specific adder within each PE is removed. For APPSAD architecture, 1-pixel partial SAD value is not the desired output result, which means that discard of adder in each PE and propagation of temporary result to next stage will not disturb the data flow of APPSAD structure. The temporary results of 8 PEs in one row of APPSAD are accumulated together. Since output of 8×1 to 8×7 partial SADs are also not a must, I use one compressor tree structure to achieve compression of these SADs. The details are shown in next section.

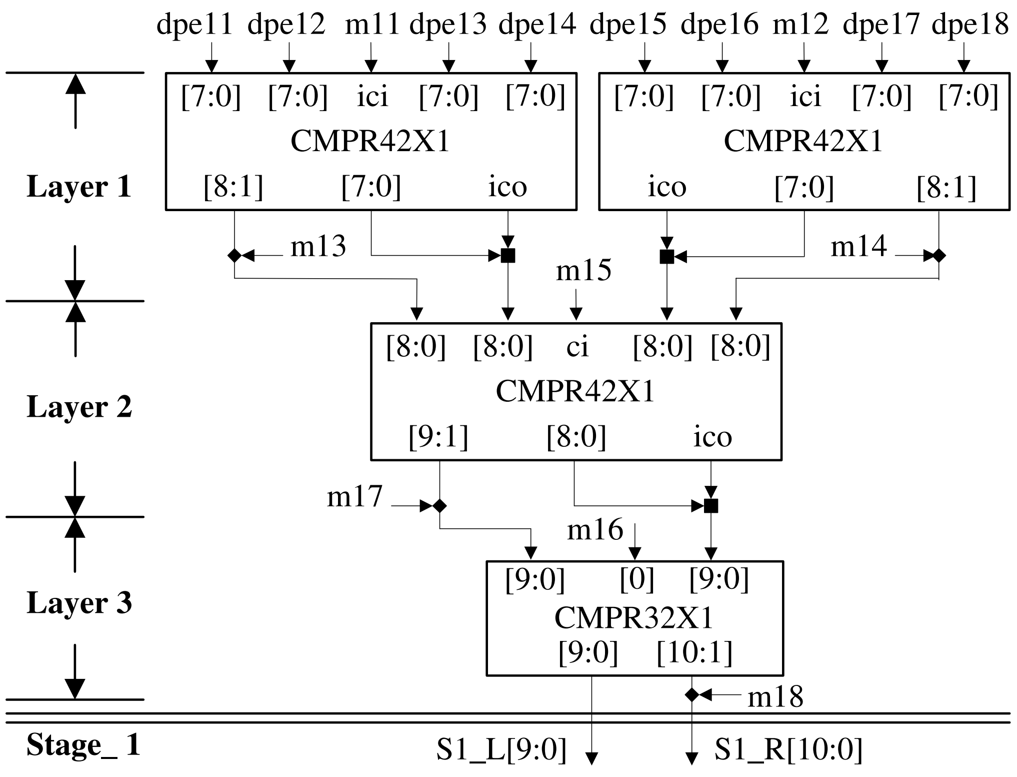


Figure 3.20: Compressor tree structure for Stage_1

3.3.5 Compressor tree based eight stage circuit optimization

As mentioned in previous section, not only the adder in each PE unit, but adder for each PE row is discarded in APPSAD architecture. Figure. 3.20 to Fig. 3.23 is the proposed eight-stage compressor tree structures. Each stage is related with each line in APPSAD structure. The detail description is as follows.

$$\begin{cases} x = 3 \;\&\&\; P = 10, & \mathit{Stage\_3\ Structure} \\ x = 5 \;\&\&\; P = 11, & \mathit{Stage\_5\ Structure} \\ x = 7 \;\&\&\; P = 12, & \mathit{Stage\_6\ Structure} \end{cases} \tag{3.11}$$

$$\begin{cases} t = 4 \;\&\&\; Q = 11, & \mathit{Stage\_4\ Structure} \\ t = 6 \;\&\&\; Q = 12, & \mathit{Stage\_6\ Structure} \\ t = 8 \;\&\&\; Q = 13, & \mathit{Stage\_7\ Structure} \end{cases} \tag{3.12}$$

For the 1st stage (Stage_1), since no temporary results are propagated from the upper stage, the structure is simple compared with rest stages. As shown in Fig. 3.20, three

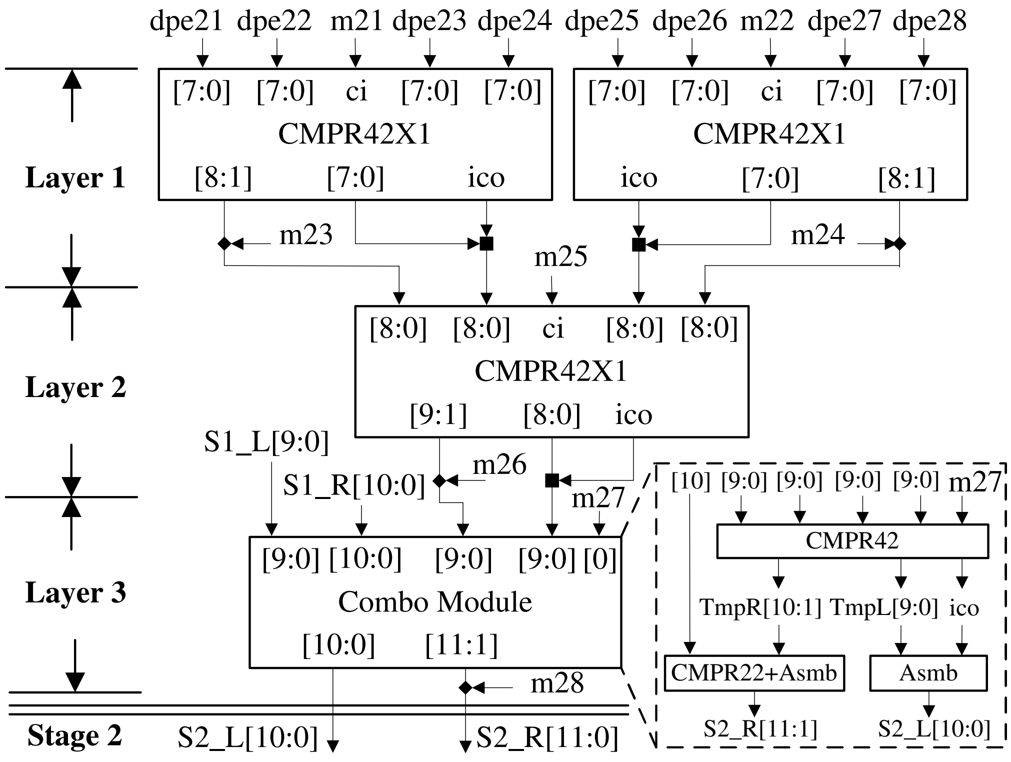


Figure 3.21: Compressor tree structure for Stage_2

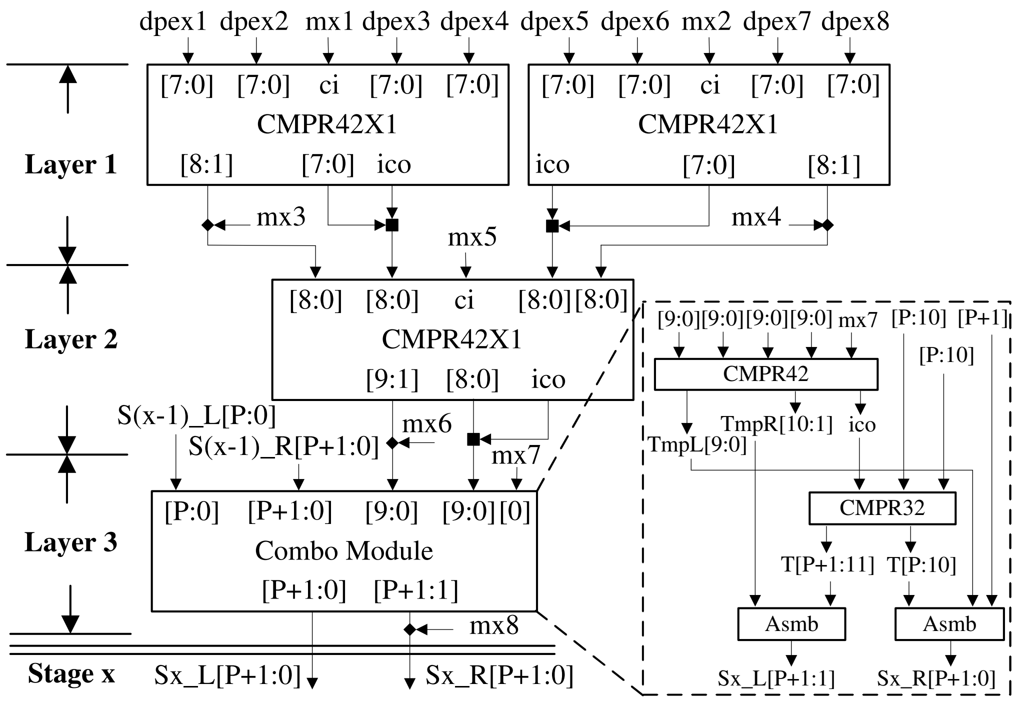


Figure 3.22: Compressor tree structure for Stage_3, Stage_5 and Stage_7

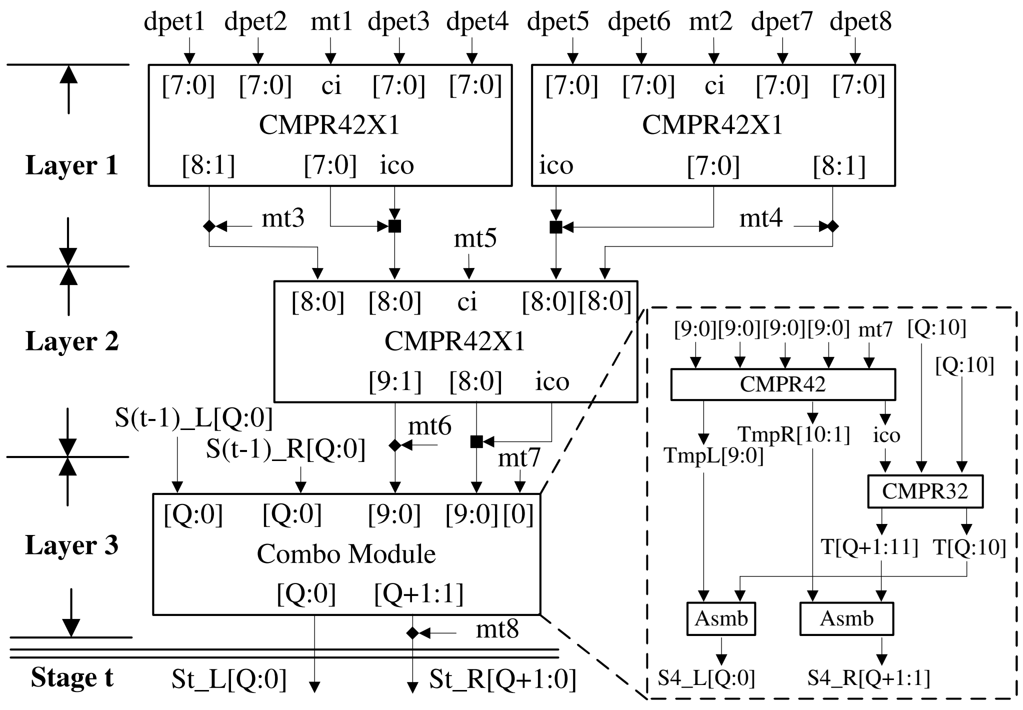


Figure 3.23: Compressor tree structure for Stage_4, Stage_6 and Stage_8

CMPR42 and one CMPR32 cells are used to generate two temporary results, namely as S1_L[9:0] and S1_R[10:0]. Here, dpe1y (y∈ [1,8]) represents difference results of 1st stage on yth column's PE. For example, dpe11 to dpe18 are difference values from eight PEs of Stage_1. The m1y is the related MSB of Stage_1. The meaning of input data for other seven stages can be traced with analogy. The square dot in Fig. 3.20 represents bit-inserted-in-head while diamond dot indicates bit-inserted-in-tail. For instance, in Layer 1, by combining m13 with CMPR42[8:1], it will form 8-bit width result [8:0]. Similarly, when ico is added to CMPR42[7:0], the result will become 9-bit width where ico is located on the top bit.

The structure from 2nd stage is different from Stage_1 because that both temporary results in current stage and results propagated from upper stage have to be compressed. Figure. 3.21 is the designed structure. Besides three CMPR42 cells, one extra combo module which consists of CMPR42 and CMPR22 cells exists in the structure. The Asmb module is introduced to assemble compressed results for output. The reason for introducing combo module is that, after Layer 2, the bit-width of input data for Layer 3 is not neat. As shown in Fig. 3.21, there are three 10 bit-width, one 11 bit-width and one 1 bit-width data to be compressed. In the proposed solution, the S1_R[10:0] is dissembled into S1_R[9:0] and S1_R[10] parts. The detail structure of combo module is shown in broken lines.

The compressor tree architecture of Stage_3 to Stage_8 can realized with two similar structures. Figure. 3.22 and Fig. 3.23 are proposed architectures. In detail, Fig. 3.22 is used for Stage_3, Stage_5 and Stage_7 compressing procedure while Fig. 3.23 represents the process of Stage_4, Stage_6 and Stage_8. The parameters setting of x, t, P and Q are shown in Eq. 3.11 to Eq. 3.12. For example, when x is set as 5 and P is 11, it is the structure for Stage_5 compressing process. The compressed results from upper layer are S4_L[11:0] and S4_R[12:0] which is the results from Stage_4 compressing structure by setting t as 4 and Q as 11 in Fig. 3.23. Therefore, it is shown that architectures in Fig. 3.22 and Fig. 3.23 are co-related to each other. Based on our eight-stage compressor tree architecture shown from Fig. 3.20 to Fig. 3.23, all the temporary adders exist in each stage are removed. One adder is used to generate final 8× SAD value.

3.4 Experiments, comparison and analysis

In this section, experiments, comparison and analysis are executed on two proposed flexible architectures. The discuss of these two structures are as follows.

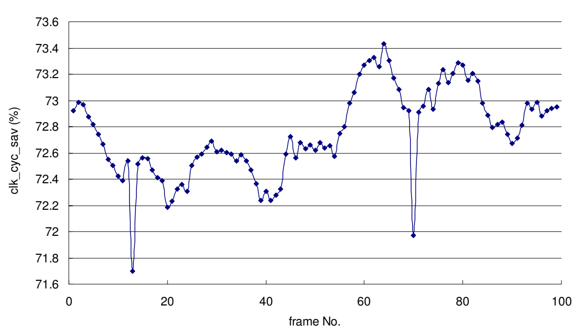
Firstly, for RSADT architecture, the target specification is set as HDTV 720p@30fps, with IPPP structure. The maximum search range is [-64,+63) in width and [-32,+31) in height with 1 reference frame. Eight parallel RSADT structures are used.

Figure. 3.24 is the clock cycle comparison between proposed structure and existing ones. Six HDTV 720p format sequences are used and I encode 100 frames under QP = 24 (quantization parameter). The required clock cycles (req_clk_cyc) for handling each frames is based on Eq. (3.13), where MB_num represents the MB numbers within one frame (3600 in our case). H_{homo}_cyc, V_{homo}_cyc, S_{homo}_cyc and N_{homo}_cyc are the req_clk_cyc for handling one H_{homo}, V_{homo}, S_{homo} and N_{homo} MB respectively. In this dissertation, the clock cycles for loading reference pixel data for each MB based on search window reuse algorithm [33] is omitted. So, H_{homo}_cyc, V_{homo}_cyc, S_{homo}_cyc and N_{homo}_cyc are 512, 512, 256, and 1024 based on 8-parallel RSADT structure. The H_{homo}_rat, V_{homo}_rat, S_{homo}_rat and N_{homo}_rat are the ratio of different type MBs within each frame. In the SADT structure [17], only the N_{homo}_rat is 1 (other ratios are all 0). The clock cycle saving (*clk_cyc_sav*) result of each frame is based on Eq. (3.14), where *clk_cyc_our* is the *req_clk_cyc* of the RSADT architecture while *clk_cyc_ori* is the *req_clk_cyc* of SADT structure. It is shown that the proposed structure can averagely save 72.75% clock cycles for sequence with abundant homogeneous MBs such as crew_720p. In case of knight-shield_720p and stockholm_720p, the clock saving is decreased to averagely 62.78% and 62.46% because of the increase of texture MBs in the image. For parkrun_720p, since the image in this sequence contains many high frequency MBs, the ratio of homogeneous MBs decreases a lot, which result in 42% clock cycle saving. Altogether, our RSADT architecture can averagely save 61.71% clock cycles while keep video quality, maintain data reuse and full utilization of hardware.

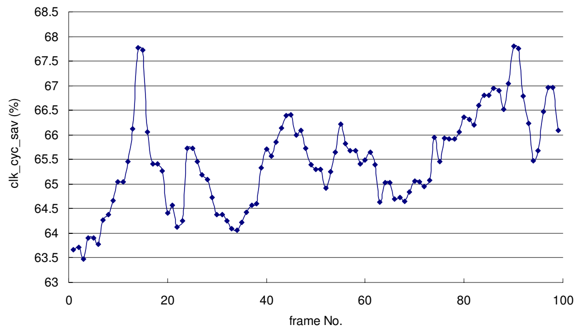
$$\begin{aligned} clk\_cyc = & MB\_num \times (H_{homo}\_rat \times H_{homo}\_cyc + V_{homo}\_rat \times V_{homo}\_cyc \\ & + S_{homo}\_rat \times S_{homo}\_cyc + N_{homo}\_rat \times N_{homo}\_cyc) \end{aligned} \tag{3.13}$$

$$clk\_cyc\_sav = \frac{clk\_cyc\_ori - clk\_cyc\_our}{clk\_cyc\_ori} \times 100\% \tag{3.14}$$

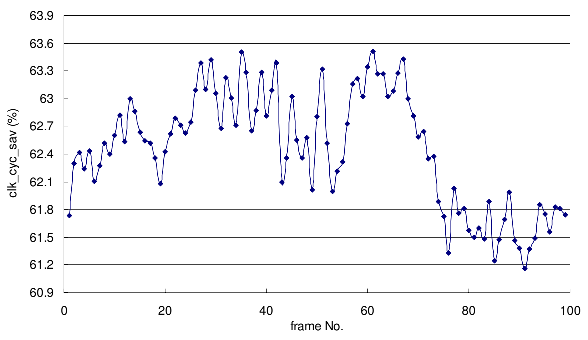
Additionally, by introducing some control logic for memory data loading and control signals for PEs, the previous SADT structure [17] can also be modified into extended version (call it [17]') for adaptive algorithm. Assume that each set of structure handles 1/8 of the search points within search window. Table 3.2 is the comparison between extended SADT and proposed one. It is shown that the extended SADT can also handle three search patterns in adaptive sub-sampling algorithm. However, the pixel data reuse (*pel_reuse*) and hardware utilization (*HW_utiliz*) can not always achieve 100%. Moreover, the *req_clk_cyc* for the extended version can be shortened (reduced from 1024 to 512) when MB's type is H_{homo} or S_{homo}. It is because in these two types, it is possible to expand one extra column in RSA and apply two-column-shift (3rd to 18th columns are shifted left to 1st to 16th columns) operations directly when column shift occurs in snake scan method. The search point on the right side of current one can be processed



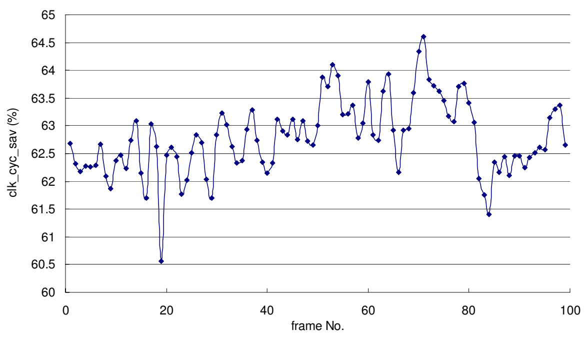
(a) clock saving of crew_720p



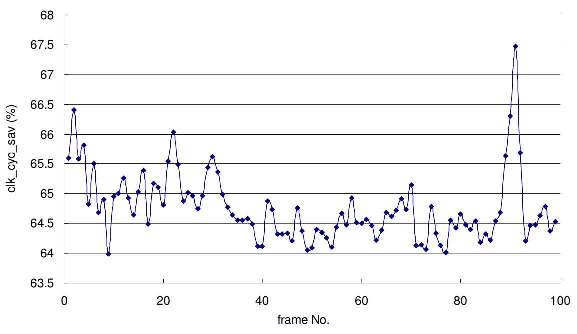
(b) clock saving of city_720p



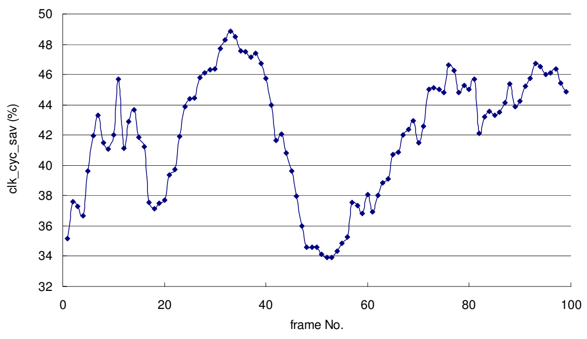
(c) clock saving of stockholm_720p



(d) clock saving of knightshields_720p



(e) clock saving of harbour_720p



(f) clock saving of parkrun_720p

Figure 3.24: Clock saving of HDTV sequences

simultaneously. However, in case of V_{homo} MB, the req_clk_cyc is still 1024 and the extended SADT can not reduce req_clk_cyc to 1/4 of original cycles in S_{homo} case. The reason is that the upper and lower pixel rows can not be skipped under original memory organization scheme. In my RSADT architecture, since data organization is applied both in memory level and architecture level, the req_clk_cyc is half of [17]'s in both V_{homo} and S_{homo} cases. As for the search point within one clock cycle (SP_per_cyc), the extended

Table 3.2: Comparison with Extended SAD Tree

MB Type	H_{homo}		V_{homo}		S_{homo}	
Architecture	[17]'	ours	[17]'	ours	[17]'	ours
pel_reuse	100%	100%	50%	100%	50%	100%
HW_utiliz	100%	100%	50%	100%	50%	100%
req_clk_cyc	512	512	1024	512	512	256
SP_per_cyc	2	2	1	2	2	4

SADT can only accomplish block matching one point by one point for V_{homo} MB. In the proposed architecture, for example, in case of S_{homo} MB, 4 search points are accomplished in one clock cycle, which speeds up the IME process by 4 times. The reduction of the clock cycles is also meaningful for power aware system. With MB feature obtained before IME stars, the processing time of IME engine can be shortened or the whole engine is set to idle after finishing its work. In this way, much power is saved for the whole system.

As for synthesis result, I pick 110.5MHz and 200MHz and compare the hardware data with existing works as shown in Table 3.3. Here, the hardware data is the sum of one set architecture and current MB module (the pixel difference calculation module is included). It is shown that the hardware cost of my design is smaller than 2-D structure [34] and a little higher than existing SADT or PPSAD architectures. However, the PPSAD architecture is not suitable for large format image because of poor parallelism. As for comparison with SADT and 1-D [35] architectures, since one more pipeline stage is inserted, our design can achieve higher speed (200MHz) than previous architectures, which result in higher PHR [20] value. The maximum work frequency is 208MHz under worst case. Moreover, the proposed architecture is flexible, which is unique to other fixed architectures. The RSADT structure can be configured for three sub-sampling patterns and still achieves full data reuse and 100% hardware utilization. In case of quarter sub-sampling, the processing capability is quadrupled. As for power consumption, the proposed RSADT has the same PE number with original SADT or PPSAD architectures. However, the processing time is greatly shortened in proposed RSADT structure. So, with normalized processing time, the final power consumption of proposed architecture is less than previous designs such

Table 3.3: Comparison of RSADT with Previous Designs

Designs	1-D [35]	2-D [34]	PPSAD [27]	SADT [17]	SADT [20]	RSADT	RSADT
Clock (MHz)	294	100	66.7	110.8	261	110.5	200
Technology (um)	0.13	0.18	0.35	0.18	0.18	0.18	0.18
PE Number	16	256	256	256	256	256	256
Area (gates)	61k	154k	79k	88.6k	151k	93.6k	104.7k
Flexibility	No	No	No	No	No	Yes	Yes
Power (mW)	573	–	737	–	484@200MHz	187	296

as [35], [27] and [20]. About 38.84% power reduction can be achieved compared with [20] under 200MHz.

Secondly, impact of parallel APPSAD structure for IME system is analyzed. The specification is the same with RSADT structure. Fifteen bottom pixel rows are added for block matching of search points on the last row. One extra pixel row is included for hardware design. The final search window size is 144×80. Figure. 3.25 is the system block diagram of IME engine based on APPSAD architecture. Eight parallel APPSAD structures are used and only one reference frame is adopted. In fact, for RSADT based system, it only needs to replace eight APPSAD structures in Fig. 3.25 with eight parallel RSADT structures. The 110.5MHz and 150MHz are picked as two synthesis frequency points and the testing result is given out in Table. 3.4. It is shown that compared with MRPPSAD architecture, the hardware cost of APPSAD is increased by 2.47% for a single PE Array with Cur.MB part and 5.37% for the whole IME engine. Compared with full mode PPSAD and SAD Tree architectures, our design still outweighs them in hardware cost because of mode reduction method. Here, compressed tree based circuit level optimization is not adopted. As for the whole IME engine under 110.5MHz work frequency, our design will incur 25k gates mainly because of pixel assemble module, and extra control logic.

Thirdly, I apply compressor tree based circuit optimization on APPSAD architecture. for one single 8×8 PE array, I synthesize it under several frequency points. As shown in Fig. 3.26, by adopting mode reduction in original design ([17]+MR), when frequency

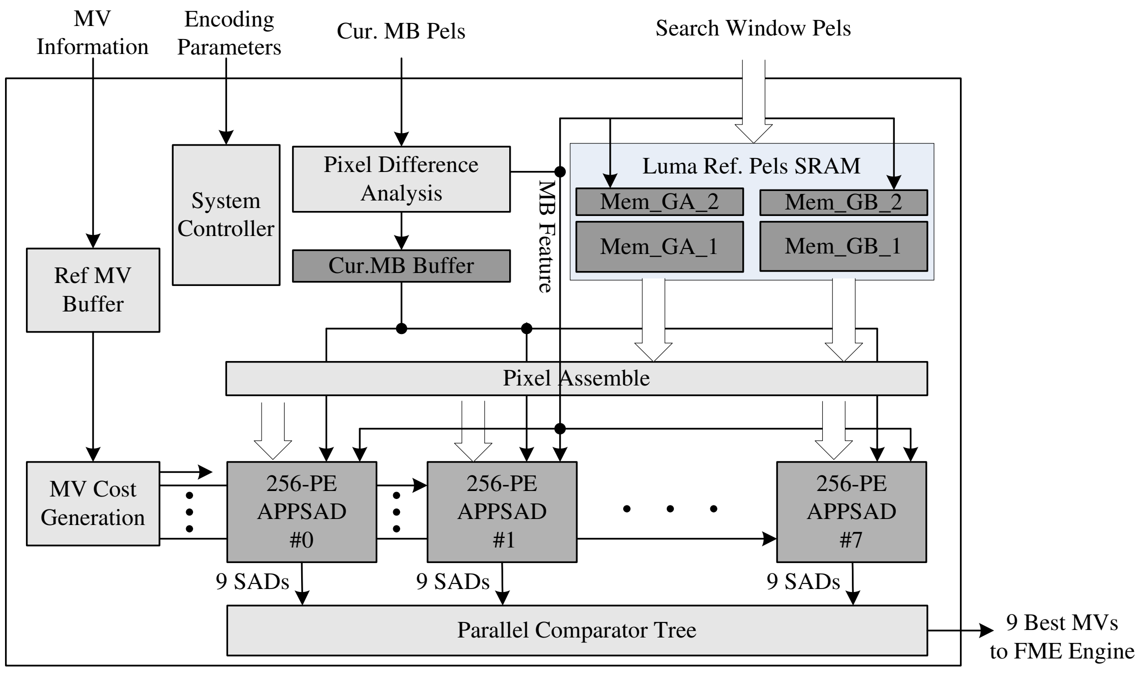


Figure 3.25: IME block diagram with APPSAD architecture

Table 3.4: Comparison of APPSAD with Previous Designs

Designs	SAD Tree [17]	PPSAD [17]	MRPPSAD [28]	APPSAD	APPSAD
Technology	0.18um	0.18um	0.18um	0.18um	0.18um
Frequency	110.8MHz	110.8MHz	110.5MHz	110.5MHz	150MHz
PE Array & Cur.MB	88.6k	81.5k	68.7k	70.4	73.3
Whole Engine	-	-	465k	490k	509k
Optimized	-	-	-	481k	498k
Flexibility	No	No	No	Yes	Yes

is less than 160MHz, about 24.4% hardware of one 8×8 PE array in [17]'s PPSAD is saved. However, the saving decreases greatly with the increase of frequency such as 180MHz and 200MHz. By further applying circuit optimization ([17]+MR+Opt), the proposed structure is superior to [17] even under high frequency points. Averagely, about 26.9% hardware can be saved. The optimized result of whole engine which consists of 8-set APPSAD architectures is shown in the second last line of Table. 3.4. About 9k and 11k hardware can be reduced for 8 parallel APPSAD architectures under 110.5 and 150MHz frequency points respectively. So the overall hardware increase of whole IME

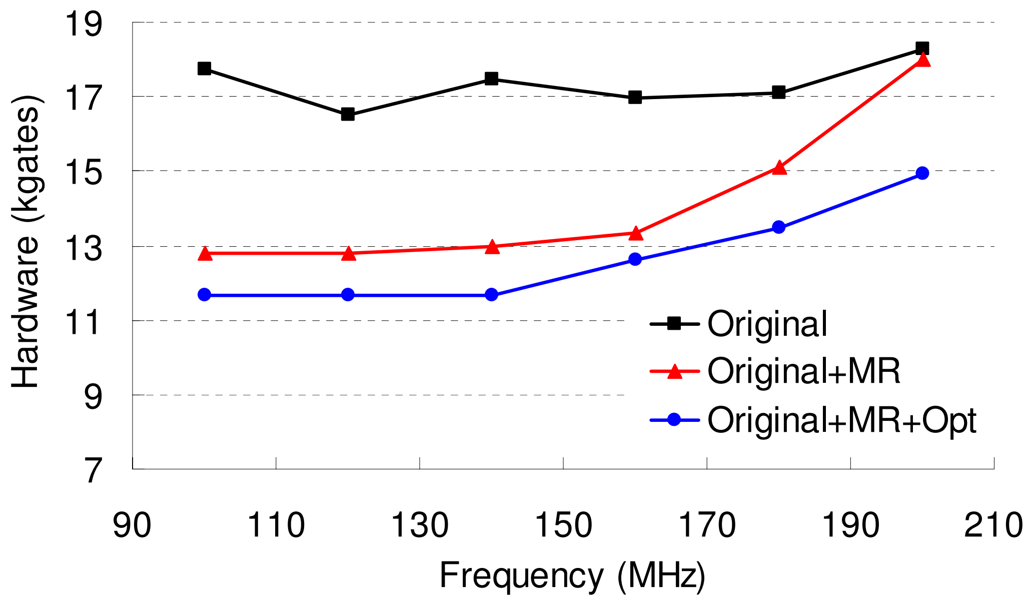


Figure 3.26: Hardware cost saving of 8x8 PE array

engine is reduced to only 3.44% compared with mode reduction based PPSAD architecture (MRPPSAD).

Fourthly, the power dissipation between proposed structure and original design is analyzed. The power of one 8×8 PE array is given out in Fig. 3.27. Since mode reduction technique reduces many redundant registers and the proposed circuit optimization discards all unnecessary adders, the whole 8×8 PE array can averagely achieve 11.7% saving of power. Additionally, I pick two typical HDTV 720p format sequences to test gate level power consumption of whole system. Figure. 3.28 is the power consumption comparison between proposed work and previous design. In order to make a clear comparison, no speed-up algorithm such as coarse-to-fine search is adopted. The power consumption of SRAM is demonstrated individually besides the whole IME's power dissipation. Since the reference pixel data is rearranged into two memory groups and only one memory group is enabled in case of H_{homo} and S_{homo} situation, the overall memory power consumption is lower than previous design which use all the memory bars. About 11.6% and 24.9% power consumption in memory part can be reduced for stockholm_720p and crew_720p. Apart from memory, the adaptive architecture can also adjust itself for MB with different homogeneous feature, which reduces power consumption in architecture level. Overall, 25.4%

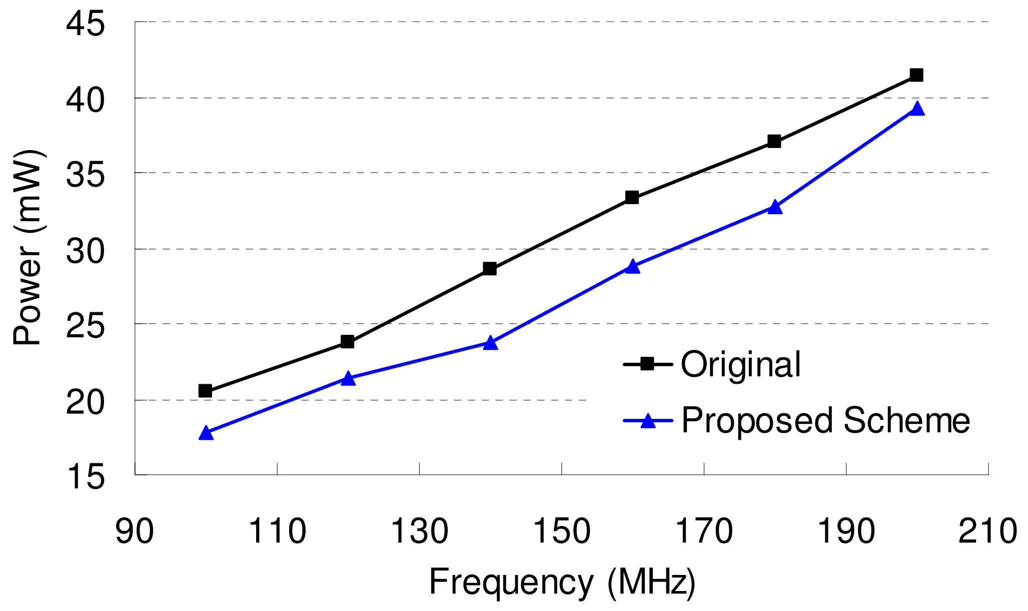


Figure 3.27: Power dissipation of 8x8 PE array

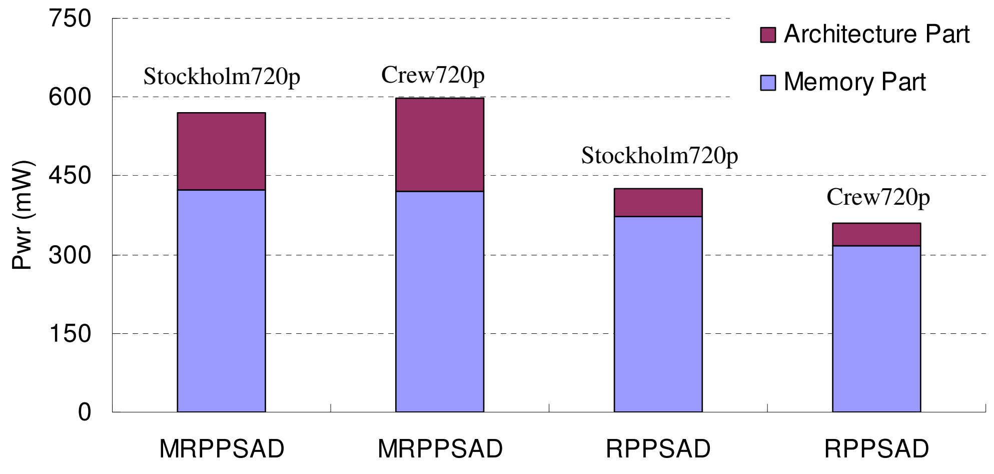


Figure 3.28: Power consumption comparison

and 39.8% power dissipation is reduced for stockholm_720p and crew_720p sequences.

Finally, in the proposed architectures, adaptive sub-sampling patterns are used for MB with different homogeneity. The complexity reduction is in the matching pattern level. Thus, the proposed scheme and architectures can be combined with other low complexity schemes. For example, the proposed hardware oriented algorithm is orthogonal to hardware algorithms such as coarse-to-fine search in [36] and frame-parallel scheme in [37]; or it can be combined with all zero block and skip mode early detection schemes [38] to further reduce complexity. From exhaustive experiments on sequences with different formats, the proposed flexible architectures can averagely achieve 53.8% reduction in power dissipation.

3.5 Conclusion remarks

In this chapter, one PDA algorithm is proposed and three sub-sampling patterns are used adaptively for different MB types. To efficiently realize adaptive algorithm, two related reconfigurable structures, namely RSADT and APPSAD are proposed. Based on different data flow of SAD Tree and PPSAD architectures, the proposed structures are optimized in different ways. Firstly, for RSADT structure, with structure level and memory level organization, the proposed architecture can averagely save 61.71% processing time with full data reuse and hardware utilization. Under normalized processing time, when comparing with previous efficient SAD Tree design, the proposed RSADT structure can achieve up to 38.84% reduction in power at 200MHz. Secondly, for APPSAD, four different processing elements are introduced for controlling of adaptive sub-sampling schemes. The interactive data loading scheme can keep full data reuse and can achieve 11.6% and 24.9% reduction in memory power consumption. Moreover, one eight-stage based circuit optimization is proposed for APPSAD structure which further reduces hardware cost and power consumption. When eight parallel APPSAD structure is applied in IME engine, with circuit optimization, the overall power saving for typical HDTV720p sequences by using APPSAD architecture is up to 39.8%. Averagely, about 53.8% power reduction can be achieved among different sequences.

Chapter 4

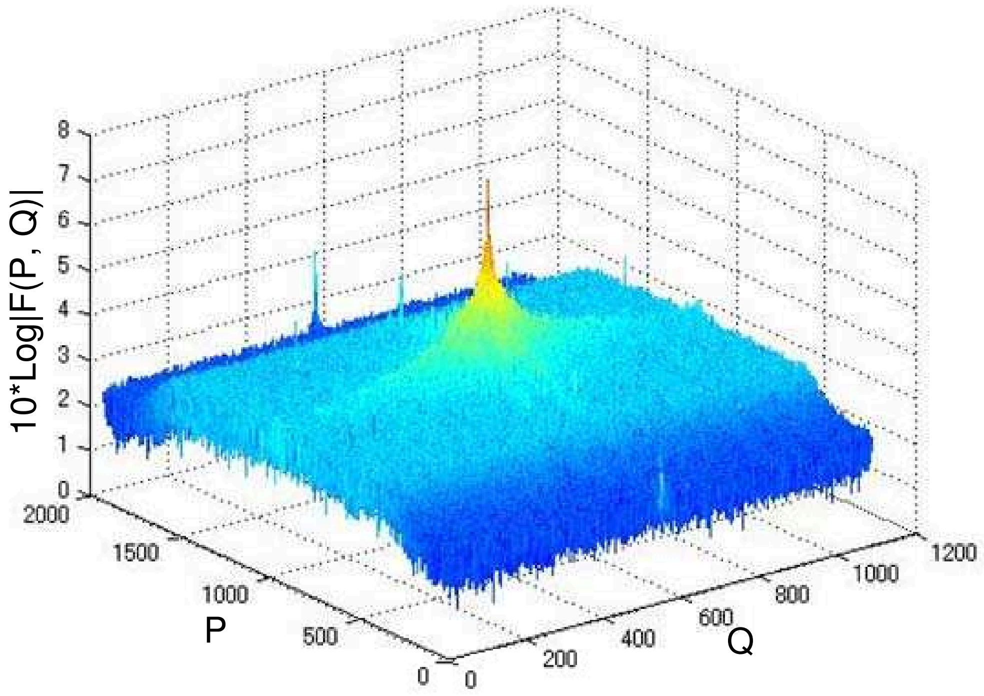
Low design effort VLSI engine for super high-vision application

4.1 Introduction

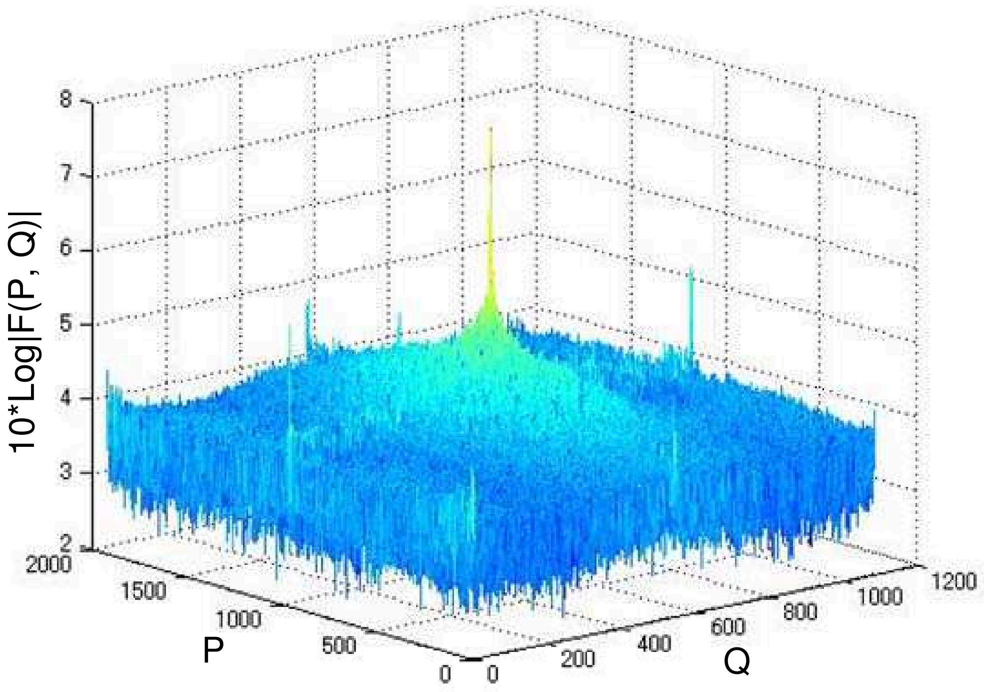
With the increasing demand of high video quality and large image size, the throughput issue for realizing real-time encoding process in ASIC design is greatly intensified. For H.264/AVC based high complexity system, besides IME engine, the FME and intra engines are also two important parts which occupy two separate pipeline stages. In this section, solutions in FME and intra parts for large image size such as 4k×4k and 4k×2k are given out. A brief introduction of FME part, intra engine and the impact of image size are given out firstly as follows.

The introduction of fractional motion estimation (FME) which is implemented with half and quarter pixel refinements contributes a lot to the video quality. As analyzed by [39], the discard of FME will cause 2-6 dB PSNR loss. With FME part, the inevitable aliasing problem [21] is greatly compensated. However, the new technique also bring about complexity problem which makes it unfavorable for hardware design. As analyzed by [6], the FME part occupies almost 40% computation, which is the second biggest one.

In hardware field, the complexity problem is directly related with throughput issue, which makes pipeline stage a must for real-time processing. In [18], the 4-stage based



(a) Pedestrian 1080p



(b) NHK sakura_tree

Figure 4.1: Spectrum comparison of HDTV1080p with SHV

real-time encoder is given out, which arranges FME engine in a single stage [40] [18]. The maximum specification of [18] is HDTV720p format. In [19] [41], one 3-stage based HDTV 1080p encoder is designed and FME also occupies one single stage. To reduce the complexity of FME engine, one fast FME engine is proposed in [42], which saves 40% hardware cost and 14% searching time. However, the video quality loss of [42] is larger than previous designs because of very few searching points, which means that the aliasing problem can not be compensated well in [42]. Moreover, even though searching points and processing units are reduced in [42], it also obeys 'first-half, then-quarter' pixel refinement procedure, which is the basic processing flow in H.264/AVC standard [2]. So, it still has long processing cycles, which is unfavorable to higher design specification such as HDTV1080p.

In 2006, the Japanese broadcaster NHK puts forward concept of Super Hi-Vision (SHV). The real SHV image is captured by special camera which can provide features such as 7680×4320@60 fps and 4:4:4 luminance to chrominance ratio. With high sensitivity video sensor, the noise generated during capturing image is increased, which further intensifies the aliasing problem and increases the importance of FME process. Figure.4.1 is the spectrum comparison with conventional HDTV1080p sequences. Here, I use SHV test sequences 'Sakura_tree' which are provided by NHK. It is shown that the high frequency components are much more abundant in SHV clips than HDTV1080p case.

Under current processing technology, it is impossible to handle raw SHV image with a single encoder. Although it is possible to divide one 8k×4k image into 2k×1k image and use 16 HDTV1080p encoder [19] to achieve real-time process. However, this adoption will cause boundary effect in the reconstructed image because of 3 horizontal boundaries and 3 vertical boundaries among 16 HDTV1080p blocks. Moreover, [19] is target for baseline profile where only forward prediction of P frame is involved. When it is extended into main profile for higher compression capability, the advent of B frame which involves forward and backward prediction will double the processing cycles. So, the design effort is greatly increased. Here, the design effort is defined as minimum required frequency (Min_Freq) for the engine, as shown in Eq. 4.1. The cyc_per_MB is the required processing clock cycles for one MB and fps is the frames to be encoded in each second. The frm_width and frm_height is the width and height of each frame. In fact, Eq. 4.1 is a direct reflection of throughput issue in hardware. Thus, when handling main profile 4k×4k@60fps, the cyc_per_MB will be three times of baseline profile (forward, backward and 1 iteration of Bi-prediction) and the final Min_Freq for [19] on SHV specification will be 9.33GHz. From existing works [18] [19], the work speed of such designs are always restricted within 200MHz. It is because that higher frequency will not only cause higher power dissipation but also incur difficulty for synthesis tools during generation of net-list. In [43], one frame-parallel based main profile encoder is proposed. The design effort is greatly decreased when handling HDTV720p@30fps case. However, when this scheme is directly extended to 4k×4k@60fps, even if the AMPD2 algorithm [40] is adopted and only forward and backward prediction are considered, the proposed FME engine in [43] will still need 5.18GHz work frequency to fulfill the throughput requirement.

$$Min\_Freq = \ cyc\_per\_MB \times fps \times \frac{frm\_width \times frm\_height}{256} \tag{4.1}$$

Another important part is low design effort intra prediction engine. For bit rate reduction, temporal prediction offers a strong impact on the final bit stream and many works have been done to reduce complexity of motion estimation. As for image quality, intra frame plays a more important role. With more intra frames in the encoding structure, the video quality is obvious improved. Thus, many researchers still focus on the refinement of intra prediction in both software [22] [44] [45] and hardware [46] [47]. In [22], edge gradient is utilized to filter out unpromising modes and about 60% intra frame encoding time can be saved. Literature [44] uses both entropy and edge information for further reduction of candidate modes. The improvement to [45] is about 8% on average. In [46], the fast intra prediction algorithm is achieved by analyzing dominant edge strength and one dedicated VLSI engine is designed. Literature [47] gives out the whole intra engine which support full prediction modes.

In [31], one four stage real-time encoder is designed and intra prediction (IP) engine is separately arranged in one single stage. For the whole IP engine, the most significant part is the intra predictor generation. As listed in [47], in one 4×4 sized sub-block, there are totally 30 cycles required for generating predictors of all intra 4×4 prediction modes (I4MB) and 10 cycles for intra 16×16 modes (I16MB). Since sixteen 4 × 4 sub-blocks exist in one MB, the total cycles will around 640. Although fast algorithms such as [22] [44] can achieve reduction of candidate intra mode to some extent, full support of all modes in hardware is a must to keep the video quality. In the worst case, all the prediction modes are required for the system. In [46], the fast algorithm is implemented in hardware and it serves as a pre-process for the IP engine. However, no optimization on intra predictor generation is mentioned. For example, when remaining candidate 4×4 modes are DC, mode4, mode5, mode6; and candidate 16×16 modes are mode3 and DC, still 384 cycles are required for generating all these intra predictors within one MB. Moreover, the minimum required frequency (Req_Freq) for predictor generation will determine the design effort for the whole engine. According to Eq. 4.1, When the specification is extended to Full HD (1080p) or 4k×2k@60fps, the existing sequential generation method in [47] will cause extreme high design effort(1.24GHz), which is impossible to be accomplished.

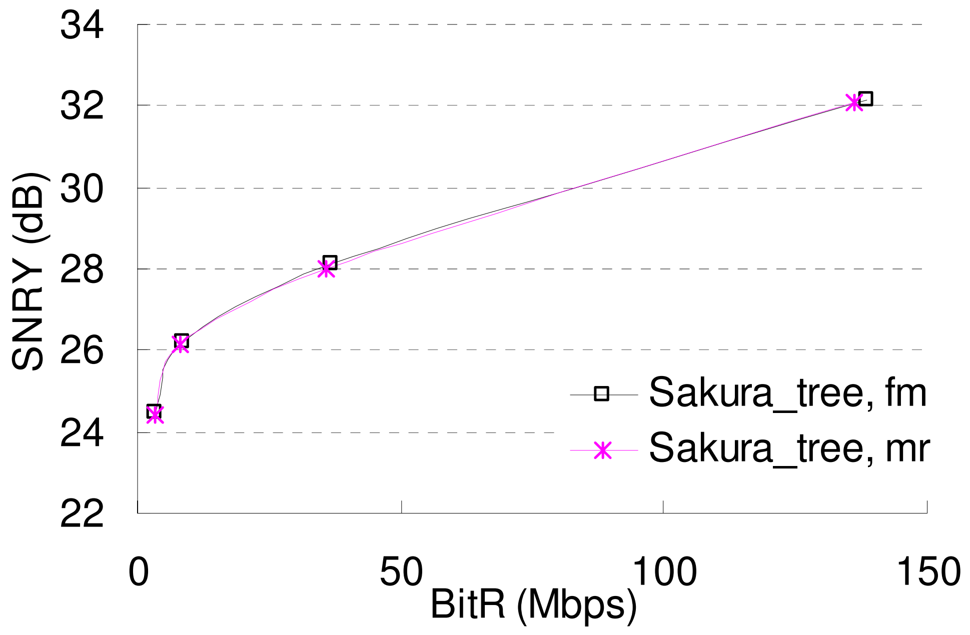
In this chapter, solutions for low design effort FME and intra engines are given out. For FME engine, I fully utilize the existing techniques and contribute one main profile FME engine for SHV 4k×4k@60fps. Firstly, based on the existing works, two algorithms namely mode reduction based mode pre-filtering and motion cost oriented directional one-pass schemes are proposed to reduce design effort and achieve hardware cost reduction. Secondly, two parallel improved schemes called 16-pixel (16-Pel) based processing and MB-parallel scheme are proposed to enhance the performance. Thirdly, to save memory access, a unified pixel block loading scheme is proposed and memory organization is applied on MB-parallel scheme. As for intra engine, one low design effort intra predictor generation engine is given out. By analyzing the data dependency among 4×4 sub-blocks, one 2-block parallel processing flow is proposed. Compared with original 1-block sequential way, about 37.5% processing time can be saved. Secondly, for the predictor generation structure, one dedicated fully utilized hardware architecture is proposed, which simultaneously generates predictors of all the I4MB and I16MB modes. So, the number of processing cycles for each 4×4 sub-block are further reduced. The details of these two parts are described in the following sections.

4.2 Low complexity fractional motion estimation algorithm

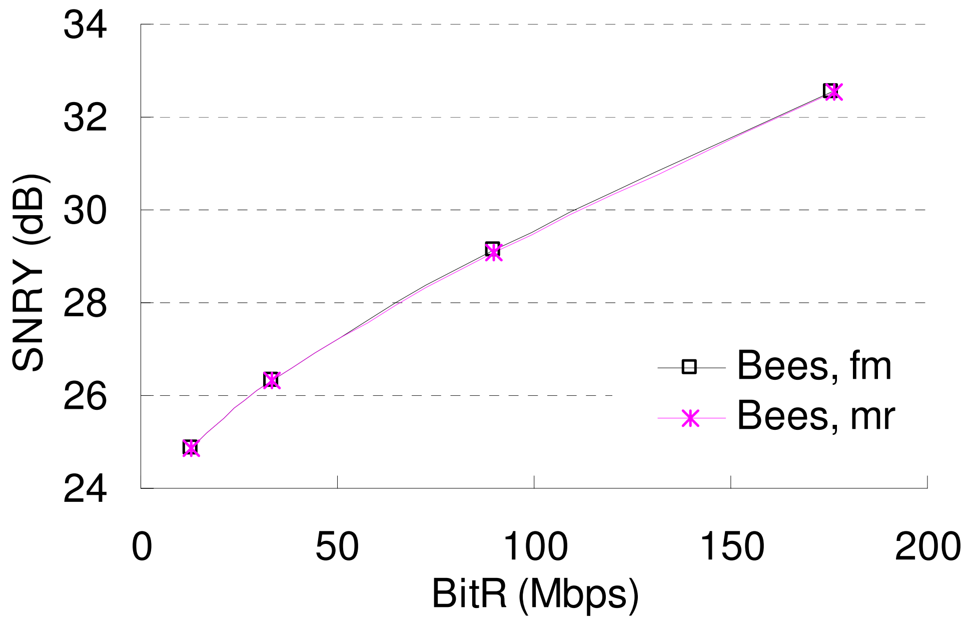
4.2.1 Mode reduction based mode pre-filtering scheme

The conventional FME engine consists of two major steps, that is, interpolation and SATD (sum of transformed absolute difference) calculation. In [40], it firstly gives out a high data reuse FME architecture. Based on the 1-D 6-tap FIR and 4×4 based processing unit (PU), the whole interpolation and SATD operations are executed simultaneously, and the FME process can be finished with about 1600 cycles. Since the interpolation is executed twice because of half and quarter pixel refinement, the total circles for one MB's processing is quite long. Moreover, the exhaustive operation among all the inter modes also deteriorates its performance when extended to large image size. To solve the problem, an optimized advanced mode pre-decision (AMPD2) algorithm is given out in [40] . However, this optimization still has throughput problem for SHV application.

In SHV case, the impact of small inter modes is very limited. In my work, I adopt mode reduction technique and discard refinement of inter modes below 8×8. Although removing small inter modes will cause some quality loss, the computation saving is significant. About 51.92% interpolation cycles can be saved when only focusing on modes above 8×8. The reduction in processing cycles also leads to improvement of design effort. For example, when no extra proposals and algorithms are adopted, full mode FME engine based on [40] will result in 19.40GHz design effort while introducing mode reduction scheme into SHV FME engine can reduce design effort to 9.33GHz. The quality comparison of encoding with full modes and modes above 8×8 is shown in Fig. 4.2. It is obvious that the mode reduction (mr) technique will cause negligible quality loss compared with full mode (fm) case. Thus, the mode reduction based mode pre-filter (MRMPF) algorithm is given out, as shown in Fig. 4.3. It means that after IME on 16×16, 16×8, 8×16 and 8×8 mode, the 9 integer motion vectors (IMVs) are merged into four MBs and check the integer motion cost (IMC) of them. The IMC of mode below 16×16 is shown from Eq. 4.2 to Eq. 4.4. During the FME stage, I only focus on the first two modes whose IMCs are smaller than



(a) Comparison of SHV sakura_tree



(b) Comparison of SHV bees

Figure 4.2: Impact of mode reduction on SHV

other two modes (for example, 16×8 and 8×8 modes of Fig. 4.3 based on Eq. 4.5). So, in worst case, 48% clock cycle saving can be achieved compared with AMPD2 algorithm in [40].

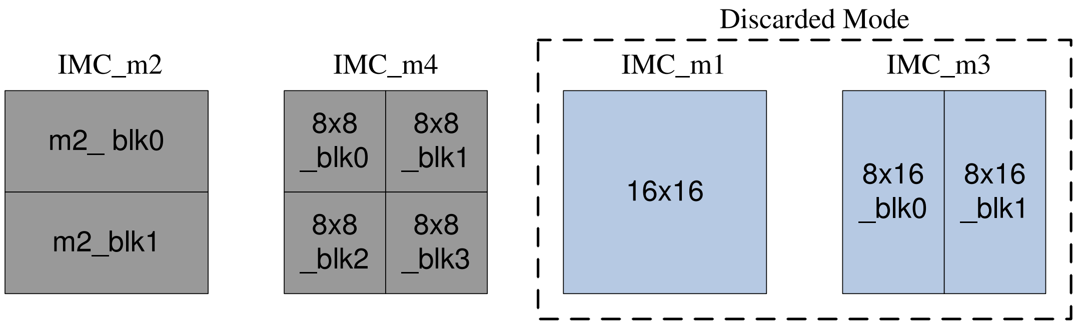


Figure 4.3: Mode reduction based mode pre-filtering scheme

$$IMC\_m2 = IMC\_m2\_blk0 + IMC\_m2\_blk1 \tag{4.2}$$

$$IMC\_m3 = IMC\_m3\_blk0 + IMC\_m3\_blk1 \tag{4.3}$$

$$IMC\_m4 = \Sigma_{i=0}^{i=3} IMC\_m4\_blki \tag{4.4}$$

$$IMC\_m2 < IMC\_m4 < IMC\_m1 < IMC\_m3 \tag{4.5}$$

4.2.2 Motion cost oriented directional one-pass scheme

Although MRMPF scheme can shorten the clock cycle for FME process, the optimization is far from enough considering the specification of SHV. In the AMPD2 and MRMPF algorithm, they both follow the 'first half, then quarter refinement' flow, which applies interpolation twice for one MB.

In [48], it gives out a one-pass algorithm which handles half pixel and quarter pixel interpolation simultaneously. So, 50% processing time is saved. However, the number of PUs in one set of FME engine is increased from 9 to 25, which results in surge of hardware cost. For SHV case, the adoption of multiple sets of engine is a must for high throughput requirement. In case of 4 parallel sets, the required PU number will be 100 based on algorithm in [48]. In this dissertation, I fully exploit information of neighboring integer pixels and proposes a motion cost oriented directional one-pass scheme (MCDOP).

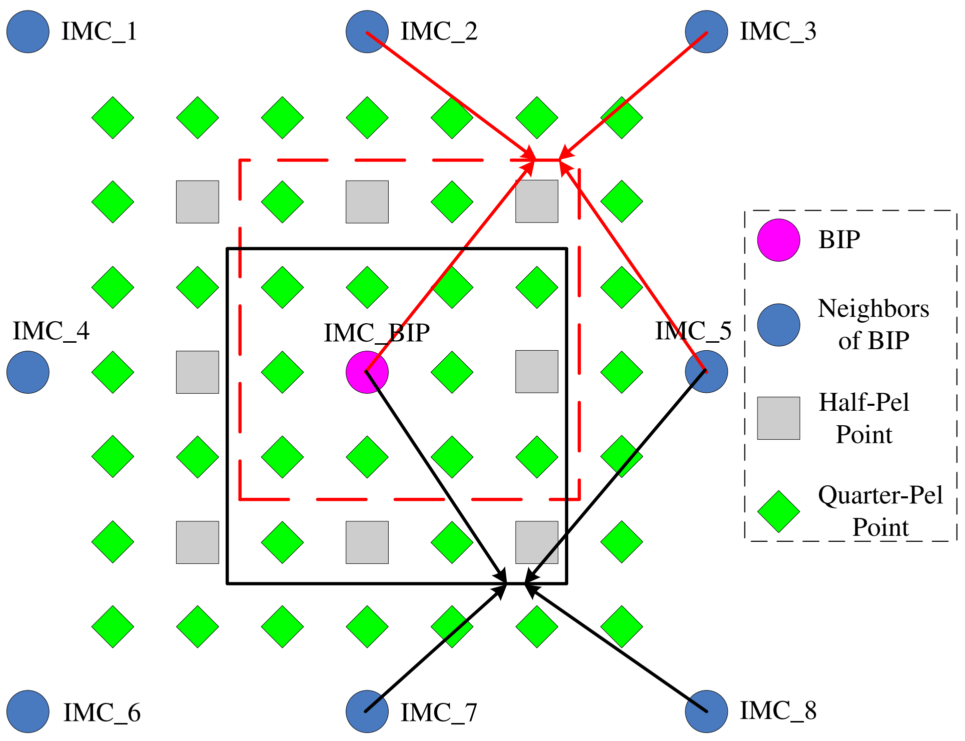


Figure 4.4: Motion Cost Oriented One-pass Scheme

The proposed scheme is shown in Fig. 4.4. It means that before the interpolation is executed on the best integer point (BIP), I analyze the IMC of BIP's neighbors (IMC_1 to IMC_8). In my work, as shown in Eq. 4.6 to Eq. 4.9, I calculate the sum of three IMC on four corner parts and get left-up IMC (IMC_LU), right-up IMC (IMC_RU), bottom-left IMC (IMC_BL) and bottom-right IMC (IMC_BR), respectively. The moving window which consists of candidate search points is selected based on the motion cost analysis. For example, as shown in Fig. 4.4, when IMC_RU is the minimum one, then search points within red broken lines will become candidate points. In case of IMC_BR, the points within black solid lines will be our candidate ones. Since I use integer motion cost to decide moving window and the half and quarter pixel refinement are handled simultaneously, the proposed algorithm is a motion cost based directional one-pass scheme. Moreover, compared with original scheme in [48] which always focus on the centering 25 search points, the required processing units number is reduced from 25 to 16 based on the motion cost feature. So, 36% hardware cost is reduced for one set of engine.

$$IMC\_LU = IMC\_1 + IMC\_2 + IMC\_4 \tag{4.6}$$

$$IMC\_RU = IMC\_2 + IMC\_3 + IMC\_5 \tag{4.7}$$

$$IMC\_BL = IMC\_4 + IMC\_6 + IMC\_7 \tag{4.8}$$

$$IMC\_BR = IMC\_5 + IMC\_7 + IMC\_8 \tag{4.9}$$

Moreover, to further reduce the hardware, 1/4 sub-sampling technique is introduced in the proposed hardware design. Based on quarter sub-sampling scheme, SATD generation will be executed with interval of 1 pixel both horizontally and vertically. So, for each processing unit (PU) in hardware, 75% hardware cost is reduced. The detail description will be given in section 4.3.1.

4.2.3 Overall hybrid schemes

The pseudo codes of JM FME algorithm and my proposed low complexity algorithm is shown in Fig. 4.5. I use JM 11.0 version and the modification to the JM algorithm is marked with italic font. The parts with broken lines represent MRMPF and MCDOP schemes. Firstly, the MRMPF scheme reduces number of IMV for FME process. Instead of loop all the 41 MVs, the proposed scheme can keep IMV number between 3 to 6. Secondly, the original two-step refinement is replaced with our MCDOP scheme. So, the half pixel and quarter pixel are constructed simultaneously. The two-step refinement turns to one-pass way, which saves 50% clock cycles. Moreover, the cost oriented adaptive window selection can reduce 36% PUs and 1/4 sub-sampling scheme further achieve 75% hardware cost reduction for each PU. The quality comparison of my algorithm to original JM one is given out in Fig. 4.6. It is shown that the modification to the JM original full mode (fm) algorithm will cause 0.2 dB quality loss in SHV Sakura_tree clip. In case of SHV clip of Bees, the quality loss is negligible. The merit of proposed algorithm is that it can reduce original long processing cycle [40] from 1664 to 224. When no hardware parallel schemes are adopted, the proposed algorithm can decrease the design effort from 19.40GHz to 2.61GMHz. Although complexity reduction can be achieved in algorithm

```
Loop Reference Frame
   Loop 41 IMVs
        Half Pel Interpolation
        SATD Calculation
        Update Best Half Pel Point
        Quarter Pel Interpolation
        SATD Calculation
        Update Final Best Point
   End Loop
End Loop
```

(a) Original FME algorithm

```
Loop Reference Frame                              MRMPF
   MR based Mode Pre-filtering
   Loop IMVs of Candidate Mode
        Analyze Neighboring IMC
MCDOP   Directional One-pass Scheme
        SATD Calculation
        Update Final Best Point
   End Loop
End Loop
```

(b) Proposed FME algorithm

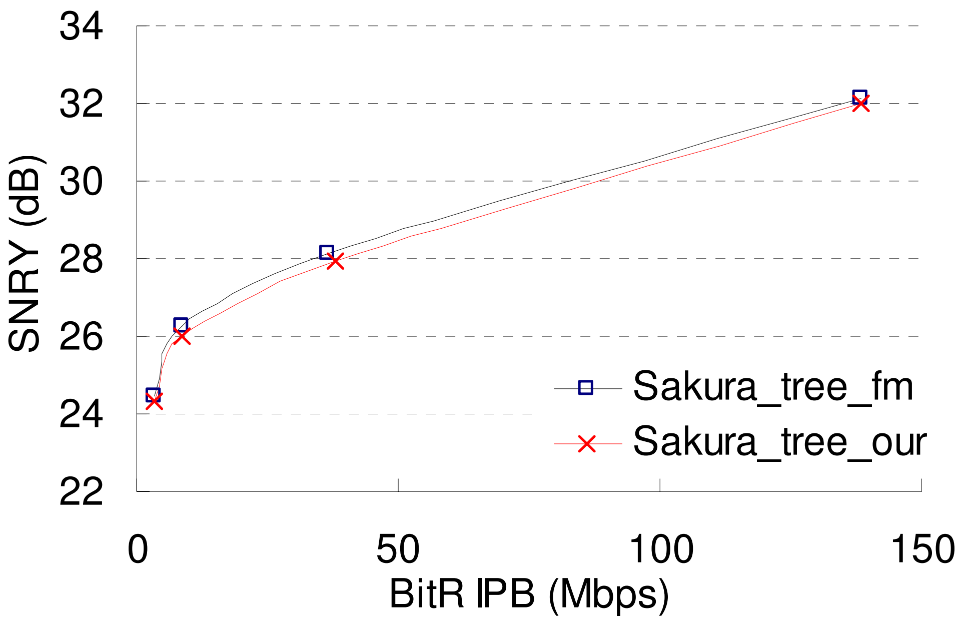
Figure 4.5: Pseudo codes of FME algorithm

level, the minimum required frequency is still very high. So, parallel schemes in hardware architecture is also a must for SHV design.

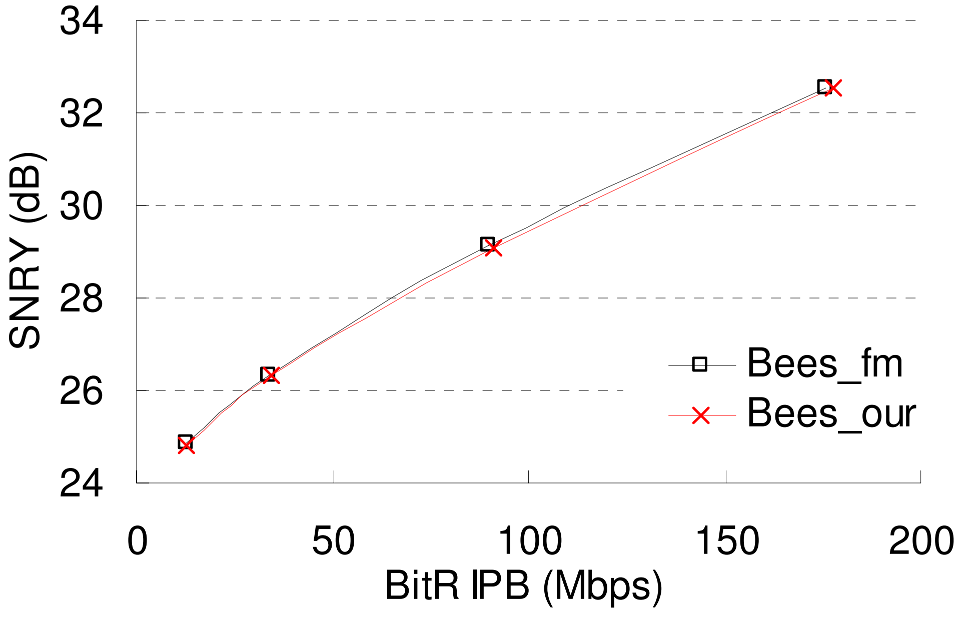
4.3 Architecture level parallel improved schemes

4.3.1 Parallel improved 16-Pel processing

From the previous sections, the design effort of SHV FME engine is reduced to 2.61GMHz. In order to achieve hardware engine with reasonable design effort, parallel processing is required in the architecture level. In the previous designs, all the processing units and interpolation engine are 4×4 based. In this disseration, I propose a 16-Pel parallel interpolation and SATD calculation, as shown in Fig. 4.7. For 16×8 and 16×16 cases, they are just the extension of previous 4×4 interpolation process. In each clock cycle, one row containing 22 pixels is loaded. Altogether, the required pixels for 16×8 and 16×16 mode are 22×14 and 22×22, respectively. For 8×8 and 8×16 cases, I handle interpolation of two 8-Pel-width block simultaneously. Although the interpolation process is the same with other two modes, more pixels are required for these two modes. The reason is that the motion vectors for these two modes are discontinuous. In the worst case, if no pixel overlapping exists between two sub-blocks, the maximum pixels required for parallel interpolation of 8×8_blk0 and 8×8_blk1 are 28×14 pixels. For 8×16_blk0 and 8×16_blk1



(a) SHV sakura_tree



(b) SHV bees

Figure 4.6: RD curve comparison

parallel processing, 28×22 pixels are required in the worst case.

Once pixels are loaded from SRAM row by row, the required pixels will be distributed to two 8×8 based parallel interpolation engines. For example, as shown in Fig. 4.7(a) and

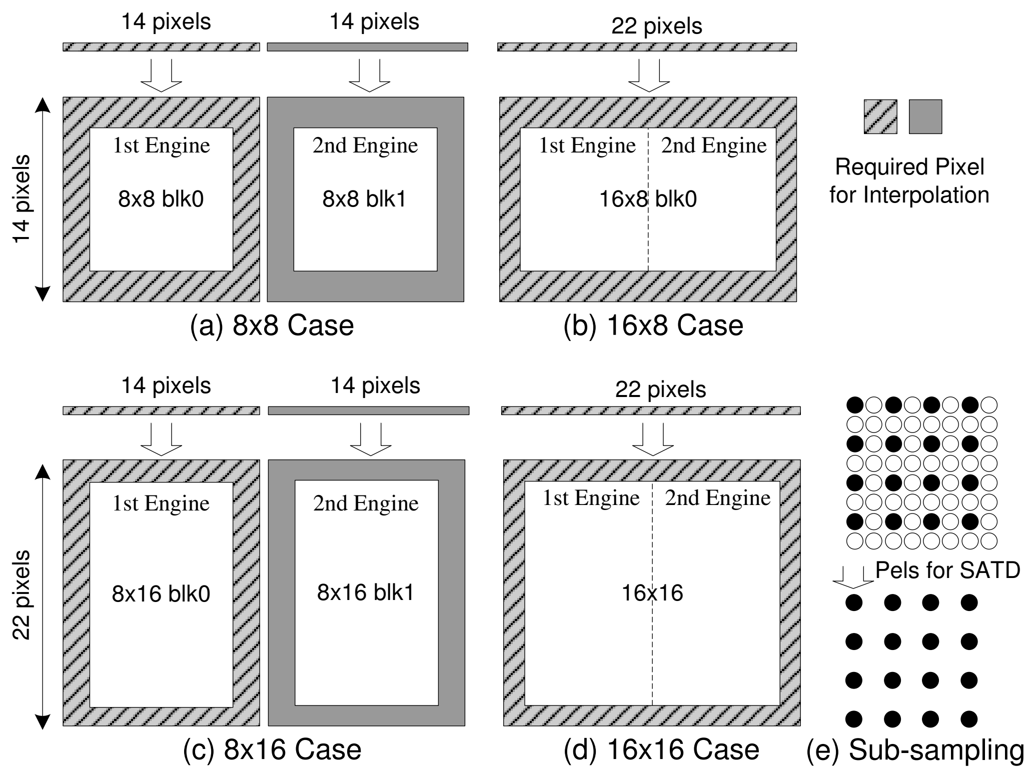


Figure 4.7: 16-Pel interpolation process

(c), 14 pixels with diagonal lines are distributed to one 8×8 based parallel engine and the other 14 pixels with grey color are for the second engine. As for Fig. 4.7(b) and (d), since the two MVs in each sub-block are continuous, the 22 pixels are divided into two 14-Pel parts with overlapping of centering 6 pixels (1st to 14th pixels are for 1st engine and 9th to 22th pixels are for 2nd engine). So, the overlapped pixels are reused in case of Fig. 4.7(b) and (d). In fact, for Fig. 4.7(a) and (c), I also propose a unified pixel block loading scheme to enable pixel reuse if the two 8×8 MV are close to each other. The detail description is shown in following section.

In the next stage, the interpolated pixels are propagated to PU groups for SATD calculation. Since the interpolated pixels are of 16-Pel width, the SATD calculation also should be 16-Pel width for parallel processing. In that case, two 8×8 sized PUs are required for handling SATD calculation of two parallel engines in Fig. 4.7. Considering the MCDOP algorithm, the total hardware cost of PU group will be dilated significantly. Thus, as shown in Fig. 4.7(e), I apply 1/4 quarter sub-sampling technique for each 8×8 sub-block. In detail, each pixel within 8×8 sub-block is used to represents its neighboring three pixels. Compared with original 8×8 sized PU, 75% hardware cost is reduced in our sub-sampled PU. In all, by introducing 16-Pel parallel interpolation and SATD calculation, the minimum required frequency is reduced to 290MHz.

4.3.2 MB-parallel schedule

In [43], a frame-parallel scheme is introduced and the same reference frame is reused for interpolation. In our design, the IBBP encoding structure is also adopted and I further improve the parallelism by applying MB-parallel processing. As shown in Fig. 4.8, after loading of Ref_0 pixels, I apply MRMPF to obtain the candidate modes for current MB on three frames (B0, B1, and P frame). Also, to save memory access, I apply unified pixel block (UPB) loading scheme to analyze IMVs of candidates mode. The detail discussion of our UPB scheme is in next section. After that, the same reference SRAM is used for parallel interpolation of current MB on B0, B1, and P frame simultaneously. So, the processing time is greatly shortened compared with original data flow which sequentially executes each interpolation. In my design, no bi-directional prediction is incurred and the final required frequency is only 145MHz for Super Hi-Vision 4×4@60fps in main profile. The detail pixel assignment for three interpolation (IP) modules is shown in the bottom right part of Fig. 4.8. It is shown that the required pixels for P frame IP module are loaded from reference SRAM directly. As for IP modules of B0 and B1 frames, their required pixels are loaded together from a unified pixel block (UPB), which is proposed to save redundant memory access. Moreover, since the pixels for UPB and P frame interpolation are loaded from the same SRAM, the memory access problem may occur. So, a parity pixel organization scheme is proposed in this paper to solve this problem. The details are discussed in the following two sections.

4.3.3 Unified pixel block loading

In section 4.3.1, it is clear that 28 different pixels are needed in each cycle under worst case for 8×8 and 8×16 modes. In fact, the worst case rarely happens due to the continuity

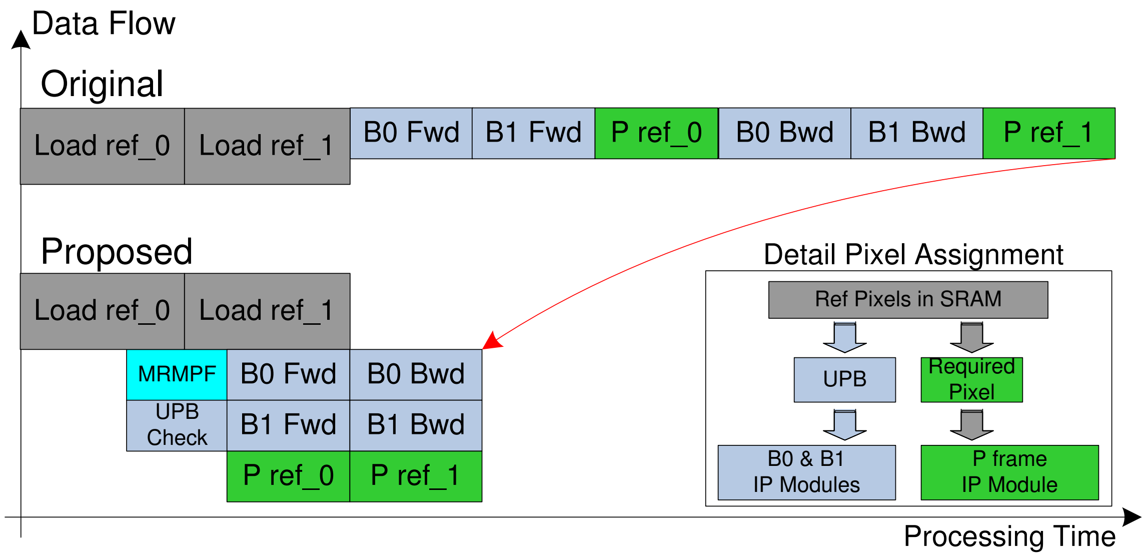
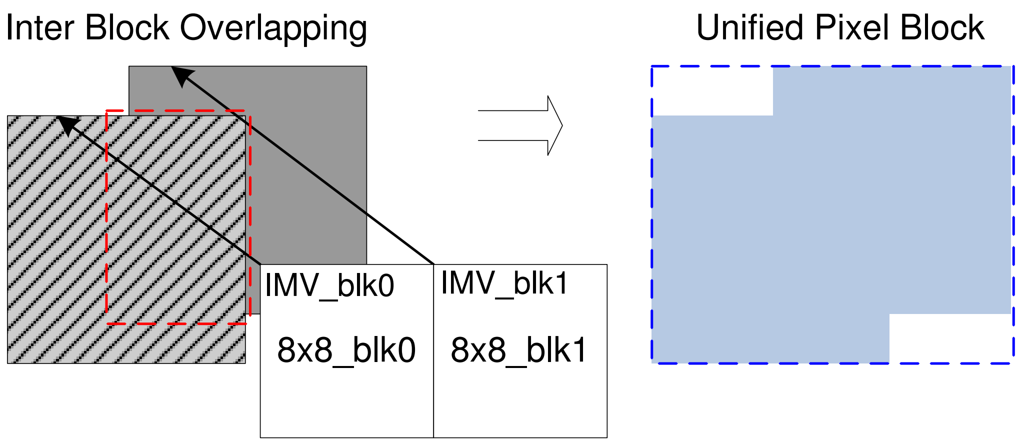


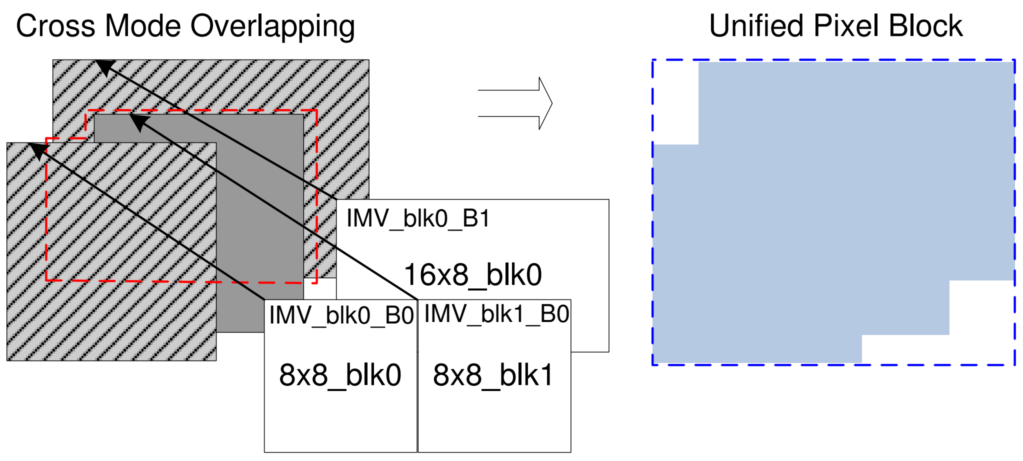
Figure 4.8: MB parallel processing schedule

of motion. In order to avoid multiple access to the same pixels, I propose a unified pixel block (UPB) loading scheme for current MBs' processing on two B frames.

Figure. 4.9 is the proposed UPB scheme. It is shown that there are two types of situations, called inner block overlapping (IBO) and cross mode overlapping (CMO). In IBO case, the redundant pixels only exist between required pixel of two blocks, as shown with red broken lines on the left side of Fig. 4.9(a). For CMO case, I assume that first candidate mode for B0 and B1 are 8×8 and 16×8, respectively. So, one 16×8 block on B1 and two 8×8 block on B0 frame are interpolated simultaneously based on our 16-Pel interpolation and MB-parallel scheme in section 4.3.1 and 4.3.2. As shown in Fig. 4.9(b), many pixels are overlapped among 8×8_blk0, 8×8_blk1 and 16×8_blk0. In fact, the overlapping situation can occur among all kinds of modes (from 16×16 modes to 8×8 modes) on both P and B frames. In order to simplify the implementation, only the candidate IMVs (remaining IMVs after MRMPF) for B0 and B1 frame after loading of ref_0 SRAM are analyzed. Then the UPB is generated and pixel rows are propagated to two parallel 16-Pel scale engines for B0 and B1 frames' processing. The blue broken lines on the right side of Fig. 4.9 is an example of final UPB region. It is true that our scheme will load some redundant pixels, as shown in white part within blue broken lines in Fig. 4.9(a) and Fig. 4.9(b). However, from the experimental results, the proportion of



(a) Inner block overlapping



(b) Cross mode overlapping

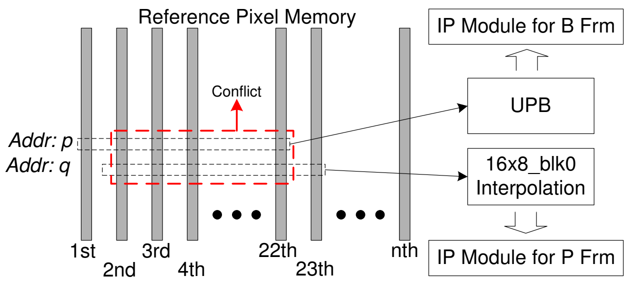
Figure 4.9: Unified pixel block loading scheme

extra access is very trivial considering the saving of memory access.

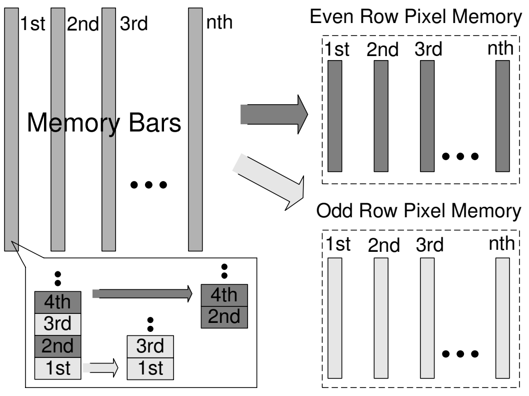
4.3.4 Parity pixel organization for parallel processing

In section 4.3.2, I propose an MB-parallel schedule to shorten the whole processing cycles. However, the simultaneous access of three engines will incur memory access problem inevitably.

Since interpolations on B0 and B1 frames obtain the pixel row from the UPB, the access problem only happens between interpolations of B frame and P frame. Assume that the reference pixel memory contains n 1-Pel width memory bars. When the interpolations



(a) Conflict in memory access



(b) Parity pixel row organization

Figure 4.10: Solution to memory access conflict

of B and P frames require pixel rows from the same memory bars, the access conflict occurs. As shown in Fig. 4.10(a), during loading of pixels in address p of 1st to 22th bar and address q of 2nd to 23th bar, the pixels from 2nd to 22th bar in address p and q can not be loaded within one cycle. So, the parallel processing is not feasible in the original memory organization. In this dissertation, I give out a parity pixel organization to solve the problem. In detail, as shown in bottom left part of Fig. 4.10(b), the odd and even pixel of each memory bar are arranged separately into two different bars. Then, the total memory bars will be divided into even-row pixel bars and odd-row pixel bars. Therefore, the access conflict in Fig. 4.10(a) can be solved. For instance, when conflict occurs, I set p as 2m+1 and q as 2m, the required pixel row for three processing engines can be loaded simultaneously from odd-row memory and even-row memory, respectively.

4.4 Low design effort architecture for H.264/AVC intra predictor generation

4.4.1 Parallel processing flow for intra predictor generation

The generation of intra predictor in hardware will incur long processing cycles for the whole system, which is the main reason of performance degradation. In [47], the whole intra predictor generation is based on the 4×4 sub-block scale. As shown in Fig. 4.11, one 16×16 MB is separated into sixteen 4×4 sub-blocks. The processing flow is based on the raster scan order because of the data dependency problem. For example, when handling 4×4 blk1 in Fig. 4.11, all the required pixels (M, I, J, K, L, A, B, C, D, E, F, G, H) are already available. All the nine intra 4×4 modes (m0 to m8) use these pixels to generate their corresponding sixteen predictors of 4×4 blk1. The generation process is based on sequential way and each mode causes 4 clock cycles. When processing of blk1 is finished, the whole system turns to blk2 as next 4×4 sized sub-block. The required pixels for blk1 (D, I1, J1, K1, L1, E, F, G, H, E1, F1, G1 and H1 in Fig. 4.11) are then available because of the best intra 4×4 mode for blk1 has already been decided and the vertical pixels (I1, J1, K1, L1) are the reconstructed pixels based on best intra 4×4 mode. There are some bubbles between handling blk1 and blk2 due to the decision of best mode and reconstruction of pixels. In [47], the bubble period is fully utilized by inserting predictor generation of intra 16×16 modes. As shown in left part of Fig. 4.11, the predictor generation of I16MB modes are also organized in the scale of 4×4 sub-block, which means that predictors of each intra 16×16 modes are obtained in 16 separate stages. I use circle marked with number to indicate each stage. It is shown that each stage of I16MB modes is arranged between the already processed 4×4 sub-block and next unprocessed one. The whole processing flow is based on sequential raster scan order, which is from left to right and top to bottom. Predictors of four chrominance (chroma) 8×8 mode is generated after luminance (luma) modes and the process is similar with luma I16MB mode. In fact, such kind of processing order is not a must, and parallel scheme can be achieved lossless.

Figure. 4.12 is the proposed processing flow. Firstly, for current MB in process, the original '16-stage' based flow is optimized into '10-stage' way. So, about 37.5% processing time is reduced. From Fig. 4.12, it is also obvious that my proposal is a lossless optimization toward original raster scan order. In the first MB, the 4×4_blk1 and 4×4_blk2 are individually processed in two stages. In the following part, the predictor generation is in the form of 2-block scale, which means that two 4×4 sub-blocks are handled simultaneously by two parallel engines. For example, in stage 3 of Fig. 4.11, the 4×4_blk3 is the sub-block in process. Since 4×4_blk1 and 4×4_blk2 are the two sub-blocks already processed, there are no data dependency problem for 4×4_blk5. So the predictor generation

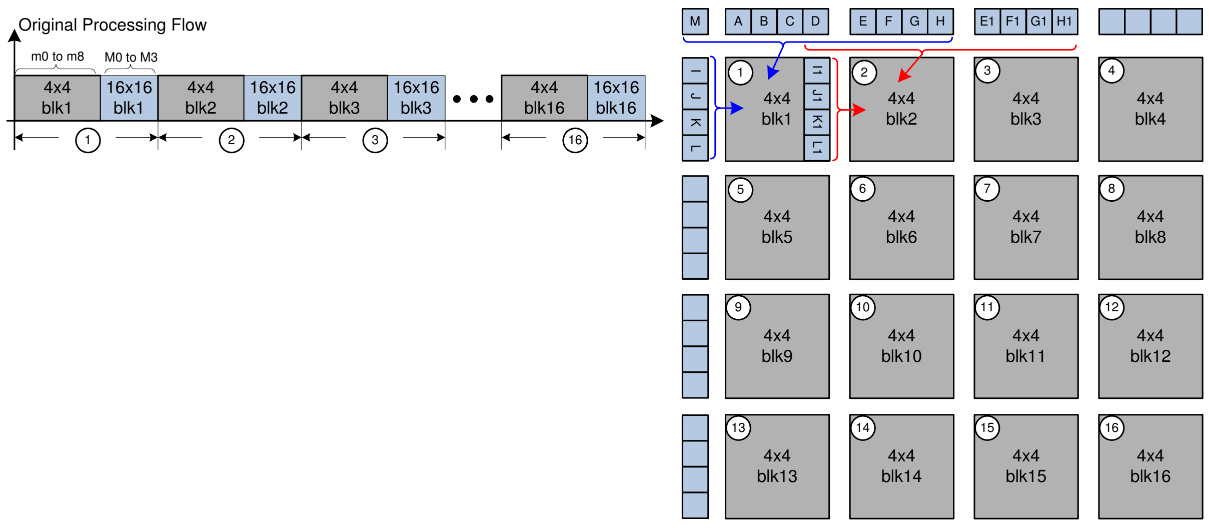


Figure 4.11: Original processing flow

of 4×4_blk5 can be executed together with 4×4_blk3 with no quality loss. Secondly, the last two stages of current MB is handled together with first two stages of the next MB, as shown in the top of Fig. 4.12. Therefore, full hardware utilization can be achieved during the whole intra predictor generation process.

4.4.2 Fully utilized parallel intra predictor generation architecture

From the above paragraph, one 2-block based parallel processing flow is proposed and 37.5% processing time is reduced. However, such adoption is also not enough to achieve low design effort engine because of the long processing cycles for handling all the I4MB and I16MB modes within one 4×4 sub-block.

In [47], except horizontal and vertical modes in I4MB and I16MB (1 cycle is enough for horizontal and vertical mode), the required processing time for rest I4MB or I16MB mode are in the period of 4 clock cycles, which means that, for one specific mode (mode of I4MB or mode of I16MB), the 16 predictors of one 4×4 sub-block can be obtained after 4 clock cycles. So, the overall processing cycles for generating all luminance predictors of I4MB and I16MB modes are 640, which occupies large proportion of computation time. When this structure is extended into Full HD or 4k×2k@60fps, the design effort

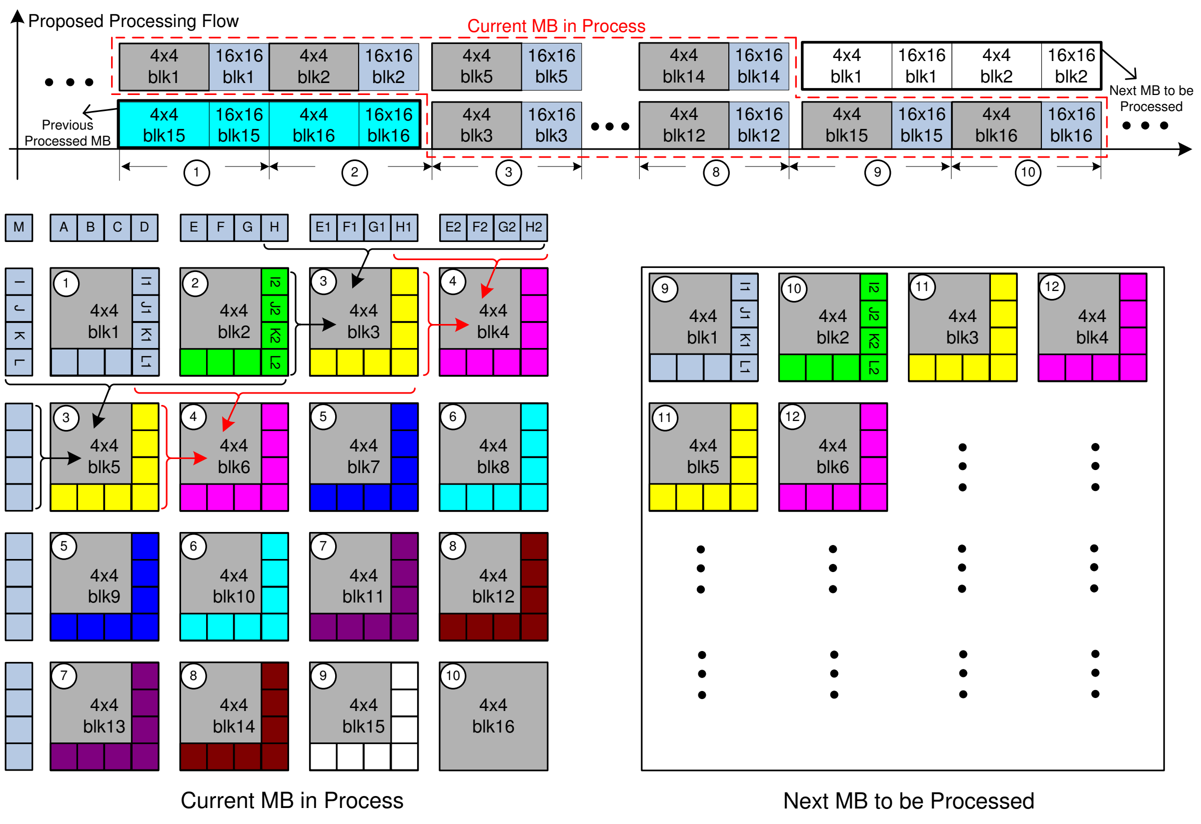


Figure 4.12: Proposed processing flow

Table 4.1: Predictors of I4MB modes in 4×4 sub-block

Pred(y,x)	V	H	DC	DDL	DDR	VR	HD	VL	HU
Pred(0,0)	A	I	Z	**A+2B+C**	I+2M+A	M+A	I+M	A+B	J+I
Pred(0,1)	B	I	Z	B+2C+D	M+2A+B	A+B	I+2M+A	B+C	K+2J+I
Pred(0,2)	C	I	Z	C+2D+E	**A+2B+C**	B+C	M+2A+B	C+D	K+J
Pred(0,3)	D	I	Z	D+2E+F	B+2C+D	C+D	**A+2B+C**	D+E	L+2K+J
Pred(1,0)	A	J	Z	B+2C+D	J+2I+M	I+2M+A	J+I	**A+2B+C**	K+J
Pred(1,1)	B	J	Z	C+2D+E	I+2M+A	M+2A+B	J+2I+M	B+2C+D	L+2K+J
Pred(1,2)	C	J	Z	D+2E+F	M+2A+B	**A+2B+C**	I+M	C+2D+E	L+K
Pred(1,3)	D	J	Z	E+2F+G	**A+2B+C**	B+2C+D	I+2M+A	D+2E+F	3L+K
Pred(2,0)	A	K	Z	C+2D+E	K+2J+I	J+2I+M	K+J	B+C	L+K
Pred(2,1)	B	K	Z	D+2E+F	J+2I+M	M+A	K+2J+I	C+D	3L+K
Pred(2,2)	C	K	Z	E+2F+G	I+2M+A	A+B	J+I	D+E	L
Pred(2,3)	D	K	Z	F+2G+H	M+2A+B	B+C	J+2I+M	E+F	L
Pred(3,0)	A	L	Z	D+2E+F	L+2K+J	K+2J+I	L+K	B+2C+D	L
Pred(3,1)	B	L	Z	E+2F+G	K+2J+I	I+2M+A	L+2K+J	C+2D+E	L
Pred(3,2)	C	L	Z	F+2G+H	J+2I+M	M+2A+B	K+J	D+2E+F	L
Pred(3,3)	D	L	Z	G+3H	I+2M+A	**A+2B+C**	K+2J+I	E+2F+G	L

Z=L+K+J+I+A+B+C+D, V: Vertical, H: Horizontal,

DDL: Diagonal Down Left, VR: Vertical Right, VL: Vertical Left,

DDR: Diagonal Down Right, HD: Horizontal Down, HU: Horizontal Up

will be increased to 157MHz and 1.24GHz, which is beyond maximum work frequency (55MHz). In fact, for one 4×4 sub-block, data reuse can be achieved among nine I4MB modes. Table. 4.1 demonstrates the calculation of all predictors under each I4MB mode. To simplify the description, the shift operations for generating final result are omitted. It is shown that the value of many predictors within same I4MB mode or across different I4MB modes are the same. For example, The predictor on (0,0)(called Pred(0,0)) in DDL mode is the same as Pred(0,2) in DDR mode. I use bold fonts to mark all the predictors with value (A+2B+C). It is obvious that five I4MB modes consist of this value. For predictors of other values (for example,(B+2C+D) and (C+2D+E)), they also occurs several times within one mode or across different modes. Thus, many operations are wasted in generating predictors of the same value based on sequential generation order. If all the repetitive operation can be saved, the processing cycles will be greatly decreased.

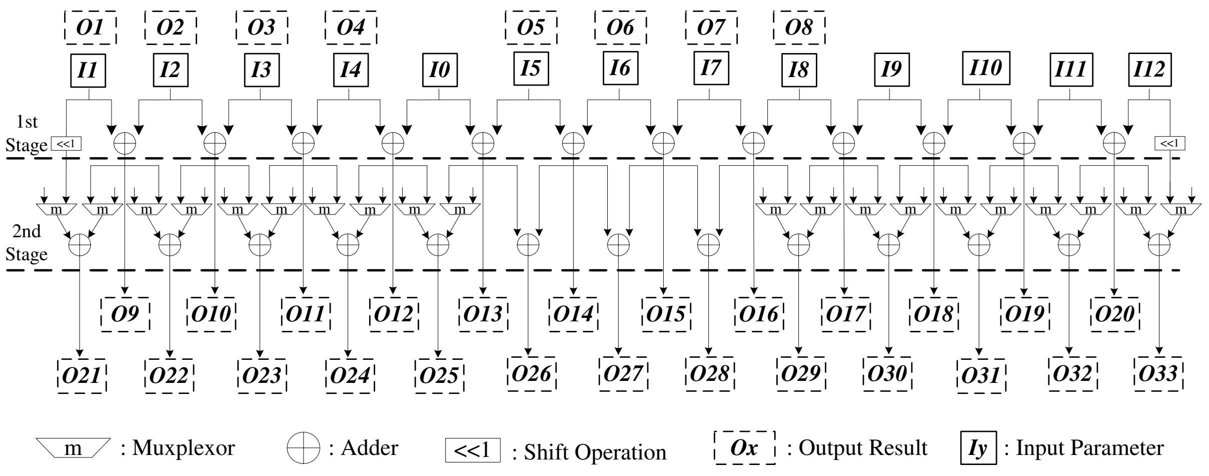


Figure 4.13: Proposed predictor generation engine

In our design, I fully enable the data reuse among all I4MB modes and propose one fully utilized parallel intra predictor generation engine.

Figure. 4.13 is the proposed parallel engine. Compared with original design [47], four features exist in our work. Firstly, after all the input parameters are ready, the predictors can be generated within two cycles. Two pipeline stage is inserted to output these results. Secondly, in the original design, many large multiplexors exist to control the type of input data and decide the candidate ones, which will incur complex control logic and long critical path. In the proposed design, I use several small multiplexors which only have two candidate inputs. So the architecture becomes more compact and easy to be controlled. Also, the proposed architecture does not consist any loop-back operations since no temporal result is required for generating the final predictor. Thirdly, instead of generating I4MB predictors sequentially (one mode by one mode), the proposed architecture works in a parallel way. Predictors of all the I4MB modes can be obtained after two clock cycles. Fourthly, the proposed architecture is also compatible for predictor generation of I16MB modes. Except DC modes, predictors of horizontal, vertical and plane modes in I16MB can also be obtained within 2 clock cycles. Moreover, for DC mode of I4MB and I16MB, I use compressor tree to facilitate the generation process. The detail descriptions are in the following paragraphs.

For I4MB modes (except DC mode), the predictors within one 4×4 sub-block can be

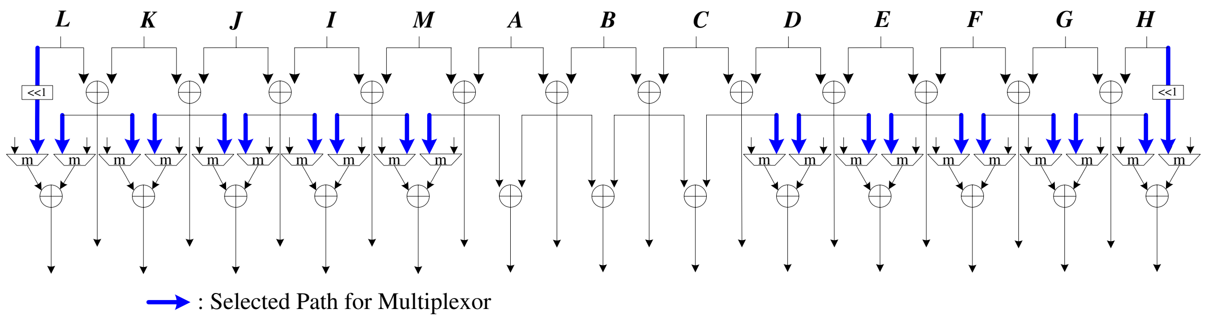


Figure 4.14: Proposed architecture for I4MB modes

obtained by configuring structure of Fig.4.13 into Fig.4.14. The bold blue arrow is the selected path. The input data of Fig.4.14 is the left, up and up-right pixels of current sub-block (for example, A to H, I to L, and M for 4×4_blk0 in Fig 4.11). The output result after two clock cycles are listed in Table.4.2. From Fig.4.14, it is shown that predictors from O1 to O8 equal to the input values; and these values are output at the 1st pipeline stage together with O9 to O20 in Fig.4.14. For rest predictors (O21 to O33), they are output and stored at 2nd stage.

For I16MB modes, the horizontal and vertical modes can be easily implemented by our architecture in Fig.4.13 because the 16 predictors of one 4×4 sub-block can be directly obtained from the input data.

As for I16MB plane mode, our architecture can also generate all 16 predictors of one 4×4 sub-blocks by using 2 processing cycles. As defined in standard, the Eq.4.10 is the calculation of plane predictor in each position (Pred(y,x)), where a, b, c are constant value for one MB and they can be calculated based on Eq.4.11 to Eq.4.13. Pel(-1,15) and Pel(15,-1) are pixels from previous MBs. The UR_w and LC_w are sum of weighted differences of upper row and left column, respectively. So, I change Eq.4.10 to Eq.4.14 to realize plane mode in our architecture. The Sd are the seed value depending on the location of 4×4 sub-block. There are four seed value namely Sd1 to Sd4 listed in Eq.4.15 to Eq.4.18. Each Sd is for one column of 4×4 sub-blocks. For example, Sd1 is used for blk1, blk5, blk9, blk13; Sd3 is used for blk3, blk7, blk11, blk15. Since the I16MB is also processed in a 4×4 block scale, the difference of blk5 to blk1 is only 4c and this value can be added during the shift operation. The configuration of I16MB mode is shown in

Table 4.2: Output predictors of I4MB modes in 4×4 sub-block

O1	O2	O3	O4	O5	O6	O7	O8	O9	O10
HU(2,2), H(3,0)	H(2,0)	H(1,0)	H(0,0)	V(0,0)	V(0,1)	V(0,2)	V(0,3)	HU(1,2)	HU(0,2)
HU(2,3), H(3,1)	H(2,1)	H(1,1)	H(0,1)	V(1,0)	V(1,1)	V(1,2)	V(1,3)	HU(2,0)	HU(1,0)
HU(3,0), H(3,2)	H(2,2)	H(1,2)	H(0,2)	V(2,0)	V(2,1)	V(2,2)	V(2,3)	HD(3,0)	HD(2,0)
HU(3,1), H(3,3)	H(2,3)	H(1,3)	H(0,3)	V(3,0)	V(3,1)	V(3,2)	V(3,3)		HD(3,2)
HU(3,2), HU(3,3)									

O11	O12	O13	O14	O15	O16	O17	O18	O21	O22
HU(0,0)	HD(0,0)	VR(0,0)	VL(0,0)	VL(0,1)	VL(0,2)	VL(2,2)	VL(2,3)	HU(1,3)	DDR(3,0)
HD(1,0)	HD(1,2)	VR(2,1)	VR(0,1)	VL(2,0)	VL(2,1)	VL(0,3)		HU(2,1)	HD(3,1)
HD(2,2)			VR(2,2)	VR(0,2)	VR(0,3)				HU(1,1)
				VR(2,3)					HU(0,3)

O23	O24	O25	O26
DDR(2,0), DDR(3,1)	DDR(1,0), VR(2,0)	DDR(0,0), VR(1,0)	DDR(0,1), VR(1,1)
HD(2,1), HD(3,3)	DDR(2,1), HD(1,1)	DDR(1,1), VR(3,1)	DDR(1,2), VR(3,2)
VR(3,0), HU(0,1)	DDR(3,2), HD(2,3)	DDR(2,2), HD(0,1)	DDR(2,3), HD(0,2)
		DDR(3,3), HD(1,3)	

O27	O28	O29	O30
DDL(0,0), VR(3,3)	DDL(0,1), VR(1,3)	DDL(0,2), VL(1,2)	DDL(0,3), DDL(3,0)
DDR(1,3), HD(0,3)	DDL(1,0), VL(1,1)	DDL(2,0), VL(3,1)	DDL(1,2), VL(1,3)
DDR(0,2), VL(1,0)	DDR(0,3), VL(3,0)	DDL(1,1),	DDL(2,1), VL(3,2)
VR(1,2),			

O31	O32	O33
DDL(1,3), DDL(3,1)	DDL(2,3)	DDL(3,3)
DDL(2,2), VL(3,3)	DDL(3,2)	

Fig.4.15. The input data and output result can be traced in Fig.4.15 and Table.4.3.

$$Pred(y,x) = (a + b \times (x-7) + c \times (y-7) + 16) >> 5 \tag{4.10}$$

$$a = 16 \times Pel(-1,15) + 16 \times Pel(15,-1) \tag{4.11}$$

$$b = (5 \times UR_w + 32) >> 6 \tag{4.12}$$

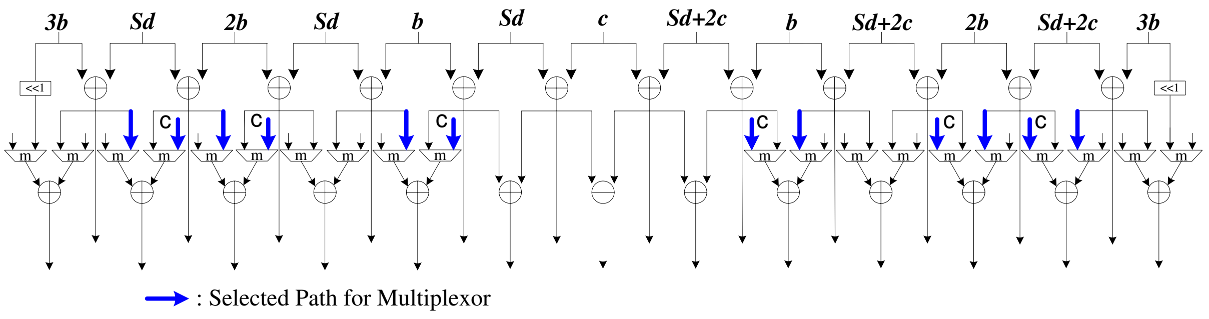


Figure 4.15: Proposed architecture for I16MB plane mode

Table 4.3: Output predictors of I16MB plane mode

O5	O7	O9	O10	O12	O14	O15	O17	O19
Sd	Sd+2c	Sd+3b	Sd+2b	Sd+b	Sd+c	Sd+3c	Sd+b+2c	Sd+2b+2c

O20	O22	O23	O25	O29	O31	O32
Sd+3b+2c	(Sd+3b)+c	(Sd+2b)+c	(Sd+b)+c	Sd+b+3c	Sd+2b+3c	Sd+3b+3c

$$c = (5 \times LC_w + 32) >> 6 \tag{4.13}$$

$$Pred(y, x) = ((Sd + x \times b) + y \times c) >> 5 \tag{4.14}$$

$$Sd1 = a + b \times (-7) + c \times (-7) + 16 \tag{4.15}$$

$$Sd2 = a + b \times (-3) + c \times (-7) + 16 \tag{4.16}$$

$$Sd3 = a + b + c \times (-7) + 16 \tag{4.17}$$

$$Sd4 = a + b \times 5 + c \times (-7) + 16 \tag{4.18}$$

$$Pred(y, x) = (\sum_{0}^{15} Ui + \sum_{0}^{15} Lj) >> 5, \;\; y \in [0, 15], x \in [0, 15] \tag{4.19}$$

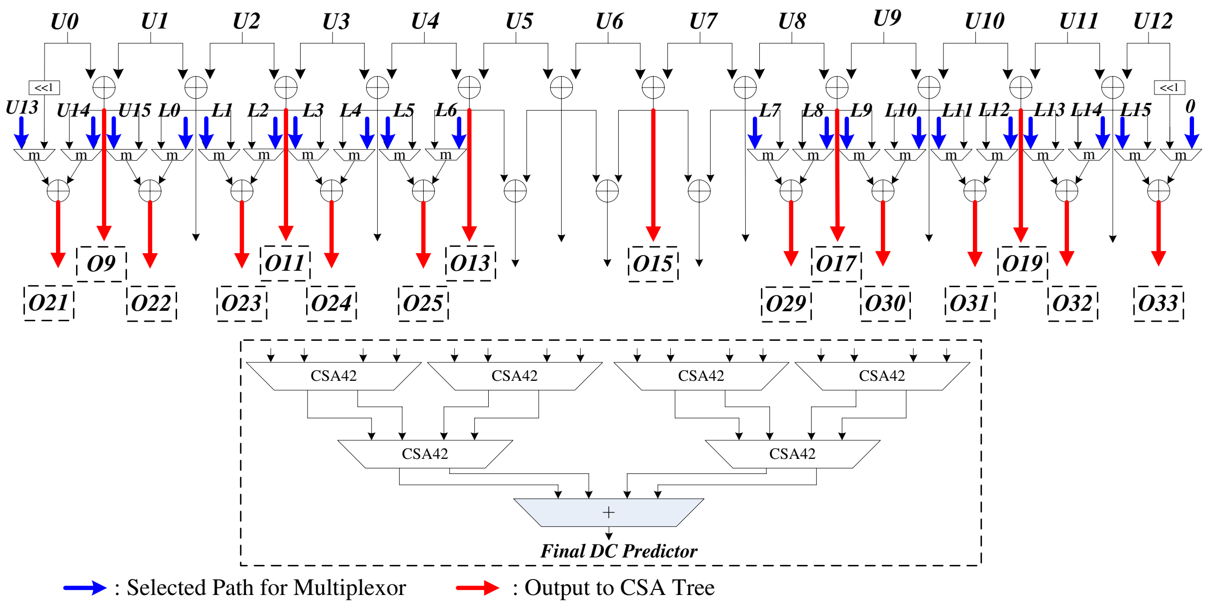


Figure 4.16: Proposed architecture configured for I16MB and I4MB DC Mode

$$Pred(y,x) = (A + B + C + D + I + J + K + L) >> 3, \;\; y \in [0,3], x \in [0,3] \qquad (4.20)$$

The DC mode in H.264/AVC is the average value of upper and left pixels. Specifically, for I16MB DC mode, the upper pixels are the last row of its upper MB; and left pixels are the column on the right of its left MB. I use U0 to U15 to represent upper pixels; and L0 to L15 for its left pixels. So the calculation of predictors in DC mode can be traced in Eq.4.19. For I4MB DC mode, it is the average of upper and left 4 pixels in the previous processed 4×4 sub-blocks. For example, in case of blk1 in Fig. 4.11, its DC predictors are calculated in Eq.4.20. For implementation, the configured structure is shown in Fig.4.16. All the 32 input pixels in I16MB DC mode are annotated on the architecture. Eight temporary output values are marked with red arrows and I use a compressor tree structure (as shown in lower part of Fig.4.16) to generate final result. For I4MB DC mode, the compressor tree structure is also enough to generate the final result. With our structure, the four chroma 8×8 modes can be realized with analogy.

4.5 Experimental result of low design effort engines

The proposed FME architecture is implemented with TSMC 0.18μm technology. Figure. 4.17 is the system architecture of our design. The search window is set as 128×64 for implementation. Based on the parity pixel organization, each SRAM part in Fig. 4.17 consists of 64 rows and 64 columns, which is 32k bits. The basic pixel width in our system is 8-bit. The input buffer of current MB is 128-bit width, which means that 16 pixels of one row are loaded in one clock cycle. The IMV buffer stores the integer motion vectors from IME engine, which is 15-bit width. I assume that the maximum IMV difference is within 16 pixels and the SRAM bandwidth for loading pixels is 44 pixels. According to Fig. 4.7(a) and Fig. 4.7(c) cases, 28 input pixels are required. So, bit width for three IP modules is 224-bit. As for PUH and PUQ modules, since quarter sub-sampling technique is used for SADT calculation, the bit width for these modules is 64-bit.

Firstly, the IMV information which comes from IME engine is analyzed in MRMPF, UPB Analysis and MCDOP modules. So, the candidate modes, UPB for B frame interpolation and search points of one-pass algorithm are determined. Secondly, there are two memory groups in our design and each group is divided based on the parity pixel organization. The pixel distributor module assembles the pixel rows loaded from memory and propagates to IP modules. Thirdly, three IP modules are incurred to handle interpolation of current MBs on B0, B1, and P frames simultaneously. The processing capability can be further enhanced by doubling the number of engines, which executes forward and backward interpolations at the same time. However, this adoption will dilate the hardware cost greatly. So, in this dissertation, I only use 3 parallel 16-Pel IP modules and apply forward and backward interpolation sequentially. Fourthly, the interpolated pixels are fed to PU groups. Since MCDOP scheme is introduced, the PUs are classified as PU for half pixel refinement (PUH) and quarter pixel refinement (PUQ). Finally, the best modes of current blocks on B0, B1 and P frames are output simultaneously. So, our FME engine operates in both MB-parallel and frame-parallel mechanism.

Figure. 4.18 shows the optimization of each schemes to the existing design. Since most real-time encoder designs have their maximum performances within 200MHz, I also set fre-

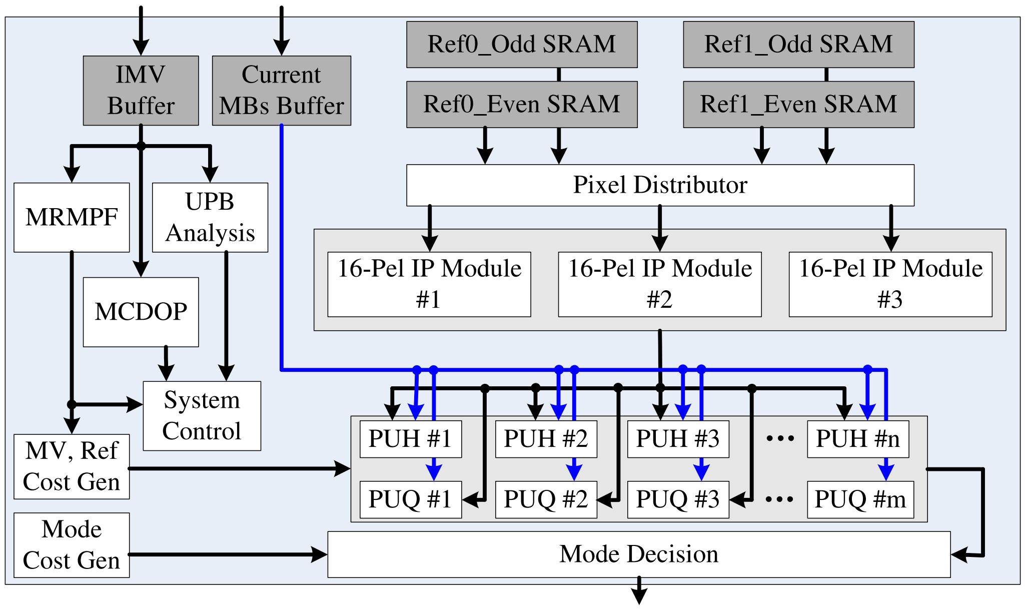
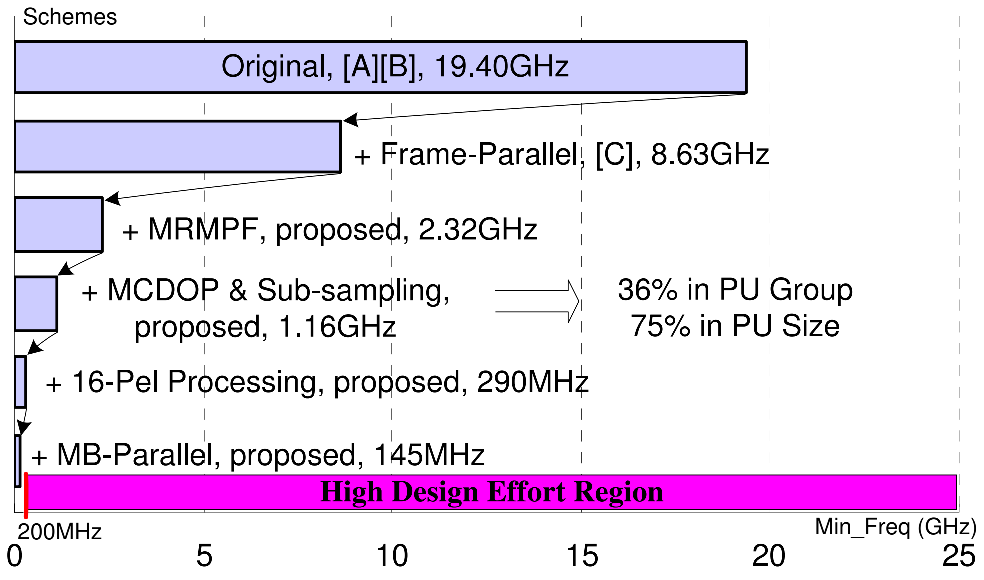


Figure 4.17: 4kx4k Super Hi-Vision FME architecture

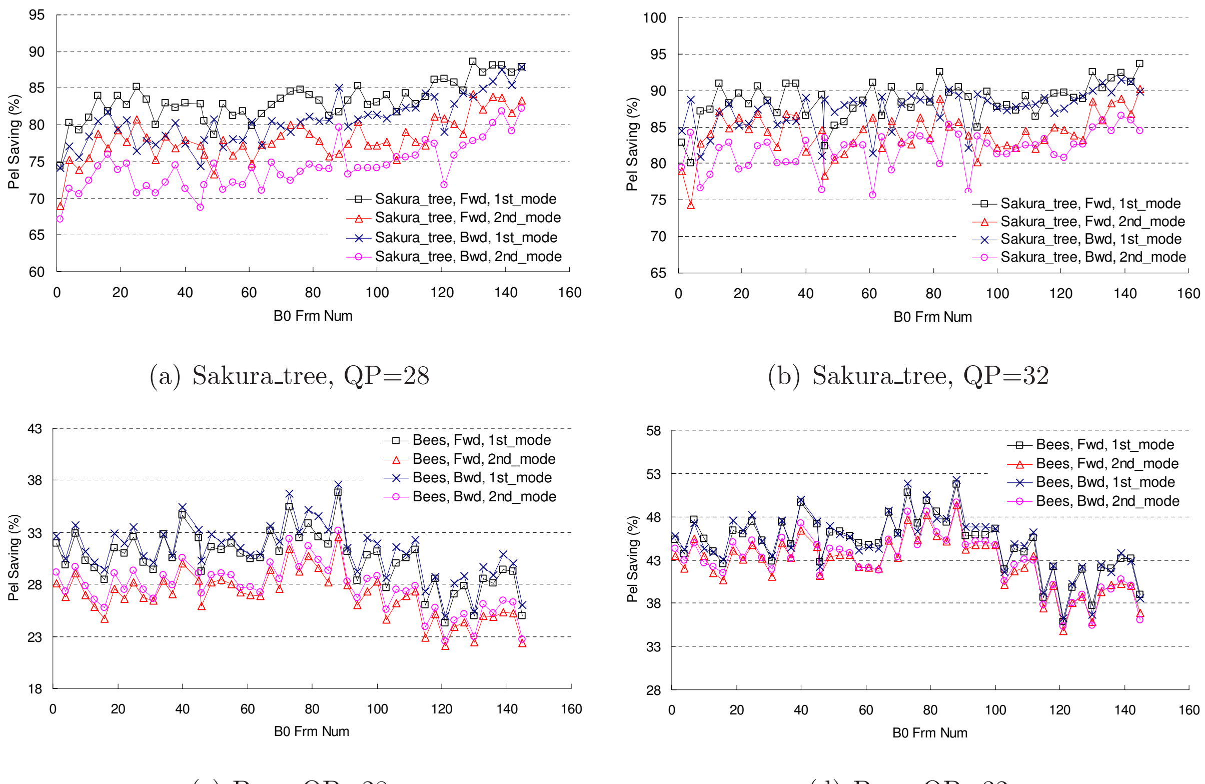
quency within 200MHz as low design effort region. As mentioned in the previous section, the direct extension of [18] to SHV specification will cause unaccomplished design effort. Even if the frame-parallel scheme in [43] is adopted, the minimum required frequency is still very high. However, based on our schemes such as MRMPF and MCDOP, the minimum required frequency can be reduced to 1.16GHz. By applying 16-Pel processing, frame and MB parallel processing, the final minimum required frequency is only 145Mhz, which is quite reasonable considering current technology. Moreover, the MCDOP scheme also helps to achieve 36% reduction of required PUs' number and sub-sampling technique will reduce 75% cost of each PU.

Figure. 4.19 shows the pixel saving ratio based on proposed UPB scheme. The X axis represents the B0 encoding frame number. Here, I give out examples of B frames encoding under QP equals 28 and 32. In the proposed FME system, the forward (Fwd) interpolations for current MB on B0 frame and its co-located current MB on B1 frame are handled simultaneously. Similarly, the backward (Bwd) interpolations are also parallel processed. For each of these two current MBs, it will have 2 inter modes to be refined based on our MRMPF scheme. The UPB scheme saves pixel loading by analyzing the overlapped part which is required for interpolation of two current MBs. The saving of


Schemes
Original, [A][B], 19.40GHz
+ Frame-Parallel, [C], 8.63GHz
+ MRMPF, proposed, 2.32GHz
+ MCDOP & Sub-sampling, proposed, 1.16GHz
36% in PU Group
75% in PU Size
+ 16-Pel Processing, proposed, 290MHz
+ MB-Parallel, proposed, 145MHz
High Design Effort Region
200MHz
Min_Freq (GHz)
0
5
10
15
20
25

A: reference [18], B: reference [40], C: reference [43]

Figure 4.18: Scheme for SHV FME engine



(a) Sakura_tree, QP=28

(b) Sakura_tree, QP=32

(c) Bees, QP=28

(d) Bees, QP=32

Figure 4.19: Pixel saving ratio of UPB scheme

Table 4.4: Hardware statistics (1.62V,125°C)

Designs	([42]+ [43])@100MHz	([18]+ [43])@108MHz	([48]+ [43])@54MHz	([41]+ [43])@200MHz	ours@145MHz
Max_Spec.	SDTV	HDTV720p	Mobile Use	HDTV1080p	SHV
Min_Freq	8.63GHz	8.63GHz	4.31GHz	4.15GHz	145MHz
Min_Freq'	2.16GHz	2.16GHz	1.08GHz	1.03GHz	
IP Modules	46.3k	71.6k	71.6k	38.7k	87.5k
PU Group	64.0k	104.5k	290.3k	200.7k	188.5k
Others	33.9k	62.0k	-	628.5	136.0k
Total	144.2k	238.1k	-	867.9k	412.0k

SHV Sakura_tree and Bees clips are shown in Fig. 4.19. I use Eq. 4.21 to calculate the saving ratio (*Pel_Sav*). The *Ld_Pel* is the pixels loaded from related SRAM based on our UPB scheme (The overlapped part for two inter modes is loaded only once). The *Olp_Pel* is the overlapped part in the original sequential processing flow. It is the clear that the proposed UPB scheme will not influence the video quality but to reduce redundant access to memory. The average saving of forward refinement in Sakura_tree is 80.68% and 86.39% when QP equals 28 and 32. In case of Bees, as shown in Fig. 4.19(c) and Fig. 4.19(d), the saving ratio is lower than sakura_tree case because of large motion in Bees clip. Averagely, about 28.67% (QP=28) and 43.70% (QP=32) pixel loading is saved in forward refinement. For backward refinement, the saving ratio is 77.41% (QP=28), 84.70% (QP=32) for Sakura_tree clip; and 29.49% (QP=28), 43.91% (QP=32) for Bees clip. With the increase of QP value, the reconstructed frame becomes smoother, which will lead to higher saving ratio for UPB scheme.

$$Pel\_Sav = \frac{Olp\_Pel}{Ld\_Pel} \times 100\% \tag{4.21}$$

The final synthesis result and comparison are shown in Table. 4.4. Since there are no existing work directly targets at SHV specification, I compare my result with the extension of previous works on SHV application, which is main profile 4k×4k@60fps. The frame-parallel scheme in [43] is applied on previous designs to enhance the processing capability. It is shown that the difference between Min_Freq for SHV and their own maximum performance is rather large, which means that extremely high design effort is required. To make a fair comparison, my MB-parallel scheme is added to these designs. So, three parallel engines are required and hardware cost under their maximum work frequency can be evaluated. The optimized design effort (Min_Freq') under our MB-parallel scheme is also shown in Table. 4.4. It is obvious that the hardware cost of ([42]+ [43]) and ([18]+ [43]) are quite small compared with others. However, both of them only employs 4-Pel interpolation scheme which incurs very long interpolation cycles. So, the design effort is still very high (2.16GHz in Min_Freq'). For ([48]+ [43]), it uses fixed one-pass algorithm which reduces Min_Freq' to 1.08GHz. However, the fixed search window in centering 25 points will increase the PU number greatly, which result in 290.3k hardware for PU group. Moreover, the Min_Freq' of ([48]+ [43]) is still far from the practical implementation. The maximum performance of ([48]+ [43]) is only 54MHz, which is only useful for small size video format such as mobile application. The performance of [41] is close to SHV application. By combining [43], the maximum performance can be kept. However, when dealing with 60fps and 4k×4k image size, even our MB-parallel processing is adopted, the Min_Freq' for ([41]+ [43]) is still very high (1.03GHz). In my design, the Min_Freq for SHV specification is only 145MHz due to the low complexity algorithm and highly parallel architecture. Although the 16-Pel based 3 IP modules (87.5k gates), 12 PUH and 36 PUQ (188.5k gates) occupy large hardware cost, my design does not require huge buffers size as in [41]. Also, the 1/4 sub-sampling technique greatly reduces hardware cost of each PU and the total cost of PU group is smaller than ([48]+ [43]) and ([41]+ [43]). The whole hardware of our FME engine is a little higher than ([42]+ [43]) and ([18]+ [43]); and much smaller than ([41]+ [43]). With 412k gates, the proposed design can achieve real-time FME processing for SHV 4k×4k@60fps. Compared with extension most recent works ([41]+ [43]), about 85.92% design effort is reduced. Take hardware cost into consideration, about 93.31% estimated power is reduced.

The proposed intra predictor generation structure is also synthesized with TSMC 0.18um technology under worst case condition. In this dissertation, I only focus on the intra predictor generation and propose a fully utilized structure. The synthesis result is shown in Table. 4.5. Since two parallel engines are used in our design, the hardware cost of proposed design is larger than previous one. However, considering the whole encoder design, 30k gates is not a significant value. The merit of our architecture is very obvious. Since no complex multiplexor is used in our design, the whole architecture is highly pipelined with simple and regular structure. The maximum work frequency is about 4

Table 4.5: Experimental result and comparison

Design	[47]	ours
Technology	0.18um	0.18um
Gate Count	12945	30112
Max Freq.	55MHz	200MHz
Max Spec.	SDTV@31fps	4kx2k@60fps

Table 4.6: Comparison of processing cycles for one 4×4 sub-block

Design	[47]	ours
I4MB DC	4	1
I4MB rest modes	26	2
I16MB DC mode	4	3
I16MB rest modes	6	3
Totally	40	9
Req_Freq for 4k×2k@60fps	1.24GHz	175MHz

times than previous design. Moreover, as shown in Table.4.6, for one 4×4 sub-block, instead of using 40 clock cycles for all the modes in I4MB and I16MB, my architecture totally requires only 9 cycles, which saves 77.5% processing cycles. Moreover, the related design effort is greatly reduced when extending to higher specification. For example, by extending the structure in [47] 4k×2k@60fps, the minimum required frequency (Req_Freq) will become 1.24GHz, which is extremely high design effort for existing technology. By using proposed structure with parallel processing flow, only 175MHz is needed to fulfill the throughput requirement for 4k×2k@60fps real-time processing. As for the power consumption, although proposed engine consumes 132% hardware cost of original design, the final estimated power saving is 67.24% due to its 85.88% reduction in design effort. Also, around 20k gates extra hardware will not cause serious performance degradation for the whole encoder system.

4.6 Conclusion remarks

In this chapter, two large image size targeted VLSI engines are given out. Firstly, for Super Hi-Vision 4k×4k@60fps, one fractional motion estimation engine is proposed. The proposed engine solves the throughput problem by utilizing algorithm level optimization and architecture level parallelism. In the algorithm level, the MRMPF and MCDOP schemes optimize the original high complexity FME process and reduce both design effort and required PU number. In hardware level, two parallel improved schemes, namely 16-Pel processing and MB-parallel scheme, are proposed in the hardware level. Also, the sub-sampling technique is adopted and 75% hardware cost is reduced for each PU. Additionally, one UPB scheme is proposed and achieves 28.67% to 86.39% pixel reuse for FME process. To solve the memory access conflict of MB-parallel processing, one parity pixel organization scheme is also proposed. With 412k hardware at only 145MHz, the proposed FME engine can handle real-time processing of Super Hi-Vision 4k×4k@60fps. Secondly, for intra part, one fully utilized and low design effort structure for H.264/AVC intra predictor generation is proposed. The data dependency problem in the conventional flow is analyzed and one parallel flow is given out, which achieve 37.5% reduction in processing time. After that, one fully utilized architecture which can generate predictors of all I4MB and I16MB modes with only 22.5% cycles of previous design is given out. Based on proposed parallel processing flow and efficient predictor generation architecture, the proposed design can achieve real-time intra predictor generation for 4k×2k@60fps with less than 200MHz. Compared with recent works, the proposed FME and intra engines reduce 85.92% and 85.88% design effort of original designs, which leads to only 6.69% and 32.76% estimated power of original FME and intra designs.

Chapter 5

Analysis of macroblock feature to fast inter mode decision

5.1 Introduction

The mode decision part in H.264/AVC plays an important role in the whole encoding process. The intra prediction part and block matching process in inter prediction are all included in the whole mode decision procedure. In H.264, besides skip mode, there are 7 block modes (as shown in Fig. 5.1) for inter prediction, 9 modes for 4×4 intra prediction and 4 modes for 16×16 intra prediction. The encoding process will loop all these modes and select one with the minimum cost. When rate distortion is incurred, all the prediction modes will be involved in a real encoding process. So, the complexity is insurmountable considering the real-time application.

In [22], a fast intra prediction algorithm is proposed, which greatly speeds up intra prediction process while still keep the quality. However, the decision on inter modes is more complicated compared with intra modes. It is because the motion estimation (ME) process adopts block matching on the plane of both current image and reference image, which incurs huge calculation on all the candidate points within the search window. The split, occlusion and fast motion in the moving video increase the ratio of temporal feature among frames, which makes it almost impossible to make a pre-decision on inter prediction.

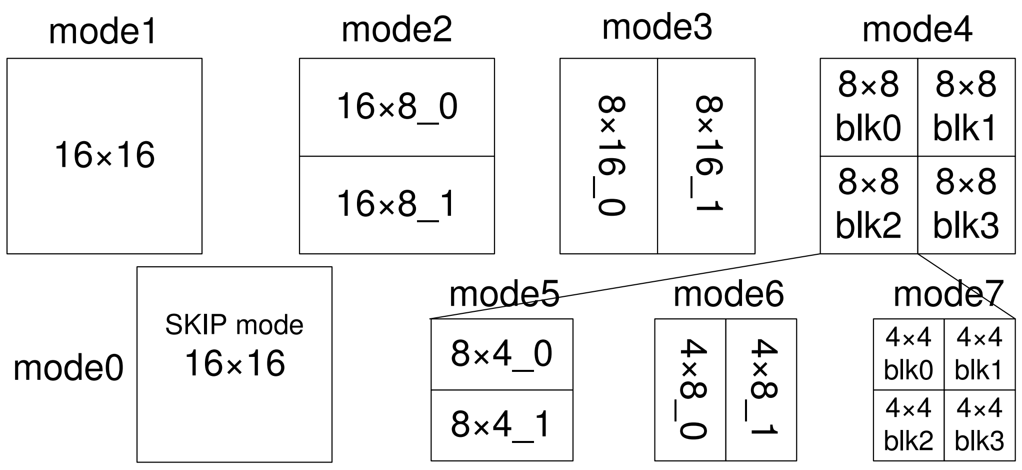


Figure 5.1: Inter Block Modes in H.264/AVC

Many works have been done to solve the problem. In [49], a pre-encoding scheme is proposed, which abstracts a down sampled small image and restrict the inter block modes within a small subset. Literature [50] tries the way of mean of absolute frame difference (MAFD) to filter out unpromising inter modes. In [51], an adaptive mode decision process based on all-zero coefficients block is proposed. Literature [52] and [53] focus on the optimization of early skip mode decision to release complexity of inter mode decision. However, the idea of introducing pre-processing in [49] and [50] will intensify the computation burden of whole encoding system. With the expansion of image size, for example HDTV application, the 1/2 down sampling or the MAFD calculation of the original frame will increase power dissipation and system latency dramatically. As for all-zero block and skip mode early detection based algorithms [52][53], there exist obvious limitations. With several foreground objects moving irregularly on the complicated background, or the decrease of quantization parameter, the ratio of all-zero block and skip modes will decrease significantly, which deteriorates the efficiency of inter mode filtering. In [54], a very fast mode decision algorithm is proposed, which dramatically reduces complexity for both low-motion and high-motion sequences. However, the compression capability is deteriorated obviously since the bit rate increase is quite large. In this dissertation, the complexity problem of inter mode decision is solved in several stages. Firstly, in the pre-stage (before ME starts), the homogeneity of current macroblock (MB) and the features of encoded

MBs on both current and previous frames are inspected to detect skip mode and filter out unpromising modes. Secondly, during ME process, the motion information is collected to discard unnecessary modes. I focus on the information of motion vector predictor's accuracy, the block overlapping situation , rate distortion cost and SAD's smoothness. The details are shown in following sections.

5.2 Pre-stage inter mode decision schemes

The fast inter mode decision algorithm targets at finding most candidate modes for the rate distortion based matching process. The early decision can be made either before or during encoding stage. In this dissertation, I firstly try to narrow down the candidate modes in both stages. In this section, two pre-stage inter mode decision schemes are described in detail.

5.2.1 MV oriented spatial-temporal inter mode check

In the conventional video sequences, spatial and temporal redundancy always exist within frame or between frames. In this dissertation, I propose a spatial-temporal skip mode early detection scheme which is applied before encoding process (named pre-stage scheme).

Since the encoding process is executed in a raster scan order, the only spatial information available for using is from the encoded MBs. Therefore, as shown in Fig.5.2(a), the left-up MB (LU.MB), left MB (L.MB) and upper MB (U.MB) of current MB (Cur.MB) are used for mode check. However, because only three MBs provide the mode information, the efficiency and correctness are quite limited. Therefore, the temporal information is also added. Besides co-located MB (Co.MB), there are 8 MBs around Co.MB in the previous frame and I classify all these MBs into three categories, as shown in Eq.5.1 to Eq.5.3. The Co.MB is the only element in the C0 category. The left-up (LU.MB), right-up (RU.MB), bottom-left (BL.MB) and bottom-right (BR.MB) MBs belong to C1 category. As for C2 category, it includes four MBs in cross direction such as U.MB, L.MB, right MB (R.MB) and bottom MB (B.MB) around Co.MB. The temporal mode check algorithm only depends on the dominant category of C1 and C2. Obeying the rule that the motion is continuous in the succeeding frames [13], I use the motion vector (MV) of MBs in C1 and C2 to decide dominant category. As shown in Eq.5.4 to Eq.5.6, the delta MV between C1 and C0 ($\Delta MV(C1, C0)$) is calculated based on the accumulation of absolute MV difference in the x ($\Delta MV_x(C1, C0)$) and y ($\Delta MV_y(C1, C0)$) direction. For example, Eq. 5.5 means that, the absolute difference between MV_x of MB in C0 category and that of each MB in C1 category is calculated in the first step. Then, $\Delta MV_x(C1, C0)$ is obtained based on sum of all the absolute difference results. Here, MV_x represents MV in x direction. The delta MV between C2 and C0 ($\Delta MV(C2, C0)$) is calculated based on the same principle. As shown in Eq.5.7, the category with minimum MV difference will be chosen as candidate category. For instance, when $\Delta MV(C1, C0)$ is smaller than $\Delta MV(C2, C0)$, it means that the motion vector difference between co-located MB (C0 category) and MBs in diagonal positions (C1 category) is smaller than the difference between C0 and C2 category. So, the co-located MB is more similar to C1 category in terms of motion vector and mode information in C1 category is used as reference for fast mode decision. The pseudo codes of our spatial-temporal algorithm is shown in Fig.5.3(a). It means that before ME, I apply mode check based on spatial and temporal information. The MV difference is used as a criterion to select candidate category (C') for temporal mode check. If all the modes of MBs in spatial (LU.MB, U.MB and L.MB in Fig. 5.2(a)) and temporal category (C' and C0) are skip modes (mode0), mode0 is selected as best inter mode. Otherwise, full encoding modes of H.264/AVC are enabled during the following process.

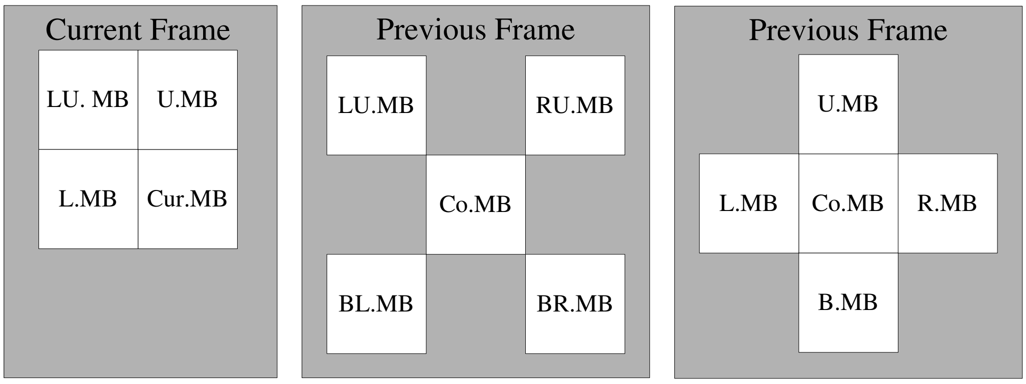
$$C0 \in \{Co.MB\} \tag{5.1}$$

$$C1 \in \{LU.MB, RU.MB, BL.MB, BR.MB\} \tag{5.2}$$

$$C2 \in \{U.MB, L.MB, R.MB, B.MB\} \tag{5.3}$$

$$\Delta MV(C1, C0) = \Delta MV_x(C1, C0) + \Delta MV_y(C1, C0) \tag{5.4}$$

$$\Delta MV_x(C1, C0) = \Sigma|C1\{MV_x\} - C0\{MV_x\}| \tag{5.5}$$



(a) Spatial Check (b) Temporal MB in C1 (c) Temporal MB in C2

Figure 5.2: Spatial-temporal Skip Mode Check

```
Spatial MB mode check
Loop MBs in C1 category
    Calculate delta MV (C1, C0)
End Loop
Loop MBs in C2 category
    Calculate delta MV (C2, C0)
End Loop
Decide temporal category (C’ + C0)
If MBs in Spatial and Temporal category
  are all mode0
    Only mode0 for Cur.MB
Else
    Full modes of H.264/AVC
```

(a) Spatial-temporal Skip Mode Check

```
Loop blk8x8_i of current MB, i = [0, 3]
    Edge gradient analysis of blk8x8_i
    If homogenous blk8x8_i
        Discard 8x4, 4x8, 4x4 modes
End Loop

If all blk8x8_i are all homogenous
    Disable 8x8 inter modes
```

(b) Edge Based Inter Mode Filtering

Figure 5.3: Pseudo Codes of Pre-Stage Inter Mode Decision

$$\Delta MV_y(C1, C0) = \Sigma|C1\{MV_y\} - C0\{MV_y\}| \tag{5.6}$$

$$\begin{cases} \Delta MV(C1, C0) < \Delta MV(C2, C0), & C1\ is\ adopted \\ \Delta MV(C2, C0) \leq \Delta MV(C1, C0), & C2\ is\ adopted \end{cases} \tag{5.7}$$

5.2.2 Edge gradient based inter mode filtering

The edge detection is another useful technique in both image processing and pattern recognition field. In [22], it uses Sobel edge operator to obtain candidate intra modes. In fact, the same method can also be extended into inter mode filtering. Figure. 5.4

demonstrates the mode distribution among different sequences and Fig. 5.5 is the corresponding edge gradient histogram of each frame. Here, the edge gradient is obtained by using Sobel operator on each MB. Since gradients of sequences with smooth and regular motion demonstrate similar distribution among different encoding frames. I extract edge gradients of 20th frame as an example. It is shown that the gradient distribution between 'mobile_qcif' and 'container_qcif' is quite big, which is in accordance with subjective observation. For mode distribution, the proportion of mode above 8×8 in 'container_qcif' is much more than that of 'mobile_qcif'. The situations of 'tempete_qcif' and 'grandma_qcif' are similar, where 'grandma_qcif' is more favorable to big inter modes. So, the decrease of gradient in image will increase the possibility of big inter modes in the final mode decision stage.

To be compatible with H.264/AVC encoding flow in JM, the Sobel edge detection oriented inter mode filtering is applied on the basis of MB level. Specifically, before block matching process, I analyze current MB and obtain the edge gradient of each pixel based on Eq. 5.8 to Eq. 5.10, where $P(m, n)$ is the pixel value on coordinate (m, n) of current MB. The G_x and G_y are the gradients in horizontal and vertical directions, respectively. These two gradient is simply summed up to get $G(m, n)$ as gradient of $P(m, n)$.

In JM software, the inter prediction in H.264/AVC standard is implemented based on block matching process from mode1 to mode7 sequentially. Based on this mechanism, I set a threshold on each of the four 8×8 block (blk8x8_0 to blk8x8_3) within current MB. As shown in Eq. 5.11, if the gradient of every pixels within one 8×8 block (blk8x8_i, i∈{0,1,2,3}) is within a predefined threshold, this 8×8 block is regarded as homogeneous (homo) sub-block. Otherwise, it is an edge 8×8 block. The edge based inter mode filtering algorithm is shown in Fig.5.3(b). For homogenous 8×8 block, small inter modes (mode5 to mode7) are removed before ME process. If all the four 8×8 blocks are homogenous ones, even the mode4 inter mode is filtered.

As for the threshold setting, it is always a trade-off between quality and complexity. The prediction error e in block matching process can be assumed as a jointly Gaussian source with zero mean and variance σ^2. According to [25], the distortion of quantization D is approximated as $QP^2/3$, where QP is the quantization parameter. So, the

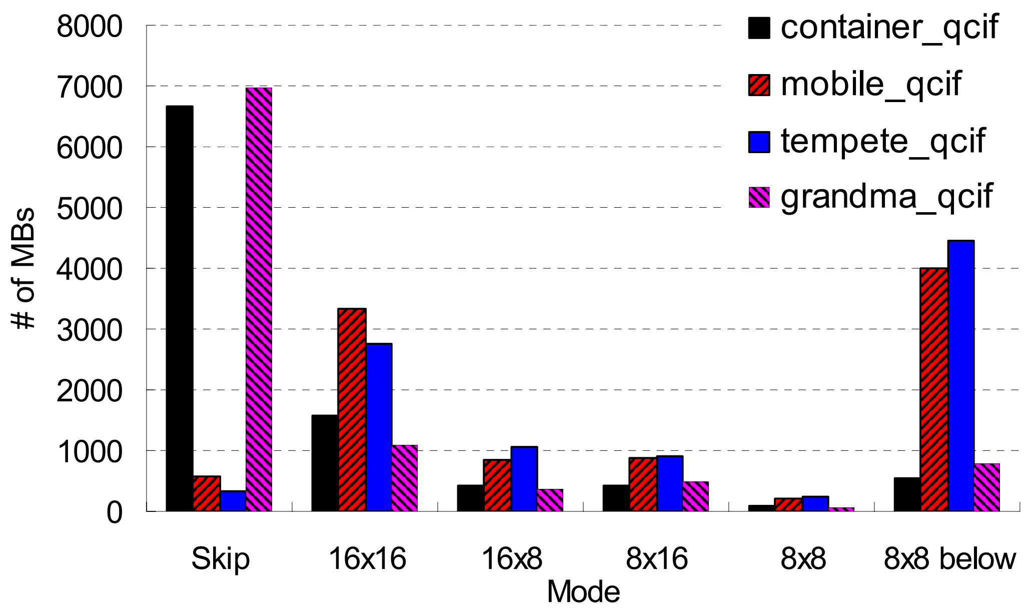


Figure 5.4: Inter Mode Distributions

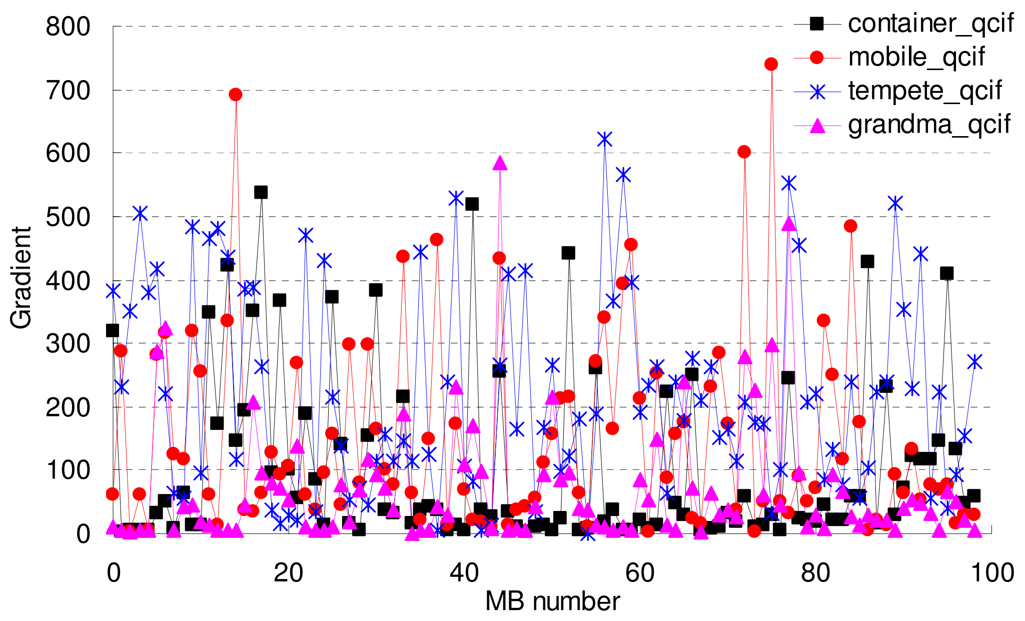


Figure 5.5: Gradient Distributions of 20th Frame

rate distortion function [26] can be represented as Eq. 5.12, where $R(D)$ is the related transmission bit-rate for distortion D. The σ^2 represents maximum distortion based on Gaussian model. When distortion D equals to zero, it indicates that original signal is reconstructed without any loss in image detail. All the information of image (including textures and noise) is exacted the same as original source image. Maximum transmission bit-rate is required for keeping the related information. In fact, such case is one ultimate state which will never happen in real video encoding system, like H.264/AVC. The reason is that the transform and quantization will cause some loss in image detail, which makes distortion between original source image and reconstructed one occur inevitably. On the other hand, when D is larger than σ^2, the related transmission bit-rate for D will become zero. This conclusion is in accordance with QP setting in H.264 encoding system. With the increase of QP, the smoothness of reconstructed frames is increased, which results in decline of image's details. The related residue value is also decreased. It means that quality degradation for edge abundant image is quite obvious under big QP. In the extreme case, all the details are removed by one very large QP and the residue information is vanished, which indicates that no transmission bit-rate is required. Thus, from theoretical analysis of [25] and [26], it is possible to simply set threshold as linear relationship with QP value. With exhaustive experiments, the Thr_G in Eq. 5.11 is defined as 4×QP to balance the quality and complexity reduction. In the following sections, the related thresholds are also set linearly with QP.

$$\begin{aligned} G_x(m,n) = |&P(m-1,n-1)+2P(m-1,n) \\ &+P(m-1,n+1)-P(m+1,n-1) \\ &-2P(m+1,n)-P(m+1,n+1)| \end{aligned} \tag{5.8}$$

$$\begin{aligned} G_y(m,n) = |&P(m-1,n-1)+2P(m,n-1) \\ &+P(m+1,n-1)-P(m-1,n+1) \\ &-2P(m,n+1)-P(m+1,n+1)| \end{aligned} \tag{5.9}$$

$$G(m,n) = G_x(m,n)+G_y(m,n) \tag{5.10}$$

$$\begin{aligned} &P(m,n) \in blk8x8\_i, i \in \{0,\ 1,\ 2,\ 3\} \\ &\begin{cases} G(m,n) < Thr\_G, & homo\ \ blk8x8\_i \\ otherwise, & edge\ \ blk8x8\_i \end{cases} \end{aligned} \tag{5.11}$$

$$R(D) = \begin{cases} \frac{1}{2}log\frac{\sigma^2}{D}, & 0 \le D \le \sigma^2 \\ 0, & D > \sigma^2 \end{cases} \tag{5.12}$$

5.3 Motion feature based fast inter mode decision schemes

In the pre-stage, the unpromising inter modes are filtered before ME starts. However, as mentioned above, the reduction of complexity is quite limited due to the motion feature of MB. In fact, during the ME procedure, it is also possible to skip unnecessary inter modes. The more motion information available for observation, the more it is feasible to narrow down the candidate modes.

5.3.1 MVP accuracy and block overlapping analysis

In JM software, the block matching process starts in the motion vector predictor (MVP), which is obtained by the neighboring coded MBs. For sequence with smooth or regular motion, the prediction of start point is very accurate. Fig. 5.6 shows the distribution of best integer point (BIP) of 16×16 mode among typical clips. The search window is divided into several layers. The layer 0 is the MVP point while layer 1 indicates the 8 points around MVP. The meaning of other layers can be traced by analogy. It is shown that large proportion of BIP are located in MVP position even for foreman_qcif and carphone_qcif cases. The high accuracy in MVP also indicates that the current MB is seldom split into small blocks. Since small modes (such as 4×4) is easily to be trapped into local minimum, I only use the information of 16×16 mode. In my algorithm, motion information of 16×16 mode is analyzed after it's ME search. If criterion of Eq. 5.13 is satisfied, current MB will be treated as big mode MB. In detail, Eq. 5.13 sets constraints on both MV and motion cost. Firstly, the MV of 16×16 mode ($MV\_16\times16$) must equal to its own MVP ($MVP\_16\times16$). Secondly, its motion cost ($mcost$) must be within one empirical threshold (Thr_MVP), which is set as 20×QP based on experiments. In our paper, mode1 to mode3 in Fig. 5.1 are defined as big modes while mode4 to mode7 are treated as small inter modes. So, I discard mode4 to mode7 during following ME process when current MB is a big mode one. For the rest MBs, whose best 16×16 MVs are not MVPs, I further analyze the motion information after mode3's block matching. The related mode decision criterion is shown in Eq.5.14 to Eq.5.16. It means that when

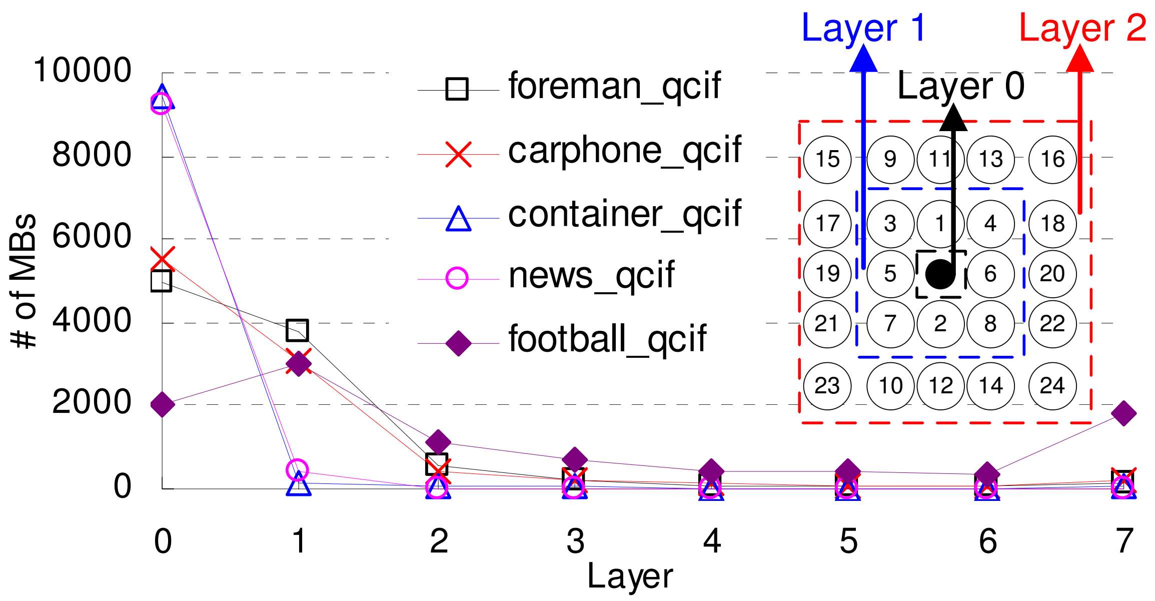


Figure 5.6: BIP Distribution of 16×16 Mode in 100 Frames

the absolute coordinates of MVs of block0 in mode2 and mode3 (16×8_0 and 8×16_0 in Fig. 5.1) are the same with MVs of mode1; and the MVs of block1 in mode2 and mode3 are only 8 pixel displacement in x or y direction. Then the previous three inter modes are overlapped each other, which indicates that current inter modes are well enough to express the motion trend. In this case, ME on mode5 to mode7 is bypassed. In our paper, the mode4 is remained to keep the video quality of our fast algorithm.

5.3.2 Smoothness of sum of absolute difference (SAD)

The SAD value which is obtained after ME process is another useful information. With search point approaching to the potential best one, the SAD value decreases gradually, which leads to less bits in the final encoding stage. On the contrary, the occurrence of big SAD value can indicate the necessity of ME on further small modes, which results in split of current MB. In this dissertation, I fully utilize SAD information to guide mode decision process. In JM software, ME is divided into integer motion estimation (IME) and fractional motion estimation (FME) stages. Since IME is well enough to represent image and object's overall motion trend, I use motion feature of IME stage in the proposed algorithm. Specifically, during IME on 16×16 mode, the four 8×8 SAD blocks are recorded, namely left-up 8×8 SAD (SAD8×8_LU), right-up 8×8 SAD (SAD8×8_RU), bottom-right 8×8 SAD (SAD8×8_BR), and bottom-left one (SAD8×8_BL). If Eq. 5.17 to Eq. 5.20 are all satisfied, it indicates that the distributions of four 8×8 size SAD value are quite smooth. So, further process on small modes is rarely needed and the *mode*5 to *mode*7 are discarded in the proposed scheme. When any of Eq. 5.17 to Eq. 5.20 is dissatisfied, the ME on mode2 and mode3 are skipped and the algorithm directly turns to mode4 to mode7 for precise matching process. The Thr_SAD in my algorithm is set as 15×QP based on our exhaustive experiments.

$$\begin{cases} MV\_16\times16 = MVP\_16\times16 \\ mcost\_16\times16 \leq Thr\_MVP \end{cases} \tag{5.13}$$

$$MV\_16\times16 = MV\_16\times8\_0 = MV\_8\times16\_0 \tag{5.14}$$

$$MV_x\_16\times16 = MV_x\_16\times8\_1 = MV_x\_8\times16\_1 - 8 \tag{5.15}$$

$$MV_y\_16\times16 = MV_y\_8\times16\_1 = MV_y\_16\times8\_1 - 8 \tag{5.16}$$

$$|SAD8\times8\_LU - SAD8\times8\_RU| < Thr\_SAD \tag{5.17}$$

$$|SAD8\times8\_BL - SAD8\times8\_BR| < Thr\_SAD \tag{5.18}$$

$$|SAD8\times8\_LU - SAD8\times8\_BL| < Thr\_SAD \tag{5.19}$$

$$|SAD8\times8\_RU - SAD8\times8\_BR| < Thr\_SAD \tag{5.20}$$

5.3.3 Rate distortion cost analysis on big inter modes

In the high complexity mode of H.264/AVC, after ME and intra prediction loop over all inter and intra modes, the rate distortion (RD) costs of each mode are checked exhaustively by minimizing the Lagrangian function, as shown in Eq. 5.21. The SSD is sum of squared difference between original source MB (s) and its reconstructed one (r). The R represents the rate after quantization and λ_{mode} is the Lagrange multiplier. It is shown that all the factors are related with QP and mode which is decided by inter and intra predictions. To ensure that best mode is not missed, the whole encoding process causes huge computation resources among several exhaustive loops. In the proposed algorithm, the RD cost analysis is involved in the inter prediction process. Specifically, at the end of ME on mode1, its RD cost ($Jmode1$) is compared with average RD cost of mode1 MBs (Ave_Jmode1) in the previous encoded frames. If Eq. 5.22 is satisfied, only mode0 and mode1 are candidate inter modes. When Eq. 5.22 is not satisfied, I further analyze the RD cost of mode2 ($Jmode2$) and mode3 ($Jmode3$) after ME of these two modes. Eq. 5.23 is adopted to judge whether current inter modes are well enough for the motion compensation process. It means that both RD cost of mode2 (Jmode2) and mode3 (Jmode3) must smaller than $0.5 \times (Ave\_Jmode2 + Ave\_Jmode3)$, where Ave_Jmode2 is the average RD cost of mode2 MBs in previous encoded frame and Ave_Jmode3 is the corresponding one of mode3 MBs. Since inter mode decision on modes below 8×8 is quite complicated, the RD cost early check is only inserted after ME on big modes such as mode1 to mode3 in the proposed algorithm.

$$\begin{aligned} J(s,r,mode|QP,\lambda_{mode}) = SSD(s,r,mode|QP) \\ +\lambda_{mode} \times R(s,r,mode|QP) \end{aligned} \tag{5.21}$$

$$Jmode1 < Ave\_Jmode1 \tag{5.22}$$

$$\{Jmode2, Jmode3\} < \frac{Ave\_Jmode2 + Ave\_Jmode3}{2} \tag{5.23}$$

5.4 Overall algorithm and experiments

The overall flow chart of proposed algorithm is shown in Fig. 5.7. The parts with bold font are original JM mode decision flow which consists of inter prediction, intra prediction and RD cost check. In the inter prediction part, the ME process executes block matching

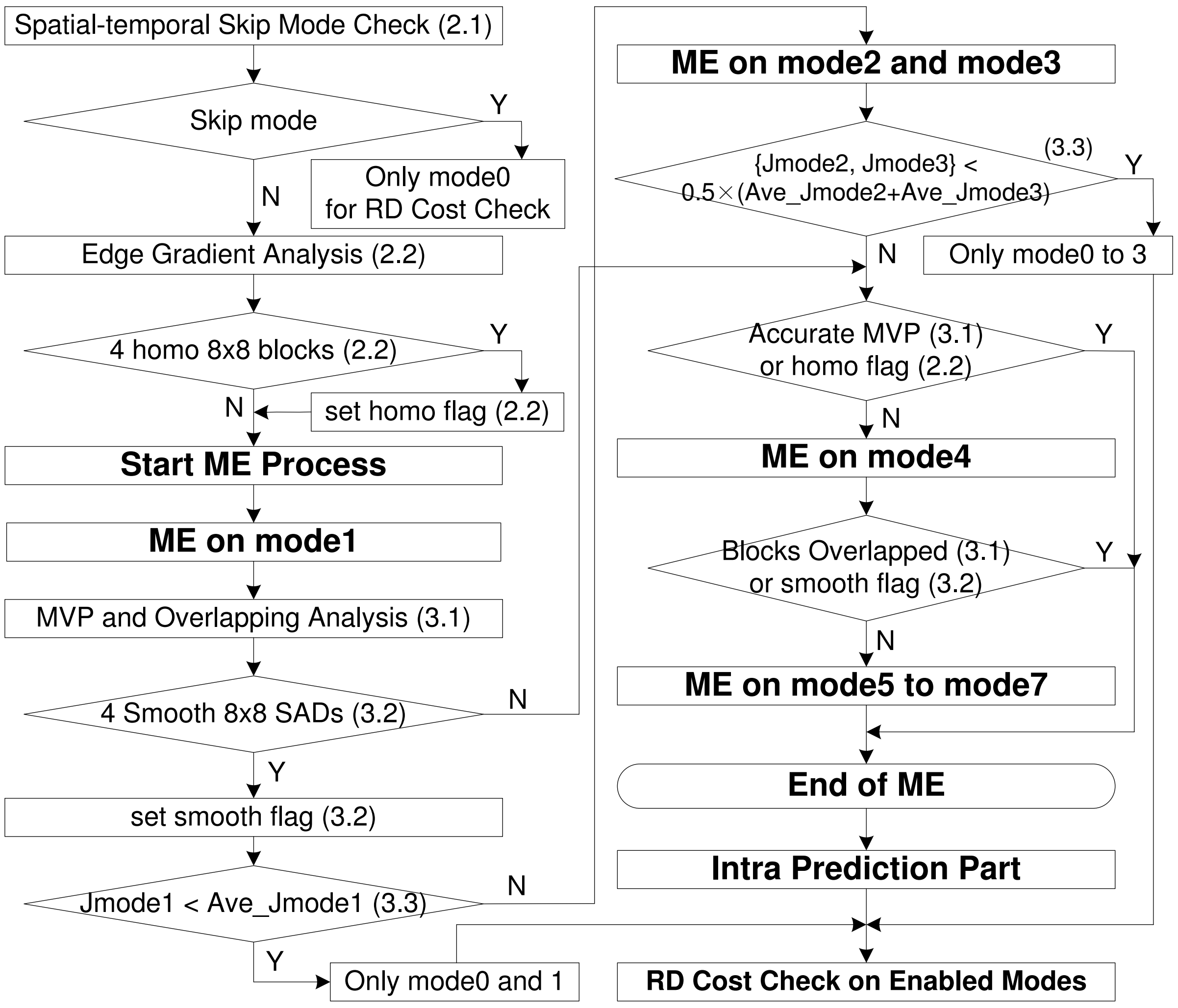
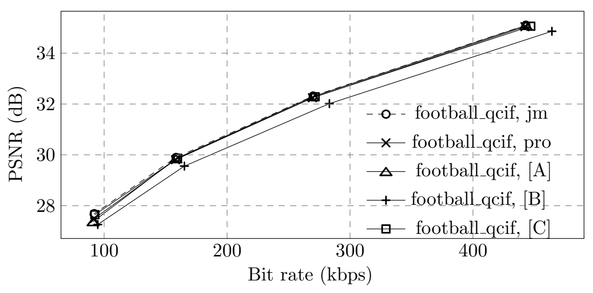

Spatial-temporal Skip Mode Check (2.1)
Skip mode
Y
N
Only mode0 for RD Cost Check
Edge Gradient Analysis (2.2)
4 homo 8x8 blocks (2.2)
set homo flag (2.2)
Start ME Process
ME on mode1
MVP and Overlapping Analysis (3.1)
4 Smooth 8x8 SADs (3.2)
set smooth flag (3.2)
Jmode1 < Ave_Jmode1 (3.3)
Only mode0 and 1
ME on mode2 and mode3
{Jmode2, Jmode3} < 0.5×(Ave_Jmode2+Ave_Jmode3)
(3.3)
Only mode0 to 3
Accurate MVP (3.1) or homo flag (2.2)
ME on mode4
Blocks Overlapped (3.1) or smooth flag (3.2)
ME on mode5 to mode7
End of ME
Intra Prediction Part
RD Cost Check on Enabled Modes

Figure 5.7: Overall Flow Chart of Proposed Algorithm

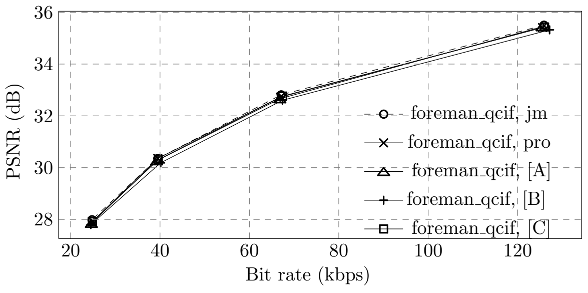
process from mode1 to mode7 sequentially. The proposed schemes described in this paper are noted with its section number in parentheses. It is shown that schemes in section 2.1 and 2.2 work before ME start and the rest schemes are involved with the ME process.

$$\Delta\Gamma = \frac{\Gamma_{pro} - \Gamma_{jm}}{\Gamma_{jm}} \times 100\%,\ \Gamma \in \{MET, Bits\} \tag{5.24}$$

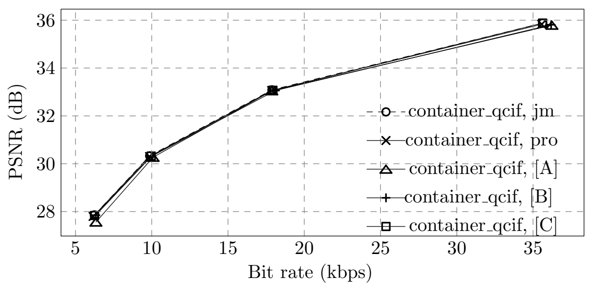
The proposed algorithm is implemented in JM 11.0 software [29]. Several QCIF and CIF clips with different features are used for simulation. I encode 200 frames with RD optimization enabled. The QP value ranges from 28 to 40 with interval of 4. The encoding structure is IPPP under baseline profile and 1 reference frame. The search range for QCIF



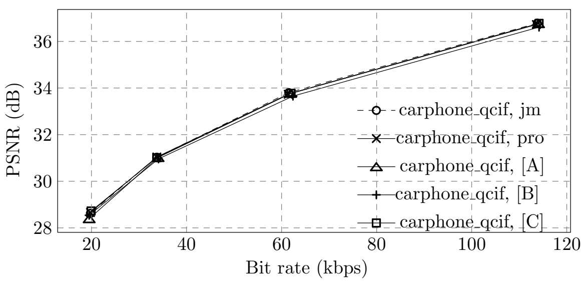
(a) football_qcif



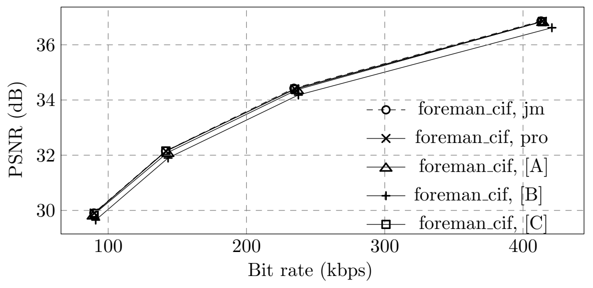
(b) foreman_qcif



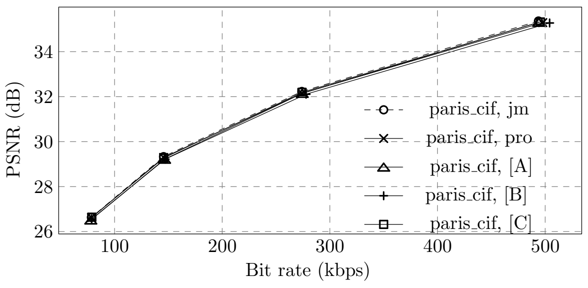
(c) container_qcif



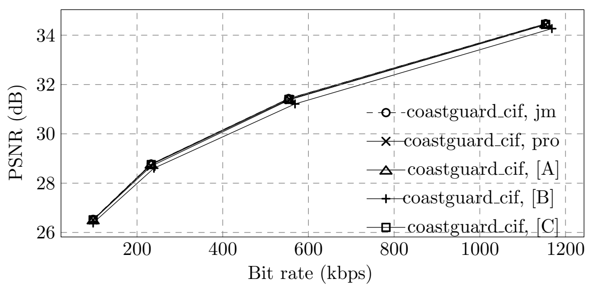
(d) carphone_qcif



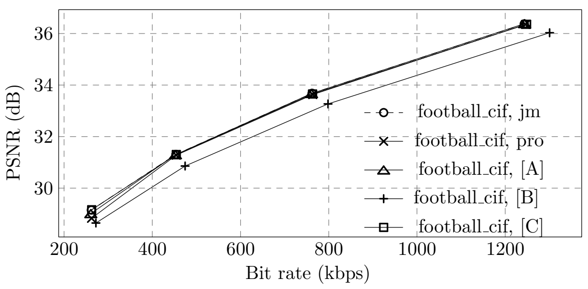
(e) foreman_cif



(f) paris_cif



(g) coastguard_cif



(h) football_cif

[A]: Reference [52], [B]: Reference [54], [C]: Reference [50]

Figure 5.8: Comparison of RD Curves

Table 5.1: Complexity Analysis based on $-\Delta MET$ (%)

clips	QP=28				QP=32				QP=36				QP=40			
	[52]	[54]	[50]	pro	[52]	[54]	[50]	pro	[52]	[54]	[50]	pro	[52]	[54]	[50]	pro
(1)	5.7	38.4	24.6	**24.9**	8.5	42.3	24.8	**23.0**	10.1	41.9	25.6	**21.6**	12.7	41.8	28.9	**23.9**
(2)	9.5	45.9	22.5	**30.6**	11.2	47.6	26.2	**27.2**	13.4	49.6	28.5	**34.2**	17.9	49.5	34.6	**35.0**
(3)	51.8	45.3	26.7	**51.7**	56.6	42.4	28.6	**52.3**	57.0	41.8	33.9	**50.2**	66.5	40.6	32.3	**53.4**
(4)	9.6	47.6	21.4	**35.2**	13.9	49.9	23.8	**33.7**	20.0	50.1	28.1	**33.0**	32.5	48.3	30.3	**30.3**
(5)	8.0	47.2	26.2	**49.0**	11.0	47.9	27.3	**46.5**	15.0	47.9	30.9	**45.6**	21.0	48.5	36.3	**46.1**
(6)	28.9	45.4	38.1	**40.2**	33.8	44.8	37.3	**39.1**	37.8	45.3	36.3	**40.3**	41.5	46.5	36.8	**44.7**
(7)	2.1	51.0	24.8	**39.8**	6.5	50.6	30.8	**38.1**	9.3	49.2	35.3	**40.5**	12.5	48.4	35.3	**46.5**
(8)	7.3	47.4	31.8	**37.3**	4.7	44.6	26.7	**29.9**	4.4	45.5	27.2	**31.8**	10.9	45.1	27.4	**32.8**

(1): football_qcif, (2): foreman_qcif, (3): container_qcif, (4): carphone_qcif,
(5): foreman_cif, (6): paris_cif, (7): coastguard_cif, (8): football_cif

Table 5.2: Quality Analysis based on C1 and C2 (C1: $\Delta PSNR$ (dB); C2: $\Delta Bits$ (%))

clps	Crit-erion	QP=28				QP=32				QP=36			
		[52]	[54]	[50]	pro	[52]	[54]	[50]	pro	[52]	[54]	[50]	pro
(1)	C1	-0.01	**-0.24**	-0.04	-0.06	-0.02	**-0.30**	-0.03	-0.07	-0.05	**-0.34**	-0.03	-0.09
	C2	+0.00	**+4.78**	+0.90	-0.17	-0.02	**+4.88**	+0.59	-0.23	-0.10	**+4.29**	+0.77	-0.31
(2)	C1	-0.06	**-0.18**	-0.07	-0.09	**-0.15**	**-0.21**	-0.05	-0.09	-0.09	**-0.17**	-0.05	-0.00
	C2	-0.33	**+1.00**	+0.10	-0.27	-0.45	+0.41	+0.48	-0.28	-0.64	**+1.64**	+0.28	+0.00
(3)	C1	-0.09	-0.07	-0.00	-0.05	-0.04	-0.03	-0.02	-0.03	-0.06	-0.02	-0.01	-0.03
	C2	**+1.72**	**+1.57**	+0.01	-0.16	-0.04	**+1.25**	+0.35	-0.11	**+2.27**	**+1.17**	**+1.01**	-0.10
(4)	C1	-0.02	**-0.17**	-0.01	-0.06	-0.03	**-0.15**	-0.03	-0.07	-0.01	-0.07	-0.02	+0.01
	C2	+0.22	+0.32	+0.32	-0.17	+0.63	**+1.42**	+0.77	-0.20	+0.80	**+1.10**	+0.00	+0.04
(5)	C1	-0.02	**-0.24**	-0.02	-0.01	-0.05	**-0.25**	-0.04	-0.02	-0.05	**-0.25**	-0.00	-0.01
	C2	+0.29	**+1.90**	+0.21	+0.25	**+1.10**	**+1.36**	+0.26	+0.24	**+1.18**	**+1.28**	+0.13	+0.25
(6)	C1	-0.06	-0.08	-0.04	-0.07	-0.08	**-0.12**	-0.04	-0.08	**-0.12**	**-0.11**	-0.06	-0.04
	C2	+0.05	**+2.09**	+0.36	+0.08	+0.14	**+1.43**	+0.06	+0.03	+0.61	**+1.00**	+0.07	+0.01
(7)	C1	-0.00	**-0.20**	-0.03	-0.04	-0.01	**-0.22**	-0.04	-0.04	-0.00	**-0.17**	-0.03	-0.05
	C2	+0.05	**+1.28**	+0.00	+0.01	+0.00	**+2.71**	+0.15	+0.52	+0.02	**+2.85**	+0.13	+0.04
(8)	C1	-0.01	**-0.34**	-0.01	-0.02	-0.01	**-0.41**	-0.03	-0.05	-0.02	**-0.45**	-0.01	-0.03
	C2	+0.05	**+4.62**	+0.42	+0.23	+0.16	**+4.74**	+0.14	+0.20	+0.21	**+4.61**	+0.20	+0.15

and CIF are ±16 and ±24 respectively.

The experiments and comparisons are shown in Table. 5.1, Table. 5.2 and Fig. 5.8. I use Eq. 5.24 to analyze the ratio of motion estimation time (MET) and bit increment. The Γ_{pro} is the element of proposed method (our method or others' methods) and Γ_{jm} is the related element caused by original JM full mode search which loops all inter modes. The Γ can be MET or $Bits$. As for $\Delta PSNR$, it is calculated by subtracting $PSNR$ of proposed algorithm from that of JM's. The '+' in Table. 5.2 represents PSNR gain and increment of bits. The meaning of '-' in Table. 5.2 means PSNR's drop and decrease of bits. It is shown that my scheme is superior to [50] in terms of complexity reduction, especially clips with slow motion feature such as 'container_qcif'. In case of [52], it can achieve high complexity reduction for clips such as 'container_qcif' and 'paris_cif'. However, the situation of fast motion ('football_qcif/cif'), complex background ('coastguard_cif') or camera's shaking ('foreman_qcif/cif') will deteriorate the efficiency of this algorithm greatly. As for [54], the ΔMET is always large among different clips. However, the quality trade-off is also very significant. Figure. 5.8 is the comparison of RD curve among original JM algorithm, others' works and my scheme. It is shown that the RD curves of proposed scheme and algorithms of [52] and [50] are all very close to JM's curve. However, for [54], the quality loss is very obvious, especially those fast motion clips such as 'football_qcif/cif' and 'coastguard_cif'. The quality loss for [54] in 'foreman_qcif/cif' case is also very big due to irregular shaking of camera. The detail quality analysis is shown in Table. 5.2. Since video quality variation is more vulnerable to small QP value, I give out PSNR and bit rate analysis of QP equals 28, 32, and 36 as an example. The $\Delta PSNR$ which is below -0.1dB and $\Delta Bits$ which is larger than 1% are marked with bold font. It is shown that most bold font cases fall into [54] and the bits increment in fast motion clips ('football_qcif/cif') is very large. In the proposed scheme, the quality loss and bits gain are always trivial while my scheme also achieves large complexity reduction for clips with static feature and comparative big reduction for clips of different motion types. For the QCIF format, the bits increment is always negative with only negligible PSNR loss. In all, for sequences with different motion type, the proposed algorithm can achieve 21.6% to 53.4% complexity reduction for the inter mode decision process.

5.5 Conclusion remarks

One fast inter mode decision algorithm is contributed in this chapter. In the pre-stage, the spatial-temporal information is used to detect skip mode in an early stage. The current MB's homogeneity is also extracted to filter out unpromising small modes. In the motion stage, the MVP's accuracy, the block overlapping and SAD distribution are analyzed to bypass unnecessary inter modes. Furthermore, the RD costs of big modes are obtained in an early stage and compared with historical ones to speed up mode decision procedure. Experiments show that the proposed algorithm can achieve up to 53.4% speed-up ratio with trivial quality loss and bit increment.

Chapter 6

Conclusions and future work

In this dissertation, the gap between software algorithm is hardware implementation is solved by hardware oriented algorithm and low power low cost architectures. The application fields ranges from small image size such as QCIF and CIF format, to HDTV image like 720p and 1080p, and finally reaches the Super Hi-Vision (SHV) application. As shown in Fig. 6.1, the whole thesis can be concluded with four phases.

Firstly, in the hardware algorithm level, this dissertation gives out hardware oriented fast motion estimation algorithms. The proposed hardware oriented schemes provide complexity reduction based on hardware data flow. The complexity reduction of this part mainly located in three categories. The first one is the MRF technique. With analysis in frequency domain, the MRF on the low frequency image part is removed. Also, similarity analysis on the centering 9 points is executed for further reduction of stationary part within the image. The second category is the search range (SR) adjustment. By extracting motion features during block matching process on the first frame, MB with small motion trend is restricted within centering region. So, redundant search points are eliminated. Moreover, for other motion MB, one recursive search range adjustment scheme is adopted for further reduction of search points. The third category is the matching pattern (MP). Compared with conventional direct sub-sampling, the proposed adaptive scheme not only take quality into consideration, but achieve complexity reduction in a reasonable way. By combining all the scheme, it is shown that the proposed hardware oriented algorithm can averagely achieve 88.53% reduction of ME time among different sequences. Also, the proposed scheme can be easily applied in to existing 4-stage based real-time encoder.

With some extra control module, the proposed MRF and SR schemes can be realized in existing encoder with only 27.68% of original processing cycles.

Secondly, based on adaptive sub-sampling, two flexible architectures are given out. The pros and cons of adaptive algorithm to existing fixed architectures are analyzed in this dissertation. The proposed architectures are based on optimization of existing SAD Tree and PPSAD structures which are two efficient hardwired structures for various application. In the proposed structures, pixel organization is applied in both architecture level and memory level. So, full data reuse and hardware utilization can be achieved, which result in low power and low processing time. Moreover, circuit optimization is discussed in this dissertation and further reduction in hardware cost and power dissipation can be attained based on my proposal. Experimental results show that the proposed RSADT and APPSAD can achieve up to 38.8% and 39.8% reduction in power consumption, respectively. Averagely, about 53.8% power can be reduced by proposed flexible architectures.

Thirdly, with the expansion of image size, the throughput issue for extreme large image come into existence. In detail, the hardware accelerator for SHV image has become a heated topic. By simply extending existing designs into SHV specification, the hardware size, power and design effort becomes impossible to be accomplished. In this dissertation, two low design effort hardware accelerators in FME and intra processing are given out. With algorithm optimization, parallel architecture, and 2-level (MB and frame level) data flow, the proposed FME engine can handle 4k×4k@60fps real-time processing. As for the intra engine, based on proposed 2-block data flow and fully utilized predictor generation architecture, this dissertation gives out one high speed intra predictor generation engine for 4k×2k@60fps specification. All in all, 85.92% and 85.88% design effort in reduced for SHV based FME and intra engine, respectively.

Fourthly, the mode decision part is also discussed in this dissertation. Although it is very hard to realize fast mode decision algorithm in hardware, the fast decision for image with different feature is very promising in video compression field. In this dissertation, proposals that fully considers the feature of image are given out. The complexity reduction in the proposed algorithm is realized in a multi-stage way. About 53.4% complexity in

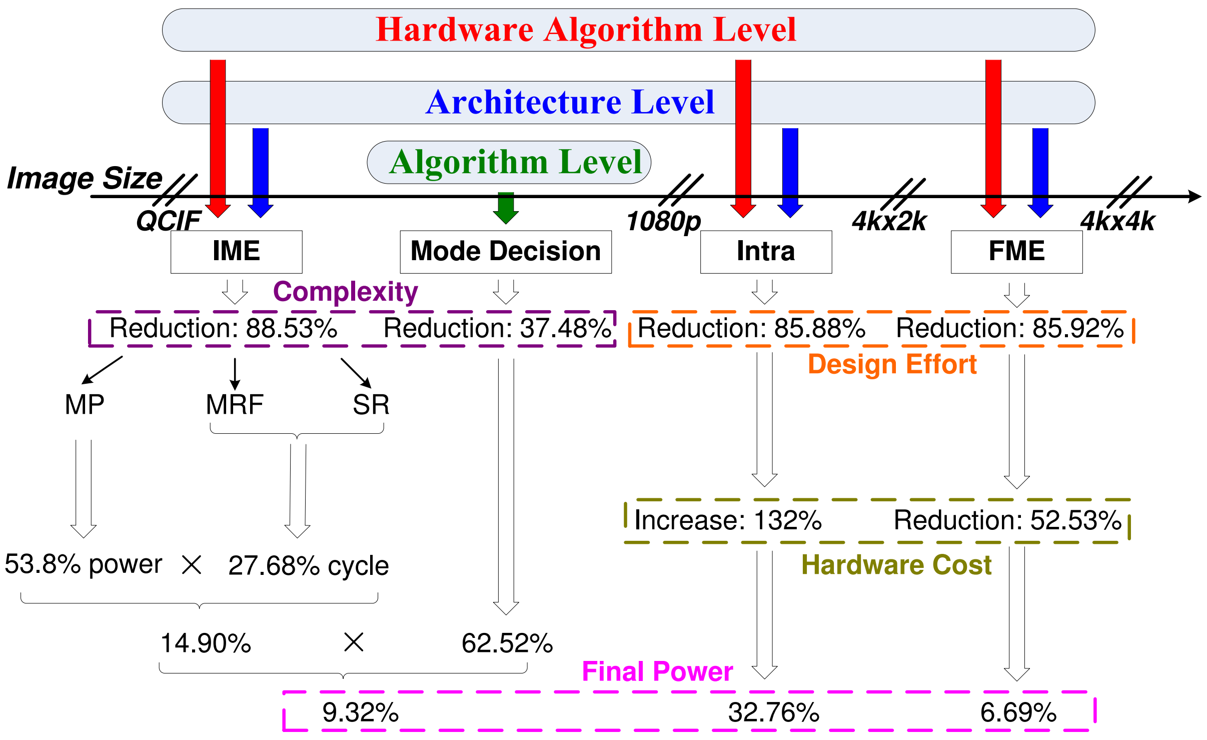


Figure 6.1: Whole conclusion of dissertation

inter mode decision part can be reduced and the proposed algorithm is superior to other schemes for sequences with various motion feature.

As for the power issue, as shown in Fig. 6.1, by combine all the hardware oriented algorithms and fast mode decision algorithm, the final power consumption in IME part is only 9.32% of original 4-stage based design. As for the FME and intra parts, compared with extension of recent works, the final estimated power is only 6.69% in FME engine and 32.76% in intra part.

In the future, the H.265 will come into existence. Questions like how much H.265 can achieve or comparison between H.265 and H.264/AVC will become a heated topic once H.265 standard is completed. Also, the ever increasing demand for ultra high resolution image makes low power and low cost real-time encoder attractive to the market. In this dissertation, I mainly give out solutions of FME and intra part. For the whole encoder part, problems such as high throughput IME engine, efficient arithmetic coding tools are still remained to be solved. Furthermore, 3-D video processing has attracted much attention in recent years. Some researchers have already proposed some algorithm and parallel architectures. However, current status is still far from satisfactory. The ultra low power and high quality video processing requires deep exploration in not only signal and video processing fields but also circuit design and support of manufacturing technology.

To sum up, this dissertation covers a wide research area in video compression field. The complexity reduction is achieved in a hardware oriented way. With related flexible structures and low design effort architectures, key issues in ASIC design such as hardware cost, power consumption, and throughput are solved by several proposals in this dissertation.

Acknowledgements

First of all, I would like to show my deepest gratitude to my loving wife, my parents and all my family members. Owing to your support, I have accomplished a lot in my life.

Secondly, I would like to express my gratitude to my supervisor, Associate Professor Takeshi Ikenaga. Under his guidance, I finish my Ph.D research and complete this dissertation. Thank you very much for your guidance during my stay in Waseda University. Also, many thanks to Professor Yasuo Matsuyama, Professor Jiro Katto, Professor Shinji Kimura, Professor Takeshi Yoshimura. Thank you for offering me lot of suggestions to the final completion of this dissertation.

Thirdly, I would like to thank all members in Ikenaga laboratory. Thank you for sharing wonderful time with me. Especially, I would like to thank Dr. Qin Liu for his encouragement and collaboration with my research. Many thanks to Mr. Lkhagvajantsan Damdinsuren, Miss Jia Su, Mr. Lei Wang, Mr. Jiachen Zhou, Mr. Shuijiong Wu, Mr. Zhewen Zheng, Mr. Jingbang Qiu, Mr. Tianci Huang, Mr. Xiaocong Jin, Mr. Jin Zhou, Mr. Bingrong Wang, Mr. Lei Sun, Miss. Ying Lu and Miss Chenjiao Guo, for everything you have done for me. Also, many thanks to Japanese students Mr. Takahiro Mori, Mr. Koichi Nakamura, Mr. Shinsuke Ushiki, Mr. Takahiro Sakayori, Mr. Kodai Kawane, and Mr. Tuyoshi Sasaki, for your kindly help in both research and my daily life.

Furthermore, I would like to thank Dr. Yoshiro Tsuboi, Mr. Masaki Nakagawa and Dr. Shunichi Ishiwata of Toshiba Corporation Semiconductor Company for offering suggestions and opinions during research discussion. Also, I would like to thank Dr. Shinichi Sakaida and Mr. Kazuhisa Iguchi of Japan Broadcasting Corporation for fruitful discussion on Super Hi-Vision image evaluation.

Lastly, I would like to acknowledge the support from CREST and Global COE program. Thank you very much for the support of all those international and domestic conferences.

References

[1] Joint Video Team. Draft ITU-T recommendation and final draft international standard of joint video specification,ITU-T Rec.H.264 and ISO/IEC 14496-10 AVC. March 2003. 1

[2] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, July 2003. 1, 86

[3] Information technology-coding of audio-visual objects-part 2: Visual, ISO/IEC 14496-2. 1999. 1

[4] Video coding for low bit rate communication, ITU-T Rec.H.263. 1998. 1

[5] Information technology–generic coding of moving pictures and associated audio information: Video, ISO/IEC 13818-2 and ITU-T Rec.H.262. 1996. 1

[6] Y. Huang, B. Hsieh, S. Chien, S. Ma, and L. Chen. Analysis and complexity reduction of multiple reference frames motion estimation in H.264/AVC. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(4):507–508, April 2006. 7, 8, 9, 48, 84

[7] J. Tham, S. Ranganath, M. Ranganath, and A. Kassim. A novel unrestricted center biased diamond search algorithm for block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(4):369–377, 1998. 8

[8] S. Zhu and K. Ma. A new diamond search algorithm for fast block matching motion estimation. In *ICICS '97. Proceedings of the 1997 International Conference on Infor-*

mation, Communication and Signal Processing, volume 1, pages 292–296, September 1997. 8

[9] K. Wang and O. Chen. Motion estimation using an efficient four-step search method. In *ISCAS '98. Proceedings of the 1998 International Symposium on Circuits and Systems*, volume 4, pages 217–220, May 1998. 8

[10] X. Jing and L. Chau. An efficient three-step search algorithm for block motion estimation. *IEEE Transactions on Multimedia*, 6(3):435–438, June 2004. 8

[11] A. M. Tourapis. Enhanced predictive zonal search for single and multiple reference frame motion estimation. In *VCIP '02. Proceedings of 2002 Visual Communications and Image Processing*, pages 1069–1079, January 2002. 8

[12] A. M. Tourapis, O. C. Au, and M. L. Liou. Highly efficient predictive zonal algorithms for fast block-matching motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(10):934–947, October 2002. 8

[13] Z. Chen, P. Zhou, and Y. He. Fast integer pel and fractional pel motion estimation for JVT. *JVT-F017.doc, 6th Meeting: Awaji, Island, JP*, pages 5–13, December 2002. 8, 9, 39, 120

[14] M. Chen, Y. Chiang, H. Li, and M. Chi. Efficient multi-frame motion estimation algorithms for MPEG-4 AVC/JVT/H.264. In *ISCAS '04. Proceedings of the 2004 International Symposium on Circuits and Systems*, volume 3, pages 737–740, May 2004. 8, 9, 29, 48

[15] Q. Liu, Z. Chen, S. Goto, and T. Ikenaga. Fast motion estimation algorithm based on edge block detection and motion vector information. In *ISPACS '07. Proceedings of the 2007 International Symposium on Intelligent Signal Processing and Communication Systems*, pages 241–244, November 2007. 8, 9, 29, 48

[16] P. Yin, H. Y. Tourapis, A. M. Tourapis, and J. Boyce. Fast mode decision and motion estimation for JVT/H.264. In *ICIP '03. Proceedings of the 2003 International Conference on Image Processing*, pages 853–856, September 2003. 8, 9, 48

[17] C. Chen, S. Chien, Y. Huang, T. Chen, T. Wang, and L. Chen. Analysis and architecture design of variable block size motion estimation for H.264/AVC. *IEEE Transactions on Circuits and Systems I*, 53(3):578–593, March 2006. 9, 34, 41, 47, 51, 54, 62, 65, 76, 77, 78, 79, 80

[18] T. Chen, S. Chien, Y. Huang, C. Tsai, C. Chen, T. Chen, and L. Chen. Analysis and architecture design of an hdtv 720p 30 frames/s H.264/AVC encoder. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(6):677–679, June 2006. 9, 34, 84, 86, 87, 111, 112, 113, 114

[19] Z. Liu, Y. Song, M. Shao, S. Li, L. Li, S. Ishiwata, M. Nakagawa, S. Goto, and T. Ikenaga. A 1.41w H.264/AVC real-time encoder SOC for HDTV1080P. In *VLSI Symposium '07. Proceedings of the 2007 VLSI Symposium*, pages 12–13, June 2007. 9, 51, 86, 87

[20] Z. Liu, Y. Song, T. Ikenaga, and S. Goto. A fine-grain scalable and low memory cost variable block size motion estimation architecture for H.264/AVC. *IEICE Transactions on Electronics*, E89-C(12):1928–1936, December 2006. 9, 21, 65, 67, 78, 79

[21] T. Wedi and H. G. Musmann. Motion and aliasing compensated prediction for hybrid video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):577–586, July 2003. 11, 32, 84

[22] F. Pan, X. Lin, S. Rahardja, K. Lim, Z. 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*, 15(6):813–821, July 2005. 14, 87, 88, 117, 121

[23] D. Wu, F. Pan, K. P. Lim, S. Wu, Z. Li, X. Lin, S. Rahardja, and C. Ko. Fast intermode decision in H.264/AVC video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(6):953–958, July 2005. 14

[24] G. J. Sullivan, T. Wiegand, and K. Lim. Joint model reference encoding methods and decoding concealment method. *JVT I049*, September 2003. 14

[25] H. Gish and J. N. Pierce. Asymptotically efficient quantizing. *IEEE Transactions on Information Theory*, 14(5):676–683, Sept 1968. 16, 17, 122, 124

[26] T. Berger. Rate distortion theory. *Prentice Hall*, 1971. 16, 17, 123, 124

[27] Y. Huang, T. Wang, B. Hsieh, and L. Chen. Hardware architecture design for variable block size motion estimation in MPEG-4 AVC/JVT/ITU-T H.264. In *ISCAS '03. Proceedings of the 2003 International Symposium on Circuits and Systems*, volume 2, pages 796–799, May 2003. 32, 79

[28] Y. Huang, Z. Liu, Y. Song, S. Goto, and T. Ikenaga. Parallel improved HDTV720p targeted propagate partial sad architecture for variable block size motion estimation in H.264/AVC. *IEICE Transactions on Fundamentals*, E91-A(4):987–997, April 2008. 32, 41, 59, 60, 80

[29] JM11.0. http://iphome.hhi.de/suehring/tml/. January 2007. 32, 129

[30] G. Bjøntegaard. Calculation of average psnr differences between rd-curves. *VCEG-M33*, April 2001. 39

[31] Y. Huang, T. Chen, C. Tsai, C. Chen, T. Chen, C. Chen, C. Shen, S. Ma, T. Wang, B. Hsieh, H. Fang, and L. Chen. A 1.3TOPS H.264/AVC single-chip encoder for HDTV applications. In *ISSCC '05. Proceedings of the 2005 International Solid-State Circuits Conference*, pages 128–130, Febrary 2005. 51, 87

[32] J. Tuan, T. Chang, and C. Jen. On the data reuse and memory bandwidth analysis for full-search block-matching vlsi architecture. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(1):61–72, January 2002. 67

[33] T. Wang, Y. Huang, H. Fang, and L. Chen. Performance analysis of hardware oriented algorithm modifications in H.264. In *ICASSP '03. Proceedings of the 2003 International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 493–496, April 2003. 76

[34] M. Kim, I. Hwang, and S. Chae. A fast VLSI architecture for full-search variable block size motion estimation in MPEG-4 AVC/H.264. In *ASP-DAC '05. Proceeding*

of the 2005 Asia and South Pacific Design Automation Conference, volume 1, pages 631–634, January 2005. 78, 79

[35] S. Y. Yap and J. V. McCanny. A VLSI architecture for variable block size video motion estimation. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 51(7):384–389, October 2004. 78, 79

[36] H. Chang, J. Chen, C. Su, Y. Yang, Y. Li, C. Chang, Z. Chen, W. Yang, C. Lin, C. Chen, J. Wang, and J. Guo. A 7mw-to-183mw dynamic quality-scalable H.264 video encoder chip. In *ISSCC '07. Proceedings of the 2007 International Solid-State Circuits Conference*, pages 280–281, Febrary 2007. 83

[37] Y. Chen, T. Chuang, Y. Chen, C. Li, C. Hsu, S. Chien, and L. Chen. An H.264/AVC scalable extension and high profile HDTV 1080p encoder chip. In *VLSI Symposium '08. Proceedings of the 2008 VLSI Symposium*, pages 104–105, June 2008. 83

[38] S. Mochizuki, T. Shibayama, M. Hase, F. Izuhara, K. Akie, M. Nobori, R. Imaoka, H. Ueda, K. Ishikawa, and H. Watanabe. A low power and high picture quality H.264/MPEG-4 video codec ip for hd mobile applications. In *ISSCC '07. Proceedings of the 2007 Asian Solid-State Circuits Conference*, pages 176–179, November 2007. 83

[39] T. Chen, Y. Chen, C. Tsai, and L. Chen. Low power and power aware fractional motion estimation of H.264/AVC for mobile applications. In *ISCAS '06. Proceedings of the 2006 International Symposium on Circuits and Systems*, pages 5331–5334, May 2006. 84

[40] T. Chen, Y. Huang, and L. Chen. Fully utilized and reusable architecture for fractional motion estimation of H.264/AVC. In *ICASSP '04. Proceedings of the 2004 International Conference on Acoustics, Speech and Signal Processing*, pages 9–12, May 2004. 86, 87, 89, 90, 93, 112

[41] Z. Liu, Y. Song, M. Shao, S. Li, L. Li, S. Ishiwata, M. Nakagawa, S. Goto, and

T. Ikenaga. HDTV 1080p H.264/AVC encoder chip design and performance analysis. *IEEE Journal of Solid-State Circuit*, 44(2):594–608, February 2009. 86, 113, 114

[42] Y. Wang, C. Cheng, and T. Chang. A fast algorithm and its VLSI architecture for fractional motion estimation for H.264/MPEG-4 AVC video coding. *IEEE Transactions on Circuits and Systems For Video Technology*, 17(5):578–583, May 2007. 86, 113, 114

[43] Y. Chen, T. Chuang, Y. Chen, C. Tsai, and L. Chen. Frame-parallel design strategy for high definition b-frame H.264/AVC encoder. In *ISCAS '08. Proceedings of the 2008 International Symposium on Circuits and Systems*, pages 29–32, May 2008. 87, 97, 111, 112, 113, 114

[44] G. Tian, T. Zhang, T. Ikenaga, and S. Goto. A fast hybrid decision algorithm for H.264/AVC intra prediction based on entropy theory. In *MMM '09. Proceedings of the 2009 International Conference on Multimedia Modeling*, pages 85–95, January 2009. 87, 88

[45] Y. Lin and Y. Chang. Fast block type decision algorithm for intra prediction in H.264/FRext. In *ICIP '05. Proceedings of the 2005 International Conference on Image Processing*, pages 585–588, September 2005. 87

[46] J. Wang, J. Wang, J. Yang, and J. 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*, 17(10):1414–1422, October 2007. 87, 88

[47] Y. Huang, B. Hsieh, T. Chen, and L. Chen. Analysis, fast algorithm, and VLSI architecture design for H.264/AVC intra frame coder. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(3):378–401, March 2005. 87, 88, 100, 101, 102, 105, 115

[48] T. Chen, Y. Chen, C. Tsai, S. Tsai, S. Chien, and L. Chen. 2.8 to 67.2mw low-power and power-aware H.264 encoder for mobile applications. In *VLSI Symposium '07.*

Proceedings of the 2007 VLSI Symposium, pages 222–223, June 2007. 91, 92, 113, 114

[49] D. Zhu, Q. Dai, and R. Ding. Fast inter prediction mode decision for H.264. In *ICME '04. Proceedings of the 2004 International Conference on Multimedia & Expo*, pages 1123–1126, June 2004. 118

[50] X. Jing and L. Chau. Fast approach for H.264 inter-mode decision. *Electron Letter*, 40(17):1050–1052, September 2004. 118, 130, 131, 132

[51] Y. Kim, J. Yoo, S. Lee, J. Shin, J. Paik, and H. Jung. Adaptive mode decision for H.264 encoder. *Electron Letter*, 40(19):1172–1173, September 2004. 118

[52] C. Grecos and M. Yang. Fast inter mode prediction for p slices in the H.264 video coding standard. *IEEE Transaction on Broadcasting*, 51(2):256–263, June 2005. 118, 130, 131, 132

[53] I. Choi, J. Lee, and B. Jeon. Fast coding mode selection with rate-distortion optimizaion for MPEG-4 part-10 AVC/H.264. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(12):1557–1558, December 2006. 118

[54] L. Salgado and M. Nieto. Sequence independent very fast mode decision algorithm on H.264/AVC baseline profile. In *ICIP '06. Proceedings of the 2006 International Conference on Image Processing*, pages 41–44, October 2006. 118, 130, 131, 132

Publications

Journal Papers (with review)

[1] Yiqing Huang, Takeshi Ikenaga, "Highly Parallel Fractional Motion Estimation Engine for Super Hi-Vision 4k $\times$ 4k@60fps", IEICE Electronics, March, 2010 (accepted).

[2] Yiqing Huang, Qin Liu, Shuijiong Wu, Zhewen Zheng, Takeshi Ikenaga, "Macroblock and Motion Feature Analysis to H.264/AVC Fast Inter Mode Decision", IEICE Fundamentals, Vol.E92-A, No.12, pp.3361-3368, December, 2009.

[3] Yiqing Huang, Qin Liu, Satoshi Goto, Takeshi Ikenaga, "Adaptive Sub-sampling based Reconfigurable SAD Tree Architecture for HDTV Application", IEICE Fundamentals, Vol.E92-A, No.11, pp.2819-2829, November, 2009.

[4] Yiqing Huang, Qin Liu, Satoshi Goto, Takeshi Ikenaga, "VLSI Oriented Fast Motion Estimation Algorithm Based On Pixel Difference, Block Overlapping And Motion Feature Analysis", IEICE Fundamentals, Vol. E92-A, No. 8, pp. 1986-1999, August, 2009.

[5] Yiqing Huang, Qin Liu, Takeshi Ikenaga, "Macroblock Feature based Adaptive Propagate Partial SAD Architecture for HDTV Application", IPSJ Transactions on System LSI Design Methodology, Vol. 3, pp. 263-273, August, 2009.

[6] Yiqing Huang, Qin Liu, Takeshi Ikenaga, "Macroblock Feature based Complexity Reduction for H.264/AVC Motion Estimation", IEICE Fundamentals, Vol. E91-A, No. 10, pp. 2934-2944, October, 2008.

[7] Yiqing Huang, Zhenyu Liu, Yang Song, Satoshi Goto, Takeshi Ikenaga, "Parallel Improved HDTV720p Targeted Propagate Partial SAD Architecture for Variable Block Size Motion Estimation in H.264/AVC", IEICE Fundamentals, Vol. E91-A, No. 4, pp. 987-997, April, 2008.

[8] Yiqing Huang, Jinyi Zhang, "IP Core Testing Method Based on New MSN Frame", Journal of Shanghai University (Natural Science), Vol. 11, No. 6, December, 2005.

[9] Qin Liu, Yiqing Huang, Satoshi Goto, Takeshi Ikenaga, "Hardware-Oriented Early Detection Algorithms for 4 $\times$ 4 and 8 $\times$ 8 All-Zero Blocks in H.264", IEICE Fundamentals, Vol. E92-A, No.4, pp. 1063-1071, April, 2009.

[10] Qin Liu, Yiqing Huang, Satoshi Goto, Takeshi Ikenaga, "Edge Block Detection and Motion Vector Information Based Fast VBSME Algorithm", IEICE Fundamentals, Vol. E91-A, No. 8pp. 1935-1943, August, 2008.

International Conference (with review)

[1] Yiqing Huang, Qin Liu, Takeshi Ikenaga, "Fully Utilized and Low Design Effort Architecture for H.264/AVC Intra Predictor Generation", International Conference on Multimedia Modeling (MMM 2010), Chongqing, China, January, 2010.

[2] Yiqing Huang, Qin Liu, Takeshi Ikenaga, "Macroblock Feature and Motion Involved Multi-stage Fast Inter Mode Decision Algorithm in H.264/AVC Video Coding", IEEE International Conference on Image Processing (ICIP 2009), Cairo, Egypt, November, 2009.

[3] Yiqing Huang, Takeshi Ikenaga, "Architecture Optimization for H.264/AVC Propagate Partial SAD Engine in HDTV Application", International SOC Design Conference (ISOCC 2009), Busan, Korea, November, 2009.

[4] Yiqing Huang, Takeshi Ikenaga, "Parallel Improved Low Design Effort H.264/AVC Fractional Motion Estimation Engine for Super Hi-Vision Application", 8th International Conference on ASIC (ASICON 2009), Changsha, China, October, 2009.

[5] Yiqing Huang, Qin Liu, Takeshi Ikenaga, "Highly Parallel Fractional Motion Estimation Engine for Super Hi-Vision 4k $\times$ 4k@60fps", IEEE 2009 International Workshop on Multimedia Signal Processing (MMSP 2009), Rio de Janeiro, Brazil, October, 2009.

[6] Yiqing Huang, Qin Liu, Takeshi Ikenaga, "Fast Inter Mode Decision Algorithm Based On Macroblock and Motion Feature Analysis For H.264/AVC Video Coding", 17th European Signal Processing Conference (EUSIPCO 2009), Glasgow, Scotland, August, 2009.

[7] Yiqing Huang, Qin Liu, Takeshi Ikenaga, "Spatial Feature Based Reconfigurable H.264/AVC Integer Motion Estimation Architecture for HDTV Video Encoder", 16th International Conference on Digital Signal Processing (DSP 2009), pp. 1-6, Santorini, Greece, July, 2009.

[8] Yiqing Huang, Qin Liu, Takeshi Ikenaga, “Content aware configurable architecture for H.264/AVC integer motion estimation engine”, IEEE International Conference on Multimedia & Expo (ICME 2009), pp. 37-40, June, 2009.

[9] Yiqing Huang, Qin Liu, Satoshi Goto, Takeshi Ikenaga, “Reconfigurable SAD Tree Architecture based on Adaptive Sub-sampling in HDTV Application”, ACM Great Lakes Symposium on VLSI (GLSVLSI 2009), pp. 463-468, May, 2009.

[10] Yiqing Huang, Takeshi Ikenaga, “VLSI Oriented Fast Motion Estimation Algorithm Based on Macroblock and Motion Feature Analysis”, The 5th International Colloquium on Signal Processing and its Applications (CSPA 2009), Kuala Lumpur, Malaysia, pp. 166-171, March, 2009.

[11] Yiqing Huang, Qin Liu, Takeshi Ikenaga, “Compressor Tree Based Processing Element Optimization in Propagate Partial Sad Architecture”, IEEE Asia-Pacific Conference on Circuits and Systems (APCCAS 2008), Macao, China, pp. 1786-1789, December, 2008.

[12] Yiqing Huang, Qin Liu, Takeshi Ikenaga, “Half Pixel Cost Distribution based Simplified Fractional Motion Estimation”, The 10th IASTED International Conference on Signal and Image Processing (SIP 2008), Hawaii, USA, pp. 156-161, August, 2008.

[13] Yiqing Huang, Qin Liu, Satoshi Goto, Takeshi Ikenaga, “Adaptive Subsampling and Motion Feature based Fast H.264 Motion Estimation”, Congress on Image and Signal Processing (CISP 2008), Sanya, China, pp. 671-675, May, 2008.

[14] Yiqing Huang, Satoshi Goto, Takeshi Ikenaga, “VLSI Friendly Computation Reduction Scheme in H.264/AVC Motion Estimation”, IEEE International Symposium on Circuits and Systems (ISCAS 2008), Seat-

tle, USA, pp. 844-847, May, 2008.

[15] Yiqing Huang, Zhenyu Liu, Satoshi Goto, Takeshi Ikenaga, "Hardware Freiendly Background Analysis Based Complexity Reduction in H.264/AVC Multiple Reference Frames Motion Estimation", International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS 2007), Xiamen, China, pp. 594-597, November, 2007.

[16] Yiqing Huang, Zhenyu Liu, Satoshi Goto, Takeshi Ikenaga, "Cost Efficient Propagate Partial SAD Architecture for Integer Motion Estimation in H.264/AVC", The 7th International Conference on ASIC (ASICON 2007), Guilin, China, pp. 782-785, October, 2007.

[17] Yiqing Huang, Zhenyu Liu, Satoshi Goto, Takeshi Ikenaga, "Adaptive Edge Detection Pre-Process Multiple Reference Frames Motion Estimation in H.264/AVC", 2007 International Conference on Communications, Circuits and Systems (ICCCAS 2007), Kokura, Japan, pp. 787-791, July, 2007.

[18] Xiaocong Jin, Yiqing Huang, Qin Liu, Shuijiong Wu, Takeshi Ikenaga, "Fast Spatial Direct Mode Decision for B Slice based on Temporal Information in H.264 Standard", International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS 2009), Kanazawa, Japan, December, 2009.

[19] Shuijiong Wu, Yiqing Huang, Qin Liu, Takeshi Ikenaga, "Rate-Distortion Optimized Multi-Stage Rate Control Algorithm For H.264/AVC Video Coding", 17th European Signal Processing Conference (EUSIPCO 2009), Glasgow, Scotland, August, 2009.

[20] Shuijiong Wu, Yiqing Huang, Qin Liu, Takeshi Ikenaga, "Bit-Usage Analysis Based Frame Layer QP Adjustment for H.264/AVC Rate Control at Low Bit-Rate", The 24th International Technical Conference on Circuits and Systems, Computers and Communications (ITC-CSCC

2009), July, 2009.

[21] Zhewen Zheng, Yiqing Huang, Qin Liu, Takeshi Ikenaga, "Intra Mode Decision for Reducing Block Types and Prediction Modes Based on Edge Information in H.264/AVC", The 24th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC 2009), July, 2009.

[22] Shuijiong Wu, Yiqing Huang, Takeshi Ikenaga, "A Macroblock-Level Rate Control Algorithm for H.264/AVC Video Coding with Context-Adaptive MAD Prediction Model", International Conference on Computer modeling and simulation (ICCMS 2009), February, 2009.

[23] Qin Liu, Yiqing Huang, Takeshi Ikenaga, "Early Detection Algorithms for 8 $\times$ 8 All-Zero Blocks in H.264/AVC", IEEE 2008 International Workshop on Multimedia Signal Processing (MMSP 2008), October, 2008.

[24] Qin Liu, Yiqing Huang, Takeshi Ikenaga, "4 $\times$ 4 SAD and SATD based all Zero Block Detection Algorithm in H.264/AVC", The 10th IASTED International Conference on Signal and Image Processing (SIP 2008), August, 2008.

[25] Qin Liu, Yiqing Huang, Takeshi Ikenaga, "Early Detection Algorithms for 4 $\times$ 4 and 8 $\times$ 8 All-Zero Blocks in H.264/AVC", 16th European Signal Processing Conference (EUSIPCO 2008), August, 2008.

[26] Jiachen Zhou, Yiqing Huang, Takeshi Ikenaga, "A Resource Preserved MAC Protocol for QoS Provided UWB Ad Hoc Network", The 23rd International Technical Conference on Circuits and Systems, Computers and Communications (ITC-CSCC 2008), July, 2008.

[27] Qin Liu, Yiqing Huang, Satoshi Goto, Takeshi Ikenaga, "Aliasing Error Reduction Based Fast VBSME Algorithm", Congress on Image and Signal Processing (CISP 2008), May, 2008.

[28] Zhenyu Liu, Yiqing Huang, Yang Song, Lingfeng Li, Satoshi Goto, Takeshi Ikenaga, "VLSI Friendly Edge Gradient Detection Based Multiple Reference Frames Motion Estimation Optimization for H.264/AVC", 15th European Signal Processing Conference (EUSIPCO 2007), September, 2007.

[29] Zhenyu Liu, Yiqing Huang, Yang Song, Satoshi Goto, Takeshi Ikenaga, "Hardware-Efficient Propagate Partial SAD Architecture for Variable Block Size Motion Estimation in H.264/AVC", ACM Great Lakes Symposium on VLSI (GLSVLSI 2007), March, 2007.

[30] Shuijiong Wu, Peilin Liu, Yiqing Huang, Takeshi Ikenaga, "On Bit Allocation and Lagrange Multiplier Adjustment for Rate-Distortion Optimized H.264 Rate Control", IEEE 2009 International Workshop on Multimedia Signal Processing (MMSP 2009), October, 2009.

Domestic Conference (with review)

[1] Yiqing Huang, Zhenyu Liu, Yang Song, Satoshi Goto, Takeshi Ikenaga, "Inter Search Mode Reduction Based Parallel Propagate Partial SAD Architecture for Variable Block Size Motion Estimation in H.264/AVC", Karuizawa Workshop, April, 2007.

Domestic Conference (without review)

[1] Yiqing Huang, Qin Liu, Takeshi Ikenaga, "Multi-Stage Based Inter Mode Decision Algorithm in H.264/AVC", IEICE General Conference, D-11-19, March, 2009.

[2] Yiqing Huang, Satoshi Goto, Takeshi Ikenaga, "Integer Search Position Based Fast Motion Estimation in H.264/AVC", IEICE General Conference, A-4-18, March, 2008.

[3] Qin Liu, Yiqing Huang, Takeshi Ikenaga, "Bayesian Decision Based All-Zero Block Detection Algorithm in H.264/AVC", IEICE General Conference, D-11-20, March, 2009.

[4] Shuijiong Wu, Yiqing Huang, Qin Liu, Takeshi Ikenaga, "Macroblock Level Rate Control for H.264/AVC Based on Model Parameter Update and Weighted Reference Calculation", IEICE General Conference, D-11-21, March, 2009.

Invited Paper

Yiqing Huang, Takeshi Ikenaga, "Analysis of Adaptive Algorithm to Power Aware Design for H.264/AVC Integer Motion Estimation Engine in HDTV Application", The 7th International Conference on ASIC (ASICON 2007), Guilin, China, pp. 163-166, October, 2007.

Awards

CSPA 2009 Best Paper Award

ISOCC 2009 Samsung Award

2007 Excellent Student Award of The IEEE Fukuoka Section