EFFICIENT FRAMEWORK FOR MACROBLOCK PREDICTION AND PARALLEL TASK ASSIGNMENT IN VIDEO CODING

by

Muhammad Asif
PE091005

A research thesis submitted to the
Department of Electrical Engineering
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY IN ELECTRONIC ENGINEERING

Faculty of Engineering
Capital University of Science and Technology
Islamabad
August 2016

Dedicated to my parents

ACKNOWLEDGMENT

First of all thanks to Almighty Allah Who granted me the courage, determination and environment to initiate my research. All respects and regards to Holy Prophet Muhammad (P.B.U.H) who is the last messenger and whose life is a perfect model for humanity.

I am greatly honored to have the supervision of Dr. Imtiaz Ahmad Taj whose guidance always kept me confident and buoyant at every single stage of my research work. I have always been thankful to have such a supporting personality as my research supervisor. His presence there has always helped me stay afloat and saved me from getting disappointed at any point in my research.

I would like to pay gratitude to the admirable Dr. Syed Muhammad Ziauddin as he has always been a role model for me. His consistent support, encouragement and direction always led me in the way of my research. Undeniably, it is due to his mentorship that I have been able to succeed in this endeavor without getting deviated. His acumen and expertise always served as a beacon for me with which I could see through the intricacies of subject matter objectively. His vision induced predictability in my line of action and his monitoring did not let me digress from the core of my research.

I would also like to commend the significant role of National ICT R&D Fund, Ministry of Information Technology and my parent organization Streaming Networks (Pvt) Ltd making available the opportunities of attaining higher education. Their vision and programs have created a conducive environment for those who are interested in conducting research that can bring value. Without their financial and technical support, it would be likely impossible for me to come up with this level of research work.

Especially, I would like to thanks my friend Dr. Maaz bin Ahmad for their constant cooperation and back-up in achieving this goal. They guided me throughout my research work and elucidated the way of confronting the hiccups in achieving every milestone in this way. In addition to the support I received from my colleagues at the Streaming Networks and university, I greatly acknowledge the noteworthy contribution of my friends Muhammad Tahir, Atif Raza, Imran ul Haq, Syed Ali Hassan, Waqas Ellahi and Moeen Tayyab for their support during my research and in accomplishing this dissertation.

I am indebted to my parents for their robust upbringing, their special prayers for my success, and the encouragement and support they provided in every achievement that I ever attained in my life. The patience, moral support and backing of my family during my PhD blessed me the leverage to focus on my research wholeheartedly. I am also grateful to my wife for her fortitude, understanding and incessant encouragement during the course of my research work. Put in a nutshell, this dissertation would not have been a possibility, had my family not gone out of their skin in supporting my efforts.

I highly appreciate the endorsement provided by my brothers Amir and Ahsan. They always helped me out whenever I lost focus on my research work due to some socio economical pressures. Their love and affection gave me the strength to stand tall whenever a hindrance came in my way to this research.

This dissertation is dedicated to my family for their love, deep understanding, endless patience and encouragement at all times. It is also dedicated to all those who helped me in any way in my research work. It is also dedicated to every feeling I had that convinced me that I was doing it right. It is dedicated as well to the never-ending credence from Allah, almighty that triggered self-belief and faith in me to make this work possible.

ABSTRACT

Video coding is an integral part of numerous real-time multimedia applications such as video telephony, telemedicine, video conferencing and video streaming. In real-time multimedia systems or power constrained systems, the coding performance of modern video coding standards such as High Efficiency Video Coding (HEVC) and H.264/MPEG-4 Advanced Video Coding (AVC), is limited by computational complexity. This thesis presents research work to develop techniques to reduce the computational complexity of video encoders and to exploit their data and task level parallelism. These techniques aim to provide significant complexity saving as well as improving coding efficiency.

A computationally efficient framework for macroblock prediction is developed to reduce the computational complexity and overheads related to the macroblock prediction process in video encoding. The framework consists of several innovative techniques to exclude as many intra and inter prediction modes as possible prior to the RDO (rate distortion optimization) process. In the best case, the proposed framework selects one MB type either intra or inter and one corresponding near-optimal prediction mode, so that the complete RDO process is neglected. Simulation results show that the proposed framework achieves significant complexity savings without any significant degradation in video quality.

In addition, a complexity reduction technique for motion compensation is developed to perform inter prediction. This addresses the computational complexity issues related to both interpolation and data manipulation modules of the motion compensation process. The end results of the experiments display that this method prominently decreases the computational complexity without loss in rate-distortion performance.

Finally, an end-to-end hybrid hardware-software implementation scheme based on pipelining and multitasking for advanced video coding is presented. This scheme exploits the task and data level parallelism in video encoders to improve their coding efficiency. The parallelism is exploited at both coarse-grain level and fine-grain level. The coarse-grain level parallelism exploitation is done by concurrently executing multiple tasks on different processing cores while fine-grain level parallelism is achieved by using SIMD (single instruction multiple data) instructions. Such exploitation of parallelism also helps to better utilize the computational power offered by advanced media processors. The outcomes of the experiments reveal that suggested scheme has resulted in enhancing the encoding rate and reducing power consumption.

In the field of video coding the main achievement of this research can be given in a nut shell as: (a) Development of computationally efficient techniques for macroblock prediction type and partition selection. (b) Development of complexity reduction algorithm based on intra and inter prediction mode selection. (c) Development of a computationally efficient scheme for motion compensation. Finally, (d) development of end-to-end hybrid implementation scheme for H.264/AVC encoder that exploits its data and task level parallelism to improve coding efficiency.

These innovative techniques may prove handy in real-time implementation of H.264/AVC and HEVC video encoders in computationally constrained environments as is the case in general purpose computers and low-power mobile devices.

LIST OF PUBLICATIONS

Muhammad Asif, Imtiaz A. Taj, S. M. Ziauddin, Maaz Bin Ahmad and M. Tahir " *A hybrid scheme based on pipelining and multitasking in mobile application processors for advanced video coding* ", Scientific Programming Journal, vol. 2015, Article ID 197843, 2015. doi:10.1155/2015/197843 (I.F 0.559)

Muhammad Asif, Imtiaz A. Taj, S. M. Ziauddin, Maaz Bin Ahmad and Atif Raza " *An efficient inter prediction mode selection scheme for advanced video coding based on motion homogeneity and residual complexity* ", IEEJ Transactions on Electrical and Electronic Engineering, (Article in Press) (I.F 0.213)

Muhammad Asif, Imtiaz A. Taj, S. M. Ziauddin, Maaz Bin Ahmad and M. Tahir " *An efficient framework for prediction parameters selection in advanced video coding* ", Elsevier Journal of Measurement, (Article Under Review) (I.F 1.484)

Muhammad Asif, Imtiaz A. Taj, S. M. Ziauddin, Maaz Bin Ahmad and M. Tahir " *An Efficient scheme for intra prediction block size and mode selection in advanced video coding* ", 13th International Conference on Frontiers in Information Technology (FIT2015), 2015

Muhammad Asif, Saqib Majeed, Imtiaz A. Taj, S. M. Ziauddin and Maaz Bin Ahmad " *Exploiting MB Level Parallelism in H.264/AVC Encoder for Multi-Core Platform* ", 2014 IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA), pp. 125-130, ISSBN/ISSN: DOI: 10.1109/AICCSA.2014.7073188, 2014

Muhammad Asif, Masood Farooq and Imtiaz A. Taj, " *Optimized Implementation of Motion Compensation for H.264Decoder* ", 5th International Conference on Computer Sciences and Convergence Information Technology (ICCIT2010), Pages: 216-221, ISSB-N/ISSN: ISBN: 978-1-4244-8567-3, 2010

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AVC	Advanced Video Coding
CABAC	Context-Adaptive Binary Arithmetic Coding
CAVLC	Context Adaptive Variable Length Coding
DCT	Discrete Cosine Transform
DVD	Digital Versatile Disc
DED	Dominant Edge Direction
DES	Dominant Edge Strength
DSP	Digital Signal Processor
GOP	Group-Of-Picture
GRC	Global Residual Complexity
IDCT	Inverse Discrete Cosine Transform
ISO/IEC	International Standards Organization
ITU-T	International Telecommunications Union
JVT	Joint Video Team
LRC	Local Residual Complexity
MAD	Mean Absolute Difference
MB	Macroblock
MPEG	Motion Picture Experts Group
MSAD	Minimum Sum Absolute Difference
MSE	Mean Squared Error
MV	Motion Vector
MBRP	Macroblock Region Partition
MC	Motion Compensation
ME	Motion Estimation
MPM	Most probable Mode
NAL	Network Abstraction Layer
PSNR	Peak Signal Noise Ratio
PDA	Personal Digital Assistant
PC	Personal Computer
PPE	Power Processor Element
QCIF	Quarter Common Intermediate Format
QP	Quantization Parameter
RD	Rate-Distortion
RDO	Rate-Distortion Optimization
SAD	Sum of Absolute Differences
SATD	Sum of Absolute Transform Differences
SPE	Synergistic Processors Elements
SIMD	Single Instruction Multiple Data
TPC	Tagged Procedure Calls
VCEG	Video Coding Experts Group
VLSI	Very-Large-Scale Integration
VLIW	Very-Long-Instruction-Word
CPU	Central Processing Unit

VPU	Video Processing Unit
DMA	Direct Memory Access Units
AMS	Asynchronous Multiprocessor
VMAU	Vector Matrix Arithmetic Unit
MCE	Motion Compensation and Estimation Engine
DBLK	De-Block
MIPS	Microprocessor without Interlocked Pipeline Stages

Chapter 1

INTRODUCTION

This is the 21st century, an era that heralds latest advancements in the field of computers and electronics. Notable among the new and hugely popular gadgets are the smart devices such as smart phones and tablet PCs with enhanced media processing capabilities including digital video encoding and decoding [1]. Increase in storage capacities and the availability of broadband technologies has made it possible to share videos with others in real-time. Despite the availability of higher processing power and larger storage disk capacity, there is still a need to represent video contents in a compact form in order to make video storage and transmission more efficient [2]. The emerging video coding standards address this problem, making it feasible to store high quality video on a limited-storage disc or transmit it within the limited-bandwidth that todays network can provide [3]. It also drives the development of many widely used storage and real-time video processing applications like Digital Versatile Disc (DVD), video conferencing, video on demand, video surveillance, video streaming and so on [4].

The compression and decompression functions for video image data are performed by the video codec (encoder/decoder) in order to store and transmit it in a suitable format [5]. The encoder part of the codec compresses the video signal and the decoder part reconstructs the original video on the receiver side [6]. In the past, a dedicated hardware was required to perform compression and decompression functions as these functions are computationally intensive [7]. However, with the advancement in technology, powerful embedded processors and DSPs are now available that have made software implementations feasible for video codecs [8]. These solutions offer more flexibility to the application designers and their performance has greatly improved [9].

The video encoding and decoding are highly demanding tasks that consume maximum processing power and also need efficient memory management and data handling. With the advent of high-definition (HD) video coding the processing requirements of codecs have increased tremendously [10],[11]. For example, the processor has to process 88.99 mega pixels per second for real-time encoding of full HD 1080p (1920×1080) video content. The performance of video encoders running on general purpose desktop or smart devices is therefore limited by their computational complexity [12]. This problem is significantly higher in case of real-time multimedia applications running on hand-held mobile devices [13], [14]. In a desktop system, the video encoder has to share resources with other applications, while in hand-held smart devices power consumption is closely related to processor utilization and it may be desirable to restrict computations in order to maximize battery life [15],[16]. Computational complexity of modern video encoders therefore becomes a major hurdle in achieving high coding efficiency on multimedia systems [17],[18].

Video coding standards have evolved primarily through the development of the well-known ISO/IEC Moving Picture Experts Group (MPEG) and ITU-T Video Coding Experts Group (VCEG) standards. The ISO/IEC produced MPEG-1 [19] and MPEG-4 Visual [20], ITU-T developed H.261 [21] and H.263 [22], and these two organizations jointly developed the H.262/MPEG-2 Video [23], H.264/MPEG-4 Advanced Video Coding (AVC) [24] and High Efficiency Video Coding (HEVC) [25] standards. The jointly developed standards particularly H.262/MPEG-2 and H.264/MPEG-4 have a strong impact on the industry and are used in wide ranging of products that are common in our daily lives.

H.264/AVC supersedes the previous coding standards due to its better performance in terms of compression, enhanced peak signal-to-noise ratio (PSNR) and visual quality. It provides 50% compression gain for equivalent perceptual quality as compared to the prior video standards [26]. In order to achieve better performance, this coding standard adopts several new techniques like directional prediction of intra coded blocks, variable block size motion estimation (ME), rate distortion optimization (RDO), multi-reference frame ME, integer transform (DCT), de-block filtering and context-based adaptive binary arithmetic coding (CABAC) [26]. The latest video coding standard HEVC provides 50% coding efficiency as compared to H.264/AVC for same video quality [27]. HEVC originates from H.264/AVC and retains its basic hybrid coding architecture. Moreover, the basic tool set of video encoding in HEVC standard is quiet similar to H.264/ AVC coding standard. However, HEVC uses additional tool set to improve compression performance and throughput speed (particularly for parallel processing architectures) [27] [28], [29].

This research work primarily focuses on H.264/AVC video coding standard because of its wide acceptance in many applications, including real-time conversational applications such as video conferencing and video chat, security applications, camcorders, video content acquisition and editing systems, Internet and mobile network video and broadcast of HD TV signals over different transmission media including cable, terrestrial transmission systems and satellite.

1.1 Motivation

The video encoders comprise a number of specified tools and techniques to enhance the compression efficiency by removing spatial, temporal and statistical redundancy of video data [30]. The prediction of each Macroblock (basic coding unit of 16×16 block pixels) is used to remove temporal and spatial redundancy. The prediction of each Macroblock(MB) is performed from regions of previous frames or the current frame that have already been encoded. Moreover, the standards support a numerous prediction options such as prediction type (intra or inter), prediction modes and suitable prediction block sizes etc. to perform the prediction process [31]. This flexible choice of MB prediction increases compression efficiency at the cost of computational complexity. To determine the prediction parameters that produce maximum coding efficiency, the encoder evaluates a numerous prediction options and tries to select the best.

H.264/AVC provides various MB modes for intra and inter prediction in order to better represent the spatial and temporal details of an MB [32], [33]. For intra prediction, the standard offers nine prediction modes of 4×4 luminance blocks (N4), four of entire 16×16 luminance components (N16) and four of chrominance component 8×8 (N8) of MB. For inter prediction, different prediction block sizes such as 16×16, 16×8, 8×16, 8×8, 8×4, 4×8 and 4×4 are supported. A technique called Lagrangian rate-distortion optimization (RDO) [34] is used for optimum mode selection by H.264 encoder in order to improve coding efficiency. To choose the best MB prediction mode, RDO examines a combination of all possible modes by calculating the Rate Distortion (RD) cost and then selects the mode with the minimum RD cost. Therefore, for each MB, the number of possible intra prediction mode combinations are $\text{N8}\times(16\times\text{N4} + \text{N16}) = 4 \times (16\times9 + 4) = 592$, and for inter prediction there are 20 different possible block size combinations. Thus, to select the best intra and inter prediction mode for one MB, the H.264/AVC encoder carries out 592 and 20 RDO calculations, respectively. As a result of this brute-force searching, the computational complexity of the encoder increases tremendously [32], [34], [35]. Most of the existing techniques related to complexity reduction of macroblock prediction process target either intra or inter mode selection. In order to reduce the computational complexity of the encoder, there is a need to develop an efficient framework for macroblock prediction that should not only target selection of optimum intra and inter prediction mode (variable block size) but also incorporate optimum prediction type (intra prediction or inter prediction) and macroblock partitions suitable for better intra prediction. Such a framework is a missing link in the existing research work.

The H.264/AVC standard supports motion estimation and compensation for inter prediction up to quarter pixel accuracy with multiple block sizes and multiple reference frames. According to the complexity analysis of Horowitz et al [36], motion compensation module takes about 25% of the decoding time. Moreover, it also takes a significant amount of encoding time. Hence, it is desirable to minimize the computational complexity of motion compensation algorithm to improve the speed of encoding and decoding process. Complexity analysis made by Horowitz et al. [36] is shown in Fig. 1.1.

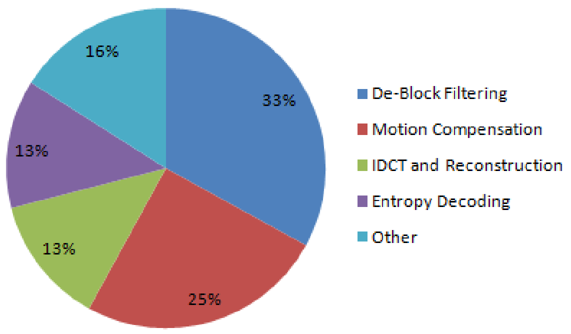


FIGURE 1.1: Complexity Analysis of H.264/AVC Decoder

Motion compensation comprises two sub-modules i.e. data manipulation and interpolation. Both of these modules contribute significantly to the overall computational complexity of motion compensation and may be optimized to improve the efficiency of H.264/AVC video codec. Most of the literature available in this area comprises techniques which either address data manipulation bottlenecks or interpolation issues. Only a few techniques cover both aspects but even these need further improvement.

Based on the above discussion it is evident that, development of a computationally efficient framework for macroblock prediction with negligible loss in coding efficiency is indispensable for high quality real-time video encoding. This constitutes one of the primary motivating factors for our research work.

The latest generation of mobile application processors is equipped with multiple-cores, dedicated hardware processing units, direct memory access (DMA) units and SIMD (Single Instructions Multiple Data) instruction set to handle the large amount of processing involved in signal processing algorithms and multimedia applications. H.264/AVC encoder involves an integrated framework where task and data dependency is very high [37]. There is a need to exploit its task and data level parallelism in order to improve the coding efficiency. Such exploitation of parallelism may also help to better utilize the computational power offered by advanced media processors [38].

Moreover, some processing architectures used in ultra-low powered devices offer a hybrid (hardware-software) encoding framework where a sub-set of encoding tools are available as hardware modules and the rest of the tools have to be implemented in software. The implementation of H.264/AVC encoder for these processors is a challenging task because H.264 encoder has an integrated framework where task dependency is large and the encoder tasks distribution, scheduling and synchronization among hardware and software modules is needed. Furthermore, it demands efficient memory management and simultaneous exploitation of all processing units for achieving real-time encoding. Most of the existing schemes fail to describe the hybrid implementation of H.264/AVC encoder. Hence, there is a need to work in this direction to highlight the issues involved in such kind of implementation.

1.2 Problem Statement

The aim of this research is to develop a computationally efficient framework for macroblock prediction in order to reduce the computational complexity of video encoders. This framework should greatly reduce the encoder's processing time with negligible loss in quality and increase in bit-rate. Moreover, the work aims to develop novel schemes which effectively exploit the data and task level parallelism in video encoders in order to improve their coding efficiency. These schemes should enable the encoders to make efficient use of the processing resources offered by latest generation of the mobile application processors to maximize the encoding rate, resulting in high quality video processing for real-time multimedia applications on general purpose desktop and smart devices.

The purpose of this research is to reduce the computational complexity of video encoders through efficient framework for macroblock prediction and to improve their coding efficiency by exploiting their data and task level parallelism

1.3 Research Objectives

In order to fulfill the aim of this research work the following are the research objectives:

- Develop a computationally efficient framework for macroblock prediction method that should reduce the computational complexity and overheads related to prediction process with the following features.
  - Select appropriate prediction type (intra or inter)
  - Decide appropriate partitioning size for intra prediction(4×4 or 16×16)
  - Select better intra prediction mode
  - Choose appropriate inter prediction mode for motion estimation (SKIP, 16×16, 16×8, 8×16, 8×8, 8×4, 4×8 or 4×4)
- Develop a computationally efficient technique for Motion Compensation (MC) to perform inter prediction that should speedup the video encoding and decoding process.

- Develop an end-to-end hybrid hardware-software implementation scheme that should exploit data and task level parallelism of video encoders to improve their coding efficiency. This scheme should enable the encoder to ensure efficient use of the computational power offered by the latest generation of mobile applications processors to increase the encoding rate.
- To ensure that all of these schemes are suitable for deployment on general purpose desktop and smart/mobile devices with limited processing power.

1.4 Research Contributions

The objectives of this work are achieved by developing several innovative approaches and schemes that reduce the computational complexity of video encoders and exploit their data and task level parallelism. Key contributions of this research work can be summarized as follows:

- **Efficient Framework for Macroblock Prediction**

  To reduce the computational complexity of the prediction process, an efficient framework for macroblock prediction is developed which comprises several new algorithms. The proposed framework is structured into several stages to exclude as many intra and inter prediction modes as possible prior to the RDO process without any significant degradation in video quality. It uses Adaboost classifier, spatial and temporal features and adaptive thresholds to achieve this goal. The proposed framework not only selects optimum intra and inter prediction mode but also incorporates optimum prediction type (intra or inter) and macroblock partitions to improve the prediction process and reduce its computational complexity. The development of these new algorithms led to a conference paper [39] and a journal publication [40].

- **Motion Compensation Complexity Reduction**

  To perform inter prediction a technique for motion compensation is developed that significantly reduces the complexity of data manipulation and interpolation modules to achieve real-time implementation of video encoder and decoder. The development of this new technique led to a conference paper [41].

- **Parallel Task Assignment in Hybrid Hardware-Software Video Coding**

  An end-to-end hybrid hardware-software implementation scheme based on pipelining and multitasking in mobile application processors for advanced video coding is proposed in this work. This scheme exploits the task and data level parallelism in video encoders to improve their coding efficiency. The parallelism is exploited at both coarse-grain level and fine-grain level. The coarse-grain level parallelism exploitation is done by concurrently executing multiple tasks on different processing cores while fine-grain level parallelism is achieved by using SIMD instructions. Such exploitation of parallelism also helps to better utilize the computational power offered by advanced media processors. The development of these new schemes led to a conference paper [42] and a journal publication [43].

There are two main areas that lead to the improvement of video coding efficiency. The first one belongs to complexity reduction of computationally intensive modules of video codecs and the second covers the exploitation of their parallelism. The proposed work targeted both of these areas. The first two contributions are related to complexity reduction of macroblock prediction process to increase the coding rate. These contributions optimize the prediction parameters selection technique RDO and motion compensation (MC) algorithm to perform the inter prediction for an MB. The third contribution exploits the task and data level parallelism in video encoders to improve their coding efficiency.

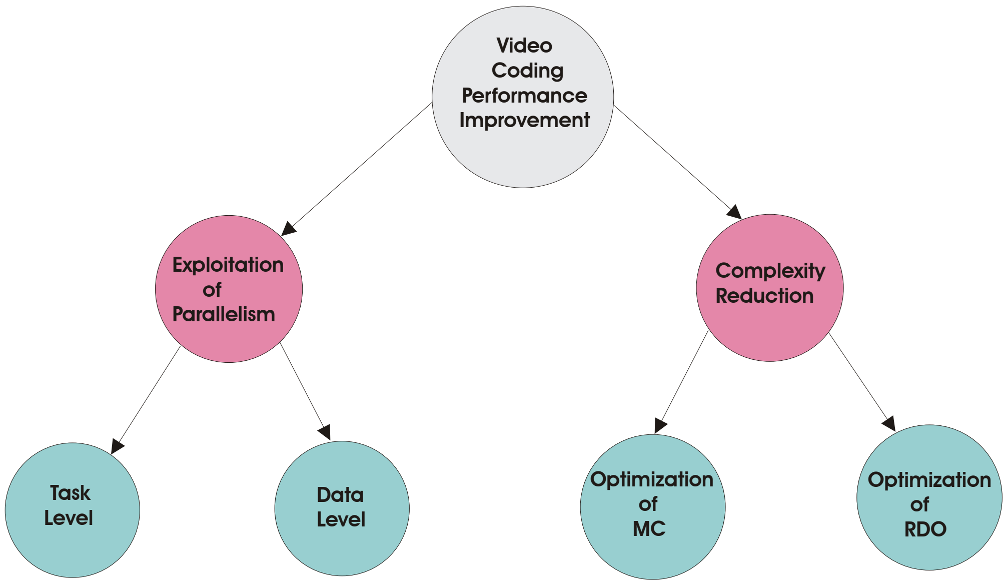


FIGURE 1.2: Improvement of Video Coding Performance vs Research Contributions

The Fig. 1.2 shows the areas that lead to the improvement of video coding efficiency and also highlights the research contributions made in this research thesis.

1.5 Thesis Organization

The organization of the thesis is made as follows:

Chapter 1 – (This one) Presents introduction to the thesis and gives an overview about the importance of the research problem. It address in detail the video coding, importance and applications of video coding, why latest video coding standard demands large amount of computational resources to perform real-time encoding. This chapter also explains the macroblock prediction and parallel task assignment problem of video coding and the motivation behind this research work.

Chapter 2 – Gives an overview of the H.264/AVC video coding standard. The tools and techniques used in this standard to enhance the compression efficiency are briefly discussed. This chapter also highlights the reasons why high computational resources are required for the implementation of H.264/AVC coding standard.

Chapter 3 – Describes the previous work done in the area of macroblock prediction and parallel task assignment in H.264/AVC. It also categorizes the work done in this area into different categories and addresses advantages and disadvantages of the available literature.

The development of innovative approaches and algorithms to curtail the computational complexity of the macroblock prediction process and parallel task assignment in hybrid video coding are presented in Chapter 4 - Chapter 6.

Chapter 4 – This chapter briefly describes the proposed computationally efficient framework for macroblock prediction which uses Adaboost classifier, spatial and temporal features and adaptive thresholds. Computational saving is achieved by reducing candidate predication modes prior to RDO calculation. This constitutes the low-complexity framework developed in this work for H.264/AVC. Experimental analysis of the suggested framework is also covered in this chapter.

Chapter 5 – Presents the proposed scheme for Motion Compensation (MC) to perform inter prediction that minimizes its computational complexity for real-time implementations of H.264/AVC video encoder and decoder. Experimental analysis of the proposed motion compensation scheme is also presented in this chapter.

Chapter 6 – This chapter describes the proposed hybrid software-hardware implementation scheme for advanced video coding. This scheme exploits data and task level parallelism of video encoders to improve coding efficiency. The performance evaluation of the recommended scheme is also made in this chapter.

Chapter 7 – This last chapter presents conclusion and future directions of this research work. A summary of the main contributions made in the field of video processing also describe in this chapter.

Chapter 2

H.264/AVC VIDEO CODING STANDARD

2.1 Introduction

The H.264/AVC video coding standard is jointly developed by the ITU-T Video Coding Experts Group (VCEG) and ISO/IEC Moving Picture Experts Group (MPEG). This coding standard evolve with the aim to give enhanced compression efficiency as compared to predecessors. This chapter describes an overview of H.264/AVC standard. The rest of the chapter is organized as follows. Section 2.2 describes the H.264/AVC encoding and decoding flow. Section 2.3 focuses on some of the specific coding tools available in the video coding layer. Section 2.4 summarizes the H.264/AVC codec.

2.2 The H.264 Codec

H.264 defines syntax and methods of encoding /decoding video bit-stream rather than explicitly defining a CODEC (enCOder / DECoder pair). H.264 compliant encoder/decoder consists of several functional elements. Most of these elements are also the part of previous coding standards (H.263, H.261, MPEG-4, MPEG-2, MPEG-1) [44]. The de-blocking filter functional block is the only exception. Also the details and working of these functional blocks differ from previous standards. As shown in the Fig. 2.1. the encoder part of the codec comprises of two data flow paths. First path is from left-to-right called forward path and the other is from right-to-left called reconstruction path. The Fig. 2.2. of decoder shows data flow path from right-to-left in decoder representing the similarities between decoder and encoder. Here, first we present the main process of encoding/decoding a frame of video stepwise before going in details of H.264/AVC.

2.2.1 Encoder

2.2.1.1 Forward Path

In H.264/AVC coding standard, macroblock (16×16) is the fundamental processing unit of a video frame/picture. It may be encoded using intra or inter prediction mode. The pixel values belonging to the macroblocks of the current, previous and future frames (that have already been encoded, decoded and reconstructed) are used to generate a prediction signal. To predict the macroblock in intra mode, the neighboring samples in the current frame/slice are used. On the other hand, to predict the macroblock in inter mode, motion compensation is performed that makes use of sets of list 0 and/or list 1 reference frames.

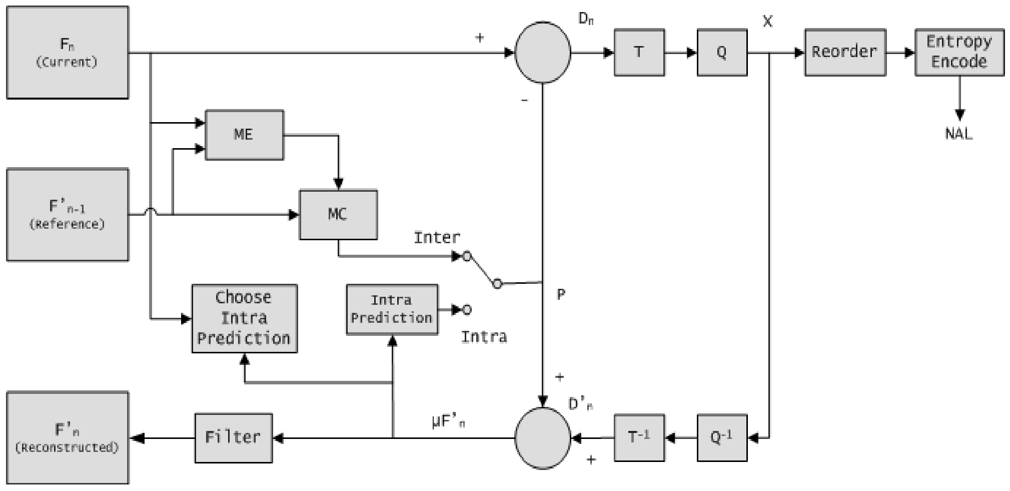


FIGURE 2.1: H.264/AVC Encoder

As illustrated in Fig. 2.1 the previous encoded frame is used to represent the reference frame. Each macroblock partition can be predicted either from past or from future reference frame. These frames previously have been encoded, reconstructed and filtered. In order to obtain a residual block Dn, from the current block, the prediction is subtracted. After that, residual block under goes the process of transformation and quantization. The resultant set of quantized transform coefficients is denoted by X. The X is first reordered and then entropy encoding is done with it. Along with side information, the coefficients of entropy encoded are required in order to decode each block within the macroblock. Then a compressed bit-stream is formed to pass it to a Network Abstraction Layer for storage or transmission.

2.2.1.2 Reconstruction Path

Along with the encoding and transmission of each block in a macroblock, the encoder also provides a reference for more predictions. It reconstructs (decodes) each block in a macroblock to achieve this goal. The difference block D'n is obtained by inverse scaling (IQ) the coefficients X and inverse transforming (IDCT). The reconstructed block uF'n is created by adding together the prediction block and difference block. The reconstructed block is unfiltered thus to minimize the effects of blocking distortion, a filter is applied. A series of blocks uF'n is used to create the reconstructed reference picture.

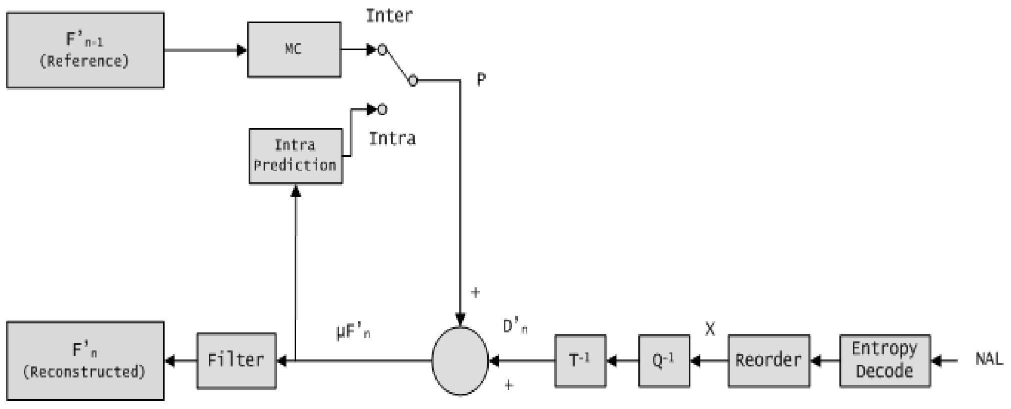


Figure 2.2: H.264/AVC Decoder

2.2.2 Decoder

The decoder produces a decoded frame from compressed bit-stream. Initially entropy decoding is performed on compressed bit-stream to obtain quantized coefficients X. After this, decoder performs inverse quantization and inverse transform using X in order to produce residual block D'n. Then it produces a prediction block by making use of the decoded header information of the bit-stream. The prediction block and residual block are added together to produce reconstructed block uF'n which after filtering results in each decoded block. A series of blocks uF'n is used to create the reconstructed decoded frame. The Fig. 2.2 shows the block diagram of H.264/AVC decoder.

2.3 Functional Blocks

The H.264/AVC standard provides a high level of compression efficiency by incorporating wide range of coding tools. The important functional blocks and coding tools of this standard have been described in the following sections.

2.3.1 Intra Prediction

In H.264/AVC intra prediction makes use of spatial correlation between adjacent macroblocks/blocks. In intra prediction, the prediction signal for current macroblock is formed with the help of adjacent pixels of previously encoded and reconstructed top and left macroblocks. For the luminance (luma) component Y the prediction signal can be generated for 16×16 macroblock or for each 4×4 sub-block of macroblock. Nine predictions modes are used to predict 4×4 luma blocks. To predict the luma 16×16 and 8×8 chrominance (chroma) components (U and V) four prediction modes are used.

2.3.1.1 Intra 4×4 Prediction for Luma Samples

In intra 4×4 prediction modes, the prediction block is formed by the adjacent pixels of top and left 4×4 blocks. The 4×4 prediction blocks sizes are useful to encode high detail areas of frames. The intra prediction modes for luma 4×4 block are illustrated in Fig. 2.3. The arrows show the prediction directions while shaded squares represent the already decoded neighboring pixels.

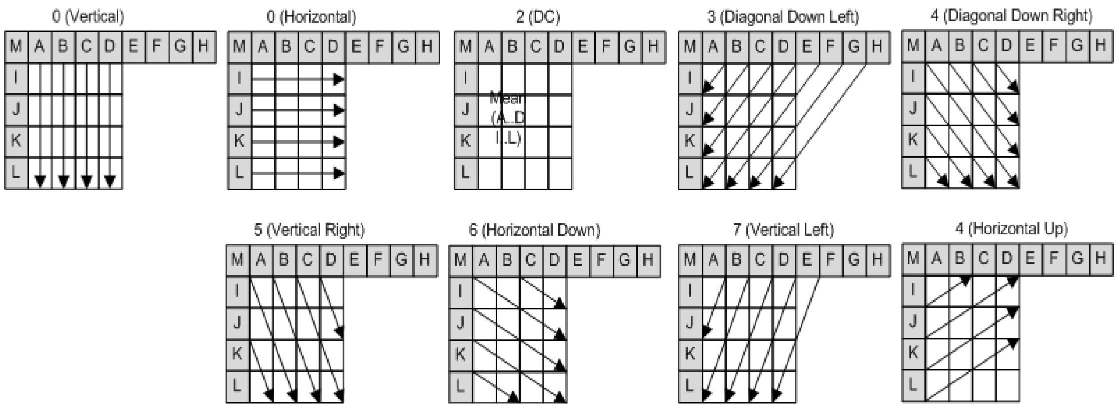


FIGURE 2.3: 4×4 Luma Intra Prediction Modes

The Fig. 2.4 shows a 4×4 luma block to be predicted. The prediction block comprises of the samples a,b,c...p. These samples can be generated using A-M samples as follows. For example Mode 2 (DC prediction) is varied according to the availability of samples A-M which have previously been coded. Based on the availability of all required prediction samples, all other modes can be used.

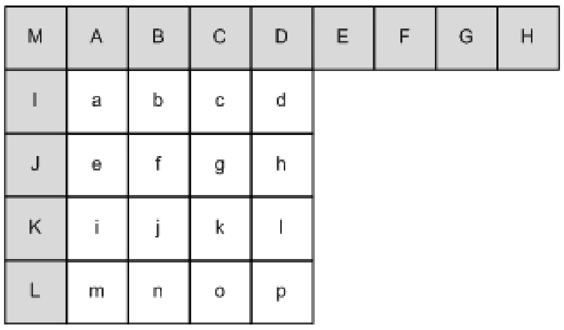


FIGURE 2.4: 4×4 Luma Intra Predicted Samples Generation

Weighted average of the prediction samples A-M are used to form the predicted samples for modes 3-8. e.g. if selected mode is 4, the following (2.1) is used to generate top-right sample 'd'.

$$d = round(\frac{B}{4} + \frac{C}{2} + \frac{D}{4}) \tag{2.1}$$

The detail of nine intra prediction modes for 4×4 luma blocks is presented in Table 2.1.

Table 2.1: Intra 4×4 Luma Prediction Modes

Mode	Name	Description
0	I-Pred-4×4-Vertical	The vertical extrapolation of A,B,C,D samples.
1	I-Pred-4×4-Horizontal	The Horizontal extrapolation of I,J,K,L samples.
2	I-Pred-4×4-DC	The mean of A...D and I...L. samples.
3	I-Pred-4×4-Diagonal Down-Left	The interpolation of samples between upper right and lower-left at a 45 angle.
4	I-Pred-4×4-Diagonal Down-Right	The extrapolation of samples down and to the right at a 45 angle.
5	I-Pred-4×4-Vertical-Right	The interpolation of samples to the left of vertical at an angle of approximately 26.6 .
6	I-Pred-4×4-Horizontal-Down	The interpolation of samples below horizontal at an angle of approximately 26.6.
7	I-Pred-4×4-Vertical-Left	The interpolation of samples to the right of vertical at an angle of approximately 26.6 .
8	I-Pred-4×4-Horizontal-Up	The extrapolation of samples above horizontal at an angle of approximately 26.6 .

2.3.1.2 Intra 16×16 Prediction for Luma Samples

A single operation may be used to predict the entire 16×16 luma component of a macroblock. There are four prediction modes available for it as shown in the Fig. 2.5.

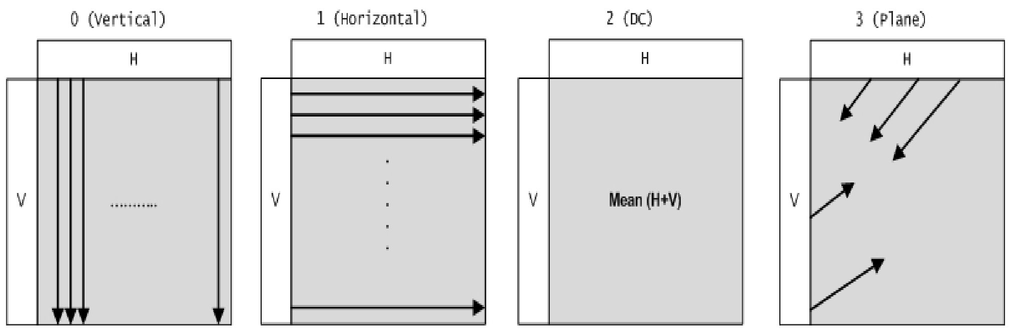


Figure 2.5: 16×16 Luma Intra Prediction Modes

These are suitable for homogeneous areas not containing much of the details. Table 2.2 illustrated the prediction modes for intra 16×16 .

Table 2.2: Intra 16×16 Luma Prediction Modes

Mode	Name	Description
0	I-Pred-16×16-Vertical	The upper samples (H) are extrapolated.
1	I-Pred-16×16-Horizontal	The left samples (V) are extrapolated.
2	I-Pred-16×16-DC	Mean of left hand and upper samples (H+V).
3	I-Pred-16×16-Plane	A linear Plan function is fitted to the left-hand samples V and upper H.

2.3.1.3 Intra 8×8 Prediction for Chroma Samples

The 8×8 chroma component is predicted without partitioning because the chroma samples are more homogeneous than luma samples. Both chroma components (U and V) are predicted using same prediction mode. The four prediction modes of chroma are same as that of luma 16×16 illustrated in Fig. 2.5. having only the difference in their numbering. The detail of these modes is presented in Table 2.3.

Table 2.3: Intra 8×8 Chroma Prediction Modes

Mode	Name	Description
0	I-Pred-Chroma-DC	Mean of left hand and upper samples (H+V).
1	I-Pred-Chroma-Horizontal	The left samples (V) are extrapolated.
2	I-Pred-Chroma-Vertical	The upper samples (H) are extrapolated.
3	I-Pred-Chroma-Plane	A linear 'Plane' function is fitted to the left-hand samples V and upper H .

2.3.1.4 I_PCM

A different kind of intra coding mode called I_PCM which does not use prediction, transformation and quantization. The sample values are transmitted directly. This mode is useful in scenarios where information loss may not be tolerable.

2.3.2 Inter Prediction

The inter prediction is performed to remove the temporal redundancy between the frames. This tool contributes a lot to improve the compression rate of H.264/AVC as compare to predecessors. Block based motion estimation and compensation is used to create the inter prediction signal from one or more reference frames or fields.

2.3.2.1 Variable Block Size Motion Compensation

The macroblocks are divided into macroblocks and sub-macroblocks partitions to perform motion compensation. The partitions of luma component of macroblocks for motion compensation are shown in Fig. 2.6. Each macroblock can be motion compensated as four 8×8 partitions, two 8×16 partitions, two 16×8 partitions or one 16×16 macroblock partition.

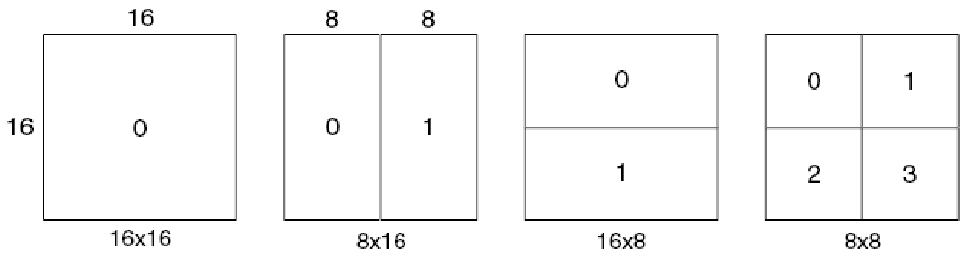


FIGURE 2.6: Macroblock Partitions for Luma Component

There are four ways to further divide the every 8×8 partition of macroblock as shown in Fig. 2.7 for the case if the 8×8 mode is selected. For each partition of macroblock or sub-macroblock separate motion vector is required. The choice of partition(s) along with each motion vector has to be coded.

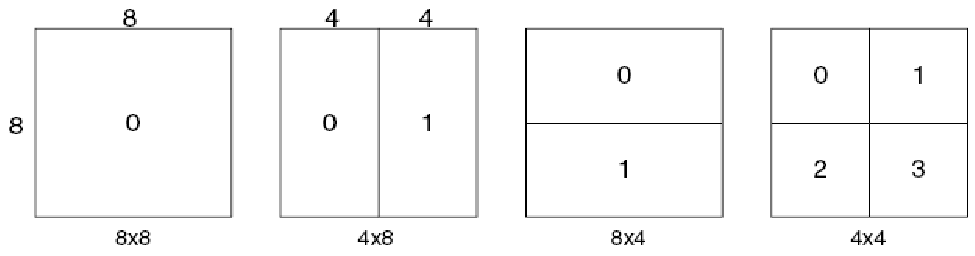


FIGURE 2.7: Sub-Macroblock Partitions for Luma Component

Selection of large partition size (8×16, 16×8 and 16×16) implies that choice of motion vector(s) and partition type may be signaled by using fewer bits. On the other hand, sufficient energy may be carried by motion compensated residual. Selection of small partition size (4×4, 4×8, 8×4 and 8×8) implies that choice of partition type and motion vector(s) must be signaled by using large number of bits. On the other hand, lower energy in frame areas may be carried by motion compensated residual.

In a macroblock, every chroma component (Cb and Cr) has half of the horizontal and vertical resolution of the luma component. Both chroma block and luma block are partitioned in the same way. The only difference is the partition sizes which have exactly half the horizontal and vertical resolution (an 8×16 partition in luma corresponds to a 4×8 partition in chroma; an 8×4 partition in luma corresponds to 4×2 in chroma and so on). Every motion vectors components are halved when applied to the chroma blocks.

2.3.2.2 Motion Vectors

Motion estimation and compensation is performed to generate the inter prediction signal of same size from one or more reference frames for each macroblock or sub macroblock partition. The offset of reference frames matching area is indicated by motion vectors. The H.264/AVC support motion vectors up to quarter and 1/8th pixel for luma and chroma components, respectively. The reference frame does not have luma and chroma pixel values corresponding to sub-pixel position. Therefore, interpolation from neighboring pixels values is performed to generate them.

In Fig. 2.8. in order to predict a block of size 4×4 in the current frame (a) an area of the reference frame in the neighborhood of the current block position is used. The motion vector for which both of its components (horizontal and vertical) are integers (b), the prediction samples actually exist in reference frame as shown by gray dots. For fractional values of vector component(s) (c), interpolation is performed in reference frame among adjacent samples (as shown by white dots) to generate prediction samples.

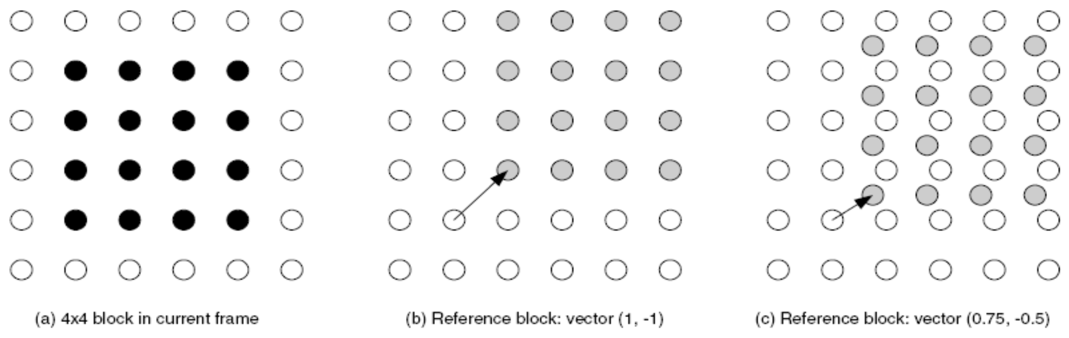

(a) 4x4 block in current frame
(b) Reference block: vector (1, -1)
(c) Reference block: vector (0.75, -0.5)

FIGURE 2.8: Example of Integer and Sub-Sample Prediction

2.3.2.3 Generating Interpolated Samples

For luma sample interpolation, first the half-pel samples between full-pel sample positions in the reference frame are generated, indicated by gray markers in Fig. 2.9. To obtain a half-pel sample values, a 6-tap (1/32,-5/32, 5/8, 5/8,- 5/32, 1/32) FIR filter is used. For example, the six horizontal integer samples K, L, M, N, P and Q are used to calculate the the half-pel sample 's' (2.2) .

$$s = round((K - 5L + 20M + 20N - 5P + Q)/32) \tag{2.2}$$

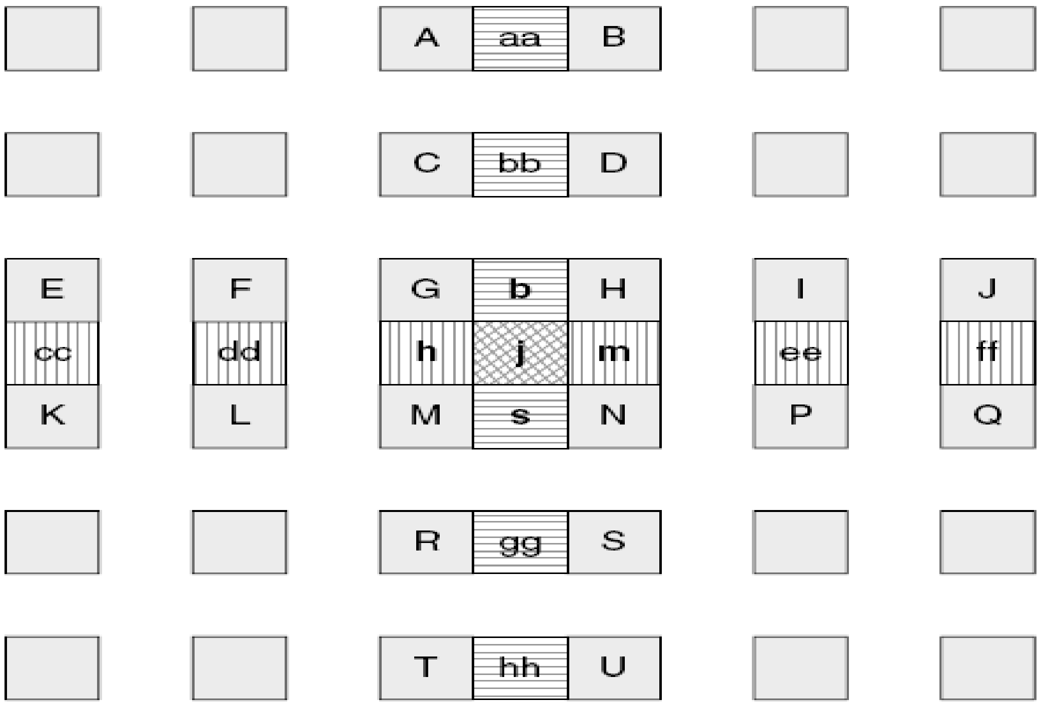


FIGURE 2.9: Interpolation of Luma Half-Pel Positions

Similarly, filtering of B, D, H, N, S and U interpolates the half-pel sample 'm'. After the calculation of all half-pel samples values vertically and horizontally adjacent to full-pel samples, interpolation is performed among 6 vertical or horizontal half-pel samples in order to calculate remaining half-pel positions. For example, filtering ff, ee, m, h, dd and cc generates 'j'.

The quarter pixel (e.g. k, i, c, a and q, n, f, d in Fig. 2.10) values are achieved through linear interpolation of full pixel and/ or half pixel values for example (2.3) :

$$a = round((G + b)/2) \tag{2.3}$$

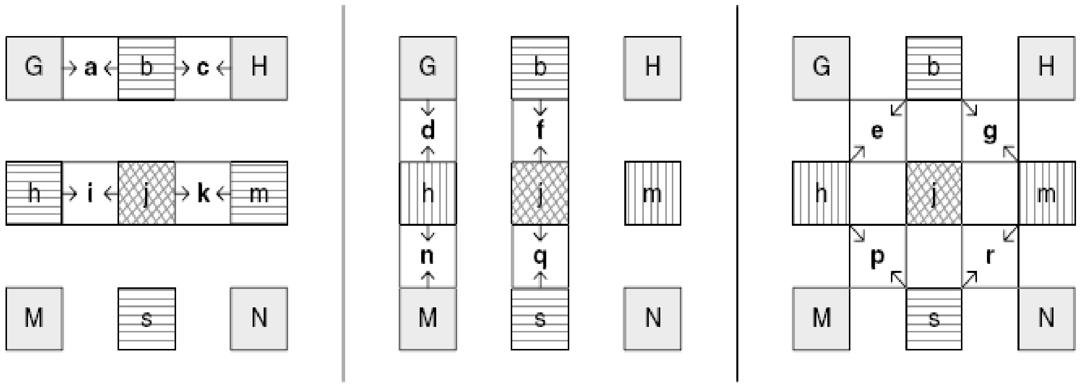


FIGURE 2.10: Interpolation of Luma Quarter-Pel Positions

The interpolation between a pair of diagonally opposite half-pel samples gives the the rest of quarter-pel positions (r, p, g and e in the Fig. 2.10). For example, interpolation between 'b' and 'h' gives 'e'.

For 4:2:0 sampling for luma component the motion vectors of quarter-pel resolution needs vectors of eighth-sample resolution in the chroma components. For each chroma component linear interpolation is used to generate interpolated samples at eighth-sample intervals between integer samples as shown in Fig. 2.11.

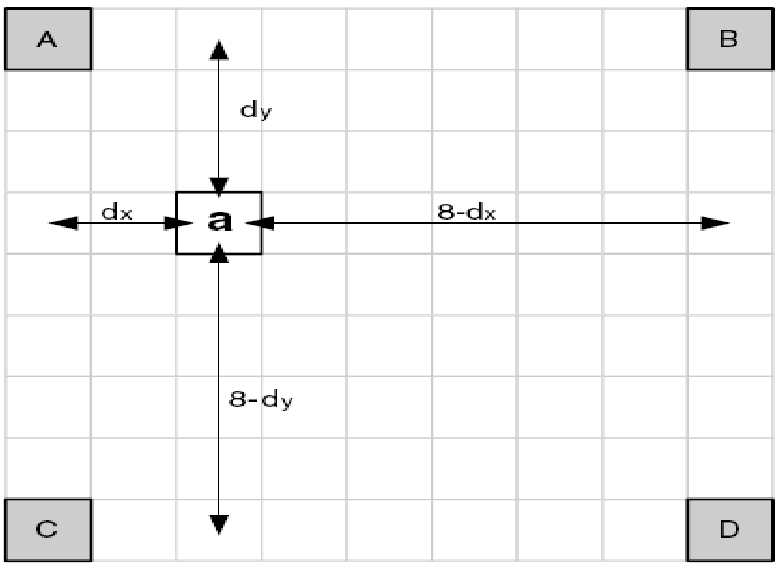


FIGURE 2.11: Interpolation of Chroma Eighth-Sample Positions

The linear combination A, B, C and D neighboring integer sample positions gives the sub-sample ‘a’ (2.4) :

$$a = round([(8-d_x)\cdot(8-d_y)A+d_x\cdot(8-d_y)B+(8-d_x)\cdot d_yC+d_x\cdot d_yD]/64) \quad (2.4)$$

2.3.3 Skipped Macroblocks

Skipped macroblocks are specialized type of macroblocks specified by H.264/AVC. For it no coded information is sent to the decoder. The skipped macroblocks are indicated to the decoder with the help of a syntax element in slice data. These macroblocks are named as P-Skip if present in P slices and in B-Slices are called B-Skip. While using P-Skip mode for encoding a particular MB, the decoder assumed the several conditions:

- 16x16 macroblock partition (i.e., macroblock have a single motion vector).
- Actual Motion vector is same as that of predicted motion vector. The X, Y component of motion vector difference (MVD) is zero.
- The previous frame in display order is the chosen prediction reference (list0 reference index is 0).
- After transform and quantization, all coefficients are zero.

For a B-Skip coded macroblock, the decoder perceives that the macroblock has zero quantized transform coefficients. No motion information or residual data is used by the decoder for the skipped macroblock. The predicted motion vector is used to generate the predicted macroblock because it is equivalent to actual motion vector(s) and the motion vector differences are zero. So, the predicted macroblock become the reconstructed macroblock. The predicted MB is very similar to the original MB because skipped macroblocks happen in regions with low movements.

If CAVLC is used as entropy encoding scheme then “mb_skip_run” parameter is used to inform the decoder about number of consecutive skipped macroblock since the last coded macroblock.

2.3.4 Transformation

The subtraction of predicted macroblock from original macroblock results in residual macroblock. In order to remove spatial correlation, the residual macroblock is transformed. Some profiles (Baseline, Main and Extended) use 4x4 integer transform based on DCT [44, 45]. In these profiles, in order to transform the macroblock residual data, it is divided into 4×4 blocks. On the other hand, ‘High’ profiles adaptively make use of 8×8 integer transform [27] along with 4×4 integer transform for the inter macroblock partitions (8×8, 8×16, 16×8,16×16). If intra 8×8 mode is used to predict the macroblocks, 8×8 transform is used in ‘High’ profiles. ‘Hadamard transform’ (4×4) [46] is used only in intra 16×16 prediction mode to further transform DC coefficients of the 4×4 transform blocks. ‘Hadamard transform’ (2×2) is used to further transform DC coefficients of each chroma component.

2.3.5 Quantization

The transformed residual data is quantized in order to achieve lossy compression. Quantization Parameter (QP) is used to describe the quantization step size. The QP lies in the range -(Offset) to 51, where, Offset = 6×(Bit depth-8). The sample accuracy is reflected by Bit depth. So, QP ranges from 0 to 51 for 8-bit samples. Every additional bit in bit depth causes increment by 6 in the range. Every 6 increments of the QP results in doubling the quantization step size. It may allow the wide range of image quality and bit-rates. The chroma quantization value is extracted from the specified luma quantization value. Different quantization parameters values may be used to encode every macroblock. Qp is coded differentially so only change in QP is transmitted to decoder. The encoder may select different QP values in order to independently encode both chroma components. In order to enhance quality of image, the encoder uses perceptual-based quantization scaling matrices and may modify quantization scaling factors for transform coefficients.

2.3.6 Reordering/Scanning

Before performing entropy coding quantized transform coefficients are arranged in linear fashion. The scanning process projected all the non zeros coefficients together. The block may be a progressive or interlaced frame block which determines its order of scanning. For progressive frame blocks the gathering of non zero coefficients tends to occur around top left DC coefficient. On the other hand, for interlaced frame blocks the probable gathering of non zeros coefficients occur at top left of the block. If an intra 16×16 mode is used for coding a MB, same pattern is used for scanning the DC coefficients of 4×4 luma blocks. In the luma blocks the remaining coefficients are scanned starting from the first AC coefficient. Raster scan order is used to scan DC coefficients of the chroma components. Scanning of AC coefficients is performed in the Zig-Zag order.

2.3.7 Entropy Coding

In order to reduce the size of coded data, entropy encoding incorporates loss-less coding techniques. The encoded data may be picture/sequence parameter sets, slice data, slice headers, macroblock modes, motion vectors and macroblock residual coefficients. The type of data determines different entropy coding to be used with it. e.g. in order to encode syntax elements of higher layer and slice headers universal variable length codes or fixed length codes are used. Context Adaptive Binary Arithmetic Coding (CABAC) or Context Adaptive Variable Length Coding (CAVLC) are used to encode slice data and other macroblock layer syntax depending on the chosen entropy coding mode.

The probability of occurrence determines the assignment of CAVLC variable length codes to data elements. To more frequently occurring data element shorter codes are allocated while longer codes are allocated to less frequently occurring data elements. According to local sequence statistics, different code tables are used. It results in more efficient coding. CABAC coding is done only to residual coefficients. Though it achieves higher compression efficiency but at a cost of more computational complexity. Following stages are involve to encode data element through CABAC:

- Binarisation - Syntax elements e.g. motion vectors or a transform coefficient contain non-binary values. Before arithmetic coding these elements are binnarized i.e. translated to a string of binary code. These binary strings are individually decodable code words. A bin is a notation used to represent each position in the binary string.
- Context Modeling - The context determines the assigned probability model to each bit. It means that the probability estimate of each bin relies on the context such as the syntax element type and previous syntax elements values. Actual bin values are feed backed after encoding each bin in order to update these context models.
- Binary Arithmetic Coding - Current bin probability estimates and bin values help in selecting the sub range, thus helping the arithmetic coding.

The compression efficiency of CABAC is 9% - 14% higher than CAVLC [47].

2.3.8 De-blocking Filter

Blockiness in the reconstructed picture may resulted from the quantization of block transform coefficients. The de-blocking filter is specified by the H.264/AVC standard to minimize blocking artifacts. The reconstructed and filtered frames serve as reference frame to perform inter prediction. Both the encoder and the decoder use the same filter parameters to avoid any prediction errors. Rather than unfiltered reconstruction, a filtered frame yields a closer match to the actual frame. So, in order to obtain a higher objective and subjective quality, better inter prediction might be achieved using the filtered reference frame [48]. The de-block filtering is performed to smooth the horizontal or vertical edges of 4×4 block in macroblocks. The filtering is not performed for edges on slice boundaries. The filter strength (amount of filtering) is the function of neighboring blocks coding modes, quantization parameters and the gradient of frame pixel values across the edges. Also, the encoder can explicitly change filter strength or may completely turn o the filter.

2.3.9 Rate Distortion Optimization

Rate Distortion Optimization (RDO) is a technique used by H.264/AVC encoder for optimum coding parameters selection. It combines the distortion (D) and bit-rate (R) into a single cost function (J) known as rate-distortion cost (RDcost) (2.5) .

$$J = D + \lambda R \tag{2.5}$$

The ultimate objective of RDO mode selection technique is to search for a mode to minimizes the RDcost. The Lagrange multiplier λ in the above equation is used to control the trade-off between rate and distortion. A smaller value of λ help in minimizing the distortion at the cost of higher bit-rate. A larger value of λ minimizes the bit-rate with the increase in distortion. So, to find out the best λ value for a video sequences is not an easy task [49]. The empirical approximations techniques helps in deciding an effective choice for the value of λ in any practical scenario of mode selection [34]. By calculating λ as a function of QP may result in good value selection (2.6) [50].

$$\lambda = 0.852^{\frac{QP-12}{3}} \tag{2.6}$$

Sum of Squared Distortion (SSD) is used to calculate the distortion (D) (2.7) :

$$D_{SSD} = \sum_{x,y} (b(x,y) - b'(x,y))^2 \tag{2.7}$$

Where x, y are the sample positions in block, b(x,y) indicate the actual pixel values and decoded pixel values are denoted by b'(x,y). There are some other distortion metrics helpful for the selection of best motion vector for a block e.g. Sum of Absolute Transformed Differences (SATD) or Sum of Absolute Differences (SAD). A different λ calculation is required for a different distortion metric. To find the most appropriate mode for a MB, the steps of RDO algorithm are:

For each macroblock

- For every possible coding mode n.
  - Encode the macroblock using mode n and calculate R, the number of bits required to encode the macroblock.
  - Reconstruct the macroblock and calculate the distortion D between the actual and decoded macroblocks.
  - Calculate the RDcost Jn for mode n with the help of equation (2.5) using appropriate value of λ.
- Select the mode that minimizes the RDcost Jn.

Integer DCT, Q, IQ, and IDCT have been used to calculate the distortion. On the other hand, entropy encoding of motion vector, prediction mode, quantized transform coefficients and quantization parameter contribute in the rate calculation.

This is an exhaustive search algorithm so it is highly computational intensive. The reason is that there may be hundreds of possible mode combinations to determine the best encoding mode MB needs to coded hundreds of times.

2.4 Summary

The H.264/AVC coding standard provides better compression efficiency as compared to former coding standards. This compression efficiency is achieved by adding a number of new compression techniques including spatial intra-prediction, (4×4) integer transform, variable block sizes for inter frame coding, quarter pixel motion estimation, multiple reference frame, de-block filtering and content-adaptive arithmetic coding. Additionally, this coding standard provides a large encoding options such as prediction type (intra or inter), prediction modes and suitable prediction block sizes. Moreover, this standard indicates only the encoding syntax and the decoding mechanism of the encoded bit-stream. This empowers a flexible encoder implementation. However, the encoder evaluates all the possible coding modes and block sizes during encoding process in order to obtain the best possible rate-distortion performance that significantly increased the computational complexity. Therefore, encoder implementation should effectively exploit the available encoding tools and parameters to acquire the desired compression efficiency with possible processing capabilities.

Chapter 3

LITERATURE REVIEW

H.264/AVC video encoding requires significant computing power, especially when dealing with real-time high definition (HD) content. There are two research areas that lead to the improvement of its video coding efficiency. The first one belongs to complexity reduction of computationally intensive blocks of H.264/AVC encoder and the second covers the exploitation of its parallelism. This chapter presents the existing work related to complexity reduction of macroblock prediction parameters selection process and motion compensation module of H.264/AVC video coding. It also describes the existing schemes for implementing H.264/AVC encoder that exploit its parallelism to improve coding efficiency.

The arrangement of the remaining chapter is made as follows. Section 3.1 describes the existing techniques for complexity reduction of macroblock prediction parameters selection process. Section 3.2 describes the reasons of computational complexity of JM reference software implementation of motion compensation technique and existing techniques for its complexity reduction. Section 3.3 presents the literature review of parallelism exploitation schemes for implementing H.264/AVC video encoder. Finally, Section 3.4 summarizes this chapter.

3.1 Macroblock Prediction Parameters Selection

In H.264/AVC video coding standard, macroblock prediction is used to remove temporal and spatial redundancy. This standard supports a large number of prediction options such as prediction type (intra or inter), prediction modes and suitable prediction block sizes etc. to perform prediction process. In H.264/AVC, RDO technique is employed to evaluate the all possible prediction options and selects the better one. The RDO calculates the RDcost for all possible coding modes and then selects the mode with minimum RDcost as the optimal mode. The RD-cost calculation increases the computational complexity because it involves both encoding and decoding processes. To achieve real-time encoding, this computational complexity becomes a bottleneck. So, it is highly desirable to implement computationally efficient techniques for macroblock prediction parameters selection to reduce the encoder complexity without any significant coding loss.

In literature several techniques for macroblocks predictions mode selection have been reported. Some targeted inter prediction mode selection while others addressed intra prediction mode selection.

3.1.1 Inter Prediction Mode Selection Techniques

The existing techniques for inter prediction mode selection can be generally divided into four different approaches. The first approach is based on sum of absolute difference (SAD) in which current macroblock (MB) SAD is compared with certain threshold to shortlist the candidate inter prediction modes for RDO process. Jing and Chau [51], presented inter prediction mode decision algorithm in which sum of absolute differences (SADs) are compared with predefine threshold to decide best inter prediction mode. Kim et al. [52] exploited the coefficient thresholding and all zero quantized coefficients block detection to eliminate the unnecessary modes. However, to make fast mode decision this method needed transform coefficients along with several predefined thresholds.

Bu et al. [53] used current MB SAD and adaptive thresholds to select inter prediction mode. If SAD is less than the threshold then then large block sizes are selected otherwise small block sizes. The threshold is an adaptive function of the surrounding MBs mode information. Kuo et al. [54] tried to speedup the inter mode selection process. In their work, they used an adaptive rate-distortion model and multi-resolution motion estimation technique to achieve the desired task. Feng et al. [55] minimized the candidate modes for inter prediction using the characteristics of the motion compensation of residual image. Two most probable candidate modes are then selected from the shortlisted set of candidate modes, on the basis of the modes of top and left MB. Martinez-Enriquez et al. [56] presented a content adaptive scheme based on RDcost statistics. The main aim of this technique is to curtailed total number of evaluated inter prediction modes using adaptive thresholds. These thresholds are functions of differences between the RDcost for each mode. Lee et al.[32] exploited MB activity to shortlist the inter prediction modes. The MB activity is a function of global and local residual complexities (GRC, LRC). Although SAD based techniques are generally efficient and promising, the main challenge is to define accurate thresholds as their inaccuracy may degrade the quality significantly.

In the second approach, the algorithms exploit the spatial homogeneity measures including texture, edge information or variance of an MB to shortlist the candidate modes for RDO calculation. The basic assumption of these techniques is that homogeneous texture regions exhibit similar motion while the complex textured regions tend to have disordered motion. Zhu et al. [57] applied Soble operator to obtain the edge information of an MB in a down sampled image. The edge information is exploited to select the inter prediction modes for RDO process. Though these schemes give satisfactory results in some cases but may not work in case of textured objects with smooth motion.

The third approach is based on exploiting the temporal homogeneity. The main idea is to find out mean absolute frame difference (MAFD), i.e. difference between current and previous frame, or mean absolute difference (MAD) of current MB, i.e. difference between current MB and its collocated MB in former frame. Both of these differences are used to calculate the temporal homogeneity. This temporal homogeneity serves as a base to shortlist the candidate modes for RDO calculation. Jing et al. [58] skipped unnecessary modes for homogeneous MBs. In order to investigate whether the current MB is a part of homogeneous areas of the frame or not, MAD and MAFD are used. Such kind of techniques are suitable for video sequences with static background, while it cannot make sure that non-essential prediction modes could be neglected for background regions with smooth motion in the video frames caught by a moving camera.

Many techniques exploit both temporal and spatial homogeneity of MB for inter prediction mode selection, i.e. merging the above mentioned second and third approaches. Zhou [59] and Wu et al. [60] presented inter prediction mode selection technique based on temporal stationary and spatial homogeneity. The spatial homogeneity is calculated using Sobel operator while the MAD of current gives the temporal stationary. Wu et al. [61] used the spatial homogeneity based on edge information and temporal stationary based on MB differencing to shortlist prediction modes for RDO calculation. Bharanitharan et al. [62] exploited spatial and temporal homogeneity to present a classified region method. The candidate modes for inter prediction are reduced by exploiting the block homogeneity in both temporal and spatial domain. The 16×16 and 8×8 block patterns are used to calculate MB homogeneity.

The fourth approach is based on motion homogeneity of an MB or motion activity in temporally and spatially adjacent MBs. According to this category, if an MB is motion homogeneous or its motion activity is low then large block sizes are selected otherwise small block sizes. Zeng et al.[63] presented inter mode selection technique based on motion activity. The algorithm selects the candidate modes by analyzing the status of motion activity in temporally and spatially adjacent MBs. Liu et al.[64] made use of motion homogeneity to decide inter prediction mode. In this technique, motion homogeneity is analyzed on a normalized motion vector field. This field is obtained by the motion vectors resulting from motion estimation of 4×4 blocks. To choose the candidate inter prediction modes, three directional motion homogeneity is exploited. These schemes may be more effective if residual complexity is also incorporated along with motion homogeneity.

3.1.2 Intra Prediction Mode Selection Techniques

The existing techniques for intra prediction mode selection can be generally divided into two different approaches.

The first approach made use of local edge direction of the block to shortlist the candidate modes that go for RDO calculation. The basic difference between such kinds of techniques is the method adopted to calculate the local edge direction of block. Pan et al. [65] proposed a dominant edge direction (DED) based intra mode decision technique. This technique used the approximation of discrete cosine transform to measure the DED of a given block that helped to shortlist the candidate modes. Kim et al. [66] presented an efficient intra prediction mode decision technique based on local edge direction and neighboring MBs mode information. In this technique, directional masks are used to calculate local edge direction. Pan et al. [67] made use of local edge direction to select intra prediction mode. The edge direction and its amplitude against each pixel are found with the help of Soble operator. Based on the distribution of the edge direction histogram most probable modes are selected for RDO process. This algorithm may be unsuitable for embedded systems applications because calculation of edge direction involves float addition and number of division operations that increase its computational complexity.

Wang et al. [68] presented a technique in which dominant edge strength (DES) for each block is calculated then based on it a subset of modes are determined for RDO calculation. Su et al. [69] presented a scheme based on integer transform and adaptive threshold for intra mode decision. In this technique, integer transform is performed on original image to find out the local edge direction and adaptive threshold balance the compression and computational complexity. Initially, based on edge direction, RDO is performed for curtail intra prediction modes. The RDO calculation is terminated if the minimum mean absolute error (MMAE) of the reconstructed block belonging to the most suitable prediction mode is lesser than an adaptive threshold which is a function of quantization parameter (QP). Otherwise, more possible modes needed to be tested. Wei et al. [70] and Li et al. [71] applied non-normalized Haar transform (NHT) to extract the sub-block edges. The sub-block edge direction is used to shortlist the candidates modes that goes for RDO calculation. However, this approach increased the complexity because of pre calculations needed edge calculation.

Wang J.C. et al. [72] suggested a strategy to decide intra mode using edge histogram descriptors. The edge histogram descriptor is used to determine the DES, and only a few modes are chosen for RDO process. Elyousfi et al. [73] exploited the directional information and similarity between dominating direction of a smaller and bigger block. For luma component, RDost is computed for four modes. After this RDcost along with similarity with adjacent block mode, selection of best 4×4 modes is done. For 16×16 and 8×8 luma component the most probable modes are shortlisted by exploiting the fact of similar dominating direction of both smaller and bigger block. For chroma component only DC mode is used. Byeongdu et al. [74] presented a dominant edge direction (DED) based intra mode decision method. The DED is computed through pixel value subtraction and summation in the vertical and horizontal directions. Based on DED, RDO is performed for three most likely modes only. Li et al. [75] used the directional difference of each mode for intra prediction mode selection. For each directional mode, the average SAD between two adjacent actual pixels is defined as the direction difference. Based on the directional difference several most probable modes evaluated instead of using the full search. Elyousfi [35] used vector of the blocks gravity center to decide intra mode selection. The direction of the blocks gravity center vector is used to shortlist the candidate intra prediction modes. These selected candidate modes undergo RDO calculation. Bharanitharan et al. [76] exploited the directional difference between two neighboring pixels, which correspond to four major texture directions including vertical, horizontal, diagonal-down-right and diagonal-down-left to skip the unlikely intra prediction modes. However, performance of such kind of techniques is highly correlated with local edge direction calculation methods. However, performance of such kind of techniques is highly correlated with local edge direction calculation methods. These techniques may reduce the computational complexity at the cost of decrease in PSNR and increase in bit-rate.

In the second approach, the algorithms define certain criteria to shortlist the most probable intra prediction modes for RDO process. Meng and Au [77] presented a simplified cost function for mode selection. In this technique, a down-sampled pixels instead of a 4×4 blocks are used for partial cost calculation (to simplify the cost function). The mode selection process is performed using simplified cost function. Chen et al. [78] proposed three conditions for selecting candidate prediction modes for 4×4 luma blocks through skipping the fewer probable modes. These three conditions are based on the correlation of neighboring blocks prediction modes. Kim et al. [79] presented intra prediction mode selection scheme based on transform domain and spatial features. The sum of absolute differences (SAD) and Sum of absolute transform difference (SATD) have been used to shortlist the most of the candidate modes. For optimum mode selection 2-3 candidate modes undergoes for RDO calculation. Yu et al.[80] used the knowledge of the frequency spectrum to acquire the most likely modes for 4×4 luma component. The RDO is performed for these selected candidate modes. The fast inter mode selection technique identify low detailed macroblocks based on complexity measure that required less processing power. However, performance of such kind of techniques is depending upon the accuracy of pre-defined criteria. These techniques may reduce the computational complexity at the cost of degradation in visual quality.

Most of the existing techniques related to macroblock prediction targets either intra or inter mode selection. These techniques either simplify the prediction cost function or curtail the set of candidate modes of intra and inter prediction using local features (DED, DES, SAD, residual complexity, motion homogeneity and etc.). Some of these techniques required an additional computation to determine the block features used to shortlist the candidate prediction modes, while others minimize the computational complexity at the cost of quality degradation. The quality degradation is mainly due to the application of the subset of candidate modes. Unfortunately, a comprehensive framework for macroblock prediction that not only targets selection of optimum intra and inter prediction mode but also incorporates optimum prediction type (intra or inter) and macroblock partitions suitable for better intra prediction is missing in the existing literature.

3.2 Motion Compensation

In predictive coding, motion compensation (MC) is performed to temporally predict a macroblock. In H.264/AVC, MC module consumes about one-fourth of the decoding time and also takes significant part of encoding time [36]. It comprises two sub-modules i.e. data manipulation and interpolation. Both of these modules contribute significantly in increasing the overall computational complexity of motion compensation, and therefore need to be optimized.

3.2.1 Reasons Behind the Computational Complexity of Motion Compensation Algorithm

The reasons of higher computational cost of the JM reference implementation [81] of motion compensation algorithm are:

- To avoid illegal memory access in case of unrestricted motion vectors, clipping operation is performed that increases computational complexity of MC algorithm [82].

- The MC algorithm performs inter prediction on 4×4 blocks for each macroblock (16×16) without taking into account a macroblock and sub-macroblock partitioning. This adversely affects the performance due to looping overhead, repeated reading of motion vectors, reference frame indices and prediction list utilization flags.

- In reference algorithm, same pixel data is repeatedly accessed from memory in order to perform linear interpolation. Six pixel values are needed to compute the half pixel values. For the calculation of two adjacent pixel values, five out of six pixels are the same and can be reused. However, the reference algorithm instead of reusing these pixel values repeatedly read them from memory. Such repeated access of memory increases computational load on the processors.

- The algorithm sequentially computes the predicted samples without exploiting parallelism. This causes repeated memory access and delay in pixel value calculation.

The first two reasons belong to data manipulation sub module and rest of two falls in the area of data interpolation.

3.2.2 Existing Complexity Reduction Techniques

In the current available literature, several studies target the computational complexity of motion compensation algorithm. We may divide these techniques into two categories. The first one encompasses the techniques based on hardware implementations while the second category consists of implementations which are software based.

Wang et al. [83] proposed VLSI design of motion compensation to reduce the time taken by motion vector prediction module. This reduction is done by applying context switch buffers and 4×4 block based parallelism. Feng et al. [84] suggested a motion compensation technique for MPEG-4 video coding. In this work a local memory storage scheme implemented on CMOS technology is used to avoid storage of differential data of motion information in the video packet layer. It results in a reduction of 33% of total memory access bandwidth. A design for building a prototype for motion compensation algorithm is presented by Azevedo et al. [85]. Luma and chroma components processing is done in parallel traversing independent data paths. A prototype has been synthesized in FPGA and standard cell technologies.

The second category belongs to software based implementations. It may be further categorized into two sub categories.

The techniques belong to first sub category only target complexity reduction of the data interpolation module of motion compensation and make use of SIMD instructions to achieve this goal. Lee et al. [86] implemented H.264/AVC decoder using Intel Pentium 4 processor with MMX extensions. H.264/AVC high profile decoder implementation was done on Intel Pentium 4 processor by Bhatia [87]. It used optimization techniques based on SIMD and Windows based multithreading. Sheshadri [88] proposed a method to implement H.264/AVC decoder for ARM9TDMI processor. In these implementations, SIMD instructions are exploited in order to calculate half and quarter pixel values for linear interpolation. The use of SIMD instructions for interpolation speedup the motion compensation process. However, these techniques does not take any step toward complexity reduction of the data manipulation module of motion compensation.

The techniques fall in second sub category target computational complexity of both data manipulation and interpolation modules. Khan et al. [82] proposed an optimized algorithm for motion compensation and evaluated on Trimedia TM-1300

DSP processor. In this algorithm, motion compensation process is performed in three steps. In first step the edge creation around the image is performed. In the second step the half pixel values for the entire image are calculated. Then, in the last step the SIMD instructions are used to calculate the predicted sample values. In this technique, edge creation step reduced the clipping overhead which is part of the data manipulation module and rest of two steps simplified interpolation process. However, this technique calculate half pixel values for the entire image with out noticing either these pixel values are required or not. As in case of full pel, there is no need to calculate half pixel values. Moreover, data manipulation module can be further optimized by reducing loop overhead and repeated access of memory.

In short, the literature available in this area consists of techniques which either address the computational complexity issues related to data manipulation module or data interpolation module. Only a few techniques cover both the aspects simultaneously and even these need further improvement.

3.3 Exploiting Parallelism

In the literature, numerous schemes for implementing H.264/AVC encoders on different platforms are proposed. These techniques exploit the encoder parallelism at different levels like Group-Of-Picture (GOP)-level, frame-level, slice-level and macroblock-level to improve the encoder efficiency.

Sankaraiah et al. [89] proposed parallelization method based on GOP for H.264 encoder. In this scheme, each GOP is encoded independently in separate thread and the frames being referenced are included in the GOP. However, this method required a large storage memory for all frames and therefore this technique maps well to multi-core architectures that fall into generalpurpose processors (GPP) categories. In addition, GOP-level parallelism causes a very high latency that cannot be affordable in real-time applications. Therefore, this technique is not a best choice for multi-core platforms in which the memory is shared by all the processors because of cache pollution. Rodriguez et al. [90] exploited the frame-level parallelism along with a group of frames for H.264 encoding in a clustered workstations using Message Passing Interface (MPI). Although real-time encoding may be achieved with this technique, but the latency is very high.

Chen et al. [91] targeted Intel Pentium 4 processors H.264 codec in order to optimize its performance. It used SIMD instructions and multi-threading technology to exploit parallelism. It facilitated to run different multimedia applications on personal computers with good performance. It worked well when encoded sequence carries more non referenced B frames. Shuwei et al. [92] proposed macroblock region partition (MBRP) based parallel scheme. In this scheme, several macroblock regions of video frames are constructed, where each region comprises many adjoining columns of macroblocks. The encoding of these regions can be performed by using either single processor or a multi-processor system. However, this scheme is not a scalable when number of available processors exceeded more than half of the MB columns of frame. The results showed that real-time encoding was not achieved. Y. K. Chen et al. [93] suggested the parallel algorithms based on Intel Hyper-Threading architecture in which video frame is divided into several slices and multiple threads are used to process these slices. However, the drawback of this technique is the additional overheads on bit-rates due to additional information for each slice. The scalability of this scheme is dependent on number of slices.

Z. Zhao and P. Liang [94] used wave-front technique to achieve parallelism in H.264 baseline encoder. The different frames or MB rows are mapped on to different processors which speed it up around 3 times on software simulator with 4 processors. Zrida et al. [95] used task and data level parallelism of H.264 encoder for embedded SoCs. The Kahn process network (KPN) model and the run time YAPI programming C++ library serve as a base to accomplish this task. Wen et al. [96] proposed macroblock and block level parallelism for inter-frame and intra-frame prediction, respectively along with generation of parallel bit-stream method at macroblock level for CAVLC. It improved the performance of these modules of H.264 encoder to 30KB. For Cell processor, Alvanos et al. [97] divides the MB encoding process in three phases including analyzes and encode, entropy encoding and de-blocking. After that each phase is executed as a separate task. In order to resolve the dependency, they made use of 2D-wavefront parallelism technique [98] in which all macroblocks are issued in an anti diagonal based manner and wait before issuing the next anti-diagonal. However, the most of the time consuming tasks analyze and encode are assigned to the main Synergistic Processing Element (SPE). Therefore, the scheme cannot be implemented for real-time processing on low-power CPUs.

Park and Ha [99] macroblock level parallelization scheme for the cell processor. They offloaded the macroblock analysis module which is the first phase of the encoder to Synergistic Processors Elements (SPEs). However, all other parts of encoder remain on the Power Processor Element (PPE). They estimated the expected performance and discovered the over head factors. Di Wu et al. [100] targeted on the kernal optimization instead of parallelization for the efficient implementation of H.264 encoder. However, they did not consider other auxiliary parts of the encoder including entropy encoding, rate control and frame initialization. For real-time HD video coding, Xun et al. [101] exploited a decentralized pipelined parallel coding scheme using eight SPEs. The decentralized task creation decreased the task management overhead in this implementation. They used on-chip communication and multi-buffering to transfer data among different encoder modules for efficient communication. Lu and Hang [102] and Schwab et al. [103] presented an adaptive motion estimation (ME) techniques for GPUs. The main objective of these techniques is the reduction of computational complexity of ME. However, these techniques mostly exploited data-level parallelism and only target prediction-loop of the H.264/AVC encoder.

For scalable H.264 video coding, Cheung et al. [104] and Azevedo et al. [105] proposed the application of scheduling techniques at the MB level for embedded multicore devices. In these techniques a dedicated hardware is used for task scheduling. However, the MB level implementation of the scheduling control makes this model rather expensive for CPU+GPU platforms. Chen and Hang [106] presented parallel implementations of the H.264/AVC encoder on CPU + GPU systems. This approach completely offloaded the ME task (including the interpolation and subpixel ME) to the GPU, keeping the rest of the encoder modules to be executed on the CPU. However, this approach did not consider the possibility of concurrent and asynchronous implementation of the video coding algorithm on both the CPU and the GPU devices, which could potentially decrease the overall processing time. Momcilovic et al. [107] proposed parallel dynamic model for hybrid GPU+CPU systems. In this model entire inter prediction loop of the encoder is parallelized on both the CPU and the GPU. To dynamically distribute the computational load among CPU and GPU, a computationally efficient model is also described. The suggested model consist of load balancing and dependency aware task scheduling algorithms. Momcilovic et al. [37] in contrast to [107] presented a complete parallelization strategy comprising all inter frame processing modules of the H.264/AVC encoder along with an improved dynamic dependency aware scheduling algorithm for collaborative video encoding on hybrid GPU +CPU platforms.

In short, most of these schemes lack in describing the hybrid implementation of H.264/AVC encoder in which some blocks of encoder are implemented in hardware and the rest in software. As such implementations require efficient encoder tasks distribution, scheduling and synchronization among hardware and software blocks. It also demands exploitation of encoder parallelism to efficiently utilize the processing cores. So, there is a need to work in this direction to highlight the issues involved in such an implementation.

3.4 Summary

The problem of complexity reduction of macroblock prediction parameters selection process in H.264/AVC coding standard is not trivial. Different researchers have worked in this domain from different prospective. Some target intra mode selection while others present techniques for inter mode selection. Unfortunately, there is not a single methodology available that optimized the entire prediction parameters selection process. There is a need to develop a comprehensive method to select prediction parameters such as prediction type (intra or inter), prediction modes and suitable prediction block sizes etc. to perform better prediction.

The techniques available related to complexity reduction of motion compensation algorithm either address the computational complexity issue related to the data manipulation module or the data interpolation module. Only a few techniques cover both the aspects and can be further improved by reducing looping overhead, repeated access of memory and refining interpolation process.

Most of the existing implementation schemes for H.264/AVC encoder are software based and exploit the encoder parallelism at different levels like macroblock-level, slice-level, frame-level and GOP-level. These schemes lack in describing the hybrid hardware-software implementation of encoder in which some blocks of encoder are implemented in hardware and the rest in software. Hybrid implementation demands exploitation of parallelism at both coarse as well as fine-grain level to efficiently utilize the available processing resources. So, there is a need to work in this direction to improve the coding efficiency of hybrid encoders.

Chapter 4

AN EFFICIENT FRAMEWORK FOR MACROBLOCK PREDICTION

4.1 Introduction

This chapter presented an efficient framework for macroblcok (MB) prediction to reduce the computational complexity of H.264/AVC encoder. It incorporates several innovative techniques to decide macroblock prediction type (intra or inter), skip macroblock early detection, appropriate block size selection for intra prediction, directional mode selection for intra prediction and inter mode (block sizes) selection. This framework exploits temporal and spatial statistics of video sequence, coding modes information of previously encoded spatial and temporal neighboring MBs, motion-field statistics and QP-based thresholds to exclude as many intra and inter prediction modes as possible prior to the RDO process. The results based on experiments indicate that the suggested framework can significantly reduces the computational complexity of prediction process with marginal loss in visual quality.

The rest of the chapter is arranged as described in the forth coming lines. Observation and analysis is made in Section 4.2. Section 4.3 presents the proposed framework. Section 4.4 describes the experimental analysis and discussion. Finally, the summary is presented in Section 4.5.

4.2 Observation and Analysis

Generally, based on the spatial and temporal statistics, the regions of the natural video sequences can be categorized as darker, brighter, smooth, rigid or textured, motionless, slow-motion, high-motion, homogeneous-motion, complex-motion and many combination of prior mentioned categories. Extensive experiments are performed on different video sequences using exhaustive parameters selection technique of the H.264/AVC reference software to obtain data for statistical analysis of prediction parameters (modes and block sizes). The test conditions are set as follows: MV search range is 32 pels, entropy coding is set to CABAC, RDO and fast motion estimation are enabled in encoder main profile, MV resolution is 1/4-pel, number of reference frame is set to 1, and 300 frames are encoded. To compute the probability of coding parameters, encoding results at five different QPs including 24, 28, 32, 36 and 40 are used. Table 4.1 lists the averaged probability of selecting each prediction type and mode when each test video sequence is encoded with IPPPPP structure.

TABLE 4.1: Probability (%) of Intra and Inter Prediction Modes Selection

Sequences	Format	SKIP	16×16	16×8	8×16	8×8p	I4×4	I16×16
Coastguard		36.52	29.48	7.24	7.25	19.26	0.06	0.19
Claire		84.18	6.98	2.36	2.17	4.23	0.01	0.07
Container	QCIF	83.34	7.32	2.75	1.82	4.6	0.04	0.13
Foreman	(176×144)	39.62	25.23	8.73	9.75	15.96	0.4	0.13
Highway		67.77	15.84	5.78	3.61	6.87	0.03	0.1
Akiyo		88.46	5.12	1.8	2.09	2.53	0	0
Mobile		28.75	25.34	7.08	7.15	31.58	0.04	0.06
MaD	CIF	77	12.39	3.6	3.92	2.86	0.14	0.09
Silent	(352×288)	75.56	9.78	3.37	4.28	5.8	0.91	0.3
Tempete		32.44	25.4	8.79	7.64	23.74	1.37	0.62
Flower		35.94	23.84	11.32	6.94	21.32	0.38	0.26
Football		39.75	20.34	8.53	9.91	11.54	6.39	3.54
Intros	NTSC	75.25	11.27	4.03	3.85	2.56	1.78	1.26
Mobile	(720×480)	27.28	23.67	9.98	10.01	28.64	0.15	0.27
Vtc1nw		94.54	3.05	0.91	0.79	0.7	0	0.01
Parkrun		34	23.06	7.04	7.59	28.06	0.11	0.14
Shield	720p	60.85	19.14	4.98	4.94	9.1	0.34	0.65
Stockholm	(1280×720)	62.43	16.94	5.64	5.12	9.14	0.22	0.51
Average		57.98	16.9	5.77	5.49	12.69	0.69	0.48

Table 4.1 illustrates that for 98.83% MBs inter prediction type is selected and intra prediction type is selected for the remaining 1.17% of MBs. Moreover, 86.14% MBs select larger block sizes i.e. 16×16, 16×8 and 8×16 for inter prediction. Furthermore, the percentage of selecting inter 16×16 mode is about 74.88%. The SKIP mode in fact strongly prevails (57.98%) among all the possible modes, particularly for those sequences comprising slow-motion content or motionless i.e. *Claire*, *Akiyo*, *Vtc1nw* and etc. It is also observed that for inter prediction larger block sizes are selected within the objects either having low texture or high texture. On the other hand, smaller block sizes including 8×8, 8×4, 4×8 and 4×4 are mostly selected at objects boundaries or in case of irregular motion. It is also observed that percentage of selecting larger block sizes for both intra and inter prediction is increased with increase in QP and vice versa.

Table 4.2 enlists the averaged probability of selecting macroblock partition for intra prediction when all the frames in test video sequences are encoded as intra. It depicts that 65.56% MBs are encoded as intra 4×4 and rest of 34.44% MBs are encoded as intra 16×16. The probability of selecting intra 4×4 is high for video sequences having high textured scene including *Foreman*, *Mobile*, *Tempet* and *Parkrun*. The probability of selecting intra 16×16 is high for video sequences having homogeneous regions i.e. *Intros*, *Vtc1nw*, *Akiyo* and *Claire*.

This statistical analysis clearly indicates that the optimum selection of prediction parameters is highly correlated with spatial and temporal statistics of the video contents. In case of motionless or slow-motion scenes the SKIP mode is expected to be the optimal choice. On the other hand, probability of selecting intra prediction type is high for MBs belong to low motion regions with low texture. For inter prediction, larger block sizes i.e 16×16, 16×8, 8×16 are selected within the objects having homogenous motion. On the contrary, smaller block sizes including 8×8, 8×4, 4×8 and 4×4, are mostly selected at objects boundaries or in case of irregular motion. For intra prediction, intra 4×4 is highly suitable for detailed regions MB

Table 4.2: Probability (%) of Intra Prediction Block Size Selection

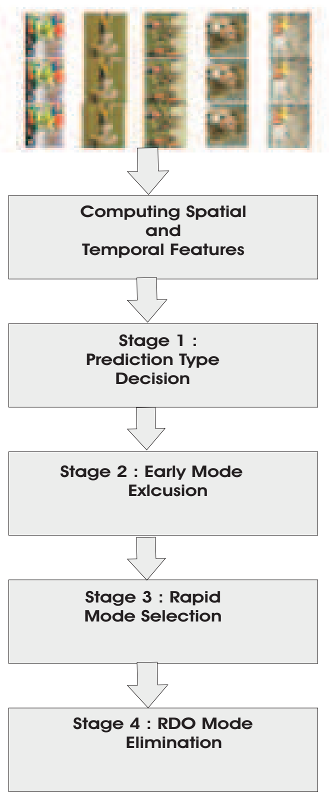
Sequences	Format	I4×4	I16×16
Coastguard	QCIF (176×144)	76.83	23.17
Claire		46.32	53.68
Container		60.79	39.21
Foreman		83.43	16.57
Highway		66.04	33.96
Akiyo	CIF (352×288)	45.87	54.13
Mobile		93.95	6.05
MaD		54.36	45.64
Silent		73.42	26.58
Tempete		84.49	15.51
Flower	NTSC (720×480)	74.39	25.61
Football		60.16	39.84
Intros		34.86	65.14
Mobile		83.89	16.11
Vtc1nw		31.11	68.89
Parkrun	720p (1280×720)	80.3	19.7
Shield		68.26	31.74
Stockholm		61.55	38.45
Average		65.56	34.44

and intra 16×16 is appropriate for an MB belonging to smooth regions of image. Moreover, the coding modes of a macroblock are also highly correlated with the prediction parameters of its spatial and temporal neighboring macroblocks.

From the above analysis, it can be concluded that most of the prediction parameters of a video frame can in fact be anticipated using temporal and spatial statistics of the present and former video frames.

4.3 Proposed Framework

The proposed framework consists of four stages: (1) Prediction type decision, (2) Early mode exclusion, (3) Rapid mode selection, and (4) RDO mode elimination as shown in Fig. 4.1. These stages comprise several algorithms to decide macroblock prediction type (I-MB or P-MB), SKIP mode early detection, appropriate intra-prediction block size selection, directional mode selection for intra-prediction and inter-prediction mode (block sizes) selection. These algorithms exploit spatial and temporal features of video frames to accomplish the desired tasks. The following sections describe the spatial and temporal features used in this work and each stage of the proposed framework.



Computing Spatial and Temporal Features
Stage 1 : Prediction Type Decision
Stage 2 : Early Mode Exlcusion
Stage 3 : Rapid Mode Selection
Stage 4 : RDO Mode Elimination

FIGURE 4.1: Framework for Macroblock Prediction

4.3.1 Features Selection

The analytical study made in section 4.2 indicates that the spatial and temporal features of the video sequences are adequate to differentiate an MB thus to foretell a probable prediction type and mode. The selection of particular features can be made taking into consideration the tradeoff between computational cost and the given accuracy in the early mode prediction. In this work, following features are selected to predict the macroblock prediction type and mode.

Average Brightness gives the information about the brightness of an MB and is utilized to classify an MB as bright or dark. It is the mean of luminance component values X (i,j) of an MB.

$$\mu_{m,n} = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} X(i,j) \tag{4.1}$$

Variance provides the information about statistical dispersion, thus it is used as measurement of texture or descriptor of smoothness. For MBs belong to smooth/flat areas all the samples have same brightness and their variance is zero. The variance of each MB is roughly estimated as

$$\sigma_{m,n} = \sum_{i=1}^{M}\sum_{j=1}^{N} |X(i,j) - \mu_{m,n}| \tag{4.2}$$

Zero SAD (Z_SAD) is the sum of absolute difference between current MB and its collocated MB in the previous frame in display order. It gives the information about degree of motion or stillness or change between two frames/MBs. The Zero SAD is calculated as

$$Z\_SAD = \sum_{i=1}^{M}\sum_{j=1}^{N} |X(i,j) - Y(i,j)| \tag{4.3}$$

Where X(i,j) indicates the current MB and Y(i,j) is its collocated MB in previous frame.

MB residual complexity (MB_RC) is the SAD of the current MB calculated through light weight recursive motion estimator in which the vector field tends towards true object motion. The motion estimation on 8×8 block size is performed in order to calculate MB_RC of an MB. For performing motion estimation, 3-D Recursive Search (3-D RS) motion estimator [108] is used. If an MB is located at the mth row and nth column of the video frame and $SAD_{i,j}$ be the SAD of its corresponding 8×8 blocks then its $MB\_RC_{m,n}$ is calculated as

$$MB\_RC_{m,n} = \frac{1}{4}\sum SAD_{i,j}, i \in [8m, 8m+1], j \in [8n, 8n+1] \tag{4.4}$$

Motion-Field Statistics are acquired through motion characteristics (SAD) of the spatial (in the current frame Ft) and temporal (in the previous frame Ft-1) neighboring MBs as follows:

$$SAD_{Spatial} = \frac{1}{4}(SAD_{TL} + SAD_{T} + SAD_{TR} + SAD_{L}) \tag{4.5}$$

$$\begin{aligned} SAD_{Temporal} = \frac{1}{9}(&SAD_{TL} + SAD_{T} + SAD_{TR} + SAD_{L} \\ &+SAD_{Collocated} + SAD_{R} + SAD_{DL} \\ &+SAD_{D} + SAD_{DR}) \end{aligned} \tag{4.6}$$

Coding-Mode-Field Statistics are attained by the use of coding mode information of the temporal and spatial neighboring MBs encoded as an intra MB i.e I-MB.

$$I\_MB_{Spatial} = isI(MB_{TL}) + isI(MB_{T}) + isI(MB_{TR}) + isI(MB_{L}) \tag{4.7}$$

$$\begin{aligned} I\_MB_{Temporal} &= isI(MB_{TL}) + isI(MB_{T}) + isI(MB_{TR}) \\ &+ isI(MB_{L}) + isI(MB_{Collocated}) + isI(MB_{R}) \\ &+ isI(MB_{DL}) + isI(MB_{D}) \\ &+ isI(MB_{DR}) \end{aligned} \tag{4.8}$$

Fig.4.2 gives the conception of temporal and spatial neighboring macroblocks of current MB. In this work, a down size image is used in order to calculate average brightness, variance and Zero SAD of an MB. The downsizing factor of an image is selected as four. In scaled down image, 4×4 block size of an actual image is represented by one pixel and 16×16 block size is mapped to 4×4 block size. The use of scaled down image for feature extraction helps to reduce the computational complexity.

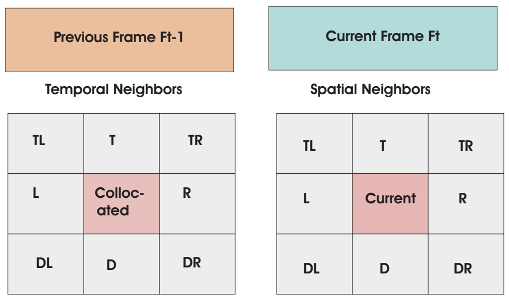


FIGURE 4.2: Spatial and Temporal neighboring Macroblocks of Current MB

4.3.2 Prediction Type Decision

At this stage, decision of macroblock prediction type (intra predicted macroblock (I_MB) or inter predicted macroblock (P_MB)) is performed with the help of spatial and temporal features of the video sequence. The analysis made in section 4.2, clearly indicates that the P_MB prediction type indeed dominates in inter frame encoding particularly for those sequences encompassing slow motion or homogeneous motion or motionless content. On the other hand, the probability of selecting I_MB prediction type is high in case of random motion. This observation implies that the prediction type (I_MB or P_MB) decision should be performed at the start of the prediction parameter selection process for each MB so that an early prediction type decision effectively decreases the computational complexity.

We formulate macroblock prediction type decision problem into classification problem and propose a learning-based solution in which each MB is classified into one of the three below listed classes:

- Class 1: MB is encoded as intra I_MB.
- Class 2: MB is encoded as inter P_MB
- Class 3: MB can be encoded as intra I_MB or inter P_MB. The decision of MB prediction type for this class is made at the later stages of the framework.

The consideration of Class 3 helps to reduced the margin of the error. To decide the prediction type, Adaboost classifier is trained using spatial and temporal features including Average Brightness, Variance, Z_SAD, $SAD_{Spatial}$, $SAD_{Temporal}$, $I\_MB_{Spatial}$ and $I\_MB_{Temporal}$. The training data is obtained from variety of video sequences. Then, the aforementioned features and the appropriate prediction type determined by the RDO are taken as the training set for Adaboost classifier.

The class of each $MB_{m,n}$ with feature vector F_V$_{m,n} = \{\mu, \sigma, Z\_SAD, MB\_RC$, SAD$_{Spatial}$, $SAD_{Temporal}$, $I\_MB_{Spatial}$, $I\_MB_{Temporal}\}$ based on class conditional probabilities is decided as

- If P1 <P2 and P1 $>\tau$, then Class 1 is assigned
- If P2 >P1 and P2 $>\tau$, then Class 2 is selected
- Otherwise, Class 3 is assigned.

Where P1 and P2 are the probabilities of each macroblock $MB_{m,n}$ which belongs to Class 1 and Class 2, respectively. The value of τ is set to 0.6 through experiments. As a result of this classification, for most of the MBs probable prediction type can be selected that significantly reduced the computational cost of RDO process.

4.3.3 Early Mode Exclusion

At this stage of the framework, early detection of SKIP mode of a macroblock is performed for inter prediction when certain conditions are satisfied. Moreover, appropriate block size selection for intra prediction of an MB is made based on spatial features. These two decisions exclude the highly unlikely modes for both I_MB and P_MB prediction types. The statistical analysis shows that for P_MB type, SKIP mode in fact strongly prevails among all the available modes, particularly for slow-motion or motionless content carrying sequences. This observation indicates that the early detection of SKIP mode should be made at the start of the inter mode decision process for each MB to effectively decrease the computational complexity. Similarly, for I_MB type, intra 4×4 is highly suitable for detailed regions MB and intra 16×16 is appropriate for an MB belonging to smooth regions of the image. In H.264/AVC, RDO is performed for both intra 4×4 and 16×16. However, the MB is either encoded as intra 4×4 or intra 16×16. This results in increasing the encoder computational complexity and overall encoding time. If it is possible to detect appropriate block size to predict an MB prior to RDO process then the computations for the other block size can be avoided.

4.3.3.1 SKIP Mode Early Detection Algorithm

In H.264/AVC JM reference encoder [81] the skip mode is assessed along with other possible coding modes by encoding the macroblock using each mode and selecting the mode which minimizes the RDcost. The objective of this research work is to minimize the computational cost through early identification of skip macroblock. The early detection of skip mode avoids the RD calculations for rest of the prediction modes. The procedure to perform skip mode early detection for each MB is given below.

- Perform motion estimation for 16×16 MB partition size keeping previous frame in display order as a reference frame for prediction.
- Compute the motion vector difference (MVD) relative to the predicted motion vector for skip mode.
- Calculate quantized coefficients after transformation of residual data.
- If MVD and all the quantized coefficients are zero then skip mode is selected for an MB.

The conditions used in this work for skip early detection are same as that of used in reference JM encoder contributed by Jeon et al.[109].

4.3.3.2 Intra Prediction Block Size Selection Schemes

Generally, if an MB is a part of smooth regions, including low textured areas or objects then intra 16×16 is appropriate to select. In these cases the selection of intra 16×16 causes significant decrease in prediction error. On the other hand, intra 16×16 results in large residual error for an MB belonging to the regions with high textures. In such cases intra 4×4 is the appropriate mode as it can predict an MB more accurately. Therefore, it can be concluded that if low textured MBs can be identified, then time consuming RD cost calculation can be avoided for most of the MBs, as 4×4 intra prediction modes can be skipped. Similarly, detection of high textured MBs can help to skip RDO calculations for intra 16×16 modes. In this work two schemes are proposed to select appropriate block size for intra prediction.

Scheme 1:

This scheme exploits the variance of an MB to select appropriate block size from 4×4 and 16×16 for intra prediction. For each $MB_{m,n}$ with variance $\sigma_{m,n}$, the selection of block sizes for intra prediction is done as follows:

- If $\sigma_{m,n} < \tau_1$, Only intra 16×16 is selected for prediction.
- If $\sigma_{m,n} > \tau_2$, Only intra 4×4 is selected for prediction.
- Otherwise, Select both 16×16 and 4×4 block sizes for intra prediction.

In order to model thresholds τ_1 and τ_2 as a function of QP, a lot of experiments are performed. For this purpose, ten QPs 12, 16, 20, 24, 28, 32, 36, 40, 44, and 48 are used. After plotting the thresholds τ_1 and τ_2 against each QP listed in Table 4.3 onto Fig.4.3, it is concluded that these thresholds are estimated as the exponential function $f(QP) = p \times e^{(q \times QP)}$. Accordingly, the τ_1 and τ_2 values in Table 4.3 are determined as follows:

TABLE 4.3: Thresholds for Qps

Qp	τ_1	τ_2
12	26.6	141.0
16	27.5	140.25
20	28.5	139.5
24	29.5	138.0
28	30.5	137.25
32	31.5	136.50
36	32.5	135.50
40	33.75	134.75
44	34.5	134.0
48	36.0	133.0

$$\tau_1 = 22 \times e^{0.005 \times QP} \tag{4.9}$$

$$\tau_2 = 157 \times e^{-0.003 \times QP} \tag{4.10}$$

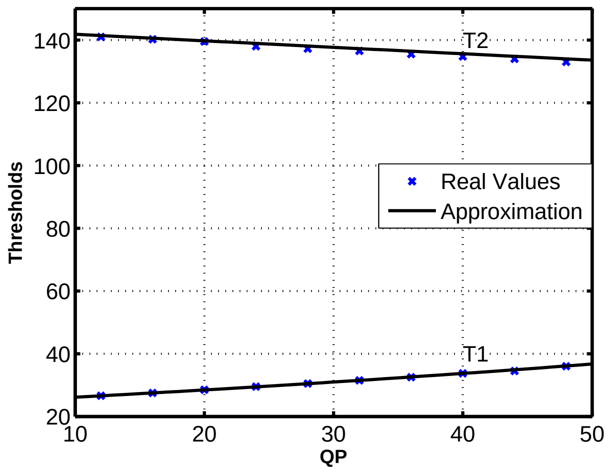


FIGURE 4.3: τ_1 and τ_2 Approximation

Scheme 2:

In this approach, selection of suitable block size for intra prediction is taken as a classification problem. To solve this problem AdaBoost classifier is trained using brightness and variance of an MB as features. The following are the three defined classes

- Class 1: Intra 4×4
- Class 2: Intra 16×16
- Class 3: Both 4×4 and 16×16 are candidate for intra prediction. The decision of block size for intra prediction is made at later stages

Each $MB_{m,n}$ with feature vector F_VC $= \{\mu, \sigma\}$ is classified as follows:

- If P1 <P2 and P1 $>\tau$, then Class 1 is assigned.

- If P2 >P1 and P2 >τ, then Class 2 is selected.
- Otherwise, Class 3 is assigned.

Where P1 and P2 is the probability of each macroblock $MB_{m,n}$ to belong to Class 1 and Class 2, respectively. The value of τ is set to 0.6 through experiments. The aforementioned features and the appropriate block size determined by the RDO are taken as the input training data for Adaboost classifier. The training samples are acquired from different video sequences.

4.3.4 Rapid Mode Selection

At this stage of the framework, further analysis is performed which provides a reduced set of candidate intra and inter prediction modes for RDO process. The following sections explain the schemes used to shortlist inter and intra prediction modes for RDO process.

4.3.4.1 Inter Prediction Mode Selection Algorithm

1. **Hypothesis**

   In motion estimation, inter prediction modes (block sizes) can decrease the residual or prediction error efficiently. Generally, if an MB is a part of regions with homogeneous motion, including zero motion of a static background, uniform motion of rigid objects and smooth motion of a moving background and then large block sizes (16×16, 16×8, 8×16) are appropriate to select. In these cases the selection of large block sizes causes significant decrease in prediction error. On the other hand, large block sizes result in large residual error for an MB belonging to the regions with high textures or complex motion. In such cases the smaller block sizes (8×8, 8×4, 4×8, and 4×4) are appropriate modes as they can capture complex motion accurately.

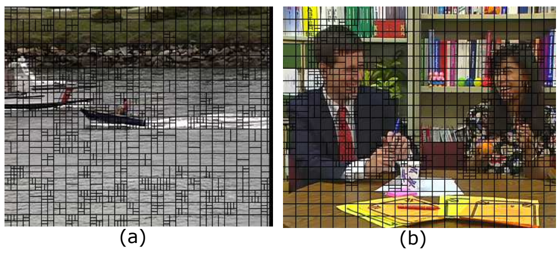


FIGURE 4.4: Selected Block Sizes using Full Inter Prediction Mode Selection.
(a) Coastguard (b) Paris

Fig.4.4 shows two example frames of CIF resolution in which the optimal inter prediction modes are represented by different sized boxes overlaid on the corresponding MB. These modes are selected using full search algorithm RDO. The 29th frame of Coastguard sequence is shown in Fig.4.4(a) in which camera panning activity results in causing illusion of background motion. The background region reflects a homogenous translation motion. So, most of the MBs in this region are coded as 16×16 e.g. shore and still water region. On the other hand, smaller block sizes are likely to be selected in the boundary regions between boats and water and the ripple regions etc. Fig.4.4(b) shows the 40th frame from the test video sequence Paris with static back ground. It clearly demonstrates that most of the MBs belonging to static background are coded using large block sizes. In the Fig.4.4(b) the smaller block sizes belong to the regions at the boundary between objects with different motion, such as the face, head and hand and clothes, in order to achieve smaller residual error. One observation in the above cases is that smaller block sizes are mostly selected at object boundaries or in case of irregular motion regions and larger block sizes are selected within objects both for low as well as high textures.

In order to perform statistical analysis on the probability of different inter prediction modes selected using full inter mode decision, a variety of CIF video sequences, with different motion activities, are encoded including

Akiyo, *Coastguard*, *Container*, *Foreman*, *Mother and Daughter (MaD)*, *Mobile*, *News*, *Paris*, *Silent* and *Tempete*. The test conditions are set as follows: MV search range is 32 pels, CABAC, RDO and fast motion estimation are enabled in encoder Main profile, MV resolution is 1/4-pel, number of reference is 1, and 300 frames are encoded with IPPP structure for each test video sequence. Table 4.4 lists the averaged probability of selecting each inter prediction mode. To compute each probability, encoding results at five QPs, including 24, 28, 32, 36 and 40, are used. Table 4.4 illustrates that the 86.8% MBs select larger block sizes i.e. 16×16, 16×8 and 8×16 for inter prediction. Moreover, the percentage of selecting inter 16×16 is about 76.3%. It is also observed that percentage of selecting larger block sizes is increased with increase in QP and vice versa.

TABLE 4.4: Probability (%) of Inter Prediction Modes Selection

Sequences	16x16	16x8	8x16	8x8p
Akiyo	93	2	2	3
Coastguard	64	10.3	9.4	16.3
Container	90.8	2.5	2.5	4.2
Foreman	72.1	8.5	8.5	10.9
Mother and Daughter	89	3.7	4.3	3
Mobile	54	7	7.3	31.7
News	86.5	2.9	3.5	7.1
Paris	81.2	3.2	3.4	12.2
Silent	84.7	3.7	4.6	7
Tempete	47.3	8.5	7.2	37
Average	76.3	5.2	5.3	13.2

Therefore, it can be concluded that if motion homogeneous MBs with low residual complexity can be identified, then time consuming RDcost calculation can be avoided for most of the MBs, as smaller inter prediction modes can be skipped. Optimal inter prediction mode for an MB is highly correlated with its motion homogeneity and residual complexity. If the MB is homogeneous with small residual complexity, it results in selection of large block size as the optimal mode. However, if the MB is non-homogeneous or its residual complexity is not small enough then the smaller block sizes are appropriate to be used as the optimal block size mode. This work exploits motion homogeneity and residual complexity together for an MB to shortlist the candidate inter modes for RDO calculation, a merger which is missing in the existing literature.

2. **MB Classification based on Motion Homogeneity and Residual Complexity**

   According to the above observations, optimal inter prediction mode is highly correlated with the motion homogeneity exhibited in MB and residual complexity or prediction error. This work exploits motion homogeneity and residual complexity for MB classification. The motion estimation on 8×8 block size is performed in order to calculate the motion homogeneity and residual complexity of an MB. For performing motion estimation, 3-D Recursive Search (3-D RS) motion estimator [108] is used because it is light weight and tends towards actual or true motion of objects. One disadvantage of 3-D RS motion estimator is its convergence time. Generally, it converges after one or two frames. So, the results of 3-D RS are used in this work after convergence has been achieved.

   The MB residual complexity (MB_RC) and motion homogeneity (MH) using corresponding blocks in an MB are calculated as follows. If an MB is located at the mth row and nth column, it is denoted as $MB_{m,n}$. The motion vectors MVs and SAD of its corresponding 8×8 blocks are represented as $MV_{i,j} = \{mvx_{i,j}, mvy_{i,j}\}, SAD_{i,j}, i \in [8m, 8m+1], j \in [8n, 8n+1]$. For each partition P_k which may be row R_k in Fig.4.5 (a) or column C_k in Fig.4.5(b), the mean deviation of MVs is defined as

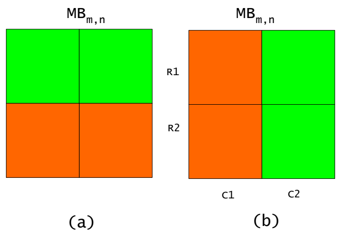


Figure 4.5: MB Partitions used to Calculate Motion Homogeneity and Residual Complexity

$$MD(P_k) = \frac{1}{2}\sum_{P_k}\{|mvx_{i,j} - \frac{1}{2}\sum_{P_k} mvx_{i,j}| + |mvy_{i,j} - \frac{1}{2}\sum_{P_k} mvy_{i,j}|\} \tag{4.11}$$

The vertical and horizontal motion homogeneity and residual complexity of an $MB_{m,n}$ can be calculated using Eq.4.12, Eq.4.13 and Eq.4.14, respectively:

$$H\_MH_{m,n} = \frac{1}{2}\sum_{k=1}^{2} MD(R_k) \tag{4.12}$$

$$V\_MH_{m,n} = \frac{1}{2}\sum_{k=1}^{2} MD(C_k) \tag{4.13}$$

$$MB\_RC_{m,n} = \frac{1}{4}\sum SAD_{i,j} \tag{4.14}$$

Based on the above MB directional motion homogeneity and residual complexity measures, each $MB_{m,n}$ is classified into one of the following five classes when the specified conditions hold true.

Class 1: Homogeneous motion with low residual complexity

$$MB\_RC_{m,n}<\tau_1 \ \& \ H\_MH_{m,n}<\nu_1 \ \& \ V\_MH_{m,n}<\nu_1 \tag{4.15}$$

Class 2: Complex motion (exhibits no obvious homogeneity) with high residual complexity

$$MB\_RC_{m,n}>\tau_2 \ \& \ H\_MH_{m,n}>\nu_2 \ \& \ V\_MH_{m,n}>\nu_2 \tag{4.16}$$

If an $MB_{m,n}$ does not belong to above two classes then it is further classified into one of the following three classes:

Class 3: Motion likely to be homogenous in vertical direction

$$H\_MH_{m,n}>V\_MH_{m,n} \tag{4.17}$$

Class 4: Motion likely to be homogenous in horizontal direction

$$H\_MH_{m,n}<V\_MH_{m,n} \tag{4.18}$$

Class 5: Motion likely to be homogenous in horizontal or/and vertical direction

$$H\_MH_{m,n}=V\_MH_{m,n} \tag{4.19}$$

The thresholds ν_1 and ν_2 for directional motion homogeneity are set to 0.5 and 1.0, respectively. The threshold ν_1 is set to 0.5 in order to cater for one out-lier MV in the motion homogeneous MB. On the other hand, threshold ν_2

was selected to 1.0 after extensive experiments.The thresholds τ_1 and τ_2 for residual complexity are selected based on frame residual complexity (FRC) and average MB residual complexity (AMB_RC) according to the Eq.4.20 and Eq.4.21, respectively.

$$\tau_1 = \frac{1}{FRC} \times W1 \times AMB\_RC \tag{4.20}$$

$$\tau_2 = \frac{1}{FRC} \times W2 \times AMB\_RC \tag{4.21}$$

where AMB_RC and FRC are defined as

$$AMB\_RC = \frac{1}{K} \sum_{i=1}^{K} SAD_i \tag{4.22}$$

$$FRC = \frac{1}{MN} \sum_{i=1}^{K} SAD_i \tag{4.23}$$

Where K indicates the number of MBs in frame and M, N are the horizontal and vertical dimensions of the frame.

In order to obtain weights $W1$ and $W2$ for each QP, a lot of experiments are performed. Then these values are modeled as a function of QP. For this purpose, ten QPs, 12, 16, 20, 24, 28, 32, 36, 40, 44, and 48 are used. After plotting the weights $W1$ and $W2$ against each QP listed in Table 4.5 onto Fig.4.6, it is concluded that these weights are estimated as the exponential function, $f(QP) = p \times e^{(q \times QP)}$. Accordingly, the $W1$ and $W2$ values in Table 4.5 are determined as follows:

Table 4.5: Weights for Qps

Qp / Weights	12	16	20	24	28	32	36	40	44	48
W1	1.12	1.16	1.20	1.24	1.28	1.32	1.36	1.40	1.44	1.48
W2	1.28	1.38	1.50	1.60	1.75	1.90	2.05	2.25	2.40	2.60

$$W1 = e^{0.0085 \times QP} \tag{4.24}$$

$$W2 = e^{0.02 \times QP} \tag{4.25}$$

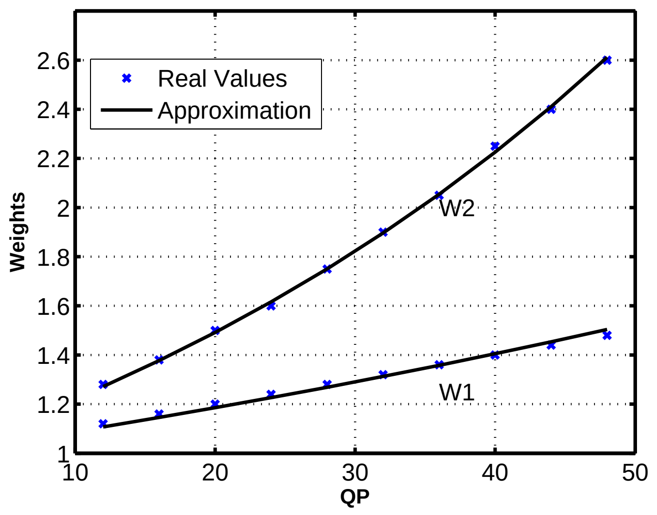


Figure 4.6: W1 and W2 Approximation

3. **Candidate Inter prediction mode selection**

   Based on the above classification, Table 4.6 summarizes the candidate inter prediction modes that goes for RDO. For example, if class 3 is selected, RDO process is performed only for 16×16 and 8×16.

Table 4.6: Candidate Inter Prediction Modes

MB Class	Candidate inter prediction modes
1	16×16
2	16×16, 16×8, 8×16, 8×8p
3	16×16, 8×16
4	16×16, 16×8
5	16×16, 16×8, 8×16

Except for class 2, for which each 8×8 block in an MB will be further examined to decide whether three smaller inter prediction modes, i.e. 8×4, 4×8 and 4×4 are selected as its candidate modes or not. For each 8×8 block B_k in $MB_{m,n}$ with SAD_k the selection of candidate inter prediction modes is done as follows:

- If $SAD_k < \tau_3$, none of the smaller inter prediction modes is selected.
- If $SAD_k > \tau_4$, select all smaller inter prediction modes.
- Otherwise , select 8×4 and 4×8 modes only

Where

$$\tau_3 = \frac{1}{4} \times W1 \times AMB\_RC \tag{4.26}$$

$$\tau_4 = \frac{1}{4} \times W2 \times AMB\_RC \tag{4.27}$$

4. **Algorithmic flow**

Fig.4.7 shows the algorithmic flow of the proposed inter prediction mode selection algorithm. The algorithm is described as follows:

- Perform ME on 8×8 block sizes for current frame and then calculate AMB_RC and FRC using Eq.4.22 and Eq.4.23, respectively.

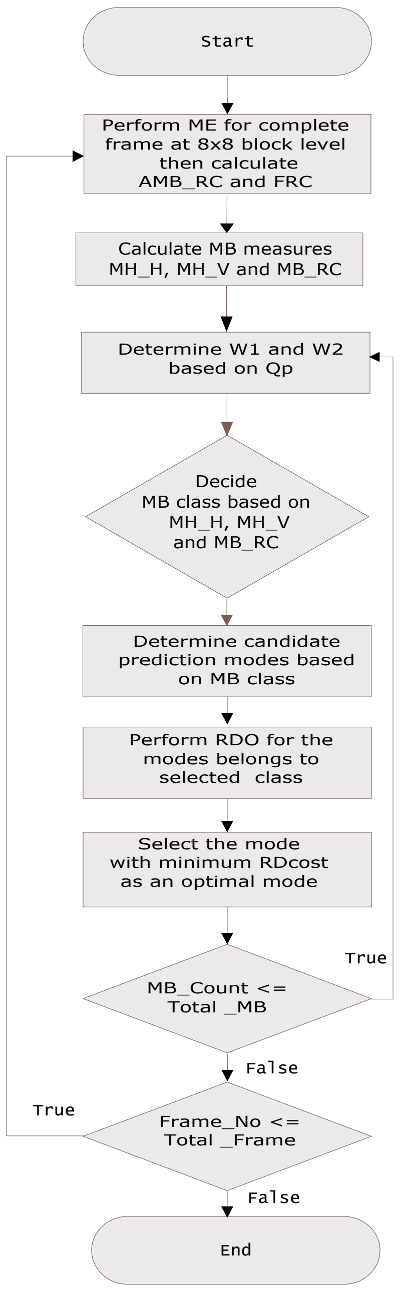

Start
Perform ME for complete frame at 8x8 block level then calculate AMB_RC and FRC
Calculate MB measures MH_H, MH_V and MB_RC
Determine W1 and W2 based on Qp
Decide MB class based on MH_H, MH_V and MB_RC
Determine candidate prediction modes based on MB class
Perform RDO for the modes belongs to selected class
Select the mode with minimum RDcost as an optimal mode
True
MB_Count <= Total _MB
False
True
Frame_No <= Total _Frame
False
End

FIGURE 4.7: Flowchart of the Proposed Inter Prediction Mode Selection Scheme

- Calculate MB directional motion homogeneity and residual complexity using Eq.4.12, Eq.4.13 and Eq.4.14, respectively.
- Determine W1 and W2 based on QP using Eq.4.24 and Eq.4.25, respectively.

- Decide the class of the current MB according to its residual complexity and motion homogeneity measures based on conditions listed in Table 4.6.
- Based on the class of current MB determine candidate inter prediction modes.
- Perform RDO process for candidate inter prediction modes.
- Select the mode with minimum RDcost as the optimal mode.
- If current MB is the last MB in current frame, go to the step 9. Otherwise, go to the step 2 for the next MB mode decision belonging to current frame.
- Finalize the encoding process, if current frame is the last frame. Otherwise, go to the step 1 for the next frame

4.3.4.2 Intra Prediction Mode Selection Algorithm

It is observed that the intra prediction modes that provide the least prediction or residual error will also result in minimum rate R and hence minimize the RD-cost. Moreover, the pixels along the local edge direction of the block normally have similar values. Therefore, a good prediction could be obtained if predicted pixels are generated using their neighboring pixels that are in the same directional correlation of the block. In this work local edge direction of the block is used to determine the intra prediction modes. The rest of the parts of this section describe the suggested technique.

1. **Calculation of local edge direction**

   In this work, Soble edge operator is used to acquire the local edge information of the block. The Sobel operator is applied on down sized image to create edge map which is used to calculate edge information i.e. edge direction and magnitude. It consists of two convolution operators that give the degree of difference in horizontal and vertical direction. The main advantage of this filter is that it provides both differencing and smoothing effect. Each element in edge map is an edge vector and is associated with 4×4 block of actual image. The edge vector for each block located at the ith row and jth column, in luma and chroma component of image can be define as

$$\begin{aligned} Vx_{i,j} = {} & X_{i-1,j+1} + 2 \times X_{i,j+1} + X_{i+1,j+1} \\ & -X_{i-1,j-1} - 2 \times X_{i,j-1} - X_{i+1,j-1} \end{aligned} \tag{4.28}$$

$$\begin{aligned} Vy_{i,j} = {} & X_{i+1,j-1} + 2 \times X_{i+1,j} + X_{i+1,j+1} \\ & -X_{i-1,j-1} - 2 \times X_{i-1,j} - X_{i-1,j+1} \end{aligned} \tag{4.29}$$

Where $Vx_{i,j}$ and $Vy_{i,j}$ denote the degree of difference in horizontal and vertical directions, respectively. Therefore, block edge magnitude roughly calculated as

$$Mag_{i,j} = |Vx_{i,j}| + |Vy_{i,j}| \tag{4.30}$$

The block edge direction can be calculated as

$$\theta_{i,j} = \frac{180}{\pi} \times \arctan(Vy_{i,j}/Vx_{i,j}), \theta_{i,j} \in [0, 2\pi] \tag{4.31}$$

In this work, a down size image is used in order to calculate local edge direction of the block. The downsizing factor of an image is selected as four. In scaled down image, 4×4 block size of an actual image is represented by one pixel and 16×16 block size is mapped to 4×4 block size. The use of scaled down image for feature extraction helps to reduce the computational complexity.

2. **Luma 4×4 prediction modes**

   As mentioned in section 2.3.1.1, there are 8 directional prediction modes and one DC prediction mode for 4×4 luma. Fig.4.8 shows that the border between any two adjacent directional prediction modes is the bisectrix of the two corresponding directions. For example, the border of Mode 1 (0) and Mode 8 (26.6) is the direction at 13.3, this is because that for Mode 8, prediction is done at an angle of approximately 26.6 above horizontal direction. It is important to note that Mode 3 and Mode 8 are adjacent due to circular symmetry of the prediction modes.

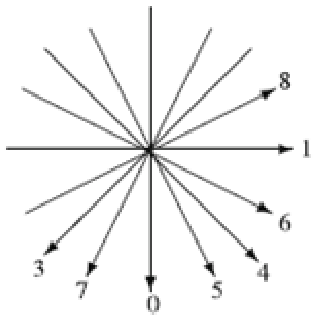
   

   FIGURE 4.8: Intra 4x4 Prediction Directions

   The edge direction of each 4×4 luma block is used to determine the prediction mode. The most probable intra prediction modes of 4×4 blocks based on local edge direction are calculated as follows.

   Let $\theta 1$ be the edge direction of each 4×4 block, and let $\alpha 1 = \theta 1 + \pi/2$ be its prediction direction, then Table 4.7 lists the prediction mode corresponding to its edge direction.

   As the DC mode has no direction and is used for smooth areas, it is always a candidate mode for intra 4×4. Furthermore, the two neighbors of the primary prediction mode in terms of directions are also taken as candidate

TABLE 4.7: Luma 4×4 Prediction Modes

Mode	Condition
1	$\alpha1 \in$ [0.0, 11.25] U]168.75, 191.25] U]348.75, 360.0]
8	$\alpha1 \in$] 11.25, 33.75] U] 191.25, 213.75]
3	$\alpha1 \in$] 33.75, 56.25] U] 213.75, 236.25]
7	$\alpha1 \in$] 56.25, 78.75] U] 236.25, 258.75]
0	$\alpha1 \in$] 78.75, 101.25] U] 258.75, 281.25]
5	$\alpha1 \in$] 101.25, 123.75] U] 281.25, 303.75]
4	$\alpha1 \in$] 123.75, 146.25] U] 303.75, 326.25]
6	$\alpha1 \in$] 146.25, 168.75] U] 326.25, 348.75]

modes. For example, if the primary prediction mode is Mode 1 then two additional candidate prediction modes will be Mode 8 and Mode 6. In short, for each 4×4 intra block, RDO calculation is performed only for 4 modes instead of 9.

3. **Luma 16×16 and chroma 8×8 prediction modes**

In the case of 16×16 luma and 8×8 chroma blocks, there are only two directional prediction modes, plus a plane prediction and a DC prediction mode as mentioned in section 2.3.1.2 and section 2.3.1.3, respectively. Therefore, the edge direction histogram for this case is based on three directions, i.e., horizontal, vertical and diagonal (plane) directions, as shown in Fig.4.9. Note that both diagonal down right and diagonal down left prediction modes are associated with the plane prediction.

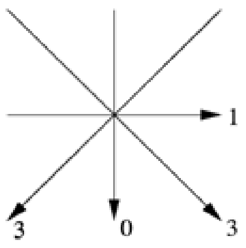


FIGURE 4.9: Intra 16×16 Prediction Directions

For 16×16 luma and 8×8 chroma blocks, the edge direction is calculated through edge direction histogram of their corresponding 4×4 blocks. The edge direction histogram for 16×16 luma block is calculated as Table 4.8 lists. Let $\theta2$ be the edge direction of its each constituent 4×4 block and let $\alpha2 = \theta2 + \pi/2$ be the prediction direction, then each bin in the edge direction histogram sums up the edge magnitude of all the 4×4 blocks having same edge direction. The histogram bin with maximum magnitude indicates the dominant block edge direction and is thus considered as most probable prediction direction. The primary prediction mode is selected based on this dominant block edge direction. It is important to note that only those bins are considered for selection of prediction direction which represents at least five similar directions 4×4 blocks.

TABLE 4.8: Luma 16×16 Prediction Modes

Mode	Condition	Edge Magnitude	Count
0	$\alpha2 \in$]67.5, 112.5] U]247.5, 292.5]	M0+=Mag($V_{i,j}$)	C0++
1	$\alpha2 \in$ [0.0, 22.5] U]157.5, 202.5] U]337.5, 360.0]	M1+=Mag($V_{i,j}$)	C1++
3	else	M3+=Mag($V_{i,j}$)	C3++

In order to select primary prediction mode for 8×8 chroma blocks, the same procedure is applied, except that the order of mode numbers is different and no limit is applied on similar direction blocks in histogram bin. Table 4.9 lists the edge direction histogram calculation method for 8×8 chroma component. Therefore, one primary prediction mode for a 16×16 luma block and 8×8 chroma block is selected through above described algorithm .

TABLE 4.9: Chroma 8×8 Prediction Modes

Mode	Condition	Edge Magnitude
1	$\alpha2 \in$ [0.0, 22.5] U]157.5, 202.5] U]337.5, 360.0]	M1+=Mag($V_{i,j}$)
2	$\alpha2 \in$]67.5, 112.5] U]247.5, 292.5]	M2+=Mag($V_{i,j}$)
3	else	M3+=Mag($V_{i,j}$)

Generally, 16×16 intra prediction mode is more suitable for MBs belonging to low textured or very smooth areas of a picture and mostly chroma components are very smooth. Therefore, DC prediction mode is always a candidate mode for luma 16×16 and chroma 8×8 blocks. The prediction candidate modes of luma 16×16 and chroma 8×8 are the DC mode and the primary prediction mode selected by dominant block edge direction. The RDO is performed for these candidate modes only.

4. **Algorithmic flow**

   Fig.4.10 shows the algorithmic flow of the proposed intra prediction mode selection algorithm.

   - Down size the input image by a factor 4. In scaled down image, 4×4 block size of an actual image is represented by one pixel and 16×16 block size is mapped to 4×4 block size.
   - Calculate local edge information of the 4×4 blocks in down sized image using Sobel edge operator.
   - Determine primary prediction mode for luma 4×4, luma 16×16 and chroma 8×8 based on local edge direction.
   - Choose candidate prediction modes for luma 4×4, luma 16×16 and chroma 8×8 (a). For each luma 4×4 block, the candidate prediction modes are: the primary prediction mode selected based on the local edge direction, two neighbors of the primary prediction mode in terms of directions and DC mode. (b) For each luma 16×16 and chroma 8×8 block, the primary prediction mode selected based on the local edge direction and DC mode are the candidate prediction modes.
   - Perform RDO process only for candidate intra prediction modes.
   - Decide the mode with minimum RDcost as an optimal mode.

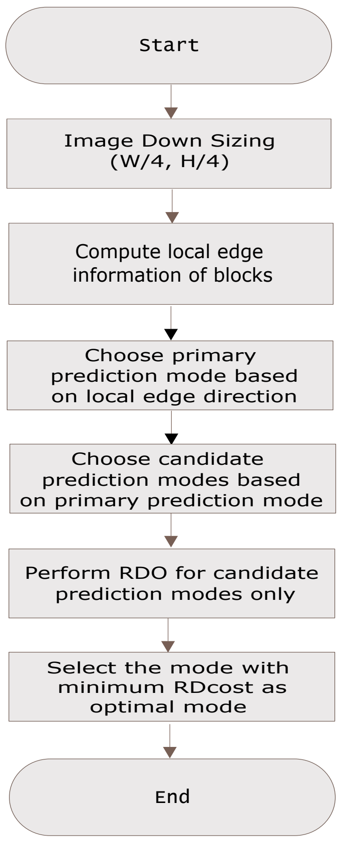

Start
Image Down Sizing (W/4, H/4)
Compute local edge information of blocks
Choose primary prediction mode based on local edge direction
Choose candidate prediction modes based on primary prediction mode
Perform RDO for candidate prediction modes only
Select the mode with minimum RDcost as optimal mode
End

FIGURE 4.10: Flowchart for the Proposed Intra Prediction Mode Selection Scheme

Table 4.10 lists the number of candidate intra prediction modes selected for RDO calculation by the proposed scheme.

TABLE 4.10: Candidate Intra Prediction Modes

MB Component	Block Size	Total Modes	Candidate Modes
Luma (Y)	4×4	9	4
Luma (Y)	16×16	4	2
Chroma (U,V)	8×8	4	2

4.3.5 RDO Mode Elimination

At this stage of the framework, reference RDO algorithm is used to calculate RDcost for shortlisted prediction modes of an MB. The prediction mode with minimum RDcost is selected as coding mode of an MB. In the proposed framework

RDO calculation is performed only for shortlisted candidate modes. This helps to significantly reduce the computational complexity of the encoding process.

4.3.6 Overall Framework Description

In the proposed framework, all training process of Adaboost classifiers is accomplished offline. The trained models of classifier are loaded at the beginning of encoding process. The encoding flow of the proposed framework is described as follows.

1. Down size the current frame for features calculation.
2. Perform 8×8 ME using 3-D RS motion estimator with one reference frame.
3. Calculate the features for prediction type decision.
4. Determine the most probable prediction type.
5. Go to Step 6, if prediction type is intra. Otherwise go to step 8.
6. Calculate features for the selection of suitable intra prediction block size.
7. Select a most probable block size to perform intra prediction.
8. Go to Step 9 if prediction type is inter. Otherwise go to step 12.
9. Perform skip early detection.
10. Calculate features for inter prediction mode selection.
11. Obtain candidate inter prediction modes.
12. Calculate features for intra prediction mode selection.
13. Obtain candidate intra prediction modes.
14. Perform RDO for shortlisted modes only.

15. Select the prediction type and mode with minimum RDcost as an optimal coding parameters for an MB.

16. Go to Step 4 to process next MB.

4.4 Experimental Analysis and Discussion

The proposed framework is incorporated into the H.264/AVC JVT Reference Software (Version JM 12.2) [81]. To evaluate the performance of proposed schemes, experiments are conducted on a PC with intel core i3-2100 CPU @ 3.1 GHz x and 2 GB RAM by using the video sequences covering a wide range of motion activities. The test conditions are set as follows:

- The used quantization parameters are 28, 32, 36 and 40.
- Encoder Main profile is used.
- Entropy encoding is set to CABAC.
- RDO are enabled.
- Fast motion estimation is enabled.
- MV resolution is set to 1/4 pel.
- MV search range is set to 32.
- Number of reference frames is set to 1 or 5.
- Five different frame formats, QCIF (144×176), CIF (352×288), NTSC (720×480), 720p (1280×720) and 1080p (1920×1080) are used.
- All test sequences are in 4:2:0 formats.
- Group of picture (GOP) structure is full I or IPPPPP.

- Frame rate is set to 30 fps.
- Number of frames encoded per sequences are 100 or 300.
- To calculate the mean results each test sequence is encoded three times independently for each quantizer.

Bjontegaard delta bit-rate (BDBR) [110], Bjontegaard delta peak signal-to-noise ratio (BDPSNR) and time saving (TS) are the three metrics used to evaluate the performance of proposed schemes. TS can be defined as follows.

$$TS = \frac{T_p - T_r}{T_r} \times 100\% \tag{4.32}$$

Where T_r and T_p are the encoding times of the reference software and the proposed algorithm, respectively. The positive values of BDPSNR, BDBR and TS indicate an increase whereas negative values represent a decrease.

4.4.1 Prediction Type Decision

In order to evaluate the performance of the proposed prediction type decision algorithm, three different frame formats, QCIF, CIF and NTSC are considered. For each test sequences, 100 frames are encoded in IPPPP structure with 5 reference frames. The period of I-frame is set to 100 i.e. all frames are encoded as P-frames except the first one which is encoded as I.

Table 4.11 lists the experimental results for variety of test sequences. It indicates that the proposed technique is about 25.61% faster than the full search method with negligible coding loss in terms of BDPSNR and BDBR i.e by the amount of 0.006 dB and 0.169%, respectively. The presented scheme shows a consistent gain in encoding time savings for all sequences ranging from 19.77% in *Akiyo* to

32.51% in *Washdc*. This is achieved with a maximum BDPSNR loss of 0.045 dB or a maximum increase in BDBR of 1.072%, and is thus negligible.

TABLE 4.11: Experimental Results of Prediction Type Decision Algorithm

Sequences	Format	TS (%)	BDBR (%)	BDPSNR (dB)
Foreman	QCIF	-31.53	0.167	-0.009
Claire		-20.56	-0.576	0.037
Coastguard		-30.9	-0.249	0.008
Container		-22.36	-0.021	0
Hall		-23.73	-0.011	0
Highway		-28.34	-0.508	0.019
Akiyo	CIF	-19.77	0.001	0
Mobile		-29.02	0.069	-0.003
MaD		-19.93	0.548	-0.025
Paris		-30.78	0.072	-0.004
Silent		-27.11	0.638	-0.027
Tempet		-30.15	0.429	-0.018
Flower	NTSC	-22.19	0.023	-0.002
Football		-25.76	1.072	-0.045
Mobile		-21.87	-0.062	0.003
Vtc1nw		-21.48	0.34	-0.011
Washdc		-32.51	0.419	-0.018
Galleon		-22.93	0.699	-0.022
Average		-25.61	0.169	-0.006

Table 4.12 shows the frequency of each class for all test sequences that is averaged using the results under four different QPs including 28, 32, 36 and 40. It can be inferred from the percentage of MBs belonging to particular class, that the reduction in encoding time is maximum for the sequences for which most of the MBs are classified into Class 1 and Class 2. For instance, in case of *Washdc* sequence, only 6.6% of MBs are classified to Class 3 and therefore, for the remaining 93.94% MBs, RDcost calculation is performed for either intra or inters prediction type only. This results in reduction of around 32.51% in encoding time. On the other hand, in case of *Akiyo* sequence, 31.94% of MBs are classified to Class 3 and for these MBs time consuming RDcost calculation is performed for both intra and inter prediction types resulting in comparatively lower time saving i.e., 19.77%.

Table 4.12: Frequency Distribution of Each Class (%) for Prediction Type Decision Algorithm

Sequences	Format	Class 1	Class 2	Class 3
Foreman	QCIF	0.02	92.59	7.39
Claire		0.01	66.83	33.16
Coastguard		0.03	90.56	9.41
Container		0	72.21	27.79
Hall		0	79.89	20.11
Highway		0.03	81.04	18.93
Akiyo	CIF	0.06	68	31.94
Mobile		0.2	89.7	10.1
MaD		0.18	66.04	33.78
Paris		0.01	92.21	7.78
Silent		0.46	87.09	12.45
Tempet		0.27	89.57	10.16
Flower	NTSC	0.05	65.58	34.37
Football		1.35	78.71	19.94
Mobile		0.04	69.28	30.68
Vtc1nw		0	67.24	32.76
Washdc		0.01	93.93	6.06
Galleon		0.64	71.44	27.92
Average		0.19	78.99	20.82

The Rate Distortion (RD) curves of the prediction type decision scheme and the JM reference exhaustive mode decision algorithm RDO are demonstrated in Fig.4.11. It shows that the suggested prediction type decision scheme achieves a similar RD performance as that of the JM reference software full search algorithm.

4.4.2 Early Mode Exclusion

4.4.2.1 SKIP Mode Early Detection Algorithm

In order to evaluate the skip mode early detection algorithm, three different frame resolutions, QCIF, CIF and NTSC are considered. For all test sequences 100 frames are encoded with 5 reference frames. GOP structure is set to IPPPPP and period of I-frame is set to 100.

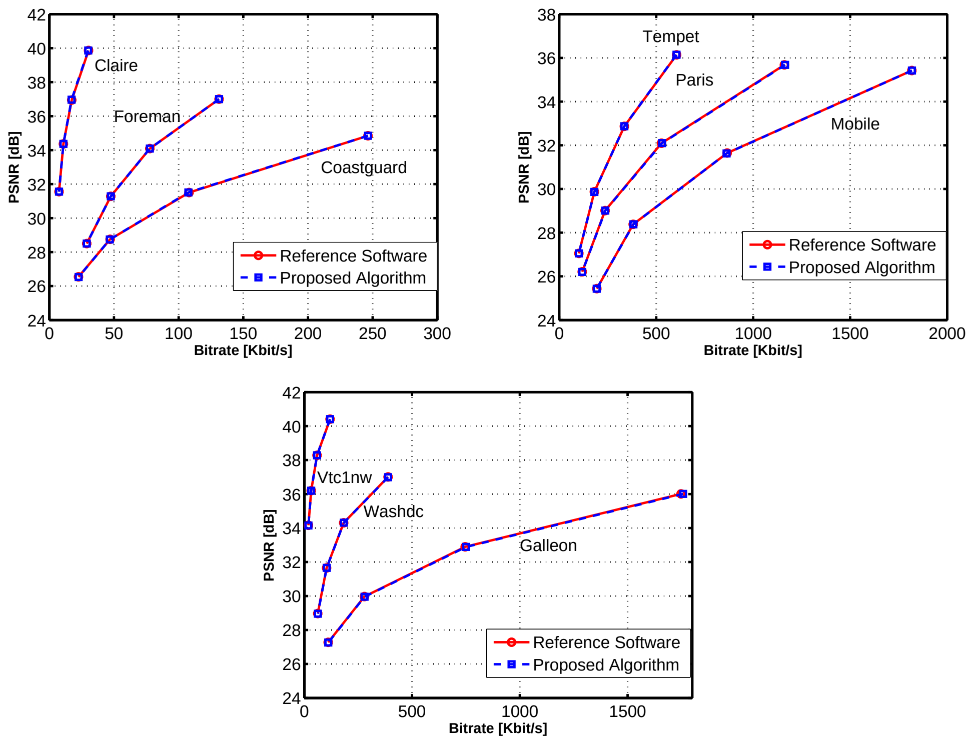


FIGURE 4.11: RD Curves of Proposed Prediction Type Decision Scheme and Reference Software

Table 4.13 enlists the experimental results for variety of test sequences in terms of Ts, BDPSNR and BDBR. It also gives the distribution of macroblocks for which SKIP mode is early detected. Experimental results show that on the average SKIP mode early detection algorithm is about 23.97% faster than the full exhaustive method with 0.019% decrease in BDBR and 0.002 dB increase in BDPSNR. It is important to note that SKIP early detection not only reduces the encoding time but also improves the rate-distortion performance.

Table 4.13 also shows that the percentage distribution of SKIP mode early detection is greatly dependent on motion activities exhibited in the sequence. The frequency of SKIP mode early detection is high for those sequences containing motionless or slow-motion content i.e. *Claire*, *Hall*, *Container*, *Akiyo*, *Vtc1nw* and etc that results in reduction of encoding time. For instance, in case of *Container* sequence, SKIP mode is detected for 65.6% of MBs and RDO process is performed

TABLE 4.13: Experimental Results of SKIP Mode Early Detection Algorithm

Sequences	Format	TS (%)	BDBR (%)	BDPSNR (dB)	SKIP Mode (%)
Foreman	QCIF	-9.78	-0.107	0.006	17.59
Carphone		-13.83	-0.505	0.026	25.73
Claire		-35.52	-0.743	0.048	59.16
Coastguard		-12.76	-0.339	0.01	19.61
Container		-39.51	-0.094	0.003	65.6
Hall		-37.32	0.255	-0.015	56.66
Highway		-17.52	0.228	-0.009	39.77
Akiyo	CIF	-37.73	-0.072	0.004	65.5
MaD		-32.6	0.136	-0.007	54.71
Paris		-14.59	0.04	-0.002	36.45
Silent		-32.33	0.017	0	53.52
Tempete		-7.17	-0.053	0.002	12.76
Football	NTSC	-13.2	0.196	-0.008	20.44
Intros		-29.66	-0.025	0	52.6
Mobile		-4.88	-0.062	0.002	7.82
Vtc1nw		-38.66	0.191	-0.006	64.63
Washdc		-28.59	-0.047	0.009	52.59
Galleon		-25.88	0.643	-0.02	47.78
Average		-23.97	-0.019	0.002	41.83

for the remaining 34.4% MBs. This has resulted in reduction of about 39.51% in encoding time. On the other hand, for high motion activity sequences such as *Mobile*, *Tempet*, *Foreman* and *Coastguard*, frequency of SKIP mode detection is small that results in comparatively lower time saving. For example, in case of *Mobile* sequence, SKIP mode early detection is performed for 7.82% of MBs only and for these MBs time consuming RDcost calculation is avoided that results in comparatively lower time saving i.e 4.88%.

The Rate Distortion (RD) curves of the SKIP mode early detection scheme and JM reference exhaustive mode decision algorithm are demonstrated in Fig.4.12. It shows that the the SKIP mode early detection algorithm achieves a similar RD performance as that of the JM reference software full search algorithm.

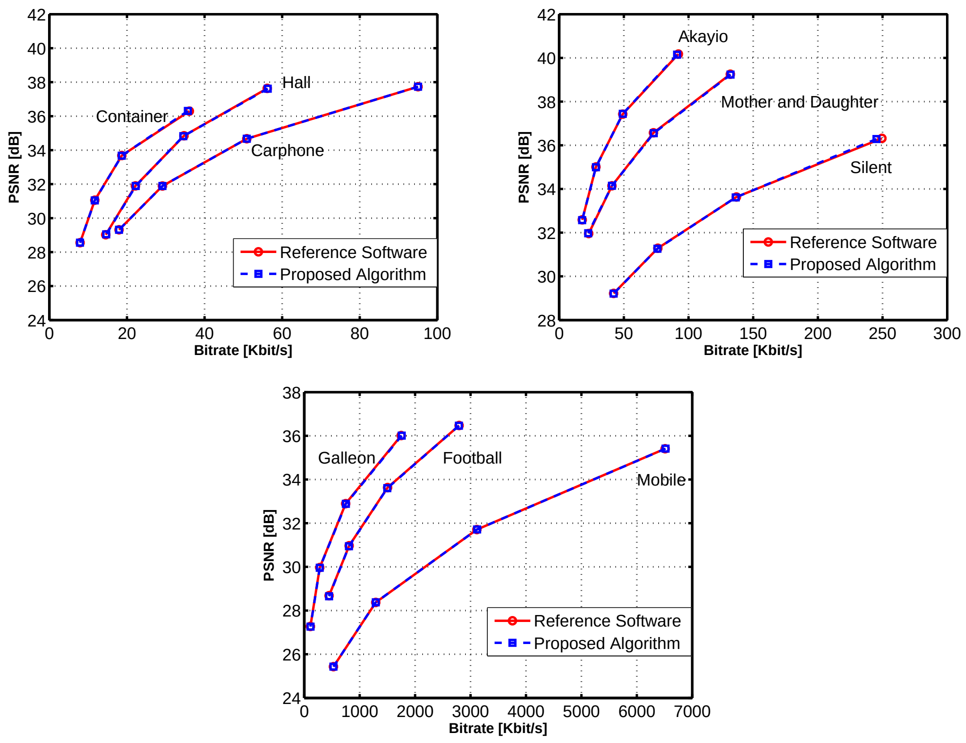


FIGURE 4.12: RD Curves of SKIP Mode Early Detection Scheme and Reference Software

4.4.2.2 Intra Prediction Block Size Detection Schemes

In order to evaluate the intra prediction block size detection schemes, three different frame resolutions QCIF, CIF and NTSC are used. All test sequences are having 100 frames. GOP structure is set to all intra i.e., all frames are encoded as I-frames. Table 4.14 shows the experimental results in terms of Ts, BDPSNR and BDBR. It demonstrates that on the average proposed adaptive threshold (scheme 1) and Adaboost classifier (scheme 2) based schemes have saved about 13.75% and 18.34% encoding time, respectively. In case of adaptive threshold based algorithm, time saving is achieved with marginal loss in prediction quality i.e. 0.379% increase in BDBR and 0.0318 dB decrease in BDPSNR. On the other hand, for Adaboost classifier based approach, the increased in BDBR is 0.548% and decrease in BDPSNR is about 0.045 dB on average. It can be noted that Adaboost classifier based scheme saved about 4.59% more encoding time as compared to adaptive threshold based scheme with 0.169% increase in BDBR and 0.013dB decrease in BDPSNR.

TABLE 4.14: Experimental Results of Intra Prediction Block Size Detection Schemes

Method		Scheme 1			Scheme 2		
Sequences	Format	TS (%)	BDBR (%)	BDPSNR (dB)	TS (%)	BDBR (%)	BDPSNR (dB)
Foreman	QCIF	-17.94	0.847	-0.058	-22.17	0.913	-0.065
Coastguard		-7.52	0.653	-0.034	-13.53	0.698	-0.068
Container		-9.53	0.182	-0.013	-12.08	0.82	-0.027
Hall		-10.98	0.642	-0.054	-14.4	0.739	-0.065
Mobile	CIF	-18.47	0.065	-0.006	-26.18	0.099	-0.009
Paris		-13.76	0.428	-0.034	-15.13	0.563	-0.044
Silent		-7.8	0.366	-0.042	-15.6	0.504	-0.069
Tempete		-18.1	0.273	-0.02	-24.27	0.408	-0.029
Flower	NTSC	-14.42	0.107	-0.009	-18.33	0.304	-0.024
Mobile		-18.98	0.226	-0.019	-24.95	0.268	-0.023
Washdc		-14.3	0.586	-0.079	-18.71	0.637	-0.094
Galleon		-13.25	0.177	-0.013	-14.78	0.626	-0.025
Average		-13.75	0.379	-0.0318	-18.34	0.548	-0.045

Table 4.15 illustrates the frequency of each class for both proposed schemes. The results under four different QPs including 28, 32, 36 and 40 are taken for all test sequences and are averaged to calculate class frequency. It can be seen that for both schemes, the reduction in encoding time is maximum for the sequences for which most of the MBs classified into Class 1 and Class 2.

For instance, in case of Adaboost scheme, 15.80% of MBs are classified to Class 3 and therefore, for the remaining 84.20% MBs, RDcost calculation is performed for either intra 4×4 or intra 16×16 only. This resulted in reduction of around 18.34% in encoding time. On the other hand, in case of adaptive threshold scheme, 25.62% of MBs are classified to Class 3 and for these MBs time consuming RDcost calculation is performed for both intra 4×4 and intra 16×16 resulting in comparatively lower time saving i.e 13.75%.

Table 4.15: Frequency Distribution of Each Class (%) for Intra Block Size Detection Schemes

Method		Scheme 1			Scheme 2		
Sequences	Format	Class 1	Class 2	Class 3	Class 1	Class 2	Class 3
Foreman	QCIF	81.76	3.19	15.05	88.04	3.56	8.44
Coastguard		58.3	0.28	41.41	73.54	0.42	26.04
Container		49.65	12.6	37.75	55.66	12.64	31.7
Hall		58.5	6.09	35.4	69.48	6.86	23.65
Mobile	CIF	84.72	3.6	11.68	91.45	3.83	4.71
Paris		68.18	3.56	28.26	76.24	4.4	19.36
Silent		55.45	3.42	41.13	78.48	3.48	18.04
Tempete		81.02	5.14	13.84	87.43	5.58	6.98
Flower	NTSC	57.76	20.76	21.47	63.59	22.8	13.61
Mobile		81.32	8.7	9.97	84.88	9.41	5.7
Washdc		75.39	2.08	22.53	86.51	2.93	10.56
Galleon		55.61	15.37	29.02	61.96	17.28	20.76
Average		67.31	7.07	25.62	76.44	7.76	15.80

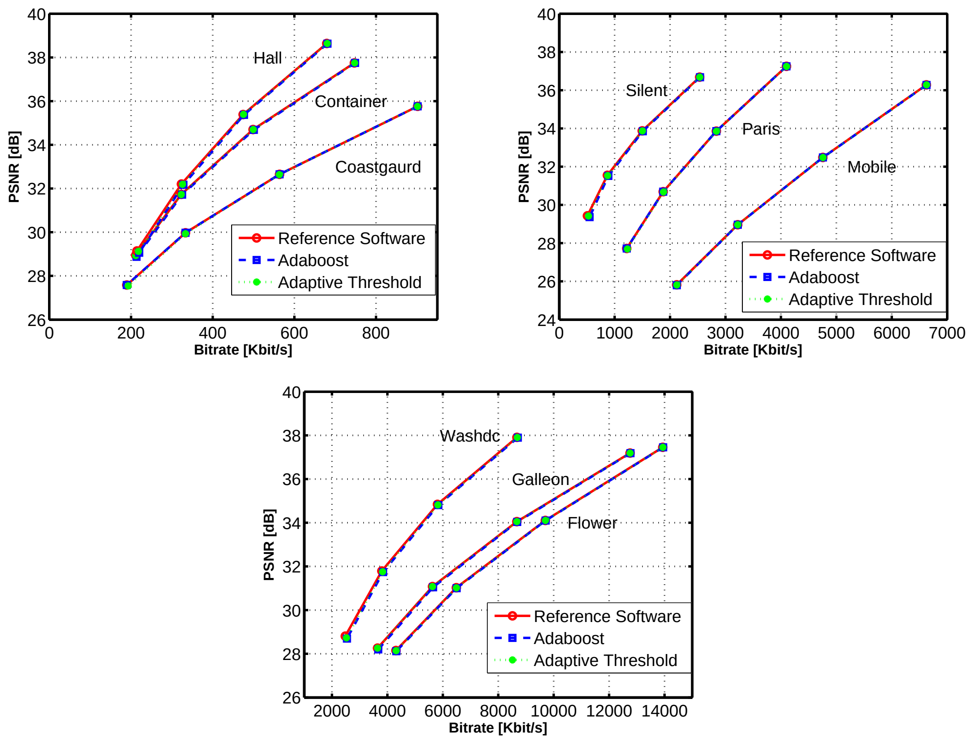


Figure 4.13: RD Curves of Intra Block Size Detection Schemes and Reference Software

The Rate Distortion (RD) curves of the intra prediction block size detection schemes and JM reference full search mode selection techniques are displayed in Fig.4.13. It shows that the proposed schemes achieve a similar RD performance as that of the JM reference software full search algorithm.

4.4.3 Rapid Mode Selection

4.4.3.1 Inter Prediction Mode Selection Scheme

In order to evaluate the performance of proposed inter prediction mode selection scheme, 100 frames for each test sequence are encoded. All frames except the first frame are encoded as P-frames and the number of reference frames is set to 5. The comparison of proposed scheme is done with three preceding works, Lee et al.'s [32], Enrquez et al's. [56] and Jeonet al.'s [109]. The performance comparison in terms of TS, BDPSNR and BDBR is shown in Table 4.16,Table 4.17 and Table 4.18, respectively.

TABLE 4.16: Performance Comparison of Inter Prediction Mode Selection Scheme in Terms of TS (%)

Method Sequences	Format	Proposed	Lee et al.'s [32]	Enrquez et al.'s [56]	Jeon et al.'s [109]
Carphone	QCIF	-60	-64	-39	-9
Claire		-73	-74	-65	-20
Coastguard		-58	-47	-28	-7
Highway		-67	-73	-53	-12
Miss-America		-68	-75	-60	-20
News		-65	-63	-57	-19
Akiyo	CIF	-76	-76	-62	-22
Container		-67	-69	-55	-23
Football		-56	-48	-23	-10
Foreman		-59	-62	-36	-10
Mobile		-58	-56	-29	-6
Paris		-61	-54	-40	-15
Average		-64	-63	-46	-14

Table 4.17: Performance Comparison of Inter Prediction Mode Selection Scheme in Terms of BDPSNR (dB)

Method Sequences	Format	Proposed	Lee et al.'s [32]	Enrquez et. al's [56]	Jeon et al.'s [109]
Carphone	QCIF	-0.01	-0.09	-0.06	0.01
Claire		-0.02	-0.08	0.01	0.01
Coastguard		-0.01	0.00	-0.04	-0.01
Highway		-0.01	0.01	-0.08	-0.02
Miss-America		-0.03	-0.06	0	0
News		-0.02	-0.05	-0.03	-0.02
Akiyo	CIF	0.04	-0.03	0	0
Container		-0.02	-0.02	-0.01	0.01
Football		-0.04	-0.04	0	0
Foreman		-0.06	-0.04	-0.03	0
Mobile		-0.05	-0.04	-0.02	-0.01
Paris		-0.01	-0.02	-0.05	0
Average		-0.02	-0.04	-0.02	0

Table 4.18: Performance Comparison of Inter Prediction Mode Selection Scheme in Terms of BDBR (%)

Method Sequences	Format	Proposed	Lee et al.'s [32]	Enrquez et al.'s [56]	Jeonet al.'s [109]
Carphone	QCIF	0.12	0.10	0.84	-0.27
Claire		0.2	-0.02	-0.17	0.1
Coastguard		0.42	0.06	1.74	0.12
Highway		0.2	1.50	2.55	0.67
Miss-America		0.02	-0.36	0.01	-0.14
News		0.28	-0.14	0.65	0.24
Akiyo	CIF	-0.27	-0.33	-0.14	-0.29
Container		0.15	-0.67	0.18	-034
Football		0.63	0.67	0.03	0.01
Foreman		0.56	0.34	0.49	-0.05
Mobile		0.71	0.87	0.34	0.17
Paris		0.34	0.53	0.89	0.03
Average		0.28	0.21	0.61	0.02

Experimental results illustrate that the proposed scheme reduces the encoding time by 64% on the average with negligible coding loss in terms of BDPSNR and BDBR i.e by the amount of 0.02 dB and 0.28%, respectively. The proposed inter prediction mode selection scheme shows a consistent gain in encoding time savings for all sequences ranging from 56% *Football* to 76% in *Akiyo*. This is achieved with a maximum BDPSNR loss of 0.06 dB or a maximum increase in BDBR of 0.71%, and is thus negligible.

The proposed inter prediction mode selection scheme also outshines Jeon et al.'s, Enrquez et al.'s and Lee et al.'s in terms of TS by 50%, 18% and 1% on the average, respectively. The overall gain in encoding time savings of proposed scheme as compared to Lee et al.'s method is not very significant (only 1%) and both algorithms perform equally well for low to medium motion sequences e.g *Akiyo*, *Claire*, *Container* and *News*. However, in addition, for test sequences having homogeneous motions e.g. *Coastguard*, *Football*, *Mobile* and *Paris*, the proposed scheme is faster than Lee et al.'s method and reduces 7% encoding time without sacrificing video quality. The reason behind this gain in encoding time saving for such sequences is the efficient convergence of 3-D RS motion estimator for homogeneous motion.

On the other hand, for test sequences having complex motion or haphazard motion e.g. *Carphone*, *Foreman* and *Highway*, the convergence of conventional 3-D RS motion estimator relatively slow. So for these sequences, Lee et al.'s technique is faster than the proposed scheme and reduces 4.33% encoding time on the average at the cost of 0.353% increase in BDBR and 0.0134 dB decrease in BDPSNR. Hence, it can be concluded that the proposed scheme provides higher and consistent gain in the encoding time saving for most of the video sequences as compared to Jeon et al.'s, Enrquez et al.'s and Lee et al.'s techniques. Only for the sequences having complex motion, Lee et al.'s method is slightly better than the proposed technique in terms of speed.

Table 4.19 shows the frequency of each MB class for all test sequences, that is averaged using the results under four different QPs including 28, 32, 36 and 40. It can be seen that the percentage distribution is greatly dependent on motion activities exhibited in the sequence.

Table 4.19: Frequency of Each MB Class (%) for Inter Prediction Mode Selection Scheme

Sequences	Format	Class 1	Class 2	Class 3	Class 4	Class 5
Carphone	QCIF	51.3	25.8	4.7	4.4	13.8
Claire		82.3	9.5	0.6	0.8	6.8
Coastguard		48.3	29.5	3.2	1.8	17.2
Highway		57.5	12.8	5.3	4.9	19.5
Miss-America		68.2	10.6	4.3	5.7	11.2
News		79.4	14.0	0.3	1.4	4.9
Akiyo	CIF	84.0	8.0	0.5	1.5	6.0
Container		59.6	16.2	2.4	2.5	19.3
Football		39.0	33.5	4.9	3.9	18.7
Foreman		44.2	27.3	4.2	4	20.3
Mobile		42.5	30.7	3.1	2.6	21.1
Paris		61.7	28.5	1.1	1.7	7.0
Average		59.84	20.53	2.88	2.93	13.82

It can be inferred from the percentage of MBs belonging to Class 2, that the reduction in encoding time is maximum for the sequences with lower to medium motion such as *Akiyo*, *Claire*, *Miss-America* and *News*. For instance, in case of *Akiyo* sequence, only 8% of MBs are classified to Class 2 and therefore, for the remaining 92% MBs, RDcost calculation is performed for large prediction block sizes only. This resulted in reduction of around 76% in encoding time.

On the other hand, for high motion activity sequences such as *Coastguard*, *Football*, *Foreman* and *Mobile*, higher number of MBs are classified to class 2 resulting in comparatively lower time saving. For example, in case of *Football* sequence, 33.5% of MBs are classified to Class 2 and for these MBs time consuming RDcost calculation is performed up to small prediction block sizes resulting in comparatively lower time saving i.e 56%.

Table 4.20 enlists the simulation results for variety of test sequences in term of BDPSNR, BDBR and TS.

Table 4.20: Experimental Results of Inter Prediction Mode Selection Scheme

Sequences	Format	BDPSNR (dB)	BDBR (%)	TS (%)
Carphone	QCIF	-0.01	0.12	-60
Claire		-0.03	0.2	-73
Coastguard		-0.02	0.42	-58
Highway		-0.01	0.2	-67
Miss-America		-0.03	0.02	-68
News		-0.05	0.28	-65
Mother and daughter		0	-0.12	-70
Akiyo	CIF	0.03	-0.27	-76
Container		-0.03	-0.15	-67
Football		-0.04	0.63	-56
Foreman		-0.03	0.56	-59
Mobile		-0.02	0.71	-58
Paris		-0.01	0.34	-61
Tempete		-0.03	0.7	-55
Silent		0	-0.01	-69
Flower	NTSC	-0.05	0.74	-57
Galleon		-0.02	-0.07	-61
Intros		0	0.01	-66
Vtc1nw		0	0.19	-69
Washdc		0.02	-0.3	-65
Mobcol	720p	0.01	0	-57
Parkrun		-0.03	0.75	-56
Shields		-0.01	0.23	-61
Stockholm		0	-0.07	-63
Pedestrain_area	1080p	0.01	0.23	-61
Rush_hour		-0.02	0.63	-60
Average		-0.014	0.241	-63

The Rate Distortion (RD) curves of the proposed inter prediction mode selection scheme and JM reference exhaustive mode decision algorithm are shown in Fig.4.14. It depicts that the proposed inter prediction mode selection scheme achieves a similar RD performance as that of the JM reference software full search algorithm.

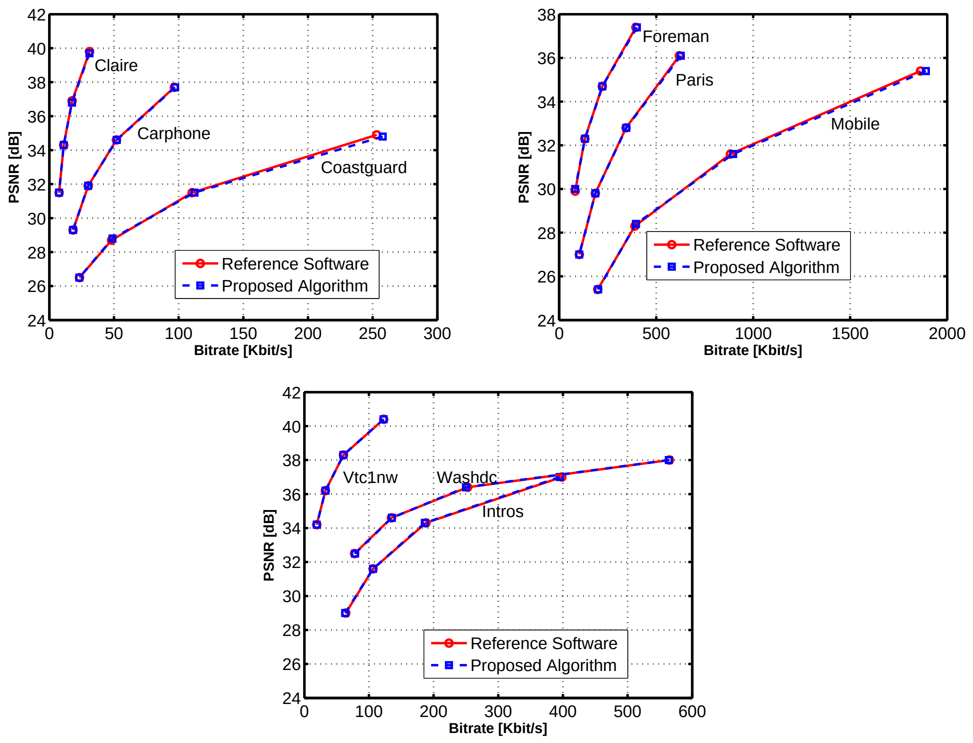


FIGURE 4.14: RD Curves of Proposed Inter Prediction Mode Selection Scheme and Reference Software

4.4.3.2 Intra Prediction Mode Selection Scheme

In order to perform the experimental analysis of the proposed intra prediction mode selection scheme, three different frame resolutions, QCIF, CIF and NTSC are considered. All test sequences having 300 frames and are encoded using all intra GOP structure. Table 4.21, Table 4.22 and Table 4.23 compare the proposed scheme with previous works in terms of TS, BDPSNR and BDBR, respectively. The comparison of proposed scheme is done from two preceding works, Pan et al.'s [65] and Wang at el.'s [83]. The simulation results indicate that the proposed scheme reduces the encoding time by 61.27% on the average with negligible coding loss in terms of BDPSNR and BDBR i.e by the amount of 0.232 dB and 3.195%, respectively. The proposed scheme also outshines Pan et al.'s and Wang et al.'s schemes in terms of TS by 3.22% and 3.1% on the average, respectively.

TABLE 4.21: Performance Comparison of Intra Prediction Mode Selection Scheme in Terms of TS (%)

Method Sequences	Format	Pan et al.'s [65]	Wang et al.'s [83]	Proposed
News	QCIF	-55.34	-58.03	-63.75
Container		-56.36	-57.32	-66.76
Silent		-65.17	-60.66	-62.82
Coastguard		-55.03	-57.78	-59.91
Mobile	CIF	-59.09	-58.07	-59.83
Paris		-57.78	-58.78	-59.79
Stefan		-57.97	-58.56	-58.9
Tempete		-57.70	-56.86	-58.38
Average		-58.06	-58.26	-61.27

TABLE 4.22: Performance Comparison of Intra Prediction Mode Selection Scheme in Terms of BDPSNR (dB)

Method Sequences	Format	Pan et al.'s [65]	Wang et al.'s [83]	Proposed
News	QCIF	-0.294	-0.348	-0.277
Container		-0.234	-0.293	-0.232
Silent		-0.183	-0.255	-0.218
Coastguard		-0.106	-0.255	-0.108
Mobile	CIF	-0.255	-0.237	-0.286
Paris		-0.23	-0.274	-0.238
Stefan		-0.242	-0.301	-0.284
Tempete		-0.229	-0.254	-0.213
Average		-0.221	-0.278	-0.232

Table 4.24 illustrates the simulation results for variety of test sequences in term of BDPSNR, BDBR and TS. It indicates that the proposed scheme shows a consistent gain in encoding time savings for all sequences ranging from 56.67% in *Washdc* to 68.49% in *Vtc1nw*. This is achieved with a maximum BDPSNR loss of 0.286 dB or a maximum increase in BDBR of 4.448%, and is thus negligible. The time saving is high for those sequences that contain strong spatial homogeneity like *Vtc1nw* and *Akaiyo*. On the other hand, the sequences *Washdc* and *Foreman* have more non-homogeneous regions in frames. Therefore, the time saving for these sequences is not as much compared with previous sequences.

Table 4.23: Performance Comparison of Intra Prediction Mode Selection Scheme in Terms of BDBR (%)

Method Sequences	Format	Pan et al.'s [65]	Wang et al.'s [83]	Proposed
News	QCIF	3.902	4.451	3.573
Container		3.695	4.44	3.42
Silent		3.54	4.58	3.982
Coastguard		2.361	4.034	1.937
Mobile	CIF	3.168	2.871	3.189
Paris		3.21	3.678	3.064
Stefan		3.717	3.889	3.408
Tempete		3.514	3.735	2.988
Average		3.388	3.960	3.195

Table 4.24: Experimental Results of Intra Prediction Mode Selection Scheme

Sequences	Format	TS (%)	BDBR (%)	BDPSNR (dB)
Foreman	QCIF	-57.62	4.448	-0.282
Carphone		-58.7	3.776	-0.258
Coastguard		-59.91	1.937	-0.108
Container		-66.76	3.42	-0.232
Hall		-58.68	2.837	-0.217
Highway		-58.81	2.731	-0.135
Akiyo	CIF	-67.35	1.804	-0.121
Mobile		-59.83	3.189	-0.286
MaD		-67.2	2.241	-0.116
Paris		-59.79	3.064	-0.238
Silent		-64.88	2.209	-0.103
Tempete		-58.38	2.988	-0.213
Football	NTSC	-60.56	4.394	-0.211
Intros		-63.98	2.977	-0.104
Mobile		-60.02	3.294	-0.272
Vtc1nw		-68.49	2.592	-0.124
Washdc		-56.67	3.225	-0.233
Galleon		-61.71	2.864	-0.207
Average		-61.63	2.999	-0.192

The Rate Distortion (RD) curves of the proposed intra prediction mode selection scheme and JM reference exhaustive mode decision algorithm are shown in Fig.4.15. It indicates that the proposed efficient intra prediction mode selection scheme achieves almost similar RD performance as that of the JM reference software full search algorithm.

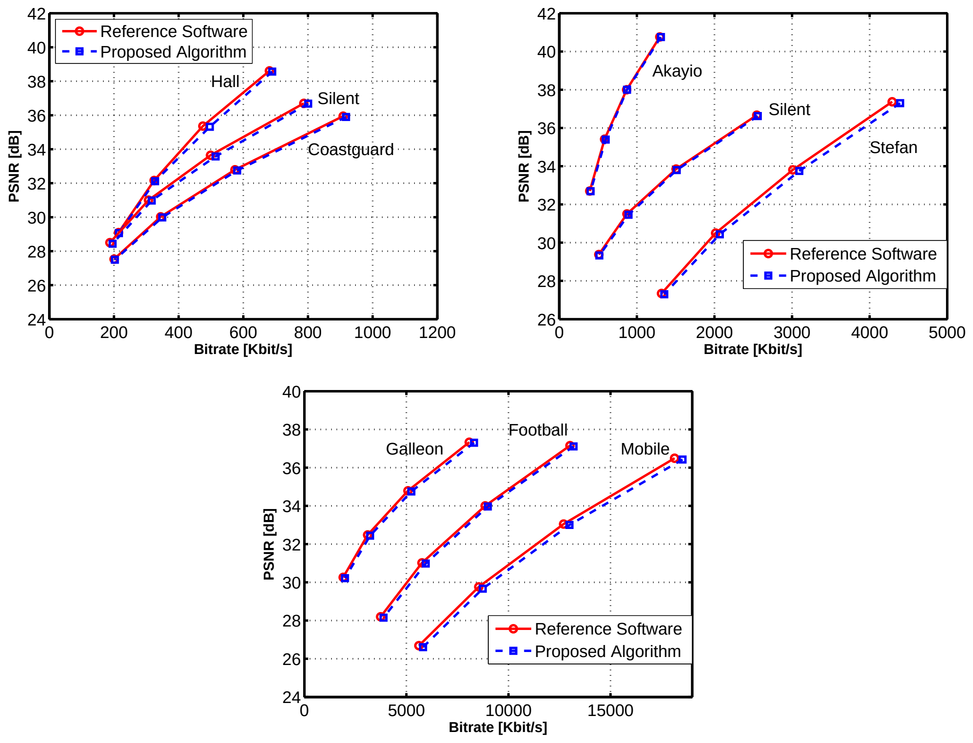


FIGURE 4.15: RD Curves of Proposed Intra Mode Selection Scheme and Reference Software

4.4.4 Overall Performance Evaluation of the Proposed Framework

In order to evaluate the performance of over all proposed framework, five different frame resolutions, QCIF and CIF, NTSC, 720p and 1080p are considered. Experiments are performed using two different encoding structures IPPPPPP and IIIIIII i.e. all I-frames.

4.4.4.1 Experiments on IPPPPP Sequences

Table 4.25 explains the simulation results for variety of test sequences in term of BDPSNR, BDBR and TS. In this experimental analysis, 100 frames for each test

TABLE 4.25: Experimental Results on IPPPPP Sequences

Sequences	Format	BDPSNR (dB)	BDBR (%)	TS (%)
Carphone	QCIF	-0.012	0.145	-71
Claire		-0.035	0.458	-85
Coastguard		-0.024	0.567	-67
Highway		-0.028	0.251	-79
Miss-America		-0.039	0.399	-81
News		-0.032	0.413	-76
Mother and daughter		-0.016	0.11	-82
Akiyo	CIF	-0.012	0.106	-87
Container		-0.036	0.518	-75
Football		-0.064	0.789	-67
Foreman		-0.065	0.906	-66
Mobile		-0.029	0.391	-69
Paris		-0.067	0.956	-65
Tempete		-0.043	0.852	-66
Silent		-0.013	0.123	-80
Flower	NTSC	-0.054	0.949	-65
Galleon		-0.02	0.126	-70
Intros		-0.016	0.06	-78
Vtc1nw		-0.039	0.351	-81
Washdc		-0.015	0.394	-74
Mobcol	720p	-0.01	0.128	-66
Parkrun		-0.052	0.856	-65
Shields		-0.044	0.363	-70
Stockholm		-0.025	0.153	-69
Pedestrain_area	1080p	-0.008	0.528	-84
Rush_hour		-0.023	0.913	-69
Riverbed		-0.035	0.629	-71
Sunflower		-0.029	0.467	-73
Average		-0.032	0.461	-73.25

sequence are encoded and the period of I-frames is set to 100, i.e., first frame is encoded as I-frame and rest of the frames are encoded as P-frames. The number of reference frames for motion estimation is set to 5. Table 4.25 shows that the proposed framework achieves 73.25% encoding time saving, which means that the proposed framework only takes about 1/4th of the time that is taken by the exhaustive search algorithm in JM12.2. This time saving is achieved with negligible loss in BDPSNR (0.032 dB) and increments in BDBR (0.461%). The proposed framework shows a consistent gain in encoding time savings for all sequences ranging from 65% in *Football* to 87% in *Akiyo*. This is achieved with a maximum BDPSNR loss of 0.067 dB or a maximum increase in BDBR of 0.956%, and is thus negligible.

TABLE 4.26: Performance Comparison on IPPP Sequences in Terms of TS (%)

Method Sequences	Format	Proposed	Lee et al.'s [32]	Enrquez et al.'s [56]	Jeon et al.'s [109]
Carphone	QCIF	-71	-64	-39	-9
Claire		-85	-74	-65	-20
Coastguard		-67	-47	-28	-7
Highway		-79	-73	-53	-12
Miss-America		-81	-75	-60	-20
News		-76	-63	-57	-19
Akiyo	CIF	-87	-76	-62	-22
Container		-75	-69	-55	-23
Football		-67	-48	-23	-10
Foreman		-66	-62	-36	-10
Mobile		-69	-56	-29	-6
Paris		-65	-54	-40	-15
Average		-74	-63	-46	-14

TABLE 4.27: Performance Comparison on IPPP Sequences in Terms of BDPSNR (dB)

Method Sequences	Format	Proposed	Lee et al.'s [32]	Enrquez et al.'s [56]	Jeon et al.'s [109]
Carphone	QCIF	-0.02	-0.09	-0.06	0.01
Claire		-0.04	-0.08	0.01	0.01
Coastguard		-0.02	0.00	-0.04	-0.01
Highway		-0.03	0.01	-0.08	-0.02
Miss-America		-0.04	-0.06	0	0
News		-0.03	-0.05	-0.03	-0.02
Akiyo	CIF	-0.01	-0.03	0	0
Container		-0.04	-0.02	-0.01	0.01
Football		-0.07	-0.04	0	0
Foreman		-0.06	-0.04	-0.03	0
Mobile		-0.02	-0.04	-0.02	-0.01
Paris		-0.06	-0.02	-0.05	0
Average		-0.04	-0.04	-0.02	0

TABLE 4.28: Performance Comparison on IPPP Sequences in Terms of BDBR (%)

Method Sequences	Format	Proposed	Lee et al.'s [32]	Enrquez et al.'s [56]	Jeon et al.'s [109]
Carphone	QCIF	0.14	0.10	0.84	-0.27
Claire		0.45	-0.02	-0.17	0.1
Coastguard		0.56	0.06	1.74	0.12
Highway		0.25	1.50	2.55	0.67
Miss-America		0.39	-0.36	0.01	-0.14
News		0.36	-0.14	0.65	0.24
Akiyo	CIF	0.11	-0.33	-0.14	-0.29
Container		0.52	-0.67	0.18	-034
Football		0.78	0.67	0.03	0.01
Foreman		0.91	0.34	0.49	-0.05
Mobile		0.39	0.87	0.34	0.17
Paris		0.95	0.53	0.89	0.03
Average		0.48	0.21	0.61	0.02

Table 4.26, Table 4.27 and Table 4.28 show the performance comparison of the proposed framework in term of TS, BDPSNR and BDBR, respectively. The comparison is made with Jeon et al.'s [109], Enrquez et al.'s [56], and Lee et al.'s [32] algorithms for various sequences with IPPPP coding structure. The simulation results illustrate that the proposed framework also outshines Jeon et al.'s, Enrquez et al.'s and Lee et al.'s in terms of time saving Ts by 60%, 28% and 11% on the average, respectively. Jeon et al.'s, Enrquez et al.'s and Lee et al.'s algorithms have reduced the coding time on the average 14, 46, and 63%, respectively with the average of 0.221, 0.278, and 0.270 % increment in BDBR and 0, 0.02 and 0.04 dB losses in BDPSNR, respectively.

The Rate Distortion (RD) curves of the proposed framework and JM reference full search algorithm RDO are demonstrated in Fig.4.16. It shows that the proposed framework achieves a similar RD performance as that of the JM reference software exhaustive mode decision technique.

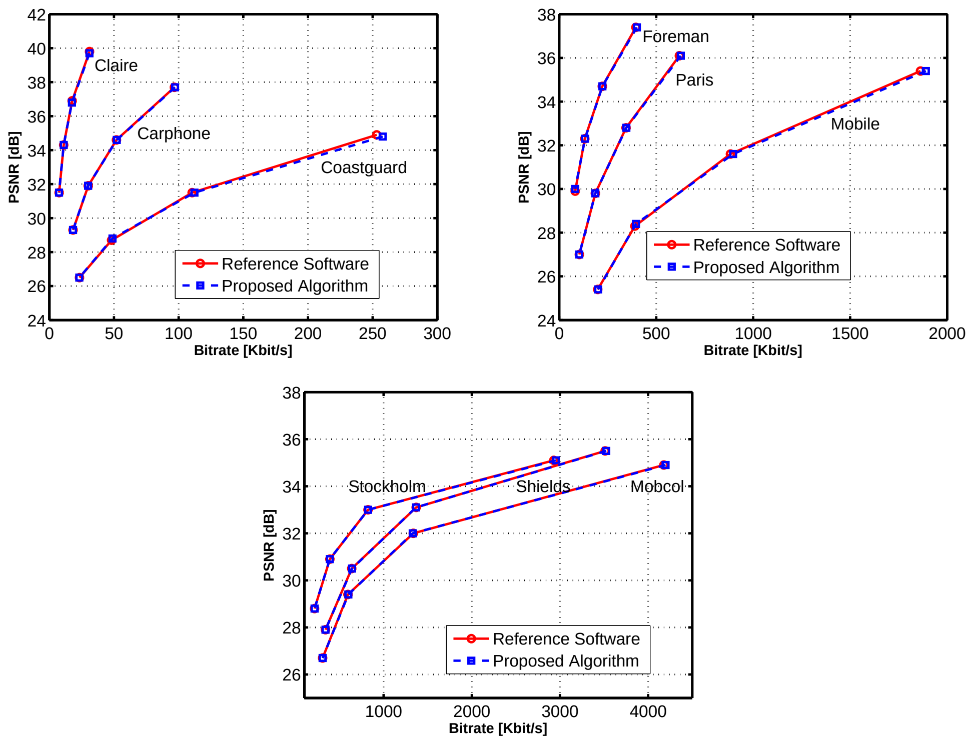


FIGURE 4.16: RD Curves of Proposed Framework and Reference Software for IPPPPP Sequences

4.4.4.2 Experiments on All Intra Frames Sequences

In this simulation, for each sequence 300 frames are encoded. The period of I-frames is set to 1.

TABLE 4.29: Performance Comparison of All Intra Frame Sequences in Terms of TS (%)

Method Sequences	Format	Pan et al.'s [65]	Wang et al.'s [83]	Bharanitharan et al.'s [76]	Proposed
News	QCIF	-55.34	-58.03	-68.02	-73.06
Container		-56.36	-57.32	-67.74	-79.53
Silent		-65.17	-60.66	-69.46	-71.77
Coastguard		-55.03	-57.78	-69.26	-76.51
Mobile	CIF	-59.09	-58.07	-68.89	-75.64
Paris		-57.78	-58.78	-67.83	-71.44
Stefan		-57.97	-58.56	-67.04	-70.12
Tempete		-57.70	-56.86	-69.53	-75.95
Average		-58.06	-58.26	-68.47	–74.25

Table 4.30: Performance Comparison of All Intra Frame Sequences in Terms of BDPSNR (dB)

Method Sequences	Format	Pan et al.'s [65]	Wang et al.'s [83]	Bharanitharan et al.'s [76]	Proposed
News	QCIF	-0.294	-0.348	-0.285	-0.298
Container		-0.234	-0.293	-0.241	-0.256
Silent		-0.183	-0.255	-0.232	-0.287
Coastguard		-0.106	-0.255	-0.181	-0.149
Mobile	CIF	-0.255	-0.237	-0.250	-0.286
Paris		-0.23	-0.274	-0.288	-0.268
Stefan		-0.242	-0.301	-0.290	-0.312
Tempete		-0.229	-0.254	-0.252	-0.254
Average		-0.221	-0.278	-0.270	-0.264

Table 4.31: Performance Comparison of All Intra Frame Sequences in Terms of BDBR (%)

Method Sequences	Format	Pan et al.'s [65]	Wang et al.'s [83]	Bharanitharan et al.'s [76]	Proposed
News	QCIF	3.902	4.451	3.439	3.621
Container		3.695	4.44	2.976	3.685
Silent		3.54	4.58	4.013	4.992
Coastguard		2.361	4.034	2.820	2.585
Mobile	CIF	3.168	2.871	2.860	3.104
Paris		3.21	3.678	2.891	3.487
Stefan		3.717	3.889	3.730	3.721
Tempete		3.514	3.735	3.601	3.482
Average		3.388	3.960	3.270	3.584

Table 4.29, Table 4.30 and Table 4.31 show the comparison of the proposed framework with three previous intra prediction mode selection techniques Pan et al.'s [65], Wang et al.'s [83] and Bharanitharan et al.'s [76] in terms of TS, BDPSNR and BDBR, respectively.

Experimental results display that the proposed scheme reduces the encoding time by 74.25% on the average with negligible coding loss in terms of BDPSNR and BDBR i.e by the amount of 0.264 dB and 3.584%, respectively. The proposed scheme also outperforms Pan et al.'s, Wang et al.'s and Bharanitharan et al.'s in terms of TS by 11.44%, 11.24% and 4.93% on the average, respectively. Where

Pan et al.'s, Wang et al.'s, and Bharanitharan et al's algorithms have reduced the coding time on the average 58.05, 58.26 and 69.32%, respectively with the average of 0.221, 0.278, and 0.270 dB losses in BDPSNR and 3.338, 3.960, and 3.270% increments in BDBR, respectively.

TABLE 4.32: Experimental Results of All Intra Frame Sequences

Sequences	Format	TS (%)	BDBR (%)	BDPSNR (dB)
Foreman	QCIF	-69.16	4.649	-0.312
Carphone		-74.61	3.946	-0.269
Claire		-82.63	2.318	-0.13
Coastguard		-75.64	2.585	-0.149
Container		-79.53	3.685	-0.256
News		-73.06	3.621	-0.298
Silent		-71.77	4.992	-0.287
Akiyo	CIF	-84.07	2.107	-0.155
Mobile		-75.64	3.104	-0.286
MaD		-76.59	2.264	-0.131
Paris		-71.44	3.487	-0.268
Stefan		-70.12	3.721	-0.312
Tempete		-75.95	3.482	-0.254
Flower	NTSC	-66.35	4.814	-0.337
Football		-67.47	4.528	-0.259
Intros		-75.17	3.242	-0.139
Vtc1nw		-80.03	2.825	-0.159
Washdc		-70.53	3.326	-0.262
Galleon		-74.26	2.985	-0.241
Mobcol	720p	-70.29	3.336	-0.171
Parkrun		-64.97	4.237	-0.294
Shields		-68.01	3.743	-0.268
Stockholm		-70.85	2.615	-0.179
Pedestrain_area	1080p	-76.36	2.679	-0.202
Rush_hour		-72.67	2.967	-0.18
Riverbed		-73.48	3.367	-0.228
Sunflower		-77.3	3.408	-0.231
Average		-73.63	3.409	-0.232

Table 4.32 shows the simulation results of the proposed framework in all intra encoding structures for variety of video sequences. It indicates that the proposed framework achieves 73.63% encoding time saving, which means that the proposed

framework only takes about 1/4th of the time that is taken by the exhaustive search algorithm in JM12.2. This time saving is achieved with negligible loss in BDPSNR (0.337 dB) and increments in BDBR (3.409%). The proposed framework shows a consistent gain in encoding time savings for all sequences ranging from 64.97% in *Parkrun* to 84.7% in *Akiyo*. This is achieved with a maximum BDPSNR loss of 0.067 dB or a maximum increase in BDBR of 4.992%, and is thus negligible.

The Rate Distortion (RD) curves of the proposed framework and JM reference exhaustive mode decision algorithm are demonstrated in Fig.4.17. It explains that the proposed framework achieves almost similar RD performance as that of the JM reference software full search algorithm.

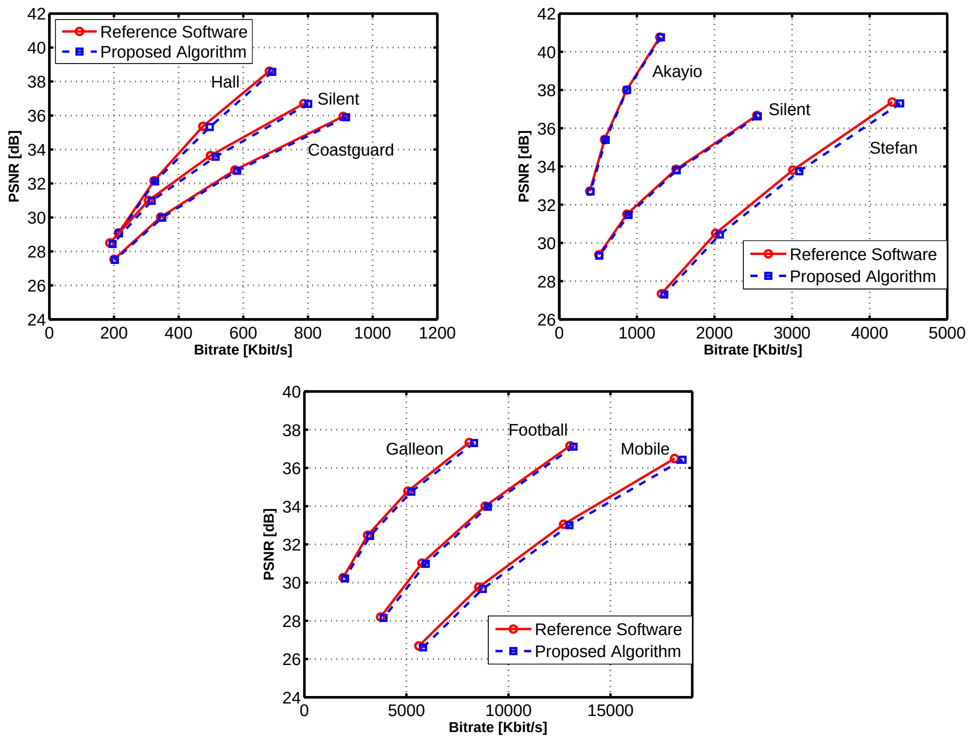


FIGURE 4.17: RD curves of Proposed Framework and Reference Software for All Intra Frame Sequences

4.5 Summary

This chapter presents an efficient framework for macroblock prediction to minimize the computational complexity of the macroblock prediction process. It consists of several innovative algorithms to decide macroblock prediction type, prediction modes and block size (intra or inter), skip macroblock early detection, appropriate block size selection for intra prediction, directional mode selection for intra prediction and inter mode (block sizes) selection. This framework exploits spatial and temporal statistics of video sequence in order to exclude as many intra and inter prediction modes as possible prior to the RDO process. Experiments indicate that the presented framework reduces the encoding time up to 74% with 0.032 dB decrease in BDPSNR and 0.461% increase in BDBR as compared to exhaustive RDO method. The suggested framework is equally beneficial for desktop and power-critical embedded systems having constraints of processing and computational power.

The research contributions related to this chapter:

- Muhammad Asif, Imtiaz A. Taj, S. M. Ziauddin, Maaz Bin Ahmad and Atif Raza " *An efficient inter prediction mode selection scheme for advanced video coding based on motion homogeneity and residual complexity* ", IEEJ Transactions on Electrical and Electronic Engineering.
- Muhammad Asif, Imtiaz A. Taj, S. M. Ziauddin, Maaz Bin Ahmad and M. Tahir " *An efficient framework for prediction parameters selection in advanced video coding* ", Elsevier Journal of Measurement.
- Muhammad Asif, Imtiaz A. Taj, S. M. Ziauddin, Maaz Bin Ahmad and M. Tahir " *An Efficient scheme for intra prediction block size and mode selection in advanced video coding* ", 13th International Conference on Frontiers in Information Technology (FIT2015).

Chapter 5

COMPLEXITY REDUCTION OF MOTION COMPENSATION

5.1 Introduction

In predictive coding, motion compensation is performed to temporally predict a macroblock. In H.264/AVC, motion compensation takes about one-fourth of the decoding time and also takes significant part of encoding time [36]. It comprises two modules i.e. data manipulation and interpolation. Both of these modules contribute significantly to increase the computational complexity of motion compensation and need to be optimized. Most of the existing techniques related to complexity reduction of motion compensation algorithm either address the computational complexity issue related to the data manipulation module or the data interpolation module. Few techniques which cover both the aspects do not fully optimize the looping overhead, access of memory and the interpolation process.

This chapter presents a computationally efficient scheme for motion compensation in order to perform inter prediction of a macroblock. The proposed technique creates margin around the reference frame to prevent the clipping of unrestricted motion vectors. Moreover, it uses the inter prediction mode (variable block sizes) information of a macroblock to reduce the looping overhead and repeated access of memory. Furthermore, to speedup the interpolation process, parallelism is exploited at both coarse-grained and fine-grained levels that enable simultaneous calculation of multiple predicted samples with the help of SIMD instructions. Experimental analysis indicate that the suggested technique significantly reduced the computational complexity of the motion compensation process without loss in video quality.

The rest of chapter is organized as follows: Section 5.2 presents the proposed technique for the motion compensation. Experimental results are described in Section 5.3. Summary of this chapter is presented in Section 5.4.

5.2 Proposed Scheme

As discussed in section 3.2.1, the major factors contributing to the computational complexity of the reference software implementation of motion compensation algorithm include clipping of unrestricted motion vectors, looping overhead, repeated access of motion vectors, reference frame indices, prediction list utilization flags and pixel data from memory, and sequential calculation of predicted sample values. In the proposed scheme, these factors in the reference implementation of motion compensation algorithm are handled in three different stages. The first stage belongs to edge creation around the reference frame that helps to prevent the clipping of unrestricted. In the second stage, macroblocks are separated on the basis of inter prediction mode information (macroblock or sub-macroblock partitioning) to reduce the looping overhead and repeated access of memory. Finally, the SIMD optimization is performed for the simultaneous calculation of multiple predicted samples. The first two stages of the proposed scheme reduce the performance degradation in data manipulation module and the last one simplifies the linear interpolation module of motion compensation. The block diagram of the proposed scheme is shown in Fig. 5.1.

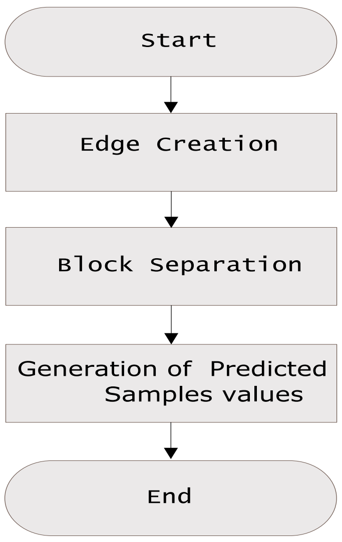


FIGURE 5.1: Block Diagram of the Proposed Scheme

5.2.1 Edge Creation

In the reference implementation, the clipping process has to be performed before reading of each pixel value from the memory. The basic purpose of clipping operation is to cater for unrestricted motion vectors that cause accessing illegal memory location. Generally, the edge pixel values are indicated by the unrestricted motion vectors. So, the clipping operations enforce the reading of edge pixel values corresponds to unrestricted motion vectors. Such a clipping process contributes significantly to the computational complexity of motion compensation module.

For the motion compensation of single frame of CIF resolution (352×288) having only full pel cases the clipping operation on the average would have to be performed 202240 times taking into account intra macroblocks in inter frame. This clipping count further increases if the the frame has sub pixel prediction cases.

To prevent the clipping overhead, edges are created around the reference frame by extending the border pixels up to 16 pixels (i.e. the size of macroblock). The left and right edges are created by extrapolating the first and last column of the reference frame. Similarly, the upper and lower edges are created by extrapolating the first and last row of the reference frame. The reason behind the 16 pixel extension is that the motion vectors do not indicate outside the reference frame more than one macroblock. In the proposed scheme integer pointer is used to access frame data from memory which reduces access time. For accessing memory through integer pointer, data should be byte aligned. The edge extension up to one macroblock also maintains the byte alignment of data.

For performing edge extension of CIF size frame (352×288), a total of 7680 copy operations are required. While in the case of clipping, even for full pel case, the clipping needs to be performed 202240 times. So the edge extension operation is about 3.8% of the clipping operation. The edge extension along with the use of integer pointer for data accessing from memory minimizes the computationally complexity caused by clipping operation.

5.2.2 Blocks Separation

The H.264/AVC reference implementation performs motion compensation at 4×4 block level for all maroblock and sub-macroblock partitioning. The data manipulation sub-module reads the motion vectors, reference frame indices and prediction list utilization flags for the target block. After that a block from reference frame is copied into the interpolation sub-module. Such a processing of macoblocks at 4×4 block level without taking into account the macoroblock inter prediction mode causes a repeated access of same motion vectors, reference frame indices, prediction list utilization flags and same pixel data from memory. These repeated accesses of same data from memory contribute a lot in the computational complexity of data manipulation sub-module. Macroblock and sub-macroblock partitions are shown in Fig. 5.2.

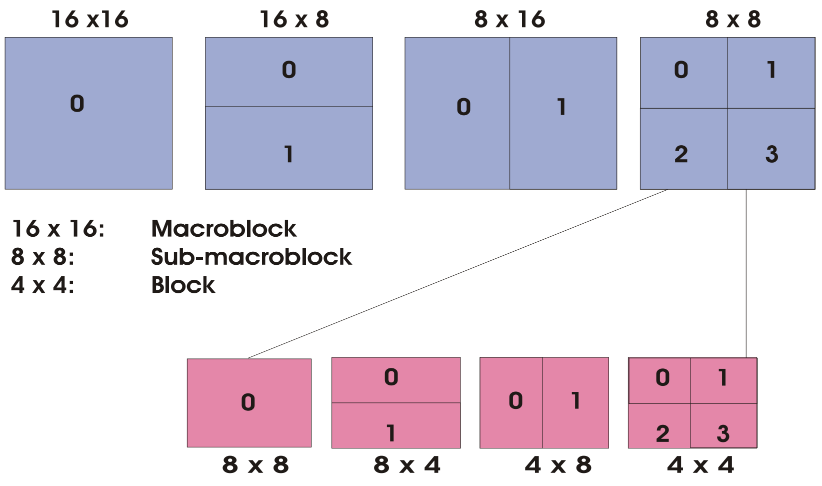


FIGURE 5.2: Macroblock and Sub-Macroblock Partitioning

Each partition corresponds to a particular macroblock mode which is given in Table 5.1.

TABLE 5.1: Mode Information of Macroblock

Mode	**Macroblock and sub-macroblock Partitioning**
0	16×16
1	16×16
2	16×8
3	8×16
4	8×8
5	8×4
6	4×8
7	4×4

In the proposed methodology, data manipulation over head is reduced through separtion of macroblcokos on the basis of prediction mode. Fig. 5.3. shows the block diagram of this stage. This separation of macroblocks leads to a significant gain in reading of motion vectors, reference frame indices and prediction list utilization flags. These gains in terms of number of readings saved per macroblock are listed in Table 5.2 for different macroblock and sub-macroblock partitioning.

It also helps in reducing the processing load by avoiding repeated access of same pixel values from memory.

Table 5.2: Memory Reading Saved Per MB

Mode	Reading Per MB		Gain per MB
	JM Ref. algorithm	Proposed algorithm	
0	16+16+16	1+1+1	15+15+15
1	16+16+16	1+1+1	15+15+15
2	16+16+16	2+2+2	14+14+14
3	16+16+16	2+2+2	14+14+14
4	16+16+16	4+4+4	12+12+12
5	16+16+16	8+8+8	8+8+8
6	16+16+16	8+8+8	8+8+8
7	16+16+16	16+16+16	0 +0+0

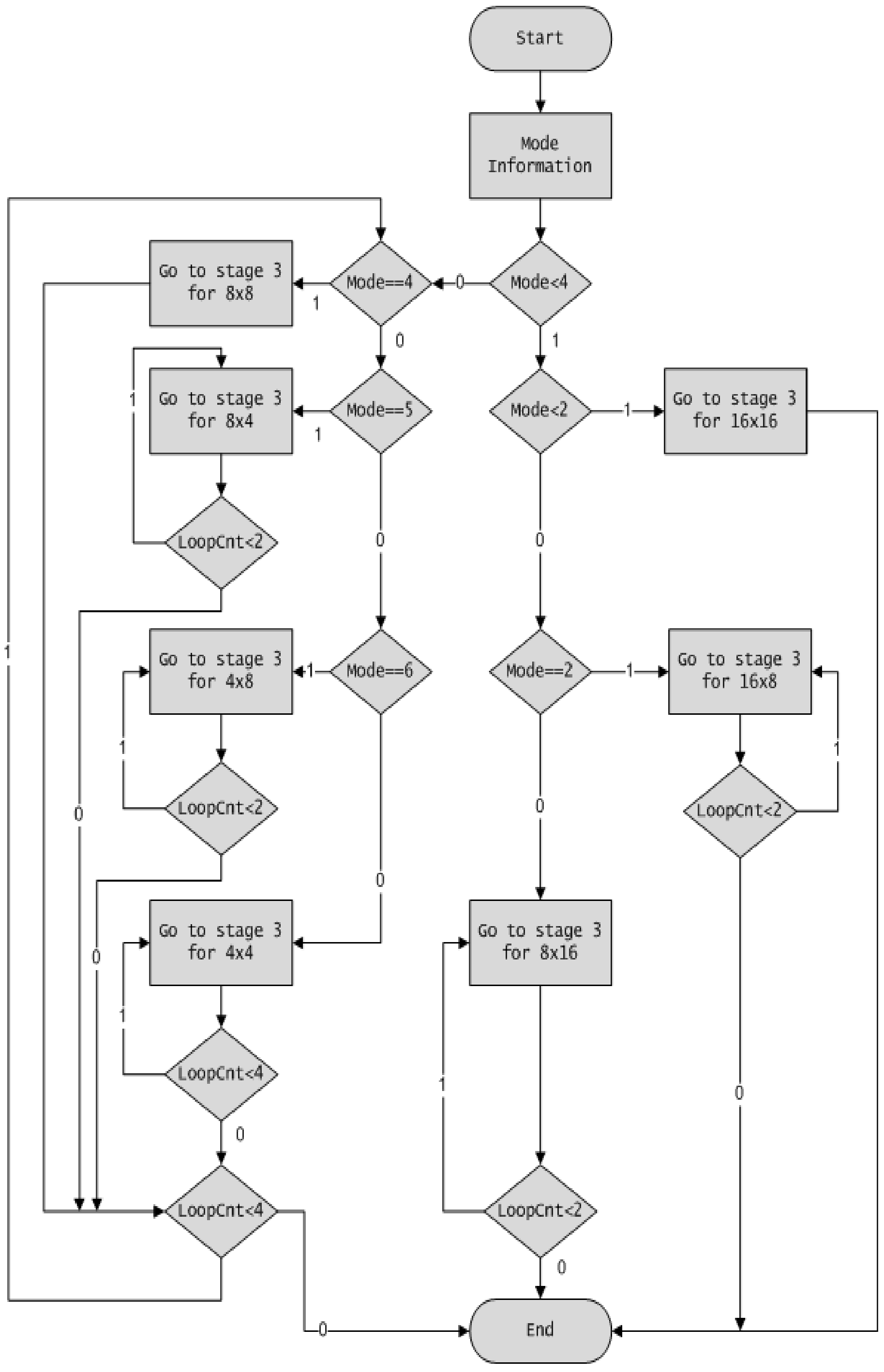


FIGURE 5.3: Blocks Separation

5.2.3 Generation of Predicted Sample Values

The H.264/AVC standard allows motion estimation for luma component at half-pixel and quarter-pixel level. The predicted macroblock against fractional-pel motion vector is generated through interpolation of reference frame. The interpolation module generates the predicted blocks for each motion vector then arranges all the predicted blocks to form predicted macroblock. For luma component, a six tap interpolation filter [1 -5 20 20 -5 1]/32 is employed to compute a predicted macroblock corresponds to half-sample position motion vector. The averaging of two half-sample position values gives the quarter-pel position value. On the other hand, a bilinear filter is employed to predict the chrominance component of macroblock pointing by sub sampled motion vectors.

In the reference implementation, the same pixel values are accessed repeatedly from memory. The calculation of half-pel value required six full-pel pixel values. For calculating two neighboring half-pixel values, five out of six values can be reused. However, these pixels are read again from the memory. The repeated access of memory places heavy load on the processors. Moreover, the generation of predicted macroblock in case of half-pel and quarter-pel involves computationally intensive arithmetic calculations. The interpolation process can be optimized through exploitation of parallelism and reducing the repeated access of same memory.

At this stage of the proposed scheme, the predicted macroblock for each inter prediction mode is generated based on motion vectors separately. This separation of macroblocks based on the prediction mode reduces the looping overhead and repeated memory access of same data from memory. Fig. 5.4. shows the block diagram of this stage.

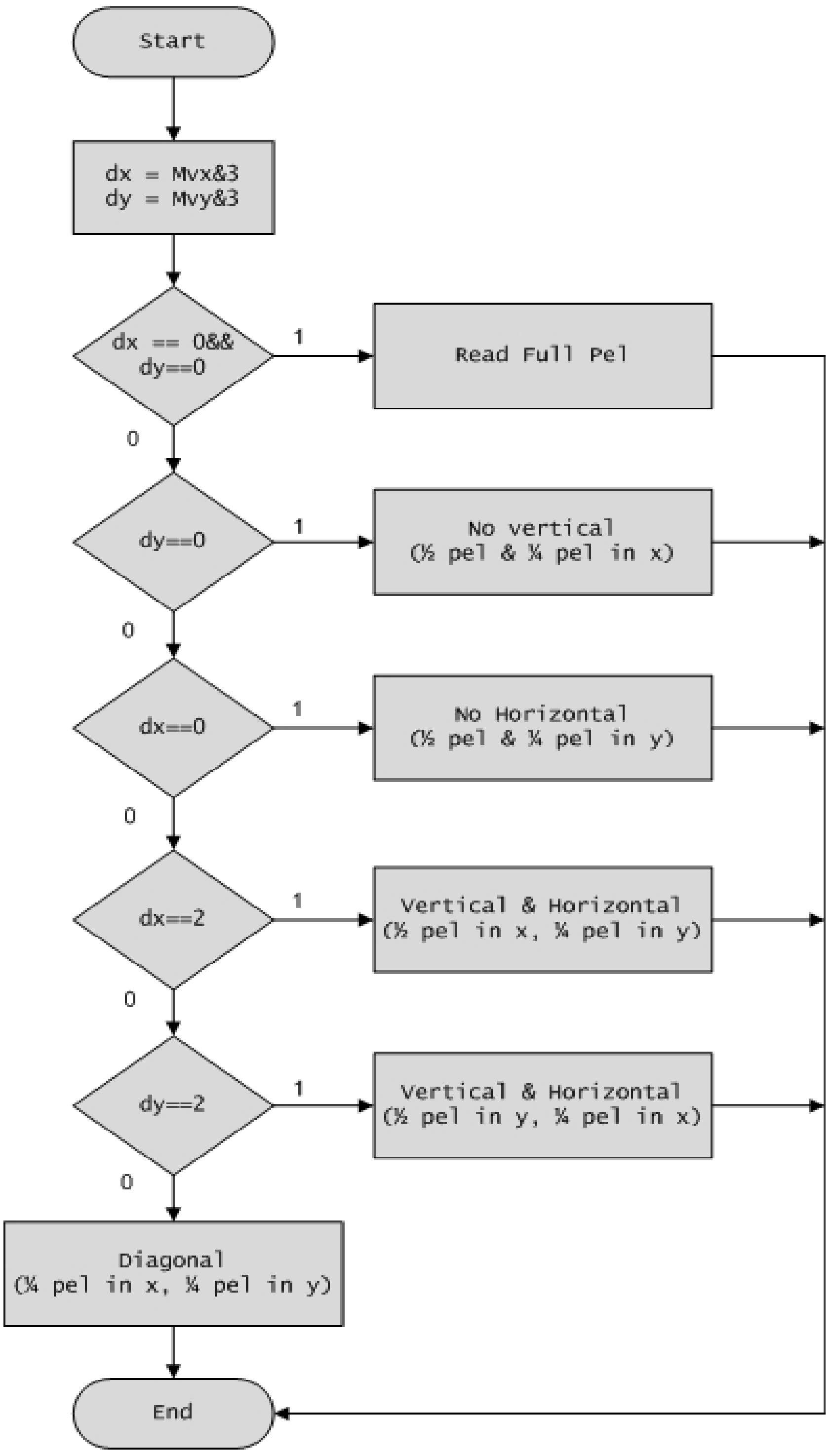
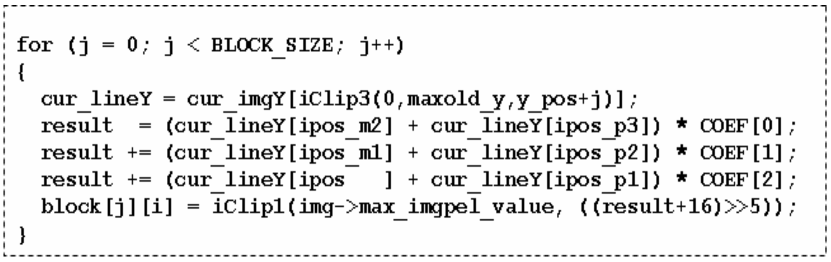


FIGURE 5.4: Generation of Predicted Sample Values

In order to perform half-pel interpolation fine-grained method is applied for 6-tap filtering. While in quarter-pel interpolation coarse-grained methods is exploited to process 4×4 block data simultaneously (parallel) with the help of SIMD instructions instead of computing pixel by pixel (sequential). The data reading is also made faster through integer pointer.

For horizontal half-pel interpolation of 4×4 block, four pixels are loaded simultaneously using integer pointer. So it takes twelve load operations to read all the 48 pixels required for interpolation. This loading reduces the memory reading overhead by one fourth. Each filter coefficient is represented in 8 bits, so four coefficients are packed in 32 bit register. After that filtering is performed using SIMD instructions for multiplication, packing, addition and shifting operations. The Fig. 5.5. shows the reference implementation for half-pel horizontal interpolation and Fig. 5.6. describe the calculation of half-pel horizontal case using SIMD instructions (_dotpus4, _packl4, _sadd2, etc) of TMS6467 digital signal processor. In the case of horizontal quarter-pel interpolation, first half-pel interpolation is performed then averaging of full-pel and half-pel pixels gives quarter-pel location pixels. The averaging can also be performed through SIMD instruction. Similarly, vertical and diagonal half-pel and quarter-pel interpolation is performed. The SIMD operations can also be used to optimize bilinear interpolation in case of chroma component. Hence, the uses of SIMD instructions convert the sequential interpolation process to parallel.



```
for (j = 0; j < BLOCK_SIZE; j++)
{
  cur_lineY = cur_imgY[iClip3(0,maxold_y,y_pos+j)];
  result  = (cur_lineY[ipos_m2] + cur_lineY[ipos_p3]) * COEF[0];
  result += (cur_lineY[ipos_m1] + cur_lineY[ipos_p2]) * COEF[1];
  result += (cur_lineY[ipos   ] + cur_lineY[ipos_p1]) * COEF[2];
  block[j][i] = iClip1(img->max_imgpel_value, ((result+16)>>5));
}
```

Figure 5.5: JM Reference Implementation of Half-Pel Case for Horizontal Motion Compensation

```
{
  Y0=(_dotpus4(R0,CONS1) + _dotpus4(R1,CONS2));
  Y1=(_dotpus4(R0,CONS3) + _dotpus4(R1,CONS4));
  Y2=(_dotpus4(R0,CONS5) + _dotpus4(R1,CONS6));
  Y3=(_dotpus4(SHIFT_1(R2,R0),CONS7)+ _dotpus4(R1,CONS8));
  Z0=CLIP2(_shr2(_sadd2(_pack2(Y1,Y0),0x00100010),5),0x00FF00FF);
  Z1=CLIP2(_shr2(_sadd2(_pack2(Y3,Y2),0x00100010),5),0x00FF00FF);
  D0=_packl4(Z1,Z0);
  R2=SHIFT_2(R1,R0);
  if(dx2&1)R2=SHIFT_3(R1,R0);
}
```

Figure 5.6: Proposed Implementation of Half-Pel Case for Horizontal Motion Compensation

5.3 Experimental Results

The TM320DM6467 DaVinchi digital signal processor (DSP) is used for the evaluation of the proposed motion compensation technique. This processor operates at a clock rate of 594 MHz. Three test sequences namely, *Mobile*, *Soccer* and *Foreman* are used for evaluation. The *Mobile* and *Soccer* are in CIF size while the *Foreman* is in QCIF (176×1444) resolution. Furthermore, for impartial comparative analysis of the proposed scheme with some other existing implementations, the same set of inter prediction modes is allowed during generation of test sequences as used in those studies. The test sequences are generated using baseline profile configuration of H.264 encoder with first frame encoded as I while the remaining as P frames. For encoding *Mobile* two separate schemes are presented, the first one using all macorblock and sub-macroblock partitioning while the second one using only 16×16 macroblock sizes. The *Soccer* sequence is encoded using 4×4, 8×8, and 16×16 block sizes and to encode *Foreman* sequence all block sizes are used.

The experimental results depict that for all test streams in terms of consumed processing power (mega cycles) the proposed scheme performs uniformly and equally efficient. On the other hand, reference implementation consumes much more processing power (mega cycles) with lots of variations for streams with different macorblock and sub-macroblock partitioning. The Fig. 5.7 clearly depicts the stability and low computational complexity of the proposed technique. It can be concluded that the proposed scheme on the average takes around 5 to 6 times lesser processing power (mega cycles) as compared to JM reference implementation.

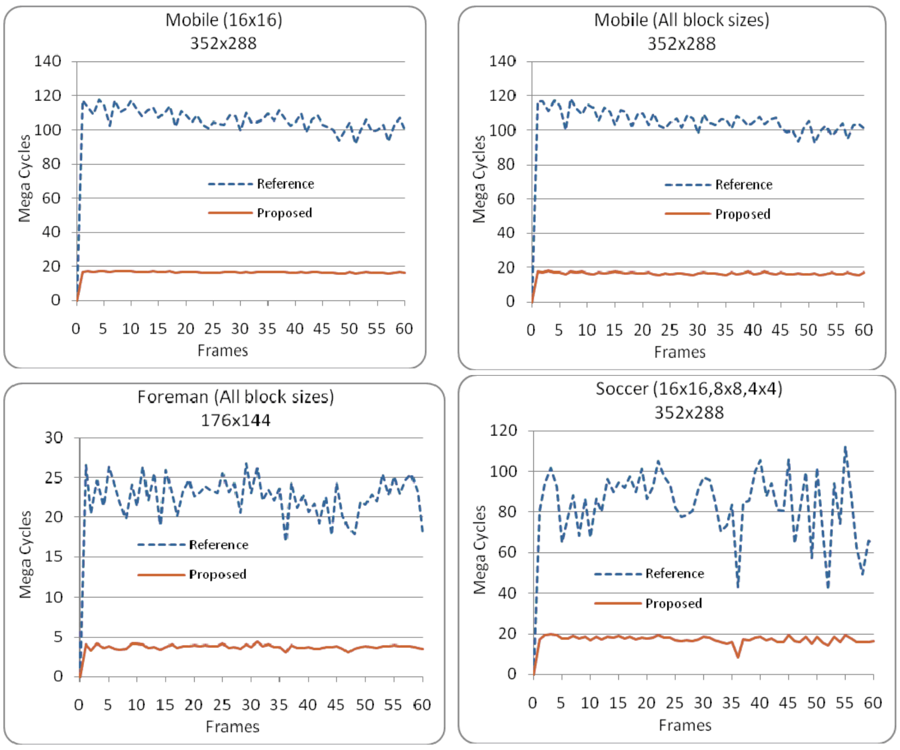


FIGURE 5.7: Comparison of the Proposed Scheme with JM Reference Implementation

The comparison of the proposed scheme with some existing techniques in terms of percentage gain as compared to reference implementation is given in Table 5.3. The proposed scheme achieves about 78.5% gain against reference implementation, which is significantly greater than all the existing techniques.

Table 5.3: Comparison of the Proposed Scheme with Other Schemes

Schemes	% Gain against Reference Implementation	Processor
Lee et al.'s [86]	41.18%	Pentium 4 MMX
Bhatia [87]	44.44%	Pentium 4 Processor
Sheshadri [88]	43.31%	ARM9TDMI
Khan et al [82]	67.49%	Trimedia PNX1300
Proposed	78.50%	Texas Instruments TMS320DM6467

5.4 Summary

The computationally efficient scheme for motion compensation is presented in this chapter. The presented scheme significantly curtails the computational complexities of the motion compensation algorithm related to data manipulation and linear interpolation by addressing the reasons of computational cost at different stage. Furthermore, parallelism is introduced in the proposed scheme during calculation of motion compensation values through SIMD instructions. The proposed scheme decreases the time complexities by 5 to 6 times by reducing looping overhead, repeated access of same pixel from memory, utilizing the memory resources and processing resources. A comparative analysis with some recent optimized implementation scheme shows that the proposed scheme outperforms other competing schemes and greatly enhances the efficiency of the motion compensation algorithm. The proposed scheme can be effectively used in real-time applications of H.264/AVC encoder and decoder.

The publish work related of this chapter :

Muhammad Asif, Masood Farooq and Imtiaz A. Taj, " *Optimized Implementation of Motion Compensation for H.264Decoder* ", 5th International Conference on Computer Sciences and Convergence Information Technology (ICCIT2010).

Chapter 6

PARALLEL TASK ASSIGNMENTS IN HYBRID HARDWARE-SOFTWARE VIDEO CODING

6.1 Introduction

In this era, portable multimedia devices with their corresponding media technologies play an important role in our daily life. These devices perform multimedia functionalities like video recording and playing, watching TV (mobile TV), video-telephony, and video streaming. Highly efficient video coding at lower power consumption is the fundamental requirement of these devices. The processors used in these devices provide specific hardware resources to handle computationally intensive video processing and interactive graphical applications. Moreover, processors designed for low-power applications may introduce limitations on the availability and usage of resources, which present additional challenges to the system designers.

Owing to the specific design of the JZ47x series of mobile application processors, this chapter presents an end-to-end hybrid software-hardware implementation scheme for H.264/AVC encoder. JZ47x is a series of low powered mobile application processors highly suitable for battery operated portable devices. These devices, for example, body worn video recorder, wireless video sensors, and video streamer, require long operation time. This series of processors introduces an innovative architecture to fulfill both high speed mobile computing and computationally intensive video coding requirements of portable multimedia devices. Particularly for H.264/AVC video coding, JZ47x processors (JZ4760, JZ4770, and

JZ4780) are equipped with dedicated hardware processing units of some encoder blocks. They have two major cores, that is, a Central Processing Unit (CPU) and a Video Processing Unit (VPU), where the VPU core controls the dedicated hardware processing units. They also have general-purpose direct memory access units (DMAs) for efficient data transfer and management during video coding. Although JZ47x architecture has the ability to meet the computational requirements of H.264/AVC encoder, it places certain limitations on the system designer as well. The efficient implementation of H.264/AVC encoder for the JZ47x processors is a challenging task because some of the encoder blocks are implemented in hardware and others need to be implemented in software. As H.264 encoder has an integrated framework, where task dependency is large, the encoder tasks distribution, scheduling, and synchronization among hardware and software modules are a real test of designer skills. Moreover, it demands efficient memory management and simultaneous exploitation of all processing units for achieving real-time encoding.

In this work, JZ4770 is taken as an example to implement the proposed scheme. The proposed scheme distributes the encoding tasks among hardware and software modules. The small size of on-chip memories and restrictions imposed by hardware processing units cause large data transferring among memories during the encoding process. To reduce this data transferring, several optimization techniques are proposed in this work. The hardware video processing units and DMAs need to be configured before each execution and take a prescribed amount of time to perform the assigned tasks. Because of inherent task and data dependencies in H.264/AVC encoder, the processor remains blocked during that time. This time is named as processing or polling time. It also causes the idling of hardware processing units. To reduce the polling and idling time of hardware processing modules, a pipelined design for encoder is proposed after exploiting fine grain macroblock (MB) level parallelism in which multiple macroblocks (MBs) are processed in parallel. Finally, a mechanism is developed to efficiently distribute the computational load among

CPU and VPU cores to increase the encoding rate.

The rest of the chapter is organized as follows. Section 6.2 describes the JZ4770 platform architecture. The proposed scheme is described in Section 6.3. Encoder performance is evaluated in Section 6.4. Finally, the summary is presented in Section 6.5.

6.2 JZ4770 Architecture

The JZ4770 is an asynchronous multiprocessor (AMP) architecture. CPU and VPU are its major cores. The CPU core contains main processor (named as J1) which operates at a clock rate of 1000 MHz. On the other hand, VPU core has Auxiliary Processor (AUX) which operates at a clock rate of 500 MHz. Both J1 and AUX are MIPS (Microprocessor without Interlocked Pipeline Stages). A master slave model exists between J1 and AUX. The operating system kernel uses J1 while the users can program the AUX. J1 loads binary on AUX and commands it to start execution.

The video processing units for H.264 encoding are vector matrix arithmetic unit (VMAU), motion compensation and estimation engine (MCE), and de-block (DBLK). There are three types of on-chip memories including two tightly coupled shared memories TCSM0 (16 KB) and TCSM1 (48 KB) along with SRAM (scratch RAM 28 KB). Furthermore, DMA0, DMA1, and DMA2 are three general-purpose DMAs coupled with TCSM0, TCSM1, and SRAM, respectively [111],[112]. These DMAs support three different data transfer types, that is, byte, short, and integer.

The hardware video processing units and DMAs need to be configured before each execution and do not support multiple instances. They take a prescribed amount of time to perform the assigned tasks.The constraints of these hardware processing units are that they operate at MB level and take input/output data from/to on-chip memory. But the reference frames used by MCE module are allocated on

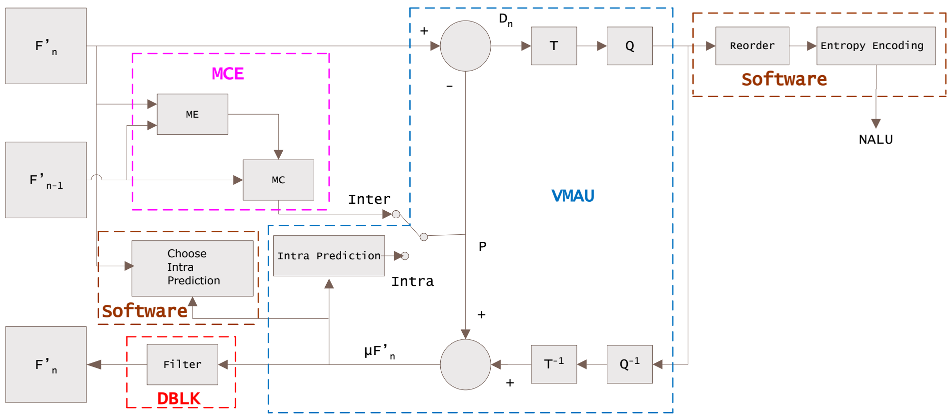


FIGURE 6.1: Hardware Software Co-Design for H.264/AVC Encoder

main memory. The current and reference frames are allocated on main memory because of limited size of on-chip memories.

6.3 Proposed Scheme

6.3.1 H.264/AVC Porting to JZ4770

The sequential software (C code implementation) based x264 encoder is ported on JZ4770 platform. In this work, baseline profile of encoder is used due to its low complexity as compared to the main and extended profiles. To distribute the encoding tasks among software and hardware processing units, x264 encoder is transformed into a highly modular and flexible structure. In this structured encoder, all the functional modules are independent software modules. It allows an easy replacement of the software modules with their corresponding hardware processing units.

As shown in Fig.6.1, hardware processing units replace the demarcated functionalities. MCE module replaces motion estimation and compensation. The VMAU module replaces the software functions that perform intra prediction, residual calculation, DCT/IDCT, Q/IQ, and MB reconstruction. The DBLK module is used to remove de-blocking artifacts from reconstructed frame. The software modules include intra prediction mode selection, reordering, Context-Adaptive Variable-Length Coding (CAVLC), edge extension, bit-rate control, file reading or video capturing, and file writing. All these software modules execute main processor. The main processor is also responsible for the configuration of hardware modules before each execution.

6.3.2 H.264/AVC Encoder Data Dependencies and Profiling Analysis

Before optimization, parallel processing, and tasks mapping to speedup the H.264/AVC video encoder, an extensive analysis is required. This analysis encompasses computational requirements of various functional elements of the encoder and data dependencies among them.

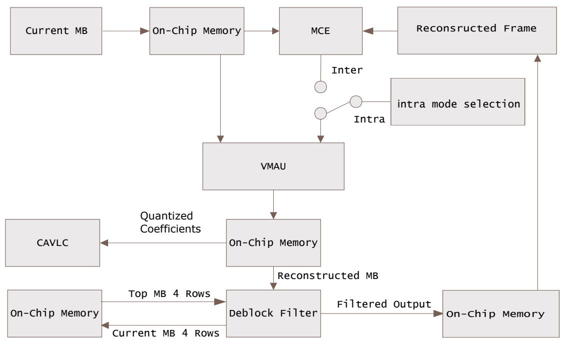


FIGURE 6.2: MB Level Processing Flow of Hybrid Encoder

In H.264/AVC encoding, MB is a basic processing unit of a frame. For encoding the frame, raw data of each MB is transferred from main memory to on-chip memory. The encoding mode of MB is selected either intra or inter. For intra mode, the inputs to VMAU unit are raw MB data and intra prediction mode. For inter mode, the inputs are raw MB data and motion compensated data. In intra encoding, the suitable intra prediction mode selection is done through software module. The VMAU is configured for intra mode. The outputs of VMAU unit are quantized coefficients, coded block pattern (CBP), and reconstructed MB. Entropy coding encodes quantized coefficients and CBP, while the DBLK unit removes the blocking artifacts of reconstructed MB. The output of DBLK unit is transferred from on-chip memory to reconstructed frame on main memory. In inter encoding; the MCE processing unit performs motion estimation. The outputs of MCE are motion vectors and motion compensated data. The VMAU is configured for inter mode. The rest of processing is similar to intra mode encoding.

Fig.6.2 indicates the sequential dependencies between functional modules of the encoder; that is, the output data of one module corresponds to the input data of another subsequent module. Because of these dependencies, one hardware processing unit is used at a time and all the remaining units are idle. To assure the compliance of the whole set of strict dependencies imposed by the encoder, it can be observed that the exploitation of fine-grained level parallelism may help to minimize the idling time of hardware processing units.

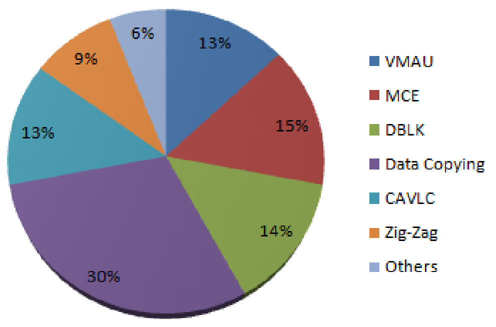


FIGURE 6.3: Time Breakdown of Hybrid Encoder

Fig. 6.3 represents the breakdown of hybrid H.264/AVC encoder processing time with respect to the various functional modules. In order to evaluate the computational complexity of the encoder, NTSC resolution test video sequences including *Mobile*, *Football*, *Intros*, *Garden*, *Galleon*, *Vtc1nw*, and *Washdc* are encoded. The encoding parameters are set as follows: MV search range is 32 to +32 pels and block size is 16×16, RD optimization is disabled, one reference frame is for motion estimation and compensation, motion estimation scheme is diamond search, MV resolution is 1/4 pel, GOP is chosen as 25 with structure IPPP, and four quantization parameters (QPs) are 24, 28, 32, and 36.

Fig. 6.3 shows that data copying/transferring among memories is a dominant task taking about 30% of the total processing time. This data transferring is necessary because of the restrictions imposed by the hardware processing units and the limited sizes of on-chip memories. On the other hand, VMAU, MCE, and DBLK take significant portion of the overall processing time.The time taken by these hardware modules is their configuration and processing time. The CAVLC, Zigzag scanning, and others take the rest of the processing time.

Considering the above analysis, there is a need to exploit the parallelism in encoder for efficient utilization of hardware resources by reducing their idling and polling time. Moreover, efficient memory management is required to reduce the data transferring and copying operations during the encoding process.

6.3.3 Optimization Techniques for Memory Management and Data Transfer

From the profiling analysis presented in section 6.3.2, it is clear that data transferring and accessing among memories is the most computationally intensive task. This section presents numerous optimization techniques to reduce the data transferring time and to increase data reuse ratio.

6.3.3.1 Memory Efficient Design for DBLK Module

The de-blocking filtering of an MB needs pixel data from neighboring MBs, i.e., top MB bottom edge (last four rows (4×16)) and left MB right edge (last four columns (16×4)). The memory constraint of the hardware DBLK unit is that the current MB and left MB right edge data must be adjacent in on-chip memory for both input and output as shown in Fig.6.4.

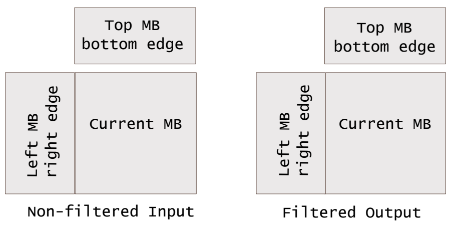


Figure 6.4: Input and Output Data Placement of DBLK Module

The outputs of DBLK unit are reconstructed filtered pixel data of current MB (16×16), top MB bottom edge (4×16), left MB right edge (16×4), and non filtered current MB bottom edge (16×4). The pixels data of top MB bottom edge can be placed anywhere in on-chip memory. The left MB right edge and top MB bottom edge is usually copied from reconstructed frame on main memory to on-chip memory. Similarly, the output generated by DBLK module is copied from on-chip memory to main memory. This data transferring is necessary because reconstructed frame is stored on the main memory due to limited size of on-chip memory. In order to reduce these data copying operations, an efficient memory design for DBLK module is proposed as shown in Fig.6.5.

According to this design, the output of VMAU module is taken as the input to the DBLK module. This avoids data coping of current MB from main memory to on-chip memory. Moreover, the output of the DBLK module is stored adjacent to its input (current MB reconstructed pixel data from VMAU module) in such a way that the right edge of output overlaps with the left edge of its input as shown in Fig.6.5. This eliminates the need of copying the right edge of left MB, required in next iteration. The non-filtered current MB bottom edge, output of DBLK is stored in on-chip memory in such a way that the bottom edge of complete MB row of a frame is stored in contiguous memory. In this way, it can be reused as top MB bottom edge non-filtered input of DBLK module for next MB row processing. Finally, only the filtered output is copied in reconstructed frame. The boundary cases, i.e., top MB row, left MB column and bottom MB row, have been handled

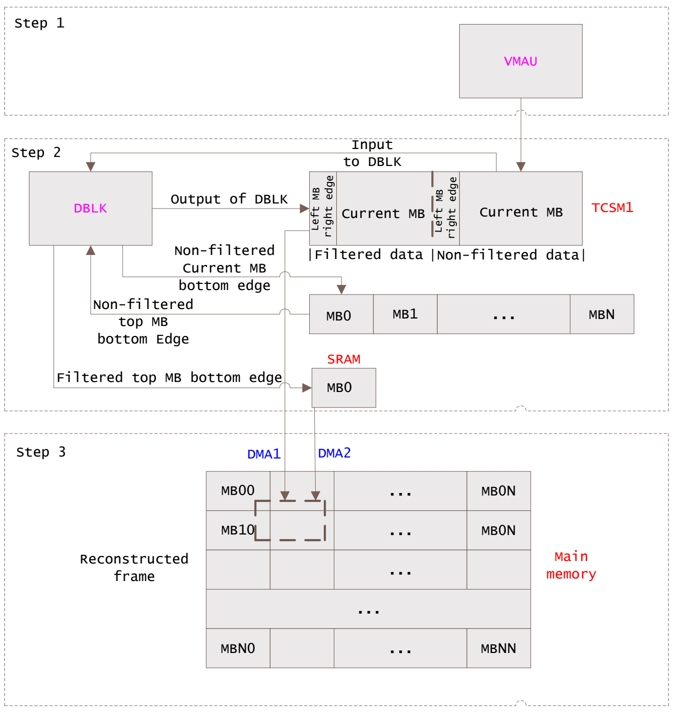


FIGURE 6.5: Design for DBLK Module

as per requirement. This design resulted in increasing the data re-usability and avoiding the excessive data copying operations.

6.3.3.2 Efficient Data Transfer

As mentioned in section 6.2, hardware video processing units take input from and store output data to on-chip memories. Because of the limited size of on-chip memories, input and reconstructed frames are stored on main memory. For processing a frame, each MB data is copied to the on-chip memory. Similarly, the reconstructed MB needs to be transferred from on-chip memory to main memory. This requires multiple data transfer operations in which memory copy routines do involve CPU. This is a time consuming task as the CPU cannot execute any other operation during this time. The DMA is a handy tool to carry out these transfers without invoking the CPU.

In the proposed scheme, DMAs are used to speedup the data transferring among memories. Moreover, on-chip memory allocation for current and corresponding reconstructed MB is done in such a way that all DMAs (DMA0, DMA1 and DMA2) can be performed in parallel for transferring data among memories. The memory for current MB is allocated on TCSM0 while its corresponding reconstructed filtered pixel data and left MB right edge is stored on TCSM1. The SRAM is used for storing reconstructed filtered pixels data of top MB bottom edge. Such memory allocation helps in performing all DMAs in parallel to speedup the data transferring. DMA0 is responsible for transferring current MB data from main memory to on-chip memory TCSM0 for encoding. The reconstructed MB is transferred back from on-chip memories TCSM1 and SRAM to main memory through DMA1 and DMA2, respectively. The data transfer size of DMA0 for an MB is 384 bytes. The DMA1 and DMA2 transfer 288 bytes and 96 bytes of data in each execution, respectively. For an MB encoding, 384 bytes of data is transferred from main memory to on-chip memory and vice versa. The DMAs incorporation in the design speeds up the data transferring during the encoding process.

6.3.3.3 Algorithmic Optimization using SIMD Instructions

Zig-Zag scanning and edge extension algorithms belong to the software part of hybrid encoder because their support is not available in hardware. These algorithms involve multiple memory access and data copying operations that significantly contribute in computational cost. This section describes the optimization of these algorithms using SIMD instructions to speedup the encoding. The SIMD instructions enable multiple arithmetic or logic operations to be executed simultaneously. Ingenic processor JZ4770 provides 60 SIMD instructions for the multimedia codec optimization [113]. These instructions operate on 32 bits registers.

Zig-Zag scanning is performed on quantized coefficients before entropy encoding. The 16 bits quantized coefficients are accessed from on-chip memory. In this work, 32 bits load and store operations are used to make quantized coefficients loading and storing faster.The SIMD instructions are also used to arrange two coefficients

```
(a)
// Optimized Implementation for Zig-Zag Scanning
 {
/*
  For Input

  1 2 3 4
  5 6 7 8
  9 10 11 12
  13 14 15 16

 Output  after Zig-Zag scannig is

 1 2  5 9  6 3   4 7  10 13   14 11   8 12    15 16
*/

 /* load quantized cefficents to MXU registers */

  S32LDD(xr1,dct,0x0);    // xr1 =  2 1
  S32LDD(xr2,dct,0x4);    // xr2 = 4 3
  S32LDD(xr3,dct,0x8);    //  xr3 = 6 5
  S32LDD(xr4,dct,0xc);    // xr4  =8  7
  S32LDD(xr5,dct,0x10);   // xr5 =10 9
  S32LDD(xr6,dct,0x14);   // xr6 =12 11
  S32LDD(xr7,dct,0x18);   // xr7 =14 13
  S32LDD(xr8,dct,0x1c);   // xr8 = 16 15

 /* adjust positions  of coefficients according to Zig-Zag order  */

  S32SFL(xr9,xr6, xr4,xr10,3);//xr6 =12 11 xr4=8 7  xr9=12 8  xr10=11 7
  S32SFL(xr4,xr10,xr7,xr6, 3);//xr10=11 7 xr7=14 13 xr4=11 14 xr6=7 13
  S32SFL(xr7,xr6, xr2,xr10,3);// xr6=7 13  xr2=4  3  xr7=7 4 xr10=13 3
  S32SFL(xr6,xr10,xr5,xr2, 3);//xr10=13 3 xr5= 10 9 xr6= 13 10 xr2= 3 9
  S32SFL(xr5,xr2, xr3,xr10,3);//xr2=3 9  xr6= 6  5   xr5=3     6  xr10= 9 5

 /* reordered coefficients are stored to main memory from  MXU registers */

  S32STD(xr1, level,0x0);     // xr1= 2 1
  S32STD(xr10,level,0x4);    // xr10= 9 5
  S32STD(xr5, level,0x8);    //  xr5=3 6
  S32STD(xr7, level,0xc);    // xr7=7 4
  S32STD(xr6, level,0x10);  // xr6=13 10
  S32STD(xr4, level,0x14); // xr4=11 14
  S32STD(xr9, level,0x18); // xr9= 12 8
  S32STD(xr8, level,0x1c); // xr8=16 15

(b)
// Optimized Implementation for top edge extension of luma component
  for (j=0;j<(Lwidth/16);j++)
  {
     S32LDD(xr1,ExtndDataUP_Inp,0x0); // loading data
     S32LDD(xr2,ExtndDataUP_Inp,0x4);
     S32LDD(xr3,ExtndDataUP_Inp,0x8);
     S32LDD(xr4,ExtndDataUP_Inp,0xc);

     for (i=0;i<16;i++)
      {
        S32STD(xr1, ExtndDataUP_Out,0x0); // edge  extension
        S32STD(xr2, ExtndDataUP_Out,0x4);
        S32STD(xr3, ExtndDataUP_Out,0x8);
        S32STD(xr4, ExtndDataUP_Out,0xc);
        ExtndDataUP_Out+=4;
      }
      ExtndDataUP_Inp+=64;
  }
```

Figure 6.6: (a) SIMD Implementation of Zig-Zag Scanning (b) SIMD Implementation of Edge Extension

simultaneously instead of one. Fig.6.6(a) shows the Zig-Zag scanning algorithm optimization using SIMD instructions (S32LDD, S32SFL and S32STD). In which quantized coefficients are loaded from on-chip memory to dedicated MXU registers (xr1-xr8) using S32LDD instruction. After that, S32SFL is performed on these registers in pairs to adjust the positions of coefficients according to Zig-Zag order. Then final reordered coefficients are stored to main memory from dedicated MXU registers (xr1, xr10, xr5, xr7, xr6, xr4, xr9 and xr8) using S32STD instruction. The S32SFL(r1, r2, r3, r4, 3) is packing instruction. This instruction packs the two most-significant halfwords (16 bits) from the arguments r2 and r3 into r1. The halfword from r2 is packed into the most-significant halfword of r1; the halfword from r3 is packed into the least-significant halfword of r1. Similarly, it also pack two least-significant halfwords from the arguments r2 and r3 into r4.

For performing motion estimation, the reference frame is padded all around. This edge extension may be several pixels wide, depending on the motion estimation mode. The upper and lower edges are generated by copying the first and last row of the frame into the border or edges. Similarly, left and right edges are generated by copying the first and last column of the frame into the border. In this work, 32 bits loading and storing operations are used instead of 8 bits operations. The use of SIMD instructions help to extend four pixels in parallel. Fig.6.6(b) shows the top edge extension using SIMD instructions (S32LDD and S32STD).

Table 6.1 lists the achieved speedup gain through each optimization technique.

TABLE 6.1: % Gain Through Optimization Techniques

Encoder Modules	% Gain
Data Transfer	27%
DBLK Design	32%
Edge extension and Zig-Zag	35%

6.3.4 Encoder Pipeline Design

Considering the data dependency analysis made in section 6.3.2, it is observed that the heterogeneous structure of the H.264/AVC encoder have inherent sequential modular data dependency i.e., output of one module becomes input of its subsequent module. Because of that modular dependency one hardware processing unit is used at a time and all of the other units remain idle. Moreover, the polling of hardware processing units and DMAs take significant part of encoding time.

These factors not only decreases the throughput of hardware processing unit but also has adverse affect on encoder performance. For efficient utilization of all hardware processing units and reducing polling time, a pipeline design for H.264 encoder has been proposed after exploiting fine-grained parallelism at MB level. Exploiting MB level parallelism requires satisfying the data dependencies between MBs. For a given MB, intra prediction, inter prediction, de-block filtering and entropy encoding need data from neighboring (left, top, top left and top right) MBs. This data include pixel data, prediction mode, motion vectors, number of non zero coefficients and etc. The simplest way to resolve all these dependencies is to process MBs in scan order.

In proposed design, an MB encoding process is divided into several stages and each stage is executed on a separate processing core. After that these processing cores are pipelined at MB level through which multiple MBs are processed simultaneously in scan order. For processing intra and inter MB, a pipeline design is described below.

Intra MBs: The encoding task for intra MB is performed in four major pipeline stages. These stages are DMA0 stage (transfers raw MB data from main memory to on-chip memory), VMAU stage, DBLK+CAVLC stage (this stage performs de-block filtering and entropy encoding in a same time slice) and DMA1 stage (transfers filtered reconstructed MB data from on-chip memory to main memory). Each intra MB goes through the final DMA1 stage after four MBs cycle time latency. This four stage pipeline for intra mode is shown in Fig.6.7.

Inter MBs: The encoding task for inter MB is performed in five major pipeline stages. These stages are DMA0 stage, MCE stage (perform motion estimation), VMAU stage, DBLK+CAVLC stage and DMA1 stage. The MB0 goes through

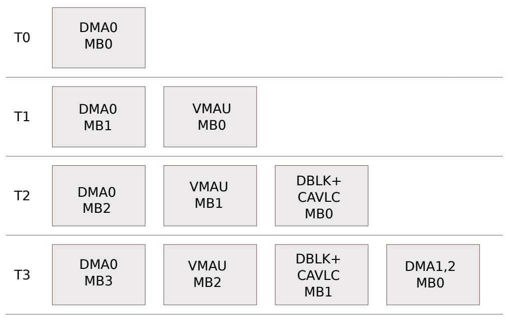


FIGURE 6.7: Four Staged Pipeline for Intra MB's

the DMA1 stage after five MBs cycle time latency. This five stage pipeline for inter mode is shown in Fig.6.8.

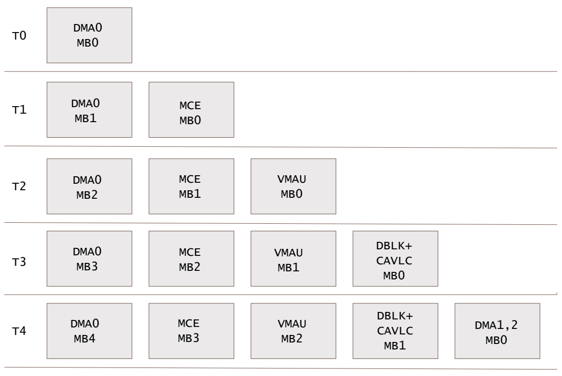


FIGURE 6.8: Five Staged Pipeline for Inter MB's

Fig.6.9 presents the pseudo-codes of proposed encoder pipelined design. In which hardware modules for processing current MB are configured before polling them for previous MB. The usage of hardware modules in this way reduce the polling time.

To show the advantage of the proposed pipelined design, profiling analysis is carried out before and after exploiting the MB level parallelism.The time taken by each module in hybrid encoder before pipelined design is shown in Fig.6.10. Table 6.2 shows that the polling and configuration of hardware video processing units take 51% and 20% of total processing time, respectively.

```
while(mbno < total_mbs+n)
 {        //where n=no. of pipeline stages - 1
// DMA0 Stage
    If(mbno < total_mbs)
         Configure DAM0 ()  // for current MB
         If(mbno>0)
              Poll DMA0()     // for previous MB
         End If
         exeute DMA0()     // for current MB
         If( last MB)
             Poll DMA0()  // for current MB
         End If
    End If
// MCE Stage
    If(slice is P_TYPE)
         mbno_mce = mbno-1;
         If(mbno_mce>=0 && mbno_mce<total_mbs)
              If(mbno_mce>0)
                   If( current MB is inter type)
                      Configure MCE () // for current MB
                    End If
                   If( prvious MB is inter type)
                      poll MCE()   // for previous MB
                    End If
              End If
              If( current MB is inter type)
                   exeute MCE()      // for current MB
                  If( last MB)
                     Poll MCE() // for current MB
                  End If
              End If
         End If
    End If
// VMAU Stage
    mbno_vmau=mbno-2;
    If(mbno_vmau>=0 && mbno_vmau<total_mbs)
         Configur VMAU()      // for current MB
         If(mbno_vmau > 0 )
             poll VMAU()              // for previous MB
         End If
         exeute VMAU()      // for current MB
         If(last MB)
            poll VMAU()       // for current MB
         End If
    End If
// DBLK Stage
    mbno_dblk=mbno-3;
    If(mbno_dblk>=0 && mbno_dblk<total_mbs)
        Configur DBLK()       // for current MB
        If(mbno_dblk>0)
             Poll DBLK()              // for previous MB
        End If
        exeute DBLK()      // for current MB
        Entropy encoding ()     // for current MB
        If(last MB)
           poll DBLK()    // for current MB
        Endif
    End If
// DMA12 Stage
   mbno_dma12=mbno-4;
   If(mbno_dma12>=0 && mbno_dma12<total_mbs)
        Configur DMA12()    // for current MB
        If(mbno_dma12>0)
            Poll DMA12()          // for previous MB
        End If
        exeute DMA12()      // for current MB
        If(last MB)
            poll DMA12()     // for current MB
        End If
   End If
   mbno++;
}
```

Figure 6.9: Pseudo-Code of Proposed Encoder Pipelined Design

Table 6.2: % of Total Processing Time Taken by HW Processing Units Before Pipelined Design

HW Processing Units	Configuration Time	Polling Time	Total Time
VMAU	2%	12%	14%
MCE	1%	13%	14%
DBLK	7%	12%	19%
DMA0	3%	7%	10%
DMA1,2	7%	7%	14%
Total	20%	51%	71%

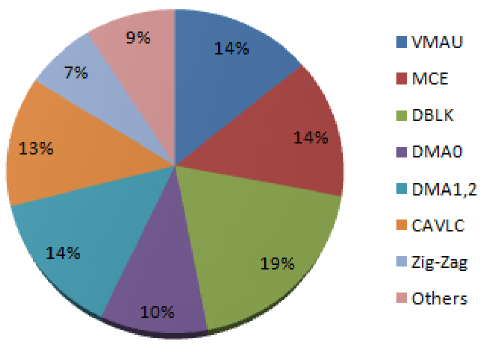


FIGURE 6.10: Time Breakdown before Pipelined Design

After pipelined design, the distribution of processing time among different encoding blocks is shown in Fig.6.11. Table 6.3 shows that polling time reduces from 51% to 6%.

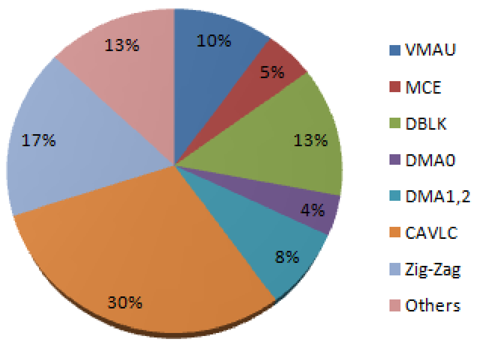


FIGURE 6.11: Time Breakdown after Pipelined Design

TABLE 6.3: % of Total Processing Time Taken by HW Processing Units After Pipelined Design

HW Processing Units	Configuration Time	Polling Time	Total Time
VMAU	9%	1%	10%
MCE	3%	2%	5%
DBLK	12%	1%	13%
DMA0	3%	1%	4%
DMA1,2	7%	1%	8%
Total	34%	6%	40%

6.3.5 Encoder Partitioning between CPU and VPU Cores

Even though the pipeline design significantly shortens the encoding time, the encoder overall performance still heavily depends on distribution of encoding tasks between CPU and VPU. The profiling analysis made after pipeline design demonstrates that hardware modules configuration and processing take 40% while CAVLC and Zig-Zag scanning consume 47% of the overall processing time. So, the potential advantage is to distribute these computationally demanding modules among CPU and VPU core. As in proposed pipelined design, entropy encoding (CAVLC and Zig-Zag) and de-block filtering (DBLK) modules both are at the same stage. There is no data dependency between them and they can be parallelized. In order to take the advantage of this parallelism encoder tasks are distributed between CPU and VPU cores.

Fig.6.12 shows the distribution of encoder between CPU and VPU cores. This distribution is made by considering the parallel processing, load balancing between the cores and scalability. The tasks of encoding performed by CPU core are the initialization of the encoder and hardware processing units, video capturing, frame level parameter setting and controlling, zig-zag scanning, motion vector prediction, entropy encoding, bit-rate control, edge extension and file writing. The VPU core is responsible for configuration and controlling of hardware modules, MB level scheduling of hardware processing units, encoding parameters selection and data transferring between main and on-chip memories. The encoder tasks distribution between CPU and VPU resulted in efficiently utilization of all parallelization capabilities of this platform for high-performance video encoding.

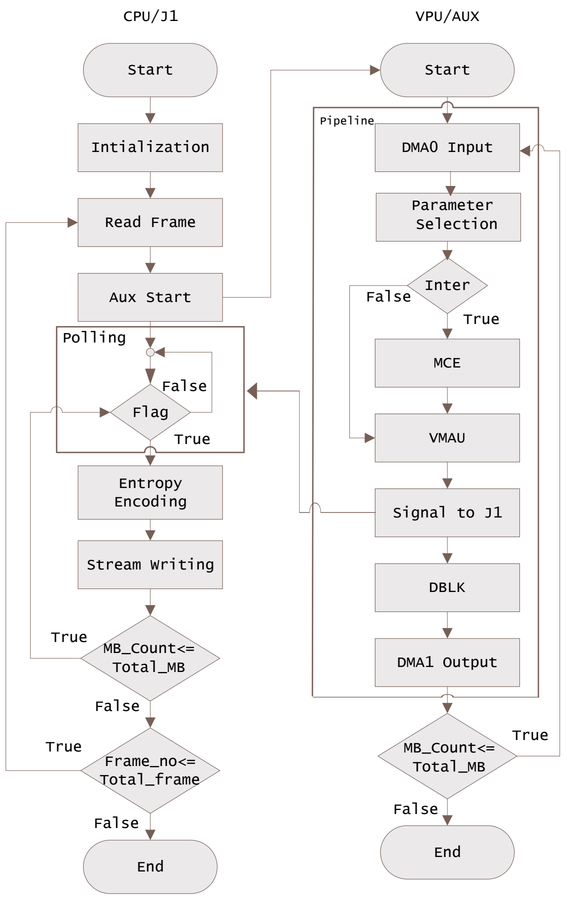


FIGURE 6.12: Proposed Encoder Flowchart

6.3.6 Scheduling Strategy and Encoder Flow

In the proposed design, CPU and VPU cores are working concurrently and they are pipelined at MB level. They used on-chip memories for exchanging data and/or messages. Ten buffers are allocated and used in a ping pong manner for sending and receiving frame and bit-stream data between the cores. Fig.6.12 shows the flowchart of the proposed H.264 encoder. The encoding process is described as.

1. The raw data of the frame is read and preprocessed by CPU core.
2. After CPU sets the frame level parameters, it starts the VPU and waits for a completion signal.

3. The VPU core processes MBs using hardware processing units as described in section 6.3.4.

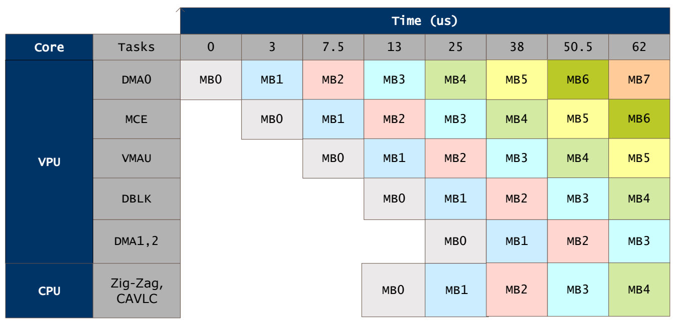
4. After completion, the VPU signals CPU and sends MB parameter structures. This structure contains motion vectors, residual quantized coefficients, CBP and other parameters required to encode.

5. The CPU performs entropy encoding, writes bit stream, control the rate and receive data structure from VPU.

6. The VPU core stops after all MBs in a frame have been processed.

7. The above procedure continues until the encoding of all candidate frames.

The time-line of proposed multi-core pipelined encoder is shown in Fig.6.13. It clearly demonstrates the exploitation of fine-grained MB level parallelism and all processing capabilities of JZ4770 platform. When pipeline is full five MBs are processed in parallel.



		Time (us)							
Core	Tasks	0	3	7.5	13	25	38	50.5	62
VPU	DMA0	MB0	MB1	MB2	MB3	MB4	MB5	MB6	MB7
	MCE		MB0	MB1	MB2	MB3	MB4	MB5	MB6
	VMAU			MB0	MB1	MB2	MB3	MB4	MB5
	DBLK				MB0	MB1	MB2	MB3	MB4
	DMA1,2					MB0	MB1	MB2	MB3
CPU	Zig-Zag, CAVLC				MB0	MB1	MB2	MB3	MB4

FIGURE 6.13: Time-Line for Multi-Core Pipelined Encoder

It is important to note that the flexible design of the proposed scheme allows the integration of any fast mode selection and motion estimation algorithms. The motion estimator can be easily integrated at the place of MCE module and mode selection algorithm can be incorporated in the parameter selection module. Moreover, in case of CABAC entropy encoding, CABAC takes the place of CAVLC.

6.4 Performance Evaluation and Discussion

To evaluate the performance of the proposed scheme two different frame resolutions, NTSC (720×480) and 720p HD (1280×720) are considered. All test sequences are in 4:2:0 format and having 300 frames each. The NTSC resolution test sequences are: *Mobile*, *Football*, *Intros*, *Garden*, *Galleon*, *Vtc1nw* and *Washdc*. *Stockholm*, *Shields*, *Parkrun* and *Mobcal* are considered HD 720p resolution test sequences. The video coding parameters are set as follows: MV search range is -32 to +32 pels and block size 16×16, RD optimization is enabled with metric sum of absolute difference (SAD), rate control method is single pass constant quantizer (QP Mode), one reference frame for motion estimation and compensation, motion estimation scheme is diamond search, MV resolution is 1/4 pel and GOP is chosen 25 with structure IPPP. The RD Optimization technique improves the RD performance but decreases the encoding speed. The use of computationally intensive Psycho-Visual Optimization metric improves the RD performance and reduces the encoding speed. On the other hand, simpler metrics like SAD and sum of absolute transform difference (SATD) increase the encoding speed at the cost of degradation in RD performance.

6.4.1 Encoder Performance at Different Implementation Stages

To demonstrate the advantages of proposed scheme, results are computed at three separate stages. The first stage is based on the x264 sequential software implementation [114] executing on CPU core. The second stage belongs to section 6.3.1 in which video processing units are incorporated. Finally, stage three introduces the numerous optimization methods, pipeline design and parallel processing. Table 6.4 and Table 6.5 depict the results at different implementation stages for NTSC and 720p resolutions, respectively.

Table 6.4: Experimental Results at Different Implementation Stages for NTSC Sequences (Qp= 24)

Sequences	Stage 1 (fps)	Stage 2 (fps)	Stage 3 (fps)
Garden	3.73	18.75	44.50
Football	3.65	19.74	53.86
Intros	4.02	24.90	64.62
Mobile	3.55	18.96	48.78
Galleon	4.89	24.50	65.99
Vtc1nw	6.70	28.27	69.38
Washdc	4.79	26.17	63.80
Average	4.48	23.04	58.70

Table 6.5: Experimental Results at Different Implementation Stages for 720p Sequences (Qp= 26)

Sequences	Stage 1 (fps)	Stage 2 (fps)	Stage 3 (fps)
Stockholm	1.53	9.30	28.73
Shields	1.47	8.85	27.68
Parkrun	1.33	7.76	18.99
Mobcal	1.52	8.32	25.91
Average	1.46	8.55	25.32

The average encoding rate for NTSC resolution at first stage is 4.48 fps. After the second stage, the encoding rate increased to 23 fps. Finally, the average encoding rate is increased up to 58.70 fps after the third stage. The results demonstrate that, the addition of hardware processing units increase the encoding rate from 4.48 fps to 23 fps and the proposed optimization techniques increase the encoding rate from 23 fps to 58.70 fps. Similarly, the encoding rate for 720p resolution is increased from 1.46 fps to 25.32 fps. In this case, encoding rate is increased from 1.46 fps to 8.55 fps due to addition of hardware processing units and from 8.55 fps to 25.32 fps because of the proposed optimization techniques. It can be concluded that, although the use of hardware processing units increased the encoding rate, major advantage is due to their efficient utilization through proposed optimization methodologies.

Table 6.6: Optimization Method vs Contribution to Encoding Speedup

Optimization Method	Speedup
Memory Management and Data Transfer	20%
Encoder Pipeline Design	34.29%
Encoder Partitioning between CPU and VPU Cores	45.71%

Table 6.6 shows the impact of each optimization method on the total encoding speedup. It demonstrates that the Encoder partitioning between CPU and VPU Cores contributes the most significantly i.e 45.71%, to speedup the encoding process. The reasons behind such significant encoding gain are parallel processing and reduction of hardware configuration time. As in the proposed pipeline design, entropy encoding and de-block filtering (DBLK) modules both are at the same pipeline stage. This stage takes most of the encoding time and become a bottleneck for the rest of the pipeline stages. After encoder partitioning, these two modules are parallelized at MB level and overall encoding load is distributed between two processing cores. This resulted in speedup of the encoding process. Moreover, all hardware processing units are shifted to VPU core that can configure them using physical address of on-chip memories directly. Such configuration of the hardware processing units reduced the overall encoding time.

6.4.2 Performance Comparison with Sequential (CPU only) Implementation

A group of experiments is carried out on these test sequences with four quantization parameters, i.e., QP = 24, 28, 32 and 36. The performance comparison of the implemented encoder with highly optimized sequential x264 encoder [114] is presented in Table 6.7 for NTSC test sequences. Table 6.7 illustrates that the proposed scheme shows a consistent gain in encoding rate for all sequences ranging from 58.70 fps at QP = 24 to 72.10 fps at QP = 36.

For HD 720p resolution test sequences, the quantization parameters set are 26, 30

Table 6.7: Performance Comparison for NTSC Sequences

Qp	24		28		32		36	
Sequence	Seq. fps	Proposed fps	Seq. fps	Proposed fps	Seq. fps	Proposed fps	Seq. fps	Proposed fps
Garden	3.73	44.50	4.04	58.48	4.31	61.56	4.59	70.85
Football	3.65	53.86	4.00	54.38	4.41	59.37	4.78	71.75
Intros	4.02	64.62	5.47	67.82	6.34	70.73	6.72	72.23
Mobile	3.55	48.78	3.74	58.36	3.96	63.50	4.33	70.62
Galleon	4.89	65.99	5.51	70.15	6.24	72.29	7.01	74.63
Vtc1nw	6.70	69.38	7.42	71.11	7.75	72.50	7.97	74.22
Washdc	4.79	63.80	5.75	66.57	6.72	68.07	7.38	70.40
Average	4.48	58.70	5.13	63.83	5.68	66.86	6.11	72.10

and 34. Table 6.8 shows the frame rate comparison of the proposed encoder with sequential software implementation [114]. For most of the 720p test sequences, real-time encoding is achieved. The average increased in encoding rate is ranging from 25.32 fps at QP = 26 to 32.04 fps at QP = 34.

Table 6.8: Performance Comparison for 720p Sequences

Qp	26		30		34	
Sequence	Sequential fps	Proposed fps	Sequential fps	Proposed fps	Sequential fps	Proposed fps
Stockholm	1.53	28.73	1.87	30.42	2.32	32.41
Shields	1.47	27.68	1.80	30.17	2.17	33.77
Parkrun	1.33	18.99	1.44	23.78	1.66	28.34
Mobcal	1.52	25.91	1.79	31.65	2.17	33.67
Average	1.46	25.32	1.72	29.00	2.08	32.04

The experimental results demonstrate that for both resolutions NTSC and 720p, the proposed scheme achieved a speedup of more than 12× when compared with highly optimized sequential implementation on CPU core without loss in PSNR.

Although the proposed solution is based on x264 baseline profile but it can also be used directly for main or high profile with B frames and CABAC entropy coding by replacing CAVLC module with CABAC implemented in software and assigned to the CPU. However, the CABAC implementation may result in increase of encoding time. For HD 720p and higher resolutions video sequences real-time encoding may not be possible.

6.4.3 Performance Comparison with State-of-the-Art Implementation Schemes

Table 6.9 compares the proposed scheme with previous works in terms speedup gain. The comparison of the proposed scheme is done with three preceding works, Momcilovic et al.'s [37] , Alvanos et al.'s [97] and Sankaraiah et al.'s [89]. It can be seen in Table 6.9 that the proposed scheme outperformed the other techniques in terms of speedup gain.

Table 6.9: Performance Comparison with State-of-the-Art Implementation Schemes

Techniques	Speedup gain
Momcilovic et al.'s [37]	2.5×
Alvanos et al.'s [97]	4.7× - 8.6×
Sankaraiah et al.'s [89]	5.6× - 10×
Proposed	more than 12×

6.4.4 Energy Saved

Energy saving is another important metric to evaluate the performance of any technique for mobile devices. As the energy consumption is one of the most critical issues in case of mobile application processors, so the implementation scheme must cater for this important issue. The energy consumption for sequential (CPU only) and proposed implementation is computed as follows: applied voltage is 5 volt, steady state current (standby mode) is 180 mA, qp is 24 and number of iterations are 5. Average system current in the case of sequential and the proposed implementation is 235 mA and 242 mA, respectively.

The power consumption comparison of proposed implementation with sequential implementation is given in Table 6.10. Although instantaneous power consumed

Table 6.10: Power Consumption Comparison

Implementation	Avgerage Instant. current mA	Instant. power consumed W
Sequential	55	0.28
Proposed	62	0.31

by the proposed scheme is higher, the overall power consumed to encode 300 frames is much lower than the sequential software implementation. Table 6.11 and 6.12 provide the comparison of energy consumed to encode 300 frames of NTSC and 720p resolutions, respectively. The proposed technique saves about 4.90 mWh and 58.28 mWh energy to encode 300 frames of NTSC and 720p resolutions, respectively.

Table 6.11: Energy Consumption Comparison for NTSC Sequences

Sequences	Sequential mWh	Proposed mWh	Energy saved mWh
Garden	6.14	0.58	5.56
Football	6.28	0.48	5.80
Intros	5.70	0.40	5.30
Mobile	6.45	0.53	5.92
Galleon	4.68	0.39	4.30
Vtc1nw	3.42	0.37	3.05
Washdc	4.78	0.40	4.38
Average	5.35	0.45	4.90

Table 6.12: Energy Consumption Comparison for 720p Sequences

Sequences	Sequential mWh	Proposed mWh	Energy saved mWh
Stockholm	61.56	3.36	58.21
Shields	62.50	3.54	58.96
Parkrun	64.45	5.67	58.78
Mobcal	60.66	4.04	56.62
Average	62.26	3.99	58.28

6.4.5 Scalability of the Proposed Scheme for High Efficiency Video Coding (HEVC)

The proposed scheme is scalable and it can be easily modified for High Efficiency Video Coding (HEVC) standard. HEVC retains the basic hybrid coding architecture of H.264/AVC. Moreover, the basic tool set of video encoding in HEVC codec is quiet similar to H.264/ AVC coding standard e.g. intra and inter prediction, motion estimation and compensation, transformation, quantization, entropy encoding and de-blocking filtering. However, HEVC uses more adaptive quad-tree structure based on a coding tree unit (CTU) instead of a macroblock (MB) and several improvements have been made in tool set to improve its compression performance and throughput speed (particularly for parallel-processing architectures).

For HEVC, the basic encoding flow and building blocks of the proposed scheme remain the same as shown in Fig.6.12. However, the following modifications are required inside the building blocks.

1. In HEVC, the basic processing unit is CTU instead of MB. So, the main encoding loop will be over CTUs.
2. Both the processing cores (CPU and VPU) and hardware processing units will be pipelined at CTU level.
3. The DBLK stage in proposed design will perform both de-blocking and sample-adaptive offset (SAO) filtering.
4. In HEVC, CAVLC will be replaced by CABAC.
5. Memory requirement will increase because of large size of meta-data structures in HEVC.

6.5 Summary

This chapter presents an end-to-end software-hardware hybrid implementation scheme of H.264/AVC encoder for JZ47x series of processors. The proposed scheme not only distributes the encoding tasks between hardware and software modules but also performs synchronization between different cores and the hardware accelerator blocks. The idling and polling time of hardware processing units is also reduced by exploiting the fine-grained parallelism at MB level. The possibility of asynchronously processing on the CPU and VPU is also effectively exploited to efficiently distribute the computational load among these processing cores. This resulted in increasing the encoding rate and reducing power consumption. The implemented encoder can encode NTSC and HD 720p video sequences at 72 fps and 32 fps, respectively. This is approximately more than 12 times the encoding rate of the highly optimized sequential encoder. The performance improvements techniques presented in this chapter can also be used for other families of processors. It may be noted that the proposed scheme is scalable and it can be easily modified for High Efficiency Video Coding (HEVC) standard.

The research contributions related to this chapter:

- Muhammad Asif, Imtiaz A. Taj, S. M. Ziauddin, Maaz Bin Ahmad and M. Tahir " *A hybrid scheme based on pipelining and multitasking in mobile application processors for advanced video coding* ", Scientific Programming Journal, vol. 2015.

- Muhammad Asif, Saqib Majeed, Imtiaz A. Taj, S. M. Ziauddin and Maaz Bin Ahmad " *Exploiting MB Level Parallelism in H.264/AVC Encoder for Multi-Core Platform* ", 2014 IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA).

Chapter 7

CONCLUSION AND FUTURE DIRECTIONS

7.1 Introduction

This chapter concludes the research work. It presents summary of the contributions of this research work. In addition to it, this chapter describes some of the possible future directions of the proposed work which can be adopted by researchers to enhance this work.

7.2 Contributions to Knowledge

Computational complexity limits the coding performance of software based video encoders. The objective of this research is to establish a computationally efficient framework for macroblock prediction to reduce the computational complexity of the video encoders. Moreover, to develop novel schemes which effectively exploit the data and task level parallelism in video encoders to improve their coding efficiency. These schemes ensure that the encoders can effectively exploit the processing resources offered by latest generation of the mobile application processors to maximize the encoding rate. There are three major contributions of this research work.

7.2.1 Efficient Framework for Macroblock Prediction

This framework comprises several new algorithms to reduce the computational complexity of the macroblock prediction process. Complexity saving is achieved by excluding as many intra and inter prediction modes as possible prior to the RDO process. The objective of this framework is to decrease the computational complexity without any significant loss in visual quality. The proposed framework is distributed into four stages and at each stage certain decisions are made to shortlist the candidate prediction modes for RDO calculation. Each stage of the proposed framework for macroblock prediction is briefly described as follows:

Stage 1: Prediction type decision

The decision of macroblock prediction type (predicted as intra or inter) is made by using Adaboost classifier. The Adaboost classifier classify each MB in one of the three defined classes (Intra, Inter or both) based on spatial and temporal features of video sequence. In case of Intra or inter class, no further processing related to prediction type decision is performed. If both intra and inter are candidates than macroblock type decision is made at later stages of the framework. Experiments demonstrate that this technique is about 25.61 times faster than the full search method with marginal reduction in coding quality with respect to BDPSNR and BDBR i.e by the amount of 0.006 dB and 0.169%, respectively.

Stage 2: Early mode exclusion

This stage of the framework consists of two algorithms, first for selecting appropriate block size for intra prediction and second for performing early detection of SKIP mode of macroblock.

Intra block size selection algorithm - In this work, two schemes are proposed to select appropriate block size for intra prediction. First one is based on adaptive threshold and second is based on Adaboost classifier. These techniques classify each MB in one of the three defined classes (Intra 4×4, Intra 16×16 or both) based on spatial features of video sequence. In case of Intra 4×4 or Intra 16×16 class, no further processing is carried out related to intra prediction block size decision. If both Intra 4×4 and Intra 16×16 are candidates then block size decision is made at later stages of the framework. Moreover, if Intra 4×4 is decided as appropriate block size for intra prediction then predication modes for intra 16×16 are skipped and vice versa. Experimental results evident that in case of adaptive threshold based algorithm, 13.75% time saving is achieved with marginal loss in prediction quality i.e. 0.379% increase in BDBR and 0.0318 dB decrease in BDPSNR. On the other hand, for Adaboost classifier based approach, 18.34% time saving is acquired at the cost of 0.0.548% increase in BDBR and about 0.045 dB decrease in BDPSNR. Adaboost classifier based scheme saved about 4.59%more encoding time as compared to adaptive threshold based scheme with 0.169% increase in BDBR and 0.013 dB decrease in BDPSNR.

SKIP early detection algorithm - In case of inter MB type, early detection of SKIP macroblock is performed to exclude the rest of the prediction modes. In this case, computational savings are ensured by recognizing SKIP mode, preceding the mode decision and motion estimation process, the macroblocks may be skipped and result in further saving of processing power. The SKIP macroblock prediction is made if the conditions mentioned below are fulfilled:

- 16×16 is the selected macroblock partition size.
- Both the components MVDx and MVDy of the motion vector difference (MVD) are zero.
- The selected frame of reference is the former frame in display order.
- CBP (coded block pattern) is zero i.e. all quantized coefficients are zero.

Experimental results indicate that on average the SKIP mode early detection algorithm is about 23.97% faster than the full search method with 0.019% decrease in BDBR and increase in 0.002 BDPSNR.

Stage 3: Rapid mode selection

This stage of the framework short lists the intra and inter prediction modes for RDO process. It consists of algorithms for inter and intra mode selection.

Inter prediction mode selection algorithm - Performed the mode selection based on motion homogeneity and residual complexity measures of an MB. The motion homogeneity is calculated on the motion vector (MV) field generated by a light weight motion estimator that converges towards the actual motion at 8×8 block level. The MB residual complexity is defined by SAD of its constituent 8×8 blocks. Based on the motion homogeneity and residual complexity measures, each MB is classified into one of the five predefined classes when the specified conditions hold true. From the class information of an MB, the candidate inter prediction modes are determined for RDO process. Experiments show that the proposed scheme decreases the encoding time by 64% on average with 0.02 dB loss in BDPSNR and 0.28% increase in BDBR.

Intra prediction mode selection algorithm - Performed the mode selection based on local edge information of the blocks. In this algorithm, a Sobel operator is applied to create edge map which is used to calculate the local edge information i.e. edge direction and magnitude. The local edge information of the block is used to determine the candidate intra prediction modes for RDO process. Experiments prove that the proposed scheme speeds up the encoder by 61.63% on average with 0.192 dB loss in BDPSNR and 2.999% increase in BDBR.

Stage 4: RDO mode elimination

At this stage of the framework, reference RDO algorithm is used to calculate RDcost for shortlisted prediction modes of an MB. The prediction mode with minimum RDcost is selected as coding mode of an MB. In the proposed framework RDO calculation is performed only for shortlisted candidate modes. This helps to significantly reduce the computational complexity of the encoding process.

The experimental results indicate that the overall proposed framework provides encoding time saving up to 74% with an average with 0.032 dB loss in BDPSNR and 0.461% increments in BDBR as compared to exhaustive RDO.

7.2.2 Motion Compensation Complexity Reduction

A new complexity reduction algorithm for an H.264/AVC coding standard based on motion compensation has been introduced here. An improved performance is attained through addressing the reasons of computational cost related to data manipulation and linear interpolation sub modules of motion compensation algorithm. The factors of computational complexity in the reference software implementation of motion compensation algorithm are handled in three different stages. The first stage belongs to edge creation around the reference frame that helps to prevent the clipping of unrestricted motion vectors. In the second stage, macroblocks are separated on the basis of inter prediction mode information (macroblock or sub-macroblock partitioning) and process accordingly. Finally, parallelism is introduced in order tho calculate the predicted sample values for motion compensation through SIMD instructions. Initial two stages reduce the performance degradation in data manipulation sub-module and the last one simplifies the linear interpolation sub-module of the motion compensation. This scheme decreases the time complexities by 5 to 6 times by reducing the looping overhead, repeated access of the same pixel from memory, utilizing the memory resources and processing resources.

7.2.3 Parallel Task Assignment in Advanced Video Coding

This is an end-to-end hybrid hardware-software implementation scheme based on pipelining and multitasking for advanced video coding. This scheme distributes the encoding tasks among hardware and software modules. A series of optimization techniques are developed to speedup the memory access and data transferring among memories. Moreover, an efficient data re-usage design is proposed for the de-block filter video processing unit to reduce the memory accesses. Furthermore, task and data level parallelism in video encoder is effectively exploited to improve their coding efficiency. The parallelism is exploited at both coarse-grain and fine-grain level. The coarse-grain level parallelism exploitation is done by concurrently executing multiple tasks on different processing cores while fine-grain level parallelism is achieved by using SIMD instructions. Such exploitation of parallelism also helps to better utilize the computational power offered by advanced media processors. The experimental results tell that the proposed scheme increases the encoding rate and reduces the power consumption. The implemented encoder can encode NTSC and HD 720p video sequences at 72 fps and 32 fps, respectively. This is approximately more than 12 times the encoding rate of the highly optimized sequential encoder.

7.3 Conclusion

Modern video coding standards such as High Efficiency Video Coding (HEVC) and H.264/MPEG-4 Advanced Video Coding (AVC) supersede the previous coding standards due to their improved coding efficiency. Due to enhanced compression ability and coding flexibility, these coding standards have the ability to introduce new video services like mobile multimedia streaming and video telephony. But, the gains based on performance of these standards are achieved at a significant higher cost of computational complexity and so the processing resources needed to enforce these encoders in a low power mobile platform may become a major hurdle. The objective of this research work is to reduce the computational complexity of video encoders as well as to exploit their data and task level parallelism to improve coding efficiency.

In this research, an efficient framework for macroblock prediction and parallel task assignment in video coding has been presented. The proposed framework incorporates several innovative approaches to reduce the computational complexity of prediction process and for exploiting data and task level parallelism in video encoders to improve their coding efficiency. The framework excludes as many intra and inter prediction modes as possible prior to the RDO process without any major loss in video quality. In the best case, the proposed framework selects one MB type either intra or inter and one corresponding the most suitable prediction mode, such that the complete RDO process is skipped. Moreover, a computationally efficient technique for motion compensation is presented to perform inter prediction process that speeds up the video encoding and decoding process. Furthermore, in this work, parallelism in video encoder is exploited at both coarse-grain level and fine-grain level. The coarse-grain level parallelism exploitation is done by concurrently executing multiple tasks on different processing cores while fine-grain level parallelism is achieved by using SIMD instructions. Such exploitation of parallelism also helps to better utilize the computational power offered by advanced media processors.

The proposed techniques have been evaluated on different types of video sequences, with divers motion activities by tweaking encoder parameters. The observations based on experiments testify the effectiveness of the suggested research. The proposed techniques are suitable to perform video encoding on both general purpose desktop and smart/mobile devices having constraints of processing and computational power.

The techniques described in this thesis fulfill the aim and objectives of this research task. They are practical and can be applied to video coding applications where encoding time is an important factor, such as real-time multimedia systems, to control computational complexity whilst maintaining acceptable video quality. These algorithms can also benefit power-constrained systems, for example, mobile video phones and reduction in their power consumption for encoding video, hence potentially achieving longer battery life.

7.4 Future Directions

Although, the results obtained by encoding multiple video sequences with different motion activities prove the effectiveness of the proposed framework. There are several aspects which may be considered in the future research.

- The techniques presented in this research work are assessed using fixed quantization values. Experiments should be performed to evaluate the output of the proposed framework along with established rate control techniques. This may open further avenues for investigation.
- In the proposed framework, the Jeon et al. [109] algorithm is used for SKIP mode early detection. It should be interesting to integrate other low complexity SKIP macroblock early detection algorithms and evaluate the performance of the framework.
- The structure of the proposed framework is highly flexible and it is possible to integrate other existing low complexity algorithms of intra or inter mode selection. This may help to evaluate the effect of such replacement on encoding time and video quality.
- The encoder pipeline design proposed in this research work is evaluated using hardware video processing units. It is possible to evaluate this design on multi-core platforms. This may help to improve encoding rate.
- Although, this research work primarily focuses on H.264/AVC standard, the developed techniques are scalable and can be easily modified for other coding standards particularly for High Efficiency Video Coding (HEVC). It should be possible to extend this research work HEVC standard.

REFERENCES

[1] M. Paul P. K. Podder and M. Murshed. Fast mode decision in the hevc video coding standard by exploiting region with dominated motion and saliency features. *PLoS ONE*, 11(3):1–22, 2016.

[2] Wang Q. Radicke S., Hahn J. and Grecos C. A parallel hevc intra prediction algorithm for heterogeneous cpu+gpu platforms. *IEEE Transactions on Broadcasting*, 62(1):103–119, 2016.

[3] T. Nguyen and D. Marpe. Objective performance evaluation of the hevc main still picture profile. *IEEE Transactions on Circuits Syst. Video Technol.*, 25(5):790–797, 2015.

[4] H. Zhang and Z. Ma. Fast intra mode decision for high efficiency video coding (hevc). *IEEE Transactions on Circuits Syst. Video Technol.*, 24(4):660–668, 2014.

[5] N. Hu and E.-H. Yang. Fast mode selection for hevc intra-frame coding with entropy coding refinement based on a transparent composite model. *IEEE Transactions on Circuits Syst. Video Technol.*, 25(9):1521–1532, 2015.

[6] He J. Sun B. Yu L., Ge F. and Dai F. Mode activity based adaptive fast intra mode decision for hevc/h.265. *8th International Congress on Image and Signal Processing (CISP)*, pages 153–157, 2015.

[7] H. Sun S. Goto Z. Sheng, D. Zhou and C. Gurrin. Low-complexity rate-distortion optimization algorithms for hevc intra prediction. *MultiMedia Modeling, Springer*, pages 541–552, 2014.

[8] Liu J. Gao Y. Yu X., Liu Z. and Wang D. Vlsi friendly fast cu/pu mode decision for hevc intra encoding:leveraging convolution neural network. *IEEE International Conference on Image Processing (ICIP)*, pages 1285–1289, 2015.

[9] N. Roma S. Momcilovic, A. Ilic and L. Sousa. Dynamic load balancing for real-time video encoding on heterogeneous cpu+gpu systems. *IEEE Trans. Multimedia*, 16(1):108–121, 2014.

[10] Wang D. Han Q. Zhu J., Liu Z. and Yang Song. Hdtv1080p hevc intra encoder with source texture based cu/pu mode pre-decision. *ASP-DAC*, pages 367–372, 2014.

[11] M. Kim S. Cho S.-H. Bae, J. Kim and J. S. Choi. Assessments of subjective video quality on hevc-encoded 4k-uhd video for beyond-hdtv broadcasting services. *IEEE Transactions on Broadcast.*, 59(2):209–222, 2013.

[12] Suhring K. Bossen F., Bross B. and Flynn D. Hevc complexity and implementation analysis. *IEEE Transactions on Circuits and Systems for Video Technology*, 22:1685–1696, 2012.

[13] Zhang G. Pan Z. Yuan H. Zhang Y., Kwong S and Jiang G. Low complexity hevc intra coding for high-quality mobile video communication. *IEEE Transactions on Industrial Informatics*, 11(6):1492–1504, 2015.

[14] Zhang Z Shen L. and Liu Z. Effective cu size decision for hevc intracoding. *IEEE Transactions on Image Processing*, 23:4232–4241, 2014.

[15] Lu Y. Real-time cpu based h. 265/hevc encoding solution with intel platform technology. *Intel Corporation, Shanghai, PRC.*, 2013.

[16] Liu H. and Jie Y. Fast hevc intra-prediction mode decision based on conditional selection with hybrid cost ranking. *IEEE Workshop on Signal Processing Systems (SiPS)*, pages 1–6, 2015.

[17] Zhang Z. Shen L. and An P. Fast cu size decision and mode decision algorithm for hevc intra coding. *IEEE Transactions on Consumer Electronics*, 59:207–213, 2013.

[18] V. Sze M. Budagavi J. Lainema, W.-J. Han and G. J. Sullivan. Intra-picture prediction in hevc" in high efficiency video coding (hevc)algorithms and architectures. *Springer*, pages 91–112, 2014.

[19] ISO/IEC JTC 1 ISO/IEC 11172-2 (MPEG-1). Coding of moving pictures and associated audio for digital storage media at up to about 1.5 mbit/spart 2: Video. 1993.

[20] ISO/IEC JTC 1 ISO/IEC 14496-2 (MPEG-4 Visual version 1). Coding of audio-visual objectspart 2: Visual. 1999.

[21] version 2 ITU-T Rec. H.261. Video codec for audiovisual services at px64 kbit/s. 1993.

[22] ITU-T Rec. H.263. Video coding for low bit rate communication. 1995.

[23] ITU-T Rec. H.262, ITU-T ISO/IEC 13818-2 (MPEG 2 Video), and ISO/IEC JTC 1. Generic coding of moving pictures and associated audio informationpart 2: Video. 1994.

[24] ITU-T and ISO/IEC JTC 1. Advanced video coding for generic audio-visual services, itu-t rec. h.264 and iso/iec 14496-10 (avc). 2003.

[25] ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11. High efficiency video coding (hevc) text specification draft 10 (for fdis last call), jctvc-l1003-v34, 12th meeting, geneva. 2013.

[26] T. Wiegand, A. Sullivan, G. Bjntegaard, and A. Luthra. Overview of the h.264/avc video coding standard. *Circuit and System for Video Technology (TCSVT), IEEE Transactions on*, 13(7):560–576, 2003.

[27] Woo-Jin Han Thomas Wiegand Gary J. Sullivan, Jens-Rainer Ohm. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on Circuits and Systems for Video technology*, 22(12):1649–1668, 2012.

[28] G. J. Sullivan J.-R. Ohm B. Bross, W.-J. Han and T. Wiegand. High efficiency video coding (hevc) text specification draft 9, document jctvc-k1003. *ITU-T/ISO/IEC Joint Collaborative Team on Video Coding (JCT-VC)*, 2012.

[29] Sullivan-G.J. Schwarz H. Thiow Keng Tan-Wiegand T Ohm, J. Comparison of the coding efficiency of video coding standards including high efficiency video coding (hevc). *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):16691684, 2012.

[30] G. Bjntegaard V. Sze M. Budagavi M. Budagavi, A. Fuldseth and G. J. Sullivan. Hevc transform and quantization" in high efficiency video coding (hevc)algorithms and architectures. *Springer*, pages 141–169, 2014.

[31] C. F. Tsengand Y. T. Lai. Fast coding unit decision and mode selection for intra-frame coding in high-efficiency video coding. *IET image processing*, 10(3):215–221, 2016.

[32] J. Lee, S. Kim, K. Lim, H. J. Kim, and S. Lee. Fast intermode decision algorithm based ongeneral and local residual complexity in h.264/avc. *EURASIP J. Image and Video Process.*, 2013.

[33] W.-J. Han J. Min J. Lainema, F. Bossen and K. Ugur. Intra coding of the hevc standard. *IEEE Transactions on Circuits Syst. Video Technol.*, 22(12):1792–1801, 2012.

[34] G. Sullivan and Wiegand T. Rate-distortion optimization for video compression. In *IEEE Signal Processing Magazine*, 1998.

[35] A. Elyousfi. Fast gravity direction-based ultra-fast intraprediction algorithm for h.264/avc video coding. *Signal Image and Video Processing*, 7:53–65, 2011.

[36] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro. H264/avc baseline profile decoder complexity analysis. *Circuit and System for Video Technology, IEEE Transactions on*, 13(7):704–716, 2003.

[37] S. Momcilovic, N. Roma, and L Sousa. Exploiting task and data parallelism for advanced video coding on hybrid cpu + gpu platforms. *J,. Real Time Image Proc.*, 2013.

[38] D. Marpe V. Sze M. Budagavi H. Schwarz, T. Schierl and G. J. Sullivan. Block structures and parallelism features in hevc" in high efficiency video coding (hevc)algorithms and architectures. *Springer*, pages 49–90, 2014.

[39] Muhammad Asif, Imtiaz A. Taj, S. M. Ziauddin, Maaz Bin Ahmad, and M. Tahir. An efficient scheme for intra prediction block size and mode selection in advanced video coding. In *Frontiers in Information Technology (FIT)*, 2015.

[40] Muhammad Asif, Imtiaz A. Taj, S. M. Ziauddin, Maaz Bin Ahmad, and Atif Raza. An efficient inter prediction mode selection scheme for advanced video coding based on motion homogeneity and residual complexity. *IEEJ Transactions on Electrical and Electronic Engineering*, 13(7), 2016.

[41] Muhammad Asif, Masood Farooq, and Imtiaz A. Taj. Optimized implementation of motion compensation for h.264decoder. In *5th International Conference on Computer Sciences and Convergence Information Technology (ICCIT)*, pages 216–221, 2010.

[42] Muhammad Asif, Saqib Majeed, Imtiaz A. Taj, S. M. Ziauddin, and Maaz Bin Ahmad. Exploiting mb level parallelism in h.264/avc encoder for multi-core platform. In *11th International Conference on Computer Systems and Applications (AICCSA)*, pages 125–130, 2014.

[43] Muhammad Asif, Imtiaz A. Taj, S. M. Ziauddin, Maaz Bin Ahmad, and M. Tahir. A hybrid scheme based on pipelining and multitasking in mobile application processors for advanced video coding. *Scientific Programming Journal*, 13(7), 2015.

[44] Iain E. G. Richardson. H.264 and mpeg-4 video compression. *Willy and Sons Ltd.*, 2003.

[45] H. S. Malvar, A. Hallapuro, and M. Karzewicz. Low-complexity transform and quantization in h.264/avc. *IEEE Trans. Cirduits and System. Video Technology*, 13(7):598–603, 2003.

[46] Hadamard transform, in introduction to real-time imaging. *SPIE Optical Engineering Press*, TT19:60–67, 1995.

[47] D. Marpe, H. Schwarz, and T. Wiegand. Context-based adaptive binary arithmatic coding in the h.264/avc video compression standard. *IEEE Trans. Cirduits and System. Video Technology*, 13(7):620–636, 2003.

[48] P. List, A. Joch, J. Lainema, G. Bjontegaard, and M. Karczewicz. Adaptive deblocking filter. *IEEE Trans. Circuits and System. Video Technology*, 13(7):614–619, 2003.

[49] Ortega A. and Ramchandran K. Rate-distortion methods for image and video compression. In *IEEE Signal Processing Magazine*, 1998.

[50] K. P. Lim, Sullivan G., and T. Wiegand. Text description of joint model reference encoding methods and decoding concealment methods. In *Joint Video Team Document JVT-O079*, 2005.

[51] X. Jing and L. Chau. An efficient inter mode decision approach for h.264 video coding. In *Proceedings of IEEE International Conference on Multimedia and Expo*, volume 2, pages 1111–1114, 2004.

[52] Y. H. Kim, J. W. Yoo, S. W. Lee, J. Shin, J. Paik, and H. K. Jung. Adaptive mode decision for h.264 encoder. *Electron. Lett.*, 40(19):11721173, 2004.

[53] J. Bu, S. Lou, C. Chen, and J. Zhu. A predictive block-size mode selection for inter frame in h.264. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, page 917920, 2006.

[54] C.H. Kuo, M. Shen, and C. C. J. Kuo. Fast inter-prediction mode decision and motion search for h.264. In *Proc. IEEE International Conference on Multimedia and Expo (ICME 04)*, volume 1, page 663666, 2004.

[55] B. Feng, G. X. Zhu, and W. Y. Liu. Fast adaptive inter-prediction mode decision method for h.264 based on spatial correlation. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS 06)*, page 18041807, 2006.

[56] E. M. Enriquez, A. J Moreno, and F. D. D. Maria. An efficient intermode decision algorithm based on motion homogeneity for h.264/avc. *IEEE Trans. Consumer Electron*, 56(2):826834, 2010.

[57] D. Zhu, Q. Dai, and R. Ding. Fast inter prediction mode decision for h.264. In *Proceedings of IEEE International Conference on Multimedia and Expo*, volume 2, pages 1123–1126, 2004.

[58] X. Jing and L. P. Chau. Fast approach for h.264 inter-mode decision. *Electron. Lett.*, 40(17):10501052, 2004.

[59] Z. Zhou and M.T. Sun. Fast macroblock inter mode decision and motion estimation for h.264/mpeg-4 avc. In *Proc. International Conference on on Image Processing (ICIP 04)*, volume 2, page 243263, 2004.

[60] D. Wu, P. Pan, K. P. Lim, S. Wu, Z. G. Li, X. Lin, S. Rahardja, and C. C. Ko. Fast intermode decision in h.264/avc video coding. *IEEE Trans. Circuits Syst. Video Technol.*, 15(7):953958, 2005.

[61] D. Wu, S. Wu, K. P. Lim, F. Pan, Z. G. Li, and X. Lin. Block intermode decision for fast encoding of h.264. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 3, page 181184, 2004.

[62] K. Bharanitharan, B. D. Liu, and J. F. Yang. Classified region algorithm for fast inter mode decision in h.264/avc encoder. *EURASIP J. Adv. Signal Process.*, 2010.

[63] H. Zeng, C. Cai, and K. K. Ma. Fast mode decision for h.264/avc based on macroblock motion activity. *IEEE Trans. Circuits Syst. Video Technol.*, 19(4):491499, 2009.

[64] Z. Liu, L. Shen, and Z. Zhang. An efficient intermode decision algorithm based on motion homogeneity for h.264/avc. *IEEE Trans. Circuits Syst. Video Technol.*, 19(1):128132, 2009.

[65] F. Pan, X. Lin, S. Rahardja, K. P. Lim, Z. G. Li, G. N. Feng, D. J. Wu, and S. Wu. Fast mode decision for intra prediction. In *JVT-G013, 7th JVT Meeting, Pattaya, Thailand*, 2003.

[66] J. Kim and J. Jeong. Fast intra-mode decision in h.264 video coding using simple directional masks. In *Visual Communication and Image Procissing, VCIP, Beijing China, Proceedings of SPIE*, volume 5960, page 10711079, 2005.

[67] F. Pan, X. Lin, S. Rahardja, K. P. Lim, Z. G. Li, D. Wu, and S. Wu. Fast mode decision algorithm for intraprediction in h.264/avc video coding. *IEEE Trans. Circuits Syst. Video Technol*, 15(7):813822, 2005.

[68] J.F. Wang, J.C. Wang, J.T. Chen, A.C. Tsai, and A. Paul. A novel fast algorithm for intra mode decision in h.264/avc encoders. In *IEEE Int. Symp. on Circuits and Systems*, pages 3498–3501, 2006.

[69] R. Su, G. Liu, and T. Zhang. Fast mode decision algorithm for intra prediction in h.264/avc with integer transform and adaptive threshold. *J. Signal Image Video Process.*, 1(1):11–27, 2007.

[70] Z. Wei, H. Li, and K. N. Ngi. An efficient intra mode selection algorithm for h.264 based on fast edge classification. In *IEEE International Symposium on Circuits and Systems, ISCAS*, page 36303633, 2007.

[71] H. Li, K.N. Ngan, and Z. Wei. Fast and efficient method for block edge classification and its application in h.264/avc video coding. *IEEE Trans. Circuits Syst. Video Technol.*, 18(6):756768, 2008.

[72] J.C. Wang, J.F. Wang, J.F. Yang, and J.T. Chen. A fast mode decision algorithm and its vlsi design for h.264/avc intra-prediction. *IEEE Trans. Circuits Syst. Video Technol.*, 17(10):1414–1422, 2007.

[73] A. Elyousfi, A. Tamtaoui, and E. Bouyakhf. A new fast intra prediction mode decision algorithm for h.264/avc encoders. *Int. J. Comput. Syst. Sci. Eng. IJCSSE*, 4:8995, 2008.

[74] Byeongdu LA, M. EOM, and Y. CHOE. Dominant edge direction based fast intra mode decision in the h.264/avc encoder. *Journal of Zhejiang University SCIENCE A*, 10(6):767777, 2009.

[75] Li-Li Wang and Wan-Chi Siu. H.264 fast intra mode selection algorithm based on direction difference measure in the pixel domain. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, page 10371040, 2009.

[76] K. Bharanitharan, Jiun-Ren Ding, Bo-Wei Chen, and Jhing-Fa Wang. Selective intra block size decision and fast intra mode decision algorithms for h.264/avc encoder. *IEICE Trans. Inf. Syst.*, 95(11):27202723, 2012.

[77] B. Meng and O. C. Au. Fast intra-prediction mode selection for 44 blocks in h.264. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing,ICASSP 2003*, volume 3, pages 389–392, 2003.

[78] J.W. Chen, C.H. Chang, C.C. Lin, Y.H. Yang, J.I. Guo, and J.S. Wang. A condition-based intra prediction algorithm for h.264/avc. In *IEEE International Conference on Multimedia and Expo, ICME*, page 10771080, 2006.

[79] C. Kim, H. Shih, and C.J. Kuo. Fast h.264 intra-prediction mode selection using joint spatial and transform domain features. *J. Vis. Commun. Image Represent.*, 17(2):291310, 2006.

[80] A.C. Yu, N.K. Ngi, and G.R. Martin. Efficient intra- and inter-mode selection algorithms for h.264/ avc. *J. Vis. Commun. Image Represent.*, 17(2):310322, 2006.

[81] H.264/avc reference software. *downloaded from http://iphome.hhi.de/suehring/tml.*

[82] M. O. Khan, U. Khan, S. A. Rahim, and S. I. Ali. Optimization of motion compensation for h.264 decoder by pre-calculation. In *8th International Multitopic Conference, 2004. Proceedings of INMIC 2004.*, pages 55–60, 2004.

[83] S. Z. Wang, T. A. Lin, T. M. Liu, and C. Y. Lee. A new motion compensation design for h.264/avc decoder. In *IEEE International Symposium on Circuits and Systems, ISCAS 2005.*, volume 5, pages 4558–4561, 2005.

[84] H. W. Feng, Z. G. Mao, J. X. Wang, and D. F. Wang. Design and implementation of motion compensation for mpeg-4 as profile streaming video decoding. In *5th International Conference on ASIC, 2003. Proceeding*, volume 2, pages 942–945, 2003.

[85] A. Azevedo, B. Zatt, L. Agostini, and S. Bampi. Motion compensation decoder architecture for h.264/avc main profile targeting hdtv. In *IFIP International Conference on Very Large Scale Integration, 2006.*, pages 52–57, 2006.

[86] J. Lee, S. Moon, and W. Sung. H.264 decoder optimization exploiting simd instruction. In *The 2004 IEEE Asia-pacific conference on circuit and system, 2004.*, volume 2, pages 1149–1152, 2004.

[87] T. Bhatia. Optimization of h.264 high profile decoder for pentium 4 processor. M.s. thesis, The University of Texas at Arlington in Partial Fulfillment, December 2005.

[88] S. B. Sheshadri. Optimization of h.264 baseline decoder on arm9tdmi processor. M.s. thesis, The University of Texas at Arlington in Partial Fulfillment, December 2005.

[89] S. Sankaraiah, H. S. Lam, C. Eswaran, and J. Abdullah. Gop level parallelism on h.264 video encoder for multicore architecture. In *International Conference on Circuits, System and Simulation, IPCSIT*, volume 7, pages 127–133, 2011.

[90] A. Rodriguez, A. Gonzalez, and M. P. Malumbres. Hierarchical parallelization of an h.264/avc video encoder. In *Proc. Intl. Symp. On parallel computing in electrical engineering*, pages 363–368, 2006.

[91] Y. K. Chen, E. Q. Li, X. Zhou, and S. Ge. Implementation of h.264 encoder and decoder on personal computers. *J. Visual Communication and Image Representation*, 17(2):509532, 2006.

[92] S. Sun, D. Wang, and S. Chen. A highly efficient parallel algorithm for h.264 encoder based on macro-block region partition. In *Proceedings of the 3rd international conference on High Performance Computing and Communications*, page 577585, 2007.

[93] Y. K. Chen, X. Tian, G. Steven, and M. Girkar. Towards efficient multi-level threading of h.264 encoder on intel hyper-threading architectures. In *Proceedings of the 3rd international conference on High Performance Computing and Communications*, page 2630, 2004.

[94] Z. Zhao and P. Liang. A highly efficient parallel algorithm for h.264 video encoder. In *31st IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, 2006.

[95] H. K. Zrida, A. C. Ammari, A. Jemai, and M. Abid. High level optimized parallel specification of a h.264/avc video encoder. *International Journal of Computing and Information Sciences*, 9(1):34–46, 2011.

[96] M. Wen, J. Ren, N. Wu, H. Su, C. Xun, and C. Zhang. Data parallelism exploiting for h.264 encoder. In *International Conference on Multimedia and Signal Processing*, volume 1, page 188192, 2011.

[97] M. Alvanos, G. Tzenakis, D. S. Nikolopoulos, and A. Bilas. Task-based parallel h.264 video encoding for explicit communication architectures. In *IEEE Proc. Int. Conf. on Embedded Comp. Systems*, pages 217–224, 2011.

[98] Rob H. Gelderblom E.B. van der Tol E.G.T. Jaspers Erik B. Van Der Tol, Egbert G. T. Jaspers and R.H. Gelderblom. Mapping of h.264 decoding on a multiprocessor architecture. In *Proc. of the Image and Video Communications and Processing*, pages 707–718, 2003.

[99] J. Park and S. Ha. Performance analysis of parallel execution of h.264 encoder on the cell processor. In *Workshop on Embedded Systems for Real-Time Multimedia (ESTIMedia 07)*, pages 27–32, 2007.

[100] D. Wu, B. Lim, J. Eilert, and D. Liu. Parallelization of high-performance video encoding on a single-chip multiprocessor. In *IEEE Int. Conference on Signal Processing and Communications*, pages 145–148, 2007.

[101] X. He, X. Fang, C. Wang, and S. Goto. Parallel hd encoding on cell. In *Int. Symposium on Circuits and Systems (ISCAS 09)*, page 10651068, 2009.

[102] C. T. Lu and H. M. Hang. Multiview encoder parallelized fast search realization on nvidia cuda. In *Proc. Visual Communications and Image Processing (VCIP)*, page 14, 2011.

[103] M. Schwalb, R. Ewerth, and B. Freisleben. Fast motion estimation on graphics hardware for h.264 video encoding. *IEEE Trans. Multimed*, 11(1):1–10, 2009.

[104] N. M. Cheung, X. Fan, O.C. Au, and M. C. Kung. Video coding on multicore graphics processors. In *IEEE Signal Process. Mag.*, volume 27, page 7989, 2010.

[105] A. Azevedo, B. Juurlink, C. Meenderinck, A. Terechko, J. Hoogerbrugge, M. Alvarez, A. Ramirez, and M. Valero. A highly scalable parallel implementation of h.264. *Transactions on High-Performance Embedded Architectures and Compilers (HiPEAC)*, 4(2):111–134, 2011.

[106] W.-N. Chen and H.-M. Hang. H.264/avc motion estimation implementation on compute unified device architecture (cuda). In *Proc. International Conference on Multimedia and Expo (ICME)*, page 697700, 2008.

[107] S. Momcilovic, N. Roma, and L. Sousa. Multi-level parallelization of advanced video coding on hybrid cpu/gpu platform. In *Proc. of the 10th International Workshop on Algorithms*, pages 165–174, 2012.

[108] G. de Haan, P.W.A.C. Biezen, H. Huijgen, and O. A. Ojo. True motion estimation with 3d recursive search block matching. *EURASIP J. Image and Video Process.*, 56(3):368379, 1993.

[109] B. Jeon. Fast mode decision for h.264. In *ISO/IEC JTC1/SC29/ WG11 and ITU-T SG16, Input Document JVT-J033*, 2003.

[110] G. Bjontegaard. Calculation of average psnr differences between rd-curves. In *In VCEG-M33, 13th Meeting, Austin, Texas*, pages 2–1, 2001.

[111] Jz4770 vpu programming manual. *Ingenic Semiconductor Co.,Ltd.*, 2011.

[112] Jz4770 data sheet. *Ingenic Semiconductor Co.,Ltd.*, 2011.

[113] Mxu instruction usage guide. *Ingenic Semiconductor Co.,Ltd.*, 2011.

[114] x264. *downloaded from http://developers.videolan.org/x264.html.*