POWER EFFICIENT MOTION ESTIMATION ALGORITHMS AND ARCHITECTURES FOR HEVC/H.265

Karam Singh

POWER EFFICIENT MOTION ESTIMATION ALGORITHMS AND ARCHITECTURES FOR HEVC/H.265

A

Thesis Submitted

in Partial Fulfillment of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

by

Karam Singh

(11610202)

DEPARTMENT OF ELECTRONICS AND ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

GUWAHATI-781039, ASSAM, INDIA

February, 2019

Dedicated to

My Parents

Prem Lata and Raj Singh;

My Brothers

Ajmer Singh and Satyadev;

for their love and support

DECLARATION

*This is to certify that the thesis entitled "**Power Efficient Motion Estimation Algorithms and Architectures for HEVC/H.265**", submitted by me to the Indian Institute of Technology Guwahati for the award of the degree of **Doctor of Philosophy** is a bonafide work carried out by me under the supervision of **Prof. Shaik Rafi Ahamed**. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.*

Signed:

Karam Singh

Department of Electronics & Electrical Engineering,

Indian Institute of Technology Guwahati,

Guwahati-781039, Assam, India.

Date:

CERTIFICATE

*This is to certify that the this thesis entitled "**Power Efficient Motion Estimation Algorithms and Architectures for HEVC/H.265**" submitted by **Karam Singh (11610202)**, a research scholar in the Department of Electronics and Electrical Engineering, Indian Institute of Technology Guwahati, for the award of degree of **Doctor of Philosophy**, is a record of an original research work carried out by him under my supervision and guidance. The thesis has fulfilled all requirements as per the regulations of the institute and in my opinion has reached the standard needed for submission. The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.*

Signed:

Supervisor: Prof. Shaik Rafi Ahamed
Department of Electronics & Electrical Engineering,
Indian Institute of Technology Guwahati,
Guwahati-781039, Assam, India.

Date:

Acknowledgments

First of all, I would like to thank my parents, brothers, sister, and sister in laws for all the love, support and belief they have given me over the course of my life. No words are enough to express their selfless love, affection and support.

I would like to express my deepest and most sincere gratitude to my supervisor Prof. Shaik Rafi Ahamed, for being more than a mentor. He was always there to give me the right direction in spite of his busy schedule. It was he who taught me lessons which go beyond circuit design. My heartfelt thanks to you Sir, for the unlimited support and patience towards me.

I am also very thankful to my doctoral committee members Prof. Harshal B. Nemade, Prof. Roy P. Paily and Dr. Gaurav Trivedi for sparing their precious time to evaluate the progress of my work and providing constructive criticism. I would like to thank Prof. Anil Mahanta and Dr. Nagarjuna Nallam for their valuable suggestion and help in carrying out this work. I would also like to extend my gratitude to the Head of the Department and staff members for providing all the necessary facility for carrying out this research work. My special thanks to Josphene Ma'am for her help in all respects.

I express my sincere thanks and gratitude to both the examiners for their valuable time and effort to evaluate my PhD thesis and for their suggestions to improve the presentation and quality of the thesis.

I would like to extend my sincere thanks to all my beloved friends at the VLSI R & D Laboratory, past and present - too many to list here, for creating the best workplace to do research. You guys have been excellent and it would be difficult to find a working group as good as your's. My special thanks to my seniors Dr. Surya Parkash Matcha, Dr. Rahul Shrestha, Dr. Gaurav Saxena, Dr. Vinaya MM, and Mr. Pavan Kumar Manchi for their guidance and support to carrying out this work.

Scholarship and EDA Tools are provided by MHRD and DeitY, Government of India, without which this work would not have been possible.

Finally, I would like to thank all my beloved friends Gaurav, Navneet, Arunita, Amit,

Neeraj, Deepak, Babita, Harikrishna, Saroj, Santosh, Sikandar, Sushanta, Ankit, and Mohit for their help, love, and support during my stay at IITG.

If I have missed anyone of you please forgive me as a true friend.

Karam Singh

Abstract

In the current and future video processing standards, the requirement is not only for improving the quality of video services but also towards achieving efficient bit-rate with better quality. The need for high-quality videos is gaining importance in applications such as entertainment and medical healthcare. Transmission and storage of such high-quality videos demand huge bandwidth and storage space respectively. The video coding standards are used to compress the video before transmission and revert back into the normal format at the receiving end. High-Efficiency Video Coding (HEVC) is the state of the art video coding standard developed by motion picture expert group (MPEG) as well as video coding expert group (VCEG). This standard supports a wide range of video resolutions, but mainly developed for high-resolution videos. It reduces the bit-rate requirement around 50% as compared to H.264/AVC video coding standard. However, the reduction in bit-rate achieved at the cost of the increase in computational complexity. Motion estimation (ME) is one of the most important and computationally extensive blocks in all the video coding standards. Hence, there is a great demand towards the development of efficient algorithms and architectures for ME. In this thesis, we have made an attempt to develop power-efficient ME algorithms and architectures for HEVC.

To begin with, a detailed study of the ME algorithm is carried out at the integer level. The techniques such as pixel truncation, sub-sampling and removing of the prediction unit part sizes are used to reduce the computational complexity of the ME at integer level without much degradation in the video quality. We also examined the effect of these techniques in term of bit-rate and PSNR on HEVC. In order to reduce complexity further, using these techniques, we proposed an algorithm which uses hexagonal search pattern with a fixed number of search points at each grid and labelled as hexagonal grid search (HGS). Simulation results show that the proposed algorithm is suitable for all types of video sequences.

The second part of the thesis emphasizes in the design of a low-power architecture for the HGS algorithm. Towards this, we first modified motion vector (MV) prediction processes such that it is more suitable for hardware implementation. It is not the proposed process, but we used

this with the proposed integer ME algorithm. In order to reduce computational power, in the proposed architecture, we used a few techniques such as data reuse, 4:2 and 3:2 compressors. We also replaced multiplication circuit with the look-up table and employed suitable adder in our design.

The final part of the thesis focuses on the new fractional ME (FME) algorithm and its corresponding architecture. The proposed algorithm is developed based on the fact that most of the final MVs lies either nearer to the integer MV position or have either horizontal or vertical movements. Simulation studies revealed that the amount of degradation depends on the types of search pattern used. So, we proposed three different search patterns. All three proposed pattern required fewer search points compared to the conventional method. We also highlighted the designing of the scalable architecture for Hadamard Transform (HT) which is used to compute a sum of absolute transform difference (SATD) operation involved in FME of HEVC. The proposed architecture is based on the decomposition of larger size HT into smaller HTs. The proposed architecture is capable to compute the HTs of all sizes supported by HEVC.

Overall, this thesis has investigated new power-efficient ME algorithms and their corresponding architectures suitable for HEVC video coding standard.

Contents

List of Figures

List of Tables

List of Acronyms

AVC	Advanced Video Coding
AMVP	Advanced Motion Vector Prediction
BO	Band Offset
CABAC	Context Adaptive Binary Arithmetic Coding
CAVLC	Context Adaptive Variable Length Coding
CB	Coding Block
CIF	Common Intermediate Format
CTU	Coding Tree Unit
CU	Coding Unit
dB	decibel
DBF	De-blocking Filter
DCT	Discrete Cosine Transform
DST	Discrete Sine Transform
EO	Edge Offset
FME	Fractional Motion Estimation
GOP	Group Of Pictures
GPBs	Generalized P and B pictures
HDTV	High Definition Television
HEVC	High Efficiency Video Coding
HGS	Hexagonal Grid Search
HT	Hadamard Transform
IDR	Instantaneous Decoder Refresh

IME	Integer Motion Estimation
ITU-T	International Telecommunication Union-Telecommunication sector
ISDN	Integrated Services Digital Network
ISO	International Organization for Standardization
IEC	International Electrotechnical Commission
JCT-VC	Joint Collaborative Team on Video Coding
JVT	Joint Video Team
JVET	Joint Video Experts Team
MC	Motion Compensation
ME	Motion Estimation
MPEG	Moving Picture Experts Group
MV	Motion Vector
MVC	Multiview Video Coding
MVP	Motion Vector Prediction
NTSC	National Television System Committee
PAL	Phase Alternating Line
POC	Picture Order Count
PPS	Picture Parameter Set
PSNR	Pick Signal to Noise Ratio
PB	Prediction Blocks
PU	Prediction Unit
QCIF	Quarter Common Intermediate Format
QP	Quantization Parameter
RD	Rate Distortion
RDO	Rate Distortion Optimization
RQT	Residual Quad-tree
SAD	Sum of Absolute Difference
SATD	Sum of Absolute Transform Difference

SAO	Sample Adaptive Offset
SD	Standard Definition
SSD	Sum of Square Difference
SVC	Scalable Video Coding
TB	Transform Block
TU	Transform Unit
TZS	Test Zonal Search
UHD	Ultra-High Definition
VCEG	Video Coding Experts Group
WP	Weighted Prediction
WPP	Wavefront Parallel Processing

1

Introduction

Contents

This thesis presents several computational complexity reduction techniques and design of low-power architecture for the High-Efficiency Video Coding (HEVC) standard. In order to better understand the results presented in the subsequent chapters, we first present the background of video compression and motion estimation (ME). Next, we discuss the major issue in ME. Finally, the chapter-by-chapter contribution of the thesis is outlined.

1.1 Background and Need for Video Compression

The world has seen a dramatic growth in video coding since the late 1980s. Video applications such as videophone, video conferencing through wired and wireless medium, streaming video, digital TV/HDTV broadcast, video database service, CD/DVD storage, etc., has become an essential part of our everyday life. A digital video incorporates a sequence of two-dimensional digital images representing a dynamic three-dimensional scene on certain sampling grids and at regular time intervals. The advantage of digital video is that digital representation is more robust against channel disturbance and provides higher visual quality and more functionalities, compared with analog video. Recently, there is a drastic change in the video communication technology from lower-resolution video to ultra-high definition (UHD) video. In recent days, there is great demand for UHD video in real-time applications; as was seen in the 2016 Rio Olympics.

Transmission of raw/uncompressed video sequence requires huge data rates. For example, an uncompressed 30-fps full high-definition (HD) (1920×1080) video requires a data rate of $1920 \times 1080 \times 30 \times 3 \times 8 = 1.5$ Gbps and 1 TB memory is required to store a 90 minute video, which is impractical using the present communication and storage infrastructures. Hence, data compression is essential to transmit and store such huge data. The data compression techniques exploit the high statistical redundancies present in the video signals. In addition, the human visual system is insensitive to certain distortions present in the visual signals. Moreover, the human visual system is more sensitive to brightness than color component. The redundancy of video signals is removed by video coding standards. Several video coding standards have been developed to compress the video data by removing the temporal and spatial redundancies

presents in video signals. Compression system involves the encoder which converts the source data into a compressed form and the decoder reverts the compressed data back into an original form. The encoder/decoder pair is often described as a CODEC(enCOder/DECoder).

Many video coding standards have been proposed over several years. Historically, the first digital video coding standard H.120 was developed in 1984 by International Telecommunication Union Telecommunication Standardization Sector (ITU-T) formerly known as International Telegraph and Telephone Consultative Committee (CCITT) [18]. H.120 was designed to perform video streaming at the rate of 1,544 kbps for National Television System Committee (NTSC) and 2,048 kbps for Phase Alternating Line (PAL) [19]. The H.120 was not successful due to its poor performance. The extension of potential innovation of video coding standard began around 1988 by ITU-T and developed H.261, the first practical video coding standard with widespread applications. The first design of this ITU-T video coding standard was published in 1988 and was the first member of the H.26x family. H.261 was originally designed for transmission over Integrated Services Digital Network (ISDN) lines with data rates that are integer multiples of 64 kbps. After this, various video coding standards such as Moving Picture Experts Group-1 (MPEG-1)(1993), H.262/MPEG-2 (1994), H.263 (1995), H.263+ (1997), H.263++ (2000), MPEG-4 (2000), H.264/MPEG-4 Part 10/Advanced Video Coding (AVC) and H.265/HEVC [20] have been standardized.

Since H.261, all the video coding standards work on the block-based hybrid coding algorithm. For the first time in this standard, 16×16 macro-block motion compensation (MC) concept was used. In addition to macro-block MC, this standard also uses 8×8 discrete cosine transform (DCT), scalar quantization, zigzag scan and variable-length coding. Later, H.263 was developed with additional features such as 3-D variable length coding of DCT coefficients, median motion vector prediction, bi-directional prediction and arithmetic entropy coding. The next standard H.264 was developed in 2003, which provided a significant increment in compression ratio and also saves up to 50% bit rate as compared to its previous video coding standards.

HEVC is the latest video coding standard, which was developed by the Joint Collaborative

Team on Video Coding (JCT-VC), a collaboration between the International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) MPEG and ITU-T Video Coding Experts Group (VCEG). The first version of the HEVC standard was ratified in January 2013 and published in June 2013. This is also known as H.265 and MPEG-H Part 2. It saves 50% bit-rate as compared to H.264/AVC. This standard supports resolutions up to 8192×4320, including 8K UHD [21]. Some of the new and improved features used in HEVC [21] are as follows:

- Large size blocks for ME/MC (up to 64×64 pixels)
- Directional prediction modes (up to 34) for different prediction units (PUs) in intra prediction
- Non-rectangular motion partition
- Quad-tree partition for prediction blocks as well as transform blocks
- Large size transforms up to 32×32
- In loop Deblocking Filter (DBF) and Sample Adaptive Offset (SAO) filter
- Rotational transforms for block sizes larger than (8×8)
- Wavefront parallel processing (WPP)
- Larger size interpolation filter (8-tap and 7-tap)
- Improved Context-based Adaptive Binary Arithmetic Coding

1.2 The Importance of Standards

Since the late 1980s, many different video coding techniques have been proposed and a large number of researchers are still working to further develop the new standards. Several works based on new and innovative compression techniques have so far been reported in the literature. However, the commercial video coding applications tend to use a limited number

of standardized techniques for video compression. Standardized video coding formats have a number of potential benefits over the non-standard ones like:

- Standards simplify inter-operability between encoders and decoders from different manufacturers. This is important in applications where each 'end' of the system may be produced by a different company, e.g. the company that records a DVD is typically not the same as the company that manufactures a DVD player [22].
- Standards make it possible to build platforms that incorporate video, in which many different applications such as video codecs, audio codecs, transport protocols, security and rights management, interact in well-defined and consistent ways [23].
- Many video coding techniques are patented and therefore there is a risk that a particular video codec implementation may infringe the patent(s). The techniques and algorithms required to implement a standard are well defined and the cost of licensing patents that cover these techniques, i.e., licensing the right to use the technology embodied in the patents, can be clearly defined [23].

1.3 Literature Review on Motion Estimation

Inter prediction is the heart of all video coding standards. It is used to remove temporal redundancy of the neighbouring frames of a video sequence and enable higher compression rates. In inter prediction, first, each frame is divided into blocks. After that, instead of directly encoding each block, the encoder will try to find a block similar to the current encoding block in the previously encoded frame, referred to as a reference frame. This process is done by a block matching algorithm. If the algorithm succeeds, the block could be encoded by a vector, which is known as the motion vector (MV) and with some prediction error. MV indicates the position of the matching block at the reference frame. The process of MV determination is called ME.

The basic principle of ME is shown in Fig. 1.1. The ME involves two steps namely, integer ME (IME) and fractional ME (FME). In IME, the search operation is carried out at integer

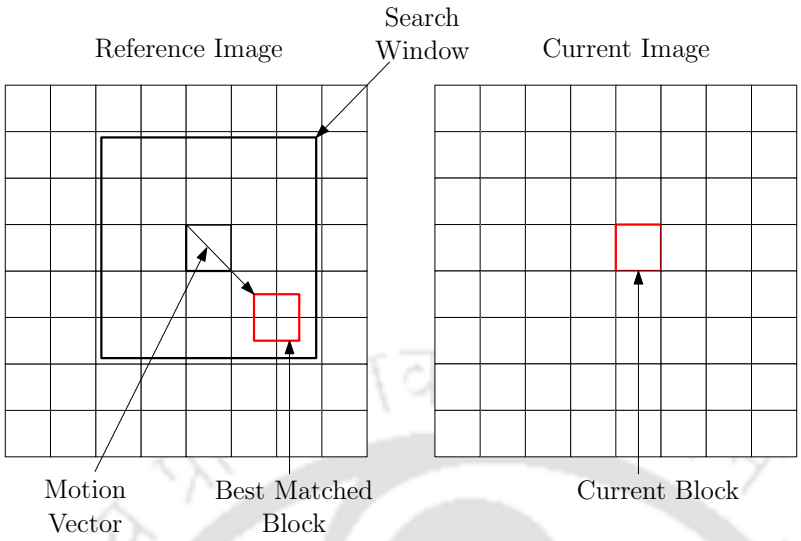


Figure 1.1: Basic Principle of Motion Estimation

pixel positions and generates the integer MV. In FME, the search is carried out at fractional pixel position using integer MV as a seed.

Various ME algorithms have been proposed in the literature since the early 1980's. The best possible ME algorithm is the full or exhaustive search (FS) algorithm. In this algorithm, each possible location in the search window is checked and the corresponding cost function is computed. This leads to the best possible match block in the reference frame. As compared to the other ME algorithms, the exhaustive search algorithm has the highest peak signal-to-noise ratio (PSNR). However, this involves the highest computations among all ME algorithms. This is due to the fact that the larger search window requires a greater number of computations. To speed up the exhaustive search, since 1981 various fast ME algorithms have been developed.

The fast ME algorithms are divided into three categories: (i) reduction in search points [1–3, 24–30] (ii) early termination [17, 31–34] and (iii) transform based ME [35–39]. In the first category, the research was focused on the reduction in the number of search points using different types of search patterns. The second category algorithms were mainly focused on developing the different type of threshold to speed up the ME process. In transform based ME algorithms, the first frames are transformed using a special type of transform and after that, a one or two-bits/pixel frame is generated and then ME is performed on one or two-bits/pixel

frame.

The first fast ME algorithm is the Three Step Search (TSS) algorithm [1], which comes under the category of reduction in search points. The search pattern of the TSS is shown in Fig. 1.2. As the name implies this algorithm is executed in three steps. The search starts at the center of the search window. In the first step, it searches eight neighboring points around the center of the search window with step size 4 and picks the one with the minimum cost function. The point with the minimum cost function is now set as the new origin and repeat the same procedure using a step size of 2. In the third step, the same procedure is repeated using a step size of 1 and the one with minimum cost function is considered the best match. TSS checks only 25 blocks in a search window of size ±7.

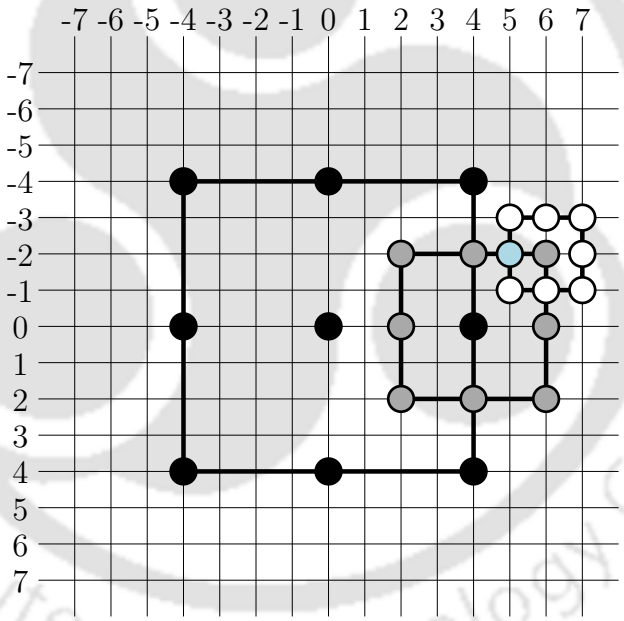


Figure 1.2: Illustration of the TSS algorithm [1]

As the name implies, the TSS algorithm is completed in three steps. So, this algorithm works for a small search window size i.e, the window size is restricted and for static blocks also it requires three steps to find the best-matched block. Thus, to overcome these limitations, after several studies, a new diamond search (DS) [2] ME algorithm was proposed which efficiently works for all types of search window sizes and takes fewer steps for static blocks. This algorithm gives better PSNR and requires fewer search points. The DS algorithm uses two types of

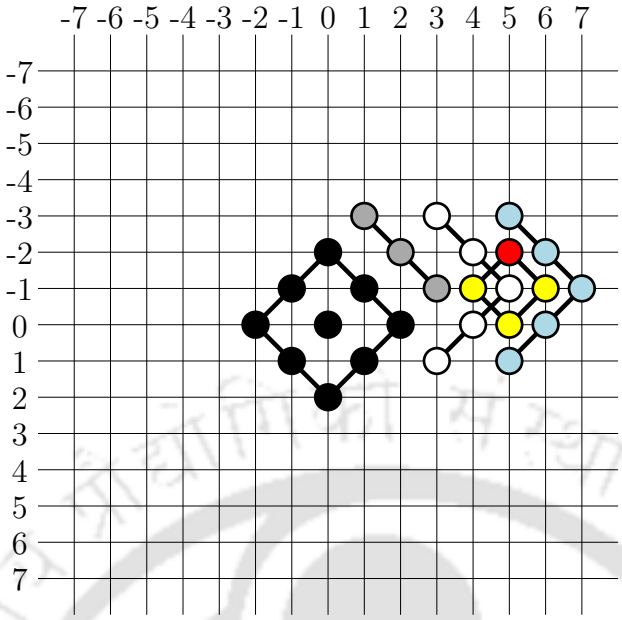


Figure 1.3: Illustration of the DS algorithm [2]

diamond search patterns: (i) large diamond search (LDS) and (ii) small diamond search (SDS). The DS algorithm flow is explained in the following steps:

Step 1 : In the first step, LDS pattern is used, in which eight positions are checked around the starting point of the search using a step size of two as shown in Fig. 1.3. Depending upon the position of minimum distortion block, the search process will proceed to step 2 or step 3.

Step 2 : With the position of minimum distortion block in the previous search step as the center, a new LDS is formed. If minimum distortion block in the previous step is at the corner of the LDS, then five new positions are checked otherwise three new positions are checked. This step is repeated continuously until the minimum distortion block is found at the center or search reaches the boundary of the search window. If the minimum distortion block is still at the center point of the newly formed LDS or searches reached the boundary of the search window, then go to Step 3.

Step 3 : In this step, the search pattern is switched from LDS to SDS. This step executes only once. SDS checks only four positions using a step size of one as shown in Fig. 1.3. The

position with minimum distortion is the final best-matched block.

To further increase the speed of the ME process, a new ME algorithm called Hexagon-based search (HEXBS) is proposed in [3]. As compared to DS algorithm, HEXBS algorithm speed up the ME process up to 80%. The HEXBS algorithm uses two types of hexagonal search pattern (i) large HEXBS and (ii) small HEXBS. In large HEXBS pattern, six positions are checked around the starting point of search and in small HEXBS, four positions are checked. The execution steps of HEXBS are the same as the DS algorithm. The only difference is in the second step is that only three new points are required to form new large HEXBS.

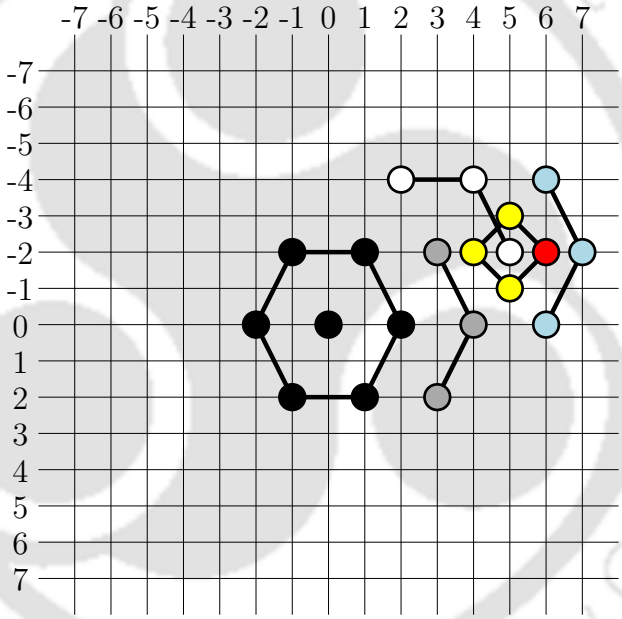


Figure 1.4: Illustration of the HEXBS algorithm [3]

The illustration of the HEXBS is shown in Fig. 1.4. HEXBS is computationally efficient as compared to TSS and DS algorithm with almost the same bit-rate and PSNR. In [40], an improved version of TSS is presented. In order to further reduce the computational complexity of ME, several search patterns are proposed in literature [24–30, 41–59].

Based on the early termination technique, several works have already been reported in the literature [17,31–34]. In [31], successive elimination algorithm was proposed. In the first step of this algorithm, the mean absolute difference (MAD) of the initial block is calculated. In order

to determine better matching candidate, one must be interested in those blocks whose MAD value is smaller than initial block MAD. In [31], Li et. al., proposed an inequality to speed up the MAD process. Now, MAD is computed only for those blocks, whose norm sum satisfy the proposed inequality. After that, various threshold based fast ME algorithms such as [17,32–34] have been proposed to further reduce computational complexity of IME.

In [4], the first transformed based fast ME algorithm is presented. In this algorithm, initially multi-bits/pixel of the video sequence is converted into one-bit/pixel video sequence by using a special type of transform and then conventional search strategies are applied on the one-bit/pixel video sequences. The procedure for converting 8-bit frames to 1-bit frames is shown in Fig. 1.5. This algorithm substantially reduces the hardware complexity and power consumption of ME. In [60], the multi-bit/pixel of video sequence is converted into two-bit/pixel video sequence to improve the video quality as compared to [4]. After that, various attempts have been made in [35–39] to reduce computational power further.

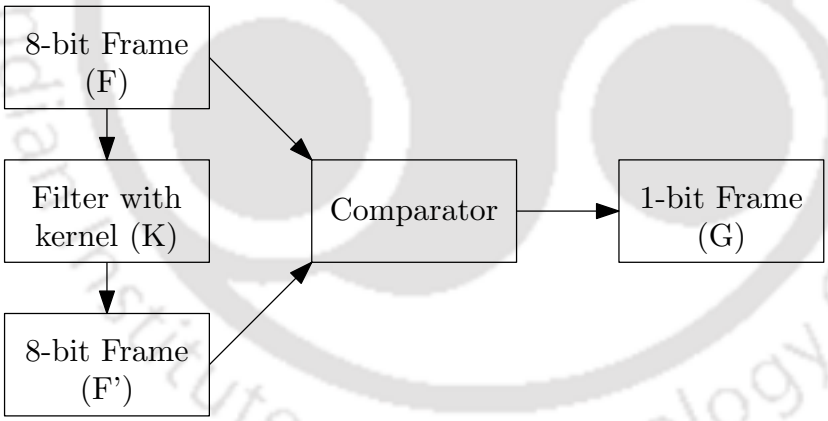


Figure 1.5: Operation for converting 8-bit frames to 1-bit frames [4]

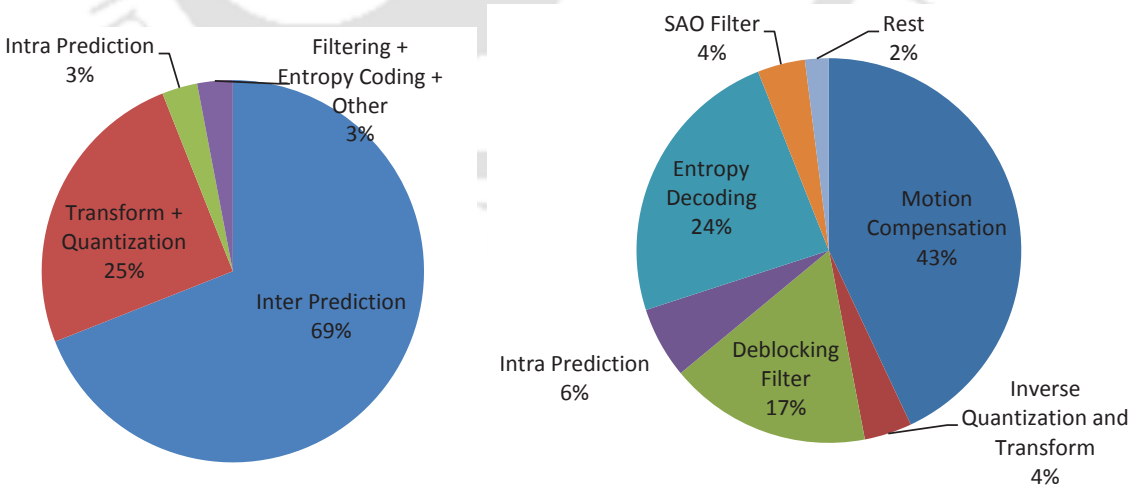
Since mid 1980's various VLSI architectures for ME algorithms have been proposed. The architecture reported in [61–72] supports only IME, the architectures reported in [73–78] supports only FME and the architectures reported in [16,79–81] supports both. In [79], a family of VLSI architectures and chip implementations for the motion compensation FS block-matching algorithm is proposed. This is the first ever VLSI architecture of ME. It is suitable for real-time video operations. Two sets of data flow designs are used in this architecture. The architectures

are based on data-flow designs which allow sequential inputs but perform parallel processing with 100% efficiency. Most of the architectures mentioned above employ the data reuse technique and pixel truncation to reduce the hardware cost.

However, most of these algorithms and architectures are mainly designed to support the previous standards. Due to the larger block size, asymmetrical block partition and larger tap interpolation filters these architectures cannot be used directly in the HEVC standard. Although some of these architectures are suitable for HEVC, they consume more power and hence are not suitable for portable devices. Hence, in this thesis, we have proposed a few computationally efficient algorithms and corresponding low-power architectures for HEVC.

1.4 Motivation

The HEVC standard improves the coding performance significantly by adopting many advanced techniques such as large size blocks for ME/MC (up to 64 × 64 pixels), directional prediction modes (up to 34) for different PUs in intra prediction, non-rectangular motion partition, Quad-tree partition for prediction block as well as transform block, large size transforms up to 32 × 32, in loop deblocking filter and SAO filter, WPP, larger size interpolation filter



(a) (b)

Figure 1.6: Average Time Distribution: (a) Encoder [5] (b) Decoder: Random Access Configuration [6]

(8-tap and 7-tap) and improved context-based adaptive binary arithmetic coding(CABAC) techniques. The aforementioned features enable the HEVC to improve coding efficiency by 50% compared to H.264 at the cost of a substantial increase in complexity. The complexity distribution of encoder and decoder is shown in Fig 1.6. It is clear from Fig. 1.6 that inter prediction/MC is the most time-consuming block of CODEC. ME is the most computationally intensive block in video CODEC. In HEVC, the complexity further increases due to larger PU sizes, asymmetrical PU partition, larger size interpolation filters and quad-tree PU partitioning.

However, HEVC is emerging as the best option for higher resolution video sequences and video on portable devices due to its low bit-rate and super visual quality. Several works so far have been reported on fast ME algorithms and their corresponding VLSI architectures suitable for HEVC. The design of a computationally efficient ME algorithm and architecture for low-power applications has remained a topic of intense research over the last few years. In this thesis, we proposed a few computationally efficient ME algorithms and architectures for low-power applications without much degradation in video quality.

1.5 Contributions and Overview of the Thesis

The focus of this thesis is on the reduction of computational complexity and developing corresponding low-power VLSI architectures for the HEVC. Throughout the thesis, we made an attempt to reduce the computational complexity of the HEVC without compromising on the quality and bit-rate. The thesis is organised as follows:

Chapter 1

The introductory chapter provides a brief description of ME principle and also provides the literature review of the existing algorithms and architectures for various video coding standards. The need for efficient video compression and problem formulation are also presented in the motivation section.

Chapter 2

The purpose of this chapter is to give the reader a comprehensive understanding of video processing and various video coding standards with a particular emphasis on HEVC. A detailed study on inter prediction process is also presented in this chapter.

Chapter 3

This chapter presents a detailed study of the pixel truncation, adaptive search range, the behaviour of the integer MV, depending upon the depth of the previously encoded block skipping some of the depth of the current block and the effect of removing some of the PU sizes. Based on the detailed study of above-mentioned techniques, a computationally efficient IME algorithm for HEVC video coding standard is proposed. Simulation results demonstrated that the proposed algorithm requires less number of search points compared to the best existing algorithms.

Chapter 4

In this chapter, a detailed work on the development of a power-efficient IME architecture for HEVC is presented. Further, the implementation details of the various blocks presented in the proposed architecture are presented in this chapter. The proposed architecture has been designed using standard 90 nm CMOS technology and simulation studies demonstrate that the proposed architecture offers low-power and less area compared to the existing architectures.

Chapter 5

This chapter presents a detailed study on the behaviour of the final MVs. This chapter showcases a detailed study of various possible FME algorithms. In addition, a VLSI architecture suitable for low-power devices is also presented. Simulation results have demonstrated that the proposed algorithm requires fewer search points compared to the existing FME algorithms. The proposed architecture has been designed using standard 90 nm CMOS technology and compared to the existing architectures.

Chapter 6

This last chapter provides major conclusions of the work presented in this thesis and outlines a few directions for future work.

2

High Efficiency Video Coding Standard

Contents

Digital video is the most important part of our lives in today's world. It plays a significant role in the field of entertainment, medical, engineering and technology. Due to the rapid growth in technology, the demand for high-quality video is further increasing. The high-quality videos require large bandwidths and more storage space. In order to enable transmission and storage of such high-quality videos, the size of the signal needs to be reduced using compression. This chapter presents the basic principles of digital video representation and video encoding. Also, a brief overview of international video coding standards and associated video encoders is outlined. Finally, the fundamental concepts and the main building blocks of the state of the art video encoder are presented.

2.1 Digital Video

A video is formed by the sequence of pictures (frames), which are presented in a specific interval of time. The number of pictures per second is called the frame rate. The frame rate ranges from 16 to 300 pictures per second [82]. PAL and Sequential Colour with Memory (SECAM) standard operate at 25 picture/s, while the NTSC standards use 29.97 pictures/s [83]. The minimum frame rate to achieve a comfortable illusion of a moving image is about 24 pictures per second. Video can be divided into analog and digital types. Analog video is represented by an analog signal, which is captured by progressive or interlaced scanning using an analog camera. The examples of analog video are PAL and NTSC. A digital video represents a visual scene in a binary format in which intensities and colors of the scene are specified with strings of bits (logical 0s and 1s). Each picture in a video sequence is formed by the two-dimensional ($M \times N$) array of picture elements (pixels) with M and N being the number of rows and columns respectively. If a picture is monochrome, i.e., has only one single color component, the picture consists of a single sample array. For representation of color pictures, each pixel is composed using three colors (R, G, and B). Correspondingly, a color picture consists of three intensity arrays, with one array for each component. For normal application, each color component is represented with an 8-bit integer. For specialized applications such as medical, broadcast, surveillance, and studio editing, etc., each color component is represented by more

than 8 number of bits. The total number of pixels in a frame is called resolution. A standard definition (SD) frame contains 720×480 pixels per frame, whereas an UHD uses 3840×2160 pixels per frame [84].

2.1.1 Frames and Fields

The raster scan process was first time introduced in analog video processing. It is of two types: progressive and interlaced scans as shown in Fig. 2.1. A digital video can also be progressive or interlaced, depending on the capturing process. In a progressive video, a frame is captured at one instant, whereas, in interlaced video, a frame is formed by two fields, each having half vertical resolution and captured at a field interval. The field containing the first line and following alternating lines in a frame is called the top field. The field containing the second line and following alternating lines in a frame is called the bottom field.

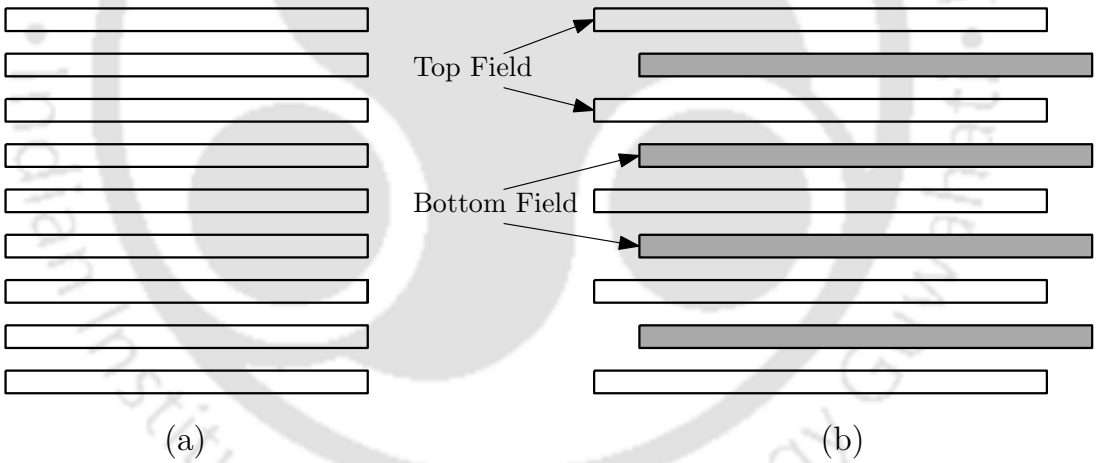


Figure 2.1: Frames and fields in a progressive and interlaced video: (a) Lines of a progressive frame. (b) Lines of the top and bottom fields of an interlaced frame.

2.1.2 Color Space and Sampling Formats

Each pixel of a color video is represented by the relative proportion of Red, Green, and Blue (RGB). In RGB color space representation, all the three colors are equally important and hence all colors are stored at the same resolution. But, it is possible to represent a color image more efficiently by separating the luminance from color information. The main reason for

separating the luminance part from color information is that the human visual system (HVS) is less sensitive to the chrominance as compared to the luminance portion. So, chrominance component can be represented with a lower resolution without affecting the visual perception. The Y:Cb:Cr color space representation is an efficient way to represent color images. The Y:Cb:Cr is composed of three components: one luminance (Y) and two chrominance or chroma (Cb and Cr). The conversion of Y, Cb and Cr from RGB and vice-versa can be represented as follows:

$$\begin{aligned} Y &= 0.299R + 0.587G + 0.114B \\ Cb &= 0.564(B - Y) \\ Cr &= 0.713(R - Y) \end{aligned} \tag{2.1}$$

$$\begin{aligned} R &= Y + 1.402Cr \\ G &= Y - 0.344Cb - 0.714Cr \\ B &= Y + 1.772Cb \end{aligned} \tag{2.2}$$

It may be noted the HVS is more sensitive to the error of luminance than to that of chrominance. Thus, this property can be exploited to make more compression on video data. As a result, the Cb and Cr components can be represented with a lower resolution than Y. As an example, Cb and Cr can be downsampled to a one-fourth size of Y. This reduces the amount of data required to represent the chrominance components without having an observable effect on visual quality. A video standard usually supports several sampling patterns for Y, Cb and Cr. The common sampling patterns are 4:4:4, 4:2:2, 4:1:1 and 4:2:0 as shown in Fig. 2.2. The numbers in the ratio $N_1 : N_2 : N_3$ indicate the relative sampling rates of each component in the horizontal direction, Here, N_1 represents the number of Y samples in both odd and even rows, N_2 the number of Cb and the number of Cr samples in an odd row and N_3 the number of Cb and the number of Cr samples in even row. Among all the patterns, the 4:2:0 sampling pattern

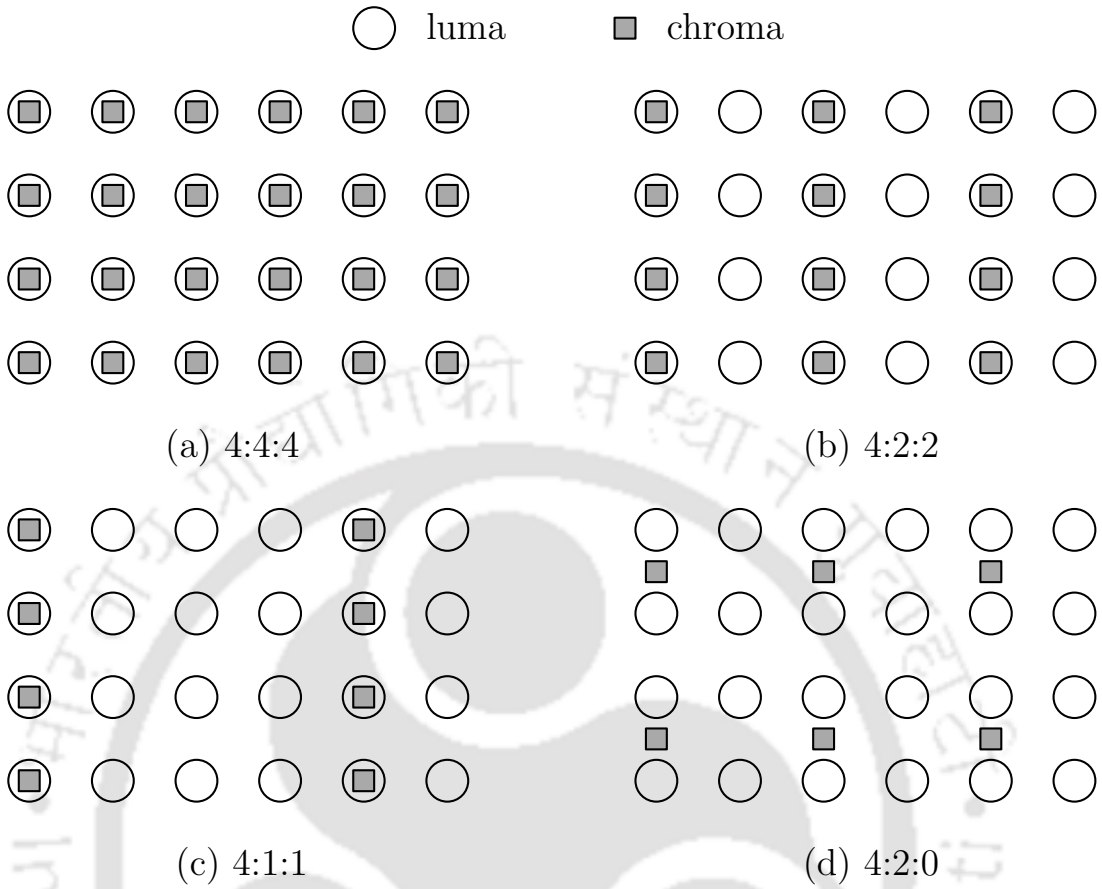


Figure 2.2: 4:4:4, 4:2:2, 4:1:1 and 4:2:0 sampling patterns.

is the most popular which is widely used for consumer applications such as video conferencing, digital television and digital versatile disk (DVD) storage. In 4:2:0 sampling pattern, for every four luminance samples in an odd row there are two Cb samples and two Cr samples, but for every four luminance samples in an even row, there are no Cb and Cr samples. The 4:2:2 and 4:4:4 patterns are used for high-quality color reproduction.

2.1.3 Video Formats

The video compression standards like H.264/AVC and HEVC can be used to compress a wide variety of video formats. The most common video formats used in transmission and storage are common intermediate, SD, HD and UHD. The common intermediate format (CIF) was first defined in the H.261 standard. The most commonly used CIF video formats are given in Table 2.1. The possibility of frame resolution depends on the application and available storage or

Table 2.1: Most Common CIF video formats

Format	Frame size in pixels (W×H)	Frame rate (Hz)
Sub-QCIF	128×96	24, 29.97
Quarter CIF (QCIF)	176×144	24, 29.97
CIF	352×288	24, 29.97
4CIF	704×576	24, 29.97

Table 2.2: Most Common SD video formats

Format	Frame size in pixels (W×H)	Frame rate (Hz)	Scanning type
480i	720×480	24, 25, 30, 50, 60	Interlaced
576i	720×576	24, 25, 30, 50, 60	Interlaced

transmission capacity. For example, 4CIF is appropriate for standard-definition television and DVD-video; CIF and QCIF are popular for video conferencing applications; QCIF or SQCIF are appropriate for mobile multimedia applications where the display resolution and the bit-rate are limited.

SD is the most widely used format for digital coding of video signals for television production. The two most common SDTV signal types are 576i, which is used by PAL systems; and 480i based on the American NTSC system as shown in Table 2.2.

HD video is a video of higher resolution and quality compared to SD. Generally, any video image with considerably more than 480 vertical lines (North America) or 576 vertical lines (Europe) is considered HD. Some of the most common HD video formats are given in Table 2.3. The UHD video sequences are the state of the art video sequences. As the name suggests, these video sequences have higher resolution than HD video sequences. Some of the UHD video formats are tabulated in Table 2.4.

Table 2.3: Most Common HD video formats

Format	Frame size in pixels (W×H)	Frame rate (Hz)	Scanning type
720p/HD Ready	1280×720	24, 30, 50, 60, 72	Progressive
1080i/Full HD	1920×1080	25, 30	Interlaced
1080p/Full HD	1920×1080	24, 25, 30, 50, 60	Progressive
1440p/ Quad HD	2560×1440	24, 25, 30, 50, 60	Progressive

Table 2.4: Most Common UHD video formats

Format	Frame size in pixels (W×H)	Frame rate (Hz)	Scanning type
2000	2048×1536	24, 60	Progressive
2160p/4K UHD	3840×2160	60, 120	Progressive
4000p	4096×3072	24, 30, 60	Progressive
4320p/ 8K UHD	7680×4320	60, 120	Progressive

2.2 International Video Coding Standards

Since 1984, several video coding standards have been developed. Mainly two international societies, namely the MPEG of ISO/IEC and VCEG of ITU-T are working on video coding standardization. The details of a few of the popular video coding standards are presented in Table 2.5.

2.2.1 H.120

H.120 was the first digital video compression standard. It was developed by the ITU-T formerly known as CCITT in 1984 [18]. Version 1 (1984) featured conditional replenishment, differential pulse-code modulation, scalar quantization, variable-length coding and a switch for sampling. Version 2 (1988) added motion compensation and background prediction. It was designed to support the target bit rate of 1544 kb/s for the NTSC and 2048 kb/s for the PAL [19]. As the videos based on this standard do not contain adequate quality, this standard is rarely used in practice.

2.2.2 H.261

H.261 is the first member of the H.26x family developed by the ITU-T (VCEG) and is the first video coding standard employed in some of the earlier practical systems. The first draft of this video coding standard was released in 1988 [85]. The basic processing unit of the design is called a macroblock (MB) and H.261 is the first standard in which the MB concept is used. Each MB consists of a 16×16 array of luma samples and two corresponding 8×8 arrays of chroma samples, 8×8 DCT, scalar quantization, and a YCbCr color space. It was originally

Table 2.5: Standard-specific key characteristics of motion estimation and compensation

Standard	H.261	MPEG-1	H.262 / MPEG-2	H.263	MPEG-4	H.264/AVC	VC	H.265/HEVC
Year of standardization	1988	1992	1994	1995/96	1998	2003	2006	2013
Organizations	ITU-T VCEG	ISO/IEC MPEG	ITU-T VCEG and ISO/IEC MPEG	ITU-T VCEG	ISO/IEC MPEG	ITU-T VCEG and ISO/IEC MPEG	Microsoft	ITU-T VCEG and ISO/IEC MPEG
MVi range	[-15, 15]	[-512, 511.5]	[-2048, 2047.5]	[-16, 15.5], [-31.5, 31.5]	[-16, 15.5] to [-1024, 1023.5]	[-2048, 2047.75]	[-64, 63.75]to [-1024, 1023.75]	[-8192, 8191.75]
MVj range	[-15, 15]	[-512, 511.5]	[-2048, 2047.5]	[-16, 15.5], [-31.5, 31.5]	[-16, 15.5] to [-1024, 1023.5]	[-64, 63.75] to [-512, 511.75]	[-32, 31.75] to [-256, 255.75]	[-8192, 8191.75]
MVP	non-median	non-median	non-median	median	median	median	median	median
Supported modes	1	1	1	1 and 4	1 and 4	1-7	1 and 4	1-7, 64 × 64 and 32 × 32
Interpolation	-	Bilinear	Bilinear	Bilinear	8-tap FIR/ Bilinear	6-tap FIR/ Bilinear	Bicubic, Bilinear	8-tap FIR/ 7-tap FIR
Bidirectional prediction	no	yes	yes	yes	yes	yes	yes	yes
No. of reference frames	0-1	0-2	0-2	0-2	0-2	0-16	0-2	0-16
Weighted prediction	no	no	no	no	no	yes	no	yes
Block size	fixed	fixed	fixed	fixed	fixed	Variable	variable	variable
Bit-rate	64-1920 kbit/s	8 kbit/s - 100 Mbit/s	2 - 100 Mbit/s	5 kbit/s - 16 Mbit/s	5 kbit/s - 1 Gbit/s	5 kbit/s - 960 Mbit/s	96 kbit/s - 135 Mbit/s	up to 340 Mbit/s
Entropy coding	-	Huffman	Huffman	Huffman	-	CAVLC and CABAC	-	CABAC
ME/MC accuracy	integer pixel	1/2-pixel	1/2-pixel	1/2-pixel	1/4-pixel	1/4-pixel	1/4-pixel	1/4-pixel
Applications	Video-conferencing, Video telephony	Video-CD, MP3	DVD-Video, HDV, Blu-ray Disc and Broadcast TV	Video-conferencing, Video telephony	broadcast television, Video telephony, Video on Mobile Phones	Blu-ray, HD DVD, Video on Mobile Phones, Digital Video Broadcasting,CCTV (Closed Circuit TV) and Video Surveillance	Blu-ray Discs, Windows Media	Video on Internet, HDTV broadcast, UHDTV, Video on Mobile Phones

designed for transmission over ISDN lines to support target bit rates of 64-2048 kbps. In fact, all subsequent international video coding standards such as MPEG-1 Part 2, H.262/MPEG-2 Part 2, H.263, MPEG-4 Part 2, H.264/MPEG-4 Part 10, and HEVC have been developed based on the H.261 design [19].

2.2.3 MPEG-1

MPEG-1 was developed by ISO/IEC JTC1 SC29 WG11 and published as ISO/IEC 11172 in 1993 [86]. This standard provides lossy compression of video and audio data. MPEG-1 most commonly uses Source Input Format (SIF) resolution: 352×240 and 352×288 with a bit-rate less than 1.5 Mbit/s. MPEG-1 also supports resolutions up to 4095×4095 (12-bits), and bit-rates up to 100 Mbit/s [86]. This standard provides superior quality video compared to H.261 when operated at higher bit rates. For motion compensation, the concept of bidirectional prediction was first employed in MPEG-1. For the very first time, MPEG-1 supported the fractional pixel-accuracy (only half pixel).

2.2.4 H.262/MPEG-2 Part 2

H.262/MPEG-2 Part 2 video coding standard was jointly developed by ITU-T VCEG and ISO/IEC MPEG in 1994 [87]. It was formally known as ITU-T Recommendation H.262 and ISO/IEC 13818-2(MPEG-2). It is the second part of the ISO/IEC MPEG-2 standard. Although MPEG-2 Video is similar to MPEG-1, it provides support for interlaced video (an encoding technique used in analog NTSC, PAL and SECAM television systems). MPEG-2 video is not optimized for low bit-rates (less than 1 Mbit/s), but it outperforms MPEG-1 at 3 Mbit/s and above [88]. For consistency of the standards, MPEG-2 video decoders are fully capable of playing back MPEG-1 video streams.

2.2.5 H.263/H.263+/H.263++

H.263 is a video compression standard originally designed to provide a low-bitrate compressed format which is used for video conferencing. H.263 has superior video quality compared

to its prior standards at all bit-rates, by a factor of two [89]. It was developed by the ITU-T VCEG and the first version of H.263 was published in March 1996. It supports video resolution from 128×96 to 1408×1152 [89]. The encoding algorithm of H.263 CODEC was similar to H.261, however, coding efficiency has been improved due to new functions such as advanced prediction mode, syntax-based arithmetic coding mode and PB-frames mode etc. H.263 has two subsequent additions such as H.263+ (1998) and H.263++ (2000) to improve the original standard by adding some new features. DBF was introduced for the first time in H.263+ [89].

2.2.6 H.264/MPEG-4 AVC

H.264, also called AVC or MPEG-4 Part 10 standard. It is one of the most commonly used formats for the recording, compression and distribution of video content. H.264 was developed by the ITU-T VCEG together with the ISO/IEC JTC1 MPEG. The project partnership is known as the Joint Video Team (JVT). The first version of the standard was completed in May 2003 [90]. The basic functional elements of H.264/AVC such as motion prediction, the transformation of the motion-compensated error residual, and the quantization of the resultant DCT coefficients as well as their entropy encoding are similar to the previous standards, such as MPEG-1, MPEG-2, H.261, H.263, etc. H.264/AVC saves 50% bit-rate compared to MPEG-2 Part 2 [91]. In H.264/AVC, a number of new features are added which allows to compress video much more efficiently than older standards and provide more flexibility. Some of the key features are:

- Multi-picture inter-picture prediction
- Variable block-size MC
- Quarter-pixel precision for MC
- Weighted prediction (WP)
- A secondary Hadamard Transform (HT) for "DC" coefficients
- Adaptive encoder selection between the 4×4 and 8×8 transform block sizes

- Six-tap filtering for derivation of half-pixel luma sample predictions and Quarter-pixel motion is derived by linear interpolation of the half pixel values
- Multiple MVs per MB
- Spatial prediction from the edges of neighboring blocks for "intra" coding
- Flexible macroblock ordering (FMO), also known as slice groups, and arbitrary slice ordering (ASO)
- Support sample bit depth precision ranging from 8 to 14 bits per sample

2.2.7 HEVC/H.265

HEVC is the state of the art video compression standard, developed by the JCT-VC, a collaboration between the ISO/IEC MPEG and the ITU-T VCEG [21]. The first version of the HEVC standard was ratified in January 2013 and was published in June 2013. It is also commonly known as H.265 and MPEG-H Part 2. It is one of the several potential successors to the widely used AVC (H.264 or MPEG-4 Part 10). In comparison to AVC, HEVC offers about double the data compression ratio at the same level of video quality or substantially improved video quality at the same bit rate [92]. It supports resolutions up to 8192×4320, including 8K UHD. Some of the new and improved features supported by HEVC are [21, 93, 94]:

- Coding tree units (CTUs) which can use larger block structures of up to 64×64 samples
- Quadtree partitioning of prediction as well as transform units (TUs)
- Asymmetrical prediction units partitions
- Parallel processing tools such as WPP
- HEVC specifies 33 directional modes for intra prediction
- For the interpolation of fractional luma sample positions, 8-tap (Half) filter and 7-tap (Quarter) filter are used (Half and Quarter)

- A signed 16-bit range (32768 to 32767) for both horizontal and vertical motion vectors (MVs)
- HEVC allows for two MV modes: Advanced Motion Vector Prediction (AMVP) and Merge mode
- HEVC specifies four TUs sizes of 4×4, 8×8, 16×16, and 32×32 to code the prediction residual
- HEVC specifies two loop filters that are applied sequentially, with the DBF applied first and the SAO filter applied afterwards
- Profiles supporting bit depths beyond 10 bits per sample
- Uses high precision WP
- Transform skip block size flexibility

2.2.8 Future Video Coding

In October 2015, experts of MPEG and VCEG started the possibilities of new video compression technique with capabilities beyond HEVC (ITU-T H.265 — ISO/IEC 23008-2). After two years of exploration by experts of both groups, a new Joint Video Experts Team (JVET) between VCEG (Q6/16) and ISO/IEC JTC1 SC29/WG11 (MPEG) was created on 27 October 2017 to develop a new video compression with capabilities beyond HEVC (ITU-T H.265 — ISO/IEC 23008-2) [95]. The new standard targets to have 30-50% better compression rate for the same perceptual quality. This new standard also targets to support YCbCr 4:4:4, 4:2:2 and 4:2:0 with 10 to 16 bits per component, variable and fractional frame rates from 0 to 120 Hz, scalable video coding for temporal (frame rate), spatial (resolution), SNR, color gamut and dynamic range differences, stereo/multiview coding, panoramic formats, and still picture coding [96]. Encoding complexity is expected 10 times higher than of HEVC. JVET issued a final "Call for Proposals" in October 2017 and expects the final standard to be approved before the end of 2020 [96].

2.3 HEVC CODEC

Data compression involves a complementary pair of systems, a compressor (encoder) and a decompressor (decoder). The encoder converts the source data into a compressed form and the decoder converts the compressed data back into an original form. The encoder/decoder pair is often described as a CODEC(enCOder/DECoder). All the international video coding standards (MPEG-1/2, MPEG-4 Visual, H.261/3, H.264/AVC, VC-1 and HEVC) specifies decoding process and output bit stream (syntax and semantics) of the encoder. Provided that the encoder design is kept compatible with the decoder specifications and manufacturers can freely optimize encoding efficiency, quality, and speed.

The general block diagram of the HEVC CODEC is illustrated in Fig. 2.3. The HEVC works on the principle of the block-based hybrid video coding scheme, in which motion prediction (inter and intra) is followed by transform coding, quantization and variable length entropy

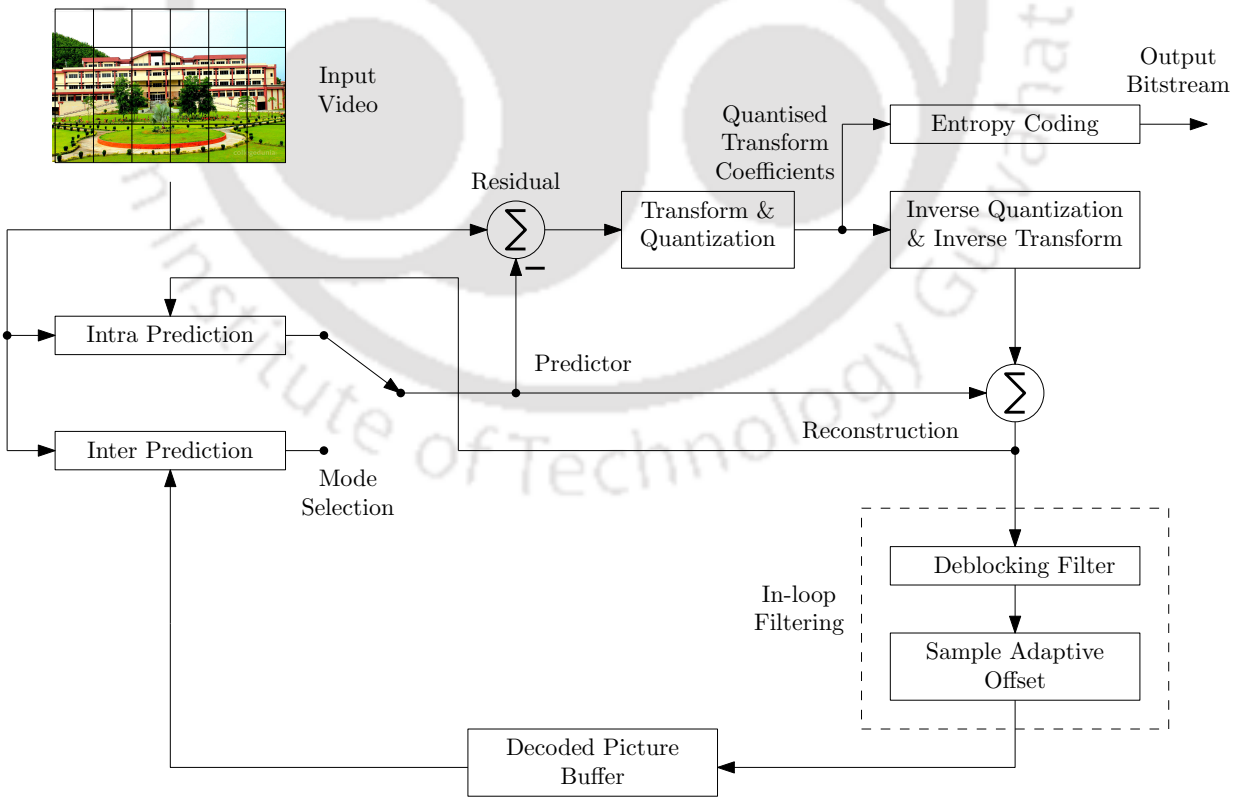


Figure 2.3: Generalized block diagram of the HEVC CODEC [7].

coding. First, the input video is divided into non-overlapping square blocks of the same size called CTUs, which is analogous to MBs in the H.264/AVC. Each CTU is predicted either by using intra or inter prediction. In intra prediction, blocks are predicted with the help of previously encoded blocks of the same picture and in inter prediction, blocks are predicted with the help of previously encoded pictures. Then, the residual error is calculated by subtracting the predicted block from the original block. The residual error is then transformed using DCT and resultant coefficients are quantized. On the one hand, the quantized DCT coefficients are converted into compressed bit-stream by CABAC entropy encoding. On the other hand, the quantized DCT coefficients are sent to inverse quantization and inverse DCT transform to generate the residual error back. The combination of residual block and predicted block are used to reconstruct the version of the original block. After that, the artifacts are removed by deblocking filter and amplitude level is increased if required by SAO filter. Finally, the reconstructed frame is stored in the decoded buffer and used as the reference frame for the next encoding frame.

In contrast to previous video coding standards, HEVC employs a quad-tree coding block partitioning structure that enables flexible use of large and small coding, prediction, and transform blocks. HEVC also allows improved intra prediction and coding, adaptive motion parameter prediction and coding, a new loop filter and an enhanced version of CABAC entropy coding. New high-level structures have also been incorporated in HEVC to aid parallel processing. In the following sections, the building blocks of the HEVC are briefly described.

2.3.1 Picture Partitioning

2.3.1.1 Coding Tree Blocks and Coding Tree Units

In HEVC, every picture is first divided into a sequence of CTUs as shown in Fig. 2.4. CTU is always a square block of the same size in a video sequence. CTU is the basic processing unit of the HEVC. It is analogous to macroblock and block in the previous standards. Each CTU consists of one block of Luma samples together with two corresponding blocks of chroma samples and the associated syntax elements as illustrated in Fig. 2.5. These blocks are called

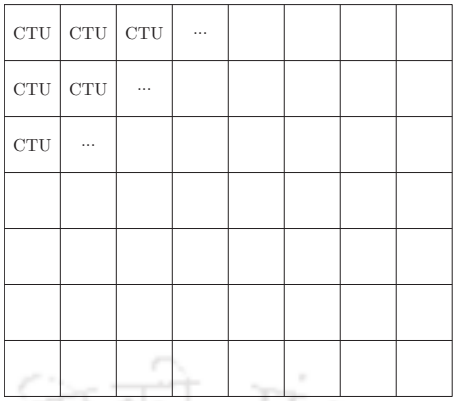


Figure 2.4: Example of a picture divided into CTUs.

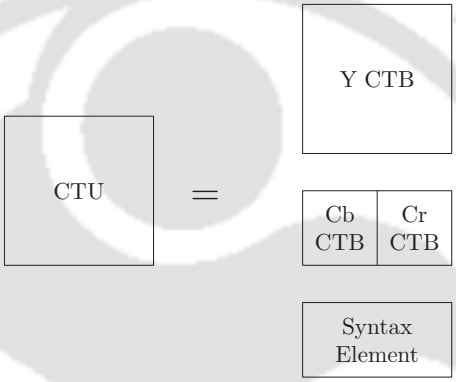


Figure 2.5: Example of a CTU divided into CTBs.

coding tree block (CTB). The possible size of the CTU is 16×16, 32×32 or 64×64 luma samples, which is specified in the main profile. The size of CTB is always the same as that of CTU.

2.3.1.2 Slice and Tile Structures

The slice is a part of the picture that can be decoded independently from the other slices of a picture. This means that no prediction (i.e. intra prediction or motion vector prediction) is performed from one slice to the other. The slice provides error robustness and supports parallel processing. A slice can either be the entire picture or a region of a picture. A slice consists of a sequence of one or more slice segments as shown in Fig. 2.6. The first segment of the slice, which includes the slice segment header is always an independent slice segment. It can be decoded independently from any other slice or slice segment in the same picture. Slice

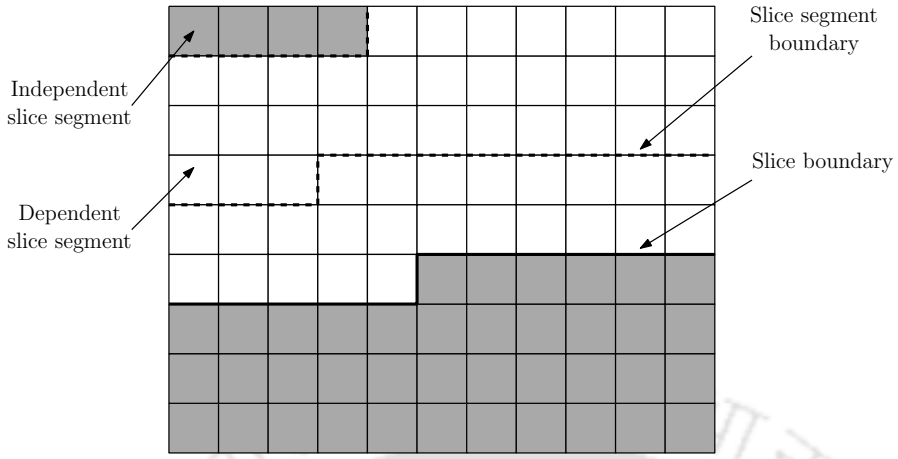


Figure 2.6: Example of slices and slice segments.

segments that follow the independent slice segment are denoted as dependent slice segments. A slice segment consists of a sequence of CTUs, which are processed in a raster scan order. Slice and its segments are not necessarily rectangular. The type of independent slice segment determines the type of dependent slice segment.

Based on the prediction methods of the CTUs slice are divided into three categories [97]. The different types of the slice are listed below:

I Slice :- Only intra prediction is applied for coding of CTUs.

P Slice :- Intra prediction, or inter prediction using a single reference picture from a reference picture list with a single motion vector per prediction partition can be applied.

B Slice :- Intra prediction, or inter prediction using one or two reference pictures from two reference picture lists with a single or two motion vectors per prediction partition can be applied.

In HEVC, the tile concept is used to organize the coding arrangement of CTUs in a picture in a more flexible way [97]. Tiles allow additional horizontal and vertical scan boundaries within a picture. Each area that is framed by such boundaries is called a tile. The boundaries modify the scan order of the CTUs in the picture. They further break prediction dependencies as intra

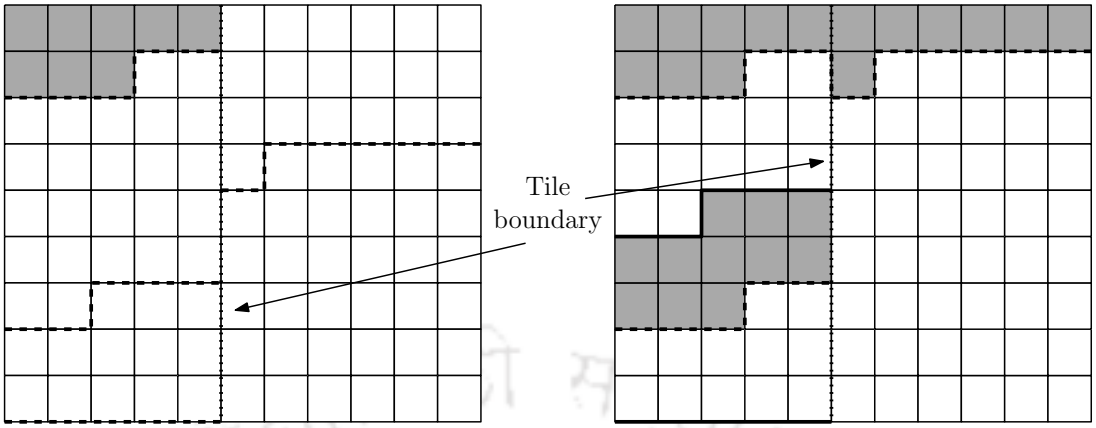


Figure 2.7: Example of tiles and slices.

prediction as well as MV prediction is disabled across tile boundaries [97]. Tiles are processed in tile raster scan order. A tile is a rectangular region which contains an integer number of CTUs. A tile may consist of CTUs contained in more than one slice as shown in Fig. 2.7. Similarly, a slice may consist of CTUs contained in more than one tile.

2.3.1.3 Coding Block, Coding Unit and Coding Tree Structure

A CTU can be split into multiple coding units (CUs) of variable sizes. The partition process is called quadtree partitioning and this quadtree is called coding quadtree. The quadtree partitioning scheme allows recursive splitting into four equally size nodes, starting from the CTU and stops when no further splitting is signalled in the bitstream (as determined by an encoder) or when the minimum CU size is reached [12]. The graphical representation of quadtree partitioning is shown in Fig. 2.8. Each leaf node of the quadtree is called CU. The CU is a square region of pixels, which shares the same prediction mode: intra, inter, or skip. The size of CU may be as large as the CTU or as small as 8×8 pixels. For color video format, similar to CTU, a CU consists of a square block of luma samples, the two corresponding blocks of chroma samples, and the syntax associated with these sample blocks. The luma and chroma sample arrays, which are present in a CU, are referred to as coding blocks (CB). The resulting quadtree partitioning is scanned in a Z-scan format [12]. The number in Fig. 2.8 represents the encoding order of CB.

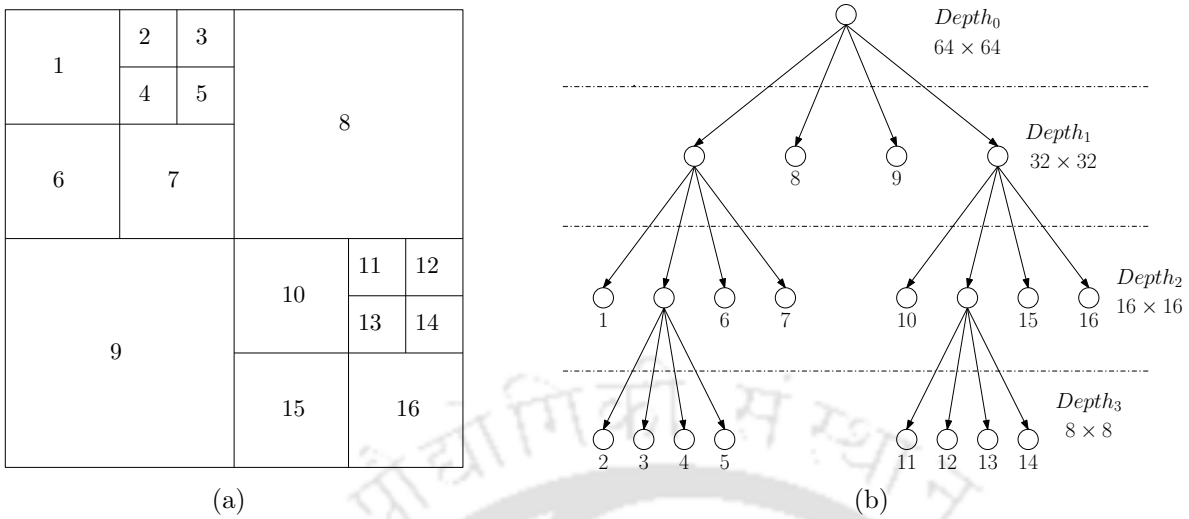


Figure 2.8: Coding Quadtree Structure [8].

2.3.1.4 Prediction Blocks and Prediction Units

For inter and intra prediction, CU can be partitioned into multiple prediction units (PUs) according to a partition mode. Similarly to CU, PU is also composed of one luma prediction block (PB) and two chroma PBs, together with the associated syntax. For each PU, a single set of motion parameters is signalled in the bitstream, which is used for motion-compensated prediction of the luma PB and the chroma PBs. In other words, PU is the basic processing unit of prediction. The possible size of PUs reported in HEVC are $2N \times 2N$, $N \times 2N$, $2N \times N$, $N \times N$, $(2N \times nU$ & $2N \times nD)$ and $(nR \times 2N$ & $nL \times 2N)$ with $N \in 4$, 8, 16 and 32 [7]. It can be noticed from Fig. 2.9 that the PU is not restricted to be square in shape. Each CU includes one, two or four PUs depending upon the partition mode of CU. Fig. 2.9 shows the eight partition modes that may be used to define the PUs for a CU. PU partitioning is divided into two parts: symmetrical and asymmetrical. The first-row partitioning of Fig. 2.9 is called symmetrical partitioning and the second-row partitioning is called asymmetrical partitioning. The asymmetrical PUs partitioning was used for the first time in HEVC [94]. The availability of asymmetrical PU sizes permits improved matching of boundaries of real objects in the picture. For an intra prediction CU, only square PUs are available i.e. $2N \times 2N$ and $N \times N$ as shown in Fig. 2.10. The partition mode $N \times N$ is only available when the CU size is equal to its minimum

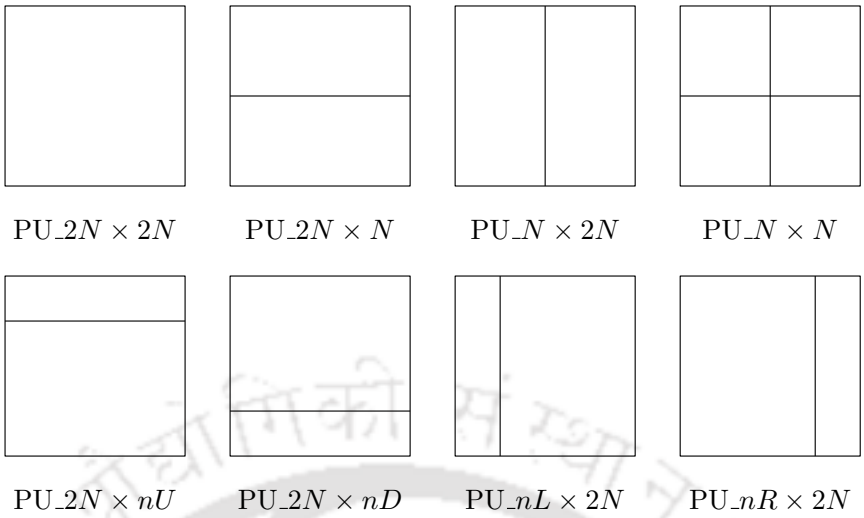


Figure 2.9: Possible Partition modes for inter PU [9]. For asymmetrical PUs partitioning, the indices U, D, L and R denotes up, down, left and right respectively with $n = N/2$.

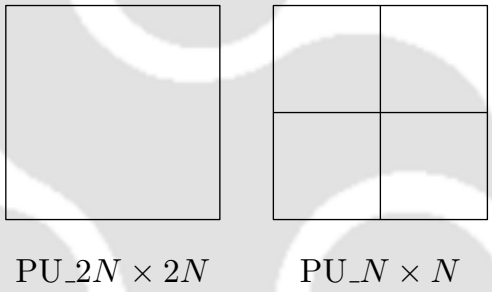


Figure 2.10: Possible Partition modes for intra PU [9].

value. In inter-prediction CUs, all eight PUs are available. The asymmetric partitioning modes are only supported for CU sizes greater than 8×8 luma samples. To minimize the worst case memory bandwidth of motion compensation, the 4×4 PU is prohibited and 4×8 and 8×4 PUs are restricted to use uni-directional prediction [12].

2.3.1.5 Transform Block, Transform Unit and Residual Quadtree Structure

In HEVC, the CU is the root of another quad-tree called residual quadtree (RQT). Each leaf of the RQT is called TU. A TU represents a square block of residual samples on which the same transform and quantization processes are applied. Each RQT is restricted by three parameters: (i) the maximum depth of the tree, (ii) the minimum allowed transform size and (ii) the maximum allowed transform size [98]. The maximum depth of RQT specified in intra

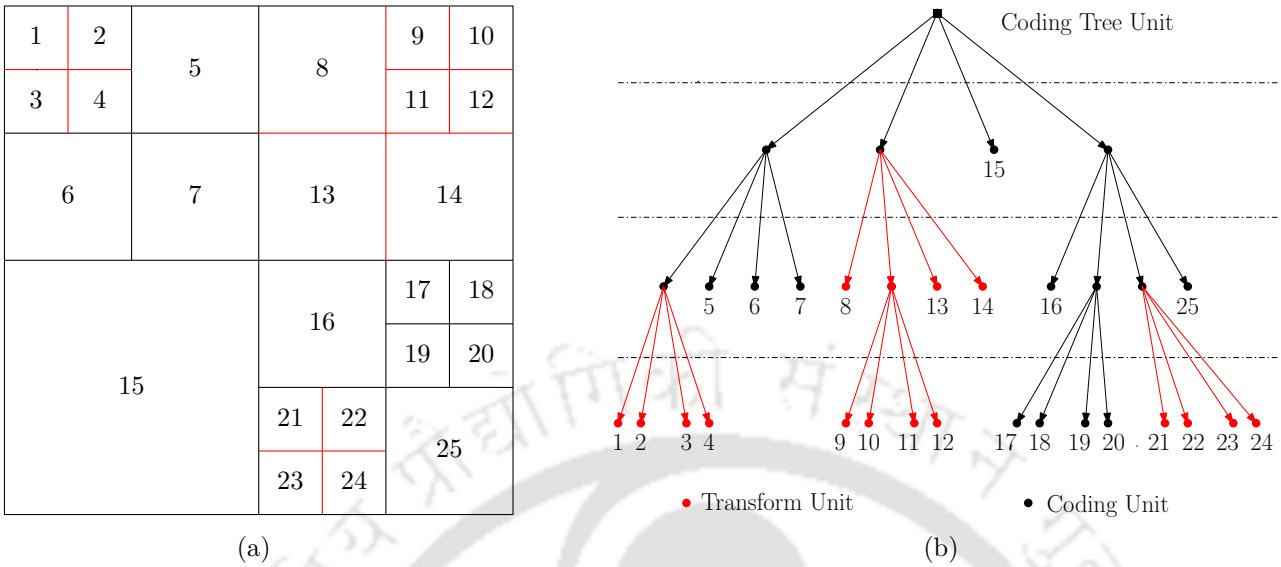


Figure 2.11: Residual Quadtree Structure [10].

only, random access main and low delay main is 3. The minimum and maximum transform sizes can vary within the range from two to five, i.e., block sizes from 4×4 to 32×32 samples. The graphical representation of RQT is presented in Fig. 2.11. The red line in Fig. 2.11(a) and Fig. 2.11(b) represents the TUs and the black line represents the CUs. TU is the combination of one transform block (TB) of luma sample, two corresponding TBs of chroma samples and syntax elements. The TB is a square region of residual picture samples which shares the same transform. The possible TB sizes are 4×4, 8×8, 16×16 and 32×32. In inter prediction, the size of TU can be larger than PU, i.e. the TU may contain PU boundaries. However, in intra prediction, the TU size cannot cross PU boundaries [93].

2.3.2 Rate Distortion Function

The Lagrangian based rate-distortion optimization (RDO) technique is used for prediction parameter or mode decision. In HEVC, the following relation is used for the prediction parameter or mode decision.

$$J = D + \lambda \times B \tag{2.3}$$

where, J is the rate-distortion (RD) cost for prediction parameter or mode decision, D is the total distortion between the current block and the best-matched candidate block and B is the total number of bits required to represent the prediction information. λ is the Lagrangian parameter, which is used to control the trade-off between D and B. As the value of λ decreases, the total bit-rate increases and vice-versa.

The total distortion is calculated by means of either the sum of square error (SSE) or the sum of absolute difference (SAD). Although SSE is the best method, SAD is mostly used due to its low complexity and its feasibility for hardware realization. The expression for SSE and SAD are present below

$$SSE = \sum_{i,j} d(i,j)^2 \tag{2.4}$$

$$SAD = \sum_{i,j} |d(i,j)| \tag{2.5}$$

where, d is the difference between the current and reference block, i and j are represents the position of the pixel in a block. The expression of d is

$$d = Current(i,j) - Reference(i,j) \tag{2.6}$$

In the fractional pixel position, the total distortion is calculated as the sum of the absolute transform difference (SATD). The expression for SATD is

$$SATD = (\sum_{i,j} |dT(i,j)|)/2 \tag{2.7}$$

$$dT(i,j) = H_m d(i,j) H_m^T \tag{2.8}$$

where, H_m is the HT matrix.

The Lagrangian parameter is calculated using the following expression

$$\lambda_{mode} = \alpha \times W_k \times 2^{((QP-12)/3)} \tag{2.9}$$

here, W_k is the weighted factor, which depends on the encoding configuration and Quantization Parameter (QP) offset value for the current picture in a group of pictures (GOP). The value of α can be calculated using the following expression

$$\alpha = \begin{cases} 1.0 - Clip3(0.0, 0.5, 0.05 \times number\_of\_B\_frames), & \text{for reference pictures} \\ 1.0, & \text{for non-reference pictures} \end{cases} \tag{2.10}$$

The SAD and SATD based cost function for prediction parameter decision at integer and the fractional pixel position is measured by the following expression

$$J_{pred,SAD} = SAD + \lambda_{pred} \times B_{pred} \tag{2.11}$$

$$J_{pred,SATD} = SATD + \lambda_{pred} \times B_{pred} \tag{2.12}$$

where, the value of λ_{pred} is

$$\lambda_{pred} = \sqrt{\lambda_{mode}} \tag{2.13}$$

2.3.3 Peak Signal-to-Noise Ratio

The objective testing is a cheaper and easier process to verify the quality of video signals as compared to subjective testing. Objective testing gives sufficient approximation results of corresponding subjective testing. The most common objective quality testing method used in video coding is PSNR. The PSNR is the relation of the mean squared error between the original signal (i.e. picture) and the corresponding reconstructed signal to the available maximum amplitude of the original signal given by

$$PSNR = 10log_{10}(\frac{A_{max}}{MSE})dB \tag{2.14}$$

where $A_{max} = 8^2 - 1$ for 8-bit video signal and MSE is the mean squared error.

2.3.4 Intra Prediction

Intra prediction is used to remove spatial redundancy in the slice. The intra coded slice does not require any reference frame i.e., the slice is coded without the help of other frames. In intra mode, CUs are predicted with the help of a reconstructed neighbouring sample of the same slice. Intra prediction is possible in I, P or B types of slices. In P or B slices both intra and inter prediction is possible. The random access picture is always comprised of I slices as they do not contain any reference picture.

2.3.4.1 Prediction Modes

In HEVC, a total of 35 intra prediction modes are available; out of which 33 are directional (angular) modes and remaining two are DC mode and Planar mode [99]. Table 2.6 shows the mode name with their corresponding intra prediction mode index. The possible intra prediction direction for the angular prediction mode is shown in Fig. 2.12. The mapping between intra prediction direction and intra prediction mode number is specified in Fig. 2.13. The 33 direction modes are divided into two parts according to their prediction direction: horizontal and vertical. The prediction modes 2-18 are called horizontal prediction modes, as they use previously encoded horizontal blocks in prediction. The modes 19-34 are denoted as vertical prediction modes. The intra prediction modes are available for prediction block of sizes from 4×4 to 32×32 samples.

Table 2.6: Intra Prediction modes

Mode Names	Intra Prediction Mode Number
Intra_Planar	0
Intra_DC	1
Intra_Angular	2-34

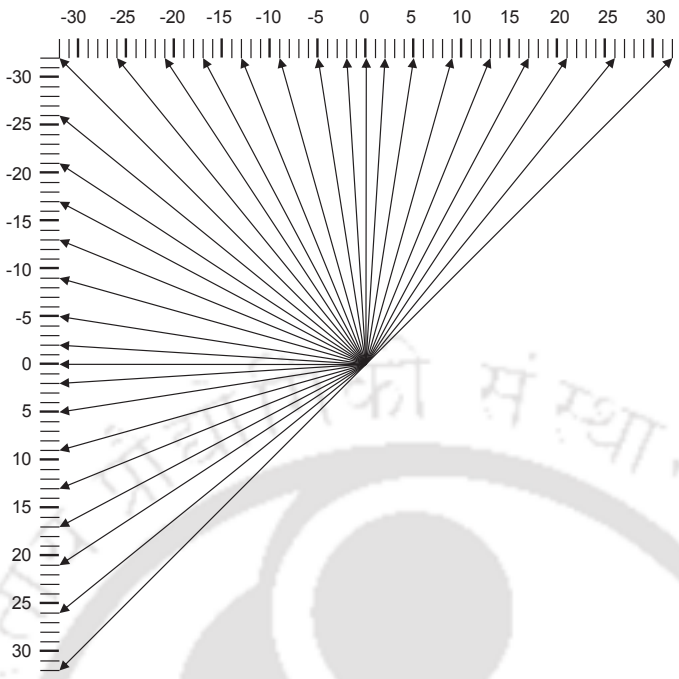


Figure 2.12: The 33 Intra Prediction Directions.

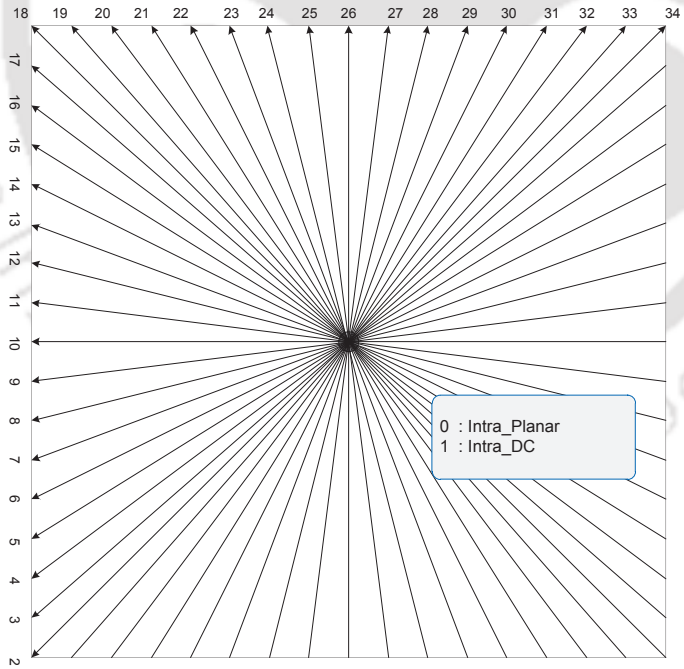


Figure 2.13: Intra Prediction Modes.

For the chroma component, the encoder selects the best chroma prediction modes among five modes such as planer, DC, horizontal, vertical and a direct copy of the intra prediction mode for the luma component [7]. The mapping between intra prediction direction and intra prediction mode number for chroma is shown in Table 2.7.

Table 2.7: Mapping between intra prediction direction and intra prediction mode number for chroma [7]

Intra chroma prediction mode	Intra prediction direction				
	0	26	10	1	X ($0 \leq X \leq 34$)
0	34	0	0	0	0
1	26	34	26	26	26
2	10	10	34	10	10
3	1	1	1	34	1
4	0	26	10	1	X

2.3.4.2 Filtering for Intra Prediction

In order to improve the likelihood of finding good prediction candidates, HEVC supports different filtering alternatives for pre-processing the reference samples prior to applying these in the actual intra prediction process. Similarly, some of the prediction modes include a post-processing step to refine the sample surface continuity on the block boundaries. Two types of filters are used for filtering the neighbouring samples. These two filters are [1, 2, 1] filter and bi-linear filter. The bi-linear filtering is conditionally used. The filtering process depends upon the given intra prediction mode and transforms block size. If the intra prediction mode is DC or the size of the transform is equal to 4×4, then the neighbouring sample is not filtered. If the distance between the given intra prediction mode and vertical mode (or horizontal mode) is larger than the predefined threshold, the filtering process is enabled. The predefined threshold is shown in Table 2.8, where nT represents the transform block size.

Table 2.8: Specification of Predefined Threshold for Various Transform Block Sizes

	nT = 8	nT = 16	nT = 32
Threshold	7	1	0

2.3.5 Inter Prediction

Inter prediction is used to remove the temporal redundancy of the video sequences. Inter prediction is also called motion compensated prediction as it predicts the current block with the help of the shifted area of the reference pictures. Here, motion between the current block and the reference picture is determined with the help of ME. In other words, ME generates the MV, which is the displacement between an area in the current picture and the previously encoded picture that is used as a reference. In HEVC, inter prediction is more generalized and achieves comprehensive improvement as compared to previous video coding standards. Inter prediction is possible in P and B types of slices. In P slice, blocks are predicted with the help of only uni-directional prediction, which is the previously decoded block. In B slices, blocks are predicted with the help of bi-directional prediction.

2.3.5.1 Motion Estimation

ME is used to find the best-matched block in the previously encoded reference pictures or in other words, it removes the temporal redundancy in the pictures. The reference picture can be a past or a future picture in the display order. Depending upon the slice type, ME can be uni-predicted or bi-predicted. In P slice, uni-prediction is used and only one MV is required. In B slice, bi-prediction is used and two MVs are required. The MV in bi-prediction can be either from one list of pictures or a combination of two lists of pictures.

The basic principle of ME is shown in Fig. 2.14. Like the previous video coding standards, ME in HEVC is also executed in two steps: IME and FME. The FME again is a two-step process: half pixel FME and quarter pixel FME. The IME is used to find the best-matched at integer pixel position. The FME is used to find the best-matched block at fractional pixel position (up to quarter pixels) around the final integer MV. For finding the pixel value at the fractional position, interpolation filters are used. Different interpolation filters are used for luma and chroma samples. For the luma filtering, an 8-tap DCT-based interpolation filter is used for 1/2 precision samples and a 7-tap DCT-based interpolation filter is used for 1/4 precision samples. Similarly, four different 4-tap DCT-based interpolation filters are used for chroma

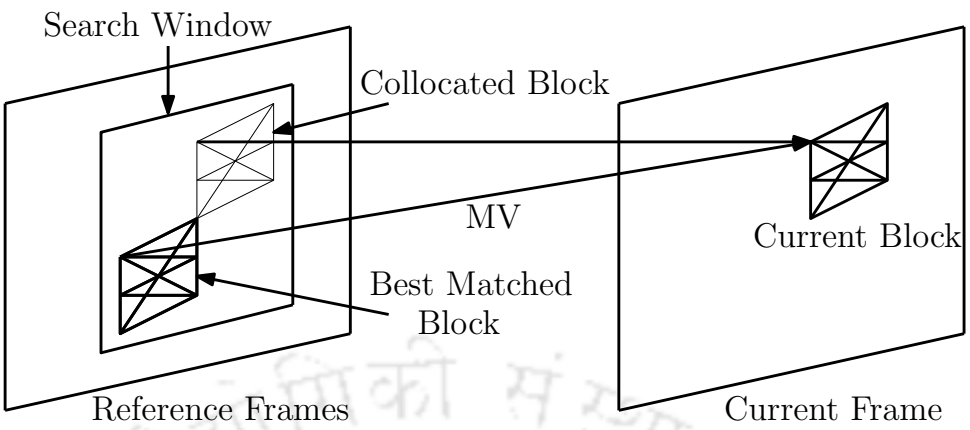


Figure 2.14: Basic Motion Estimation Principle [11].

interpolation. Since 1981, several IME algorithms have been developed. The full search ME algorithm is the best possible algorithm for finding the best-matched block. In HEVC, Test Zonal Search (TZS) algorithm is used.

2.3.5.2 Prediction Modes

Each inter-coded PU has a set of motion parameters for one or two reference picture lists. It consists of MV, reference picture index and reference picture list. When a CU is coded with skip mode, the CU is coded as one PU that has no significant transform coefficient and the motion parameters are obtained by the merge mode. The merge mode is to find out a neighbouring inter-coded PU, which has the most similar motion parameters from the spatially neighbouring and temporally neighbouring PUs and transmit its corresponding index, indicating the chosen candidate. The merge mode can be applied to any inter-coded PU. In any inter-coded PUs, the encoder can use either merge mode or explicit transmission of motion parameters.

2.3.5.3 Advanced Motion Vector Prediction

The correlation between the current blocks and the neighbouring blocks of the current picture or earlier coded pictures is very high. This is because neighbouring blocks are likely to correspond to the same moving object with similar motion and the motion of the object is not likely to change abruptly over time. Therefore, using the MVs of neighbouring blocks

as predictor reduces computational power as well as the size of the signalled MV difference. The motion vector predictions (MVPs) are usually derived from the previously decoded MVs of the spatially neighbouring blocks as well as of temporally neighbouring blocks in the co-located picture. In H.264/AVC, this is performed by computing a component-wise median of the three spatially neighbouring MVs. In HEVC, this process is more elaborate as compared to H.264/AVC, hence it is denoted as AMVP.

In AMVP, two types of MVP candidates are considered; namely, spatial and temporal. In AMVP list, initially, five spatial and two temporally co-located MVP candidates are selected as shown in Fig. 2.15. This list is modified by re-ordering to place the most probable motion predictor in the first position and by removing redundant candidates to assure minimal signalling overhead. The flow of the derivation of MVP candidates in HEVC is shown in Fig. 2.16. Among five spatial MV candidates, two MV candidates are selected, one from the left side candidates and one from the above candidates. If left side candidates are not available, two MV candidates are selected only from the above candidates.

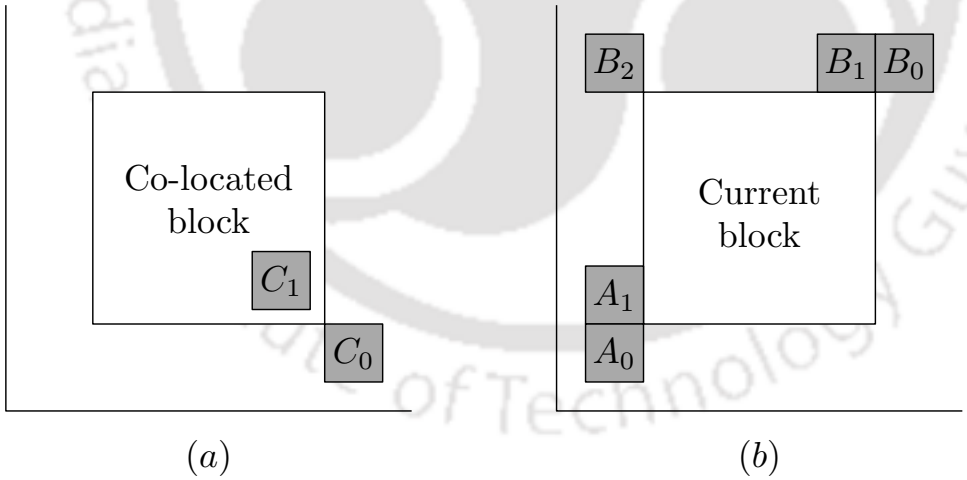


Figure 2.15: Motion vector predictor candidates: (a) Temporal and (b) Spatial [12].

For temporal MVP candidates, one MV candidate is selected from the two candidates. MVP candidate of the bottom right neighbour (C_0) is considered first and, if it is not available, MVP candidate of the co-located candidate block at the center (C_1) is used as a temporal MVP candidate. After the selection of the spatial-temporal candidate's list, the duplicated

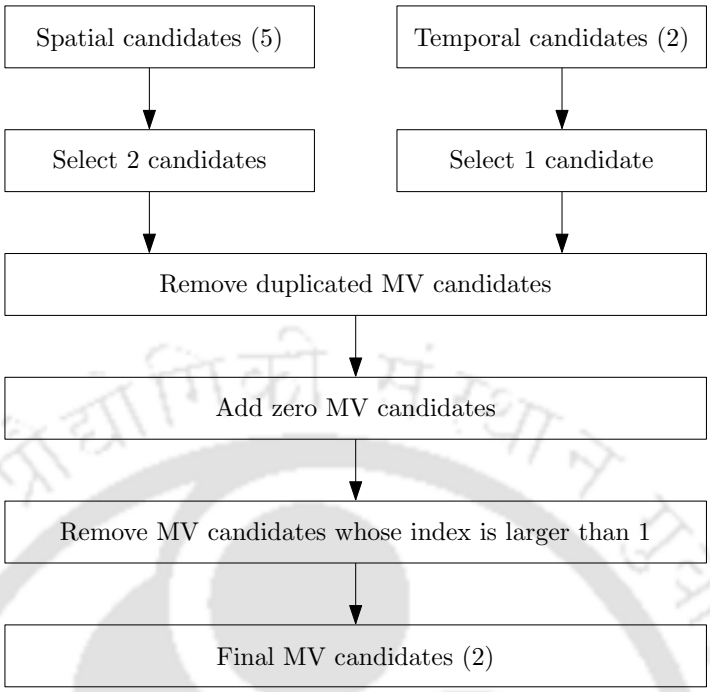


Figure 2.16: Derivation process for motion vector prediction candidates [7].

MV candidates in the list are removed. The Zero MV candidates are added to the list if the number of spatial-temporal MVP candidates is smaller than two. If the number of potential candidates are larger than two, MVP candidates whose index is larger than 1 are removed from the list. The candidates selection procedure for spatial and temporal list are presented as follows:

Spatial Motion Vector Candidates Derivation

As mentioned above, a maximum of two candidates (one from the left and one from the right) are selected among five potential candidates. The order of selection for left side candidates is $A_0 \rightarrow A_1 \rightarrow$ scaled $A_0 \rightarrow$ scaled A_1 and for right side candidates is $B_0 \rightarrow B_1 \rightarrow B_2 \rightarrow$ scaled $B_0 \rightarrow$ scaled $B_1 \rightarrow$ scaled B_2. Initially, no spatial scaling case is checked and spatial scaling cases are checked sequentially. The scaling is done only when the picture order count (POC) is different between the reference picture of the neighbouring blocks and that of the current block

regardless of the reference picture list. The scaling process is done as follows

$$mv = sign(mv_{cand} \times ScaleFactor) \times ((|mv_{cand} \times ScaleFactor| + 2^7) \gg 8) \tag{2.15}$$

where, mv_{cand} is the MV of candidate block and $ScaleFactor$ is calculated as

$$ScaleFactor = clip(-2^{12}, 2^{12} - 1, (tb \times tx + 2^5) \gg 6) \tag{2.16}$$

where, tb is the temporal distance between the current picture and the reference picture of the current block and tx is

$$tx = \frac{2^{14} + |\frac{td}{2}|}{td} \tag{2.17}$$

where td is the temporal distance between the current picture and the reference picture of the candidate block.

Temporal Motion Vector Candidates Derivation

As shown in Fig. 2.15, spatial MVP consider only the neighbouring left and the above blocks as the MVP candidates because blocks at right and below of the current blocks are not decoded yet and hence, their motion data is not available. Since the co-located reference picture is already decoded and the motion correlation between the current and co-located reference picture is very high. Hence, in temporal MVP candidate list co-located and bottom right blocks have been selected. In Fig. 2.15, C_1 and C_0 are co-located and bottom right blocks respectively. The motion data of C_0 is considered first as the temporal MVP candidate. If C_0 is not available, intra coded, or outside of the current CTU, motion data of the co-located candidate block (C_1) is used as the temporal MVP candidate [7]. In temporal MVP, scaling is mandatory and the scaling operation is same as used in spatial MVP, whereas in (2.16) and (2.17), tb is defined as the POC difference between the reference picture of the current picture and the current picture, td is defined as the POC difference between the co-located picture and

the reference picture of the co-located candidate block.

2.3.5.4 Merge Mode

When a CU is coded with skip mode, the CU is represented as one PU that has no significant transform coefficients, motion vectors, reference picture index and reference picture list. The merge mode is used to obtain the motion parameters for the current PU from neighbouring PUs, including spatial and temporal candidates. The merge mode can be applied to any inter-predicted PU, not only for skip mode.

Derivation of Merge Candidates

Merge mode candidates list is formed by spatial as well as temporal merge candidates as shown in Fig. 2.15. Derivation of the merge candidates is illustrated in Fig. 2.17. For spatial merge candidates, a maximum of four merge candidates is selected among the five spatial candidates that are located at five different positions. Candidates having the same motion parameters are removed from the candidate's list and the candidates inside the same merge

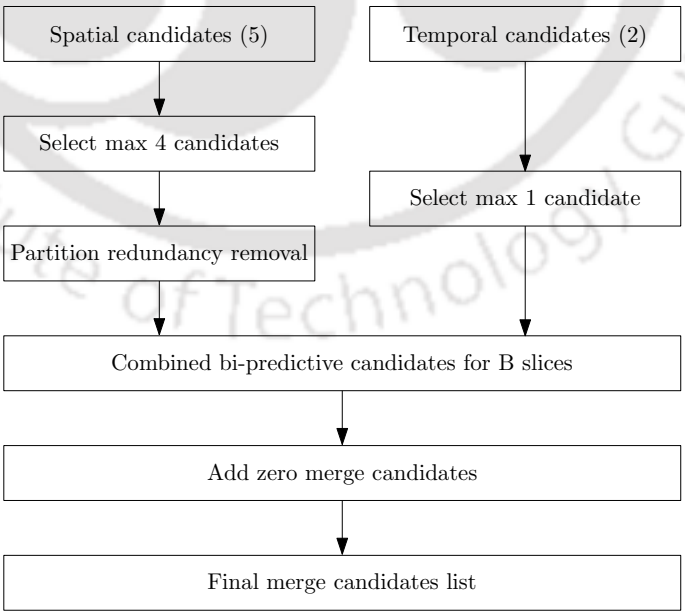
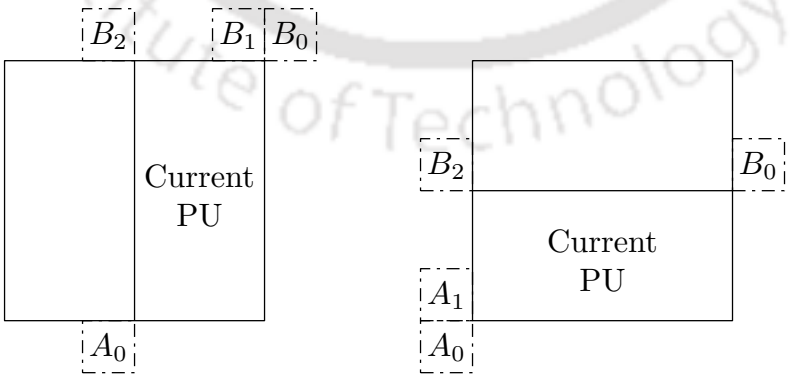


Figure 2.17: Derivation process for merge candidate.

estimation region are also not considered, which helps to incorporate parallel merge processing. For temporal merge candidate derivation, a maximum of one merge candidate is selected among two candidates. In the next step, bi-predictive candidates are generated for B slices utilizing the candidates from the list spatial-temporal candidates. The Zero merge candidates are added at the end of the list if the total number of candidates in the list are less than the maximum number of merge candidates. Derivation process is stopped if the number of candidates reaches the value of the maximum number of merge candidates. The value of the maximum number of merge candidates is set to five. If the size of CU is equal to 8, all the PUs of current CU shares a single merge candidate list.

Spatial Merge Candidates

For spatial merge candidate list, a maximum of four merge candidates is selected among five candidates that are located at the different positions as depicted left side in Fig. 2.15. The order of derivation is $A_1 \rightarrow B_1 \rightarrow B_0 \rightarrow A_0 \rightarrow (B_2)$. The B_2 is considered only when any of the A_1, B_1, B_0, A_0 is not available or intra coded. Position A_1 is not considered as a candidate for the second PU of $N \times 2N$, $nL \times 2N$ and $nR \times 2N$ partition. In this case, the order of derivation is $B_1 \rightarrow B_0 \rightarrow A_0 \rightarrow B_2$. Similarly, for the second of $2N \times N$, $2N \times nU$ and $2N \times nD$



(a) Second PU of $N \times 2N$ (b) Second PU of $2N \times N$

Figure 2.18: Positions for the second PU of $N \times 2N$ and $2N \times N$ partitions

partitions, position B_1 is not considered as a candidate and order is $A_1 \rightarrow B_0 \rightarrow A_0 \rightarrow B_2$. The example of these cases are shown in Fig. 2.18.

Temporal Merge Candidates

The derivation process for temporal merge candidate is the same as that of the temporal MVP illustrated in Section 2.3.5.3. A maximum of one merge candidate is selected among the two candidates that are located at two different positions as depicted on the right side of Fig. 2.15. Scaling of MV process is compulsory in temporal. A scaled MV is derived based on co-located PU belongs to the picture which has the smallest POC difference with the current picture within the given reference picture list. Motion data of C_0 is considered first as the temporal MVP candidate. If C_0 is not available, intra coded or outside of the current CTU, motion data of the co-located candidate block (C_1) is used as the temporal merge candidate [7]. In case of the B-slice, two MVs, one from each reference picture list (list 0 and list 1) are obtained and are combined to form the bi-predictive merge candidate.

Additional Merge Candidates

To provide more robustness, the length of the merge candidates list is fixed. So, after selecting spatio-temporal merge candidates, it could happen that the list is not yet full. In order to complete the list, two types of additional merge candidates namely, combined bi-predictive candidates and zero merge candidates are used. In bi-predictive slices, combined bi-predictive merge candidates are generated by combining spatio-temporal merge candidates. When the list is still not full after adding the combined bi-predictive candidates, zero merge candidates are used to complete the list. The Zero merge candidate has zero spatial displacements and a reference picture index which starts from zero. The reference indices are incremented by one for each additional candidate which is added to the list. The number of reference frames used by these candidates is one and two for Uni and bi-directional predictions, respectively. No

redundancy check is performed on these candidates.

2.3.5.5 Sub-Sample Interpolation

Similar to H.264/AVC, HEVC supports MV values up to a quarter-pixel accuracy for luma samples and 1/8 pixel accuracy for chroma samples. But, sample values at fractional grid are not available in natural video sequences. So, we have to interpolate the picture using integer sample values to find the pixel values at fractional positions. In H.264/AVC, 6-tap and a bilinear filter were used for interpolation of luma samples. In HEVC, several improvements are incorporated in interpolation filtering which contributes to the significant coding efficiency improvement. For luma sample interpolation, 8-tap and 7-tap DCT-based interpolation filters are used for half and quarter sample positions respectively. In chroma sample interpolation, four different 4-tap DCT-based interpolation filters are used. The standardized filter coefficients for luma and chroma samples are shown in Table 2.9 and 2.10 respectively.

Table 2.9: Luma interpolation filter coefficients

Position	Filter Coefficients
1/4	-1, 4, -10, 58, 17, -5, 1
2/4	-1, 4, -11, 40, 40, -11, 4, -1
3/4	1, -5, 17, 58, -10, 4, -1

Table 2.10: Chroma interpolation filter coefficients

Position	Filter Coefficients
1/8	-2, 58, 10, -2
2/8	-4, 54, 16, -2
3/8	-6, 46, 28, -4
4/8	-4, 36, 36, -4
5/8	-4, 28, 46, -6
6/8	-2, 16, 54, -4
7/8	-2, 10, 58, -2

In H.264/AVC, all the half sample values are calculated with the help of integer sample value and the quarter sample values are calculated with a simple average of one neighbouring integer and the half sample values. It is a two-stage cascade filtering process. However, in HEVC,

interpolation filter computes the quarter-pixels directly using a 7-tap filter, which significantly reduces the rounding error to 1/128. In HEVC, the first horizontal and vertical factional sample values are calculated with the help of the nearest integer position samples. In the second step, all the other fractional sample values are found with the help of fractional sample values calculated in the first step. The illustration of the fractional pixel for luma sample is shown in Fig. 2.19. Here, the upper-case letters represent the integer sample positions and the lower-case letters represent the fractional sample positions. In HEVC, 14-bit sub-sample scheme is specified. The sample values at the integer position in the reference picture are scaled to a dynamic range of 14 bits by performing a left-shift using the appropriate number of bits depending on the bit depth of the encoded video.

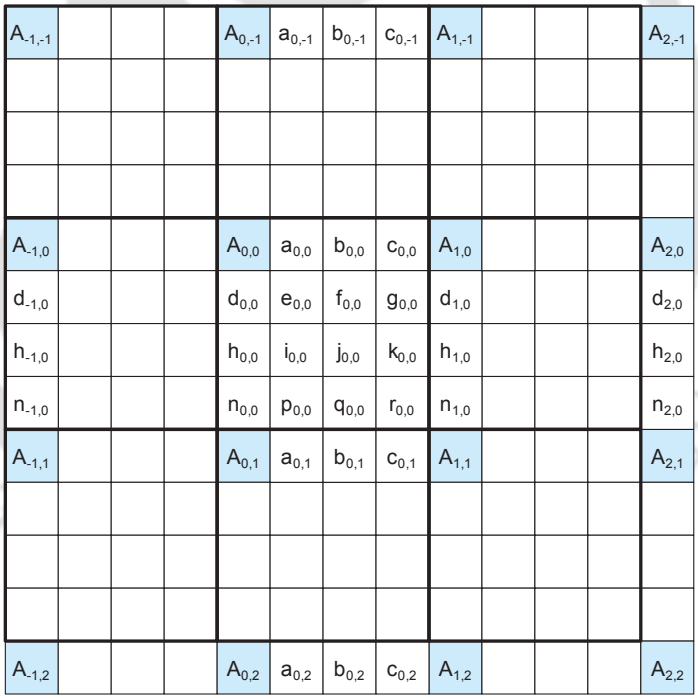


Figure 2.19: Integer and fractional positions with 1/4 pixel accuracy used in luma motion compensation.

The samples values at position $a_{0,0}$, $b_{0,0}$, $c_{0,0}$, $d_{0,0}$, $h_{0,0}$ and $n_{0,0}$ are derived by applying 7-tap and 8-tap filters to the nearest integer position samples are as follows:

$$a_{0,0} = (-A_{-3,0}+4\times A_{-2,0}-10\times A_{-1,0}+58\times A_{0,0}+17\times A_{1,0}-5\times A_{2,0}+A_{3,0}) >> shift1 \quad (2.18)$$

$$b_{0,0} = (-A_{-3,0}+4\times A_{-2,0}-11\times A_{-1,0}+40\times A_{0,0}+40\times A_{1,0}-11\times A_{2,0}+4\times A_{3,0}-A_{4,0}) >> shift1 \quad (2.19)$$

$$c_{0,0} = (A_{-2,0}-5\times A_{-1,0}+17\times A_{0,0}+58\times A_{1,0}-10\times A_{2,0}+4\times A_{3,0}-A_{4,0}) >> shift1 \quad (2.20)$$

$$d_{0,0} = (-A_{0,-3}+4\times A_{0,-2}-10\times A_{0,-1}+58\times A_{0,0}+17\times A_{0,1}-5\times A_{0,2}+A_{0,3}) >> shift1 \quad (2.21)$$

$$h_{0,0} = (-A_{0,-3}+4\times A_{0,-2}-11\times A_{0,-1}+40\times A_{0,0}+40\times A_{0,1}-11\times A_{0,2}+4\times A_{0,3}-A_{0,4}) >> shift1 \quad (2.22)$$

$$n_{0,0} = (A_{0,-2}-5\times A_{0,-1}+17\times A_{0,0}+58\times A_{0,1}-10\times A_{0,2}+4\times A_{0,3}-A_{0,4}) >> shift1 \quad (2.23)$$

The samples values at position $e_{0,0}$, $i_{0,0}$, $p_{0,0}$, $f_{0,0}$, $j_{0,0}$, $q_{0,0}$, $g_{0,0}$, $k_{0,0}$ and $r_{0,0}$ are derived by applying 7-tap and 8-tap filtering to the samples $a_{0,i}$, $b_{0,i}$, $c_{0,i}$ with $-3 \leq i \geq 4$, in the vertical direction as follows:

$$e_{0,0} = (-a_{0,-3}+4\times a_{0,-2}-10\times a_{0,-1}+58\times a_{0,0}+17\times a_{0,1}-5\times a_{0,2}+a_{0,3}) >> shift2 \quad (2.24)$$

$$i_{0,0} = (-a_{0,-3} + 4 \times a_{0,-2} - 11 \times a_{0,-1} + 40 \times a_{0,0} + 40 \times a_{0,1} - 11 \times a_{0,2} + 4 \times a_{0,3} - a_{0,4}) >> shift2 \quad (2.25)$$

$$p_{0,0} = (a_{0,-2} - 5 \times a_{0,-1} + 17 \times a_{0,0} + 58 \times b_{0,1} - 10 \times b_{0,2} + 4 \times a_{0,3} - a_{0,4}) >> shift2 \quad (2.26)$$

$$f_{0,0} = (-b_{0,-3} + 4 \times b_{0,-2} - 10 \times b_{0,-1} + 58 \times b_{0,0} + 17 \times b_{0,1} - 5 \times b_{0,2} + b_{0,3}) >> shift2 \quad (2.27)$$

$$j_{0,0} = (-b_{0,-3} + 4 \times b_{0,-2} - 11 \times b_{0,-1} + 40 \times b_{0,0} + 40 \times b_{0,1} - 11 \times b_{0,2} + 4 \times b_{0,3} - b_{0,4}) >> shift2 \quad (2.28)$$

$$q_{0,0} = (b_{0,-2} - 5 \times b_{0,-1} + 17 \times b_{0,0} + 58 \times b_{0,1} - 10 \times b_{0,2} + 4 \times b_{0,3} - b_{0,4}) >> shift2 \quad (2.29)$$

$$g_{0,0} = (-c_{0,-3} + 4 \times c_{0,-2} - 10 \times c_{0,-1} + 58 \times c_{0,0} + 17 \times c_{0,1} - 5 \times c_{0,2} + c_{0,3}) >> shift2 \quad (2.30)$$

$$k_{0,0} = (-c_{0,-3} + 4 \times c_{0,-2} - 11 \times c_{0,-1} + 40 \times c_{0,0} + 40 \times c_{0,1} - 11 \times c_{0,2} + 4 \times c_{0,3} - c_{0,4}) >> shift2 \quad (2.31)$$

$$r_{0,0} = (c_{0,-2} - 5 \times c_{0,-1} + 17 \times c_{0,0} + 58 \times c_{0,1} - 10 \times c_{0,2} + 4 \times c_{0,3} - c_{0,4}) >> shift2 \quad (2.32)$$

where $shift1 = min(4, BitDepth_Y - 8)$ and $shift2 = 6$.

For the bi-directional prediction, the bit-depth of the output of the interpolation filter is maintained at 14-bit accuracy regardless of the source bit-depth. The averaging process is done

implicitly with the bit-depth reduction process given by

$$S(x,y) = (S_1(x,y) + S_2(x,y) + offset) >> shift \tag{2.33}$$

where $shift = (15 - BitDepth)$ and $offset = 1 << (shift - 1)$.

2.3.5.6 Weighted Prediction

Similar to H.264/AVC, a WP technique is included in the HEVC. The main reason of using the WP is to improve the performance of inter prediction when the source material is subjected to fading. WP is an optional Picture Parameter Set (PPS) parameter and it may be switched on/off when necessary. For common test condition, it is not enabled in the HM reference software. The WP is controlled by separate flags: weighted_pred_flag and weighted_bipred_flag for P and B slices respectively and transmitted in the PPS. The principle of WP is to replace the inter prediction signal P by a linear weighted prediction signal P' as follows:

Uni-prediction: $P' = w \times P + o$

Bi-prediction: $P' = (w_0 \times P_0 + o_0 + w_1 \times P_1 + o_1)/2$

where w is the illumination compensation weight and o is the offset. The value of w and o are calculated by the encoder and are not specified in the HEVC specification.

In case of bi-prediction, at least one reference picture is available in each lists L_0 and L_1. For luma, P' is calculated as:

$P'(x,y) = \text{Clip3}(0, \text{ max\_value}, (P_{L0}(x,y) \times w_0 + P_{L1}(x,y) \times w_1 + ((o_0 + o_1 + 1) << log_2WD)) >> (log_2WD + 1))$

where x and y denote spatial coordinates within the prediction block and log_2WD = luma log_2 weighted denominator + 14 - bitDepth.

2.3.6 Wavefront Parallel Processing

HEVC uses WPP, in which each slice is divided into rows of CTUs. WPP can be enabled/disabled by $entropy\_coding\_sync\_enabled\_flag$. In WPP, the first row is processed in an ordinary way, the second row can begin to be processed only after two CTUs have been

processed in the first row, the third row can begin to be processed after only two CTUs have been processed in the second row, and so on(as shown in Fig. 2.20). This results in the simultaneous decoding of the picture like 'wavefront'. The dependencies between the consecutive CTU rows are maintained at the partition boundaries till the end of each CTU row except for the CABAC. The context models of the entropy coder in each row are inferred from those in the preceding row with a two-CTU processing lag. WPP provides a form of parallel processing within a slice, without the loss of compression performance that might be expected by using tiles within a slice.

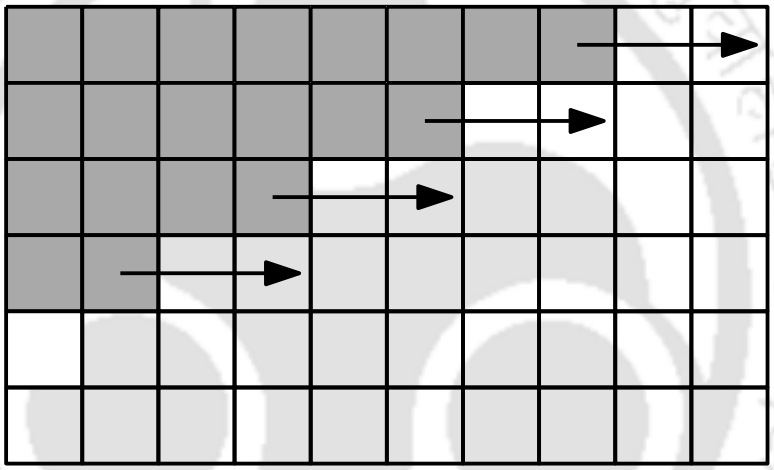


Figure 2.20: Graphical presentation of Wavefront Parallel Processing.

2.3.7 Transform and Quantization

Transform coding transforms the prediction error data from the spatial domain to the frequency domain. The transform data have an advantage of signal energy compaction, which is desirable for data compression. In the block-based hybrid video coding approach, transforms are applied to the residual signal after inter or intra picture prediction. HEVC specifies two types of two-dimensional transforms namely, DCT and discrete sine transforms (DST). DCT is the core transform and DST is used only with 4×4 luma intra prediction residual blocks [13]. The specified DCT transform matrices are of size 4×4, 8×8, 16×16 and 32×32 [13]. The smaller size DCT matrices are embedded in the larger size transform matrices. The standardized 8×8

DCT transform matrix presented in [100] is as follows:

$$DCT_{8\times8} = \begin{pmatrix} 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 \\ 89 & 75 & 50 & 18 & -18 & -50 & -75 & -89 \\ 83 & 36 & -36 & -83 & -83 & -36 & 36 & 83 \\ 75 & -18 & -89 & -50 & 50 & 89 & 18 & -75 \\ 64 & -64 & -64 & 64 & 64 & -64 & -64 & 64 \\ 50 & -89 & 18 & 75 & -75 & -18 & 89 & -50 \\ 36 & -83 & 83 & -36 & -36 & 83 & -83 & 36 \\ 18 & -50 & 75 & -89 & 89 & -75 & 50 & -18 \end{pmatrix} \tag{2.34}$$

The transform matrix of size 4×4 is derived from the transform matrix 8×8. The blue color elements in the 8×8 matrix are the elements of the 4×4 transform matrix. So, the transform matrix of size 4×4 is

$$DCT_{4\times4} = \begin{pmatrix} 64 & 64 & 64 & 64 \\ 83 & 36 & -36 & -83 \\ 64 & -64 & -64 & 64 \\ 36 & -83 & 83 & -36 \end{pmatrix} \tag{2.35}$$

The standardized 4×4 DST transform matrix represented in [100] is as follows:

$$DST_{4\times4} = \begin{pmatrix} 29 & 55 & 74 & 84 \\ 74 & 74 & 0 & -74 \\ 84 & -29 & -74 & 55 \\ 55 & -84 & 74 & -29 \end{pmatrix} \tag{2.36}$$

In video coding standards such as HEVC, the de-quantization process and inverse transforms are specified, while the forward transforms and quantization process is chosen by the designer subject to constraints on the bit-stream. The forward and inverse transform matrices are the transpose of one another. The forward and the inverse transform are designed in such a way

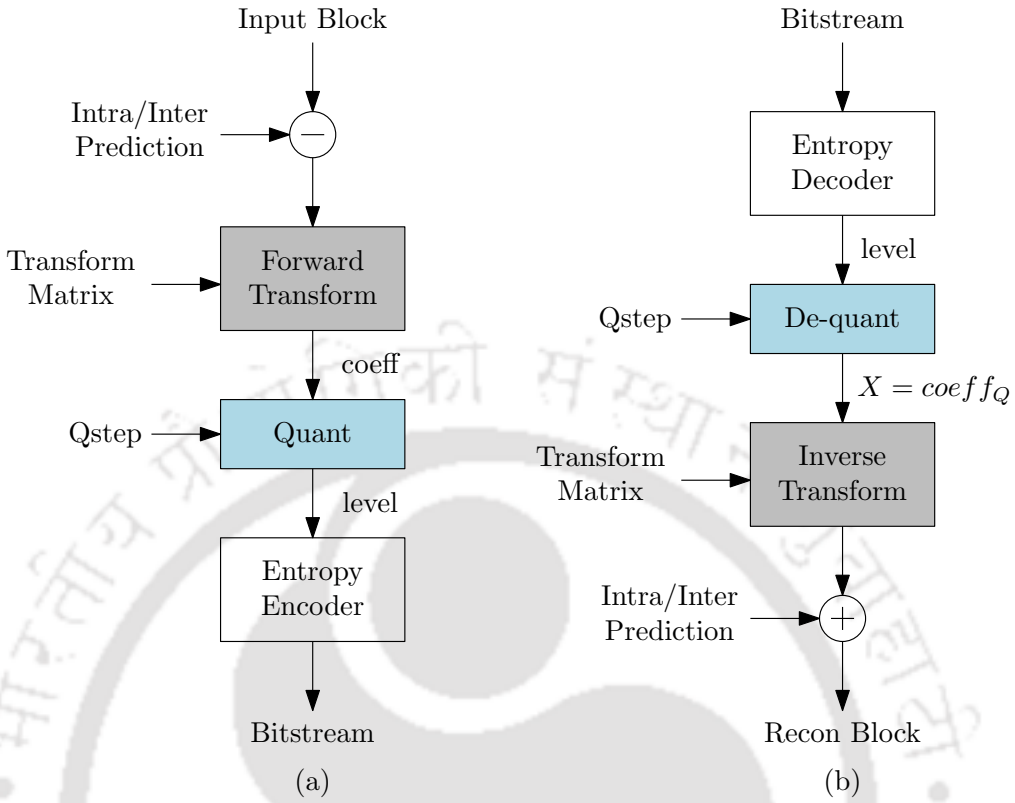


Figure 2.21: Block-based hybrid video coding (a) Encoder, (b) Decoder. *Qstep* is the quantization step size [13].

that nearly lossless reconstruction of the residual blocks can be achieved when the intermediate quantization and de-quantization steps are excluded. The flow of transforms and quantization at the encoder and decoder side is shown in Fig. 2.21. At the encoder, the residual frame is divided into square block size. In HEVC, residual blocks are partitioned into a quad-tree fashion, which is called RQT. Each residual block is then inputted to a two-dimensional forward transform. The resulting transform coefficients are then subjected to the quantization (which is equivalent to division by quantization step size) to obtain quantized transform coefficients. At the decoder, the quantized transform coefficients are then de-quantized (which is equivalent to multiplication). Finally, a two-dimensional separable inverse transform is applied to the de-quantized transform coefficients resulting in a residual block of quantized samples which is then added to the intra or inter-prediction samples to obtain the reconstructed block.

2.3.8 Loop Filtering

Loop filtering is one of the most important parts of all the latest video coding standards. They are used for improving the visual quality of the reconstructed picture and helps for better compression. In HEVC, two in-loop filters are specified, namely, DBF and SAO filter. Which can be enabled or disabled independently. The loop filters are applied in both the encoding and decoding loops. In the loop, filters are applied after the inverse quantization and before saving the picture to the decoded picture buffer. The DBF attenuates discontinuities at the prediction and transforms block boundaries. The SAO filter aims to improve the accuracy of the reconstruction of the original signal amplitudes.

2.3.8.1 Deblocking Filter

DBF is applied on each CU in the same order as the decoding process. In HEVC, a DBF is applied to the vertical boundaries (horizontal filtering) of the picture first, followed by the horizontal boundaries (vertical filtering). The DBFs are applied to an 8×8 chroma and luma sample grid. Based on the boundary strength and threshold values, DBF performs three operations such as no filtering, weak filtering and strong filtering. The value of boundary strength is 2 for strong filtering, 1 indicates weak filtering and 0 indicates no deblocking filtering. The conditions for selection of the boundary strength parameter are as follows:

- If at least one of the blocks is intra, then the boundary strength parameter is set to 2.
- If at least one of the blocks has non-zero coded residual coefficient and the boundary is a transform boundary, then the boundary strength parameter is set to 1.
- If the absolute differences between the corresponding spatial motion vector components of the two blocks are $\geqq 1$ in units of integer pixels, then the boundary strength parameter is set to 1.
- If motion-compensated prediction for the two blocks refers to different reference pictures or if the number of motion vectors is different for the two blocks, then the boundary strength parameter is set to 1.

- Otherwise, the boundary strength parameter is set to 0.

Two threshold parameters are used to control the filtering operation. The value of these thresholds depends upon the quantization parameter of these blocks.

2.3.8.2 Sample Adaptive Offset Filter

SAO filter was used for the first time in HEVC. In order to further increase the visual quality of video by removing the ringing artifacts and by changing the sample intensity in some areas of the picture, the output of DBR is applied to SAO filter. Two types of SAO modes are used in HEVC; namely, edge offset (EO) and band offset (BO) [14]. In EO, the classification of a sample is based on its neighbourhood, i.e. on the comparison between the current sample and its neighbouring samples. There are four one-directional patterns for sample classification: horizontal, vertical, 135° diagonal and 45° diagonal as shown in Fig. 2.22. In Fig. 2.22, "c" represents the current sample and "a" and "b" represents the neighbouring samples. Only one of the patterns is selected for each CTB. For a given EO class with the specific direction, each sample inside the CTB is classified into one of five categories: minimum, maximum, an edge with the sample having the lower value, an edge with the sample having the higher value, or monotonic. The classification rules for each sample are summarized in Table 2.11. Categories 1 and 4 are associated with a local valley and a local peak along the selected 1-D pattern, respectively. Categories 2 and 3 are associated with concave and convex corners along the selected 1-D pattern, respectively. If the current sample does not belong to EO categories 1 -

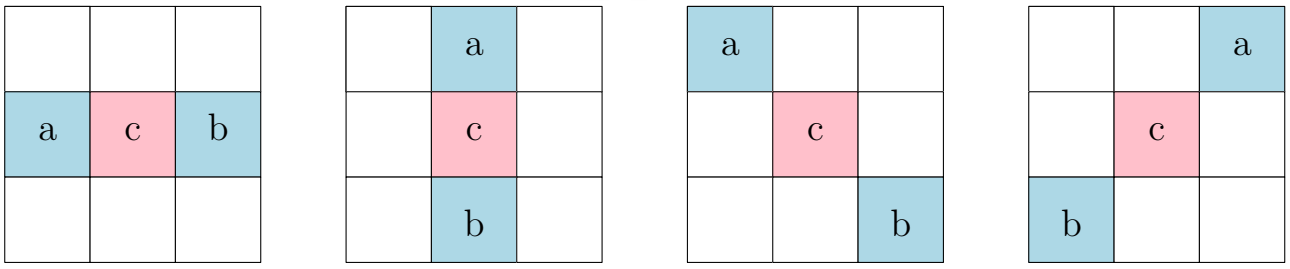


Figure 2.22: Four 1-D directional patterns for EO sample classification: horizontal (EO class = 0), vertical (EO class = 1), 135° diagonal (EO class = 2), and 45° diagonal (EO class = 3) [14].

4, then it is category 0 and SAO is not applied [14].

Table 2.11: Sample Classification Rules for Edge Offset [14]

Category	Condition
1	c < a && c < b
2	(c < a && c == b) || (c == a && c < b)
3	(c > a && c == b) || (c == a && c > b)
4	c > a && c > b
0	None of the above

In BO, the classification is based on the amplitude of the sample value and not on the neighborhood of the sample. In BO, the full sample amplitude range is uniformly split into 32 segments called bands. In other words, the pixel intensity range is divided into 32 equal segments from zero to the maximum intensity value. In BO, one offset is added to all samples whose values belong to the same band. Any four consecutive bands out of 32 are used in BO. Only four consecutive offsets and the starting band position are signaled in the BO.

2.3.9 Entropy Coding

Entropy coding is a lossless compression scheme that uses the statistical properties to compress data. It is the last stage of the HEVC encoder and first stage of the HEVC decoder. In entropy coding, the number of bits used to represent the data is logarithmically proportional to the probability of the data. For instance, frequently used characters are represented by few bits, while infrequently used characters are represented by more bits. In HEVC, CABAC is applied at the CTU level. The CABAC was originally developed with H.264/AVC standard. In H.264/AVC, one more coding technique called Context-Adaptive Variable-Length Coding (CAVLC) is also used. In HEVC, CABAC is the only type of coding scheme used. The basic principle of CABAC encoding is the same as that used in H.264/AVC. The block diagram of the CABAC encoder is presented is Fig.2.23. It involves three key elements such as binarization, context modeling, and binary arithmetic coding. Binarization maps the syntax elements to binary symbols (bins). The context modeling estimates the probability of each non-bypassed (i.e., regular coded) bin based on some specific context. Finally, binary arithmetic coding

compresses the bins to bits according to the estimated probability.

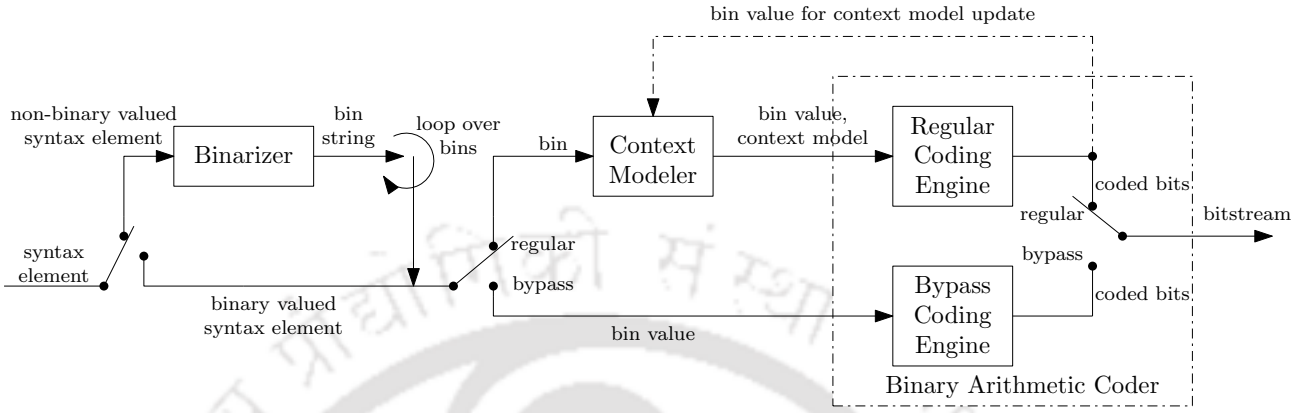


Figure 2.23: Block Diagram of CABAC Encoder [15].

2.3.10 Encoder Configuration

The HEVC encoder supports three types of temporal prediction structure depending upon the applications: intra-only, low-delay and random access.

2.3.10.1 Intra-only Configuration

In intra-only coding, each picture in a video sequence is encoded as an Instantaneous Decoder Refresh (IDR) or intra coded picture. Temporal references are not used in prediction. QP does not change during a sequence within a picture. The graphical presentation of an intra-only configuration is presented in Fig. 2.24, where the number associated with each picture represents the encoding order.

2.3.10.2 Low-delay Configuration

For low-delay coding conditions, only the first picture in a video sequence is coded as an IDR or intra coded picture and all other pictures are encoded as generalized P and B pictures (GPBs). The GPBs can use only reference picture whose picture order count is smaller than the current picture. The QP of each inter-coded picture is derived by adding an offset to the QP of the intra-coded picture depending on the temporal layer. The graphical presentation of

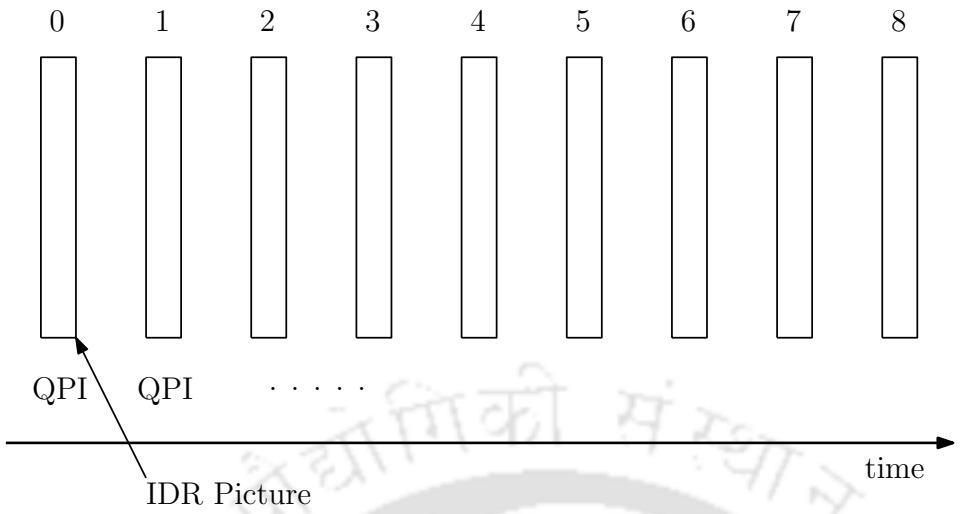


Figure 2.24: Graphical representation of intra-only configuration [7].

an intra-only configuration is presented in Fig. 2.25, where the number associated with each picture represents the encoding order.

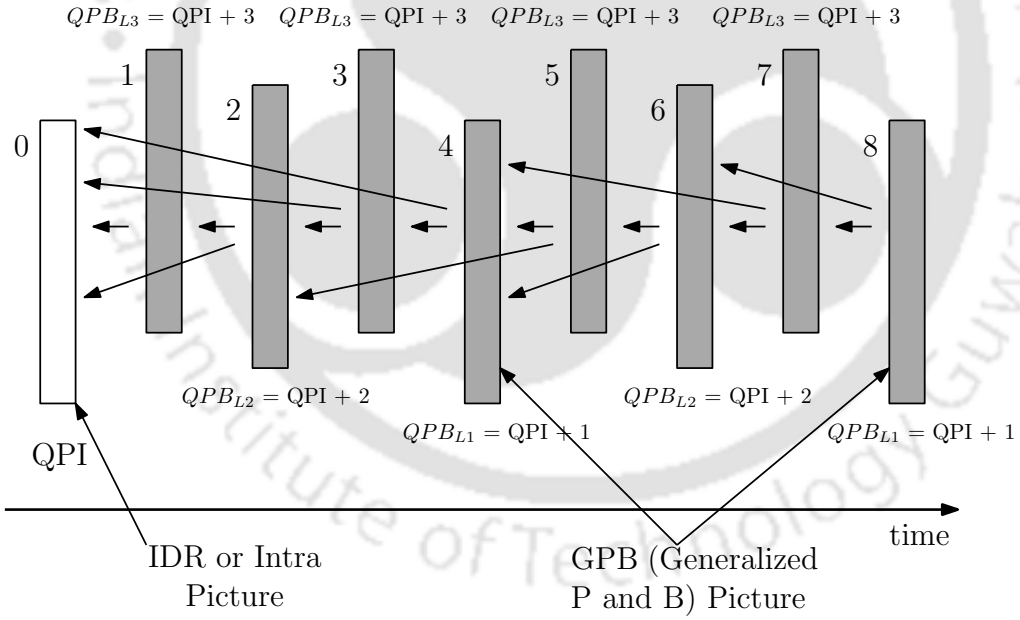


Figure 2.25: Graphical representation of low-delay configuration [7].

2.3.10.3 Random-access Configuration

In the Random-access test condition, a hierarchical B structure is used for encoding. An intra-picture is encoded at approximately one-second intervals. The picture encoded between

successive intra pictures is B-pictures. The highest temporal contain the non-referenced B picture. The QP of each inter-coded picture is derived by adding an offset to the QP of the intra-coded picture depending on the temporal layer. The graphical presentation of an intra-only configuration is presented in Fig. 2.26, where the number associated with each picture represents the encoding order.

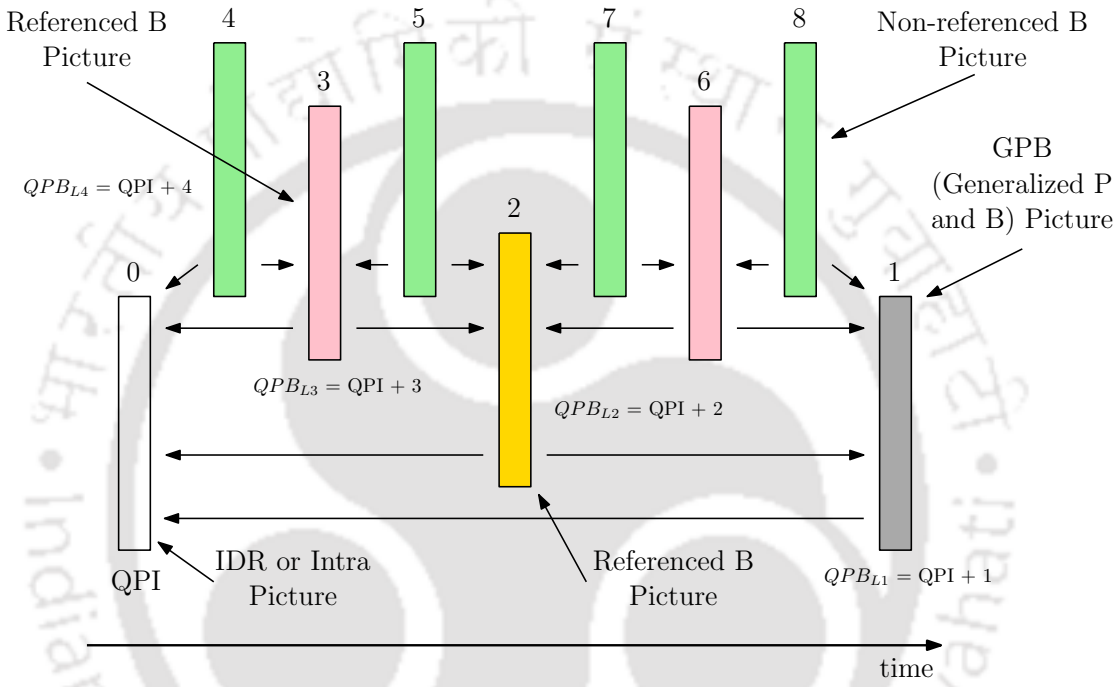


Figure 2.26: Graphical representation of random-access configuration [7].

2.3.11 Profiles and Levels

A profile consists of a subset of the algorithmic features or coding tools of the standard and a set of constrained for those features or in other words, a profile is a defined set of coding tools that can be used to create a bit-stream which conforms to that profile. An encoder has a freedom to select which features of the profile are used to generate the conforming bit-stream while a decoder for a profile must support all coding tools that can be used in that profile. The first version of the HEVC standard defines three profiles: Main, Main 10, and Main Still Picture. The second version of the HEVC adds 21 range extensions profiles, two scalable extension profiles, and one multi-view profile. HEVC also contains provisions for additional

profiles. The extensions that were added to HEVC include increased bit depth, 4:2:2/4:4:4 chroma sampling, Multiview Video Coding (MVC), and Scalable Video Coding (SVC).

The three profiles specified in the first version of the HEVC can be represented by a hierarchical structure as visualized in Fig. 2.27. The target profiles at different application spaces are detailed in the following subsections. The common constraints of these profiles are as follows:

- The main profiles only support the representation of YCbCr $4:2:0$ video.
- The size of coding tree blocks is constrained to be one of 16×16, 32×32 and 64×64 samples.
- The application has to decide either tile to be used for modification of the CTU scanning order, or to enable wavefront parallel processing, or not to use any of them. It is not allowed to use both tools at the same time.
- If tiles are used, the minimum tile size is 256×64 samples.

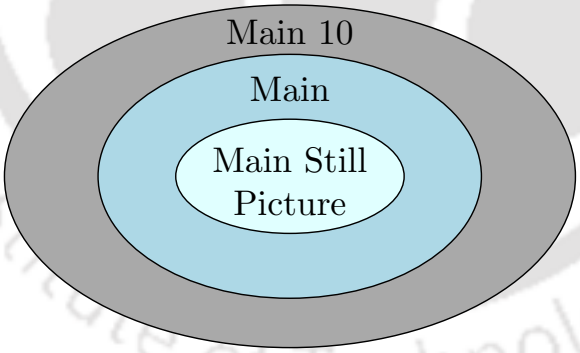


Figure 2.27: Graphical presentation of the main profiles in HEVC.

2.3.11.1 Main

The Main profile represents video data with 8-bits per sample, which is the most common type of video used with consumer devices.

2.3.11.2 Main 10

The Main 10 profile supports 8-bits to 10-bits per sample. HEVC decoders that conform to the Main 10 profile must be capable of decoding bit-streams made with Main and Main 10. This profile is a superset of the capabilities of the Main profile. The higher bit depth is used to increase the brightness dynamic range, extended color-gamut content, or simply higher fidelity color representations to avoid contouring artifacts and reduce rounding error.

2.3.11.3 Main Still Picture

The Main Still Picture profile allows for a single still picture to be encoded with the same constraints as the Main profile. This profile is a subset of the Main profile. The Main Still Picture profile allows for a bit depth of 8-bits per sample.

2.4 HM Reference Software

For verification of the results, we have used HM reference software. The HM reference software is a reference implementation of the HEVC coding standard developed by the JCT-VC. The reference software includes both encoder and decoder functionality. This is suitable for experimenting with the various features available in the HEVC coding standard and/or for checking compliance. The HM encoder works with three kinds of temporal prediction structures depending on experimental conditions: intra-only, low-delay and random access. In intra-only coding, each picture in a video sequence is encoded as an IDR or intra coded picture. No temporal reference pictures are used. For low-delay coding conditions, only the first picture in a video sequence is coded as an IDR or intra coded picture and all other pictures are encoded as GPBs. In the Random-access test condition, a hierarchical B structure is used for encoding. An intra-picture is encoded at approximately one-second intervals. The picture encoded between successive intra pictures are B-pictures. In this thesis, we used only low-delay and random access profiles.

2.5 Conclusion

In this chapter, we first presented the basic overview of digital video and video encoding. In the next section, a brief overview of international video coding standards is presented. HEVC CODEC along with fundamental building blocks, encoder configuration, profile supported by HEVC and HM reference software is presented in the last section of the chapter. The following chapters describe the main contribution of this research in relation to the primary objective.

3

Computationally Efficient Integer Motion Estimation Algorithm for HEVC

Contents

In the video coding process, the inter prediction is used to remove temporal redundancy. It gives several times more compression as compared to other redundancy techniques such as intra prediction and CABAC. In inter prediction, the motion between the successive frames is captured with the ME. ME is one of the most computationally complex blocks of video coding standards. In HEVC, ME complexity is further increased due to the larger block size, asymmetrical PU partitioning and quadtree partitioning of CTU. In this chapter, we present a new computationally efficient ME algorithm for HEVC. The proposed algorithm searches in the hexagonal pattern with a fixed number of search points at each grid. In this algorithm, we exploited the correlation between contiguous pixels within the frame, pixel truncation, adaptive search range, sub-sampling and avoided some of the asymmetrical prediction unit techniques.

The rest of the chapter is organised as follows. The chapter begins with the discussion on HEVC and ME. Low complexity realization of HEVC is presented in the next section. The chapter ends with the simulation results and the conclusion.

3.1 Introduction

Nowadays, portable devices are an inseparable part of our daily lives. Due to the large growth in 3G/4G technologies, watching a video and storing video data on portable devices have become an easy task. It is expected that in the coming few years, the monthly total mobile data traffic will increase up to 24.3 Exabyte [101]. As per CISCO, the overall smartphone data traffic is expected to be 10 times in 2019 compared to 2014 with a compound annual growth rate of 60% [101]. Digital video data needs large bandwidth and space for transmission and storage, respectively. This, in turn, demands more power. For all portable devices, power is the main concern. This problem can be efficiently addressed by employing data compression techniques. Among all the existing standards, HEVC is the state of the art video compression standard, which is jointly developed by the ISO/IEC MPEG and ITU-T VCEG as ISO/IEC 23008-2 MPEG-H Part 2 and ITU-T H.265. Like the traditional video coding standards, HEVC also uses hybrid block-based video compression technique. However, HEVC gives half the bit rate as compared to its predecessor (H.264/AVC) maintaining the same video quality. The

bit rate compression in HEVC is achieved due to incorporation of efficient techniques such as quadtree block partitioning, more flexible PU partitions, larger CTU size, AMVP scheme, new block merging techniques, more accurate interpolation filter for fractional motion estimation, more intra prediction modes, larger transform unit size and new techniques used in entropy encoding.

In all the existing video coding standards, temporal redundancy in video sequences is removed by inter prediction. ME is one of the most important operations in inter prediction, which tries to find out the best possible matched block in the search range (SR) of the previously encoded frames, and generates a MV. The MV represents the distance between the best-matched candidate block in the search window of a reference frame and position of the current block in a current frame. FS is the most accurate algorithm for finding the best-matched candidate within the previously encoded frames. However, this algorithm is time-consuming and hence demands more power. In H.264/AVC baseline encoder, ME (IME + FME + Interpolation) consumes 97.32% of the total encoder time [102]. Therefore, in order to overcome this problem, several fast ME algorithms such as TSS [1], new TSS [40], successive elimination algorithm (SEA) [31], blocked based gradient search (BBGDS) [25], four-step search (4SS) [24], DS [2], successive elimination algorithm [32], predictive motion vector field adaptive search technique (PMVFAST) [26], adaptive root pattern search (ARPS) [28], HEXBS [3], cross-diamond search [27], octagonal search [44], cross-diamond-hexagonal search (CDHS) [29], enhanced PMVFAST (E-PMVFAST) [30], new cross-diamond search (NCDS) [51], TZS [103], modified TZS algorithms [34] and modified diamond search [52] have been proposed. But, none of these algorithms are suitable for low power devices as well as high-resolution video sequences. Moreover, the ME algorithm for HEVC becomes more complex due to larger CTU size, quad-tree block partitioning, AMVP scheme, larger half and quarter pixel interpolation filter, and new block merging techniques. The total complexity of HEVC encoder and decoder is much higher as compared to H.264/AVC video coding standard. Therefore, computationally efficient IME algorithms are required for low power applications.

3.2 Existing Integer ME Algorithms

Over last two decades, several ME algorithms have been reported in the literature [1–3, 7, 17, 24–32, 34, 40, 44, 51, 52, 58, 59, 104–108]. The descriptions of a few latest algorithms are discussed below:

TZS ME algorithm is used in the reference software [103]. It provides better speed with negligible loss in visual quality as compared to full search ME. The TZS ME algorithm mainly uses three types of search shapes such as diamond, square and raster. An illustration of the TZS algorithm is shown in Fig. 3.1 and is described in the following steps:

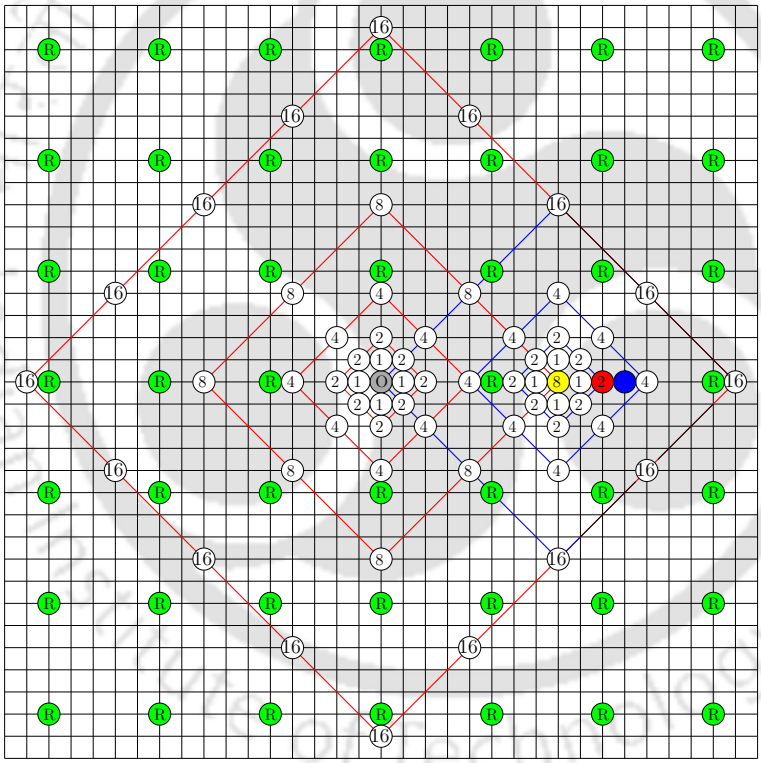


Figure 3.1: Illustration of the TZS Algorithm.

- *Step 1: Motion Vector Prediction*: - This is the first step of TZS algorithm. This step tests the predicted motion vector (PMV) position, MV at the origin and other MVs that are used in prediction process and choose the minimum distorted block as the starting point of the integer search.

- *Step 2: Initial Grid Search*: - In this step, the initial grid search was performed from a distance which varies from 1 to 64 using either diamond or square pattern. The distance in the subsequent step is maintained twice that of the previous step as shown in Fig. 3.1. By default, the diamond search is performed and it will stop either three steps after finding the best-matched candidate or the search reaches the boundary of the search window. Depending upon the distance of the best-matched candidate, either this process is stopped or performs the next step (two-point search, raster search and refinement search).

- *Step 3: Raster Search*: - Raster search is performed only if the distance of the best-matched candidate is larger than the 'iRaster' value. By default 'iRaster' value is given as '5' in the reference software. It will search the whole sub-sampled search window.

- *Step 4: Star Refinement*: - Star refinement, which is also called the final grid search, is performed only if the distance of the best-matched candidate is greater than one. This also uses two shapes like the initial grid search. By default, this uses diamond search and the process will be stopped either two steps after the best-matched candidate is obtained, or if the search reaches the boundary of the search window. Depending on the distance of the best-matched candidate, either this searching process will be stopped or searches only two points.

- *Step 5: Two Point Search*: - This search is performed only when the best-matched candidate distance in the previous step is equal to one. This step will be executed only once and the output of this step is the final best-matched candidate in the search window.

Due to the variable number of search points at different search grids and different types of search patterns used, it is not suitable for low-power design. In [34], Purnachand et al. proposed a modified TZS algorithm by replacing the diamond search grid with rotating hexagonal search grid. Even though the hexagonal grid requires less number of search points compared to diamond, it is not suitable for real-time hardware design because of rotating shapes and the

necessity of a different number of search points at different distances. In [104], Yang et al. proposed a fast ME algorithm with directional search, which is the combination of two search patterns namely, square search with a distance of one and cross-search with a distance of four. But, this algorithm fails to detect diagonal motion of the object in video sequences. In [17], Jou et al. proposed an adaptive SR based fast ME algorithm (ASRFMEA) for real-time quad full high definition (QFHD) sequences. Even though the results are appreciable and the main flow in the ASRFMEA algorithm is the same as the TZS algorithm used in reference software, this algorithm also does not support an asymmetrical partition of PU. Although this algorithm is suitable for high-resolution video sequences, there is still scope for further reduction in complexity.

For reducing the computation complexity of ME algorithms, different techniques based on reducing the number of search points (Fast ME algorithm) [1–3, 24–32, 40, 44, 107–109], pixel truncation [80, 110], sub-sampling of the image [81], removing some partition from PU partitions [17], adaptive search range [111] and early termination technique [33] were presented in the literature. In natural video sequences, the correlation between pixels is very high and hence the application of sub-sampling during ME does not affect much [81]. However, most of these techniques were applied on the previous video coding standards. To the best of our knowledge, these techniques have not been employed in the latest HEVC standard. In this chapter, we evaluated the performance of HEVC using different video sequences by employing all the above-mentioned techniques. Simulation results show that the use of these techniques reduces computational complexity greatly without much degradation in bit-rate and PSNR.

3.3 Proposed Low Complexity Realization of HEVC

As ME in HEVC involves computationally extensive operations, reduction in complexity of ME algorithms enables the HEVC to adapt in low power, less area consumer applications. Several techniques such as adaptive search range, pixel truncation, sub-sampling and skip some of PU partitions can be applied on HEVC to obtain the low-complexity realization of ME algorithms without compromising on the quality of the video. The details of these techniques

are presented in subsequent sections.

3.3.1 Adaptive Search Range

Table 3.1: Video sequences used for simulations

Video Sequences (VS)	Annotation Used
BasketballPass_416x240_50	Video_a
BQMall_832x480_60	Video_b
BasketballDrill_832x480_f500_60	Video_c
Johnny_1280x720_60	Video_d
KristenAndSara_1280x720_60	Video_e
crowd_run_1080p50	Video_f
old_town_cross_1080p50	Video_g
Traffic_2560x1600_30_crop	Video_h
Speed_bag_1080p	Video_i
Ducks_take_off_1080p50	Video_j
In_to_tree_1080p50	Video_k
Tennis_1920x1080_f240	Video_m

In order to study the effect of adaptive SR on HEVC, we performed experiments using the first eight video sequences video_a to video_h as shown in Table 3.1. We used HM reference software with *encoder_lowdelay_P_main* profile. Fig. 3.2 presents the total percentage of the best matched PBs found at each depth. The best matched PBs found at depth 0, 1, 2 and 3 are 55.52%, 23.96%, 14.78% and 5.74% respectively. The size of the best matched PBs found at each depth for different types of video sequences are shown in Fig. 3.3. It is found that a total of 95% best matched PBs are of sizes $2N \times 2N$, $2N \times N$ and $N \times 2N$ only. Here N can be 8, 16 and 32. From Fig. 3.2, it can be noted that the probability is very less to find the best-matched candidate in depth three. In addition, it is also clear from Fig. 3.3 that the probability of asymmetrical PU partitions is very low.

The MV distribution for different types of video sequences is shown in Fig. 3.4. It can be clearly noted from Fig. 3.4 that 48% of the MVs are concentrated at origin (0,0) of the search window and 93%, 96.1%, 97.3% and 99.31% of MVs lies in (±8,±8), (±12,±12), (±16,±16) and (±32,±32) ranges of search window respectively.

In ME, larger PBs are selected as the best-matched block, when movement in that region

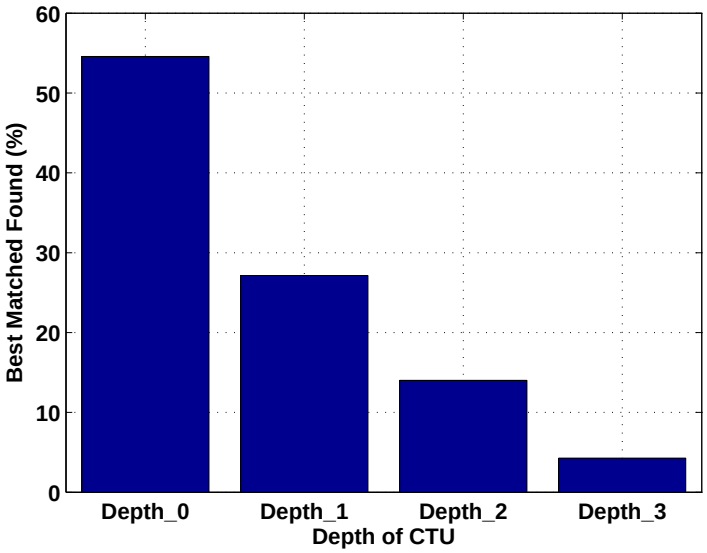


Figure 3.2: Percentage of best matched blocks found on each depth for video sequences as given in Table 3.1.

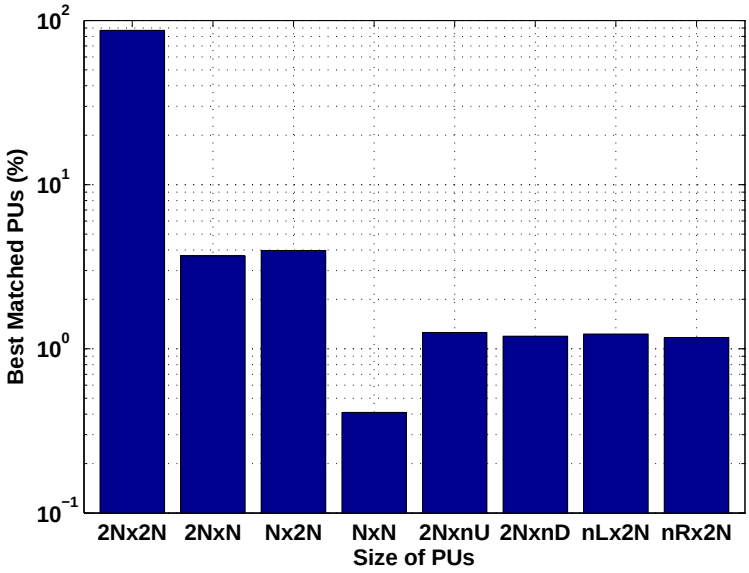


Figure 3.3: Best matched PU sizes found among all depths for video sequences as given in Table 3.1.

is very small or background of an image. On the other hand, small PBs are selected when a texture is very complex [1] which are evident from Fig. 3.2, 3.3 and 3.4 also. This motivates to develop a low complexity algorithm using the adaptive search area i.e., the small search area for larger PBs and vice-versa. We performed experiments using different SRs for different CU sizes as shown in Table 3.2 using different video sequences as shown in Table 3.1. The

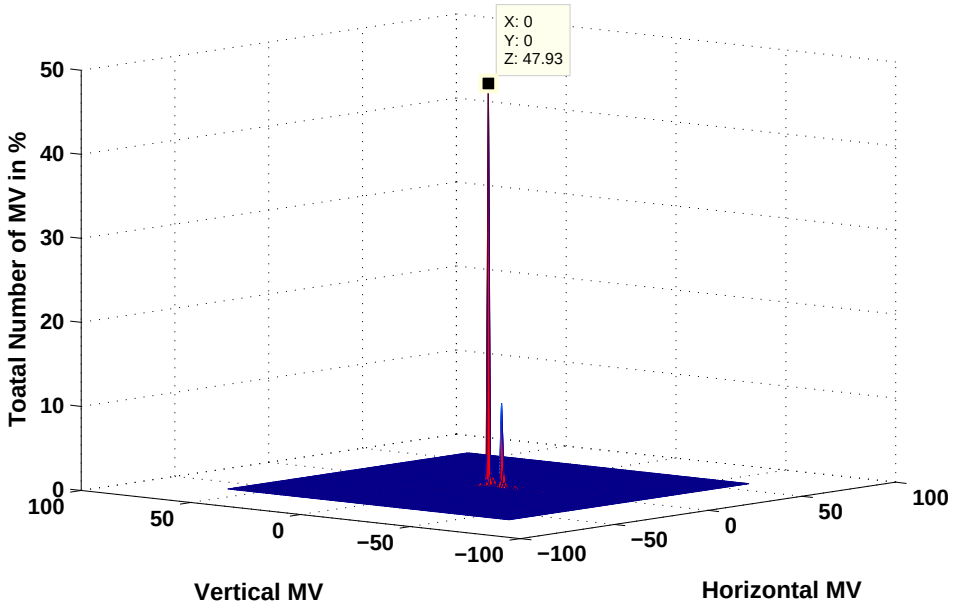


Figure 3.4: Motion vector distribution for different type of video sequences(using the first eight video sequences video_a to video_h as given in Table 3.1)

bit-rate and PSNRs obtained by the proposed adaptive SR based HEVC are compared with reference software HM based HEVC realization and the values are tabulated in Table 3.2. From Table 3.2 it is clear that, in the proposed realizations, the bit rate and PSNR losses are less in high-resolution video sequences compared to low-resolution video sequences. For example, for both the SRs 1-4-16-32 (LCU-SCU) and 2-4-16-32, the average bit-rate loss is around 0.5% with almost the same PSNR.

3.3.2 Early Skip Some of the CU Partitioning

In natural video sequences, the correlation between successive CUs is very high. So, there is a chance that successive CUs have the same type of motion and PBs. From the simulation studies it was found that if the depth of previously encoding CU is 0, the chances of the current CU partitioned up to depth 3 is only 0.9%. Therefore, based on the depth of the previously encoded CU, we can skip depth 3 partitioning of the current CU. This reduces the computational complexity involved in ME.

Table 3.2: PSNR and Bit rate Comparison for different SR using TZS Algorithm

	Bit-rate	Y-PSNR	Bit-rate	Y-PSNR	Bit-rate	Y-PSNR	Bit-rate	Y-PSNR	Bit-rate	Y-PSNR
HM 10.0	186.8245	34.2428	831.24	34.4691	277.7179	38.8014	15907.4	30.9712	2908.5714	36.0129
1-2-4-4*	0.8651 %	0.0088	4.7543 %	0.0757	-1.1371 %	-0.0143	0.2211 %	-0.0019	0.5992 %	-0.0020
1-2-4-8	0.7821 %	-0.0078	2.0042 %	-0.0082	-0.9734 %	-0.0106	0.3278 %	0.0050	0.6051 %	-0.0050
1-2-8-16	0.6379 %	-0.0003	1.2728 %	-0.0018	-0.6913 %	-0.0044	0.1625 %	0.0062	0.2613 %	-0.0062
1-4-16-32	0.6554 %	0.0027	0.5005 %	-0.0110	-0.9006 %	0.0016	0.2473 %	0.0040	0.5265 %	0.0023
2-4-16-32	0.6554 %	0.0027	0.3898 %	-0.0062	-1.0097 %	-0.0086	0.2865 %	-0.0002	0.3782 %	0.0007
4-8-16-32	0.5156 %	-0.0014	0.5630 %	-0.0025	-0.4184 %	0.0010	0.1380 %	0.0005	0.2986 %	-0.0011
8-16-32-64	1.2191 %	0.0161	0.3994 %	-0.0116	-0.9370 %	-0.0007	0.2044 %	0.0018	0.0629 %	-0.0028
16-16-32-64	1.2191 %	0.0161	0.2406 %	-0.0146	-0.5913 %	-0.0042	0.2044 %	0.0018	0.0678 %	-0.0034

$*$ 1-2-4-4 :- Search range for 64×64-32×32-16×16-8×8 PUs respectively.

3.3.3 Pixel Truncation

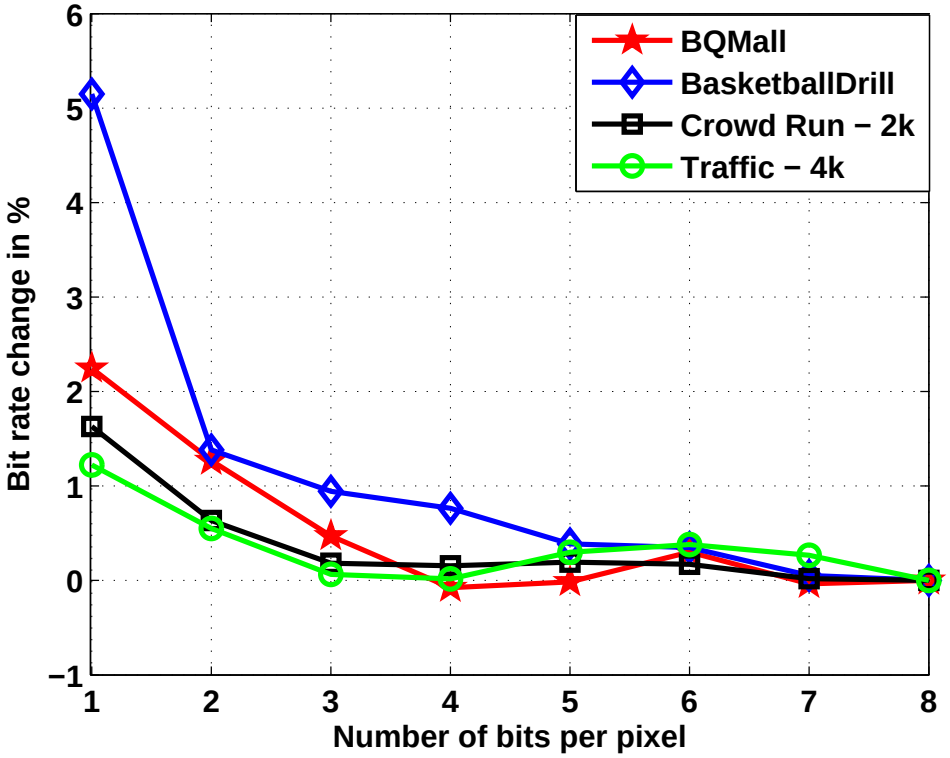


Figure 3.5: Effect of pixel truncation on bit-rate using different video sequences in the HM reference software.

For reducing the computational complexity in ME, pixel truncation is one of the most effective technique. Truncating least significant bits (LSBs) in the calculation of a sum of the absolute difference in ME algorithm does not affect the accuracy much. It can be easily observed from Fig. 3.5 and Fig. 3.6 that the truncation of up to four LSBs of the pixels results in 1% increment of bit rate and 0.01 dB decrement of average PSNR. However, reducing the pixel size by four bits saves 50% computational power in the calculation of the SAD involved in ME. It can be noted that the peak at 5 bits/pixel in case of BasketballDrill video sequence is due to its complex motion. The complex motion caused by some video sequences, in fact, results in irregularity in the PSNR plots.

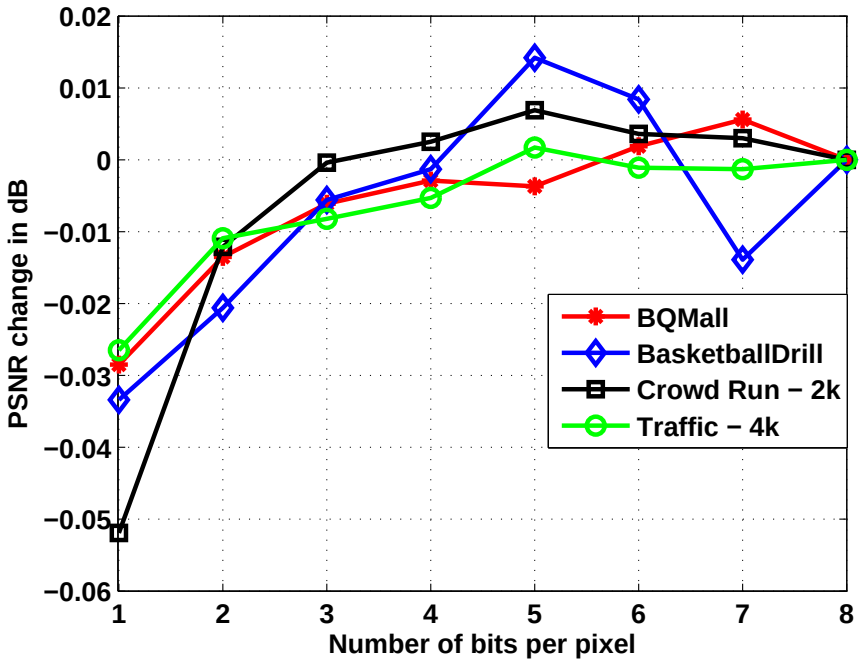


Figure 3.6: Effect of pixel truncation on PSNR using different video sequences in the HM reference software.

3.3.4 Skip Some of PU Partitions

It can be clearly observed from Fig. 3.3 that some of the PU partitions are rarely used. So, removing these partitions will reduce the computational complexity greatly. The effect of removing some of the PU modes on PSNR and bit-rate is presented in Table 3.3. It is clear from Table 3.3 that the losses in the bit rate and PSNR are very less in high-resolution video sequences. For example, in case of $SqrWith8 \times 4\&16 \times 8 - 16 \times 12$ (i.e., without the asymmetrical partition for depth 0 and 1) leads to the average bit-rate increment of less than 1.5% with negligible loss in PSNR. The average increment in bit-rate in each of the following four case is less than 2% with negligible degradation in PSNR: (i) asymmetrical partitions from all PUs, (ii) asymmetrical partition for larger PU (64×64 and 32×32), (iii) without depth 3 partitioning and (iv) symmetrical plus asymmetrical for larger PU (64×64 and 32×32). Hence, removing asymmetrical PU partition at depth 0 and 1 does not affect much on bit-rate and PSNR, but it reduces computational complexity to the great extent.

Table 3.3: PSNR and Bit rate Comparison for suitable partition using TZS Algorithm

Used PU Sizes	BasketballPass (416x240_50fps)		BasketballDrill (832x480_60fps)		KristenAndSara (1280x720_60fps)		crowd_run (1080p50)		Traffic (2560x1600_30_crop)	
	Bit-rate	Y-PSNR	Bit-rate	Y-PSNR	Bit-rate	Y-PSNR	Bit-rate	Y-PSNR	Bit-rate	Y-PSNR
HM 10.0	186.8245	34.2428	831.24	34.4691	277.7179	38.8014	15907.4	30.9712	2908.5714	36.0129
SqrOnly$^{\alpha}$	6.6372 %	-0.0839	1.5567 %	-0.0342	1.5646 %	-0.0471	2.3797 %	-0.0669	2.5580 %	-0.0435
SqrSym$^{\beta}$	2.9800 %	-0.0123	1.1669 %	-0.0032	0.7641 %	-0.0138	0.7328 %	0.0034	1.5265 %	-0.0113
SqrSymWithout 8x4$^{\gamma}$	6.3008 %	-0.0819	0.6641 %	-0.0241	1.7378 %	-0.0219	2.1920 %	-0.0575	1.7576 %	-0.0314
SqrSymAsym 64x64$^{\delta}$	2.1847 %	-0.0002	0.4162 %	-0.0226	0.1819 %	-0.0170	0.6495 %	0.0005	1.5766 %	-0.0078
SqrSymAsym Without 8x4$^{\varepsilon}$	5.6017 %	-0.0816	1.2680 %	-0.0184	1.6010 %	-0.0281	2.0659 %	-0.0603	1.8743 %	-0.0291
SqrWith8x4$^{\eta}$	2.7527 %	-0.0185	1.5904 %	-0.0297	0.9370 %	-0.0428	1.2205 %	-0.0147	2.1090 %	-0.0320
SqrWith8x4& 16x8-16x12$^{\theta}$	1.1798 %	0.0071	1.0490 %	-0.0067	1.0916 %	-0.0230	0.3519 %	0.0082	1.4656 %	-0.0185
OrgWithout 8x4$^{\lambda}$	3.3907 %	-0.0482	0.2310 %	-0.0166	-0.2183 %	-0.0130	1.1966 %	-0.0479	0.6140 %	-0.0132
OrgWithout 8x4&16x4$^{\mu}$	5.8289 %	-0.0944	0.5269 %	-0.0216	1.9285 %	-0.0206	2.1462 %	-0.0556	1.1464 %	-0.0257
OrgWithout Depth3$^{\chi}$	4.8501 %	-0.1053	0.1660 %	-0.0268	0.2911 %	-0.0168	2.4133 %	-0.0998	0.5953 %	-0.0235

α:- Square only partition; β:- Without asymmetrical partitions; γ:- Without asymmetrical and 8×4; δ:- Without asymmetrical partitions for depth 1,2 and 3; ε:- Without asymmetrical partitions for depth 1,2 and 3 and 8×4 ; η:- Square with depth 3 partitions; θ:- Without asymmetrical for depth 0 and 1; λ:- Without 8×4 partition; μ:- without asymmetrical partition for depth 2 and 8×4 partition; χ:- Without depth 3

3.3.5 Sub-Sampling

For reducing the computational complexity, sub-sampling techniques is one of the most suitable techniques. Sub-sampling a frame by 2 reduces computational complexity by 50%. But, sub-sampling may results in information loss or calculated best-matched may not be same as the actual best matched in the search window. In natural video sequences, the difference between neighboring pixels is very small. So, the information may recover by simply multiplying the sub-sample SAD value with sub-sampling value. Suppose sub-sampling value is 2, then final SAD is equal to sub-sample SAD $\times$ 2. The effect of sub-sampling on PSNR and bit-rate for different video sequences are presented in Table 3.4 for two different cases as illustrated below

- Case 1:- Final SAD = Sub-sample SAD.
- Case 2:- Final SAD = Sub-sample SAD $\times$ 2.

Table 3.4: Sub-sampling effect on PSNR and bit-rate.

VS	Original		Case 1		Case 2	
	PSNR(dB)	Bit-rate	△PSNR	△Bit-rate	△PSNR	△Bit-rate
Video_a	34.2428	186.8245	-0.0021	3.0531	-0.0008	2.4490
Video_b	33.8031	892.8200	-0.0025	1.4200	-0.0016	0.1000
Video_d	38.9985	202.2063	-0.0026	1.6611	-0.0005	0.2611
Video_f	30.9712	15907.4	-0.0034	20.0857	-0.0018	17.7714
Video_h	36.0129	2908.5714	-0.0040	12.4572	-0.0004	8.7143

.

It can be clearly observed from the Table 3.4 that information loss may be reduced by simply multiplying the sub-sampling value with the sub-sample SAD. This is possible only for small value of sub-sampling because of successive pixel value difference is very small. Hence, we used case 2 in the proposed algorithm.

3.4 Proposed Algorithm

As we presented in the previous section (3.3), the effects of pixel truncation, removing asymmetrical modes for depth 0 and depth 1, adaptive SR and sub-sampling on bit-rate and PSNR is marginal. Therefore, we proposed a computationally efficient ME algorithm by incorporating all the above-mentioned techniques and using the fixed number of search points. We labelled the proposed algorithm as hexagonal grid search (HGS) because it searches in a grid pattern in only a hexagonal manner. The salient features of the proposed algorithm are as follows:

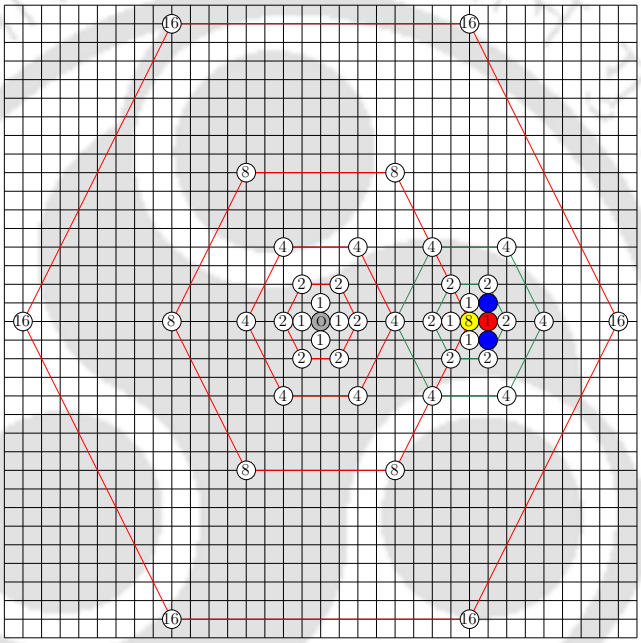


Figure 3.7: Illustrations of the proposed search algorithm.

- The diamond search pattern used in the TZS algorithm utilizes the variable number of search points. Normally, as the size of the grid increases, the number of search points also increases. However, in the proposed hexagonal grid search, we have fixed the number of search points to six (except from the first grid, where 4 points are used). The use of the fixed number of search point at each grid results in low complexity VLSI architecture at the hardware level.

- Unlike TZS, the proposed algorithm uses only hexagonal search pattern. In TZS algorithm, both diamond and raster scans will be performed. The raster scan is performed

only when the MV value in the first refinement step is greater than 5.

- The proposed algorithm uses the same search pattern as the original hexagonal search algorithm. But, the proposed algorithm searches in the grid. The difference in the flow of the proposed HGS and hexagonal search algorithm are given in Fig. 3.7 and Fig. 3.8 respectively. It is clear from Fig. 3.7 that search in the proposed HGS algorithm in more dynamic as compared to the original hexagonal search algorithm.

- The improved TZ search algorithm in [112] uses two different search patterns like hexagonal and raster. For initial search, this algorithm uses the hexagonal search pattern. However, the hexagonal grid search will be performed only at distance 1, 2, and 4. If the best match is not found within 4 pixel distance, then this algorithm uses raster search pattern. In contrast, the proposed HGS algorithm uses only one search pattern (i. e. hexagonal) with fixed number of search at each grid (except the first).

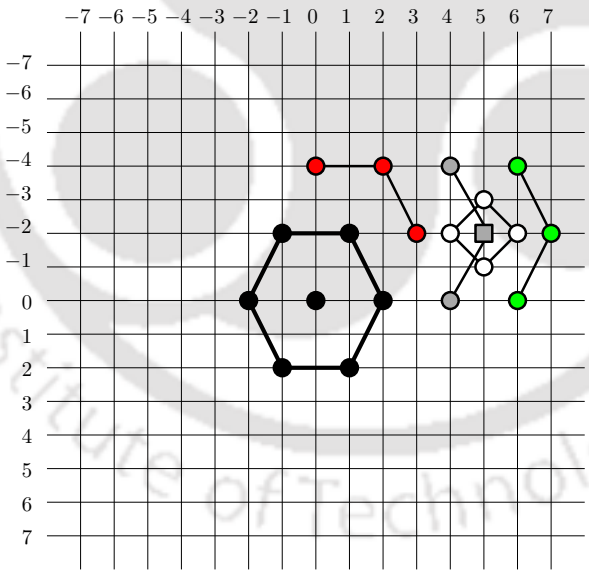


Figure 3.8: Flow of the original hexagonal search pattern [3].

The flowchart of the proposed HGS algorithm shown in Fig. 3.9, consists of the following steps:

- *Step 1: Motion Vector Prediction*: - This stage calculates the distortion at origin and PMV position to choose the best one as a seed.

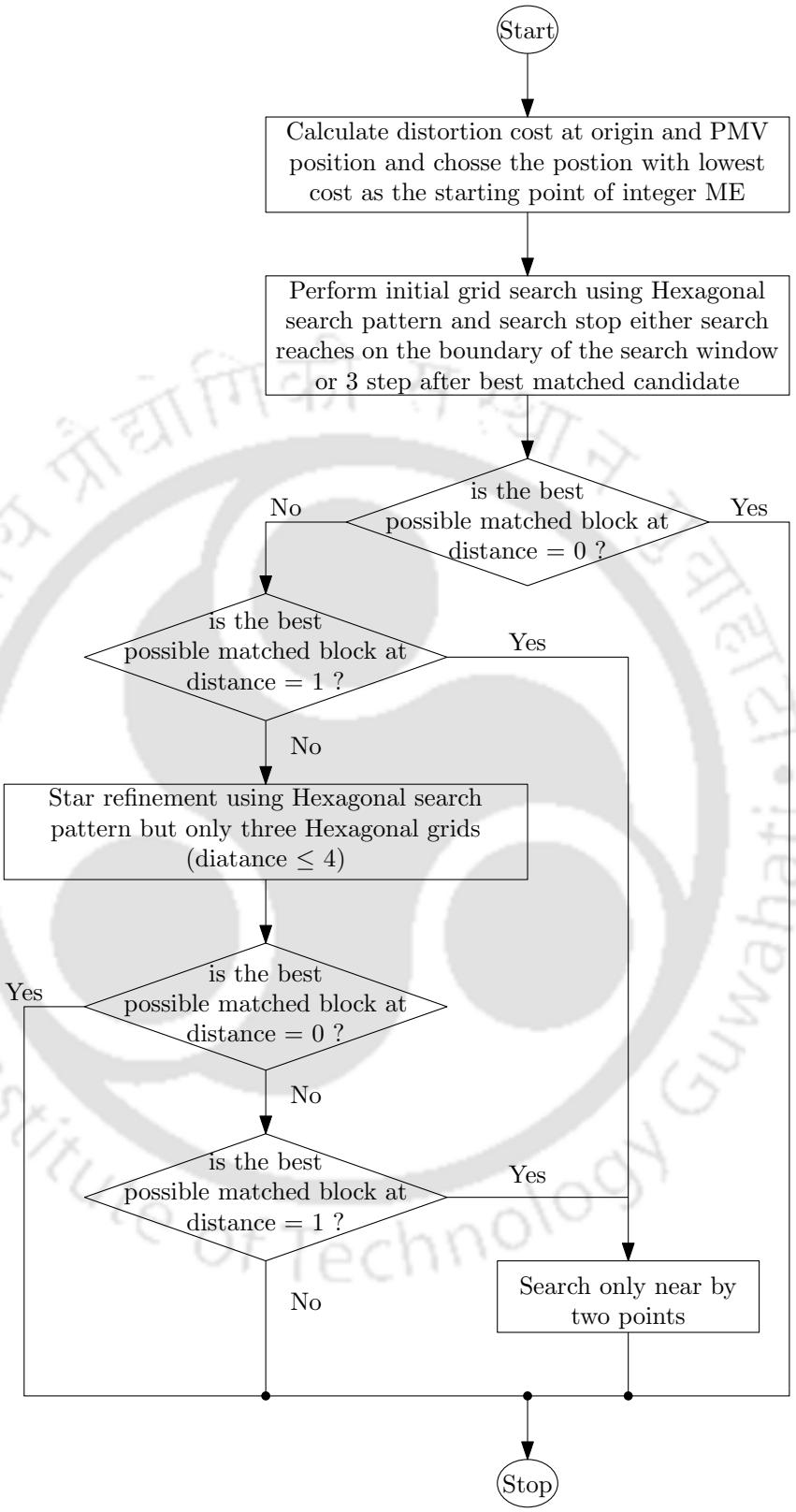


Figure 3.9: Flowchart of the proposed motion estimation algorithm.

- *Step 2: Initial Grid Search*: - Unlike TZS algorithm, the proposed algorithm uses only hexagonal search pattern with six search points for all distances > 1. Depending on the distance of the best-matched candidate, the search process will either be terminated or we perform one of the following two steps namely, star refinement or two-point search.

- *Step 3: Star Refinement*: - This step is performed only when the distance of the best-matched candidate in the previous step is greater than one. This step also uses a hexagonal search pattern like TZS but the major difference in the proposed algorithm is that this algorithm tests only three hexagonal grids (distances ≤ 4). Again, depending on the distance of the best-matched candidate block, the search process will either stop or go to two point search (Step 4).

- *Step 4: Two-Point Search*: - This step is executed only when the distance of the best-matched block in the previous step is equal to one. It executes only once and gives the final best possible matched candidate in the search window.

3.5 Simulation Results

In order to validate the proposed algorithm, we carried out simulations on various resolution video sequences (VS) ranging from 416×240 to 2560×1600 using the reference software HM with *encoder_lowdelay_P_main* and *encoder_randomaccess_main* profiles. All the simulations are performed on Intel(R) Core i5-2400 @ 3.10 GHz (8GB RAM) machine. To verify the versatility of the proposed algorithm, various parameters such as bit rate, PSNR and average search points (ASP) are obtained using the video sequences listed in Table 3.1 and obtained results are tabulated in Table 3.5 and Table 3.6. Some of these videos contain slow moving objects and the others are fast moving objects. For reducing computational complexity, we employed sub-sampling, the fixed number of search point at each grid (i.e., 6 search points) and 6 bits per pixel. None of the other algorithms listed in Table 3.5 and Table 3.6 have employed sub-sampling and pixel truncation techniques. As compared to TZS algorithm, the

Table 3.5: PSNR, bit-rate and average search points (ASP) comparison with other state of the art algorithms

VS	△PSNR			△Bit-rate			△ASP		
	TZS	**M14 [104]**	**HGS**	**TZS**	**M14 [104]** (%)	**HGS**(%)	**TZS**	**M14 [104]** (%)	**HGS**(%)
Video_a	34.2428	0.0071	-0.021	186.8245	1.2671	1.6779	75880	-52.47	-56.78
Video_b	33.8031	-0.0014	-0.0058	892.8200	0.6608	-0.2307	100051	-62.68	-64.81
Video_c	34.4691	-0.0011	0.0023	831.2400	1.0683	1.4845	145809	-69.06	-73.78
Video_d	38.9985	-0.0037	0.0004	202.2063	-2.8111	-1.7741	44656	-25.70	-32.03
Video_e	38.8014	-0.0037	-0.0212	277.7179	-0.7368	-0.8915	53351	-30.61	-41.26
Video_f	30.9712	-0.0005	-0.0054	15907.4000	0.2164	0.3495	195461	-54.23	-74.05
Video_g	33.9888	0.0009	-0.0044	1410.5143	0.3707	-0.7637	62065	-5.16	-40.48
Video_h	36.0129	0.0008	0.0004	2908.5714	0.5314	0.3899	65826	-21.47	-46.34
AC	–	-0.0002	-0.0068	–	0.0709	0.0302	–	-40.17	-53.69

Table 3.6: PSNR, bit-rate and ASP comparison with ASRFME [17]

VS	△PSNR		△Bit-rate		△ASP	
	ASRFME [17]	**HGS**	**ASRFME [17]**	**HGS**(%)	**ASRFME [17]**	**HGS**(%)
Video_a	34.2385	-0.0166	192.7592	-1.9269	22327	-12.4803
Video_b	33.7842	0.0141	913.54	-2.0141	24477	-15.3605
Video_c	34.434	0.0153	846.76	-0.1228	26607	-18.7638
Video_d	38.9539	0.0444	203.2926	-1.5907	20910	-11.7439
Video_e	38.7535	0.0205	281.5579	-2.5393	21491	-12.1031
Video_f	30.9533	0.0076	16121.429	-0.8294	33681	-15.0167
Video_g	33.9721	0.0145	1435.5714	-2.3505	25083	-11.9450
Video_h	35.9853	0.0234	2978.943	-1.6439	23887	-11.9986
Average change	–	0.0154	–	-1.6272	–	-13.6765

proposed HGS algorithm requires 53.69% fewer search points with 0.0068 dB loss in PSNR. Similarly, as compared to directional search ME algorithm [104], the proposed HGS algorithm requires 13.55% fewer search points with slightly lower bit-rate and little degradation in PSNR as shown in Table 3.5. From Table 3.5, it is evident that the proposed algorithm takes less number of search points without much degradation in PSNR. Moreover, the proposed algorithm is suitable for all types of video sequences.

The proposed algorithm is also compared with ASRFMEA [17] using the same adaptive SR and the results are shown in Table 3.6. As compared to ASRFMEA, the proposed HGS algorithm requires 13.6765% less number of search points with slightly lower bit rate and good PSNR as shown in the Table 3.6. It can be noted that both algorithms do not support asymmetrical PU partitions.

Table 3.7: PSNR, Bit-rate and ASP comparison using 4-bit adaptive search range and without asymmetrical partition for depth 0 and 1 HGS algorithm with original TZS algorithm.

VS	Algo.	△Bit-rate	△PSNR			△ASP
			Y-PSNR	U-PSNR	V-PSNR	
Video_a	TZS	186.8245	34.2428	39.323	38.4372	2124633
	HGS	1.8400 %	-0.0292	-0.2097	0.0479	-58.25 %
Video_b	TZS	892.82	33.8031	38.9462	39.6993	10405209
	HGS	0.3091%	-0.0146	-0.0203	-0.0939	-66.77%
Video_c	TZS	831.2400	34.4691	38.5152	38.5164	15164084
	HGS	2.0260 %	-0.0222	-0.0144	-0.0895	-75.99 %
Video_d	TZS	202.2063	38.9985	44.4409	44.9334	10717369
	HGS	-0.2749 %	-0.0169	-0.0281	0.1211	-32.44 %
Video_e	TZS	277.7179	38.8014	43.5303	44.5026	12804106
	HGS	-0.0728%	-0.0269	-0.0214	-0.0215	-42.19%
Video_f	TZS	1410.5143	33.9888	36.6212	37.1468	31653000
	HGS	0.1236%	-0.0061	0.0072	0.0168	-42.99%
Video_g	TZS	15907.4	30.9712	33.4719	34.1747	99685000
	HGS	0.4334 %	-0.0068	-0.0287	-0.0067	-77.62 %
Video_h	TZS	2908.5714	36.0129	37.9904	40.2847	65825786
	HGS	1.2623 %	-0.0169	-0.0043	0.0058	-47.69 %

The results of the proposed HGS algorithm with 4-bit truncation, adaptive SR, without asymmetrical partition for depth 0 and 1 and sub-sampling by two in both directions are

Table 3.8: PSNR, Bit-rate and ASP Comparison with different Quantization Parameter (QP) Values.

VS	QP	△PSNR		△Bit-rate		△ASP	
		TZS	HGS	TZS	HGS (%)	TZS	HGS (%)
Video_a	24	39.7586	-0.0341	614.8408	0.7515	2639735	-63.46
	28	36.8819	-0.0149	343.0286	0.4355	2395347	-61.75
	32	34.2428	-0.0292	186.8245	1.8395	2124633	-58.25
	36	31.9179	-0.0265	109.8694	1.1783	1896000	-54.38
	40	29.7175	-0.0410	62.9878	1.7884	1678258	-49.78
Video_b	24	38.5555	-0.0124	3131.46	0.9510	13329125	-71.41
	28	36.2183	-0.0076	1631.28	0.9870	11920709	-70.01
	32	33.8031	-0.0146	892.82	0.8091	10405209	-66.77
	36	31.4087	-0.0165	507.06	1.1884	9003792	-62.81
	40	29.0567	-0.0201	287.36	1.2873	7761250	-58.66
Video_c	24	39.2437	-0.0323	2868.1800	0.7796	20661625	-80.74
	28	36.7636	-0.0088	1523.2600	1.7633	17986917	-79.02
	32	34.4691	-0.0222	831.2400	2.0259	15164084	-75.99
	36	32.4360	-0.0155	484.4200	2.9685	12710167	-72.63
	40	30.4406	-0.0351	290.8200	1.9600	10582375	-68.76
Video_d	24	41.8791	-0.0112	1213.7937	0.1207	13250737	-39.66
	28	40.5291	-0.0194	428.5389	0.4127	11806737	-36.87
	32	38.9985	-0.0169	202.2063	0.2749	10717369	-32.44
	36	37.1983	-0.0135	116.8421	1.1243	9914685	-28.77
	40	35.1395	-0.0059	69.2463	1.5903	9545474	-28.07
Video_e	24	42.2921	-0.0050	1181.4568	0.2630	16650843	-51.03
	28	40.6768	-0.0179	529.0863	1.4898	14538948	-47.40
	32	38.8014	-0.0269	277.7179	-0.0728	12804106	-42.19
	36	36.7164	-0.0005	158.8295	1.5746	11800106	-38.79
	40	34.5030	0.0088	97.6674	-1.0864	10910316	-36.13
Video_f	24	36.0845	-0.0085	16505.2	1.0782	48062143	-57.59
	28	35.1169	-0.0027	4068.5143	0.7430	35799715	-47.93
	32	33.9888	-0.0061	1410.5143	0.1236	31653000	-42.99
	36	32.5665	-0.0213	722.5714	2.5544	29020286	-39.26
	40	30.8520	-0.0362	387.6	3.2139	25745000	-34.79
Video_g	24	36.0845	-0.0085	16505.2000	-1.1782	48062143	-57.59
	28	35.1169	-0.0027	4068.5143	0.7429	35799715	-47.93
	32	33.9888	-0.0061	1410.5143	0.1236	31653000	-42.99
	36	32.5665	-0.0213	722.5714	2.5544	29020286	-39.26
	40	30.8520	-0.0362	387.6000	3.2139	25745000	-34.79
Video_h	24	40.2706	-0.0100	14904.5429	0.2785	82007143	-54.49
	28	38.1136	-0.0060	6117.4286	1.3110	73751429	-51.72
	32	36.0129	-0.0169	2908.5714	1.2623	65825786	-47.69
	36	33.8812	-0.0134	1559.9429	1.9744	59528429	-43.89
	40	31.6725	-0.0084	847.2	2.4416	53822786	-40.15

shown in Table 3.7. As compared to the original TZS algorithm, the proposed HGS algorithm saves 55.49% search points with around 1% increment in bit rate and negligible degradation in PSNR. For checking versatility of the proposed algorithm, we have also examined the effect on PSNR, bit rate and an average number of search points using a different value of QP and the results are shown in the Table 3.8. It is clear from Table 3.8 that the proposed HGS algorithm

requires around 50% fewer search points with almost same PSNR and slight increment in bit rate as compared to the TZS algorithm for different value of QP. The loss in bit rate increases as the QP value increases, which is clear from in Table 3.8.

Table 3.9: BD-PSNR, BD-rate and MET comparison using 4-bit adaptive search range and without asymmetrical partition for depth 0 and 1 HGS algorithm with original TZS algorithm.

VS	Lowdelay P Main			Random Access Main		
	BD-rate*	BD-PSNR**	MET*	BD-rate*	BD-PSNR**	MET*
Video_a	0.89	-0.0804	-52.22	0.78	-0.1043	-71.73
Video_b	1.26	-0.0098	-64.88	1.38	-0.0139	-75.21
Video_c	1.42	-0.0212	-66.06	1.61	-0.0259	-88.05
Video_d	0.67	-0.0392	-35.85	0.85	-0.0473	-42.78
Video_f	1.07	-0.0169	-48.27	0.98	-0.0225	-61.03
Video_h	1.13	-0.0222	-42.89	1.52	-0.0424	-69.47

Note: - $\star$ = (%); $\star\star$ = ($\triangle$dB)

The BD-PSNR [113], BD-rate and motion estimation time (MET) of the proposed HGS algorithm are compared with original TZS algorithm and are tabulated in Table 3.9. The Bjntegaard Delta (BD) model is used to calculate the average PSNR and bit-rate differences between two rate-distortion (R-D) curves. In other words, it computes the average distance between two RD-curves. Two types of BD measurements are available: (a) BD-PSNR, which corresponds to the average PSNR difference in dB for the same bit rate and (b) BD-Rate, which corresponds to the average bit rate difference in percent for the same PSNR. Motion estimation time (MET) is the time required to execute the motion estimation process. It is clear from Table 3.9 that the proposed algorithm saves more than 50% and 60% motion estimation time as compared to the original TZS algorithm with approximately 1% increment in BD-rate for *encoder_lowdelay_P_main* and *encoder_randomaccess_main* profiles respectively. For video sequences like BasketballDrill_832x480_f500_60 (Video_c), the proposed algorithm reduces 88% motion estimation time for *encoder_randomaccess_main* profile with 1.61% increment in BD-rate. The rate distortion results for video sequences *BasketballPass*_416 $\times$ 240@50 and *Johnny*_1280 $\times$ 720@60 using *encoder_lowdelay_P_main* profile for QP value 32 is shown in Fig. 3.10. The BD-PSNR loss as compared to reference software HM in *BasketballPass*_416$\times$

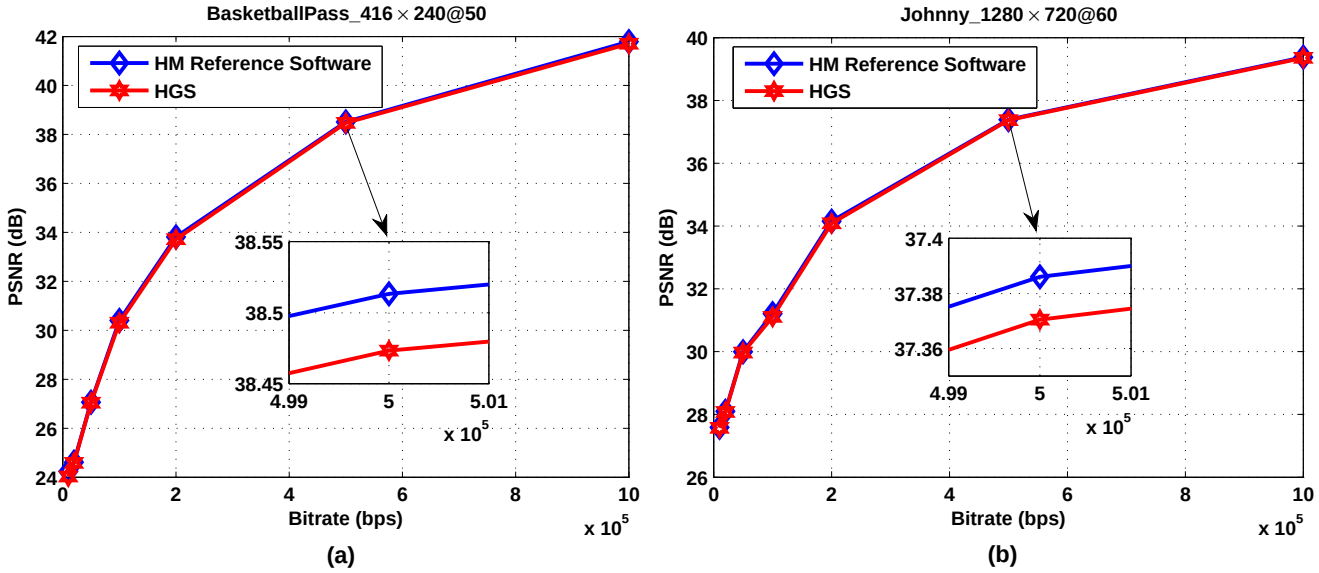


Figure 3.10: Rate-distortion results for video sequence (a) *BasketballPass*_416 × 240@50 and (b) *Johnny*_1280 × 720@60 using *encoder_lowdelay_P_main* profile with QP value of 32.

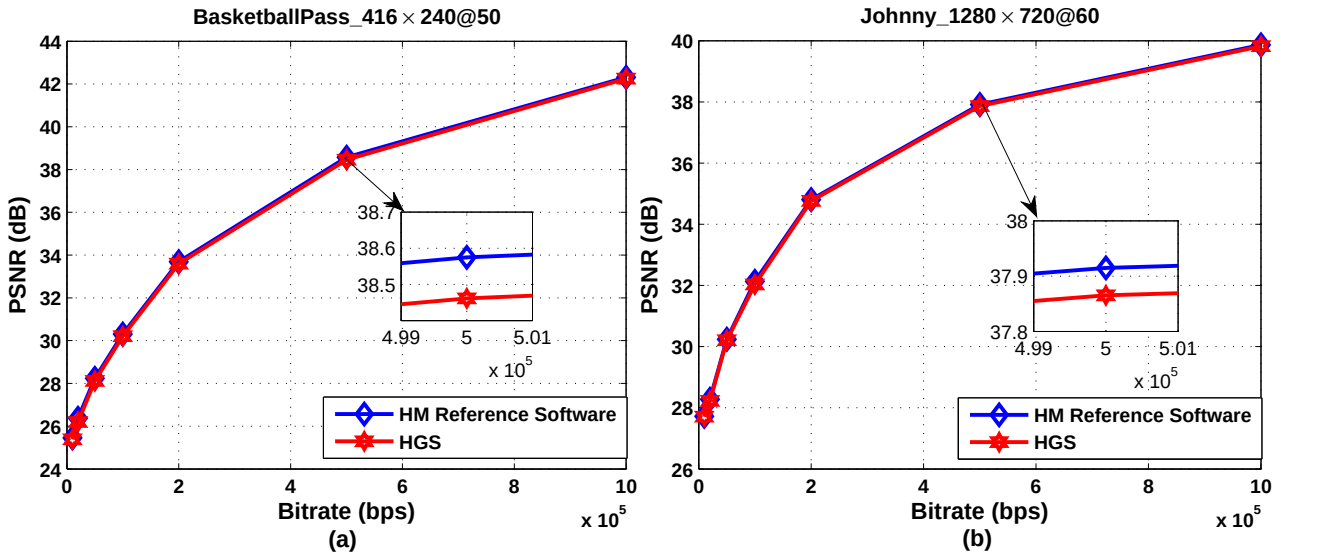


Figure 3.11: Rate-distortion results for video sequence (a) *BasketballPass*_416 × 240@50 and (b) *Johnny*_1280 × 720@60 using *encoder_randomaccess_main* profile with QP value of 32.

240@50 and *Johnny*_1280 × 720@60 video sequences are 0.0804 dB and 0.0392 dB respectively. It is clear from Fig. 3.10 that the target bit rate increases as the loss in the BD-PSNR decreases. Fig. 3.11 shows the results for *encoder_randomaccess_main* profile using QP value 32. The BD-PSNR loss for *encoder_randomaccess_main* profile are 0.1043 dB and 0.0473 dB

for $BasketballPass\_416\times240@50$ and $Johnny\_1280\times720@60$ video sequences respectively. The proposed HGS IME algorithm requires fewer number of search points and loss in BD-PSNR is also less for *encoder_lowdelay_P_main* and *encoder_randomaccess_main* profiles.

3.6 Conclusion

In this chapter, we have proposed a computationally efficient ME algorithm for HEVC. In order to reduce the computational complexity, we used the pixel truncation, adaptive SR, removing some of the PU partitionings, and sub-sampling techniques. The proposed HGS algorithm takes 55.49% fewer search points as compared to TZS algorithm. It is found that the effect of these techniques on bit rate is marginal with nearly same PSNR. The proposed HEVC realization enjoys less computational complexity and hence it can be employed in low power, less area portable devices. The BD-PSNR loss as compared to reference software HM using the *encoder_randomaccess_main* profile for video sequences like $BasketballPass\_416\times240@50$ and $Johnny\_1280\times720@60$ is 0.1043 dB and 0.0473 dB respectively

4

Low Power Integer Motion Estimation Algorithm and Architecture for HEVC

Contents

In the previous chapter, we have proposed a computationally efficient IME algorithm. In this algorithm, MVP uses left, left above, left below, top and top right blocks. Thus, it is very difficult to develop a low power architecture for the proposed algorithm due to its dependency on the left side blocks. To overcome this problem, in this chapter, we propose a low-power IME algorithm and architecture of HEVC for consumer applications. The rest of the chapter is organized as follows. The chapter is divided into two parts. The first part describes an IME algorithm which is a modified version of the algorithm presented in Chapter 3. The second part of the chapter focuses on its low-power hardware implementation. Simulation results for the proposed algorithm and architecture are presented in the next section. The conclusion of the work is presented at the end of the chapter.

4.1 Introduction

In the past decade, with the progress in technology, the quality of digital video has improved tremendously. Nowadays, HD videos have attained much popularity and several researchers are working towards the UHD video for next-generation applications. It enables the resolutions of $4K$ (3840×2160) and $8K$ (7680×4320), which is 4 to 16 times the number of pixels per frame compared with today's HD videos. To store, transmit and process such a huge amount of video data, efficient and real-time video compression is essential. Keeping this in mind, ISO/IEC MPEG and ITU-T VCEG formed a JCT-VC in 2010 for developing the next generation video coding standard [21]. As the HEVC is capable of processing huge amount of video data, the hardware implementation of HEVC is very complex and consumes an enormous amount of power for compression. It becomes almost impractical to adopt HEVC in portable devices such as mobile phones and digital cameras due to their limited power sources. To overcome this problem, in this chapter, we present a power-efficient algorithm and architecture for HEVC.

In inter prediction, motion parameters are obtained through the ME, the heart of all existing video compression standards. ME involves a two-step process namely, IME and FME. In IME, the best-matched block is found at the integer position in the search area. In HEVC, IME is carried out for 593 (13 + 52 + 208 + 320) sub-blocks, which is a very large value as compared to

the number of sub-blocks (41) used in H.264/AVC [90]. In IME, distortion is calculated using the SAD. FME refines luminance IMVs to 1/4-pixel accuracy and chrominance IMVs to 1/8-pixel accuracy [12]. For luminance samples, HEVC uses 8-tap and 7-tap interpolation filter for calculating the pixel value at 1/2 and 1/4-pixel positions respectively. For chrominance samples, the 4-tap interpolation filter is used for calculating the pixel value at 1/8-pixel positions [12]. In FME, distortion is calculated as a SATD. For complex/fast moving objects, FME is more important and complex because there is a very high probability that the best-matched blocks are small in size. FME is searched at half and quarter position for luminance samples. Due to the larger block size, asymmetrical PU partition, quadtree partition for CTU and computational complex interpolation filters, ME becomes a more computationally intensive process in HEVC. The following section details the low complexity realization of IME.

4.2 Existing Integer ME Algorithms and Architectures

The best possible algorithm for finding the best-matched candidate is FS. However, FS involves huge computational complexity. For reducing the complexity involved in ME, several efficient algorithms and architectures are reported in the literature. For example, the works in [1–3, 7, 24, 25, 27–29, 40, 44, 51–53] are based on reducing the number of search points. Pixel truncation technique was employed in [80]. In [81], Zhou et al. used sub-sampling of the images. Ismail et al. in [33], uses early termination technique. In [66, 67, 69, 114] reconfigurable architectures for H.264/AVC are presented. Several low power, non-configurable architectures were also reported in [50, 57, 62, 63, 115]. However, most of the reported architectures are designed to meet the requirements of the H.264/AVC video coding standard.

In [116], Tsai et al., proposed the first architecture for HEVC encoder. The proposed architecture was implemented for UHD specification using 28 nm technology. It dissipates 708 mW of power at 312 MHz for 8192×4320p at 30 fps encoding. In [117], Byun et al., proposed an architecture for IME and is implemented using 64 nm technology. It supports all the PU sizes and operating frequency is 250 MHz. In [106], Biswas et al., proposed an architecture for IME and is implemented on Virtex 6 FPGA and supports adaptive root search motion algorithm.

In [17], ME algorithm and its corresponding architecture are presented. For IME, a predictive search algorithm is proposed and based on the statistical analysis it selects the most probable direction for search. FME algorithm, that depends on PU size and adopts interpolation-free fractional ME for depths 0 and 1, full search mechanism for depth 2, and skips fractional ME for depth 3, is also presented in [17]. Architecture for the proposed algorithm is implemented using the TSMC 90-nm CMOS process, which supports the real-time encoding of 4096×2048 p @ 60 fps. It is operated at 270 MHz with 778.7 K logic gates and 17.4 KB SRAM/on-chip memory.

In [34], a fast ME algorithm based on rotating hexagonal search pattern for coarse search and an adaptive threshold for early termination is presented. In [104], Yang et al., proposed a fast directional search ME algorithm based on the descent search and cross-search patterns. Both the search patterns are executed in parallel. A two-stage ME algorithm and architecture is presented in [110]. In the first stage, coarse ME is performed on truncated pixel and in the second stage, refinement of ME is performed with full pixel resolution with an adaptive search pattern. In the algorithm presented in [118], first, the frames are filtered with the help of multiband-pass filter kernel and then constructs one bit-plane of the image. ME is performed on one bit-plane, which saves computational complexity in the calculation of the SAD. In [33], Ismail et al., proposed a computationally efficient accurate skipping model to speed up any block-based ME algorithm. The reduction in computational complexity for ME process has been achieved in four phases namely, initial search center (ISC), dynamic early stop search termination (DESST), dynamic padding window size (DPWS) techniques and dynamic internal and external stop search technique.

To the best of our knowledge, very few works have so far been reported to reduce the complexity of ME algorithm adapted in HEVC. Moreover, these ME algorithms and architectures consume more power and hence are not suitable for portable devices. In this chapter, we propose a low-efficient new IME algorithm and architecture for HEVC, which is suitable to be employed in portable devices.

4.3 Proposed Modified HGS Algorithm and its Architecture

The proposed IME algorithm presented in Chapter 3 uses a hexagonal search pattern with a fixed number of the search points at each grid (except the first grid). This algorithm utilizes pixel truncation, sub-sampling and adaptive search range techniques for reduction of computational complexity. In natural video sequences, the correlation between successive pixels is very high. So, sub-sampling the picture during the calculation of SAD does not affect the quality much as presented in [81]. From the simulation results presented in Section 3.3, it was found that the percentage of the best matched PBs found at depth 0, 1, 2 and 3 are respectively 55.52, 23.96, 14.78 and 5.74. It was also found that a total of 95% best matched PBs are of size $2N \times 2N$, $2N \times N$ and $N \times 2N$ only, where N can be 8, 16 and 32. We also performed experiments to obtain the MVs behaviour and found that 48% of the MVs are concentrated at origin (0,0) of the search window and 93%, 96.1%, 97.3% and 99.31% of MVs lies in $(\pm 8, \pm 8)$, $(\pm 12, \pm 12)$, $(\pm 16, \pm 16)$ and $(\pm 32, \pm 32)$ ranges of search window respectively. From all the above observations, it can be concluded that the percentage of utilization of depth 3 is only 5.74 and most of the MVs lies nearer to the origin. That means for the complex section of the video depth 3, a larger search window is required. So, instead of using a simple search window we can use an adaptive search window i.e., the small search area for larger PBs and the larger search area for small PBs. We have also noted that if the depth of previously encoding CU is 0, the chances of the current CU partitioned up to depth 3 is only 0.9%. Therefore, based on the depth of the previously encoded CU, we can skip depth 3 partitioning of the current CU. Based on the experimental results, we have chosen search range 2, 4, 8, and 16 for PUs of size 64×64, 32×32, 16×16 and 8×8 respectively.

Traditionally, the MVP in the HGS algorithm uses left, top left, top and top right blocks. For reducing the power, we have employed pipelining in the proposed architecture. Hence, for determination of the MVP of the current block, the left side block is not available. Hence, the proposed IME algorithm presented in Chapter 3, does not support left side blocks in MVP for

the starting point. In the MVP scheme, top, top-left and top-right blocks are used as shown in Fig. 4.1 [16]. Therefore, it is necessary to modify the HGS algorithm accordingly. We labelled the proposed algorithm as modified hexagonal grid search (MHGS). Except for the change in MVP, the flow of the MHGS is the same as HGS and presented in Fig. 4.2. The effect on BD-PSNR and BD-rate due to change in MVP are discussed in the result Section 4.4.

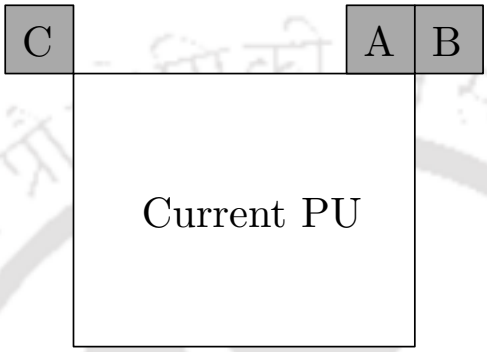


Figure 4.1: Motion Vector Prediction Technique Used [16]

In HEVC, for prediction parameter decision, the following expression is used

$$J_{pred,SAD} = SAD + \lambda_{pred} \times B_{pred} \tag{4.1}$$

Here $J_{pred,SAD}$ is the total cost, SAD is the sum of absolute difference between current block and reference candidate block, λ is the Lagrangian constant and B_{pred} is the bit cost. SAD can be calculated using (2.5). As HEVC supports various sizes of PUs, we developed an architecture for the SAD unit which can calculate the SADs for all the possible PU sizes. Due to sub-sampling, the maximum size of PU is 32×32. Pixel truncation is one of the most efficient techniques to reduce the hardware cost. It is found that truncating up to four LSBs of the pixels results in a 1% increment of bit rate and 0.01 dB decrement of average PSNR. However, reducing the pixel size of four bits saves 50% hardware cost in the calculation of the SAD and SATD units involved in ME. In total distortion, we are interested in the computation of the sum of absolute value. Hence, it does not matter if we subtract the current blocks from the reference block or the reference block from the current blocks. As the pixel value of the current block is fixed and the reference block varies within an SR, therefore, instead of

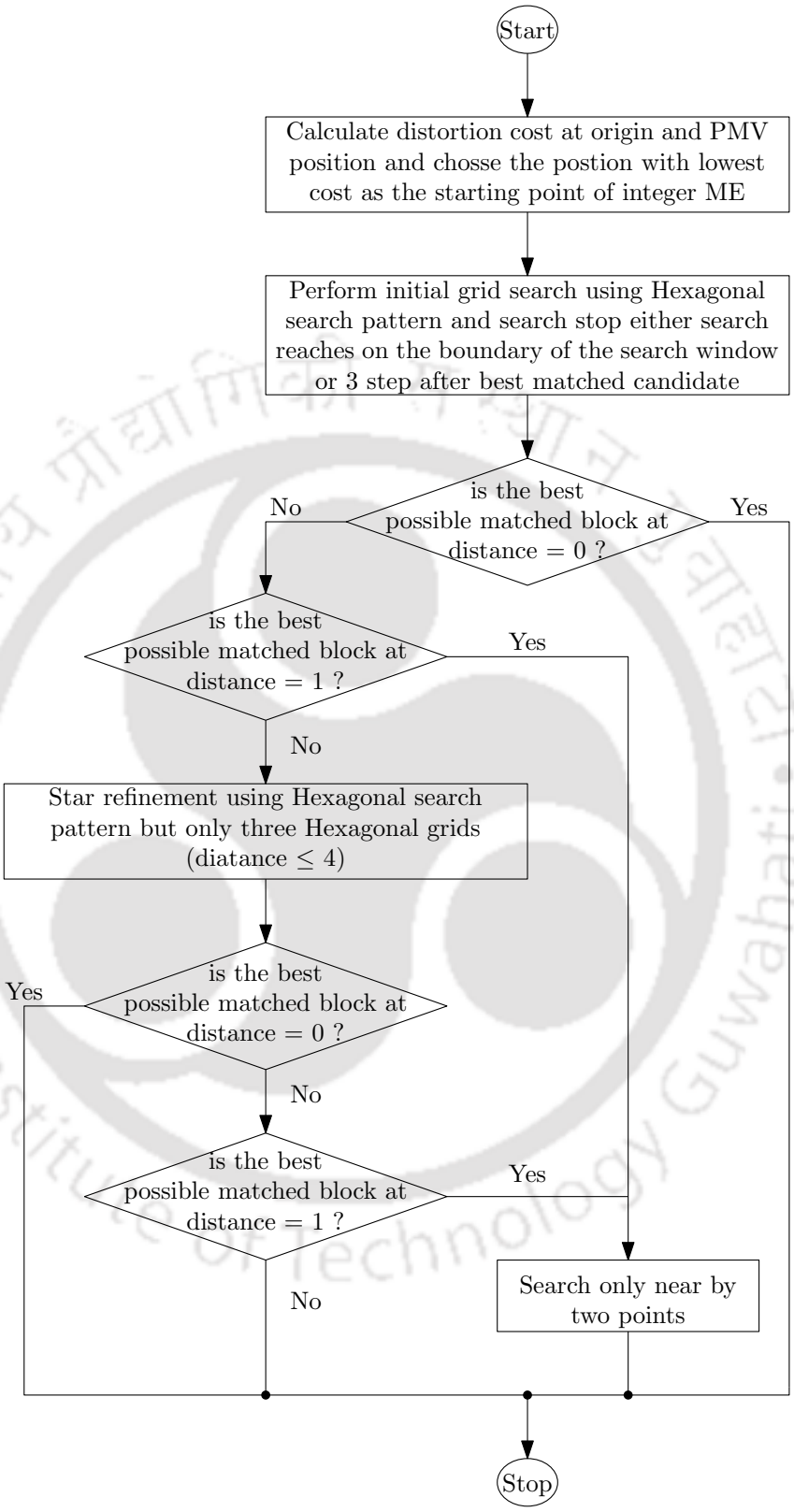


Figure 4.2: Flowchart of the proposed MHGS ME algorithm.

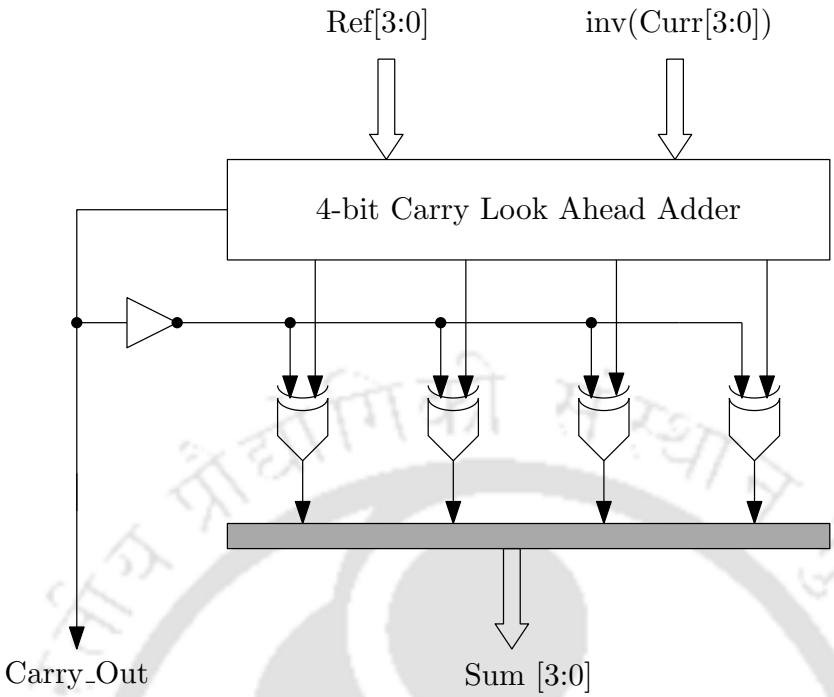


Figure 4.3: Block Diagram of the Proposed Absolute Sum Unit

subtracting reference block from current, we can subtract current from reference. Using this logic, the negation operation on the current block needs to be performed only once in a block search, which saves the power greatly. Thus, a 4-bit absolute sum (ABS) unit can be realized using the 4-bit carry look ahead adder, inverters and XOR gates as shown in Fig. 4.3. It can be noted that in the proposed ABS unit, the current block is subtracted from the reference blocks.

In order to realize larger PUs, the adder tree is required and the size of the adder increases at every stage. As the size of the adder is increased the latency of adder also increases. To reduce the latency of the adder tree, we proposed a SAD tree using 4:2 and 3:2 compressor units as shown in Fig. 4.4. The SAD tree requires less number of gates and have a smaller critical path as compared to normal addition. The proposed SAD tree takes the total of 8 inputs (4 sums + 4 carries) and generates two outputs, namely, sum and carry. Using the ABS and SAD tree unit, we developed a 2×2 SAD unit as shown in Fig. 4.5. In the final stage of the proposed 2×2 SAD unit, a 6-bit carry-lookahead adder, which is a combination of two 3-bit carry-lookahead adders, is used for the final sum of 2×2 pixels. In a similar way, SAD

for larger size PUs can also be obtained.

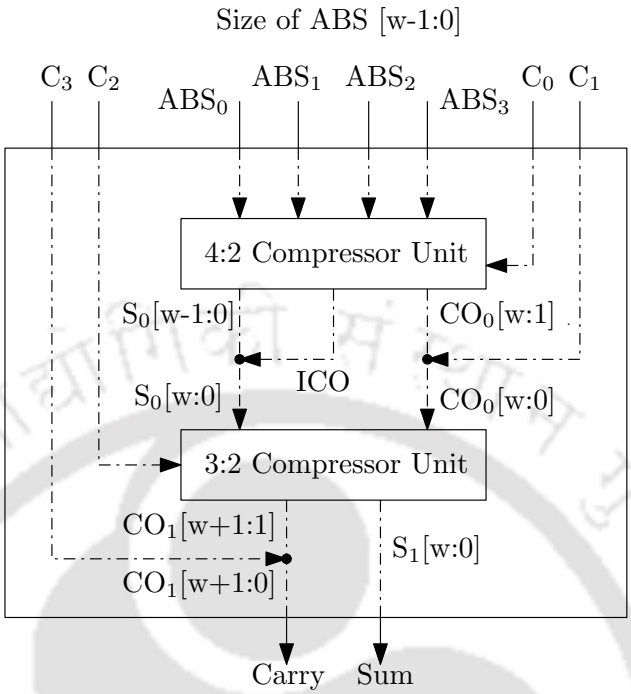


Figure 4.4: Proposed SAD Tree Unit

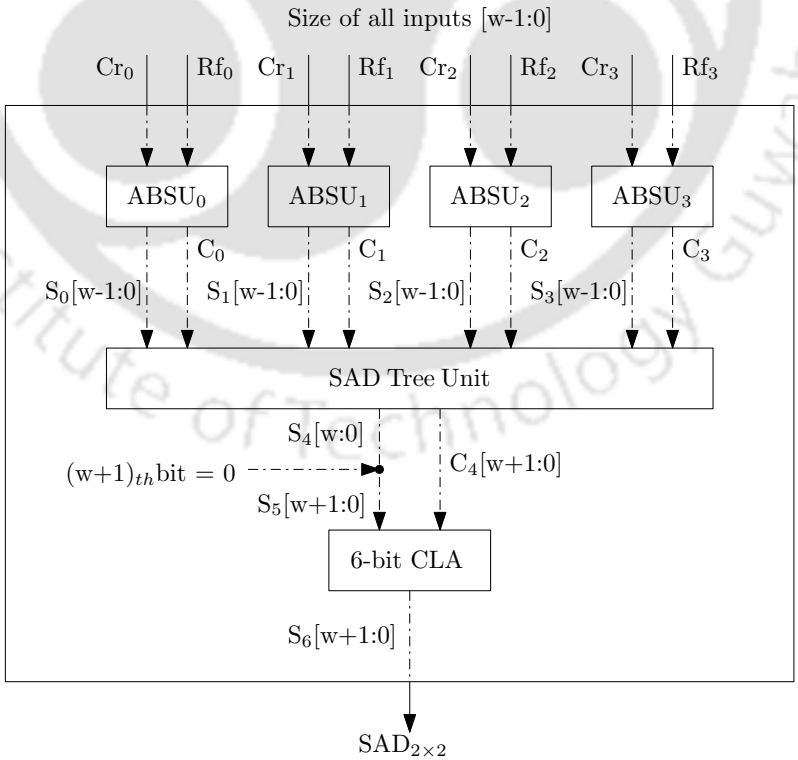


Figure 4.5: Proposed 2×2 SAD Unit

The value of the λ and B are calculated using the RD cost. The value of λ is calculated using the relation provided in the standard (i.e. $\lambda = \alpha \times W_k \times 2^{((QP-12)/3)}$). More details about λ, W_k and α is presented in Section 2.3.2 of Chapter 2. Instead of direct multiplication, we used look-up tables to implement the above expression. The block diagram of the proposed RD cost unit is shown in Fig. 4.6. The proposed RD cost unit contains six MV cost units that work parallelly and calculates the cost for six different positions. The proposed RD cost unit depends on QP, λ, GOP, frame type and the weighting factor. It can be noted from Fig. 4.6 that a two-level pipelining is employed in this architecture to enhance the throughput.

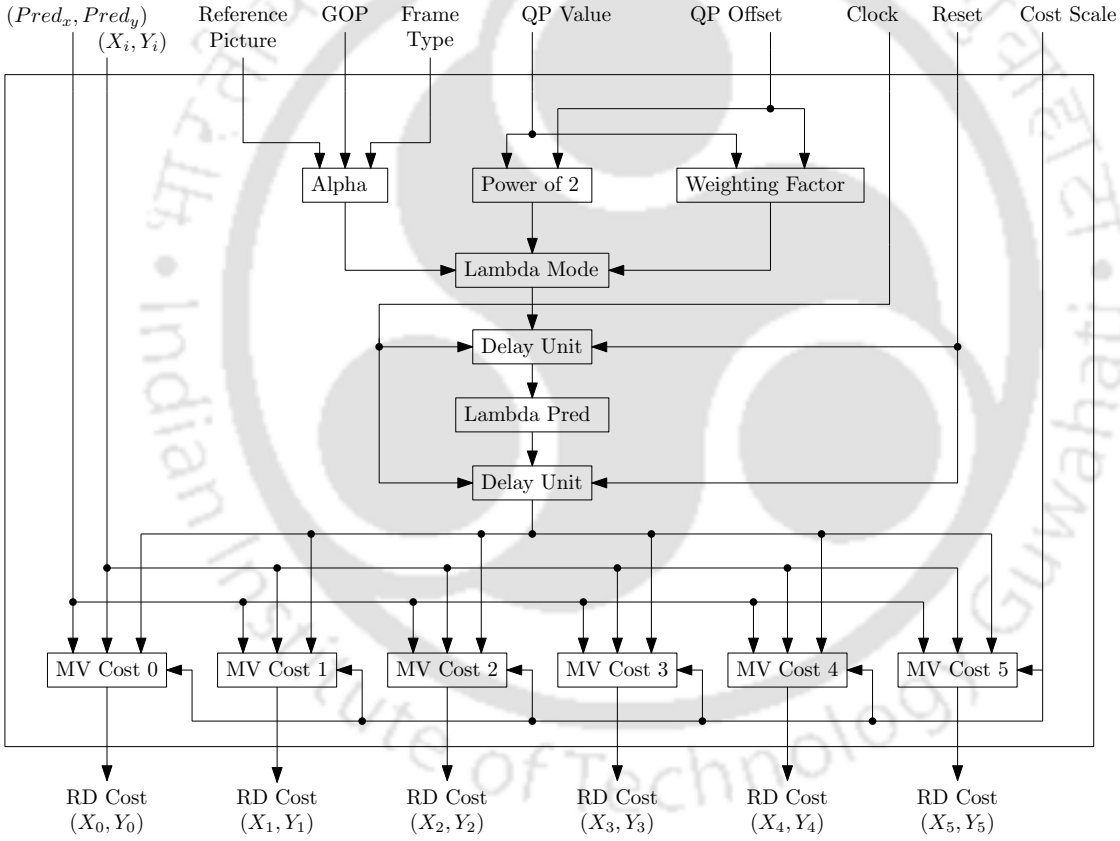


Figure 4.6: Rate Distortion Cost Unit

For storing the reference and current block pixels, we used two memories of size $64 \times 96 \times 8$ (6.144 KB) and $32 \times 32 \times 8 \times 2$ (2.048 KB) respectively. Reference memory contains 6, 32×32 pixel blocks and these blocks are partitioned into three segments as shown in Fig. 4.7. Each of the partition takes 256 clock cycles to write the data. It can be noted that for the first block,

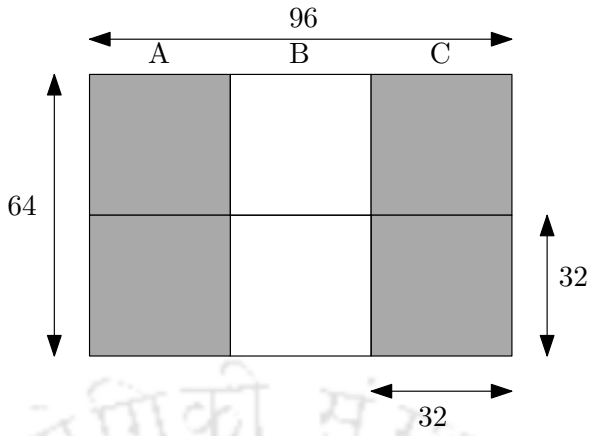


Figure 4.7: Reference Memory for IME

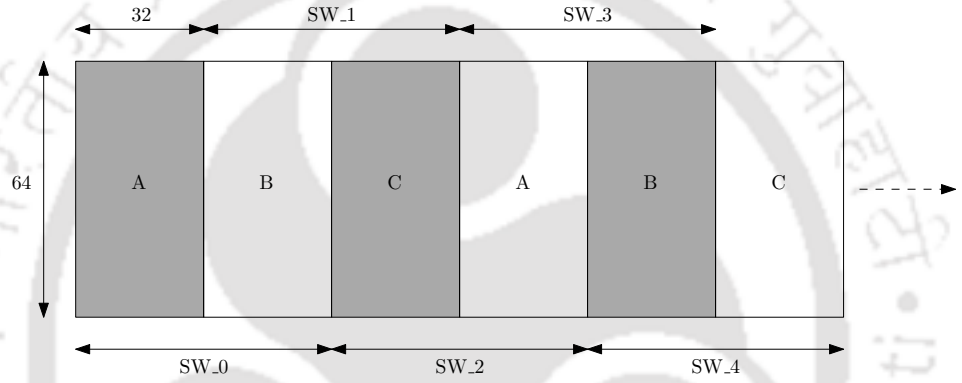


Figure 4.8: Flow of the Reference Memory

it takes 512 clock cycles. The flow of the reference memory is shown in Fig. 4.8. The first PU block search window SW_0 (formed by block A and B) is used for reference memory whereas block C is used to store the reference pixels of next PU. For the second PU, the reference memory which is formed by blocks B and C is used. A similar procedure is followed for the remaining PUs. The state diagram of the control unit for memory is given in Fig. 4.9. Here, RS, SP, and NS are reset state, starting position and next stage signals respectively. There are a total four stages: S_0, S_1, S_2 and S_3. The stage S_0 is the starting or the first stage. In this stage, data is stored in the reference memory part A and B as shown in Fig. 4.7. After 512 clock cycles, the second stage i.e., S_1 is activated. In this stage, data is stored in part C of reference memory and part A and B are used as search window for the first PU. When the S_1 stage output is 1, the control moves to stage S_2. In this stage, data is stored in memory part

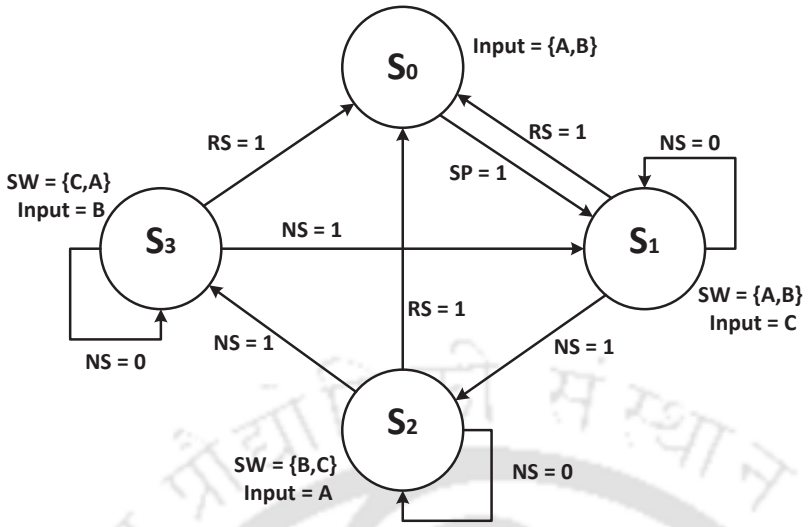


Figure 4.9: State Diagram of the Proposed Control Unit for Reference Memory

A and memory part B and C is used as a search window. The same procedure is repeated for the next stages.

For a parallel read/write operation, each part $(64 \times 32 \times 8)$ is divided into 64, 32×8 memory units. For each 32×8 memory unit, 5 : 32 decoder is required for address generation. Hence, a total of 64, 5 : 32 decoders are required for address generation of 64 memory units. The data read/write operation in all the three memory parts (i.e A, B and C) is controlled by an 8-bit counter, state of memory and final MV. The data is stored parallelly in 8, 32×8 memory units, which are addressed sequentially. The read operation mainly depends on the state of memory, size of the block and the distance from the origin/starting points. The pseudo code for memory read operation is presented in the Algorithm 1.

Using the above-mentioned architectures of the SAD unit, RD unit, memory, memory control unit, we developed the complete IME architecture for the proposed MHGS algorithm as is shown in Fig. 4.10. In the proposed architecture a total of six SAD units, each of size 32×32 are used. Each SAD unit is capable of calculating the SAD value of the 4×4 to 32×32 sub-blocks in two clock cycles. For reducing hardware complexity, we also used carry increment adder in the proposed architecture.

Input: Mem_State, Distance and Depth
Output: Chip selection and addresses for Memory
if *(Mem_State == 1)* **then**
  if *(Distance == 0 & Depth == 0)* **then**
    Read from mem_17 - mem_48 of parts A and B
    Address for part A is 17-32
    Address for part B is 1-16
  end
  else if *(Distance == 1 & Depth == 0)* **then**
    Read from mem_16 - mem_49 of parts A and B
    Address for part A is 16 for mem_17 - mem_48 and 17 - 32 for mem_16 & mem_49
    Address for part B is 17 for mem_17 - mem_48 and 1 - 16 for mem_16 & mem_49
  end
  else if *(Distance == 2 & Depth == 0)* **then**
    Read from mem_15 - mem_50 of parts A and B
    Address for mem_A is 15 for mem_17 mem_48, 16 for mem_16 & mem_49 and 16-32 for mem_15 & Mem_50
    Address for mem_B is 18 for mem_17 mem_48, 17 for mem_16 & mem_49 and 1-17 for mem_15 & mem_50
  end
  else if *(Distance == 4 & Depth == 0)* **then**
    Read from mem_13 - mem_52 of parts A and B
    Address for mem_A is 13-14 for mem_17 mem_48, 15 for mem_15, mem_16, mem_49 & mem_50 and 15-32 for mem_13, mem_14, mem_51 & mem_52
    Address for mem_B is 19-20 for mem_17 mem_48, 18 for mem_15, mem_16, mem_49 & mem_50 and 1-18 for mem_13, mem_14, mem_51 & mem_52
  end
  else if *(Distance == 8 & Depth == 0)* **then**
    Continue
  end
end
else if *(Mem_State == 2)* **then**
  Continue ;
end
else if *(Mem_State == 3)* **then**
  Continue ;
end
else
  Continue ;
end

Algorithm 1: Pseudo Code for Memory Read Operation

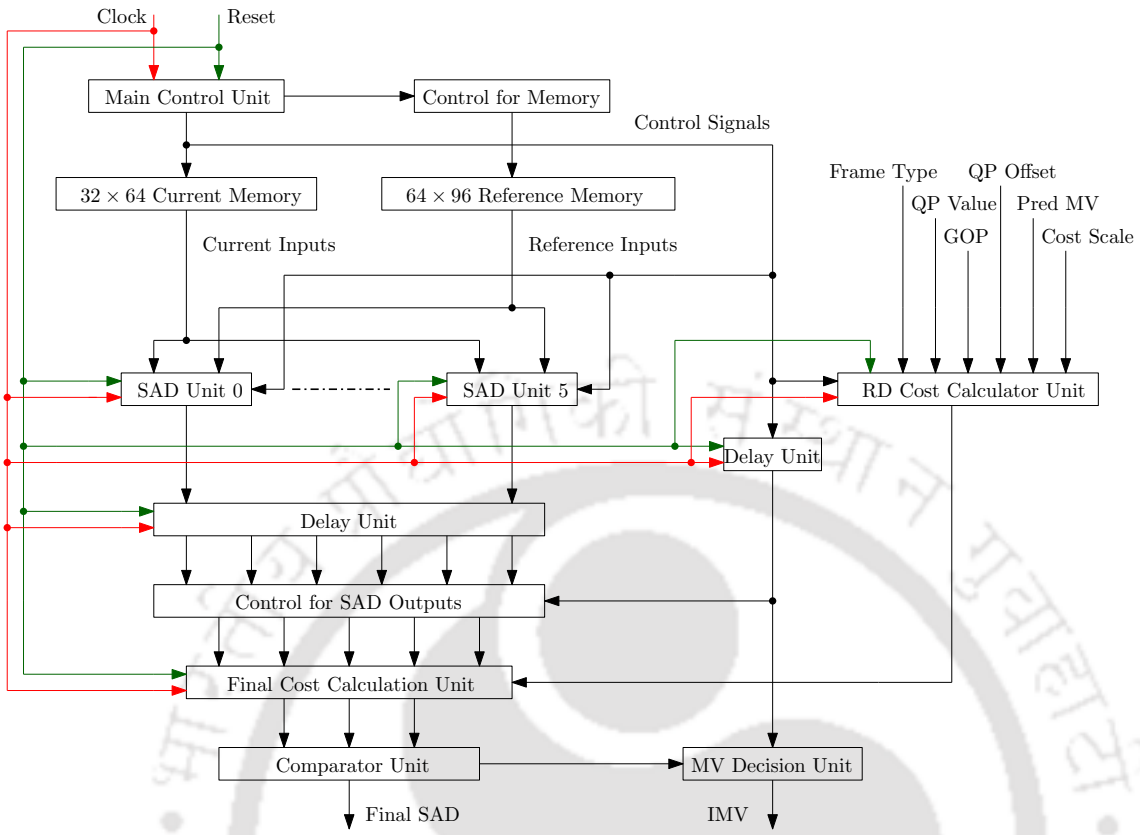


Figure 4.10: Block Diagram of the Proposed Integer Motion Estimation Architecture

4.4 Simulation Results

In order to validate the proposed algorithm, we considered the reference software HM in default configuration i.e., the TZS algorithm is used for IME search with a search range of ±64 and CU size is 64. The proposed MHGS ME algorithm is integrated into reference software HM and simulated with the *encoder_lowdelay_P_main* and *encoder_randomaccess_main* profile using different QP values of 22, 27, 32 and 37. The values of BD-PSNR, BD-bit rate, and ASP are presented in Table 4.1. In Table 4.1, MTZS is the TZS without left side MVP blocks and MHGS is the HGS without left side MVP blocks. It is clear from Table 4.1 that the proposed MHGS ME algorithm requires an average of 53.82% fewer search points in comparison to reference software HM with a little degradation in BD-PSNR and an average increment of 2% in BD-bit rate. In videos like crowd_run 1080p50fps, the saving in the number of search points is more than 73% with around 1.3% increment in bit-rate and 0.22 dB degradation in PSNR.

Table 4.1: BD-PSNR, BD-Bitrate and ASP Comparison.

Video Sequences	Algorithm	Profile	BD-Bitrate	BD-PSNR	△ASP (%)
BQMall (832x480_60fps)	MTZS	Low Delay	0.2322	-0.0202	2.6661
		Random Access	0.5609	-0.0242	3.2601
	MHGS	Low Delay	2.3662	-0.0551	-58.1684
		Random Access	2.5432	-0.1383	-63.9470
Johnny (1280x720_60)	MTZS	Low Delay	0.1013	-0.0084	1.1076
		Random Access	0.1559	-0.0423	1.5208
	MHGS	Low Delay	1.3743	-0.0013	-32.4039
		Random Access	1.8491	-0.0326	-35.0114
crowd_run (1080p50fps)	MTZS	Low Delay	0.3616	-0.0013	12.7612
		Random Access	0.5160	-0.0347	13.7067
	MHGS	Low Delay	1.1448	-0.0211	-72.5020
		Random Access	1.3327	-0.2250	-73.3427
Traffic (2560x1600_30fps)	MTZS	Low Delay	0.3011	-0.0034	4.1284
		Random Access	0.3886	-0.0092	4.8907
	MHGS	Low Delay	2.3254	-0.0248	-47.0048
		Random Access	2.5256	-0.0410	-48.1423

Therefore, it can be concluded that for the videos like crowd_run 1080p @ 50fps, around 3/4th computational power of ME process is saved. Moreover, it is evident from Table 4.1 that the proposed algorithm is suitable for all types of video sequences.

The RD results for video sequences $BasketballPass\_416\times240@50$ and $KristenAndSara\_1280\times720@60$ using *encoder_lowdelay_P_main* profile under different QP values of 22, 27, 32 and 37 is shown in Fig. 4.11. The loss in BD-PSNR [113] loss as compared to reference software HM in $BasketballPass\_416\times240@50$ and $KristenAndSara\_1280\times720@60$ video sequences are 0.0621 dB and 0.0610 dB respectively. Fig. 4.12 shows the results for *encoder_randomaccess_main* profile for QP values of 22, 27, 32 and 37. The loss in BD-PSNR for *encoder_randomaccess_main* profile using the $BasketballPass\_416\times240@50$ and $KristenAndSara\_1280\times720@60$ video sequences are 0.0646 dB and 0.1040 dB respectively. It is clear from Fig. 4.11 and 4.12 that, as the target bit rate increases, the loss in the BD-PSNR decreases. Hence, it can be concluded that the proposed HGS IME algorithm requires a fewer number of search points with negligi-

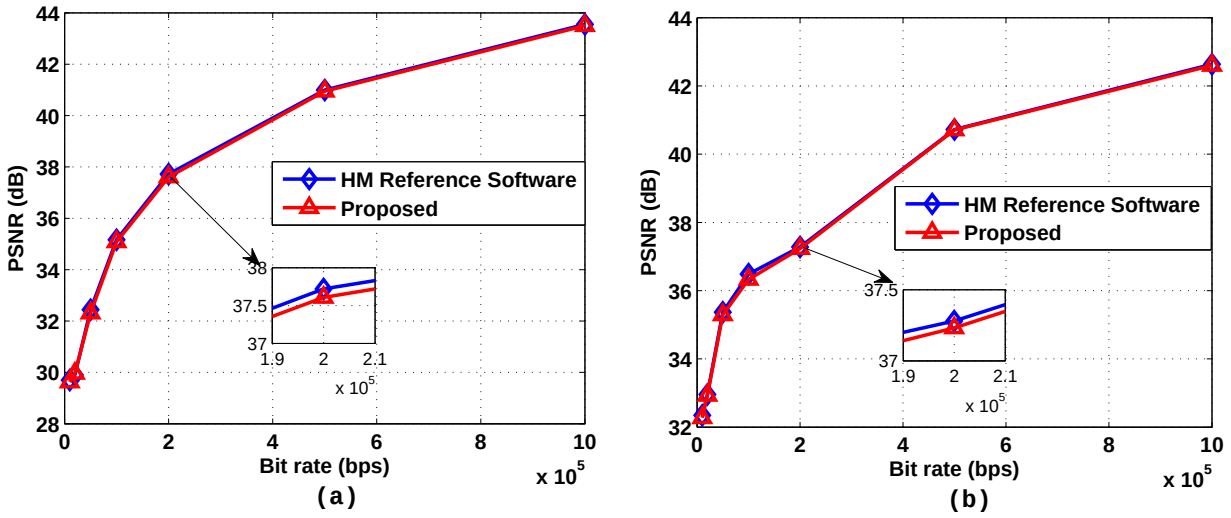


Figure 4.11: Rate-distortion results for video sequence (a) $BasketballPass\_416 \times 240@50$ and (b) $KristenAndSara\_1280 \times 720@60$ using *encoder_lowdelay_P_main* profile under different QP value of 22, 27, 32 and 37.

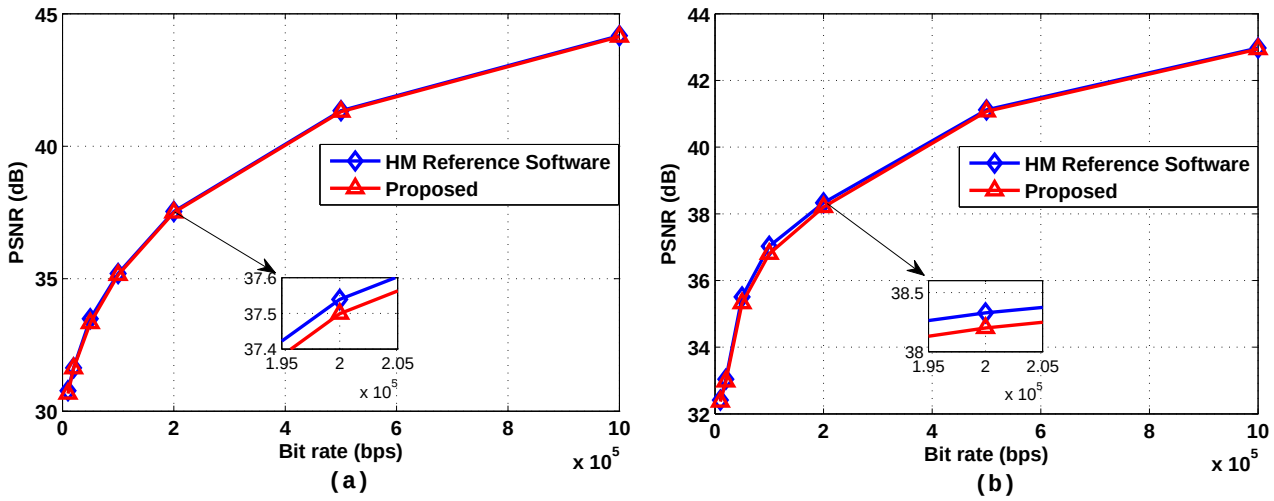


Figure 4.12: Rate-distortion results for video sequence (a) $BasketballPass\_416 \times 240@50$ and (b) $KristenAndSara\_1280 \times 720@60$ using *encoder_randomaccess_main* profile under different QP value of 22, 27, 32 and 37.

ble degradation in BD-PSNR for *encoder_lowdelay_P_main* and *encoder_randomaccess_main* profiles.

Simulations have also been carried out using Synopsys Design Compiler and Cadence Innovus software, to obtain the performance of the proposed architecture. The proposed architecture is implemented using standard 90 nm technology and the results are presented in Table 4.2. These results are compared with other state of the art architectures of ME. As compared to other state of the art ME architectures, the proposed architecture has a small area and low power. Simulation results show that the proposed architecture is able to process 3840×2160 @ 30 fps video sequences using 8.192 Kilo-bite SRAM and the corresponding circuitry occupies an area of 4.5193 mm^2. The operating frequency of the proposed architecture is obtained as 250 MHz with 151.7619 mW of power. The layout of the proposed architecture is presented in Fig. 4.13.

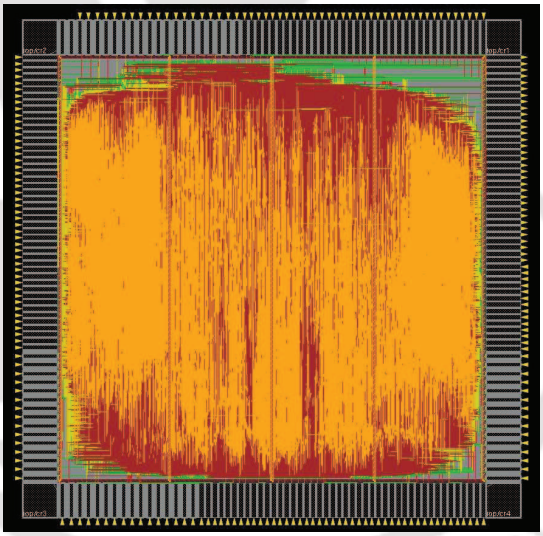


Figure 4.13: Layout of the Proposed IME Architecture

4.5 Conclusion

In this chapter, we proposed a modification to the algorithm presented in the Chapter 3 and its corresponding low-power architecture which is suitable to be employed in portable devices. For reducing the hardware complexity, we used pixel truncation, sub-sampling, data reuse,

Table 4.2: Comparison of the proposed architecture with state of the art architectures

	ME2014 [81]	ME2013 [116]	MEHEVC13 [119]	MEHEVC15 [17]	Proposed
Technology	40 nm CMOS	TMSC 28 nm	65 nm Low-Power CMOS	90 nm	90 nm
Standard	H.264/AVC	HEVC	HEVC	HEVC	HEVC
Resolution	7680 × 4320 @ 48 fps 3840 × 2160 @ 120 fps	8192 × 4320 @ 30 fps	3840 × 2160 @ 30 fps	4096 × 2048 @ 60 fps	3840 × 2160 @ 30 fps
Search Range	((±211, ±106) for P (±107, ±56) for B)	(±512, ±128)	(±64, ±64)	(±64, ±64)	Adaptive
Algorithm	FS + 5T12S	Inter + intra	FS + TSS	Modified TZS	Modified HGS
Memory	552 KB	7.14 MB	0.68 MB	17.4 KB	8.192 KB
Throughput	1.59 Gpixel/s	1062 Mpixels/s	248.8 Mpixel/s	503 Mpixel/s	248.8 Mpixel/s
Block Size	16 × 16 - 8 × 8	64 × 64 - 16 × 16	64 × 64 to 8 × 8	64 × 64 to 8 × 8	32 × 32 to 8 × 8
Area	15.52 mm^2	25 mm^2	3965 K gates	778.7 K gates	4.5193 mm^2/1441 K gates
Power	622 mW	708 mW	NA	NA	151.7619 mW
Frequency	210 MHz	312 MHz	200 MHz	270 MHz	250 MHz
Sub-sampling	4 × 4 : 1	NA	Not Used	Not Used	2 × 2 : 1
Pixel Truncation	6-bit/pixel	NA	NA	NA	4-bit/pixel
Cost Function	Not Used	NA	Not Used	Not Used	Yes

adaptive search area, removing some of the PU sizes and fewer search points ME algorithm. In the proposed algorithm, left side blocks are not used in MVP due to the pipelined structure. Simulation results show that the proposed MHGS ME algorithm requires 53.82% fewer search points as compared to the HM reference software based realization with a little degradation in PSNR and marginal increment in bit rate. The proposed architecture is simulated and synthesized using standard 90 nm technology. The proposed IME architecture can process 3840×2160 @ 30 fps video sequences using 8.192 KB SRAM and the corresponding circuitry occupies an area of 4.5193 mm^2. The ME architecture consumes 151.7619 mW of power when operates at 250 MHz.

5

Fast Fractional Motion Estimation Algorithm and Architecture for HEVC

Contents

Like other video coding standards, HEVC also uses inter prediction to remove the temporal redundancy. ME is the important block of inter prediction, which involves complex and time-consuming operations. In all the latest video compression standards, ME searches up to quarter-pixel. FME is used to improve the prediction quality at cost of the increase in the computational complexity, which is due to additional operations such as interpolation and Hadamard transform (HT). For low-power implementation, it is necessary to reduce the computational complexity at the FME level. Based on the statistical study of the final MV, we have proposed a new fast FME algorithm and architecture for HEVC.

The rest of the chapter is organized as follows. The chapter is divided into two parts. The first part describes a computationally efficient FME algorithm. The second part of the chapter focuses on the hardware implementation of the proposed FME algorithm. In the next section, simulation results of the proposed algorithm and architecture are presented. The conclusion of the proposed work is summarized at the end of the chapter.

5.1 Introduction

Portable devices are gaining popularity in the form of smartphones, smartwatches, tablets and camcorders. These devices are operated by a fixed and limited power supply. Due to power constraints, it is extremely challenging to perform video coding, especially for state-of-the-art video coding standards such as HEVC. With more and more growth in the computational speed of digital computing devices, there is a great demand for digital information including data, audio and video. Processing of digital video information is a more complicated and challenging task due to the huge amount of data involved. Hence, there is endless research actively going on over the past three decades to compress the digital video. The main drawback of such innovations is that they require huge computations to be performed, which directly translates into high-power consumption. For these platforms, a computationally efficient algorithm and low-power hardware design is necessary to make better use of battery resources.

In all the latest video coding standards, video compression is done by removing the temporal, spatial and statics redundancy. Among all the three redundancies, temporal redundancy is more

important because it compresses the video data many folds as compared to other techniques. However, it is the most computationally complex and power consuming blocks of the video compression standards. The temporal redundancy is removed with the help of inter prediction operation. ME is used in inter prediction for finding the best-matched block in the search window of the reference frame. It is a two-step process, namely, IME and FME. As discussed in Chapter 3, in IME, the search is performed at an integer pixel position and the best suitable block is found.

Like H.264/AVC, in HEVC also FME is executed in two steps, as shown in Fig. 5.1. The first step searches at eight half-pixel positions around the best integer MV and chooses the particular block with the lowest RD cost as the best-matched block. The second step searches

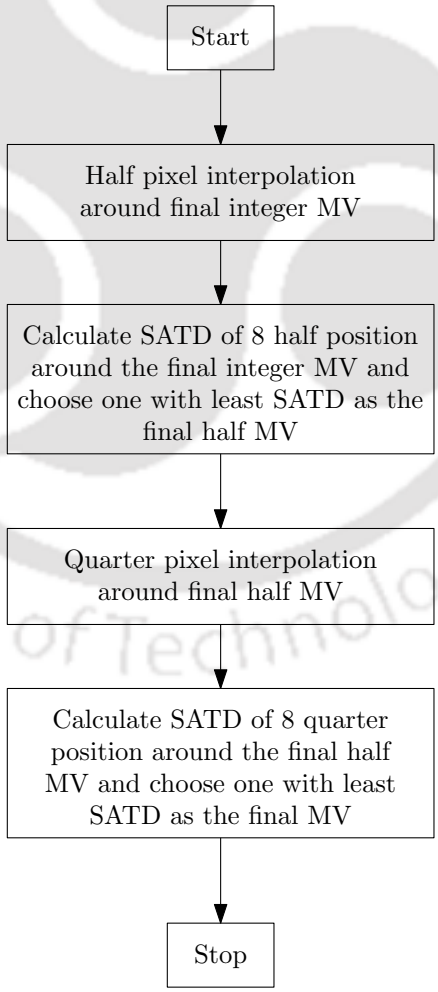


Figure 5.1: Basic Flow of FME in HEVC

at eight quarter-pixel positions around the best half MV and chooses a block with the lowest RD cost as the best-matched block. In reference frame pixel values are available at integer position. For finding the value of the pixel at the half and quarter positions, 8-tap and 7-tap DCT based interpolation filters are used respectively for luminance. For chrominance sample, 4-tap DCT based interpolation filter is used. In HEVC, first, all horizontal fraction position pixel values are calculated using only horizontal integer pixel values. In the calculation of vertical fractional pixel values, integer, as well as horizontal fractional pixel values, are used. The coefficient of the DCT based interpolation filter for luminance and chrominance are given in Table 2.9 and Table 2.10 respectively. The filter coefficients in the luminance sample for 1/4th and 3/4th position is the same.

The RD cost ($J_{pred,SAD}$) of IME is calculated by using (5.1), where λ_{pred} is the Lagrangian function, used to trade off between bit-rate and distortion and B_{pred} represents the number of bits required for prediction information. The SAD is calculated using (2.5) as presented in Chapter 2.

$$J_{pred,SAD} = SAD + \lambda_{pred} \times B_{pred} \tag{5.1}$$

In FME, SATD is used for calculation of RD ($J_{pred,SATD}$) as given by

$$J_{pred,SATD} = SATD + \lambda_{pred} \times B_{pred} \tag{5.2}$$

In order to compute SATD, firstly, the distortion matrix is computed using the following relation

$$X(i,j) = (BlockA(i,j) - BlockB(i,j)) \tag{5.3}$$

where, $BlockA$ and $BlockB$ are the current and reference blocks respectively.

Now, using distortion matrix, transformed matrix is formulated as per the following expression

$$Y(i,j) = H_m X(i,j) H_m^T \tag{5.4}$$

where H_m is a $2^m \times 2^m$ HT matrix. HT is a generalized class of Fourier transforms, which transforms 2^m real numbers x_n into 2^m real numbers X_k.

The HT can be defined in two ways: recursively or by using the binary representation of the indices n and k. In order to recursively represent HT using formula, the 1×1 HT matrix i.e. H_0 is defined as $H_0 = 1$, and for $m > 0$ HT matrices are defined as

$$H_m = \frac{1}{\sqrt{2}} \begin{pmatrix} H_{m-1} & H_{m-1} \\ H_{m-1} & -H_{m-1} \end{pmatrix} \tag{5.5}$$

where, $\frac{1}{\sqrt{2}}$ is a normalization factor, which can be sometimes be omitted. For $m > 1$, we can also define H_m as

$$H_m = H_1 \otimes H_{m-1} \tag{5.6}$$

where $\otimes$ represents the Kronecker product.

Similarly, we can also define the HT matrix by using the binary representation of the indices n and k as follows:

$$\left(H_m\right)_{n,k} = \frac{1}{2^{\frac{m}{2}}} \left(-1\right)^{\sum_j k_j n_j} \tag{5.7}$$

where the k_j and n_j are the binary digits (0 or 1) of k and n, respectively. The expression for H_2 and H_3 after omitting normalization factor can be obtained as

$$H_2 = H_{4\times 4} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \tag{5.8}$$

$$H_3 = H_{8\times8} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix}$$

$$= \begin{pmatrix} H_{4\times4} & H_{4\times4} \\ H_{4\times4} & -H_{4\times4} \end{pmatrix} \tag{5.9}$$

Finally, SATD can be computed as follows

$$SATD = (\sum_{i,j} |Y(i,j)|)/2 \tag{5.10}$$

Unlike SAD computation in IME, the SATD computation in FME involves two additional operations such as interpolation and HT which increases the complexity. In HEVC, the possible sizes of HT are 2×2, 4×4, 8×8, 16×4 and 4×16. Therefore, reducing the computational complexity at the FME level is a challenging task. Most of the works available in the literature were mainly targeted to develop fast IME algorithms [1–3,17,24–34,40,44,59,80,81,104,110,120–124]. The works presented in [125,126] were mainly focussed on the reduction of computational complexity of FME employed in H.264/AVC. Reduction in computational complexity of FME algorithms in compliance to HEVC is discussed in [78, 127].

In [125] Lin et al., proposed a single iteration FME algorithm for H.264/AVC. It searches only six points at fractional level i.e. (0,0), the PMV, and four neighbouring points. A single-iteration full search FME algorithm for H.264/AVC is presented in [126]. In this algorithm, the bilinear filter rather than the 6-tap filter is used for half-pixel interpolation. In [128], Donggil et al., proposed a VLSI architecture for interpolation filter. This architecture is implemented in

Table 5.1: Video sequences used in simulation

Video Sequences	Abbreviation Used
BasketballPass_416x240_50fps	BBPass
BQMall_832x480_60fps	BQMall
BasketballDrill_832x480_60fps	BBDrill
Johnny_1280x720_60fps	Johnny
KristenAndSara_1280x720_60fps	KASara
Crowd_run1080p_50fps	CRun
old_town_cross_1080p_50fps	OTown
Traffic_2560x1600_30fps	Traffic
HoneyBee_3840x2160_120fps	HaneyBee

130-nm technology and operated at 208 MHz to achieve the throughput of 296 cycles/32×32. In [127], an algorithm and its corresponding architecture, based on five transforms and twelve search points (5T12S) are proposed. The twelve search points are selected around the second and third positions with respect to the least distortion position. The architecture has been implemented using 65-nm technology and the measured throughput reaches 995 Mpixels/s for 7680×4320 @ 30 frames/s at 188 MHz, which is at least 4.7 times faster than the prior works. The corresponding power dissipation is 198.6 mW with a power-efficiency of 0.2 nJ/pixel. A fast FME algorithm that reduces computational complexity by reducing the number of search points is proposed in [78]. Based on the statistical observations in this work, only four search points are selected in the horizontal and vertical directions for half-pel search. On the other hand, three search points that have the smaller absolute value of the MV difference are selected from eight candidates for quarter-pel search.

Motivated by this, we have performed experiments using different types of video sequences given in Table 5.1 and observed the behaviour of final MVs. The observed MV distribution at the fractional pixel position is shown in Fig. 5.2. It is clear from Fig. 5.2 that the average number of final MVs obtained at the integer pixel position is 36.59%. The average final MVs found at $(\pm 1, \pm 1)$ position around the integer MVs is 68.53%. Overall around 75% of final MVs lies at (0,0), vertical and horizontal positions. Therefore, instead of checking all the fractional points, one can search fewer points which are close to the integer MV i.e., (0,0) and in horizontal

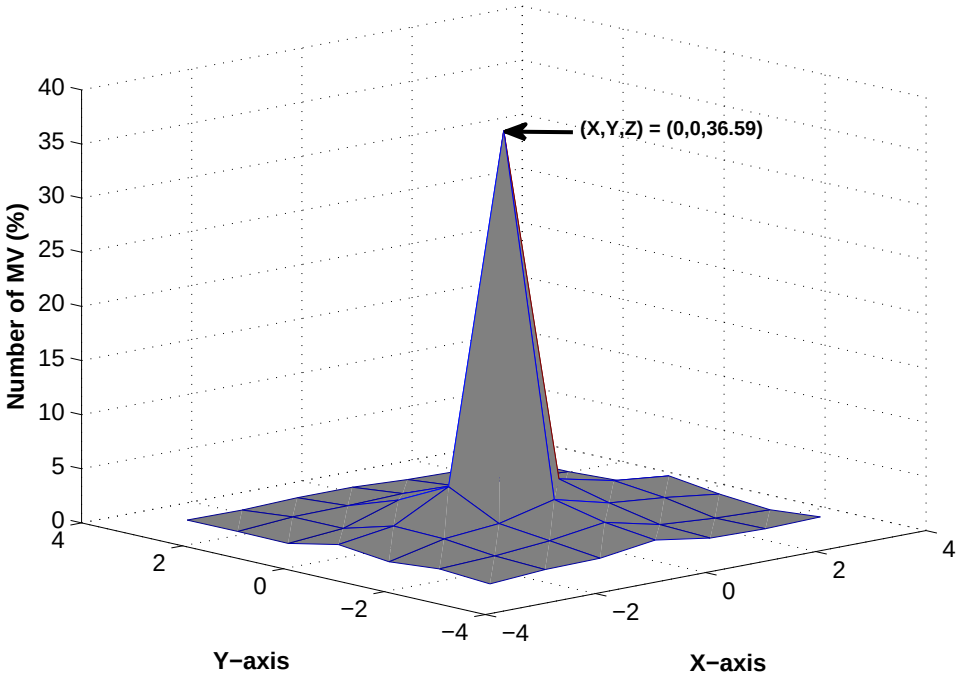


Figure 5.2: Motion vector distribution for the video sequences given in Table 5.1.

and vertical positions. Based on these observations, we developed a fast FME algorithm. The details of the proposed fast FME algorithm is presented in the following section.

5.2 Proposed FME Algorithm

Motivated by the statistics of the final MVs, we proposed three search patterns as shown in Fig. 5.3 - Fig. 5.5. Using the search pattern given in the Fig. 5.3 i.e., five search points at the half-pixel position and five search points at the quarter-pixel position, the average number of final MVs found is 73.92% and we labelled this algorithm as H5Q5. Using the search pattern given in Fig. 5.4 i.e., five search points at the half-pixel position and 5 and/or 9 search points at the quarter-pixel position depending upon the position of best-match at the half-pixel, the average number of final MVs was found to be 83.58% and is labelled as H5Q9&5. We have also tested for five search points at the half and eight search points at quarter-pel positions and correspondingly named this algorithm as the H5Q9 as shown in Fig. 5.5. Using H5Q9 search

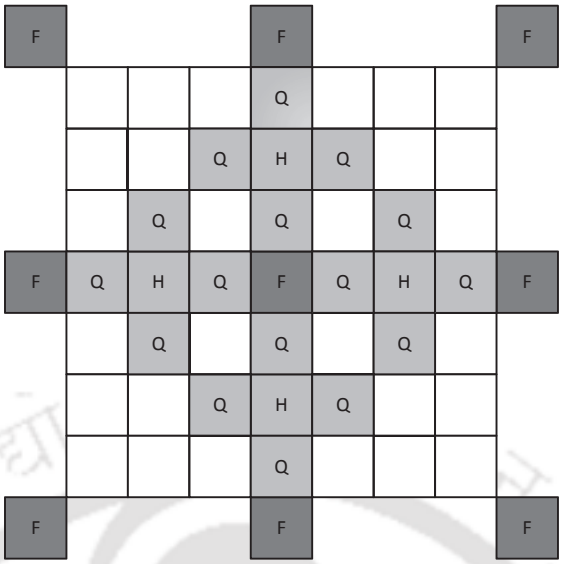


Figure 5.3: Illustrations of the H5Q5 algorithm.

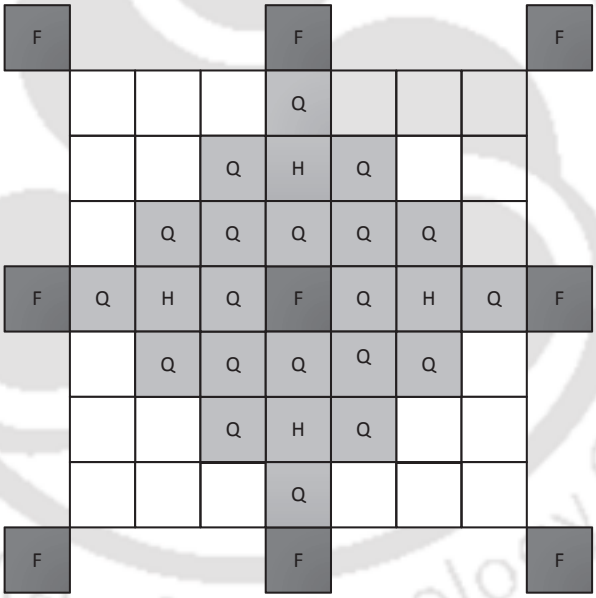


Figure 5.4: Illustrations of the H5Q9&5 algorithm.

pattern, the total number of final MVs was found to be 89.23%. Computation of HT, SAD and interpolation filter (8-tap and 7-tap) is required at every search point. Hence, reducing the number of search points directly reduces the required HT and interpolation operations. Therefore, using the proposed search pattern computational complexity can be reduced greatly.

The flow of the proposed fast FME algorithm is given in Fig. 5.6. In the first step i.e., at

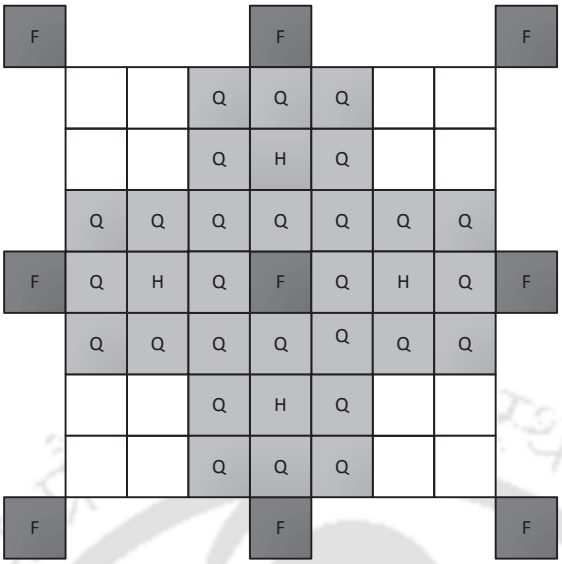


Figure 5.5: Illustrations of the H5Q9 algorithm.

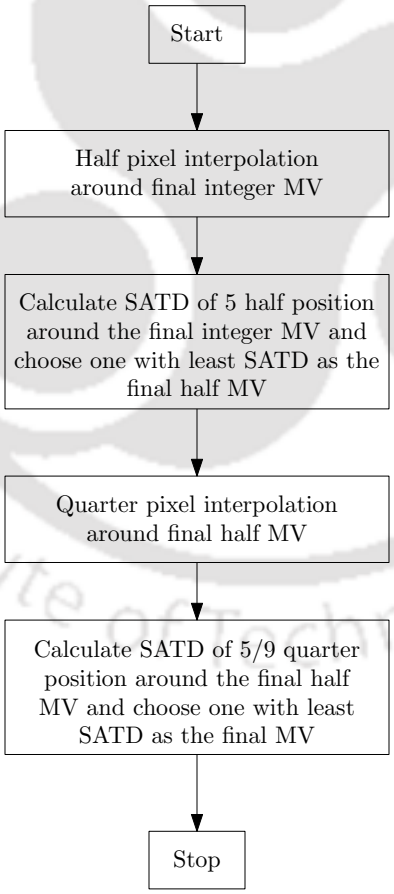


Figure 5.6: Flow of the Proposed FME algorithms.

the half-pixel position, only five search points each along horizontal and vertical positions are checked and selects the one with the least RD cost as the best match. In the second step, i.e., at the quarter-pixel position, either five or nine search points are checked depending upon the search pattern. In H5Q5, five search points are checked at a quarter-pixel position. In H5Q9&5, either five or nine search points are checked depending upon the position of the best match at the half-pixel position. If the best match is found at the center of the half-pixel search, then nine points are checked at quarter-pixel search, otherwise only five search points are checked. In H5Q9, nine search points are checked at a quarter-pixel position.

5.3 Simulation Results for Proposed Algorithm

Table 5.2: BD Bit-rate comparison with State of art algorithms

Video Sequences	HM**	Proposed_H5Q9*	Proposed_H5Q9&5*	Proposed_H5Q5*
BBPass	186.8245	1.4288	1.5381	2.4076
BQMall	892.8200	0.3069	0.4525	0.8109
BBDrill	831.2400	0.2382	0.9456	0.9720
Johnny	202.2063	-0.3748	-0.1125	-1.4993
KASara	277.7179	-0.4821	-1.0279	-0.4730
OTown	1410.5143	0.7596	0.5712	1.2741
CRun	15907.400	0.2678	0.5060	0.7924
Traffic	2908.5714	0.3271	0.8782	1.5255
Average change	–	0.3089	0.4689	0.7262

⋆: Change in % as compared HM reference software ⋆⋆: Original HM reference software

All the proposed algorithms are implemented using HM reference software on Intel(R) Core i5-2400 @ 3.10 GHz (8GB RAM) device for the profile *encoder_lowdelay_P_main* and *encoder_randomaccess_main* with different QP values of 22, 27, 32 and 37. For checking the versatility of the proposed algorithm, we carried out experiments on different types of video sequences whose resolution varies from 416×240 to 2560×1600 and the results are listed in Table 5.2 and 5.3. It is clear from Table 5.2 that when compared to the HM reference software, the average change in BD bit-rate for the proposed algorithms H5Q9, H5Q5&9 and H5Q5 are 0.3089%, 0.4689% and 0.7262% respectively. From 5.3 it can be observed that compared to HM reference, the average decrement in BD-PSNR is 0.0071 dB, 0.0155 dB and 0.0256 dB for the proposed algorithms H5Q9, H5Q5&9 and H5Q5 respectively. As compared to [78], the pro-

posed H5Q5 algorithm requires slightly more number of search points. However, the BD-rate of the proposed algorithm is 0.7262% as compared to the 0.93% as reported in [78]. Moreover, the effect on PSNR and the process of selection of diagonal motion after half-pixel ME is not mentioned in [78].

Table 5.3: BD-PSNR comparison with HM reference software algorithm

Video Sequences	HM**	Proposed_H5Q9*	Proposed_H5Q9&5*	Proposed_H5Q5*
BBPass	34.2428	-0.0012	-0.0193	-0.0191
BQMall	33.8031	-0.0026	-0.0226	-0.0228
BBDrill	34.4691	-0.0062	-0.0169	-0.0138
Johnny	38.9985	-0.0025	-0.0057	-0.0301
KASara	38.8014	-0.0120	-0.0130	-0.0204
OTown	33.9888	-0.0065	-0.0108	-0.0257
CRun	30.9712	-0.0191	-0.0186	-0.0365
Traffic	36.0129	-0.0065	-0.0174	-0.0362
Average change	–	-0.0071	-0.0155	-0.0256

⋆: PSNR difference ⋆⋆: Original HM reference software

The average number of search points required using proposed FME algorithms and the HM reference software are listed in Table 5.4. It is clear from Table 5.4 that the proposed algorithms H5Q9&5 and H5Q5 require respectively 32.00% and 47.06% fewer search points as compared to HM reference software realization. Furthermore, H5Q9&5 and H5Q5 algorithms require 2.59% and 17.65% lesser number of search points as compared to [127]. The algorithm in [126] is a single iteration algorithm, which completes the search using 47.06% more search points as compared to HM reference software.

Table 5.4: Average Search Points comparison with HM reference software algorithm

Algorithms	Effective # of Search Points	ASPS* (%)
HM10.0	1I + 8H + 8Q	-
FME2009 [126]	1I + 8H + 16Q	47.06
FME2015 [127]	1I + 4H + 7Q	-29.41
Proposed_H5Q9	1I + 4H + 8Q	-23.53
Proposed_H5Q9&5	1I + 4H + 8/4Q	-32.00
Proposed_H5Q5	1I + 4H + 4Q	-47.06

⋆: Average Number of Search Points Saved

The RD results of proposed H5Q5 algorithm for video sequences *BasketballPass*_416 × 240@50 and *Johnny*_1280 × 720@60 using the profile *encoder_lowdelay_P_main*, with different

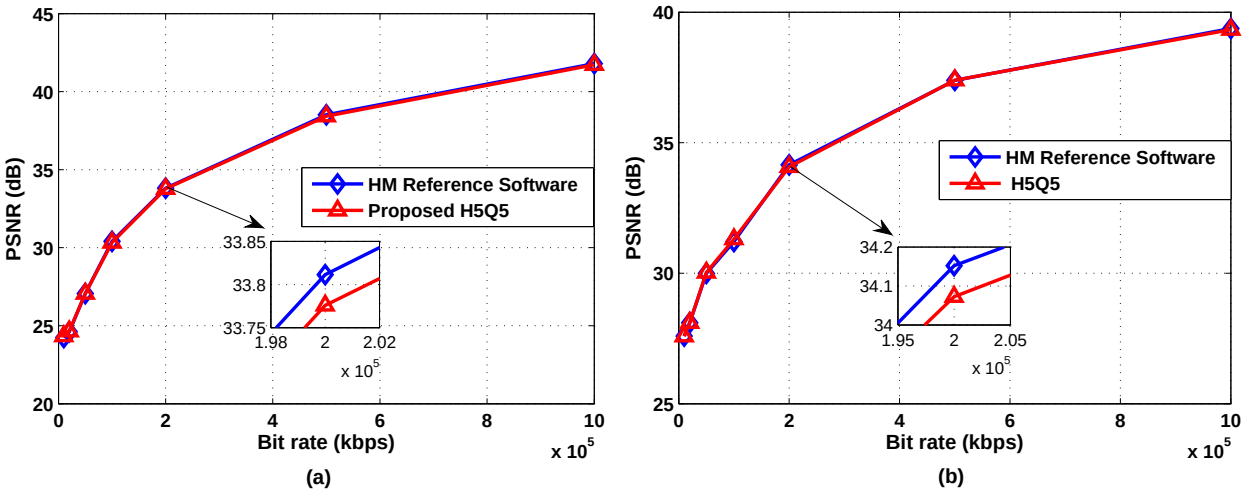


Figure 5.7: Rate-distortion results for video sequence (a) *BasketballPass*_416 × 240@50 and (b) *Johnny*_1280 × 720@60 using *encoder_lowdelay_P_main* profile under different QP value of 22, 27, 32 and 37.

QP values of 22, 27, 32 and 37 is shown in Fig. 5.7. The BD-PSNR loss as compared to HM reference software-based simulation for *BasketballPass*_416 × 240@50 and *Johnny*_1280 × 720@60 video sequences are 0.0207 dB and 0.0048 dB respectively. It is clear from Fig. 5.7 that as the target bit rate increases, the loss in the BD-PSNR decreases. We also carried out simulations based on the proposed H5Q5 algorithm using the *encoder_randomaccess_main* profile for QP values of 22, 27, 32 and 37. Using this profile, the BD-PSNR decrement for the proposed H5Q5 algorithm for the video sequences *BasketballPass*_416 × 240@50 and *Johnny*_1280 × 720@60 are 0.0372 dB and 0.0147 dB respectively. The rate distortion results of proposed H5Q5 algorithm for video sequences *BasketballPass*_416 × 240@50 and *Johnny*_1280 × 720@60 using the *encoder_randomaccess_main* profile with different QP values of 22, 27, 32 and 37 is shown in Fig. 5.8. The BD-PSNR loss for the H5Q5&9 and H5Q9 algorithms are 0.0167 dB and 0.0016 dB respectively for *BasketballPass*_416 × 240@50 video sequence using *encoder_lowdelay_P_main* profile for QP values of 22, 27, 32 and 37.

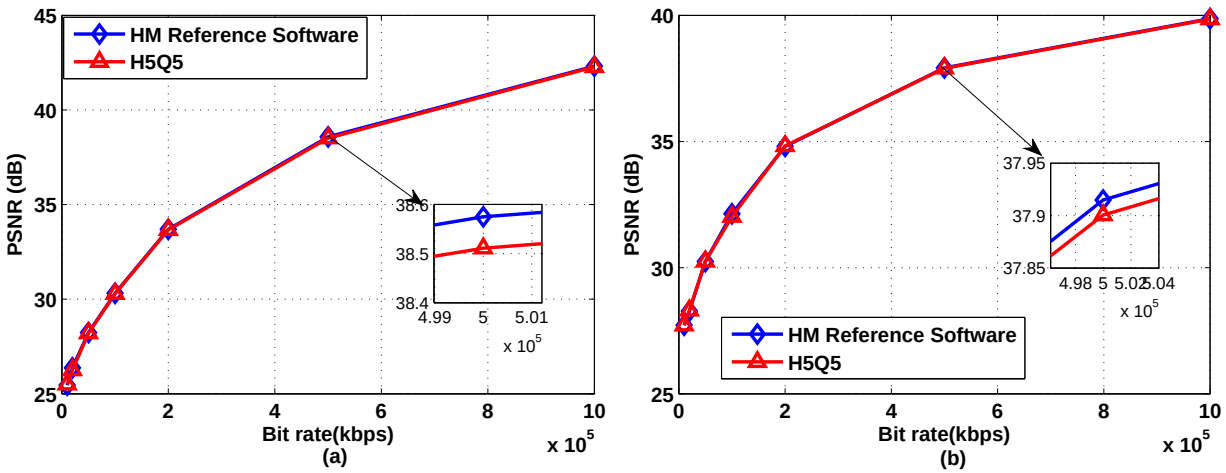


Figure 5.8: Rate-distortion results for video sequence (a) *BasketballPass*_416 × 240@50 and (b) *Johnny*_1280 × 720@60 using *encoder_randomaccess_main* profile under different QP value of 22, 27, 32 and 37.

5.4 Proposed FME Architecture

It is clear from Table 5.4 and Fig. 5.7 that, among the three proposed algorithms, H5Q5 is more efficient in terms of the number of search points with a little degradation in PSNR as compared to the HM reference software. Hence, we developed a low-power architecture for H5Q5. In the FME process, the pixel value at the fractional position is required. In order to calculate the pixel value at fractional pixel positions, DCT based interpolation filters are used. For luminance samples, 8-tap and 7-tap filters are used for the half and quarter pixel positions. The proposed architecture for a two-stage half and quarter interpolation filters are shown in Fig. 5.9 and Fig. 5.10 respectively. For the storage of calculated fractional pixel values, we need storage devices. In the proposed architecture, we used three different memory segments: one segment each to store the current block, integer reference pixels and the interpolated pixel values. The size of the current memory, integer reference memory and fractional pixel memories are 2.048 KB, 7.844 KB and 18.672 KB respectively.

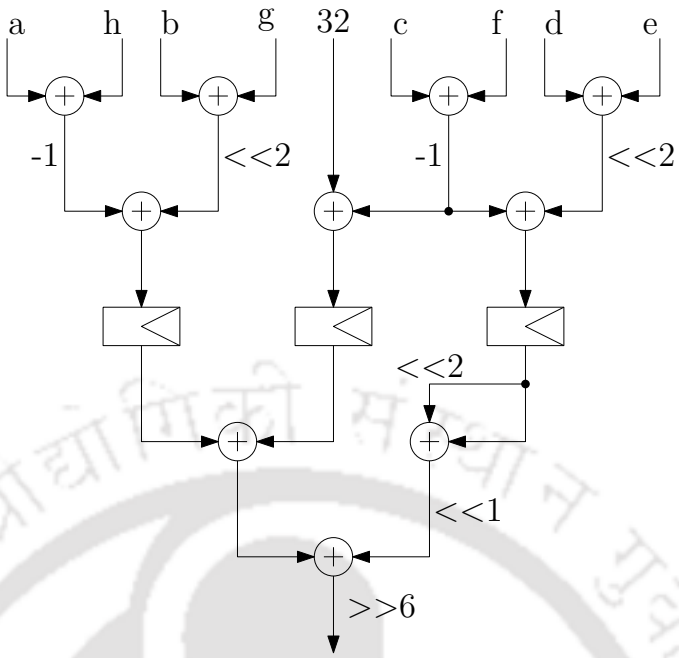


Figure 5.9: Proposed Two Stage Architecture for Half Filter.

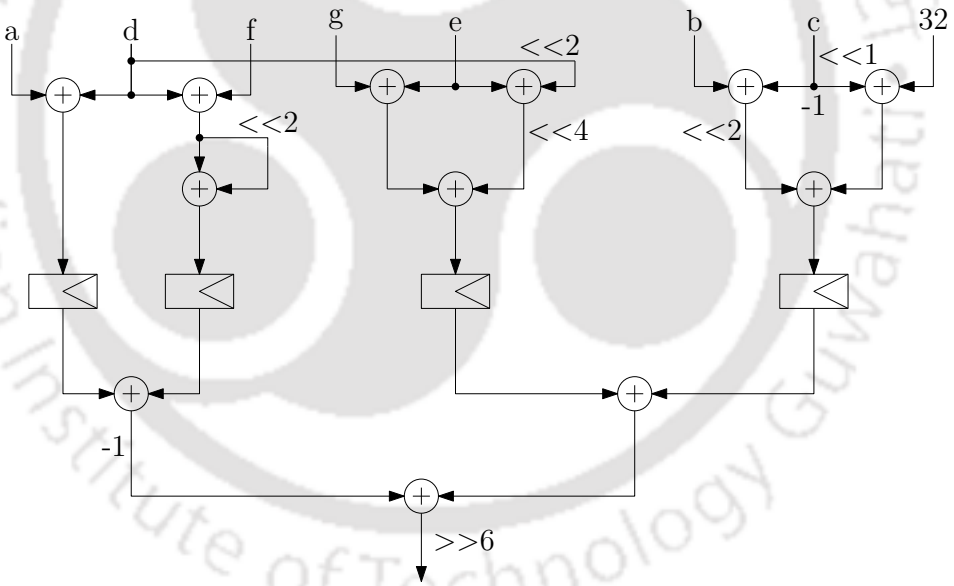


Figure 5.10: Proposed Two Stage Architecture for Quarter Filter.

As per the expression (5.2), we need to calculate the values of SATD, λ and B_{pred}. In the next few paragraphs, we focus on the development of the architectures of SATD, λ and B_{pred} one by one. First, we focus on the SATD. As the computation of SATD involves HT described by (5.5) to (5.9), there is a need for developing a low-power architecture for HT. After a thorough study of (5.5) to (5.9), it was observed that, an efficient architecture can

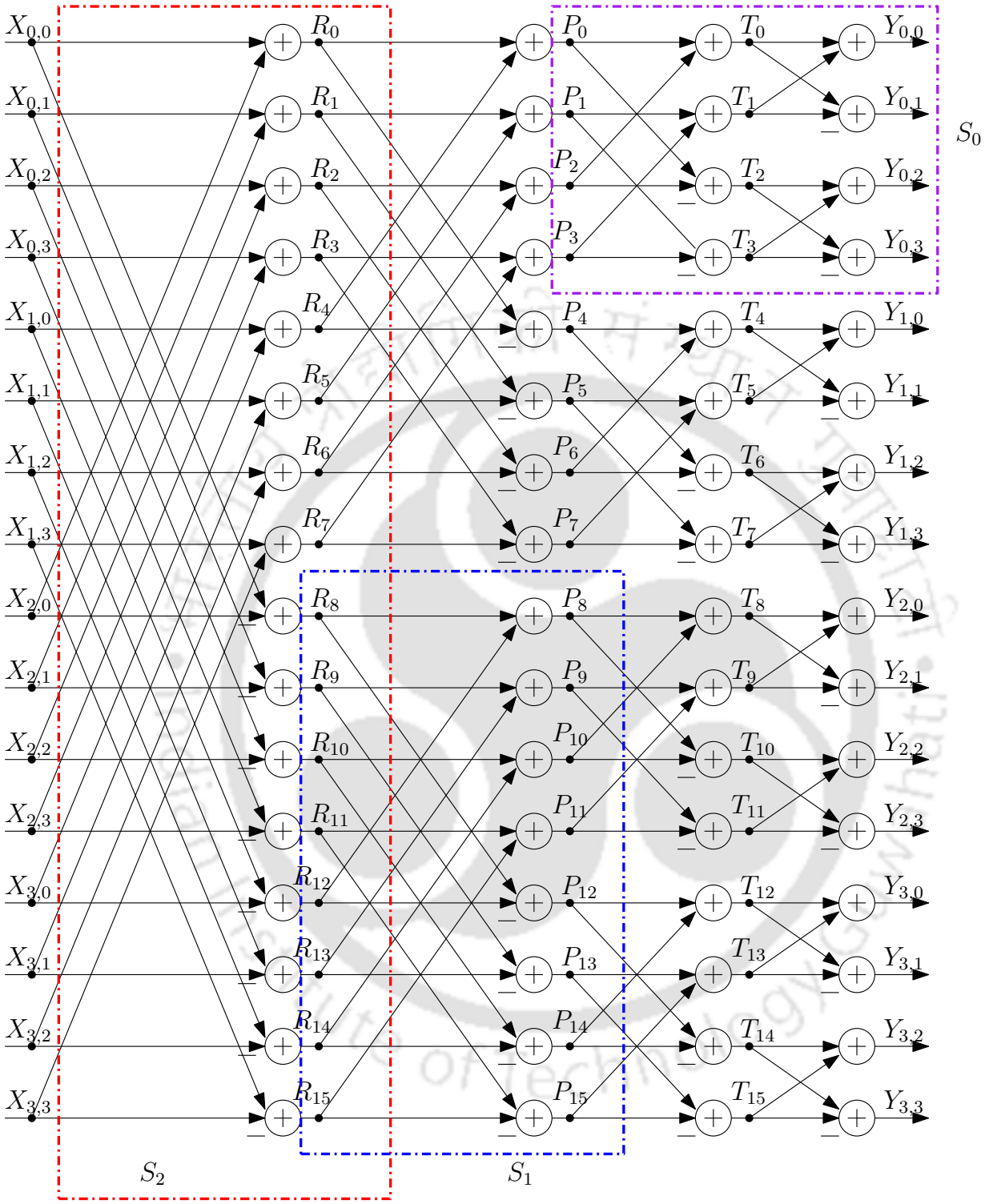


Figure 5.11: Standard Butterfly Flow Diagram for $H_{4\times4}$

be obtained by decomposing the larger HT matrix into smaller HT matrices. Initially, we considered developing an architecture for $H_{4\times4}$. In order to develop architecture for $H_{4\times4}$, we

considered its corresponding butterfly diagram described by (5.4) and (5.8). From the butterfly diagram of $H_{4\times4}$, as shown in Fig. 5.11, it is clear that it consists of one section of S_2, two section of S_1's (S_{1_0}, S_{1_1}) and four sections of S_0 (S_{0_0}, S_{0_1}, S_{0_2}, S_{0_3}). It is observed that each section in Fig. 5.11 can be implemented using simple adders. Now, using S_2, S_1, and S_0, the flow diagram for all the other sizes of HT supported by HEVC can be obtained. In order to realize these flow diagram in an efficient manner, we propose a scalable architecture as shown in Fig. 5.13. The proposed architecture takes one clock cycle to compute 2×2 HT, three clock cycles for 4×4 and four clock cycles for 8×8, 16×4 and 4×16 HTs. The number of S_2, S_1, and S_0 sections needed for the computation of different HT sizes are shown in Table 5.5. For example, 2×2 HT uses only S_{0_0} section. 4×4 HT uses S_{2_0}, S_{1_0} - S_{1_1} and S_{0_0} - S_{0_3} sections. Section S_{1_8} - $S_{1_{15}}$ are used to implement 8×8 HT. 16×4 and 4×16 HTs uses all the sections with different data flow. The data flow through the different sections for different HT sizes is shown in Fig. 5.12.

Table 5.5: Sections Used in 2×2, 4×4, 8×8, 16×4 and 4×16 HTs

HT Size	a	b	c	d	e
2×2	$\times$	$\times$	S_{0_0}	$\times$	$\times$
4×4	S_{2_0}	S_{1_0} - S_{1_1}	S_{0_0} - S_{0_3}	$\times$	$\times$
8×8	$\times$	S_{1_0} - S_{1_7}	S_{0_0} - $S_{0_{15}}$	S_{1_8} - $S_{1_{15}}$	$S_{0_{16}}$ - $S_{0_{31}}$
16×4	S_{2_0} - S_{2_3}	S_{1_0} - S_{1_7}	S_{0_0} - $S_{0_{15}}$	$\times$	$S_{0_{16}}$ - $S_{0_{31}}$
4×16	S_{2_0} - S_{2_3}	S_{1_0} - S_{1_7}	S_{0_0} - $S_{0_{15}}$	$\times$	$S_{0_{16}}$ - $S_{0_{31}}$

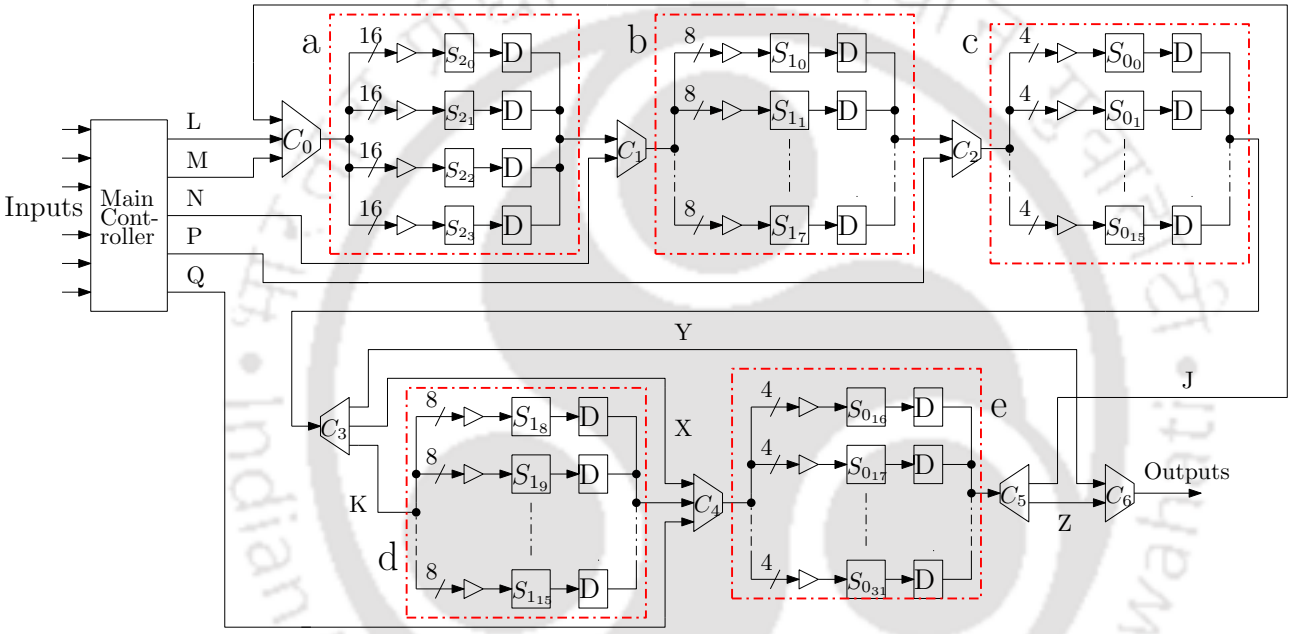
Here a, b, c, d and e are red color blocks which is presented in Fig. 5.13

$$
\begin{aligned}
HT_{2\times2} &: S_{0_0} \\
HT_{4\times4} &: S_{2_0} \longrightarrow S_{1_0} \text{ - } S_{1_1} \longrightarrow S_{0_0} \text{ - } S_{0_3} \\
HT_{8\times8} &: S_{1_0} \text{ - } S_{1_7} \longrightarrow S_{0_0} \text{ - } S_{0_{15}} \longrightarrow S_{1_8} \text{ - } S_{1_{15}} \longrightarrow S_{0_{16}} \text{ - } S_{0_{31}} \\
HT_{16\times4} &: S_{2_0} \text{ - } S_{2_3} \longrightarrow S_{1_0} \text{ - } S_{1_7} \longrightarrow S_{0_0} \text{ - } S_{0_{15}} \longrightarrow S_{0_{16}} \text{ - } S_{0_{31}} \\
HT_{4\times16} &: S_{0_{16}} \text{ - } S_{0_{31}} \longrightarrow S_{2_0} \text{ - } S_{2_3} \longrightarrow S_{1_0} \text{ - } S_{1_7} \longrightarrow S_{0_0} \text{ - } S_{0_{15}}
\end{aligned}
$$

Figure 5.12: Flow of Butterfly Used for Different Size HT

It is clear from Fig. 5.12 that a total of thirty-two S_0's, sixteen S_1's and four S_2's are required to realize all possible sizes of HT supported by HEVC. The main controller is used

to generate the control signal required to connect the paths between different sections. The matrices required for computation of 16×4, 4×4 and 4×16 HTs are applied at the inputs of multiplexer C_0 and are denoted by L, M and J respectively in Fig. 5.13. It can be noted that the input matrix for computation of 4×16 HT arrives from demultiplexer C_5, which is evident from Fig. 5.12. In a similar manner, all the connections among different sections were established using C_1, C_2, C_4 and C_6 multiplexers and C_3 and C_4 demultiplexers.



L = Input for 16×4; M = input for 4×4; N = input for 8×8; P = Input for 2×2; Q = Input for 4×16; Y = Output for 2×2, 4×4 and 4×16; Z = Output for 8×8 and 16×4; J = Intermediate output for 4×16; K = Intermediate output for 8×8; X = Intermediate output for 16×4

Figure 5.13: The Proposed Scalable HT Architecture

In order to determine the final absolute sum, adder tree shown in Fig. 4.5 of Chapter 4 is used. The proposed adder is a combination of carry lookahead adder, 4:2 compressor, 3:2 compressor and carry increment adder. Using the architecture of HT and adder tree, we developed a configurable SATD unit as shown in Fig. 5.14. It can be noted that the proposed SATD unit supports all types of prediction blocks.

The value of the λ and B are calculated using the RD cost unit. The value of λ is calculated using the relation provided in the standard (i.e. $\lambda = \alpha \times W_k \times 2^{((QP-12)/3)}$). More details about

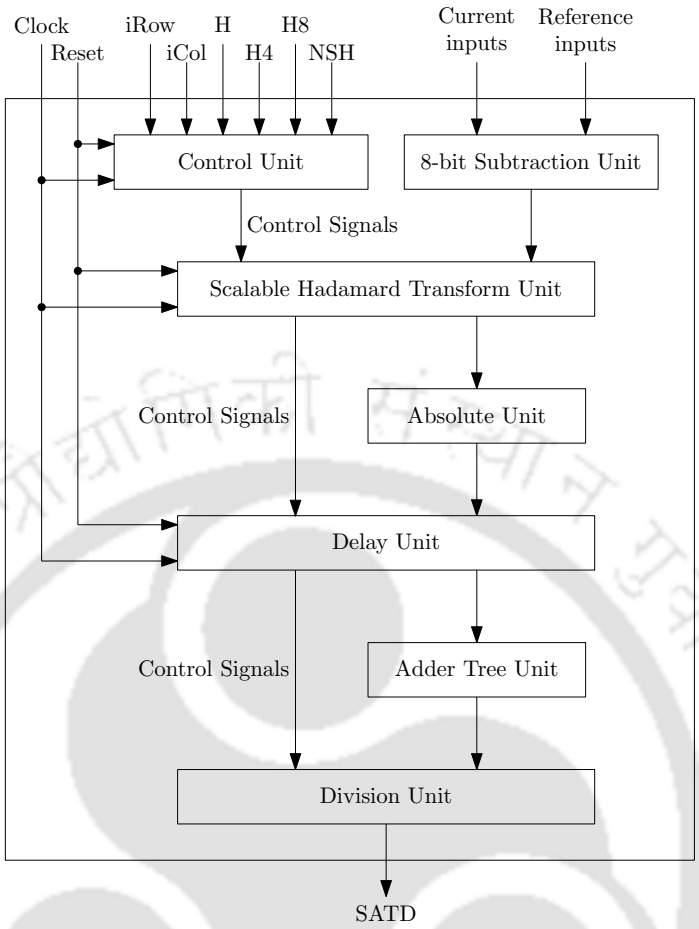


Figure 5.14: Proposed Configurable SATD Unit.

λ, W_k and α are presented in Section 2.3.2 of Chapter 2. Instead of direct multiplication, we used look-up tables to implement the weighting factor (W_k), the power of 2 and λ_{pred}. The block diagram of the proposed RD cost unit is shown in Fig. 5.15. The proposed RD cost unit contains five MV cost units that operate in parallel and calculate the cost for five different positions. The proposed RD cost unit is a three-stage architecture, which produces the output after two clock cycle delay. This RD cost unit depends on QP, λ, GOP, frame type and W_k.

Using the SATD unit, RD unit, the interpolation unit, comparator and memory for integer as well as fractional pixels, we developed a power-efficient architecture for the H5Q5 algorithm as shown in Fig. 5.16. The proposed architecture uses five SATD units because H5Q5 algorithm searches five positions each at half and quarter. Comparator unit is used to compare the SATD value of five positions and select the one with least value as the output. MV decision unit

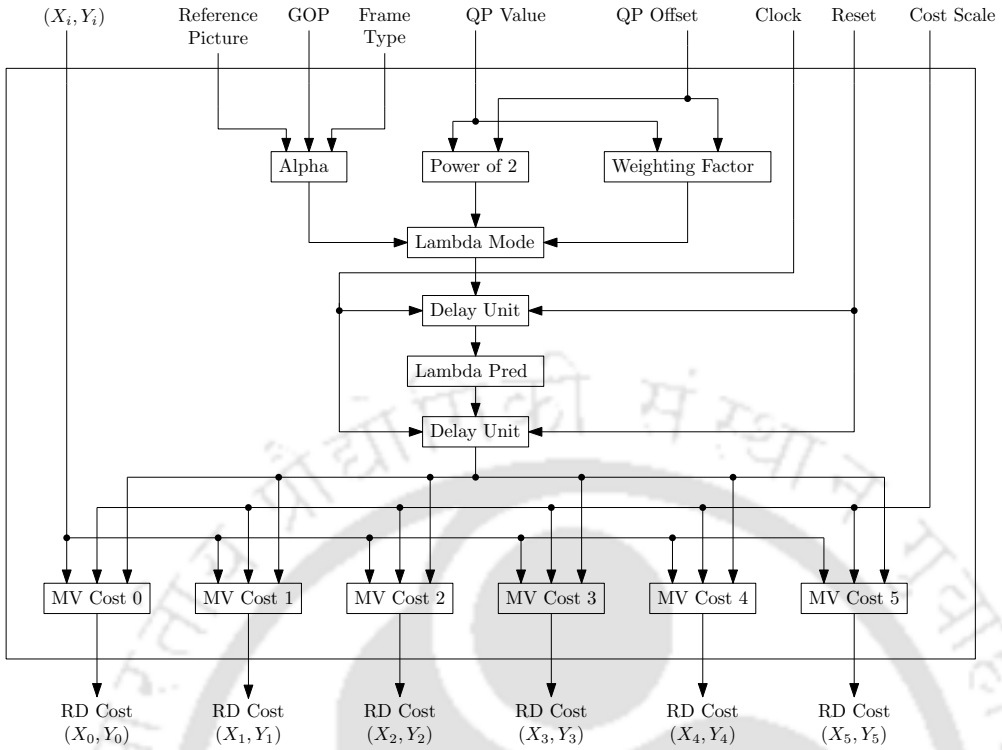


Figure 5.15: Proposed rate distortion cost calculation unit.

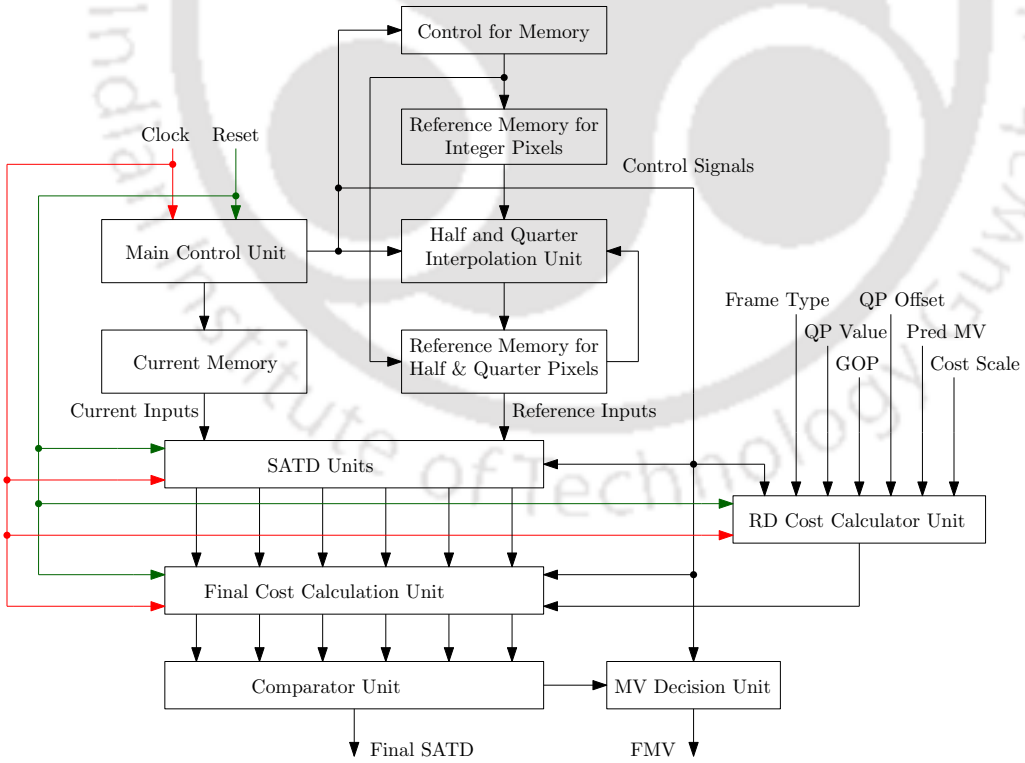


Figure 5.16: Block Diagram of the Proposed FME Architecture.

decides the position of the least SATD.

5.5 Simulation Results for Proposed Architecture

Table 5.6: Comparison of the proposed architecture with state of the art architectures

	TVLSI'2015 [127]	TVLSI'2010 [77]	FME'2013 [58]	TCSVT'2009 [129]	Proposed
Technology	65-nm CMOS	TMSC 0.18 μm	0.13 μm	TMSC 0.18 μm	90 nm
Supply	1.2V	1.2V	1.2V	1.8V	1.0V
Standard	HEVC/H.265	H.264/AVC	H.264/AVC	H.264/AVC	HEVC
Resolution	7680×4320 @ 30 fps	1920×1080 @ 30 fps	1280×720 @ 30 fps	720×480 @ 30 fps	3840×2160 @ 30 fps
Supported Transform	HT8 $\times$ 8/16 $\times$ 16 /32 $\times$ 32	HT4 $\times$ 4	HT4 $\times$ 4	HT4 $\times$ 4	Adaptive*
Algorithm	8/2-tap & 5T12S	6/2-tap & 9T9Sx2	Modified FME	6/2-tap & 9T25S	8/7-tap & H5Q5
Memory	19.2 KB	9.72 KB	NA	NA	28.564 KB
Throughput	995 Mpixel/s	62.2 Mpixels/s	21.69 Mpixel/s	10.4 Mpixel/s	248.8 Mpixel/s
Block Size	64×64 - 16×16	16×16 - 4×4	16×16 - 4×4	16×16 - 4×4	64×64 to 4×4
Area	1183K**	321K**/12.6025 mm^2	285K**	199.2K**	4.2682 mm^2
Power	198.6 mW	374 mW	NA	18.3 mW	111.1763 mW
Frequency	188 MHz	154 MHz	131 MHz	54 MHz	200 MHz
Cost Function	NA	NA	Not Used	NA	Yes

$\star$: HT supported by HEVC $\star\star$: Total Gate Count

In order to validate the proposed architecture, simulations were carried out using Design Compiler of Synopsys and Innovus of Cadence software. The proposed architecture is implemented using standard 90 nm technology. For reducing the computational power, we used 4:2 compressor, carry increment adder and look-up table instead of multiplication. The hardware results of the H5Q5 FME algorithm is compared with the existing state of the art architectures of FME and are tabulated in Table 5.6. From Table 5.6, it is clear that the proposed architecture can process 3840×2160 @ 30 fps video sequences using 28.564 KB of SRAM and occupies an area of 4.2682 mm^2. The proposed architecture consumes 111.1763 mW of power at an operating frequency of 200 MHz and consumes 198.0 mW of power at the maximum frequency of 333.33 MHz. The proposed architecture is capable of operating at a higher operating frequency compared to others state of the art architectures. Moreover, the proposed architecture also support all possible sizes of processing blocks and cost function to calculate the final MV. Additionally, the proposed architecture also includes scalable HT architecture. The layout of the proposed architecture is presented in Fig. 5.17. The layout picture is taken after nano routing and geometry verification using Innovus.

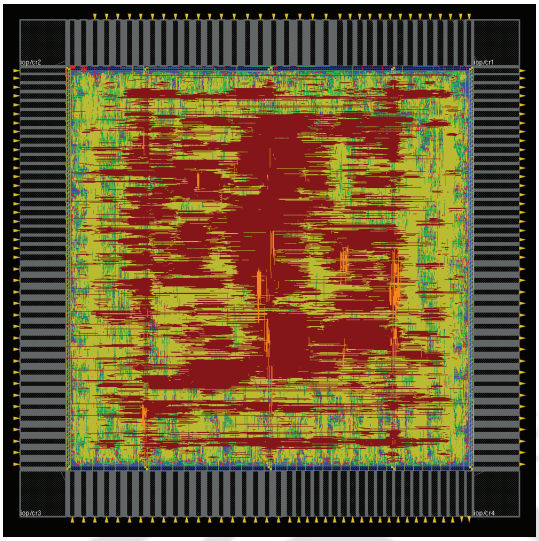


Figure 5.17: Layout of the proposed FME architecture.

5.6 Conclusion

In this chapter, we proposed a new fast FME algorithm for the latest HEVC standard. The proposed algorithm requires less number of search points as compared to the HM reference software. The proposed H5Q5 algorithm takes 47.06% fewer search points with the average increment in BD-bitrate of 0.7262% and the average decrement in BD-PSNR of 0.0256 dB in comparison to HM reference software. The proposed algorithms are particularly suitable for low-power portable devices due to reduced complexity. Using *encoder_randomaccess_main* profiles for QP values of 22, 27, 32 and 37, the BD-PSNR degradation for the proposed H5Q5 algorithm for video sequences $BasketballPass\_416 \times 240@50$ and $Johnny\_1280 \times 720@60$ are 0.0372 dB and 0.0147 dB respectively.

6

Conclusions

Contents

HEVC offers a significant gain in the compression efficiency when compared to the earlier video coding standards such as H.264/AVC. The compression gain mainly results due to new coding tools used in the HEVC standard. However, the use of new coding tools increases the CODEC complexity. Among all blocks, ME is one of the blocks, whose complexity increases to a great extent. The high complexity of the ME is a bottleneck, to adapt in real-time applications. This thesis investigated the overall complexity of inter prediction and proposed several low complexity algorithms and their corresponding low-power architecture for ME process.

This chapter summarizes the main contributions of this thesis. A few directions for future research in relation to the main findings are also presented in this chapter.

6.1 Summary of Contributions

- A detailed study of the pixel truncation, adaptive search range, behaviour of the integer MV, depending upon the depth of the previously encoded block skipping some of the depth of the current block and the effect of removing some of the PU sizes has been carried out. Based on the concepts of the above-mentioned techniques, a computationally efficient ME algorithm for HEVC video coding standard is proposed.

- Developed a power-efficient IME architecture for HEVC video coding standard. In order to validate the importance of the proposed architecture, simulations were carried out using standard 90 nm CMOS technology.

- Performed a detailed study on the behaviour of the final MV and proposed a low complexity FME algorithm. In addition, a low-power VLSI architecture corresponding to the proposed algorithm is also presented. A scalable HT architecture suitable for the HEVC standard is also proposed. The proposed architecture is capable to realize all the sizes of HTs supported by HEVC.

6.2 Future Directions

There are a number of issues which require further investigation in order to enhance the performance of the existing low complexity HEVC encoders. A few possible research directions are listed:

- In the proposed algorithm we used conventional pixel truncation, which discards the information and may lead to the wrong best-matched block in ME. This problem may be solved by using intermediate pixel truncation, which results in less information loss as compared to conventional pixel truncation at the cost of more hardware.
- Approximate computation is one of the popular methods for reducing the computation burden, but it is a lossy method. So, depending upon the application the computational power can be reduced by using approximate adders. The relative study of reduction in computational power and degradation in PSNR is a good area to investigate.
- In literature, several fast IME architectures are available, which are relatively less computationally complex than FME. Thus, in ME the complexity of FME is dominated. Therefore, there is a need for the development of low complexity FME algorithms. Therefore, to reduce the complexity of FME, the possibility of replacing multiple steps with a single step can be further investigated.

Bibliography

[1] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion-compensated interframe coding for video conferencing," in *Proc. NTC 81*, 1981, pp. C9.6.1–9.6.5.

[2] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Trans. Image Process.*, vol. 9, no. 2, pp. 287 –290, Feb 2000.

[3] C. Zhu, X. Lin, and L.-P. Chau, "Hexagon-based search pattern for fast block motion estimation," *IEEE Trans. Circuits. Syst. Video Technol.*, vol. 12, no. 5, pp. 349 –355, May 2002.

[4] B. Natarajan, V. Bhaskaran, and K. Konstantinides, "Low-complexity block-based motion estimation via one-bit transforms," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 4, pp. 702–706, Aug 1997.

[5] A. Lee, D. Jun, and J. S. Choi, "Fast motion estimation using priority-based inter-prediction mode decision method in high efficiency video coding," *J. Real-Time Image Process.*, vol. 12, no. 2, pp. 433–441, Aug 2016. [Online]. Available: http://dx.doi.org/10.1007/s11554-015-0493-7

[6] F. Bossen, B. Bross, K. Suhring, and D. Flynn, "*HEVC C*omplexity and *I*mplementation *A*nalysis," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1685–1696, Dec 2012.

[7] I. K. Kim, K. McCann, K. Sugimoto, B. Bross, and W. Han, "High *E*fficiency *V*ideo *C*oding (*HEVC*) *T*est *M*odel 10 (*HM* 10) *E*ncoder *D*escription," JCT-VC of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 12th Meeting: Geneva, CH, Tech. Rep., 14-23 Jan. 2013.

[8] J. Lee, S. Kim, K. Lim, and S. Lee, "A *F*ast *CU S*ize *D*ecision *A*lgorithm for *HEVC*," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 3, pp. 411–421, Mar 2015.

[9] I. K. Kim, J. Min, T. Lee, W. J. Han, and J. Park, "Block partitioning structure in the hevc standard," *IEEE Trans. Circuits. Syst. Video Technol.*, vol. 22, no. 12, pp. 1697–1706, Dec 2012.

[10] D. Marpe, H. Schwarz, S. Bosse, B. Bross, P. Helle, T. Hinz, H. Kirchhoffer, H. Lakshman, T. Nguyen, S. Oudin, M. Siekmann, K. Suhring, M. Winken, and T. Wiegand, "Video *C*ompression *U*sing *N*ested *Q*uadtree *S*tructures, *L*eaf *M*erging, and *I*mproved *T*echniques for *M*otion *R*epresentation and *E*ntropy *C*oding," *IEEE Trans. Circuit. Syst. Video Technol.*, vol. 20, no. 12, pp. 1676–1687, Dec 2010.

[11] I. Richardson, *Video codec design: developing image and video compression systems.* Wiley, May 2002, no. ISBN: 978-0-471-48553-7.

[12] V. Sze, H*igh E*ffi*ciency* V*ideo* C*oding (*HEVC*)* A*lgorithms and* A*rchitectures*, M. B. Vivienne Sze and G. J. Sullivan, Eds. Springer, 2014.

[13] M. Budagavi, A. Fuldseth, G. Bjntegaard, V. Sze, and M. Sadafale, "Core *T*ransform *D*esign in the *H*igh *E*fficiency *V*ideo *C*oding (*HEVC*) *S*tandard," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 6, pp. 1029–1041, Dec 2013.

[14] C. M. Fu, E. Alshina, A. Alshin, Y. W. Huang, C. Y. Chen, C. Y. Tsai, C. W. Hsu, S. M. Lei, J. H. Park, and W. J. Han, "Sample *A*daptive *O*ffset in the *HEVC S*tandard," *IEEE Trans. Circuit. Syst. Video Technol.*, vol. 22, no. 12, pp. 1755–1764, Dec 2012.

[15] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the *H.264/AVC* video compression standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 620–636, July 2003.

[16] T.-C. Chen, S.-Y. Chien, Y.-W. Huang, C.-H. Tsai, C.-Y. Chen, T.-W. Chen, and L.-G. Chen, "Analysis and architecture design of an *HDTV*720p 30 frames/s *H.264/AVC* encoder," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 6, pp. 673 – 688, Jun 2006.

[17] S. Jou, S. Chang, and T. Chang, "Fast *M*otion *E*stimation *A*lgorithm and *D*esign for *R*eal *T*ime *QFHD H*igh *E*fficiency *V*ideo *C*oding," *IEEE Trans. Circuit. Syst. Video Technol.,*, vol. 25, no. 9, pp. 1533–1544, Sept 2015.

[18] I.-T. R. H.120, *Codecs for videoconferencing using primary digital group transmission*, ITU-T Std., Mar, 1993.

[19] G. J. Sullivan, *Overview of international video coding standards (preceding H.264/AVC)*, July, 2005.

[20] K. R. Rao, D. N. Kim, and J. J. Hwang, *Video coding standards: AVS China, H.264/MPEG-4 PART 10, HEVC, VP6, DIRAC and VC-1*, 1st ed., ser. Signals and Communication Technology. Springer Netherlands, 2014.

[21] G. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the *H*igh *E*fficiency *V*ideo *C*oding (*HEVC*) *S*tandard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, 2012.

[22] I. E. Richardson, *The H.264 Advanced Video Compression Standard*, 2nd ed. Wiley, May 2010, no. ISBN: 978-0-470-51692-8.

[23] K. R. Rao, D. N. Kim, and J. J. Hwang, *Video C*oding *S*tandards AVS C*hina,* H.*264/*MPEG-*4 PART 10,* HEVC*,* VP*6,* DIRAC *and* VC-*1.* Springer, 2014.

[24] L.-M. Po and W.-C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 313 –317, Jun 1996.

[25] L.-K. Liu and E. Feig, "A block-based gradient descent search algorithm for block motion estimation in video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 4, pp. 419 –422, Aug 1996.

[26] *Fast* B*lock-*M*atching* M*otion* E*stimation using* P*redictive* M*otion* V*ector* F*ield* A*daptive* S*earch* T*echnique (*PMVFAST*).* Proceedings of SPIE - The Int. Society for Optical Engineering, Jun. 2001. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.26.5019&rep=rep1&type=pdf

[27] C.-H. Cheung and L.-M. Po, "A novel cross-diamond search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 12, pp. 1168 – 1177, Dec 2002.

[28] Y. Nie and K.-K. Ma, "Adaptive rood pattern search for fast block-matching motion estimation," *IEEE Trans. Image Process.*, vol. 11, no. 12, pp. 1442 – 1449, Dec 2002.

[29] C.-H. Cheung and L.-M. Po, "Novel cross-diamond-hexagonal search algorithms for fast block motion estimation," *IEEE Trans. Multimedia*, vol. 7, no. 1, pp. 16–22, 2005.

[30] H.-M. Wong, O. C. Au, C.-W. Ho, and S.-K. Yip, "Enhanced predictive motion vector field adaptive search technique (*E-PMVFAST*)-based on future *MV* prediction," in *IEEE Int. Conf. on Multimedia and Expo*, July 2005, p. 4 pp.

[31] W. Li and E. Salari, "Successive elimination algorithm for motion estimation," *IEEE Trans. Image Process.*, vol. 4, no. 1, pp. 105 –107, Jan 1995.

[32] X. Gao, C. Duanmu, and C. Zou, "A multilevel successive elimination algorithm for block matching motion estimation," *IEEE Trans. Image Process.*, vol. 9, no. 3, pp. 501 –504, Mar 2000.

[33] Y. Ismail, J. McNeely, M. Shaaban, H. Mahmoud, and M. Bayoumi, "Fast *M*otion *E*stimation *S*ystem *U*sing *D*ynamic *M*odels for *H.264/AVC V*ideo *C*oding," *IEEE Trans. Circuit. Syst. Video Technol.*,, vol. 22, no. 1, pp. 28–42, Jan 2012.

[34] N. Purnachand, L. Alves, and A. Navarro, "Fast *M*otion *E*stimation *A*lgorithm for *HEVC*," in *IEEE Int. Conf. Consumer Electronics - Berlin (ICCE-Berlin), 2012*, Sept 2012, pp. 34–37.

[35] O. Urhan and S. Erturk, "Constrained *O*ne-*B*it *T*ransform for *L*ow *C*omplexity *B*lock *M*otion *E*stimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 4, pp. 478–482, April 2007.

[36] S. Erturk, "Multiplication-*F*ree *O*ne-*B*it *T*ransform for *L*ow-*C*omplexity *B*lock-*B*ased *M*otion *E*stimation," *IEEE Signal Process. Lett.*, vol. 14, no. 2, pp. 109–112, Feb 2007.

[37] M. K. Gullu, "Weighted constrained one-bit transform based fast block motion estimation," *IEEE Trans. Consum. Electron.*, vol. 57, no. 2, pp. 751–755, May 2011.

[38] C. Choi and J. Jeong, "Successive *E*limination *A*lgorithm for *C*onstrained *O*ne-bit *T*ransform *B*ased *M*otion *E*stimation *U*sing the *B*onferroni *I*nequality," *IEEE Signal Process. Lett.*, vol. 21, no. 10, pp. 1260–1264, Oct 2014.

[39] S. Lee, G. Jeon, and J. Jeong, "Fast motion estimation based on enhanced constrained one-bit transform," *Electron. Lett.*, vol. 50, no. 10, pp. 746–748, May 2014.

[40] R. Li, B. Zeng, and M.-L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, no. 4, pp. 438–442, 1994.

[41] K. M. Nam, J.-S. Kim, R.-H. Park, and Y. S. Shim, "A fast hierarchical motion vector estimation algorithm using mean pyramid," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, no. 4, pp. 344–351, 1995.

[42] A. Tourapis, O. Au, and M.-L. Liou, "Highly efficient predictive zonal algorithms for fast block-matching motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 10, pp. 934–947, 2002.

[43] A. M. Tourapis, "Enhanced predictive zonal search for single and multiple frame motion estimation," in *Visual Comm. Image Process. 2002 (VCIP-2002)*. International Society for Optics and Photonics, 2002, pp. 1069–1079.

[44] L.P.Chau and C.Zhu, "A fast octagon-based search algorithm for motion estimation," *Signal Process.*, vol. 83, no. 3, pp. 671 – 675, 2003.

[45] C. Zhu, X. Lin, L. Chau, and L.-M. Po, "Enhanced hexagonal search for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 10, pp. 1210–1214, Oct 2004.

[46] X. Yi, J. Zhang, N. Ling, and W. Shang, "Improved and simplified fast motion estimation for *JM*," Santa Clara University, Tech. Rep., Jul 2005.

[47] Z. Chen, J. Xu, Y. He, and J. Zheng, "Fast integer-pel and fractional-pel motion estimation for *H.264/AVC*," *J. Visual Commun. Image Representation*, vol. 17, no. 2, pp. 264–290, 2006.

[48] C.-Y. Chen, Y.-W. Huang, C.-L. Lee, and L.-G. Chen, "One-pass computation-aware motion estimation with adaptive search strategy," *IEEE Trans. Multimedia*, vol. 8, no. 4, pp. 698–706, Aug 2006.

[49] T. C. Chen, Y. H. Chen, S. F. Tsai, S. Y. Chien, and L. G. Chen, "Fast *A*lgorithm and *A*rchitecture *D*esign of *L*ow-*P*ower *I*nteger *M*otion *E*stimation for *H.264/AVC*," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 5, pp. 568–577, May 2007.

[50] Y.-K. Lin, C.-C. Lin, T.-Y. Kuo, and T.-S. Chang, "A *H*ardware-*E*fficient *H.264/AVC M*otion-*E*stimation *D*esign for *H*igh-*D*efinition *V*ideo," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 6, pp. 1526–1535, 2008.

[51] S. Zhu, J. Tian, X. Shen, and K. Belloulata, "A new cross-diamond search algorithm for fast block motion estimation," in *IEEE Int. Conf. Image Process.*, 2009, pp. 1581–1584.

[52] O. Ndili and T. Ogunfunmi, "Algorithm and *A*rchitecture *C*o-*D*esign of *H*ardware-*O*riented, *M*odified *D*iamond *S*earch for *F*ast *M*otion *E*stimation in *H.264/AVC*," *IEEE Trans. Circuit. Syst. Video Technol.,*, vol. 21, no. 9, pp. 1214–1227, 2011.

[53] A.-C. Tsai, K. Bharanitharan, J.-F. Wang and K.-I. Lee, "Effective *S*earch *P*oint *R*eduction *A*lgorithm and its *VLSI D*esign for *HDTV H.*264/*AVC V*ariable *B*lock *S*ize *M*otion *E*stimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 7, pp. 981 –988, Jul 2012.

[54] H. Kibeya, F. Belghith, H. Loukil, M. A. B. Ayed, and N. Masmoudi, "*TZ S*earch pattern search improvement for *HEVC* motion estimation modules," in *1st Int. Conf. Advanced Technol. for Signal and Image Process. (ATSIP)*, March 2014, pp. 95–99.

[55] G. Pastuszak and M. Jakubowski, "Adaptive *C*omputationally *S*calable *M*otion *E*stimation for the *H*ardware *H.264/AVC E*ncoder," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 5, pp. 802–812, May 2013.

[56] L. Li, S. Liu, Y. Chen, T. Chen, and T. Luo, "Motion *E*stimation *W*ithout *I*nteger-*P*el *S*earch," *IEEE Trans. Image Process.,*, vol. 22, no. 4, pp. 1340–1353, 2013.

[57] G. Pastuszak and M. Jakubowski, "Adaptive *C*omputationally *S*calable *M*otion *E*stimation for the *H*ardware *H.264/AVC E*ncoder," *IEEE Trans. Circuit. Syst. Video Technol.*, vol. 23, no. 5, pp. 802–812, 2013.

[58] O. Ndili and T. Ogunfunmi, "Fast *A*lgorithm and *E*fficient *A*rchitecture for *I*nteger and *F*ractional *M*otion *E*stimation," *J. Signal Process. Syst.*, vol. 75, no. 1, pp. 55–64, 2014. [Online]. Available: http://dx.doi.org/10.1007/s11265-013-0793-8

[59] N. Al-Najdawi, M. N. Al-Najdawi, and S. Tedmori, "Employing a novel cross-diamond search in a modified hierarchical search motion estimation algorithm for video compression," *Information Sciences*, vol. 268, pp. 425 – 435, 2014, new Sensing and Processing Technologies for Hand-based Biometrics Authentication. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025513005690

[60] A. Erturk and S. Erturk, "Two-bit transform for binary block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 7, pp. 938–946, July 2005.

[61] T. Komarek and P. Pirsch, "Array architectures for block matching algorithms," *IEEE Trans. Circuits Syst.*, vol. 36, no. 10, pp. 1301–1308, Oct 1989.

[62] H. Yee and Y. H. Hu, "A novel modular systolic array architecture for full-search block matching motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, no. 5, pp. 407 –416, Oct 1995.

[63] H. Yeo and Y. H. Hu, "A modular high-throughput architecture for logarithmic search block-matching motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 3, pp. 299 –315, Jun 1998.

[64] Z. He, M. L. Lieu, P. C. H. Chan, and R. Li, "An efficient *VLSI* architecture for new three-step search algorithm," in *38th Midwest Symposium on Circuits and Systems*, vol. 2, Aug 1995, pp. 1228–1231 vol.2.

[65] S. Y. Yap and J. V. McCanny, "A *VLSI* architecture for variable block size video motion estimation," *IEEE Trans. Circuits Syst. II: Express Briefs*, vol. 51, no. 7, pp. 384–389, July 2004.

[66] T. Dias, N. Roma, L. Sousa, and M. Ribeiro, "Reconfigurable architectures and processors for real-time video motion estimation," *J. Real-Time Image Process.*, vol. 2, no. 4, pp. 191–205, 2007. [Online]. Available: http://dx.doi.org/10.1007/s11554-007-0049-6

[67] C. Wei, H. Hui, T. jiarong, L. Jinmei, and M. Hao, "A high-performance reconfigurable *VLSI* architecture for *VBSME* in *H.*264," *IEEE Trans. Consum. Electron.*, vol. 54, no. 3, pp. 1338–1345, 2008.

[68] J. Kim and T. Park, "A novel *VLSI* architecture for full-search variable block-size motion estimation," *IEEE Trans. Consum. Electron.*, vol. 55, no. 2, pp. 728–733, May 2009.

[69] J. Vanne, E. Aho, K. Kuusilinna, and T. Hamalainen, "A *C*onfigurable *M*otion *E*stimation *A*rchitecture for *B*lock-*M*atching *A*lgorithms," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 4, pp. 466 –477, Apr 2009.

[70] S. K. Chatterjee and I. Chakrabarti, "Low power *VLSI* architectures for one bit transformation based fast motion estimation," *IEEE Trans. Consum. Electron.*, vol. 56, no. 4, pp. 2652–2660, November 2010.

[71] C. Y. Kao and Y. L. Lin, "A memory-efficient and highly parallel architecture for variable block size integer motion estimation in h.264/avc," *IEEE Trans. Very Large Scale Integration (VLSI) Syst.*, vol. 18, no. 6, pp. 866–874, June 2010.

[72] T. H. Tsai and Y. N. Pan, "High *E*fficiency *A*rchitecture *D*esign of *R*eal-*T*ime *QFHD* for *H.264/AVC F*ast *B*lock *M*otion *E*stimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 11, pp. 1646–1658, Nov 2011.

[73] T.-C. Chen, Y.-H. Chen, C.-Y. Tsai, and L.-G. Chen, "Low power and power aware fractional motion estimation of *H.264/AVC* for mobile applications," in *IEEE Int. Symp. Circuits Syst.*, May 2006, pp. 4 pp.–.

[74] Y. J. Wang, C. C. Cheng, and T. S. Chang, "A *F*ast *A*lgorithm and *I*ts *VLSI A*rchitecture for *F*ractional *M*otion *E*stimation for *H.264/MPEG-4 AVC V*ideo *C*oding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 5, pp. 578–583, May 2007.

[75] Y.-H. Chen, T.-C. Chen, S.-Y. Chien, Y.-W. Huang, and L.-G. Chen, "*VLSI A*rchitecture *D*esign of *F*ractional *M*otion *E*stimation for *H.264/AVC*," *J. Signal Process. Syst.*, vol. 53, no. 3, pp. 335–347, Dec 2008. [Online]. Available: https://doi.org/10.1007/s11265-008-0213-7

[76] S. K. Chatterjee and I. Chakrabarti, "A high performance *VLSI* architecture for fast two-step search algorithm for sub-pixel motion estimation," in *2009 Int. Multimedia, Signal Process. and Commun. Technol.*, March 2009, pp. 205–208.

[77] C. Y. Kao, C. L. Wu, and Y. L. Lin, "A *H*igh-*P*erformance *T*hree-*E*ngine *A*rchitecture for *H.264/AVC F*ractional *M*otion *E*stimation," *IEEE Trans. Very Large Scale Integration (VLSI) Syst.*, vol. 18, no. 4, pp. 662–666, Apr 2010.

[78] D. B. Lim, Y. K. Choi, H. J. Lee, and S. I. Chae, "A fast fractional motion estimation algorithm for high efficiency video coding," in *Int. Conf. Electron. Inform. and Commun. (ICEIC)*, Jan 2016, pp. 1–4.

[79] K. M. Yang, M. T. Sun, and L. Wu, "A family of *VLSI* designs for the motion compensation block-matching algorithm," *IEEE Trans. Circuits Syst.*, vol. 36, no. 10, pp. 1317–1325, Oct 1989.

[80] Z. Liu, Y. Song, M. Shao, S. Li, L. Li, S. Ishiwata, M. Nakagawa, S. Goto, and T. Ikenaga, "*HDTV*1080p *H.264/AVC E*ncoder *C*hip *D*esign and *P*erformance *A*nalysis," *IEEE J. Solid-State Circuits*, vol. 44, no. 2, pp. 594–608, Feb 2009.

[81] D. Zhou, J. Zhou, G. He, and S. Goto, "A 1.59 *G*pixel/s motion estimation processor with - 211 to +211 search range for *UHDTV* video encoder," *IEEE J. Solid-State Circuit.,*, vol. 49, no. 4, pp. 827–837, Apr 2014.

[82] "Frame rate." [Online]. Available: https://en.wikipedia.org/wiki/Frame_rate

[83] "Different types of frame rate." [Online]. Available: https://documentation.apple.com/en/finalcutpro/usermanual/index.html

[84] "Display resolution." [Online]. Available: https://en.wikipedia.org/wiki/Display_resolution

[85] ITU-T Recommendation H.261, "Video codec for audiovisual services at p x 64 kbits/sec," Tech. Rep., Mar 1993.

[86] Mpeg-1. [Online]. Available: https://en.wikipedia.org/wiki/MPEG-1

[87] *Information technology - Generic coding of moving pictures and associated audio information: Video*, ITU-T Recommendation H.262 Std., Aug 1995.

[88] H.262/mpeg-2 part 2. [Online]. Available: https://en.wikipedia.org/wiki/H.262/MPEG-2_Part_2#cite_note-mpeg-2-2013-2

[89] H.263. [Online]. Available: https://en.wikipedia.org/wiki/H.263

[90] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the *H.264/AVC* video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, 2003.

[91] "H.264/mpeg-4 avc." [Online]. Available: https://en.wikipedia.org/wiki/H.264/MPEG-4_AVC

[92] J. Vanne, M. Viitanen, and T. D. Hmlinen, "*E*fficient *M*ode *D*ecision *S*chemes for *HEVC I*nter *P*rediction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 9, pp. 1579–1593, Sept 2014.

[93] K. McCann, C. Rosewarne, B. Bross, M. Naccari, K. Sharman, and G. Sullivan, "High *E*fficiency *V*ideo *C*oding (*HEVC*) *T*est *M*odel 16 (*HM* 16) *E*ncoder *D*escription," JCT-VC of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 18th Meeting: Sapporo, JP, Tech. Rep., 30 June - 7 July. 2014.

[94] High *E*fficiency *V*ideo *C*oding. [Online]. Available: https://en.wikipedia.org/wiki/High_Efficiency_Video_Coding

[95] [Online]. Available: https://www.itu.int/en/ITU-T/studygroups/2017-2020/16/Pages/video/jvet.aspx

[96] [Online]. Available: https://mpeg.chiariglione.org/standards/exploration/future-video-coding

[97] M. Wien, *High E*ff*iciency V*i*deo C*o*ding: C*o*ding T*o*ols and S*pecification.* Springer, 2015.

[98] T. Nguyen, P. Helle, M. Winken, B. Bross, D. Marpe, H. Schwarz, and T. Wiegand, "Transform *C*oding *T*echniques in *HEVC*," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 6, pp. 978–989, Dec 2013.

[99] J. Lainema, F. Bossen, W. J. Han, J. Min, and K. Ugur, "Intra *C*oding of the *HEVC S*tandard," *IEEE Trans. Circuits. Syst. Video Technol.*, vol. 22, no. 12, pp. 1792–1801, Dec 2012.

[100] A. Fuldseth, G. Bjntegaard, M. Budagavi, and V. Sze, CE*10: C*ore transform design for* HEVC, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 Std., Nov. 2011.

[101] Cisco, "Cisco *V*isual *N*etworking *I*ndex: *G*lobal *M*obile *D*ata *T*raffic *F*orecast *U*pdate, 2014-2019," Cisco, Tech. Rep., Feb, 3 2015.

[102] Y.-W. Huang, B.-Y. Hsieh, S.-Y. Chien, S.-Y. Ma, and L.-G. Chen, "Analysis and complexity reduction of multiple reference frames motion estimation in *H.264/AVC*," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 4, pp. 507–522, April 2006.

[103] "*HM R*eference *S*oftware 12." [Online]. Available: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/

[104] S.-H. Yang, J.-Z. Jiang, and H.-J. Yang, "Fast motion estimation for *HEVC* with directional search," *Electron. Lett.*, vol. 50, no. 9, pp. 673–675, Apr 2014.

[105] H. Ding, F. Wang, W. Zhang, and Q. Zhang, "Adaptive motion search range adjustment algorithm for *HEVC* inter coding," *Optik - Int. J. for Light and Electron Optics*, vol. 127, no. 19, pp. 7498 – 7506, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0030402616305289

[106] B. Biswas, R. Mukherjee, and I. Chakrabarti, "Efficient architecture of adaptive rood pattern search technique for fast motion estimation," *Microprocess. and Microsyst.*, vol. 39, no. 3, pp. 200 – 209, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0141933115000149

[107] P. Arnaudov and T. Ogunfunmi, "A *CAM* enabled fast video motion estimation based on locality sensitive signatures," in *IEEE Int. Symposium on Circuits and Systems (ISCAS)*, May 2017, pp. 1–4.

[108] ——, "Adaptive search pattern for fast motion estimation in *HD* video," in *51st Asilomar Conf. Signals, Systems, and Computers*, Oct 2017, pp. 173–177.

[109] ——, "Raster *S*earch *R*esolution *A*nalysis for *F*ast *M*otion *E*stimation in *HD* *V*ideo *C*ompression," in *IEEE Workshop Signal Processing Systems(SiSP 2018)*, Oct 2018, pp. 1–6.

[110] S. K. Chatterjee and I. Chakrabarti, "Power efficient motion estimation algorithm and architecture based on pixel truncation," *IEEE Trans. Consum. Electron.*, vol. 57, no. 4, pp. 1782–1790, Nov 2011.

[111] S. Goel, Y. Ismail, and M. A. Bayoumi, "Adaptive search window size algorithm for fast motion estimation in *H.264/AVC* standard," in *48th Midwest Symp. Circuit. Syst., 2005.*, Aug 2005, pp. 1557–1560 Vol. 2.

[112] N. Purnachand, L. N. Alves, and A. Navarro, "Improvements to *TZ* search motion estimation algorithm for multiview video coding," in *19th Int. Conf. Systems, Signals and Image Processing (IWSSIP)*, April 2012, pp. 388–391.

[113] G. Bjontegaard, "*C*alculation of average *PSNR* differences between *RD* curves," ITUT-T Q6/SG16, Doc. VCEG-M33, Tech. Rep., Apr 2001.

[114] W.-M. Chao, C.-W. Hsu, Y.-C. Chang, and L.-G. Chen, "A novel hybrid motion estimator supporting diamond search and fast full search," in *IEEE Int. Symposium Circuits Syst., ISCAS 2002*, vol. 2, 2002, pp. II–492 – II–495 vol.2.

[115] C.-Y. Chen, S.-Y. Chien, Y.-W. Huang, T.-C. Chen, T.-C. Wang, and L.-G. Chen, "Analysis and architecture design of variable block-size motion estimation for *H.264/AVC*," *IEEE Trans. Circuits Syst. 1: Regular Papers*, vol. 53, no. 3, pp. 578–593, 2006.

[116] S.-F. Tsai, C.-T. Li, H.-H. Chen, P.-K. Tsung, K.-Y. Chen, and L.-G. Chen, "A 1062*M*pixels/s 8192x4320p high efficiency video coding (*H.265*) encoder chip," in *Symp. VLSI Circuits (VLSIC)*, Jun 2013, pp. C188–C189.

[117] J. Byun, Y. Jung, and J. Kim, "Design of integer motion estimator of *HEVC* for asymmetric motion-partitioning mode and *4K-UHD*," *Electron. Lett.*, vol. 49, no. 18, pp. 1142–1143, Aug 2013.

[118] S. Lee, G. Jeon, and J. Jeong, "Fast motion estimation based on enhanced constrained one-bit transform," *Electron. Lett.*, vol. 50, no. 10, pp. 746–748, May 2014.

[119] M. E. Sinangil, V. Sze, M. Zhou, and A. P. Chandrakasan, "Cost and *C*oding *E*fficient *M*otion *E*stimation *D*esign *C*onsiderations for *H*igh *E*fficiency *V*ideo *C*oding (*HEVC*) *S*tandard," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 6, pp. 1017–1028, Dec 2013.

[120] K. Singh and S. R. Ahamed, "Modified *S*mall-*C*ross *D*iamond *S*earch *M*otion *E*stimation *A*lgorithm for *H.264/AVC*," in *Annual IEEE India Conf. (INDICON)*, Dec 2013, pp. 1–5.

[121] ——, "A *N*ew *M*otion *E*stimation *A*lgorithm for *H*igh *E*fficient *V*ideo *C*oding *S*tandard," in *Annual IEEE India Conf. (INDICON)*, Dec 2015, pp. 1–5.

[122] K. Goswami, J.-H. Lee, and B.-G. Kim, "Fast algorithm for the *H*igh *E*fficiency *V*ideo *C*oding (*HEVC*) encoder using texture analysis," *Information Sciences*, vol. 364-365, pp. 72 – 90, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025516303425

[123] K. Singh and S. R. Ahamed, "Computationally *E*fficient *M*otion *E*stimation *A*lgorithm for *HEVC*," *J. Signal Process. Systems*, vol. 90, no. 12, pp. 1713–1727, Dec 2018. [Online]. Available: https://doi.org/10.1007/s11265-017-1321-z

[124] ——, "Low *P*ower *M*otion *E*stimation *A*lgorithm and *A*rchitecture of *HEVC*/*H*.265 for *C*onsumer *A*pplications," *IEEE Trans. Consum. Electron.*, vol. 64, no. 3, pp. 267–275, Aug 2018.

[125] Y. K. Lin, D. W. Li, C. C. Lin, T. Y. Kuo, S. J. Wu, W. C. Tai, W. C. Chang, and T. S. Chang, "A 242m*W* 10mm2 1080p *H.264/AVC H*igh-*P*rofile *E*ncoder *C*hip," in *IEEE Int. Solid-State Circuits Conf. - Digest of Technical Papers*, Feb 2008, pp. 314–615.

[126] P. K. Tsung, W. Y. Chen, L. F. Ding, C. Y. Tsai, T. D. Chuang, and L. G. Chen, "Single-iteration full-search fractional motion estimation for quad full *HD H.264/AVC* encoding," in *IEEE Int. Conf. on Multimedia and Expo*, Jun 2009, pp. 9–12.

[127] G. He, D. Zhou, Y. Li, Z. Chen, T. Zhang, and S. Goto, "High-*T*hroughput *P*ower-*E*fficient *VLSI A*rchitecture of *F*ractional *M*otion *E*stimation for *U*ltra-*HD HEVC V*ideo *E*ncoding," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. PP, no. 99, pp. 1–1, 2015.

[128] D. Kang, Y. Kang, and Y. Hong, "*VLSI* implementation of fractional motion estimation interpolation for high efficiency video coding," *Electron. Lett.*, vol. 51, no. 15, pp. 1163–1165, 2015.

[129] Y. H. Chen, T. C. Chen, C. Y. Tsai, S. F. Tsai, and L. G. Chen, "Algorithm and *A*rchitecture *D*esign of *P*ower-*O*riented *H.264/AVC B*aseline *P*rofile *E*ncoder for *P*ortable *D*evices," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 8, pp. 1118–1128, Aug 2009.

List of Publications

Publication in Refereed Journals

- Karam Singh and Shaik Rafi Ahamed "Computationally Efficient Motion Estimation Algorithm for HEVC," J. Signal Process. Systems, vol. 90, no. 12, pp. 1713-1727, Dec 2018.. ISSN 1939-8115. doi:10.1007/s11265-017-1321-z.

- Karam Singh and Shaik Rafi Ahamed "Low Power Motion Estimation Algorithm and Architecture of HEVC/H.265 for Consumer Applications," IEEE Trans. Consum. Electron., vol. 64, no. 3, pp. 267-275, Aug. 2018.

Manuscripts Under Review

- Karam Singh and Shaik Rafi Ahamed "Fast Fractional Motion Estimation Algorithm and Architecture for HEVC," Under Review in IEEE Trans. Circuit Syst. Video Technol.

- Karam Singh and Shaik Rafi Ahamed "Scalable VLSI Architecture for Hadamard Transforms for HEVC/H.265 Video Coding Standard," Under Review in IEEE Trans. Consum. Electron.(Resubmitted after revision)

Manuscript Under Preparation

- Karam Singh and Shaik Rafi Ahamed "A Single Step Fractional Motion Estimation Algorithm for HEVC/H.265 Video Coding Standard," manuscript under preparation.

Publication in Refereed Conferences

- K. Singh and S. R. Ahamed, "A new Motion Estimation algorithm for high efficient video coding standard," 2015 Annual IEEE India Conf. (INDICON), New Delhi, 2015, pp. 1-5. doi: 10.1109/INDICON.2015.7443614

- K. Singh and S. R. Ahamed, "Modified small-cross diamond search motion estimation algorithm for H.264/AVC," 2013 Annual IEEE India Conf. (INDICON), Mumbai, 2013, pp. 1-5. doi: 10.1109/INDCON.2013.6726117

- K. Singh and S. R. Ahamed, "Systolic array based architecture for DS fast motion estimation algorithm," 2013 IEEE Second Int. Conf. Image Information Process. (ICIIP-2013), Shimla, 2013, pp. 335-339. doi: 10.1109/ICIIP.2013.6707611