APPROXIMATED TRANSFORM AND QUANTISATION FOR COMPLEXITY-REDUCED HIGH EFFICIENCY VIDEO CODING

A thesis submitted for the degree of Doctor of Philosophy

by

Mohd Mohd Sazali

College of Engineering, Design, and Physical Sciences

Brunel University

March 2017

Acknowledgement

My sincere acknowledgement goes to my principal supervisor, Prof. Dr Abdul H. Sadka, for his productive guidance, constructive criticisms, invaluable knowledge sharing, brilliant thoughts, and continuous support throughout my research at Brunel University London. My gratitude is extended to my second supervisor Dr Nikolaos V. Boulgouris for his valuable inputs and wonderful help. I am also thankful to my current and past colleagues, Salim Al Amri, Dr Sagir Lawan, Taha Alfaqheri, Dr Mohamed Rafiq Swash, Dr Hamdullah Mohib, Dr Obaid Fatah, Dr Abdulkadir Audu, Dr Abdulkareem Ibrahim, and many others for their technical advice and friendly company during the course of my research. Special thanks also to Tony Morris, John Morse, and their respective teams for their technical assistance.

My genuine appreciation is dedicated to my great parents, Dr Mohd Sazali Khalid and Shamsinar Jaafar, along with my helpful siblings for their continuous encouragement, financial support, and endless prayers. Most of all, I am deeply indebted to my wonderful wife, Siti Nazmin, for her long sacrifice and kind understandings to allow me completing my study far away from the family. I owe our beloved sons so much, Iman Shahdan and Irshad Shahidin, for having to spend a big part of their childhood time without me being around. To our newly born son, Imdad Sharfan, hopefully one day you can appreciate the hardships that we had gone through and be as strong as your brothers.

Last but not least, I wish to express my appreciation to my generous sponsors, Universiti Sains Malaysia (USM) and Malaysian Ministry of Higher Education (MoHE) for funding my study. Even though far from being perfect, it is hoped that this thesis will be beneficial in one way or another to the reader and the body of knowledge.

Abstract

The transform-quantisation stage is one of the most complex operations in the state-of-the-art High Efficiency Video Coding (HEVC) standard, accounting for 11–41% share of the encoding complexity. This study aims to reduce its complexity, making it suitable for dedicated hardware accelerated architectures. Adopted methods include multiplier-free approach, Multiple-Constant Multiplication architectural designs, and exploiting useful properties of the well-known Discrete Cosine Transform. Besides, an approximation scheme was introduced to represent the original HEVC transform and quantisation matrix elements with more hardware-friendly integers. Out of several derived approximation alternatives, an approximated transform matrix (T16) and its downscaled version (ST16) were further evaluated. An approximated quantisation multipliers matrix (Q) and its combination with one transform matrix (ST16 + Q) were also assessed in HEVC reference software, HM-13.0, using test video sequences of High Definition (HD) quality or higher. Their hardware architectures were designed in IEEE-VHDL language targeting a Xilinx Virtex-6 Field Programmable Gate Array technology to estimate resource savings over original HEVC transform and quantisation. T16, ST16, Q, and ST16 + Q approximated transform or/and quantisation matrices provided average Bjøntegaard-Delta bitrate differences of 1.7%, 1.7%, 0.0%, and 1.7%, respectively, in entertainment scenario and 0.7%, 0.7%, -0.1%, and 0.7%, respectively, in interactive scenario against HEVC. Conversely, around 16.9%, 20.8%, 21.2%, and 25.9% hardware savings, respectively, were attained in the number of Virtex-6 slices compared with HEVC transform or/and quantisation. The developed architecture designs achieved a 200 MHz operating frequency, enabling them to support the encoding of Quad Full HD (3840 × 2160) videos at 60 frames per second. Comparing T16 and ST16 with similar designs in the literature yields better hardware efficiency measures (0.0687 and 0.0721, respectively, in mega sample/second/slice). The presented approximated transform and quantisation matrices may be applicable in a complexity-reduced HEVC encoding on hardware platforms with non-detrimental coding performance degradations.

Keywords: Hardware complexity, HEVC, FPGA, quantisation, transform

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

1-D	One-Dimensional
2-D	Two-Dimensional
3-D	Three-Dimensional
3D-HEVC	3-D HEVC
3G	Third generation of mobile telephony technology
AI	All Intra
AMVP	Advanced Motion Vector Prediction
ASIC	Application Specific Integrated Circuit
AVC	Advanced Video Coding
AVS	Audio Video Coding Standard of China
B	Bi-predicted
BD-Rate	Bjøntegaard-Delta Bitrate
BD-PSNR	Bjøntegaard-Delta PSNR
BRAM	Block RAM
CABAC	Context Adaptive Binary Arithmetic Coding
CAVLC	Context Adaptive Variable Length Coding
Cb	Chrominance Blue
CB	Coding Block
cc	Clock Cycle
CCD	Charge-Coupled Device
CCIR	International Radio Consultative Committee
Cg	Chrominance Green
CIF	Common Intermediate Format
CM	Control Module
CMOS	Complementary MOSFET
CODEC	Encoder/Decoder
CPB	Coded Picture Buffer
CPD	Critical Path Delay
Cr	Chrominance Red
CSD	Canonical Signed Digit

CSE	Common Sub-expression Elimination
CTB	Coding Tree Block
CTU	Coding Tree Unit
CU	Coding Unit
dB	Decibel
DCT	Discrete Cosine Transform
DM	Data path Module
DPB	Decoded Picture Buffer
DRAM	Distributed RAM
DSCQS	Double Stimulus Continuous Quality Scale
D_{sf}	DCT scaled and floating
DSP	Digital Signal Processor
D_{sr}	DCT scaled and rounded
DST	Discrete Sine Transform
DVD	Digital Versatile Disc
FCVC	Fully Configurable Video Coding
FPGA	Field Programmable Gate Array
fps	Frames Per Second
FR	Full Reference
FSM	Finite State Machine
GOP	Group of Pictures
HD	High Definition
HDL	Hardware Description Language
HEVC	High Efficiency Video Coding
HLL	High Level Language
HVS	Human Visual System
I	Intra
IAU	Input Adder Unit
IC	Integrated Circuit
ICT	Integer Discrete Cosine Transform
IDCT	Inverse DCT
ID	Identification
IEC	International Electrical Committee

ISO	International Organization for Standardization
ITU	International Telecommunication Union
ITU-R	Radiocommunication Sector of ITU
ITU-T	Telecommunication Sector of ITU
JCT-3V	Joint Collaborative Team on 3D Video Coding Extension Development
JCT-VC	Joint Collaborative Team on Video Coding
JND	Just-Noticeable Difference
JVT	Joint Video Team
KLCC	Kendall Rank-Order Correlation Coefficient
LB	Low Delay with B-pictures
LP	Low Delay with P-pictures
LSB	Least Significant Bit
LUT	Look-Up Table
MAE	Mean Absolute Error
MC	Motion Compensation
MB	Macro Blocks
MCM	Multiple-Constant Multiplication
MICT	Modified ICT
MOS	Mean Opinion Score
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
MOVIE	Motion-tuned Video Integrity Evaluation
MPEG	Motion Picture Experts Group
MSB	Most Significant Bit
MSE	Mean Squared Error
MS-SSIM	Multiple-Scale SSIM
MST	Multi-Standard Transform
MV	Motion Vector
MVC	Multiview Video Coding
MV-HEVC	Multiview HEVC
NICT	Non-orthogonal ICT
NR	No Reference
NS	Next State

OAU	Output Adder Unit
OS	Operating System
P	Predicted
PB	Prediction Block
PLCC	Pearson Linear Correlation Coefficient
POC	Picture Order Count
PS	Present State
PSNR	Peak Signal to Noise Ratio
PU	Prediction Unit
QCIF	Quarter CIF
QP	Quantisation Parameter
QPI	QP Intra
QM	Quantisation Module
RA	Random Access
RAM	Random Access Memory
R-D	Rate-Distortion
RDOQ	Rate-Distortion Optimised Quantisation
RExt	Format Range Extensions
RGB	Red, Green, Blue
RMS	Root Mean Square
RR	Reduced Reference
RS	Rounding and Scaling
RVC	Reconfigurable Video Coding
S2P	Serial-to-Parallel
SAO	Sample Adaptive Offset
SAU	Shift-Add Unit
SCC	Screen Content Coding
SD	Standard Definition
SHVC	Scalable HEVC
SMIC	Semiconductor Manufacturing International Corporation
SRAM	Synchronous RAM
SRCC	Spearman Rank-Order Correlation Coefficient
SSIM	Structural Similarity Index

SVC	Scalable Video Coding
TB	Transform Block
TM	Transform Module
TQM	Transform and Quantisation Module
TSMC	Taiwan Semiconductor Manufacturing Company
TU	Transform Unit
UHD	Ultra HD
VCEG	Video Coding Experts Group
VGA	Video Graphics Array
VHDL	VHSIC HDL
VHSIC	Very High Speed Integrated Circuit
VQEG	Video Quality Experts Group
VQM	Video Quality Metric
WQXGA	Wide Quad Extended Graphics Array
WVGA	Wide VGA
Y	Luminance
YUV	YCbCr colour space sub-sampling
YUY2	4:2:2 YUV

LIST OF APPENDICES

Chapter 1

Introduction

Abstract This chapter introduces the field and topics of research, the aim, objectives, and scope of work carried out in this thesis. The study contributions to the body of knowledge and an outline of this document are included towards the end of the chapter.

1.1 Video Coding

It is no exaggeration to state that an average individual today watches video contents on a daily basis. Videos are being watched almost anytime and everywhere – at homes, work places, restaurants, parks, on travels, and many other examples. The growing number of video-enabled devices such as laptops, smartphones, tablets, always-on wearable cameras, etc. besides the common television (TV) sets is just one of many factors contributing to this trend. The increasing variety of video services created around the clock like broadcast programmes (e.g., news, sports, documentaries, entertainment shows, etc.), streaming applications, home cinemas, video chats, surveillances, and so on, is another factor influencing one's regular activities. Various other factors include improving spatial and temporal video resolutions, quality of experience, availability of internet connections, economies of scale, etc.

In 2015, more than half a billion (563 million) new mobile devices and connections were produced worldwide, with 36% of the share being smart devices (Cisco, 2016). In their study, smart devices were defined as "mobile connections that have advanced multimedia/computing capabilities with a minimum of 3G connectivity". The mobile data traffic around the globe reached 3.7 exabytes (EB) every month by the end of 2015, and more than half (55%) of this traffic was mobile video. By 2020, the monthly mobile data traffic is projected to increase by eight times to 30.6 EB globally, and three-fourths (75%) of these will be video (Cisco, 2016). Recently, the most popular sources for online TV and/or movies are YouTube, Netflix, Hulu, Amazon Prime Instant Video, and HBOGO, according to a survey (Solsman, 2014). For instance on YouTube, as of July 2015, more than 400 hours of video were uploaded per minute, increasing from 300 hours every minute in November 2014 (Jarboe, 2015; Robertson, 2015; Statista, 2016). Statistical data such as these lay emphasis on the significance of video in our daily lives.

A raw video sequence incurs a certain amount of byte size. Table 1.1 illustrates that a short video with a length of 10 minutes in different formats may require between 5 gigabytes (GB) in a low-resolution Common Intermediate Format (CIF) and 445 GB in an Ultra High Definition (UHD) 4K format. This massive size is not ideal for storing or transmitting a raw video. As a video is a form of signal or data, compression and decompression, which respectively reduces and returns the original data size, have therefore been long applied to support the handling of video contents. The art of compressing and decompressing a video along with associated processes is what referred to as video coding.

Table 1.1 Size of a 10-minute raw video in several resolution formats

Format	**Resolution (Width × Height)**	**Bits per pixel**[a]	**Frames per second**[b] **(fps)**	**Size**[c] **(GB)**
Common Intermediate Format (CIF)	352 × 288	24	30	5.1
Standard Definition (SD)	720 × 576	24	30	20.9
High Definition (HD)	1280 × 720 (720p)	24	30	46.3
	1920 × 1080 (1080p)	24	30	104.3
Ultra High Definition (UHD)	3840 × 2160 (Quad Full HD, QFHD)	24	30	417.1
	4096 × 2160 (4K)	24	30	444.9

[a] Three colour components each with 8-bit depth
[b] Other common frame rates include 25, 50, and 60 fps
[c] Size = Width × Height × Bits per pixel × FPS × Length / (2^{30} × 8)

Video coding is based on a pair of complementary systems: an encoder and a decoder. The encoder converts (encodes) an original video sequence into a compressed form or bitstream, reducing the number of bits required and making it suitable for transmission or storage. On the other hand, the decoder converts (decodes) the compressed bitstream back into a representation (exact copy or approximation) of the source video. The encoder-decoder pair is frequently referred to as a CODEC (enCOder/DECoder) (Fig. 1.1). Compression is possible by removing redundant information or elements that are unnecessary for reconstructing the data. If the decoded data are exactly identical to the original, then the coding process is classified as lossless. Otherwise, the process is lossy (Richardson, 2012).

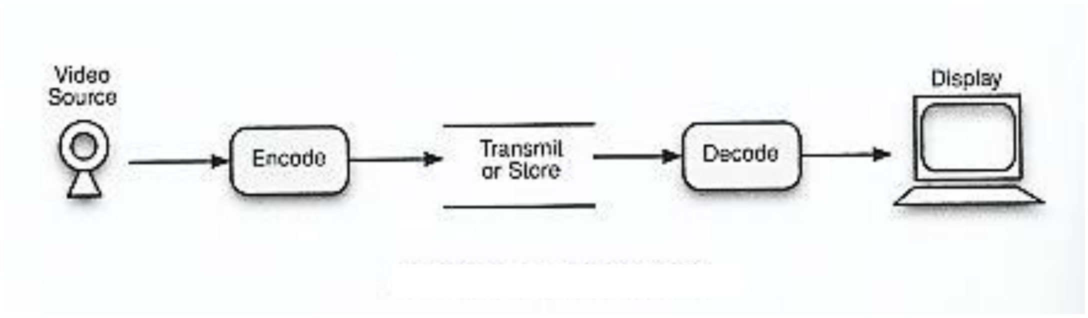


Fig. 1.1 Video encoder-decoder (CODEC) system (Richardson, 2012)

In a lossless video coding, compression is achieved by removing statistical redundancy such as bit patterns. However, lossless coding only provides a moderate amount of data reduction and may still be unsuitable for storage or transmission. On the other hand, a lossy coding delivers a higher video compression ratio by removing subjective redundancy, and thus is normally in place but at the expense of a visual quality loss. Fortunately, subjective redundant elements of a video sequence are removable without severely influencing the viewer’s visual quality perception (Richardson, 2012).

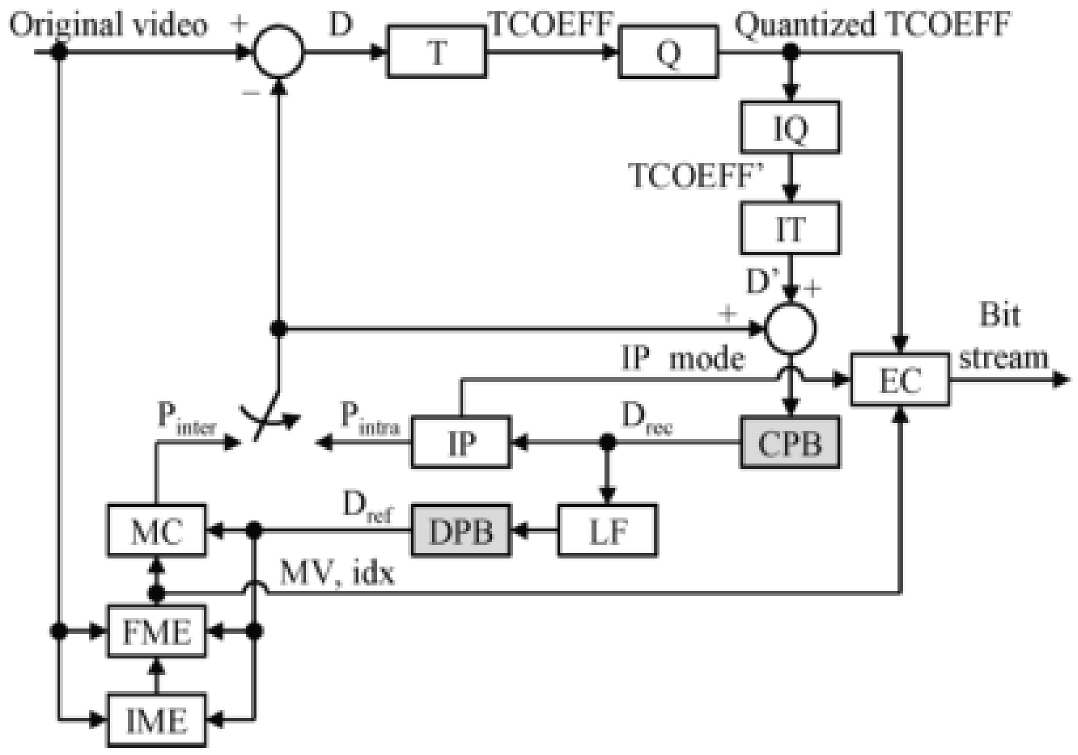
In order to have a common language or understanding between a party encoding a video sequence and another one wishing to decode the compressed video contents, standards for video coding have been developed since the last three decades. Most video coding standards apply lossy compression to obtain higher compression efficiencies. Increasing significance of video driven primarily by the growing number of video sources, applications, resolutions, quality, etc. has led to improved coding tools or even newer standards to be established. When a new video coding standard is being developed, usually one of its goals is to achieve a higher compression ratio than the best available standard at that time, without sacrificing video quality. A higher compression ratio means that more video sequences can be stored or transmitted at the same number of bits. Alternatively, the savings in the number of bits can also be utilised to deliver a better video quality such as larger resolutions or richer viewing experiences (e.g., three-dimensional (3-D), multiview, holoscopic, etc.).

A better performing video coding standard usually comes at the price of increased complexity. To achieve a higher compression ratio requires more sophisticated coding algorithms or tools attributing to more complex designs. It is almost obligatory for new video-enabled electronic devices to include a capability to use the newest available video coding standard as one of its selling points. Having this as the context, the next section presents the motivation and problem definition which this thesis is based on.

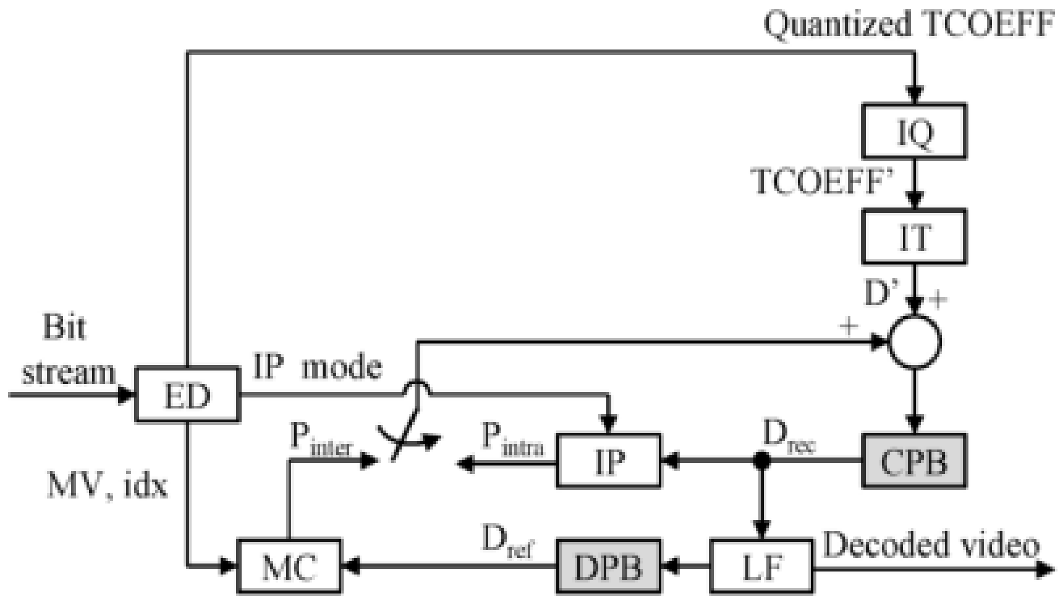
1.2 Motivation and Problem Description

The latest and most efficient video coding standard at present is the High Efficiency Video Coding (HEVC). HEVC was developed by the Joint Collaborative Team on Video Coding (JCT-VC), formed by Video Coding Experts Group (VCEG) of the Telecommunication Sector of the International Telecommunication Union (ITU-T) and Moving Picture Experts Group (MPEG) of the International Organization for Standardization (ISO)/International Electrical Committee (IEC). The first version of HEVC was published in April 2013 as ITU-T Recommendation (Rec.) H.265 (ITU, 2013). In ISO/IEC, the standard was first published in November 2013 as 23008-2 MPEG-H Part 2 (ISO, 2013b). The newly emerged UHD format was one of the driving factors leading to the development of HEVC.

When HEVC was approved, it provided almost twice compression efficiency at similar video qualities in comparison to the state-of-the-art standard, the Advanced Video Coding (AVC) (Ohm *et al.*, 2012; ITU, 2014). Like AVC, HEVC employs the classical hybrid video coding structure combining intra-/inter-prediction, transform coding, and entropy coding (Sullivan *et al.*, 2012; Vanne *et al.*, 2012). A general model of the HEVC encoder and decoder can be depicted as in Fig. 1.2(a) and 1.2(b), respectively.



(a)



(b)

Fig. 1.2 HEVC encoder-decoder model: (a) Encoder (b) Decoder (Vanne *et al.*, 2012)

> “There is no single coding element in the HEVC design that provides the majority of its significant improvement in compression efficiency in relation to prior video coding standards. It is, rather, a plurality of smaller improvements that add up to the significant gain” (Sullivan *et al.*, 2012, p. 1654).

The average software complexity of an HEVC encoder, based on the execution time, varies between 1.2× and 3.2× when compared with an AVC encoder, while on the decoder side, the corresponding value ranges from 1.4× to 2.0× (Vanne *et al.*, 2012). Tables 1.2 and 1.3 respectively show the average encoder and decoder software complexity in different coding configurations (All Intra (AI), Random Access (RA), Low Delay with B-pictures (LB), and Low Delay with P-pictures (LP)).

Table 1.2 Average shares of the most complex HEVC encoding stages (Vanne *et al.*, 2012)

Encoding stage	**AI**	**RA**	**LB**	**LP**
IME	0%	16%	18%	17%
FME/MD	9%	55%	59%	49%
IP	24%	1%	1%	1%
T/Q/IQ/IT	41%	14%	11%	18%
EC	11%	4%	3%	5%
Misc.	15%	10%	8%	10%

Table 1.3 Average shares of the most complex HEVC decoding stages (Vanne *et al.*, 2012)

Decoding stage	**AI**	**RA**	**LB**	**LP**
IME	13%	13%	12%	14%
FME/MD	0%	47%	44%	34%
IP	25%	4%	2%	3%
T/Q/IQ/IT	23%	9%	11%	12%
EC	13%	5%	5%	6%
Misc.	26%	22%	26%	31%

The most complex stages presented in Tables 1.2 and 1.3 are therefore high priority candidates for acceleration on dedicated hardware architectures.

> “Accelerating the most complex functions such as motion compensation (MC) is recommended in decoding, but an adequate decoding performance is typically obtainable through processor-based acceleration. However, HEVC codec is strongly asymmetrical in terms of complexity, so sufficient encoding performance tends to be out of

reach unless the most complex encoding functions are off-loaded to special hardware accelerators” (Vanne *et al.*, 2012, p. 1894).

With the newly approved HEVC as the motivation and its complexity as one of the main challenges to be addressed, the next section states the aim and objectives set in this thesis.

1.3 Aim and Objectives

The aim of this work is to evaluate the coding performance of novel complexity-reduced algorithms of selected HEVC algorithms. As previously mentioned, the HEVC encoder has a higher priority to be simplified than the decoder. One of the most complex encoding stages is the transform/quantisation/inverse quantisation/inverse transform (T/Q/IQ/IT) as shown previously in Table 1.2 and Fig. 1.2(a). For these reasons, the T/Q/IQ/IT stage of HEVC has been selected as the focus of this thesis.

In order to achieve the specified aim, the following objectives were underlined to be carried out:

- To assess the coding performance of simplified transform and quantisation algorithms in terms of objective video quality metrics, bitrate, and visual observations;
- To assess the hardware implementation costs of simplified transform and quantisation algorithms;
- To compare between simplified transform and quantisation algorithms and HEVC and other works in the literature.

1.4 Scope of Work

The work carried out was based on the first version of HEVC standard and do not cover extensions introduced in later versions. Although the HEVC codec comprises an encoder and a decoder, this work mainly focuses on the encoder. Furthermore in the HEVC encoder, the work concentrates on the transform and quantisation stages. All the other stages were retained as specified in the standard.

The encoding results were obtained by using HEVC model software version 13.0 (HM-13.0) (JCT-VC, 2014) as the reference. This version was the latest release when this research work started. The software was considered mature when an older version 10.0 was made available.

The hardware architecture designs of the transform and quantisation stages in this work were described in reconfigurable IEEE-VHDL hardware description language (HDL) using Xilinx™ Integrated Synthesis Environment (ISE®) design suite version 14.7, synthesised with Xilinx™ Synthesis Technology (XST) and routed to Xilinx™ Virtex®-6 xc6vl550t-2ff1760 Field Programmable Gate Array (FPGA) target device. In addition, these hardware designs were implemented up to the Placed and Routed state in ISE® software, verified using test benches and simulated in the ISE® Simulator (ISim) environment along with Mathworks® MATLAB® software.

1.5 Thesis Contributions

This thesis introduces the following contributions:

- Two approximated transform algorithms for HEVC, labelled as T16 and ST16, with similar coding performances of 1.7% and 0.7% Bjøntegaard-Delta bitrate (BD-rate) differences on average in entertainment and interactive application scenarios, respectively, over the original HEVC transform, considering video qualities of HD and above.
- An approximated quantisation algorithm for HEVC, labelled as Q, with 0.0% and -0.1% average BD-rate differences in entertainment and interactive application scenarios, respectively, against the original HEVC quantisation, considering HD video quality and beyond.
- A combination of approximated transform and quantisation stage for HEVC, labelled as ST16 + Q, with average BD-rate differences of 1.7% and 0.7% in entertainment and interactive application scenarios, respectively, with respect to the original HEVC transform and quantisation, considering HD and better video qualities.
- Two high-throughput and multiplier-free dedicated hardware architecture designs of the approximated transforms for HEVC (T16 and ST16), utilising

16.9% and 20.8% fewer resources in terms of Xilinx Virtex-6 slices compared with HEVC transform. These designs are at least 1.3× more hardware efficient than a few similar architectural designs, operating at a higher frequency (200 MHz) and capable of supporting QFHD @ 60 fps videos.

- A dedicated hardware architecture design of the approximated quantisation for HEVC (Q), using more than 20% fewer slices than HEVC quantisation hardware, achieving a higher operating frequency (200 MHz) and a better QFHD processing frame rate (60 fps).
- A dedicated hardware architecture design of the combined and approximated transform and quantisation stage for HEVC (ST16 + Q), offering more than 25% slice savings than HEVC transform and quantisation hardware on top of a higher operating frequency (200 MHz) and a better QFHD processing frame rate (60 fps).

Based on a few of the aforementioned contributions, the following manuscript has been reviewed by the corresponding journal and a revision is being prepared:

- Mohd Sazali, M., Sadka, A. H., Boulgouris, N. V. (2017) ‘Two-dimensional approximated core transforms for High Efficiency Video Coding’, *Elsevier Signal Processing: Image Communication*

1.6 Thesis Outline

This thesis is divided into seven chapters. After the introductory information given in Chapter 1, Chapter 2 provides a background description on digital video coding covering the acquisition of a digital video, colour spaces and sub-sampling, and a few quality metrics commonly used to assess a digital video. A brief history of video coding standardisation is also included. Towards the end of this chapter, a brief overview of the HEVC standard (version 1) is provided, describing its video coding layer, the supported profiles, tiers, and levels.

Chapter 3 is dedicated to explaining the HEVC transform and quantisation algorithms, as these processes form the basis of this thesis as stated earlier (Section 1.3). Some related work in the literature is discussed at the end of the chapter.

Chapter 4 describes the approximated transform (including intermediate scaling) and quantisation algorithms for HEVC. These approximated algorithms are compared with the original ones in terms of degree of approximations and/or arithmetic complexity.

Chapter 5 presents the software-based experimental coding performance results of the approximated transform and quantisation algorithms, in terms of objective quality rate-distortion and subjective visual observation. Some descriptions of the test conditions, data set, and encoder-decoder compatibility issues are discussed in this chapter.

Chapter 6 presents dedicated hardware architecture designs developed in this work for HEVC and simplified transform and quantisation algorithms. Some explanation on hardware-software co-design methodology is provided at the beginning. The performances of these designs are compared with the original HEVC algorithms as well as with some work in the literature, in terms of resource utilisation, supported spatial video resolutions, and hardware efficiency.

Finally, Chapter 7 concludes this document, discusses some of its strengths and weaknesses, and offers some suggestions as part of a future work.

Chapter 2

Background

Abstract This chapter provides a background on the concepts required to understand the novel contributions of this work. It revises the concepts of digital video capture, representation, quality, and coding, and provides an overview of the new HEVC standard.

2.1 Digital Video Capture and Representation

2.1.1 Digital Video Capture

A digital video is a sequence of images, frames or pictures, where each of these pictures represents a sample of two-dimensional (2-D) projection (discrete space and time) of a real scene (continuous space and time) (Fig. 2.1). Every picture is a rectangular matrix of pixels where the number of pixels in the horizontal (width) and vertical (height) directions determines the spatial resolution of the picture and video. Common video resolutions include SD with 720 × 480 pixels, Full HD (or 1080p) with 1920 × 1080 pixels, 4K with 4096 × 2160 pixels, etc. The number of pictures captured per second determines the temporal resolution or frame rate of the video in frames per second (fps). Common frame rates include 24 fps, 30 fps, 50 fps, 60 fps, etc. A sufficiently high capture rate gives an observer the impression of motion of the scene. The higher the picture rate is, the smoother the feeling of motion is to the viewer.

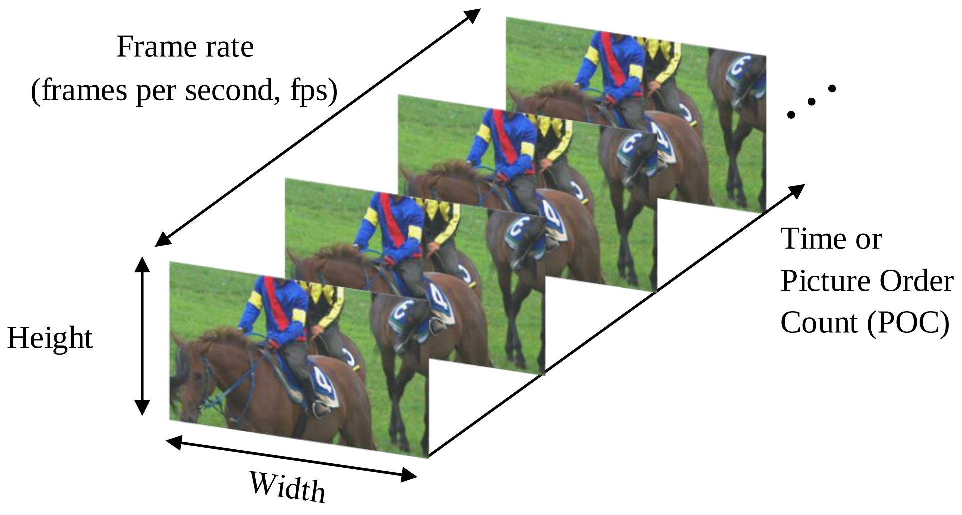


Fig. 2.1 A digital video sequence

Each image is captured by an analogue semiconductor sensor formed by an array of Charge-Coupled Devices (CCDs), where every CCD captures one pixel. Colour images normally require three matrices of CCDs, each matrix representing one colour component following a colour space. A common colour space is the Red, Green, and Blue (RGB) as most other colours can be created using certain combinations of these three base colours. In the RGB space, every pixel therefore has three colour components or samples. The intensity level of each colour samples is determined by a number of bits or bit-depth. For instance, 8-bit and 10-bit depths could allow the colour intensity of a sample to vary between zero and 255 and 1023, respectively.

2.1.2 Digital Video Representation

The human visual system (HVS) has two types of photoreceptor cells, namely rods and cones. Rods sense the brightness or intensity of light (luminance or luma), while cones sense colours (chrominance or chroma). The HVS is less sensitive to colours than brightness. Thus, a tricolour space such as RGB is normally represented in YCbCr (or YUV) space prior to storage or transmission, where Y represents the luma samples while Cb (U) and Cr (V) respectively represent the chroma blue and chroma red samples. Cg or chroma green samples are unnecessary as they can be obtained using (2.1)–(2.2) (Richardson, 2012).

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

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

The YCbCr or YUV space allows the representation of a digital video to be sub-sampled taking the advantage of the HVS property. Common YUV sampling patterns include 4:4:4, 4:2:2, and 4:2:0 (Fig. 2.2). In the 4:4:4 sampling pattern, for every square consisting of four Y samples (two-by-two), there are also four U and V samples, i.e., the number of Y, U, and V samples are exactly the same in both vertical and horizontal directions. 4:4:4 fully retains the fidelity of the chrominance component, having the same effect as the RGB. On the other hand, 4:2:2 or YUY2 has the same chrominance samples as the luminance component in the vertical directions but only half U and V samples in the horizontal directions. The 4:2:2 pattern is used for high-quality colour reproduction. The most popular among them is the 4:2:0 pattern, where each U and V components only have half the vertical and horizontal resolutions of the Y component. In other words, for every four Y samples, there are only one U and V samples. The term 4:2:0 is confusing and should instead have been 4:1:1, but was retained for historical reasons and to differentiate it from 4:4:4 and 4:2:2. Consumer applications like digital versatile disk (DVD) storage and video conferencing broadly adopt the 4:2:0 pattern (Richardson, 2012).

Table 2.1 demonstrates the number of bits involved when a 10-second 1080p HD @ 30 fps 8-bit video is captured in different YUV sampling patterns. Applying a different YUV sampling pattern could reduce the size of a video and it is, therefore, a form of image or video compression.

Table 2.1 A 10-second 1080p video in different YUV sampling patterns

Sampling pattern	Size (number of bits)
4:4:4 (or RGB)	1920 × 1080 × 3 × 30 fps × 10s × 8-bit = 14.9 × 10^9 bits
4:2:2	1920 × 1080 × 2 × 30 fps × 10s × 8-bit = 9.95 × 10^9 bits
4:2:0	1920 × 1080 × 1.5 × 30 fps × 10s × 8-bit = 7.46 × 10^9 bits

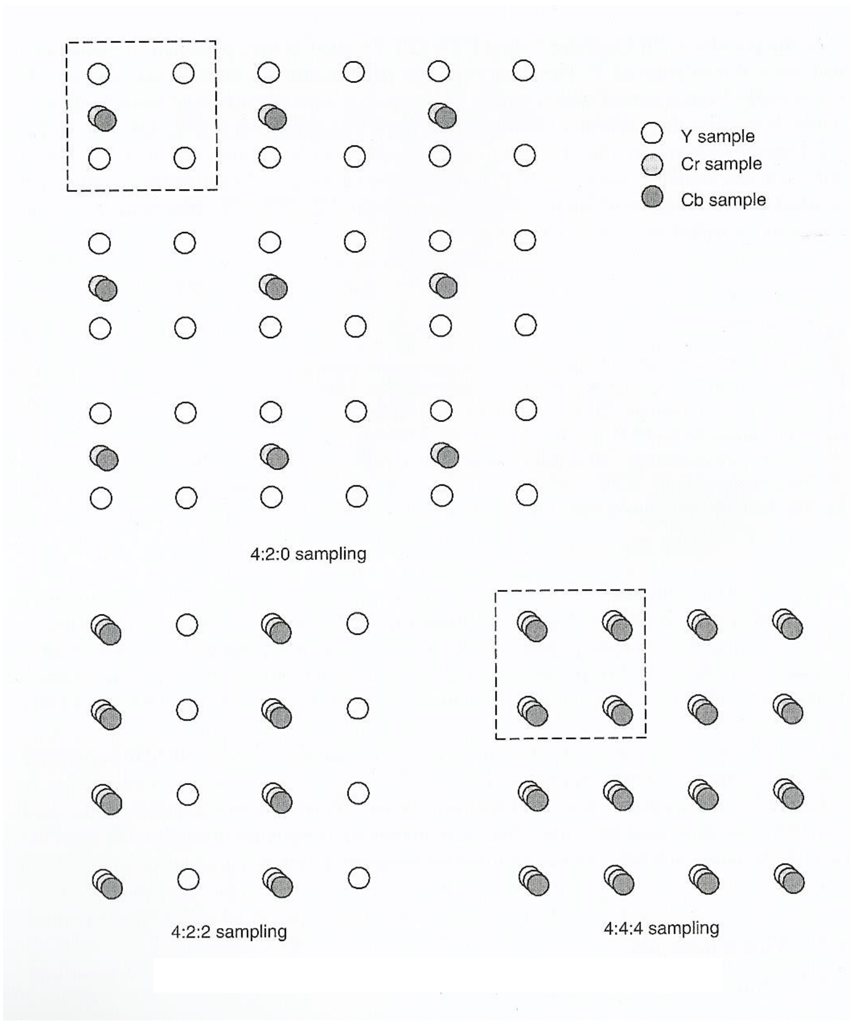


Fig. 2.2 YUV sampling patterns (Richardson, 2012)

2.2 Video Quality

The quality of a video can be evaluated using objective or subjective measurements. As viewing a video sequence is a visual experience, a subjective test is probably the best in measuring the quality of experience of a viewer. A subjective test involves human beings assessing the quality of videos. A few guidelines on subjective video evaluation are provided by ITU. In Double Stimulus Continuous Quality Scale (DSCQS) testing system, a pair of unimpaired and distorted video is displayed one after the other (Fig. 2.3). The order of either the original or the impaired video is to be displayed first is randomised to reduce biased judgements of the assessor. At the end of each test, the human assessor will mark his or her view on the relative video quality on a continuous scale, and this scale can be classified or divided into different quality levels such as five or nine quality levels (Video Clarity, 2016). Table 2.2 shows five quality levels of video quality. This type of grading is usually called Mean Opinion Score (MOS), as at the end of the whole evaluation, the mean or average score of all human assessors involved will be taken as the quality score in a particular test condition or parameter.

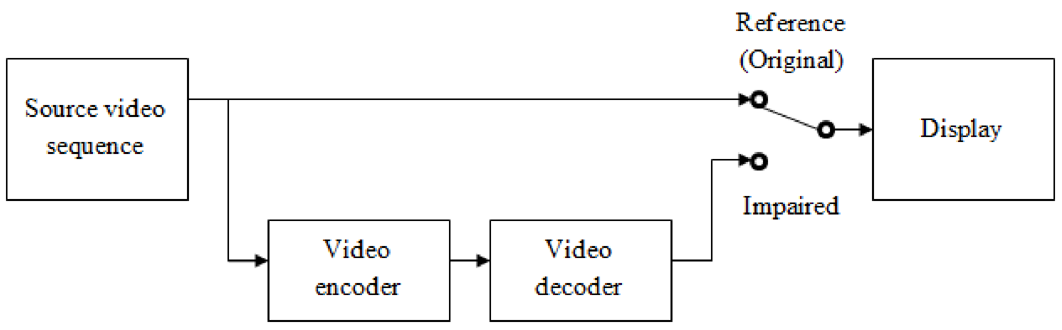


Fig. 2.3 DSCQS testing system (adapted from (Richardson, 2012))

Although a subjective test such as DSCQS is the closest in measuring a video quality from the human point of view, it has several practical drawbacks. First, it is very time-consuming and costly to gather a large pool of human assessors or evaluators. An expert evaluator, who is acquainted with the artefacts or distortions caused by video compression, tends to be biased in giving a quality score. Therefore, it is more advisable to use naive or non-expert evaluators who are inexperienced in evaluating video qualities. Even a non-expert assessor can also quickly become an expert as he/she learns to recognise artefacts during the evaluation process. It is also expensive to have the test facility set-up involving the display equipment, controlled environment (e.g., lighting and shades or reflectiveness of the wall or curtains), etc.

Table 2.2 Five quality levels of video quality (Video Clarity, 2016)

Score	**Quality**
5	Excellent
4	Good
3	Fair
2	Poor
1	Unacceptable

The eye and the brain form the components of the HVS. The interaction between the eye and the brain, components of the HVS, influences a human's visual quality perception or opinion. This perception is determined firstly by the spatial fidelity, i.e., how clearly or distorted regions of a scene, and secondly by the temporal fidelity, i.e., how smoothly motions appear in the scene. Other influencing factors include the viewing environment, the amount of the viewer's interaction with video contents (active or passive), the observer's state of mind, visual attention, i.e., the concentration of an observer on a series of points in the images instead of simultaneously taking all information into the brain ((Findlay and Gilchrist, 2003) cited in (Richardson, 2012)), and the 'recency effect', i.e., our perceived quality is affected more heavily by a recently viewed video rather than an older content ((Wade and Swanston, 2001) and (Aldridge *et al.*, 1995) cited in (Richardson, 2012)). All these factors complicate the quantitative and accurate measurement of visual quality (Richardson, 2012), as well as to have repeatable results even if using the same group of human assessors.

The cost and complexity to subjectively measure video quality make objective quality measurements using mathematical functions more desirable. Video processing system developers heavily depend on objective or algorithmic quality measurements in approximating the response of human observers. Among advantages of objective measures are quick and far less costly, repeatable, etc. Objective measures can be grouped into Full Reference (FR), Reduced Reference (RR), and No Reference (NR). Examples of FR objective measures proposed include Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM), Multi-Scale SSIM (MS-SSIM), Video Quality Metric (VQM), Just-Noticeable Difference (JND), MOtion-based Video Integrity Evaluation (MOVIE), etc., which have different degrees of success in approximating subjective measures, ranging between 70 per cent and 90 per cent. In ITU-T, Video Quality Experts Group (VQEG) is dedicated to developing industry standards on video and multimedia quality assessment. VQEG has developed Rec. J.247 covering FR quality measurement. FR quality metrics have access to an original, unimpaired copy of a video source ((ITU-T, 2008) cited in (Richardson, 2012)). Rec. J.247 lists four objective quality metrics, which are NTT FR, OPTICOM Perceptual VQM, Psytechnics FR, and Yonsei FR.

Objective methods in general proceed as follows. First, the original (reference) and test (impaired or coded) video sequences are compared in the spatial and temporal domains. Then, a set of degradation parameters is calculated such as blurring, blockiness, etc. Lastly, these parameters are combined providing a number as an estimate of subjective quality (Richardson, 2012).

However in many practical applications, a full reference or an original copy of the source video is unavailable making the task of estimating quality more challenging. For instance, the original video may be unavailable at the decoder side as well as for a user-generated video content. In such situations, an NR or RR technique can be applied. NR metrics attempt to estimate the subjective quality based only on characteristics of the decoded video such as artefacts ((Wang *et al.*, 2002) and (Dosselmann and Yang, 2007) cited in (Richardson, 2012)). RR metrics, on the other hand, calculate a quality signature, which is usually a low bitrate side information transmitted to the decoder along with the coded video, and a quality estimate is then derived ((Wang and Simoncelli, 2005) cited in (Richardson, 2012)).

PSNR is a widely-used objective video quality metric in the literature. Its definition is given in (2.3) for an n-bit signal, where the Mean Squared Error (MSE) is calculated as in (2.4), and variables r and c are the vertical and horizontal dimensions of the picture, respectively. O and R are the original and reconstructed pictures, respectively, where an R picture is an image after coding losses.

$$PSNR = 10\log_{10}\frac{(2^n - 1)^2}{MSE} \tag{2.3}$$

$$MSE = \frac{1}{rc}\sum_{i=0}^{r-1}\sum_{j=0}^{c-1}\left(R_{ij} - O_{ij}\right)^2 \tag{2.4}$$

PSNR is originally an image quality assessment (IQA) metric, but it has been widely used in the industry and the research community as a quality metric for assessing the performance of video processing systems (Huynh-Thu and Ghanbari, 2012; Hanhart and Ebrahimi, 2014). It is largely being relied upon in the standardisation of video codecs as a performance indicator, i.e., as a measure of gain in quality of a video codec optimisation tool for a specified target bitrate (Huynh-Thu and Ghanbari, 2012). PSNR has also been used as a comparison tool between video codecs and has been widely used as reference benchmark for comparing objective and subjective video quality assessment (VQA) models against well-established and state-of-the-art compression algorithms (Huynh-Thu and Ghanbari, 2012; Hanhart and Ebrahimi, 2014). The well-known Bjøntegaard model (Section 2.3) uses average PSNR values and bitrate differences between two RD curves when evaluating a video content at different bitrates.

However, PSNR has been in the centre of debate due to it being widely acknowledged to have poor correlation with subjective quality (Hanhart and Ebrahimi, 2014), such as reported for low-resolution (Quarter CIF (QCIF), CIF, and Video Graphics Array (VGA)) up to SD-quality videos (Huynh-Thu and Ghanbari, 2012). Nevertheless, although other objective VQA metrics such as VQM, SSIM, MS-SSIM, and MOVIE have been proposed, these newer models are not being used as frequently as PSNR (Tan *et al.*, 2016) for various reasons. Like PSNR, SSIM and MS-SSIM are also IQA models which do not include temporal distortions (Zeng *et al.*, 2013). On the other hand, VQM and MOVIE are not often being used mainly due to their computational complexities in calculating temporal variations (Tan *et al.*, 2016) and are still not fully successful in capturing and penalising the temporal artefacts (Zeng *et al.*, 2013). The interpretation of VQM and SSIM values has not become a common practice in video coding community (Tan *et al.*, 2016).

Although VQM, SSIM, and MS-SSIM are statistically better than PSNR (Zeng *et al.*, 2013), the rate-distortion (RD) gains obtained from these models may not necessarily be significantly better than PSNR. In (Zeng *et al.*, 2013), it was shown that despite scoring better scores in terms of Pearson Linear Correlation Coefficient (PLCC), Spearman Rank-Order Correlation Coefficient (SRCC), Kendall Rank-Order Correlation Coefficient (KLCC), Mean Absolute Error (MAE), and Root Mean Square (RMS) scores, the average rate-distortion (RD)-gains on videos of HD and Wide VGA (WVGA) (854×480) resolutions obtained by SSIM, MS-SSIM, VQM, and MOVIE are not necessarily much higher than calculated by PSNR. In fact, VQM and MOVIE may provide lower RD-gains than PSNR in spite of requiring substantially larger computational costs (1083× and 7229×, respectively, based on crude execution times) than PSNR. On the other hand, the RD-gains provided by SSIM and MS-SSIM are only slightly better than PSNR in spite of a several-fold increase in computational times (by around 6× and 11×, respectively). Moreover, all the above mentioned objective IQA/VQA models systematically underestimate the RD-gains achievable by subjective scores (Zeng *et al.*, 2013).

Tan *et al.* (2016) chooses PSNR over other objective IQA/VQA metrics to be compared with subjective quality evaluation results. In fact, the poor correlation possessed by PSNR against subjective quality leads to PSNR underestimating the BD-rate gains instead of riskily overestimating, by around 15% on average (Tan *et al.*, 2016), varying between 11% and 18% depending on the video content class, i.e., higher BD-rate gains could actually be achieved if a MOS-based RD curve were used instead. Their results corroborate with their earlier findings that for equal PSNR, the bitrate savings will be 16% lower than for equal MOS when comparing HEVC to Advanced Video Coding (AVC), based on five UHD sequences (Weerakkody *et al.*, 2014). In (Hanhart *et al.*, 2012), PSNR-based BD-rate underestimates the actual bit rate reductions by around 22% based on three UHD sequences, partly due to the saturation effect in perceived quality not captured by

PSNR. Thus, although may not be precise, it is safe to rely on PSNR measurements as the actual video coding performance gain could be much higher.

As previously mentioned, a subjective metric is widely acknowledged as the best form of video quality evaluation as it reflects human perceived quality whereas objective VQA models only provide an estimated quality measure (Li, Ma and Ngan, 2011). Besides being too expensive, extremely time-consuming, infeasible for online manipulations, and impractical for system designs, quality monitoring, etc. (Li, Ma and Ngan, 2011; Hanhart and Ebrahimi, 2014), it can be argued that a subjective model based on MOS is also an estimated quality score as the number of assessors participated in any test conducted is usually fewer than 50 people, and this count is far too low to be representing billions of human viewers across the globe. In fact, frame-level MOS results also do not correlate well with sequence-level MOS results (Zeng *et al.*, 2013). Compared with frame-level quality, temporal artefacts contribute strongly to the overall sequence-level quality (Zeng *et al.*, 2013).

In summary, PSNR can typically be reliably used as an indication of the variation of video quality as long as its limitations are considered, such as full frame rate encoding instead of decimated frame rate, i.e., without the presence of frame skipping or freezing, the saturation effect of the HVS is taken into account (Hanhart *et al.*, 2012), etc. Even though PSNR is acknowledged and widely criticised for its poor correlation with perceived quality, it has clear physical meanings (Wang *et al.*, 2004), can be reliably interpreted (Weerakkody *et al.*, 2014; Tan *et al.*, 2016), and the primary objective quality reference in a video codec development mostly by convention (Zeng *et al.*, 2013). Therefore, in this thesis, PSNR has been used as the primary VQA metric.

2.3 Bjøntegaard Delta PSNR (BD-PSNR) and bitrate (BD-rate)

It is often useful to compare video quality between two different coded videos (e.g., using different codecs) of the same input raw video. These two different coded videos may produce different PSNR values and different bitrates (the rate of a coded video in bits per second). In this case, a simple PSNR comparison is not so useful, because the coded videos also have different bitrates. In this situation, the Bjøntegaard Delta PSNR (BD-PSNR) metric can be applied. This metric is based on a curve fitting of two different Rate-Distortion (R-D) curves (one for each coded video) formed for instance by four PSNR-bitrate points. BD-PSNR represents the difference in PSNR values (in decibel (dB)) usually over the range of four bitrates while BD-rate metric represents the average bitrate difference (in %) normally over the range of four PSNR values. BD-PSNR and BD-rate calculations are described in more detail in references (Bjøntegaard, 2001, 2008).

2.4 Brief History of Video Coding

Video coding has been in existence for the last three decades. Two major organisations that have been instrumental in video technology are the International Telecommunication Union (ITU) and International Organization for Standardization (ISO) / International Electrical Committee (IEC). The history of digital video standardisation can be summarised by revising the digital video standards developed by ITU and ISO/IEC as shown in Fig. 2.4. First, Recommendation (Rec.) 601 (ITU, 2011) for uncompressed digital video representation standard was created in 1982 by the International Radio Consultative Committee (CCIR, today is known as the Radiocommunication sector of ITU (ITU-R)). Rec. 601 has been the bridge connecting the analogue era and digital video world as we know today. Two years later in 1984, the Telecommunication sector of ITU (ITU-T) published the first standard digital video compression technology, Rec. H.120 (ITU, 1993a). However, it was only in 1990 that an adequate compression design could be considered available with the release of ITU-T's Rec. H.261 (ITU, 1993b; Sullivan, 2014).

On the other hand, ISO/IEC produced its first video coding standard, MPEG-1 (ISO, 1993), in 1993. A year later in 1994, ITU-T and ISO/IEC released their first jointly developed video coding standard, Rec. H.262/MPEG-2 Video (ITU, 2012; ISO, 2013a). In 1995, ITU-T produced Rec. H.263 standard (ITU, 2005), and in 1999, ISO/IEC published MPEG-4 Visual standard (ISO, 2004).

ITU-T and ISO/IEC under a partnership known as Joint Video Coding (JVT) completed their second jointly developed video coding standard in May 2003 with the introduction of Rec. H.264/MPEG-4 Part 10 Advanced Video Coding (AVC) (ISO, 2014; ITU, 2014). The following few years witnessed extensions to AVC were subsequently developed, namely Scalable Video Coding (SVC), Multiview Video

Coding (MVC), MPEG's Reconfigurable Video Coding (RVC), and Fully Configurable Video Coding (FCVC) (Richardson, 2012).

Finally in January 2013, the newest standard Rec. H.265/MPEG-H Part 2 High Efficiency Video Coding (HEVC) (ISO, 2013b; ITU, 2013) was approved as a result of the latest partnership between ITU-T and ISO/IEC codenamed Joint Collaborative Team on Video Coding (JCT-VC). The first version of the HEVC standard was finalised in April 2013. The JCT-VC committee then proceeded to develop extensions of HEVC, namely Format Range Extensions (RExt), Scalable HEVC (SHVC) and Screen Content Coding (SCC) (HHI, 2016). Meanwhile, the multiview (MV-HEVC) and 3-D (3D-HEVC) video coding extensions of HEVC were developed by another committee, namely the Joint Collaborative Team on 3D Video Coding Extension Development (JCT-3V). The second version of HEVC which includes the RExt, SHVC and MV-HEVC extensions was concluded in October 2014, and the third version of HEVC comprising the 3D-HEVC extension was completed in February 2015 (HHI, 2016).

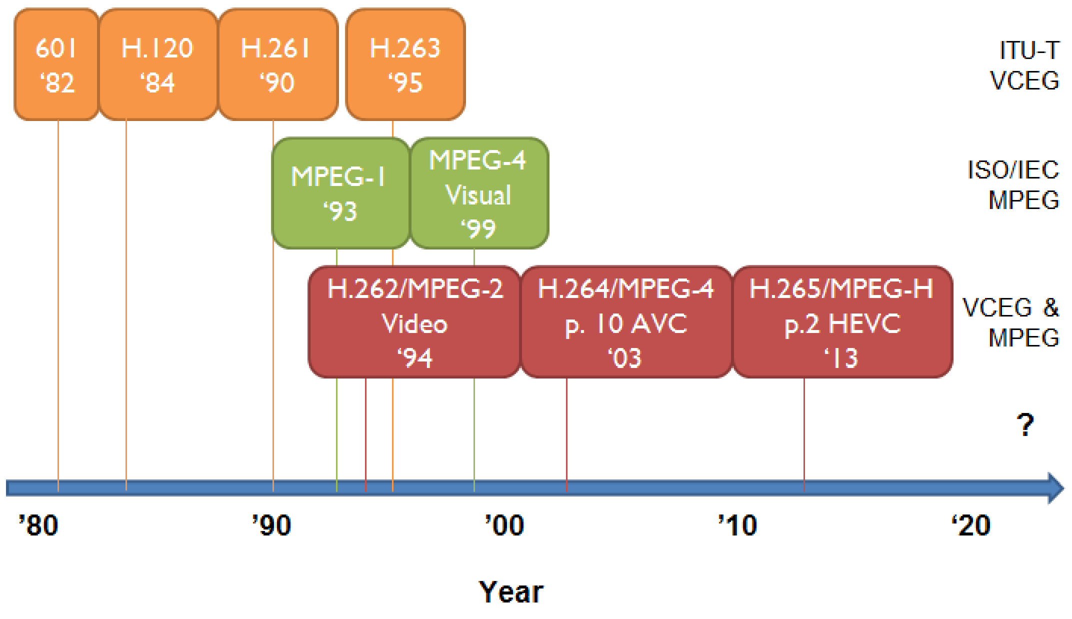


Fig. 2.4 Video coding standards by ITU-T VCEG and ISO/IEC MPEG

2.5 Prediction Structure/Configuration

The encoding and decoding processing order of the pictures in a video sequence is usually different from their arrival order. Thus, it is necessary to differentiate them by means of the bitstream order (or decoding order) and the display order (or the output order). In video coding, there are three types of pictures: I (intra), P (predicted), and B (bi-predicted) pictures. An I picture is a picture that involves only spatial intra-picture prediction and therefore can be independently decoded without needing any prediction information from other decoded pictures. A P picture requires prediction information from another I, P, or B picture to construct every block in the picture. A B picture requires prediction information from two I, P, or B pictures to build its blocks (Tabatabai *et al.*, 2014). Additionally, I and P pictures are also classified as anchor pictures. In general, the pictures from an anchor picture to the last picture just before another anchor picture is termed as a Group of Pictures (GOP), where the size of a GOP is usually fixed in a video sequence such as four, eight, twelve, etc.

To assess the coding performance of a video coding standard, different prediction configurations are defined to simulate different application scenarios. In the case of HEVC and AVC, the following configurations were established:

i. All Intra (AI)
ii. Random Access (RA)
iii. Low Delay with P pictures (LP)
iv. Low Delay with B pictures (LB)

A Quantisation Parameter (QP) in these configurations is altered by means of a ‘QP offset’. A base QP is normally configured for the first picture of the sequence (an I picture), QPI. For the remaining pictures, their QP values can be derived as QP = QPI + QP offset, where QP offset is dependent on the position or temporal ID of the pictures.

2.5.1 All Intra (AI)

As the name suggests, in AI configuration, all pictures are encoded as I pictures. AI is suitable for low delay and high bitrate applications such as storage of high-quality video contents as it does not involve inter-prediction. QP offset is always zero as the QP is retained over the whole sequence (Fig. 2.5) (Tabatabai *et* *al.*, 2014).

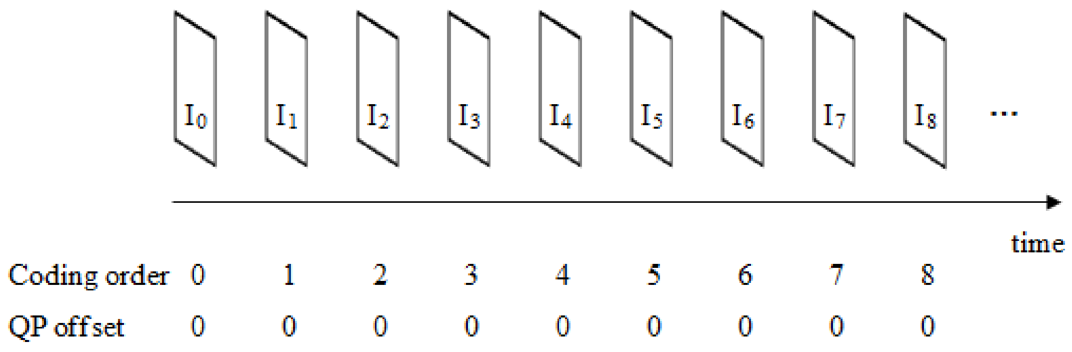


Fig. 2.5 All intra (AI) prediction structure (adapted from (Tabatabai *et al.*, 2014))

2.5.2 Random Access (RA)

RA applies a structure of hierarchical bi-predictive coding (Fig. 2.6). The coding efficiency achieved in this configuration is generally better than the other configurations. However, a larger delay is involved to reorder the pictures. The RA configuration is useful for frame skipping such as fast forward or rewind operations in the entertainment application scenario. To control ease of random access and possible error propagation, I pictures are periodically inserted based on the frame rate of the video sequence. Having I pictures inserted in this manner helps to decode a GOP independently from the previous GOPs (Tabatabai *et al.*, 2014).

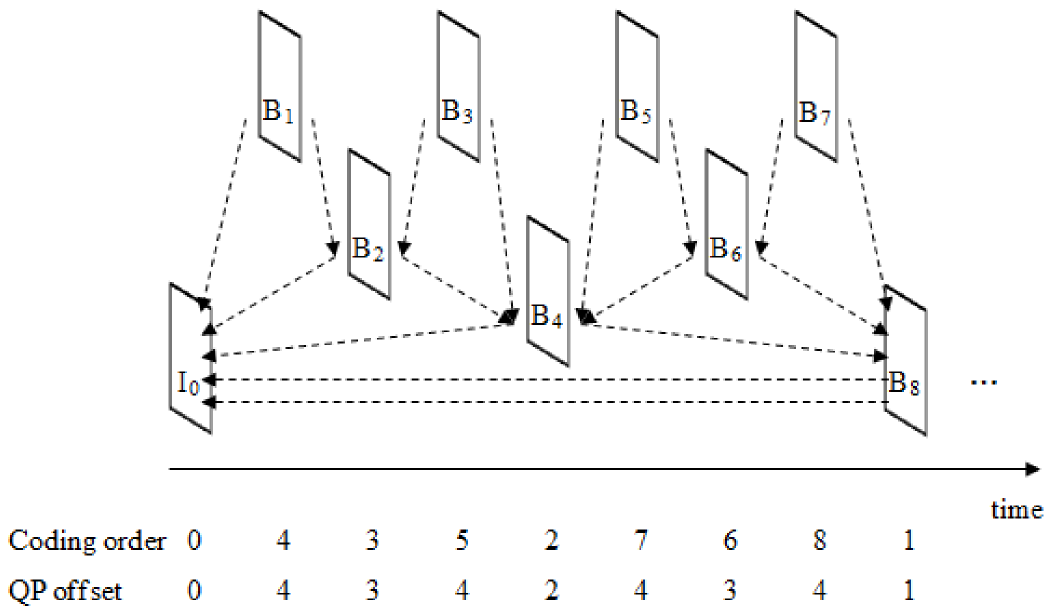


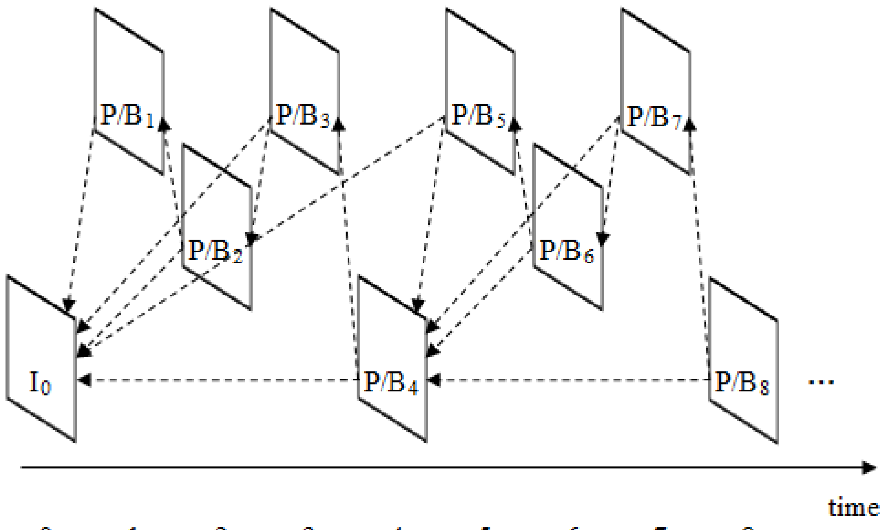
Fig. 2.6 Random access (RA) prediction structure (adapted from (Tabatabai *et al.*, 2014))

2.5.3 Low Delay with P pictures (LP)

In LP, the first picture is encoded as an I picture while all the remaining pictures are encoded as P pictures (Fig. 2.7). Picture reordering is disallowed and predictions only involve past pictures. Due to these conditions, the coding delay may be made small (Tabatabai *et al.*, 2014).

2.5.4 Low Delay with B pictures (LB)

Similar to LP, the first picture in LB configuration is encoded as an I picture while the remaining pictures are encoded as B pictures (Fig. 2.7). Picture reordering is disallowed and predictions only involve B pictures. The coding delay may also be small similar to in LP, but its coding efficiency could be better (Tabatabai *et al.*, 2014).



Coding order	0	1	2	3	4	5	6	7	8
QP offset	0	3	2	3	1	3	2	3	1

Fig. 2.7 Low delay with P pictures and B pictures prediction structure (adapted from (Tabatabai *et al.*, 2014))

2.6 Overview of the High Efficiency Video Coding (HEVC) standard

HEVC follows the typical hybrid video coding scheme comprising block-based intra-/inter-picture prediction, 2-D transform, and entropy coding as applied in previous video coding standards since H.261 (Sullivan *et al.*, 2012; Vanne *et al.*, 2012). The general encoder and decoder models in encoding and decoding an HEVC-compliant bitstream are as illustrated earlier in Fig. 1.2. Each picture from a video sequence is first divided into block regions. The first pictures of every random access point in the video sequence including the very first picture of the whole sequence are encoded with intra-picture prediction, i.e., block-wise spatial predictions within the same picture and independent from other pictures. For the other remaining pictures, inter-picture prediction is normally in place, i.e., temporal predictions from nearby blocks in neighbouring pictures. The inter-picture prediction is performed based on motion data formed by the reference picture and the motion vector (MV) to predict the samples in a block of the current picture. Identical inter-picture prediction signals are generated by both the encoder and the decoder by executing motion compensation (MC) based on the MV and inter-prediction mode decision data, which are transmitted to the decoder as side information (Sullivan *et al.*, 2012).

The differences between the original block samples and predicted block samples of the intra-/inter-picture prediction are known as prediction residuals. These residual data are then integer transformed to produce transform coefficients. These transform coefficients are then scaled, quantised and entropy encoded to form the bitstream to be delivered to the decoder.

The encoder has an in-loop decoding process so that it will continue working on predictions of subsequent blocks and pictures using identical reconstructed blocks and pictures as would be generated by the decoder. In this decoding loop, the quantised transform coefficients are inverse scaled and inverse transformed to produce the approximation of the original residual samples. These residuals are then added to the prediction, and the result of this addition may then be fed into one or two loop filters (deblocking and sample adaptive offset filters) to smooth out artefacts due to block-wise processing and quantisation. The operations continue until the reconstruction of the whole picture completes. This reconstructed picture is then stored in the decoded picture buffer (DPB) to be used for predicting subsequent pictures. The next subsection describes the video coding layer of HEVC in more detail.

2.6.1 Video coding layer of HEVC

The various features in the video coding layer of HEVC can be described as follows (Sullivan *et al.*, 2012):

1) Coding tree unit (CTU) and coding tree block (CTB) structure:

In the coding layer of previous standards like AVC, the basic units were the macro blocks (MB) of 16×16 luma samples and two corresponding 8×8 blocks of chroma samples in the case of 4:2:0 YUV colour sampling. The analogous structure in HEVC is the coding tree unit (CTU) consisting of one luma coding tree block (CTB) and two chroma CTBs along with their associated syntax elements. The size $L \times L$ of a luma CTB can be $L \in \{64, 32, 16\}$, with the larger sizes normally allowing higher compression for large homogeneous regions such as commonly found in high-resolution videos.

2) Coding units (CUs) and coding blocks (CBs):

The CTBs could then be partitioned into smaller blocks using a quadtree-like structure, namely the coding units (CUs) and coding blocks (CBs). A CU is formed by one luma CB and ordinarily two chroma CBs along with their associated syntax. The root of the quadtree at the CTU determines the largest size of the luma CU (LCU) and with the maximum depth for the partitioning of four could yield luma CUs of sizes 32 × 32, 16 × 16, 8 × 8, and 4 × 4. Thus, a CTU may consist of one or more CUs, and every CU can then be further split into prediction units (PUs) and a tree of transform units (TUs).

3) Prediction units (PUs) and prediction blocks (PBs):

Either intra-picture or inter-picture prediction to be performed on a CU is decided at the CU level. A PU partitioning structure also has its root at the CU level, and the luma and chroma CBs can be further partitioned into various PB sizes ranging from 64 × 64 down to 4 × 4.

4) Transform units (TUs) and transform blocks (TBs):

As previously mentioned, the prediction residuals are coded using 2-D transforms. Independently from the PU partitioning, a transform unit (TU) quadtree structure also has its root at the CU level. So, the luma and chroma TBs can either be identical to their corresponding CBs or further partitioned into smaller square sizes among 4 × 4, 8 × 8, 16 × 16, and 32 × 32. These TBs are integer transformed using a scaled approximation of the discrete cosine transform (DCT) commonly found in image and video compressions. Additionally, for the 4 × 4 intra-picture predicted luma TB, an alternative transform based on a scaled approximation of the discrete sine transform (DST) was also defined.

5) Intra-picture prediction:

AVC provides eight directional modes whereas HEVC supports up to 33 directional modes plus DC (flat) and planar (surface fitting) modes. Spatial prediction is performed using decoded boundary samples of adjacent blocks (upper, upper left, or left) as reference data. The most probable intra-picture prediction modes are selected based on previously decoded neighbouring PBs.

6) Motion vector signalling:

HEVC uses advanced motion vector prediction (AMVP), where a number of most probable candidates are included based on adjacent PBs and the selected reference pictures. MVs from spatially or temporally nearby PBs can also be inherited in the merge mode. Additionally, skipped and direct motion inferences have also been improved compared to AVC.

7) Motion compensation:

In AVC, a 6-tap filter was used for half-sample precision followed by a linear interpolation for quarter-sample precision. HEVC uses 7-tap or 8-tap filters for fractional-sample interpolation. Features inherited from AVC include multiple pictures referencing, uni-predictive (one MV) or bi-predictive (two MVs) coding, and weighted prediction where a scaling and an offset are applied to the prediction signals.

8) Quantisation:

Like AVC, HEVC performs uniform reconstruction quantisation (URQ) with quantisation scaling matrices for the four supported TB sizes.

9) Entropy coding:

AVC supports context adaptive variable length coding (CAVLC) in its Main profile and context adaptive binary arithmetic coding (CABAC) in the High profile. In HEVC, a much improved CABAC is defined to enhance its throughput speed, compression performance, and its context memory requirements.

10) In-loop deblocking filtering:

HEVC operates a simplified deblocking filter relative to AVC within the inter-picture prediction loop. The simplification made was in terms of its decision-making, filtering processes, and designed to be friendlier to parallel processing.

11) Sample adaptive offset (SAO):

An SAO filter follows the deblocking filter in the inter-picture prediction loop of HEVC to better reconstruct the original prediction residuals prior to storing in the DPB. The operation is performed on a region basis based on look-up tables. The SAO filtering either adds no offset, a band offset, or an edge offset.

2.6.2 Profiles, Levels, and Tiers in HEVC

Like previous standards, HEVC was developed to support a wide range of video applications by defining a large pool of video coding algorithms. Most of these applications, however, do not require using all the coding capabilities established in HEVC. Having a limitation of the coding tools or algorithms supported by an HEVC codec specifically designed for certain applications provides the benefits of reduced computing and memory requirements. Therefore, profiles were defined in HEVC to support different sets of coding tools of the standard. In the first version of HEVC, three coding profiles were defined (Sullivan, 2014):

1) Main profile: for typical applications used by most consumers, supporting videos of 8-bit per sample and 4:2:0 YUV colour sampling.

2) Main Still Picture profile: a subset of the Main profile. This profile is for snapshots from video sequences or still photography from cameras.

3) Main 10 profile: a superset of the Main profile. This profile supports up to 10-bit per sample videos, providing higher brightness dynamic range, larger colour-gamut content, and increased fidelity colour representations to reduce rounding errors and contouring artefacts.

A profile conforming decoder must be able to support all features in that profile. Within a profile, there are also different maximum or minimum requirements expected by different devices, video resolutions, etc. Thus, levels were defined to restrict the maximum number of luma samples, maximum sample rate, maximum bitrate, minimum compression ratio, and minimum DPB and coded picture buffer (CPB) sizes, which stores compressed data prior to decoding for data flow management purposes (Sullivan *et al.*, 2012). The first version of HEVC defined 13 levels, supporting small picture resolutions like Quarter CIF (QCIF, 176 × 144) and very large resolutions of up to 8K UHD (7680 × 4320). Table 2.3 summarises the 13 levels defined in the Main profile.

Among the parameters distinguishing a level from the others, some applications had requirements differing only in the maximum bitrate and CPB capabilities. Therefore, two tiers were defined each for the top eight levels: a Main Tier adequate for most applications and a High Tier particularly for most demanding applications. An HEVC decoder conforming to a certain tier and level is expected to be able to decode all bitstreams that conform to that tier and the lower tier of the same level, as well as all levels below it (Sullivan *et al.*, 2012).

Table 2.3 Supported levels in Main profile of HEVC (Sullivan *et al.*, 2012)

Level	**Max luma samples**	**Max luma sample rate (samples/s)**	**Max bitrate (1000 bits/s)**		**Min comp. ratio**
			Main Tier	**High Tier**	
1	36 864	552 960	128	-	2
2	122 880	3 686 400	1500	-	2
2.1	245 760	7 372 800	3000	-	2
3	552 960	16 588 800	6000	-	2
3.1	983 040	33 177 600	10 000	-	2
4	2 228 224	66 846 720	12 000	30 000	4
4.1	2 228 224	133 693 440	20 000	50 000	4
5	8 912 896	267 386 880	25 000	100 000	6
5.1	8 912 896	534 773 760	40 000	160 000	8
5.2	8 912 896	1 069 547 520	60 000	240 000	8
6	35 651 584	1 069 547 520	60 000	240 000	8
6.1	35 651 584	2 139 095 040	120 000	480 000	8
6.2	35 651 584	4 278 190 080	240 000	800 000	6

2.7 Summary

A few fundamental concepts on video coding were briefly introduced in this chapter to facilitate a better understanding of the research work done in this thesis. These concepts include an overview of the HEVC standard and a walkthrough of HEVC encoder. The next chapter is dedicated to describing the forward transform, intermediate scaling, and quantisation defined in the HEVC standard, as these operations are the main subjects of this research.

Chapter 3
HEVC Forward Transform, Intermediate Scaling, and Quantisation

Abstract This chapter describes the forward transform, intermediate scaling, and quantisation operations specified in the HEVC standard. The content of this chapter was heavily extracted from (Budagavi *et al.*, 2013; Budagavi, Fuldseth and Bjøntegaard, 2014) to serve as the foundation of the research work carried out in this thesis.

3.1 Introduction

A typical block-based hybrid video coding system is made of two components: an encoder and a decoder, as illustrated in Fig. 3.1. At the encoder, a picture is first partitioned into square or rectangular blocks of pixels/samples depending on the spatial characteristics of the picture. Each block then subtracts a neighbouring block in the same picture (intra-prediction mode exploiting spatial redundancies) or another block from a neighbouring picture (inter-prediction mode exploiting temporal redundancies), resulting in a prediction residual signal. This residual signal can be further divided into square blocks of size $N \times N$, where $N = 2^M$ and M is an integer. A separable two-dimensional (2-D) $N \times N$ forward transform is then performed on every residual block (U), which can equally be realised by consecutively applying a one-dimensional (1-D) N-point transform to every row and column. Up to here, the process is lossless or near-lossless depending on the adopted transform precision. Then, the resulting transform coefficients (*coeff*) are input to a quantisation (dividing by a quantisation step (*Qstep*) and necessary rounding) producing quantised transform coefficients (*level*). The quantisation is a lossy operation. These quantised transformed coefficients are then scanned and entropy encoded by exploiting statistical redundancies of the scanned *level* to be included in the final bitstream (Budagavi, Fuldseth and Bjøntegaard, 2014).

At the decoder, the encoding process is reversed. First, the received bitstream is entropy decoded to extract the quantised transform coefficients (*level*). Then, these coefficients are de-quantised (multiplying by *Qstep*) to obtain the de-quantised transform coefficients ($coeff_Q$). After that, a separable 2-D $N \times N$ inverse transform is performed on $coeff_Q$ to obtain the quantised residual samples. Finally, these quantised samples are added to the intra-/inter-prediction samples to reconstruct the original block (Budagavi, Fuldseth and Bjøntegaard, 2014).

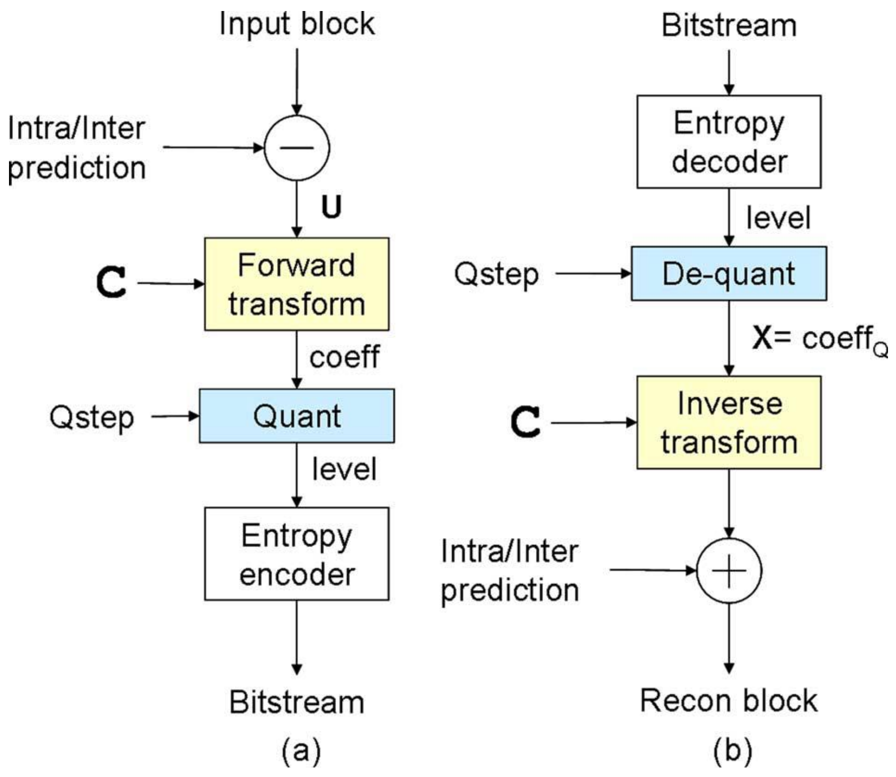


Fig. 3.1 Hybrid block-based video coding comprising (a) an encoder and (b) a decoder (**C** is the transform matrix and *Qstep* is the quantisation step) (Budagavi *et al.*, 2013; Budagavi, Fuldseth and Bjøntegaard, 2014)

3.2 HEVC Transforms

The HEVC standard defines two transform operations: a core transform and an alternative transform. The core transform is based on Discrete Cosine Transform (DCT) type II introduced by Ahmed, Natarajan, and Rao (Ahmed, Natarajan and Rao, 1974) and applicable to all luminance and chrominance TU sizes defined in the standard, as well as intra and inter PUs. On the other hand, the alternative transform is based on Discrete Sine Transform (DST) type VII and applicable to only 4 × 4 luminance intra-predicted residual blocks (Budagavi, Fuldseth and Bjøntegaard, 2014).

For input residual samples x_c, the N-point 1-D transform coefficients y_{rc} can be expressed as

$$y_{rc} = \sum_{c=0}^{N-1} t_{rc} x_c$$

where $r = 0, 1, \ldots, N-1$. For DCT type II, elements $t_{rc} = \boldsymbol{c}_{rc}$ and defined as

$$\boldsymbol{c}_{rc} = \frac{P}{\sqrt{N}} \, cos \left[\frac{(2c+1)\pi r}{2N} \right]$$

where $r, c = 0, 1, \ldots, N-1$, and P is equal to 1 for r = 0 and $\sqrt{2}$ for r > 0. Moreover, the basis vectors $\boldsymbol{c}_r$ of the DCT are defined as $\boldsymbol{c}_r = [\boldsymbol{c}_{r0}, \boldsymbol{c}_{r1}, \ldots, \boldsymbol{c}_{r(N-1)}]$ where $r = 0, 1, \ldots, N-1$ (Budagavi, Fuldseth and Bjøntegaard, 2014). The DCT is advantageous in image and video compression due to its favourable properties as listed in Table 3.1 (Rao and Yip (1990) cited in (Budagavi, Fuldseth and Bjøntegaard, 2014)).

On the other hand, for DST type VII, elements $t_{rc} = s_{rc}$ where $r, c = 0, 1, \ldots, N-1$ and defined as

$$s_{rc} = \frac{2}{\sqrt{2N+1}} \, sin \left(\frac{(2r+1)(c+1)\pi}{2N+1} \right)$$

Table 3.1 Several properties of DCT (Rao and Yip (1990) cited in (Budagavi, Fuldseth and Bjøntegaard, 2014))

No.	Property	Description	Benefit
i.	Orthogonality	Its basis vectors are orthogonal, i.e., $\boldsymbol{c}_r\boldsymbol{c}_c = 0$ for $r \neq c$ and $\boldsymbol{c}_c = \boldsymbol{c}_r^{\mathrm{T}}$ (superscript $^{\mathrm{T}}$ denotes the transpose operation).	De-correlate the transform coefficients.
ii.	Normal	Its basis vectors have equal norm, i.e., $\boldsymbol{c}_r\boldsymbol{c}_c = 1$ for $r = c$ and $r = 0, 1, \ldots, N - 1$.	Simplify the quantisation/de-quantisation process.
iii.	Energy compaction	Its basis vectors have good energy compaction.	Concentrate the energy towards the low-frequency region near the top left corner of a 2-D block.
iv.	Embedded elements	A DCT matrix of size $2^M \times 2^M$ is a subset of a DCT matrix of size $2^{M+1} \times 2^{M+1}$, which are equal to the left half of the even basis vectors of the larger matrix.	Reduce hardware costs as the involved multipliers can be shared by different matrix sizes.
v.	Small quantity of elements	For a DCT matrix of size $2^M \times 2^M$, the number of unique elements is equal to $2^M - 1$.	Low implementation costs.

vi.	Symmetry/Anti-symmetry	The even basis vectors are symmetric, while the odd basis vectors are anti-symmetric.	Lower the number of arithmetic operations.
vii.	Trigonometric relationships	DCT matrix coefficients have some trigonometric relationships.	The number of arithmetic operations can be further reduced by employing algorithms such as Chen's fast factorisation.
viii.	Separable	2-D $N \times N$ DCT are executable as two separate 1-D N-point DCTs on the rows and columns.	The same DCT matrix is reusable for the second transform operation.

3.3 Basis Vectors of HEVC Core and Alternative Transforms

Both the core and alternative transform matrices of HEVC are scaled integer approximations of the DCT or DST matrix. An obvious advantage of using a fixed precision instead of floating values is reduced computational complexity. Another benefit is that the transform elements can be specified explicitly in the standard instead of implementation-dependent. This eliminates encoder-decoder mismatch due to different developers implementing the inverse DCT/DST transform operations using slightly different floating-point precisions.

However, one disadvantage associated with approximated transform elements is that some of the useful properties previously mentioned in Table 3.1 are compromised. Therefore, a trade-off was made between reducing the computational complexity and preserving some of the transform properties (Budagavi, Fuldseth and Bjøntegaard, 2014).

HEVC defines four $N \times N$ core transform matrices, where N = 4, 8, 16, and 32. The elements of the largest core transform matrix, h_{rc}^{32}, was derived by approximating the scaled and rounded DCT elements after applying a scaling factor, $\alpha = 2^{6+5/2} = 64\sqrt{32} \approx 362.04$ to $\boldsymbol{c}_{rc}$ where *r, c* = 0, 1, …, 31, i.e.,

$$h_{rc}^{32} \approx round(\alpha \boldsymbol{c}_{rc})$$

It is worth noting that $h_{rc}^{32} \neq round(\alpha \boldsymbol{c}_{rc})$ as a hand-tuning was performed to some of the scaled and rounded DCT elements to achieve an acceptable balance between a few DCT properties (Budagavi, Fuldseth and Bjøntegaard, 2014) (shown later in Table 4.4).

Fig. 3.2 provides the left half of the 32 × 32 core forward transform matrix. The right half can easily be derived by applying the symmetry/anti-symmetry property of the basis vectors. The inverse core transform matrix of HEVC is the transpose of Fig. 3.2 (and the associated right half). The elements of the smaller transform matrices, h_{rc}^{N}, where *N* = 4, 8, 16 and *r, c* = 0, 1, …, *N* – 1, can be obtained from h_{rc}^{32} as (Budagavi, Fuldseth and Bjøntegaard, 2014)

$$h_{rc}^{N} = h_{r(32/N),c}^{32} \tag{3.1}$$

As can be seen from Fig. 3.2, the *N* × *N* core transform matrix is embedded in the 2*N* × 2*N* matrix (property iv in Table 3.1). For instance, by using (3.1) and Fig. 3.2, the 4 × 4 core transform matrix, H^4 can be obtained as (Budagavi, Fuldseth and Bjøntegaard, 2014)

$$\mathrm{H}^4 = \begin{bmatrix} h_{0,0}^{32} & h_{0,1}^{32} & h_{0,2}^{32} & h_{0,3}^{32} \\ h_{8,0}^{32} & h_{8,1}^{32} & h_{8,2}^{32} & h_{8,3}^{32} \\ h_{16,0}^{32} & h_{16,1}^{32} & h_{16,2}^{32} & h_{16,3}^{32} \\ h_{24,0}^{32} & h_{24,1}^{32} & h_{24,2}^{32} & h_{24,3}^{32} \end{bmatrix} = \begin{bmatrix} 64 & 64 & 64 & 64 \\ 83 & 36 & -36 & -83 \\ 64 & -64 & -64 & 64 \\ 36 & -83 & 83 & -36 \end{bmatrix}$$

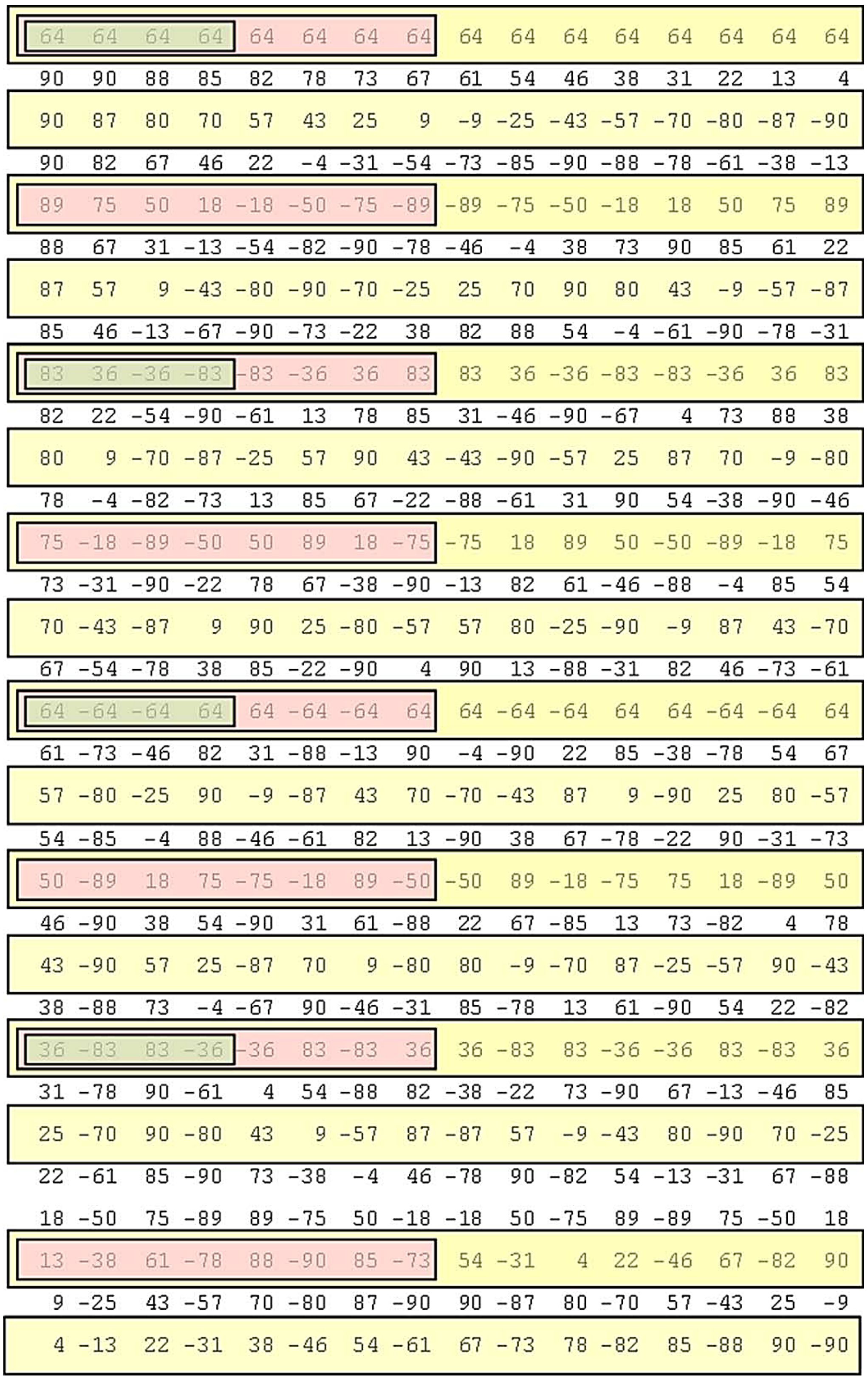
In addition, due to the unique numbers property and symmetry property inherited from DCT (properties v and vi in Table 3.1), H^4 can as well be written as

$$\mathrm{H}^4 = \begin{bmatrix} h_{16,0}^{32} & h_{16,0}^{32} & h_{16,0}^{32} & h_{16,0}^{32} \\ h_{8,0}^{32} & h_{24,0}^{32} & -h_{24,0}^{32} & -h_{8,0}^{32} \\ h_{16,0}^{32} & -h_{16,0}^{32} & -h_{16,0}^{32} & h_{16,0}^{32} \\ h_{24,0}^{32} & -h_{8,0}^{32} & h_{8,0}^{32} & -h_{24,0}^{32} \end{bmatrix} \tag{3.2}$$

It is worth noting that (3.2) only involves elements from the first column (c = 0) of Fig. 3.2, and applying this realisation greatly simplifies the implementation. Furthermore for notational simplicity, the elements $h_{i,0}^{32}$ of (3.2) will be denoted by h_i. Using this new notation, (3.2) becomes

$$\mathrm{H}^4 = \begin{bmatrix} h_{16} & h_{16} & h_{16} & h_{16} \\ h_8 & h_{24} & -h_{24} & -h_8 \\ h_{16} & -h_{16} & -h_{16} & h_{16} \\ h_{24} & -h_8 & h_8 & -h_{24} \end{bmatrix} \tag{3.3}$$

The corresponding inverse transform matrix is $\mathrm{H}^{4\mathrm{T}}$.



64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64
90	90	88	85	82	78	73	67	61	54	46	38	31	22	13	4
90	87	80	70	57	43	25	9	-9	-25	-43	-57	-70	-80	-87	-90
90	82	67	46	22	-4	-31	-54	-73	-85	-90	-88	-78	-61	-38	-13
89	75	50	18	-18	-50	-75	-89	-89	-75	-50	-18	18	50	75	89
88	67	31	-13	-54	-82	-90	-78	-46	-4	38	73	90	85	61	22
87	57	9	-43	-80	-90	-70	-25	25	70	90	80	43	-9	-57	-87
85	46	-13	-67	-90	-73	-22	38	82	88	54	-4	-61	-90	-78	-31
83	36	-36	-83	-83	-36	36	83	83	36	-36	-83	-83	-36	36	83
82	22	-54	-90	-61	13	78	85	31	-46	-90	-67	4	73	88	38
80	9	-70	-87	-25	57	90	43	-43	-90	-57	25	87	70	-9	-80
78	-4	-82	-73	13	85	67	-22	-88	-61	31	90	54	-38	-90	-46
75	-18	-89	-50	50	89	18	-75	-75	18	89	50	-50	-89	-18	75
73	-31	-90	-22	78	67	-38	-90	-13	82	61	-46	-88	-4	85	54
70	-43	-87	9	90	25	-80	-57	57	80	-25	-90	-9	87	43	-70
67	-54	-78	38	85	-22	-90	4	90	13	-88	-31	82	46	-73	-61
64	-64	-64	64	64	-64	-64	64	64	-64	-64	64	64	-64	-64	64
61	-73	-46	82	31	-88	-13	90	-4	-90	22	85	-38	-78	54	67
57	-80	-25	90	-9	-87	43	70	-70	-43	87	9	-90	25	80	-57
54	-85	-4	88	-46	-61	82	13	-90	38	67	-78	-22	90	-31	-73
50	-89	18	75	-75	-18	89	-50	-50	89	-18	-75	75	18	-89	50
46	-90	38	54	-90	31	61	-88	22	67	-85	13	73	-82	4	78
43	-90	57	25	-87	70	9	-80	80	-9	-70	87	-25	-57	90	-43
38	-88	73	-4	-67	90	-46	-31	85	-78	13	61	-90	54	22	-82
36	-83	83	-36	-36	83	-83	36	36	-83	83	-36	-36	83	-83	36
31	-78	90	-61	4	54	-88	82	-38	-22	73	-90	67	-13	-46	85
25	-70	90	-80	43	9	-57	87	-87	57	-9	-43	80	-90	70	-25
22	-61	85	-90	73	-38	-4	46	-78	90	-82	54	-13	-31	67	-88
18	-50	75	-89	89	-75	50	-18	-18	50	-75	89	-89	75	-50	18
13	-38	61	-78	88	-90	85	-73	54	-31	4	22	-46	67	-82	90
9	-25	43	-57	70	-80	87	-90	90	-87	80	-70	57	-43	25	-9
4	-13	22	-31	38	-46	54	-61	67	-73	78	-82	85	-88	90	-90

Fig. 3.2 Left half of the 32-point forward core transform matrix with embedded 4-point (green blocks), 8-point (pink blocks), and 16-point (yellow blocks) forward transform matrices (Budagavi, Fuldseth and Bjøntegaard, 2014)

On the other hand, the elements of the alternative matrix of HEVC, a^4_{rc}, was derived by scaling the DST elements, s_{rc}, by $\beta = 2^7 = 128$ and rounding to the nearest integer, i.e.,

$$a^4_{rc} = round(\beta s_{rc})$$

The alternative forward transform matrix, A^4, is therefore given by (Budagavi, Fuldseth and Bjøntegaard, 2014)

$$A^4 = \begin{bmatrix} 29 & 55 & 74 & 84 \\ 74 & 74 & 0 & -74 \\ 84 & -29 & -74 & 55 \\ 55 & -84 & 74 & -29 \end{bmatrix}$$

The corresponding inverse transform matrix is A^{4T}.

The most important transform coefficient is the DC coefficient, which is at coordinate (0, 0) of a 2-D plane. In terms of a 2-D image/picture plane, the DC transform coefficient is at the top left corner of a block, indicating the sample energy at a frequency of zero Hertz (0 Hz). The Human Visual System (HVS) is known to be more sensitive to low frequencies than high frequencies (Richardson, 2012), and the HVS is most sensitive to the DC value. This DC value is the result of multiplying the samples with the first basis vector of a transform. Therefore, the first basis vector of a transform is crucial in determining the DC coefficient.

The DST matrix is more suitable for an intra-prediction block as the residuals are smaller near the top and left boundaries and larger towards the bottom and right boundaries. In contrast to the DCT matrix which has a flat first row, the elements of the first row of a DST matrix increase from left to right making it better in modelling the spatial behaviour of the intra-prediction residuals and providing around 1% bit-rate reduction (Saxena and Fernandes, 2013). However, the DST matrix was only adopted for the 4×4 intra PUs as the additional coding gain using larger DST transforms was insignificant and the computational complexity of an $N \times N$ DST is higher than a DCT of the same size.

3.4 Complexity Analysis

This section describes the even–odd decomposition technique, multiplier-free approach, and Multiple-Constant Multiplication (MCM) technique, which are useful in reducing the computational complexity.

3.4.1 Even–Odd Decomposition

For an N-point input vector, the number of arithmetic operations for a 1-D forward/inverse transform via direct matrix multiplication is N^2 multiplications and $N(N - 1)$ additions (including subtractions). For an $N \times N$ input block, the complexity of a 1-D transform becomes N^3 multiplications and $N^2(N - 1)$ additions. The separable property of transforms such as DCT enables a 2-D transform to be implemented via two 1-D transforms with a transpose operation between them. Thus for a 2-D transform of an $N \times N$ input block, the complexity is $2N^3$ multiplications and $2N^2(N - 1)$ additions.

However, the inheritance of the symmetry property of DCT basis vectors (property vi in Table 3.1) facilitates the transform complexity to be significantly reduced. The technique that utilises this symmetry property was referred to as the even–odd decomposition (known as partial butterfly during HEVC development). A 1-D forward transform using the even–odd decomposition technique comprises the following three steps (adapted from (Budagavi, Fuldseth and Bjøntegaard, 2014)):

1. Add/subtract input data to generate an N-point intermediate vector.
2. Calculate the even part using the $N/2 \times N/2$ subset matrix formed by the even rows of the $N \times N$ transform matrix.
3. Calculate the odd part using the $N/2 \times N/2$ subset matrix formed by the odd rows of the $N \times N$ transform matrix.

For the inverse transform, the add/subtract operation is performed after the even and odd parts calculation. This technique is best demonstrated using the 4-point and 8-point transforms. Higher order transforms such as the 16-point, 32-point or higher apply the same routine.

The forward 4-point transform can be obtained as $\mathrm{Y}_c^4 = \mathrm{H}^4\mathrm{X}_c^4$, where $\mathrm{X}_c^4 = [\mathrm{X}_0, \mathrm{X}_1, \mathrm{X}_2, \mathrm{X}_3]^\mathrm{T}$ is a 4-point input vector, $\mathrm{Y}_c^4 = [\mathrm{Y}_0, \mathrm{Y}_1, \mathrm{Y}_2, \mathrm{Y}_3]^\mathrm{T}$ is the 4-point output of the transform, and H^4 is as given in (3.3). Thus, the 4-point forward transform using the even–odd decomposition is as provided by (3.4)–(3.6):

Add/sub part:

$$[\mathrm{W}_0, \mathrm{W}_1, \mathrm{W}_2, \mathrm{W}_3]^{\mathrm{T}} = [\mathrm{X}_0 + \mathrm{X}_3,\ \mathrm{X}_0 - \mathrm{X}_3,\ \mathrm{X}_1 + \mathrm{X}_2,\ \mathrm{X}_1 - \mathrm{X}_2]^{\mathrm{T}} \tag{3.4}$$

Even part:

$$\begin{bmatrix} \mathrm{E}_0 \\ \mathrm{E}_1 \end{bmatrix} = \begin{bmatrix} h_{16} & h_{16} \\ h_{16} & -h_{16} \end{bmatrix} \begin{bmatrix} \mathrm{W}_0 \\ \mathrm{W}_2 \end{bmatrix} = \begin{bmatrix} h_{16}\mathrm{W}_0 + h_{16}\mathrm{W}_2 \\ h_{16}\mathrm{W}_0 - h_{16}\mathrm{W}_2 \end{bmatrix} \tag{3.5}$$

Odd part:

$$\begin{bmatrix} \mathrm{O}_0 \\ \mathrm{O}_1 \end{bmatrix} = \begin{bmatrix} h_8 & h_{24} \\ h_{24} & -h_8 \end{bmatrix} \begin{bmatrix} \mathrm{W}_1 \\ \mathrm{W}_3 \end{bmatrix} = \begin{bmatrix} h_8\mathrm{W}_1 + h_{24}\mathrm{W}_3 \\ h_{24}\mathrm{W}_1 - h_8\mathrm{W}_3 \end{bmatrix} \tag{3.6}$$

Then, the output is $\mathrm{Y}_c^4 = [\mathrm{E}_0, \mathrm{O}_0, \mathrm{E}_1, \mathrm{O}_1]^{\mathrm{T}}$.

The direct 1-D 4-point transform $\mathrm{Y}_c^4 = \mathrm{H}^4\mathrm{X}_c^4$ would incur $4^2 = 16$ multiplications and 4(4 – 1) = 12 additions. The 2-D transform will cost $2(4)^3 = 128$ multiplications and $2(4^2)(4 - 1) = 96$ additions. On the other hand, a 1-D transform using the even–odd decomposition involves four additions for the add/sub part (3.4), two multiplications and two additions for the even part (3.5), and four multiplications and two additions for the odd part (3.6), i.e., six multiplications and eight additions in total. The corresponding separable 2-D transform will cost 2 × 4 × 6 = 48 multiplications and 2 × 4 × 8 = 64 additions, resulting in 62.5% reductions in the number of multiplications and 33.3% for the additions in comparison with the direct matrix multiplication in the 4 × 4 case (Budagavi, Fuldseth and Bjøntegaard, 2014).

Similarly, let $\mathrm{X}_c^8 = [\mathrm{X}_0, \mathrm{X}_1, \dots, \mathrm{X}_7]^{\mathrm{T}}$ be an 8-point input vector and $\mathrm{Y}_c^8 = [\mathrm{Y}_0, \mathrm{Y}_1, \dots, \mathrm{Y}_7]^{\mathrm{T}}$ be the 8-point output of the transform. The forward 8-point transform can then be attained as $\mathrm{Y}_c^8 = \mathrm{H}^8\mathrm{X}_c^8$, where H^8 is given by (Budagavi, Fuldseth and Bjøntegaard, 2014)

$$\mathrm{H}^8 = \begin{bmatrix} h_{16} & h_{16} & h_{16} & h_{16} & h_{16} & h_{16} & h_{16} & h_{16} \\ h_4 & h_{12} & h_{20} & h_{28} & -h_{28} & -h_{20} & -h_{12} & -h_4 \\ h_8 & h_{24} & -h_{24} & -h_8 & -h_8 & -h_{24} & h_{24} & h_8 \\ h_{12} & -h_{28} & -h_4 & -h_{20} & h_{20} & h_4 & h_{28} & -h_{12} \\ h_{16} & -h_{16} & -h_{16} & h_{16} & h_{16} & -h_{16} & -h_{16} & h_{16} \\ h_{20} & -h_4 & h_{28} & h_{12} & -h_{12} & -h_{28} & h_4 & -h_{20} \\ h_{24} & -h_8 & h_8 & -h_{24} & -h_{24} & h_8 & -h_8 & h_{24} \\ h_{28} & -h_{20} & h_{12} & -h_4 & h_4 & -h_{12} & h_{20} & -h_{28} \end{bmatrix} \tag{3.7}$$

The 8-point forward transform using the even–odd decomposition is as provided by (3.8)–(3.10):

Add/sub part:

$$\begin{aligned}&[\mathrm{W}_0, \mathrm{W}_1, \mathrm{W}_2, \mathrm{W}_3, \mathrm{W}_4, \mathrm{W}_5, \mathrm{W}_6, \mathrm{W}_7]^{\mathrm{T}} \\ &= [\mathrm{X}_0 + \mathrm{X}_7, \mathrm{X}_0 - \mathrm{X}_7, \mathrm{X}_1 + \mathrm{X}_6, \mathrm{X}_1 - \mathrm{X}_6, \mathrm{X}_2 + \mathrm{X}_5, \mathrm{X}_2 - \mathrm{X}_5, \mathrm{X}_3 + \mathrm{X}_4, \mathrm{X}_3 - \mathrm{X}_4]^{\mathrm{T}}\end{aligned} \tag{3.8}$$

Even part:

$$\begin{bmatrix} \mathrm{E}_0 \\ \mathrm{E}_1 \\ \mathrm{E}_2 \\ \mathrm{E}_3 \end{bmatrix} = \begin{bmatrix} h_{16} & h_{16} & h_{16} & h_{16} \\ h_8 & h_{24} & -h_{24} & -h_8 \\ h_{16} & -h_{16} & -h_{16} & h_{16} \\ h_{24} & -h_8 & h_8 & -h_{24} \end{bmatrix} \begin{bmatrix} \mathrm{W}_0 \\ \mathrm{W}_2 \\ \mathrm{W}_4 \\ \mathrm{W}_6 \end{bmatrix} = \mathrm{H}^4 \begin{bmatrix} \mathrm{W}_0 \\ \mathrm{W}_2 \\ \mathrm{W}_4 \\ \mathrm{W}_6 \end{bmatrix} \tag{3.9}$$

Odd part:

$$\begin{bmatrix} \mathrm{O}_0 \\ \mathrm{O}_1 \\ \mathrm{O}_2 \\ \mathrm{O}_3 \end{bmatrix} = \begin{bmatrix} h_4 & h_{12} & h_{20} & h_{28} \\ h_{12} & -h_{28} & -h_4 & -h_{20} \\ h_{20} & -h_4 & h_{28} & h_{12} \\ h_{28} & -h_{20} & h_{12} & -h_4 \end{bmatrix} \begin{bmatrix} \mathrm{W}_1 \\ \mathrm{W}_3 \\ \mathrm{W}_5 \\ \mathrm{W}_7 \end{bmatrix} \tag{3.10}$$

The output is $\mathrm{Y}_c^8 = [\mathrm{E}_0, \mathrm{O}_0, \mathrm{E}_1, \mathrm{O}_1, \mathrm{E}_2, \mathrm{O}_2, \mathrm{E}_3, \mathrm{O}_3]^T$.

The direct 1-D 8-point transform $\mathrm{Y}_c^8 = \mathrm{H}^8\mathrm{X}_c^8$ would incur 8^2 = 64 multiplications and 8(8 – 1) = 56 additions. The 2-D transform will cost $2(8)^3$ = 1024 multiplications and $2(8^2)(8 – 1)$ = 896 additions. Using the even–odd decomposition, it is worth noting that the even part (3.9) can be implemented using the 4-point (*N*/2-point) even–odd decomposition (3.4)–(3.6). So, for 1-D 8-point transform using this technique, the add/sub part involves eight additions (3.8), the even part costs the same as the 4-point transform, i.e., six multiplications and eight additions, while the odd part requires 16 multiplications and 12 additions (3.10). Thus, the total arithmetic complexity of 1-D 8-point transform using the even–odd decomposition is

6 + 16 = 22 multiplications and 8 + 8 + 12 = 28 additions. The corresponding 2-D transform will require 2 × 8 × 22 = 352 multiplications and 2 × 8 × 28 = 448 additions (Budagavi, Fuldseth and Bjøntegaard, 2014), i.e., giving 65.6% savings in the number of multiplications and 50% in additions, relative to the direct matrix multiplication.

The calculation of the 4-point and 8-point forward transform computational complexity can be applied in the same manner to the forward/inverse transform of larger sizes. The total complexity of multiplications and additions for the 1-D *N*-point, 1-D *N* × *N*, and 2-D *N* × *N* transforms using the even–odd decomposition, in general, can be shown to be (3.11)–(3.16) (Budagavi, Fuldseth and Bjøntegaard, 2014) and summarised in Tables 3.2–3.4.

$$O_{mult,1D,N} = 1 + \sum_{k=1}^{\log_2 N} 2^{2k-2} = O_{mult,1D,N/2} + 2^{2(\log_2 N)-2} \tag{3.11}$$

$$\begin{aligned} O_{add,1D,N} &= \sum_{k=1}^{\log_2 N} 2^{k-1}(2^{k-1}+1) \\ &= O_{add,1D,N/2} + 2^{(\log_2 N)-1}\left(2^{(\log_2 N)-1}+1\right) \end{aligned} \tag{3.12}$$

$$O_{mult,1D,N\times N} = N\left(1 + \sum_{k=1}^{\log_2 N} 2^{2k-2}\right) = N\left(O_{mult,1D,N}\right) \tag{3.13}$$

$$O_{add,1D,N\times N} = N\left(\sum_{k=1}^{\log_2 N} 2^{k-1}(2^{k-1}+1)\right) = N\left(O_{add,1D,N}\right) \tag{3.14}$$

$$O_{mult,2D} = 2N\left(1 + \sum_{k=1}^{\log_2 N} 2^{2k-2}\right) = 2N\left(O_{mult,1D,N}\right) \tag{3.15}$$

$$O_{add,2D} = 2N\left(\sum_{k=1}^{\log_2 N} 2^{k-1}(2^{k-1}+1)\right) = 2N\left(O_{add,1D,N}\right) \tag{3.16}$$

Table 3.2 Computational complexity in 1-D *N*-point HEVC core transforms

Size	**Technique**			
	Matrix Multiplication		**Even–Odd Decomposition**	
	Multiplies	**Adds**	**Multiplies (Savings)**	**Adds (Savings)**
4-point	16	12	6 (62.5%)	8 (33.3%)
8-point	64	56	22 (65.6%)	28 (50.0%)
16-point	256	240	86 (66.4%)	100 (58.3%)
32-point	1024	992	342 (66.6%)	372 (62.5%)

Table 3.3 Computational complexity in 1-D $N \times N$ HEVC core transforms

Size	**Technique**			
	Matrix Multiplication		**Even–Odd Decomposition**	
	Multiplies	**Adds**	**Multiplies (Savings)**	**Adds (Savings)**
4 × 4	64	48	24 (62.5%)	32 (33.3%)
8 × 8	512	448	176 (65.6%)	224 (50.0%)
16 × 16	4096	3840	1376 (66.4%)	1600 (58.3%)
32 × 32	32768	31744	10944 (66.6%)	11904 (62.5%)

Table 3.4 Computational complexity in 2-D $N \times N$ HEVC core transforms

Size	**Technique**			
	Matrix Multiplication		**Even–Odd Decomposition**	
	Multiplies	**Adds**	**Multiplies (Savings)**	**Adds (Savings)**
4 × 4	128	96	48 (62.5%)	64 (33.3%)
8 × 8	1024	896	352 (65.6%)	448 (50.0%)
16 × 16	8192	7680	2752 (66.4%)	3200 (58.3%)
32 × 32	65536	63488	21888 (66.6%)	23808 (62.5%)

3.4.2 Multiplier-free Implementation

Table 3.4 has shown that the even–odd decomposition technique in implementing a 2-D $N \times N$ HEVC core transform could yield significant savings in the number of multiplications and additions with respect to the direct matrix multiplication. While those numbers are true for a software-based implementation, they do not represent the actual number of multipliers involved in a hardware-based implementation. For instance, in 4 × 4 1-D HEVC core transform, as the operation is executed in a column-wise manner, only six multipliers and eight adders/subtractors are required to execute 24 multiplications and 32 additions/subtractions, respectively, i.e., the same numbers involved in a 1-D 4-point transform. Similarly for 4 × 4 2-D HEVC core transform with two separate 1-D transform engines and a transpose buffer in between, only 12 multipliers and 16 adders/subtractors are necessary to implement the 48 multiplications and 64 additions/subtractions as shown in Table 3.4.

On hardware, a multiplication is regarded as an expensive operation as it utilises quite an amount of physical resources such as in the forms of Look-Up Tables (LUTs), Distributed/Block Random Access Memories (DRAMs/BRAMs), or Digital Signal Processor (DSP) slices, requiring a large area on a silicon chip especially when implementing large algorithms such as the 2-D 32 × 32 HEVC core transform. In arithmetic, a left bit-shift operation by n-bit on an input sample x (i.e., $x << n$) simply denotes a multiplication of x by 2^n. A bit-shift operation can be implemented using a shift register, or via a concatenation operation where a number of n zero bits are added as the suffix to x, or simply rewiring the least significant bits accordingly. The cost of bit-shifts in a well-designed digital system is generally not as significant as employing multipliers. It is, therefore, a common practice for an efficient and fast hardware implementation to adopt a multiplier-free approach using appropriate combinations of left bit-shifts and additions (including subtractions).

Table 3.5 shows how a multiplication on a sample with a matrix element of $N \times N$ HEVC core transform can be equivalently implemented with a combination of left bit-shifts and additions/subtractions. Table 3.6 shows the total number of shifters and adders/subtractors required in an N-point/$N \times N$ 1-D HEVC core transform implemented using the even–odd decomposition. From this table, instead of 342 multipliers and 372 adders/subtractors, a multiplier-free implementation of 32-point/32 × 32 1-D HEVC core transform would incur a total number of 922 shifters and 1088 adders/subtractors.

Table 3.5 Equivalent shift-add operations for HEVC core transform elements

Element	**Equivalent**	**Shifts**	**Adds/Subs**
90	$2^6 + 2^4 + 2^3 + 2^1$	4	3
89	$2^6 + 2^4 + 2^3 + 2^0$	3	3
88	$2^6 + 2^4 + 2^3$	3	2
87	$2^6 + 2^4 + 2^3 - 2^0$	3	3
85	$2^6 + 2^4 + 2^2 + 2^0$	3	3
83	$2^6 + 2^4 + 2^1 + 2^0$	3	3
82	$2^6 + 2^4 + 2^1$	3	2
80	$2^6 + 2^4$	2	1
78	$2^6 + 2^4 - 2^1$	3	2
75	$2^6 + 2^3 + 2^2 - 2^0$	3	3
73	$2^6 + 2^3 + 2^0$	2	2
70	$2^6 + 2^3 - 2^1$	3	2
67	$2^6 + 2^1 + 2^0$	2	2
64	2^6	1	0
61	$2^6 - 2^1 - 2^0$	2	2
57	$2^5 + 2^4 + 2^3 + 2^0$	3	3
54	$2^5 + 2^4 + 2^2 + 2^1$	4	3
50	$2^5 + 2^4 + 2^1$	3	2
46	$2^5 + 2^4 - 2^1$	3	2
43	$2^5 + 2^3 + 2^1 + 2^0$	3	3
38	$2^5 + 2^2 + 2^1$	3	2
36	$2^5 + 2^2$	2	1
31	$2^5 - 2^0$	1	1
25	$2^4 + 2^3 + 2^0$	2	2
22	$2^4 + 2^2 + 2^1$	3	2
18	$2^4 + 2^1$	2	1
13	$2^3 + 2^2 + 2^0$	2	2
9	$2^3 + 2^0$	1	1
4	2^2	1	0

Table 3.6 Complexity in multiplier-free N-point/$N \times N$ 1-D HEVC core transform using even–odd decomposition

Size	Multipliers		Shifts	Adders/Subtractors			
	Element	**Quantity**		**Multiplier Replacement**	**Adder Tree**[a]	**Add/Sub part**	**Total**
4-point	64	2	2	0	-	-	-
	83	2	6	6	-	-	-
	36	2	4	2	-	-	-
	Total	6	**12**	8	4	4	**16**
8-point (odd rows)	89	4	12	12	-	-	-
	75	4	12	12	-	-	-
	50	4	12	8	-	-	-
	18	4	8	4	-	-	-
	Total	16	**54**	36	12	8	**56**
16-point (odd rows)	90	8	32	24	-	-	-
	87	8	24	24	-	-	-
	80	8	16	8	-	-	-
	70	8	24	16	-	-	-
	57	8	24	24	-	-	-
	43	8	24	24	-	-	-
	25	8	16	16	-	-	-
	9	8	8	8	-	-	-
	Total	64	**168**	144	56	16	**216**
32-point (odd rows)	90	32	128	96	-	-	-
	88	16	48	32	-	-	-
	85	16	48	48	-	-	-
	82	16	48	32	-	-	-
	78	16	48	32	-	-	-
	73	16	32	32	-	-	-
	67	16	32	32	-	-	-
	61	16	32	32	-	-	-
	54	16	64	48	-	-	-
	46	16	48	32	-	-	-
	38	16	48	32	-	-	-
	31	16	16	16	-	-	-

	22	16	48	32	-	-	-
	13	16	32	32	-	-	-
	4	16	16	0	-	-	-
	Total	256	**688**	528	240	32	**800**
Total		342	**922**	716	312	60	**1088**

[a] O_{add} (Adder tree) = $\frac{N}{2}\left(\frac{N}{2}-1\right)$ per even or odd part

3.4.3 Multiple-Constant Multiplication (MCM)

The previous section has demonstrated potential hardware savings attainable by replacing multiplications with combinations of bit shifts and additions/subtractions. On top of the multiplier-free approach, another useful technique in reducing the complexity of a software-based implementation and hardware-sharing architecture is Multiple-Constant Multiplication (MCM). MCM is a technique of common sub-expressions elimination or sharing in an algorithm such as a mathematical formula, involving factorisation, rearrangements, etc. For instance in the case of the odd part of 4-point HEVC core transform (3.6), both the intermediate data W1 and W3 need to be multiplied by h_8 and h_{24}, which are 83 and 36, respectively (Fig. 3.3 (a)). The two multipliers can be represented as shown in (3.17a)–(3.17b) involving five shifts and four additions, where d is either W1 or W3, the odd intermediate data after the add/sub part. Using the MCM technique, (3.17a) can be replaced by (3.17c) saving one bit shifting, assuming that the implementation cost of a subtraction is normally the same as an addition.

$$83d = ((64 +^{1} 16) +^{4} (2 +^{2} 1))d \tag{3.17a}$$

$$36d = (32 +^{3} 4) \tag{3.17b}$$

$$83d = ((64 +^{1} 16) +^{4} (4 -^{2} 1))d \tag{3.17c}$$

In the odd part of 8-point HEVC core transform (3.10), all W1, W3, W5, and W7 need to be multiplied by h_4, h_{12}, h_{20}, and h_{28}, which are 87, 70, 43 and 9, respectively (Fig. 3.4 (a)). With multiplier-free implementation (3.18a)–(3.18d), at a first glance, the complexity involved may appear to be five shifts and nine additions. By utilising MCM, the complexity becomes five shifts and seven additions only.

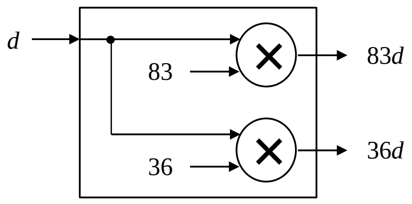
$$89d = ((64 +^{3} 16) +^{6} (8 +^{1} 1))d \tag{3.18a}$$

$$75d = (64 + 8 + 2 + 1)d = ((64 +^{4} 2) +^{7} (8 +^{1} 1))d \tag{3.18b}$$

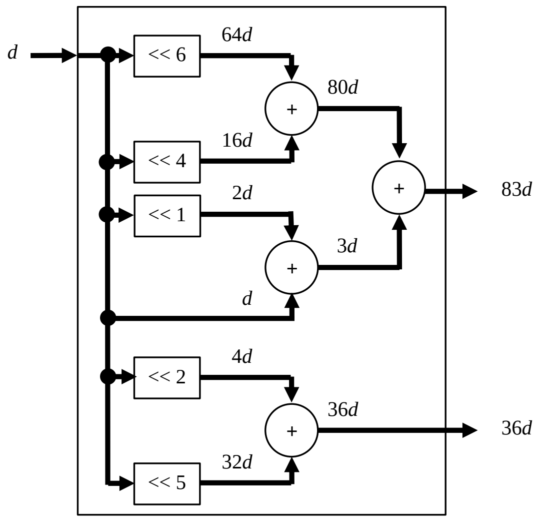
$$50d = (32 +^{5} (16 +^{2} 2))d \tag{3.18c}$$

$$18d = (16 +^{2} 2)d \tag{3.18d}$$

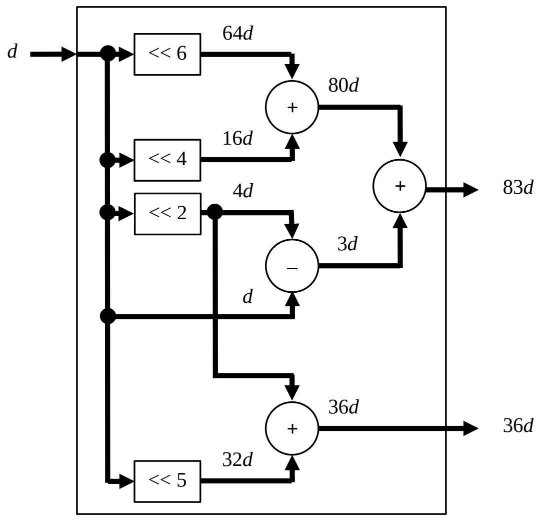
Similarly in the odd part of 16- or 32-point HEVC transforms, at a first glance, the independent multiplier-free implementation may incur six shifts and 19 or 33 additions, respectively. By applying MCM, the complexities are reduced to six shifts and 12 additions for the 16-point transform ((3.19a)–(3.19h) and (3.20a)–(3.20h)), and six shifts and 20 additions for the 32-point case (3.21a)–(3.21p). More than one configuration can yield similar savings, like in the two scenarios presented for the odd part of the 16-point HEVC core transform. Table 3.7 shows the number of bit shifts and additions/subtractions by utilising the MCM technique in the even–odd decomposition of HEVC forward transforms, and Table 3.8 provides the savings obtainable in comparison to without adopting MCM. From Table 3.8, employing MCM uses 81.1% and 24.3% fewer bit shifts and additions/subtractions, respectively, as opposed to without MCM.



(a)

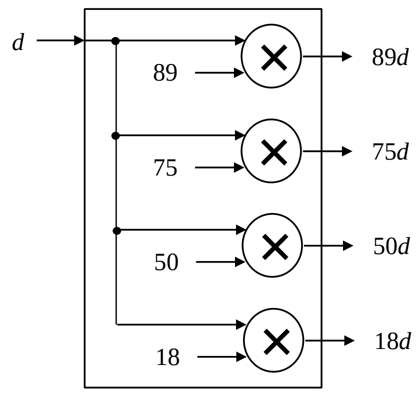


(b)

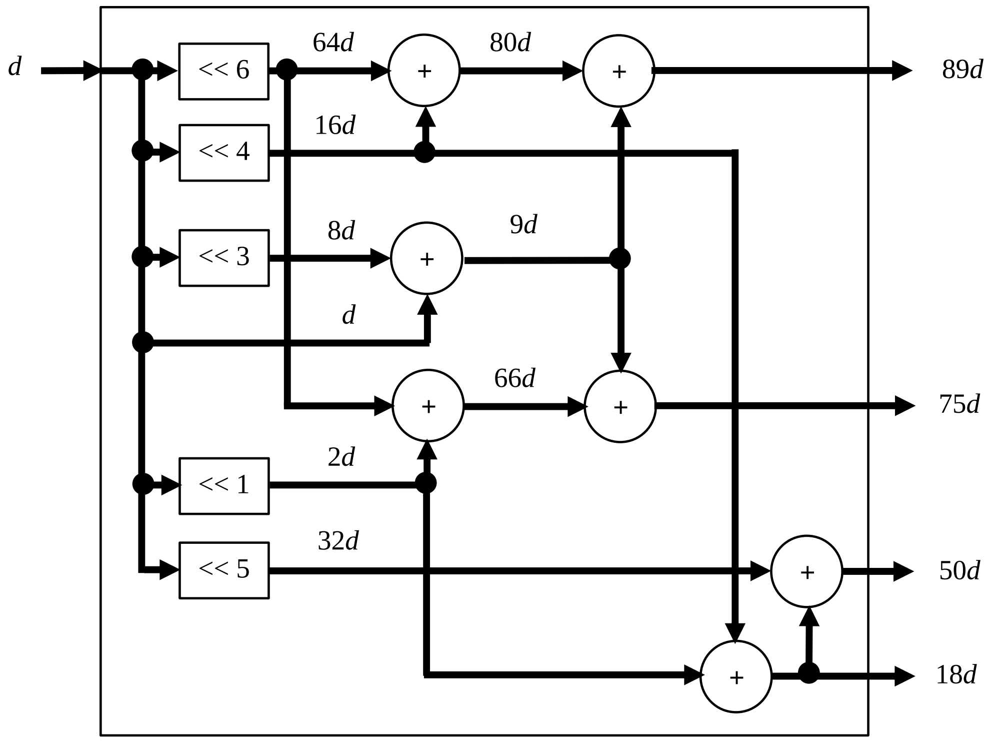


(c)

Fig 3.3 Functional block diagram of the (a) odd part, (b) less-efficient MCM multiplier-free and (c) MCM multiplier-free of 4-point HEVC core transform



(a)



(b)

Fig 3.4 Functional block diagram of the (a) odd part and (b) MCM multiplier-free of 8-point HEVC core transform

16-point MCM (Scenario A: 6 shifts, 12 additions)

$$90d = ((64 +^{1} 16) +^{7} (8 +^{5} 2))d \tag{3.19a}$$

$$87d = ((64 +^{1} 16) +^{9} (4 +^{3} 2) +^{8} 1)d \tag{3.19b}$$

$$80d = (64 +^{1} 16)d \tag{3.19c}$$

$$70d = ((64 +^{10} (4 +^{3} 2))d \tag{3.19d}$$

$$57d = (32 +^{11} (16 +^{4} (8 +^{2} 1)))d \tag{3.19e}$$

$$43d = ((32 +^{6} 2) +^{12} (8 +^{2} 1))d \tag{3.19f}$$

$$25d = (16 +^{4} (8 +^{2} 1))d \tag{3.19g}$$

$$9d = (8 +^{2} 1)d \tag{3.19h}$$

16-point MCM (Scenario B: 6 shifts, 12 additions)

$$90d = ((64 +^{1} 16) +^{7} (8 +^{5} 2))d \tag{3.20a}$$

$$87d = ((64 +^{12} 32) -^{9} (8 +^{2} 1))d \tag{3.20b}$$

$$80d = (64 +^{1} 16)d \tag{3.20c}$$

$$70d = ((64 +^{10} (4 +^{3} 2))d \tag{3.20d}$$

$$57d = (32 +^{11} (16 +^{4} (8 +^{2} 1)))d \tag{3.20e}$$

$$43d = ((32 +^{6} 2) +^{12} (8 +^{2} 1))d \tag{3.20f}$$

$$25d = (16 +^{4} (8 +^{2} 1))d \tag{3.20g}$$

$$9d = (8 +^{2} 1)d \tag{3.20h}$$

32-point MCM (6 shifts, 20 additions)

$$90_1d = (((64 +^1 16) +^7 8) +^8 2)d \quad (3.21a)$$

$$90_2d = 90_1d \quad (3.21b)$$

$$88d = ((64 +^1 16) +^7 8)d \quad (3.21c)$$

$$85d = ((64 +^1 16) +^9 (4 +^6 1))d \quad (3.21d)$$

$$82d = ((64 +^1 16) +^{10} 2)d \quad (3.21e)$$

$$78d = ((64 +^1 16) -^{11} 2)d \quad (3.21f)$$

$$73d = (64 +^{12} (8 +^3 1))d \quad (3.21g)$$

$$67d = (64 +^{13} (2 +^4 1))d \quad (3.21h)$$

$$61d = (64 -^{14} (2 +^4 1))d \quad (3.21i)$$

$$54d = ((32 +^2 16) +^{15} (4 +^5 2))d \quad (3.21j)$$

$$46d = ((32 +^2 16) -^{16} 2)d \quad (3.21k)$$

$$38d = (32 +^{17} (4 +^5 2))d \quad (3.21l)$$

$$31d = (32 -^{18} 1)d \quad (3.21m)$$

$$22d = (16 +^{19} (4 +^5 2))d \quad (3.21n)$$

$$13d = ((8 +^3 1) +^{20} 4)d \quad (3.21o)$$

$$4d = d << 2 \quad (3.21p)$$

Table 3.7 Complexity in multiplier-free *N*-point/*N* × *N* 1-D HEVC core transform using even–odd decomposition and Multiple-Constant Multiplication (MCM)

Size	**Multipliers**		**Shifts**	**Adders/Subtractors**			
	Element	**Quantity**		**Multiplier Replacement**	**Adder Tree**[a]	**Add/Sub part**	**Total**
4-point	64	2	2 * 1	0	-	-	-
	83	2	2 * 4	2 * 4	-	-	-
	36	2			-	-	-
	Total	6	**10**	8	4	4	**16**
8-point (odd rows)	89	4	4 * 5	4 * 7	-	-	-
	75	4			-	-	-
	50	4			-	-	-
	18	4			-	-	-
	Total	16	**20**	28	12	8	**48**
16-point (odd rows)	90	8	8 * 6	8 * 12	-	-	-
	87	8			-	-	-
	80	8			-	-	-
	70	8			-	-	-
	57	8			-	-	-
	43	8			-	-	-
	25	8			-	-	-
	9	8			-	-	-
	Total	64	**48**	96	56	16	**168**
32-point (odd rows)	90	32	16 * 6	16 * 20	-	-	-
	88	16			-	-	-
	85	16			-	-	-
	82	16			-	-	-
	78	16			-	-	-
	73	16			-	-	-
	67	16			-	-	-
	61	16			-	-	-
	54	16			-	-	-
	46	16			-	-	-
	38	16			-	-	-

	31	16			-	-	-
	22	16			-	-	-
	13	16			-	-	-
	4	16			-	-	-
	Total	256	**96**	320	240	32	**592**
Total		342	**174**	452	312	60	**824**

[a] O_{add} (Adder tree) = $\frac{N}{2}\left(\frac{N}{2}-1\right)$ per even or odd part

Table 3.8 Computational savings in multiplier-free N-point/$N \times N$ 1-D HEVC core transform using even–odd decomposition and Multiple-Constant Multiplication (MCM)

Size	**Technique**			
	Without MCM		**With MCM**	
	Shifts	**Adds**	**Shifts (Savings)**	**Adds (Savings)**
4-point	12	16	10 (16.7%)	16 (0.00%)
8-point (odd rows)	54	56	20 (63.0%)	48 (14.3%)
16-point (odd rows)	168	216	48 (71.4%)	168 (22.2%)
32-point (odd rows)	688	800	96 (86.0%)	592 (26.0%)
Total	**922**	**1088**	**174 (81.1%)**	**824 (24.3%)**

3.5 Intermediate Scaling

In order to maintain a reasonable trade-off between accuracy and computational complexities in the transform stage of HEVC, it was decided to limit the bit depth of the coefficients after each transform stage as 16-bit signed integers, i.e., in the range of $[-2^{15}, 2^{15} - 1]$ or [–32768, 32767] for any input bit depth, B. To achieve this requirement, additional intermediate scaling factors, S_{T1}, S_{T2}, S_{IT1}, and S_{IT2}, need to be applied as shown in Fig. 3.5. Note that Fig. 3.5 is a drilled-down diagram of the transform and quantisation blocks in Fig. 3.1.

Using the 4×4 forward transform as an example, the process of specifying the intermediate scaling factors can be illustrated as in Fig. 3.6. The assumption made in the worst-case bit-depth analysis was that for a video bit-depth of B, all samples of a residual block will have maximum amplitude of -2^B as the input to the first stage of the forward transform. A video with a bit depth of B bits will contain prediction residuals in the range of $[-2^B + 1, 2^B - 1]$ requiring a $(B + 1)$-bit representation. For instance, an 8-bit video will result in prediction residuals within [–255, 255] range requiring a 9-bit signed precision. For simplicity, it is assumed that the minimum residual value is –256, i.e., -2^8 or -2^B in general, which is still within the $(B + 1)$-bit signed precision.

The elements of both core and alternative HEVC transform matrices are 8-bit signed integers, but only the first basis vector row of these matrices are in the same sign region (positive) while all the other basis vectors oscillate between positive and negative regions. As the transform operation is a matrix multiplication process, the minimum (or absolute maximum) value of the transform coefficients after an N-point 1-D transform will be the result of multiplying the prediction residuals with the first basis vector of the transform matrix. More specifically, this will be the case when all the residual samples in the first column are equal to -2^B, i.e. –256 for $B = 8$. The elements of the first basis vector of the core transform matrix (64) are 6-bit precision. Therefore, the minimum value of the transform coefficients shall be $-2^B \times 2^6 \times 2^M$ where $N = 2^M$, resulting in a bit depth of $B + 6 + M$. For example, after a 4-point 1-D transform, the minimum transform coefficients will be $-256 \times 64 \times 4 = -2^8 \times 2^6 \times 2^2 = -65536$, i.e., requiring a bit depth of $8 + 6 + 2 = 16$.

In the alternative transform matrix, although the first basis vector contains 7-bit elements (74 and 84), the first two elements are 5- and 6-bit elements, respectively (29 and 55). Thus for B = 8, the minimum transform coefficients in the 4-point 1-D DST will be –256 × (29 + 55 + 74 + 84) = –61952, i.e., also within $B + 6 + M$ bit-depth as in the core transform case.

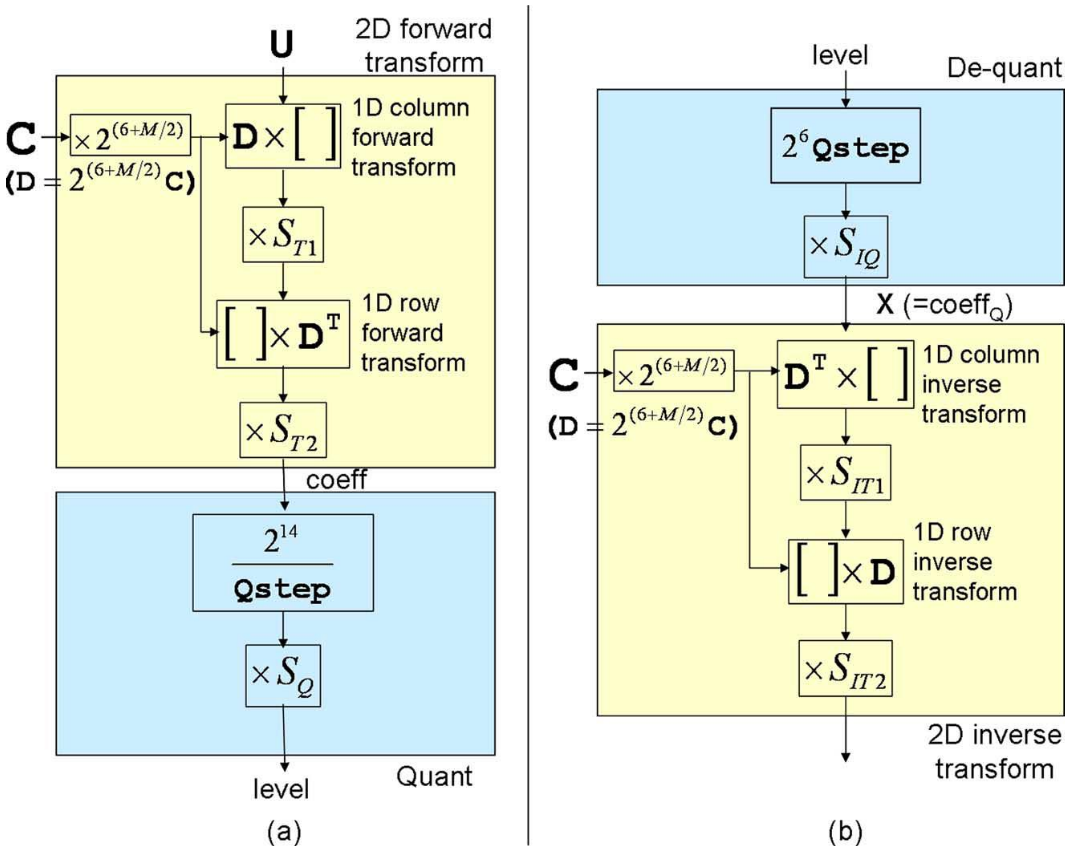


Fig. 3.5 Additional scale factors (S_{T1}, S_{T2}, S_{IT1}, S_{IT2}, S_Q, and S_{IQ}) to perform (a) forward transform and quantisation, and (b) inverse transform and quantisation of HEVC (**C** is the orthonormal DCT matrix, **D** is the scaled approximation of **C**, and $M = \log_2 N$, where N is the transform size) (Budagavi, Fuldseth and Bjøntegaard, 2014)

To maintain the bit-depth of the transform coefficients after the first N-point 1-D forward transform within 16-bit signed precision, a scaling factor of 1 / ($-2^B \times 2^6 \times 2^M \times 2^{-15}$) is, therefore, necessary. As a result, the scaling factor after the first transform stage is specified as $S_{T1} = 2^{-(B+M-9)}$.

The second stage of the forward transform involves a multiplication of the result of the first transform stage with $\mathbf{D}_4^{\mathrm{T}}$. The input into the second stage is the output from the first stage, which is a matrix having first row elements equal to -2^{15} and all other elements equal to zero as shown in Fig. 3.6 (b). Then, the output of multiplication with $\mathbf{D}_4^{\mathrm{T}}$ will be a matrix with only a DC value having a value of $-2^{15} \times 2^6 \times 2^M = -2^{(21+M)}$, while all remaining elements are equal to zero. Therefore, after the second stage of the forward transform the necessary scaling is $S_{T2} = 2^{-(M+6)}$ regardless of B.

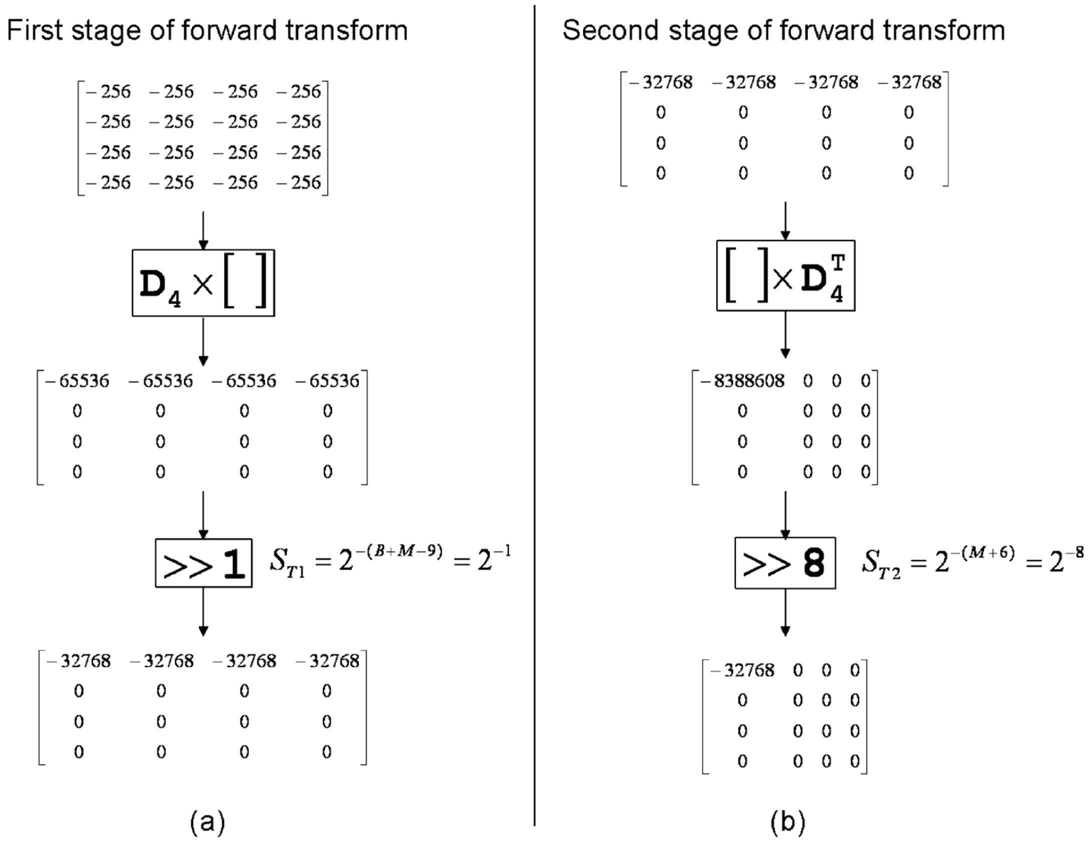

First stage of forward transform
Second stage of forward transform
$\mathbf{D}_4 \times [\ ]$
$[\ ] \times \mathbf{D}_4^{\mathrm{T}}$
$>> 1$
$S_{T1} = 2^{-(B+M-9)} = 2^{-1}$
$>> 8$
$S_{T2} = 2^{-(M+6)} = 2^{-8}$
(a)
(b)

Fig. 3.6 Intermediate scaling factor determination for (a) first and (b) second stages of forward transform to fit intermediate and output values within 16 bits (B is video bit-depth and $M = \log_2 N$, where N is the transform size) (Budagavi, Fuldseth and Bjøntegaard, 2014)

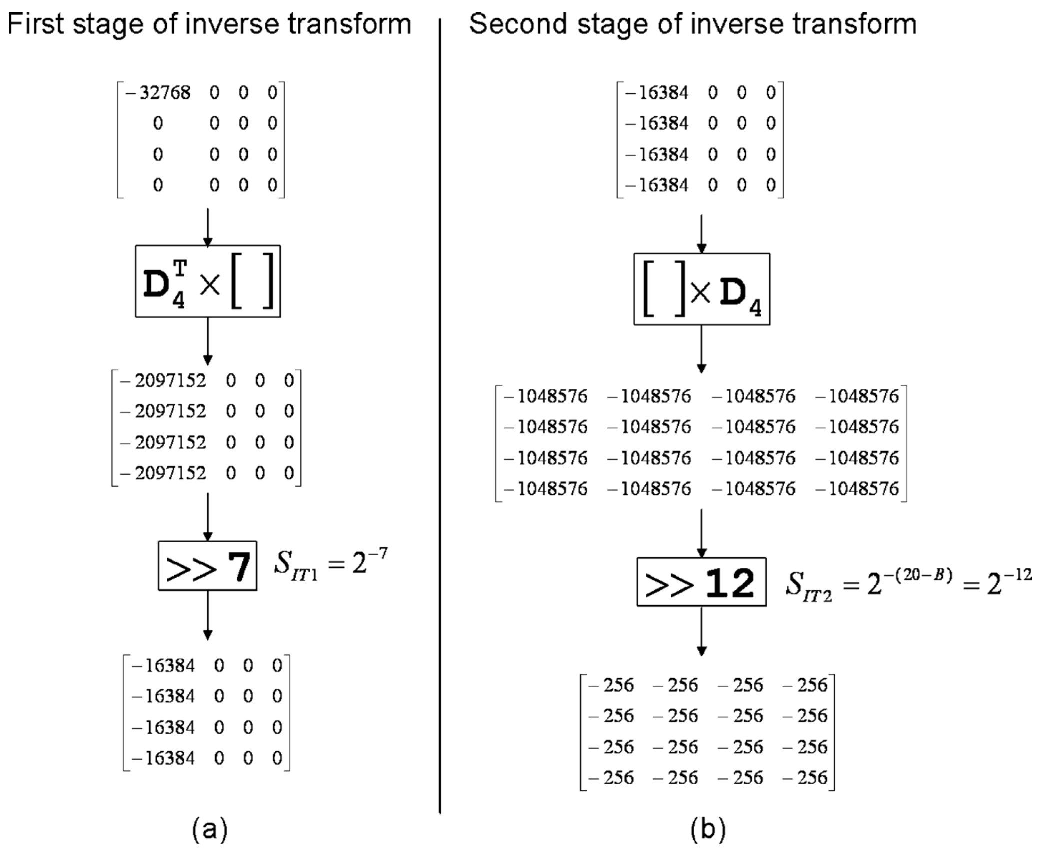
Similarly in the inverse transform, the first stage comprises a multiplication of the result of the forward transform with $\mathbf{D}_4^{\mathrm{T}}$ as shown in Fig. 3.7, assuming there are no or lossless quantisation/de-quantisation operations in between the forward and inverse transforms. Following the 4×4 1-D transform example set earlier in Fig. 3.6, the output matrix from the forward transform is input into this first stage of the inverse transform, which is a matrix with only the DC coefficient equalling to -2^{15}. The output of multiplication with $\mathbf{D}_4^{\mathrm{T}}$ will be a matrix with first column elements equal to $-2^{15} \times 2^{6} = -2^{21}$. Therefore, in order for the intermediate output to fit within 16 bits, the necessary scaling after this first stage of the inverse transform is simply $S_{IT1} = 2^{-6}$ regardless of B.

Finally, the second stage of the inverse transform involves a multiplication of the result of the first stage with $\mathbf{D}_4$. The output matrix from the first stage, which is a matrix with first column coefficients equal to -2^{15}, is input to the second stage. The output of the multiplication with $\mathbf{D}_4$ shall be a matrix with all elements equal to $-2^{15} \times 2^{6} = -2^{21}$. Therefore, the scaling required after the second stage of inverse transform in order to obtain the reconstructed output samples in the original range of $[-2^{B}, 2^{B} - 1]$ is $S_{IT2} = 2^{-(21-B)}$.

To summarise, for an input or output signal of bit-depth B, the scaling factors after the four transform stages are as follows, where $M = \log_2 N$ (Budagavi, Fuldseth and Bjøntegaard, 2014):

- After first forward transform stage, $S_{T1} = 2^{-(B+M-9)}$
- After second forward transform stage, $S_{T2} = 2^{-(M+6)}$
- After first inverse transform stage, $S_{IT1} = 2^{-6}$
- After second inverse transform stage, $S_{IT2} = 2^{-(21-B)}$

During the development of HEVC, it was decided to modify the scaling factors after each inverse transform stage; S_{IT1} and S_{IT2} to 2^{-7} and $2^{-(20-B)}$, respectively, in order to compensate for quantisation/de-quantisation errors which could possibly cause the dynamic range before each inverse transform stage to exceed 16 bits. A clipping operation was later introduced to ensure the dynamic range between the two inverse transform stages remains within 16 bits, therefore, the modification to S_{IT1} and S_{IT2} was no longer necessary, However, this modification was retained “for maturity reasons” (Budagavi, Fuldseth and Bjøntegaard, 2014). Tables 3.9 and 3.10 summarise the final choice of scaling factors of HEVC forward and inverse transform, respectively, in comparison to the orthonormal DCT (properties i and ii of Table 3.1).



First stage of inverse transform
Second stage of inverse transform
$D_4^T \times [\ ]$
$[\ ] \times D_4$
>> 7
$S_{IT1} = 2^{-7}$
>> 12
$S_{IT2} = 2^{-(20-B)} = 2^{-12}$
(a)
(b)

Fig. 3.7 Intermediate scaling factors in the inverse transform scale factors, assuming the input to be the final output of Fig. 3.6 (B = 8 is the video bit depth) (Budagavi, Fuldseth and Bjøntegaard, 2014)

Before each intermediate scaling, an offset value is also specified in HEVC to be added to perform rounding, which is equivalent to the scaling factor divided by two (2). For clarity, these offset values are not explicitly shown in Figs. 3.5, 3.6, and 3.7.

Table 3.9 Intermediate scaling factors in 2-D HEVC forward transform (Budagavi, Fuldseth and Bjøntegaard, 2014)

Stage	Scaling Factor
First forward transform	$2^{(6 + M/2)}$
After the first forward transform, S_{T1}	$2^{-(B + M - 9)}$
Second forward transform	$2^{(6 + M/2)}$
After the second forward transform, S_{T2}	$2^{-(M + 6)}$
Total scaling for forward transform	$2^{(15 - B - M)}$

Table 3.10 Intermediate scaling factors in 2-D HEVC inverse transform (Budagavi, Fuldseth and Bjøntegaard, 2014)

Stage	Scaling Factor
First inverse transform	$2^{(6 + M/2)}$
After the first inverse transform, S_{IT1}	2^{-7}
Second inverse transform	$2^{(6 + M/2)}$
After the second inverse transform, S_{IT2}	$2^{-(20 - B)}$
Total scaling for inverse transform	$2^{-(15 - B - M)}$

3.6 Quantisation

Quantisation involves division by a quantisation step (*Qstep*) and a rounding, making it a lossy operation. Conversely, inverse quantisation requires multiplication by *Qstep*. *Qstep* refers to the equivalent step size to have an orthonormal transform, i.e., without the scaling factors of Tables 3.9 and 3.10. Like H.264/AVC, HEVC also uses a quantisation parameter (QP) to obtain *Qstep*. For an 8-bit video sequence, there are 52 available QP values between 0 and 51 (Budagavi, Fuldseth and Bjøntegaard, 2014). A larger QP results in a larger *Qstep*, i.e., a heavier quantisation resulting in a more lossy output. Notably, QP = 4 was chosen to provide *Qstep* = 1, i.e., no effective quantisation, and an increase of six QP values leads to an increase of *Qstep* by a factor of two. These criteria result in the following relationship:

$$Qstep(QP) = \left(2^{1/6}\right)^{QP-4} \tag{3.22}$$

Equation (3.22) can equivalently be expressed as:

$$Qstep(QP) = G_{QP\%6} << \frac{QP}{6} \tag{3.23}$$

where $G = [G_0, G_1, G_2, G_3, G_4, G_5]^T = \left[2^{-4/6}, 2^{-3/6}, 2^{-2/6}, 2^{-1/6}, 2^0, 2^{1/6}\right]^{\mathrm{T}}$.

Additionally, frequency-dependent quantisation is also supported by HEVC by using scaling matrices. Frequency-dependent quantisation or scaling is useful in applying HVS-based quantisation where low-frequency transform coefficients are quantised with a finer quantisation step size in comparison to high-frequency coefficients. Let $W[r][c]$ represent the quantisation weight matrix for a transform coefficient at coordinate (r, c), $W[r][c] = 1$ means that there is no weighting.

In the inverse quantisation stage, for a quantised transform coefficient from the encoder, namely $level[r][c]$, the standard specifies the de-quantised transform coefficient as

$$\begin{aligned} coeff_Q[r][c] = \Bigg(\bigg(& level[r][c] \times w[r][c] \times \left(g_{QP\%6} \ll \frac{QP}{6}\right)\bigg) \\ & + offset_{IQ}\Bigg) \gg shift_1 \end{aligned} \tag{3.24}$$

where $w[r][c] = round(16 \times W[r][c])$, $offset_{IQ} = 1 \ll (M - 6 + B)$, $shift_1 = (M - 5 + B)$, and g_i is the de-quantiser multiplier specified as (Budagavi, Fuldseth and Bjøntegaard, 2014)

$$\mathrm{g} = [g_0, g_1, g_2, g_3, g_4, g_5]^T = round(2^6 \times \mathrm{G}) = [40, 45, 51, 57, 64, 72]^T \tag{3.25}$$

As previously mentioned, the HEVC standard mainly describes the decoding operations and syntax of a compliant bitstream. Thus, only the inverse quantisation is specified in the text specification (ITU, 2013) and encoder manufacturers have the flexibility to implement a quantisation scheme producing HEVC-compliant quantised transform coefficients, *level*. Reference (Budagavi, Fuldseth and Bjøntegaard, 2014) suggests that $level[r][c]$ at position (r, c) can be obtained as

$$\begin{aligned} level[r][c] = \Bigg(\bigg(\Big(& coeff[r][c] \times f_{QP\%6} \times \frac{16}{w[r][c]} + offset_Q\Big) \gg \frac{QP}{6}\bigg) \\ & \gg shift_2\Bigg) \end{aligned} \tag{3.26}$$

where $shift_2 = 29 - M - B$, and f_i is the quantiser multiplier specified as

$$f = [f_0, f_1, f_2, f_3, f_4, f_5]^T = [26214, 23302, 20560, 18396, 16384, 14564]^{\mathrm{T}}$$

When there is no frequency-dependent scaling ($W[r][c] = 1$, i.e., $w[r][c] = 16$) and $Qstep = 1$ (QP = 4), the choice of f_i and g_i provides almost unity gain through quantisation and inverse quantisation (i.e., $f_i \times g_i \times 16 \approx 1 \ll (shift_1 + shift_2) = 2^{(M - 5 + B + 29 - M - B)} = 2^{14} \times 2^6 \times 16$, $i = 0, \ldots, 5$).

3.7 Related work on Transform and Quantisation

This sub-section discusses some related work on transform and quantisation archived in the literature.

3.7.1 Related work on Transform

As a video sequence is a series of images, many image-processing techniques are applicable to video processing. There is a large amount of work done on the transform stage in image and video compression, aiming to reduce the computational complexity. A few examples in image compression include (Bouguezel, Ahmad and Swamy, 2010; Bayer *et al.*, 2012; Cintra, Bayer and Tablada, 2014; Coutinho *et al.*, 2016). Bouguezel, Ahmad, and Swamy (2010) proposed an orthogonal and multiplication-free transform of a dyadic order of up to 32-point for image compression, extended from the Integer Discrete Cosine Transform (ICT) and containing values of only ±1 and ±2. Cintra, Bayer, and Tablada (2014) presented a set of approximation matrices for 8-point ICT in image compression, based on common integer functions such as *floor, ceiling, truncation, round-away-from-zero, round-half-up/-down,* and *nearest integer* functions, involving values of 0, ±1, ±2, and ±3. Bayer *et al.* (2012) proposed a fast orthogonal algorithm and FPGA-based hardware prototype for 16-point ICT suitable for JPEG image compression, consisting of 1-bit transform matrix and a diagonal scaling matrix, which could be absorbed in the quantisation stage. Coutinho *et al.* (2016) presented a very low complexity 8×8 DCT approximation obtained via pruning, achieving 76.2% fewer arithmetic operations relative to the original DCT algorithm. Although these algorithms are beneficial for a hardware implementation, such low values of transform elements may not be too efficient for video coding. Approximation techniques such as described in (Cintra, Bayer and Tablada, 2014) however could be worth considering for future video coding standards.

References (Dong *et al.*, 2009; Haggag *et al.*, 2010; Haggag, El-Sharkawy and Fahmy, 2010; Sun *et al.*, 2012; Belghith, Loukil and Masmoudi, 2013a, 2013b; Wang *et al.*, 2013) are examples of algorithmic work on complexity-reduced transform applied in video coding. Dong *et al.* (2009) presented a non-orthogonal ICT (NICT) and a modified ICT (MICT) of 16×16 and applied these in H.264/AVC and Audio Video Coding Standard of China (AVS) Enhanced Profile. Their NICT and MICT matrices contain 6-bit and 4-bit elements, respectively. Similarly, Haggag *et al.* (Haggag *et al.*, 2010; Haggag, El-Sharkawy and Fahmy, 2010) decomposed the preliminary 16 × 16 HEVC transform kernel into the product of two sparse matrices involving 6-bit integers and provided two MICT algorithms for the odd frequency component. The first one is a quality-oriented algorithm, while the second one is computation-/speed-oriented. Belghith, Loukil, and Masmoudi (Belghith, Loukil and Masmoudi, 2013a, 2013b) showed how the 4 × 4 and 8 × 8 matrices are embedded in the 16 × 16 7-bit HEVC transform, and provided a slight modification on the odd part of 16-point. All these works exploit the dyadic symmetry property of the transform as applied by Cham and Chan (1991), and stand a good chance for an area-efficient transform implementation. However, most of the multiplications still require at least a 2-stage adder tree each. Sun *et al.* (2012) proposed an adaptive truncation re-configurable approximation (aTra) algorithm to automatically obtain approximated integers of adjustable precision. This algorithm still could not yield a satisfactory approximation for a one-stage adder pipeline as desired in this thesis.

There are many efficient hardware architecture designs proposed in recent years supporting the transform operation of HEVC, realised using CMOS (Ahmed, Shahid and Rehman, 2012; Budagavi and Sze, 2012; Park *et al.*, 2012; Shen *et al.*, 2012; Meher *et al.*, 2014; Bolaños-Jojoa and Velasco-Medina, 2015; Chang *et al.*, 2016) or/and FPGA (Zhao and Onoye, 2012; Conceição *et al.*, 2013; Kalali *et al.*, 2014; Arayacheeppreecha, Pumrin and Supmonchai, 2015; Darji and Makwana, 2015) technologies. Budagavi and Sze (2012) presented a unified and hardware-sharing architecture for both forward and inverse transforms of HEVC supporting all four sizes. They demonstrated how these operations can be executed using the even–odd decomposition technique involving three simple steps: 1) Addition/Subtraction; 2) Even part; 3) Odd part. When synthesised on a 45 nm CMOS technology, a 44% area reduction was achieved when compared with separate forward and inverse architectures. This work however only covered a 1-D transform.

During the development stage of HEVC, Park *et al.* (2012) proposed an efficient and multiplier-free architecture for 2-D 16 × 16 and 32 × 32 inverse transforms, based on Chen’s fast DCT algorithm (1977) and capable of decoding a Quad Full HD (QFHD) (3840 × 2160) video at 30 frames per second (fps). Besides using the old transform elements, the only other downside possibly is that this design did not cover the other two small sizes (4 × 4 and 8 × 8). The design by Ahmed, Shahid, and Rehman (2012) is another work published prior to the release of HEVC, but it supports all four sizes of 2-D forward transform of HEVC. They applied the folded scheme, i.e., a single 1-D transform core is reused for the second transform operation after the transpose stage. Their architecture relies on matrix decomposition into permutation matrices and Givens rotation matrices. Additionally, they applied the lifting scheme to eliminate any use of multipliers. The design was synthesised using a 90 nm CMOS library and capable of encoding a 1080p HD video at 48 fps.

The work by Chang *et al.* (2016) was also based on sparse matrix decomposition and 67% more hardware efficient than (Ahmed, Shahid and Rehman, 2012). Bolaños-Jojoa and Velasco-Medina (2015) presented three hardware designs for *N*-point HEVC transform based on Multiple-Constant Multiplication (MCM) technique. But both (Chang *et al.*, 2016) and (Bolaños-Jojoa and Velasco-Medina, 2015) only covered 1-D inverse DCT (IDCT).

The work by Shen *et al.* (2012) is an example of Multi-Standard Transform (MST) designs. Their design supports the inverse transform of MPEG-2/-4, AVC, AVS, VC-1, and HEVC standards. They applied multiplier-free architecture for 4-point and 8-point transforms and regular multipliers for 16-point and 32-point operations. The transpose buffer was designed using four single-port Synchronous Random Access Memory (SRAM) blocks instead of a regular register array. Synthesised with Semiconductor Manufacturing International Corporation (SMIC) 130 nm CMOS library, their 5-stage pipelined design can potentially support QFHD @ 30 fps videos. These results were however limited to a 1-D IDCT operation with a transpose buffer. Other MST designs include (Martuza and Wahid, 2012a, 2012b, 2015, Dias, Roma and Sousa, 2013, 2014).

Meher *et al.* (2014) presented area- and power-efficient architectures for 2-D HEVC forward transform using MCM technique to replace physical multipliers. Their hardware-oriented algorithms involve three stages for all four sizes: 1) Input Adder Unit (IAU); 2) Shift-Add Unit (SAU); 3) Output Adder Unit (OAU). They have also integrated the intermediate scaling stage after each transform operation as defined in the HEVC standard into their SAU stage, by pruning some of the least significant bits (LSBs) to maintain a maximum of 16-bit width after each 1-D transform. They synthesised their designs using Taiwan Semiconductor Manufacturing Company (TSMC) 90 nm CMOS library and proficient to encode a 7680 × 4320 UHD video at 60 fps.

The design by Darji and Makwana (2015) utilises the Canonical Signed Digit (CSD) representation, as well as Common Sub-expression Elimination (CSE), which is essentially an MCM technique. On the other hand, the design by Arayacheeppreecha, Pumrin, and Supmonchai (2015) has efficiently addressed the challenge of parallel execution of flexible transform size combinations. Both (Darji and Makwana, 2015) and (Arayacheeppreecha, Pumrin and Supmonchai, 2015) are however 1-D forward transform designs.

Recently, Raguraman and Saravanan (2016) extended several existing 8-point 1-D approximated transforms in the literature into 2-D architectures and implemented them on a Xilinx Virtex-E FPGA. More recently, da Silveira *et al.* (2017) proposed a low-complexity orthogonal 16-point approximated transform combining two instantiations of a low-complexity 8-point DCT approximation introduced in (Bayer and Cintra, 2012). The entries of the resulting transformation matrix are defined over $\{0, \pm 1\}$ making the multiplicative complexity null and the arithmetic complexity of their 1-D algorithm to be only 44 additions. The proposed 1-D approximation was realised on a Xilinx Virtex-6 XC6VLX240T FPGA requiring only 303 Configurable Logic Blocks (CLBs) and 936 Flip-Flops (FF), achieving a maximum operating frequency of 344.83 MHz.

The multiplier-free 2-D designs by Conceição *et al.* (2013) and Zhao and Onoye (2012) were realised using the even–odd decomposition approach and synthesised on Altera Stratix IV and Cyclone IV FPGA, respectively. While (Conceição *et al.*, 2013) was focused only on 32 × 32 inverse transform, (Zhao and Onoye, 2012) supports the forward transform of all four sizes defined in HEVC. Potential applications of (Conceição *et al.*, 2013) and (Zhao and Onoye, 2012) include QFHD @ 30 fps and Wide Quad Extended Graphics Array (WQXGA) (2560 × 1600) @ 30 fps videos, respectively.

To further reduce the power consumption, Kalali *et al.* (2014) also proposed a novel energy reduction technique in addition to using the MCM technique in their HEVC 2-D IDCT design, supporting all four sizes. Their design was mapped to a

Xilinx Virtex-6 FPGA and capable of decoding QFHD @ 48 fps videos. Recently, they have improved their design and produced a low utilisation (LU) hardware scheme maintaining the same decoding capability but costing fewer resources, as well as a high utilisation (HU) hardware design capable of processing UHD (7680 × 4320) @ 53 fps videos, also at a lower hardware cost than their initial design (Kalali, Mert and Hamzaoglu, 2016).

Very recently, Chen, Zhang, and Lu (2017) presented a FPGA-friendly architecture supporting all four sizes in 2-D HEVC transform and implemented their design on several FPGA platforms manufactured by Altera such as Startix III, Cyclone II, and Arria II GX, as well as Xilinx Virtex-7 and Zynq. Apart from using 16 64 × 16-bit Block RAMs (BRAMs) as the transpose buffer, their architecture exploits other internal components and characteristics of individual FPGA platforms such as DSP blocks on top of Arithmetic Logic Modules (ALM) or Look-up Tables (LUT) to realise the logic computations. Their design, particularly on a Xilinx Zynq FPGA, can sustain up to 4K @ 30 fps 4:2:0 UHD TV.

3.7.2 Related work on Quantisation

There are not as many previous works on the quantisation stage of HEVC as the transform stage. Stankowski *et al.* (2015) analysed the maximum achievable performance in terms of bitrate savings when exact rate-distortion optimised quantisation (RDOQ) calculations were used instead of estimated RDOQ calculations currently employed by HEVC. Their analysis has shown that by using exact RDOQ calculations, differences of -1.1% and -1.0% in luminance bitrate could be achieved in AI and RA configurations, respectively, but at the expense of three to four times higher computational complexity (execution times). Gweon and Lee (2012) have proposed *N*-level quantisation for HEVC instead of an equal quantisation to be applied to a TU. With a TU divided into 4 × 4 blocks, blocks towards the bottom right corner of the TU are quantised with a higher quantiser step. Having the maximum number of *N* equals three, luminance BD-rate performance could be improved by -0.3% to -0.5% in AI, LB, and LP configurations, with almost the same encoding and decoding times with the reference HEVC software. Nam, Sim, and Bajić (2012) proposed an adaptive quantisation method either in spatial or transform domain for screen content videos. Generated by computer graphics techniques, screen content videos have different texture properties than natural videos, such as having very sharp edges at text or object boundaries as well as containing significantly less noise. Common video coding tools, on the other hand, tend to smooth out sharp edges exploiting the nature of HVS. Their proposed method could yield an average of 4.1% bitrate reduction for the AI, RA, and low delay configurations. The increase in complexity was however not reported.

Dias, Roma, and Sousa (2015) presented a unified quantisation and de-quantisation architecture for HEVC, offering a reduced hardware cost compared to separate implementations of the two operations. Their synthesis results on an ASIC technology show that savings between 19.2% and 27.3% in area squared as well as 20.9% to 27.6% in gate count reduction could be achieved by their unified architecture. Their architecture designs could as well operate at operating frequencies higher than 374 MHz, and fast enough to support a real-time encoding of 4K UHD @ 30 fps videos. Pastuszak (2014) presented three FPGA architecture designs each for the quantisation and de-quantisation of HEVC. The first design was a straightforward implementation, the second design was with shifter modifications, and the third design mapped rounding adders into digital signal processor (DSP) units. The designs were synthesised for Altera Arria II GX devices and capable of working at 200 MHz. The second and third designs offer general-purpose logic reduction between 34% and 88% relative to the straightforward implementation.

3.8 Summary

HEVC core transform, intermediate scaling, and quantisation were described in this chapter as these operations serve the basis of this thesis. A literature survey on transform and quantisation in the context of HEVC was also discussed. Particularly for the transform stage, complexity-reduced techniques in the forms of multiplier-free implementation, even–odd decomposition, and Multiple-Constant Multiplication (MCM) approach were covered in this chapter and applied in many previous works. The next chapter will reveal the algorithmic contributions of this thesis, which are approximated transforms and quantisation aimed at a complexity-reduced HEVC execution without severe degradation in the reconstructed and decoded video quality.

Chapter 4

Approximated Forward Core Transform, Intermediate Scaling, and Quantisation for HEVC

Abstract This chapter explains the derivation of the approximated forward core transform matrices, the modification on the intermediate scaling, and approximated quantisation multipliers adopted in this thesis. These alterations were made with the intention to reduce the implementation complexity of an HEVC encoder bearing only a minimal tolerance on the coding performance.

4.1 Introduction

The 32 × 32 HEVC forward transform matrix, h_{rc}^{32}, can be constructed using 29 unique integers. These integer constants originated from 31 unique elements (excluding the first row, C_0) derivable from (3.1) and listed in Table 4.1 for the first column of the matrix, before the *round-to-nearest-integer* operation and a hand-tuning on a few elements (C_8, C_{21}, C_{23}, C_{24}, C_{25}, and C_{26}). Notably after the rounding and hand tuning operations, $C_0 = C_{16} = 64$ and $C_1 = C_2 = C_3 = 90$.

Table 4.1 Constants in 32 × 32 HEVC core transform matrix

Constant	Value	Before rounding and hand tuning	Constant	Value	Before rounding and hand tuning
C_0	64	64.00	C_{16}	64	64.00
C_1	90	90.40	C_{17}	61	60.78
C_2	90	90.07	C_{18}	57	57.42
C_3	90	89.53	C_{19}	54	53.92
C_4	89	88.77	C_{20}	50	50.28
C_5	88	87.80	C_{21}	46	46.53

C_6	87	86.61	C_{22}	43	42.67
C_7	85	85.22	C_{23}	38	38.70
C_8	83	83.62	C_{24}	36	34.64
C_9	82	81.82	C_{25}	31	30.49
C_{10}	80	79.82	C_{26}	25	26.27
C_{11}	78	77.63	C_{27}	22	21.99
C_{12}	75	75.26	C_{28}	18	17.66
C_{13}	73	72.70	C_{29}	13	13.28
C_{14}	70	69.96	C_{30}	9	8.87
C_{15}	67	67.06	C_{31}	4	4.44

As noted earlier in sub-section 3.4.2, a fast and cost-efficient hardware implementation typically applies a multiplier-free approach using appropriate combinations of left bit-shifts and additions (including subtractions). For instance, Fig. 4.1 illustrates a data multiplication on x by 87 (C_6 as in Table 4.1), involving three adders in a two-stage adder tree structure, assuming that this operation is fast enough to be executable in a single clock cycle (cc) and the cost of an adder is the same as a subtractor. IR_1 and OR_1 in Fig. 4.1 are the input register and the output register, respectively. Similarly, most other multipliers like 90, 89, 88, etc. from Table 4.1 may incur a two-stage adder tree. Some integers such as 80, 36, 31, 18, and 9 involve only two bit-shifts and an addition, while for the two dyadic elements, i.e., 64 and 4, only a single bit-shift is required (<< 6 and << 2, respectively). Although a complexity reduction technique such as MCM (sub-section 3.4.4) can be utilised, it is necessary to see the effect of an approximation scheme on the coding performance, such as in terms of quality (e.g. PSNR, SSIM, MOS, etc.) with respect to the bitrate, and the potential savings offered by such a scheme. The following section, Section 4.2, therefore, details out the approximated core transform algorithm applied in this work, followed by a sub-section describing the scaled version of this algorithm involving a modification in the subsequent intermediate scaling stage. Section 4.3 provides the approximated quantisation multipliers derived using the same principle.

Only the core transform draws the attention, and the alternative transform is not included in this work.

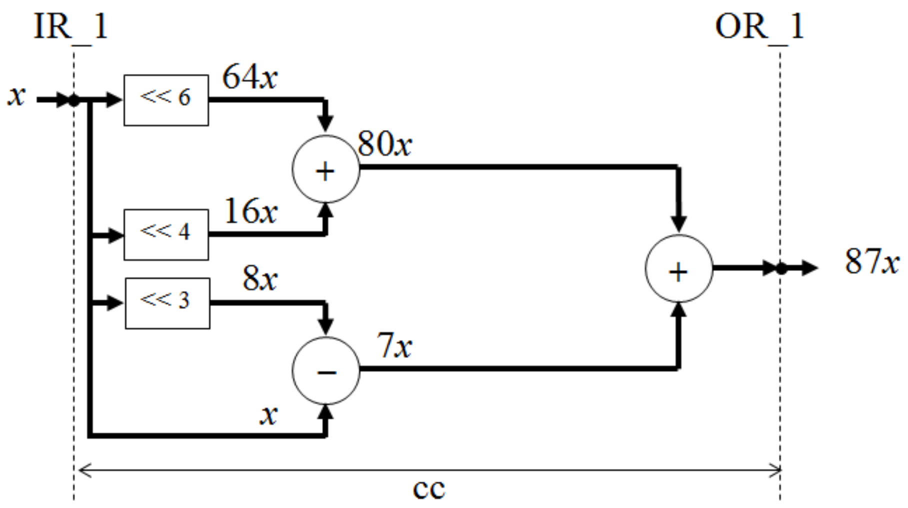


Fig. 4.1 A hardware implementation on multiplication of *x* by 87

4.2 Approximated Forward Core Transform

This section describes the process taken in deriving the approximated forward core transform used in this thesis.

4.2.1 Algorithmic Modelling

In order to further reduce the hardware requirements for the transform stage of HEVC, three criteria were predefined to approximate the original core matrix elements with more hardware-friendly integers as follows:

i. All the new integers must be multiples of four as this is the smallest multiplier in Table 4.1. Furthermore, it is a favourable dyadic number. By having a common denominator, all the new elements can be factorised and this scaling factor can subsequently be absorbed in a proceeding stage in the encoding pipeline, such as the intermediate scaling or quantisation stage;

ii. Only a single adder/subtractor can be allocated in a multiplier replacement;

iii. All combinations of bit-shifts and additions/subtractions are executable in only one clock cycle of 5 ns (200 MHz) or shorter.

This technique can thus be regarded as an elimination of sub-operations. Based on the above criteria, the following look-up table (LUT) was derived, i.e., a set of 18 hardware-friendly integers and named as LUT_4:

$$LUT_4 = \{4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 48, 56, 60, 64, 68, 72, 80, 96\}$$

A multiplication with the integers in LUT_4 can be performed by either a single left shift (4, 8, 16, 32, and 64) or two left shifts and one addition as illustrated in Table 4.2.

Table 4.2 Equivalent shift-add operations of LUT_4 integers

No.	***LUT*₄ integers**	**Equivalent shift-add operations**
1	4	<< 2
2	8	<< 3
3	12	<< 3 + << 2
4	16	<< 4
5	20	<< 4 + << 2
6	24	<< 4 + << 3
7	28	<< 5 – << 2
8	32	<< 5
9	36	<< 5 + << 2
10	40	<< 5 + << 3
11	48	<< 5 + << 4
12	56	<< 6 – << 3
13	60	<< 6 – << 2
14	64	<< 6
15	68	<< 6 + << 2
16	72	<< 6 + << 3
17	80	<< 6 + << 4
18	96	<< 6 + << 5

Next, several approximation alternatives can be systematically obtained by a search algorithm to represent each original element with an integer from LUT_4 giving the least absolute difference. If an element can be replaced by more than one integer, a mathematical function similar to *ceiling* (upwards), *floor* (downwards), or a combination of both functions was applied. Table 4.3 provides the decision criteria of 22 approximation alternatives analysed in this work, T1–T22. For instance, T1 and T2 were obtained respectively by applying the *ceiling* (upwards) and *floor* (downwards) function on the original HEVC core transform, while T3 and T4 were acquired by using the scaled and floating DCT as the reference instead of HEVC. Table 4.4 lists the first column of these 22 alternatives and Fig. 4.2 illustrates the flow chart of the search algorithm, where f(x) is the function to either perform the *ceiling* (upwards) or *floor* (downwards) approximation.

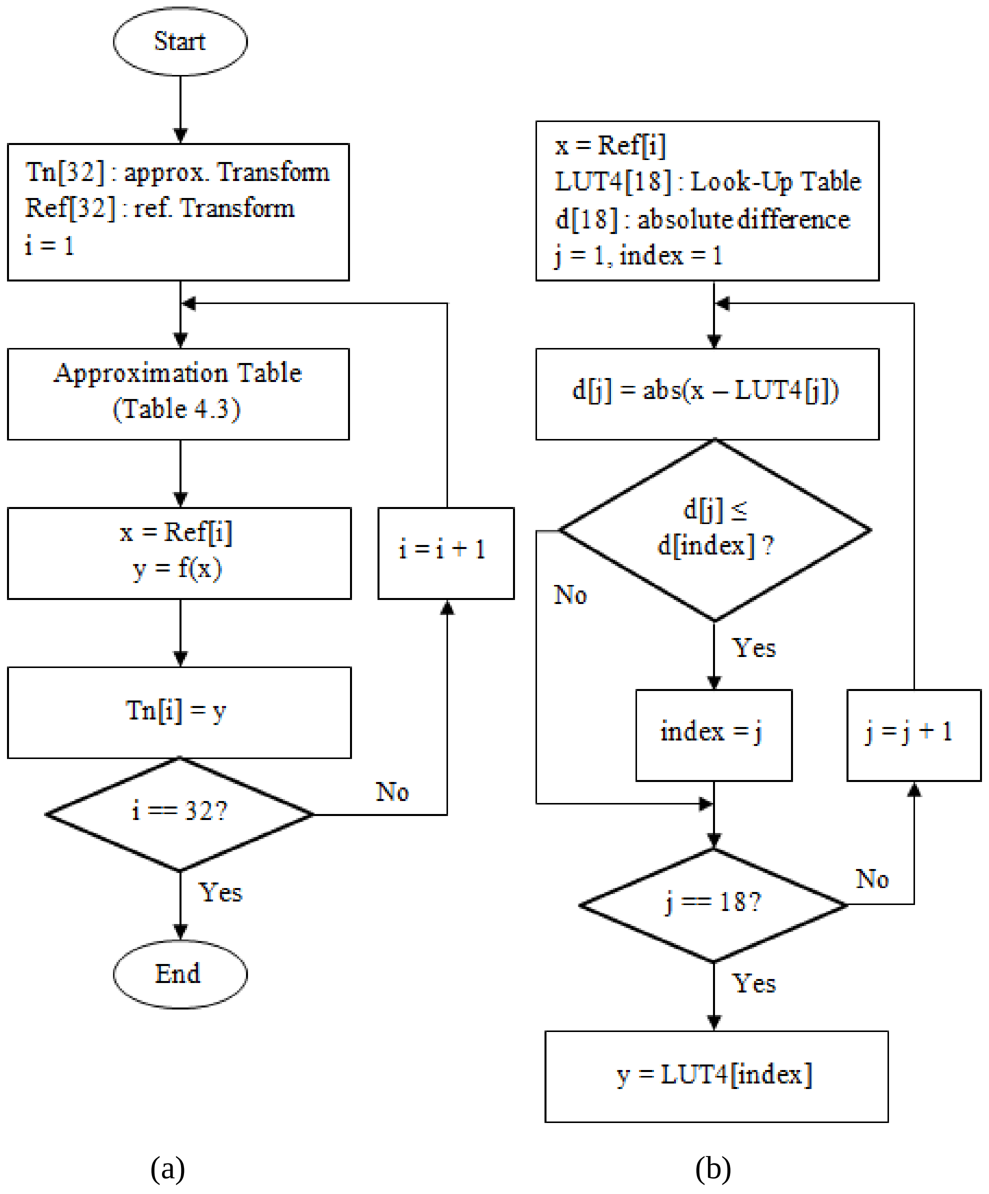


Fig. 4.2 Flow chart of the search algorithm, where (a) is the main flow and (b) is the *ceiling* (upwards) approximation flow of f(x)

Table 4.3 Decision criteria of approximation alternatives

Transform	**Reference**	**Upper half**	**Lower half**	**Even rows**	**Odd rows**
D_{sf}[a]	-	-	-	-	-
D_{sr}[b]	-	-	-	-	-
HEVC	-	-	-	-	-
T1	HEVC	*ceiling*	*ceiling*	-	-
T2	HEVC	*floor*	*floor*	-	-
T3	D_{sf}	*ceiling*	*ceiling*	-	-
T4	D_{sf}	*floor*	*floor*	-	-
T5	HEVC	*ceiling*	*floor*	-	-
T6	HEVC	*floor*	*ceiling*	-	-
T7	D_{sf}	*ceiling*	*floor*	-	-
T8	D_{sf}	*floor*	*ceiling*	-	-
T9	HEVC	-	-	*ceiling*	*floor*
T10	HEVC	-	-	*floor*	*ceiling*
T11	D_{sf}	-	-	*ceiling*	*floor*
T12	D_{sf}	-	-	*floor*	*ceiling*
T13	D_{sr}	*ceiling*	*ceiling*	-	-
T14	D_{sr}	*floor*	*floor*	-	-
T15	D_{sr}	*ceiling*	*floor*	-	-
T16	D_{sr}	*floor*	*ceiling*	-	-
T17	D_{sr}	-	-	*ceiling*	*floor*
T18	D_{sr}	-	-	*floor*	*ceiling*
T19	Scale= 362.0	*ceiling*	*ceiling*	-	-
T20	Scale = 362.0	*floor*	*floor*	-	-
T21	Scale = 360.0	*ceiling*	*ceiling*	-	-
T22	Scale = 360.0	*floor*	*floor*	-	-

[a] DCT Scaled and Floating (Scaling factor = $64*32^{1/2}$ = 362.03867196751233249323231339768)

[b] DCT Scaled and Rounded (Scaling factor = $64*32^{1/2}$ = 362.03867196751233249323231339768)

Table 4.4 Matrix elements in the first column of different 32 × 32 core transform alternatives

First column	Transform Alternatives														
	D_{sf}	**D_{sr}**	**HEVC**	**T1**	**T2**	**T3[a]**	**T5**	**T6**	**T9**	**T10**	**T13**	**T15**	**T16**	**T17**	**T18**
0	64.00	64	64	64	64	64	64	64	64	64	64	64	64	64	64
1	90.40	90	90	96	96	96	96	96	96	96	96	96	96	96	96
2	90.07	90	90	96	96	96	96	96	96	96	96	96	96	96	96
3	89.53	90	90	96	96	96	96	96	96	96	96	96	96	96	96
4	88.77	89	89	96	96	96	96	96	96	96	96	96	96	96	96
5	87.80	88	88	96	80	80	96	80	80	96	96	96	80	80	96
6	86.61	87	87	80	80	80	80	80	80	80	80	80	80	80	80
7	85.22	85	85	80	80	80	80	80	80	80	80	80	80	80	80
8	83.62	84	83	80	80	80	80	80	80	80	80	80	80	80	80
9	81.82	82	82	80	80	80	80	80	80	80	80	80	80	80	80
10	79.82	80	80	80	80	80	80	80	80	80	80	80	80	80	80
11	77.63	78	78	80	80	80	80	80	80	80	80	80	80	80	80
12	75.26	75	75	72	72	72	72	72	72	72	72	72	72	72	72
13	72.70	73	73	72	72	72	72	72	72	72	72	72	72	72	72
14	69.96	70	70	72	68	68	72	68	72	68	72	72	68	72	68
15	67.06	67	67	68	68	68	68	68	68	68	68	68	68	68	68
16	64.00	64	64	64	64	64	64	64	64	64	64	64	64	64	64
17	60.78	61	61	60	60	60	60	60	60	60	60	60	60	60	60
18	57.42	57	57	56	56	56	56	56	56	56	56	56	56	56	56
19	53.92	54	54	56	56	56	56	56	56	56	56	56	56	56	56
20	50.28	50	50	48	48	48	48	48	48	48	48	48	48	48	48
21	46.53	47	46	48	48	48	48	48	48	48	48	48	48	48	48
22	42.67	43	43	40	40	40	40	40	40	40	40	40	40	40	40
23	38.70	39	38	40	36	40	36	40	36	40	40	40	40	40	40
24	34.64	35	36	36	36	36	36	36	36	36	36	36	36	36	36
25	30.49	30	31	32	32	32	32	32	32	32	32	28	32	28	32
26	26.27	26	25	24	24	28	24	24	24	24	28	24	28	28	24
27	21.99	22	22	24	20	20	20	24	20	24	24	20	24	20	24
28	17.66	18	18	20	16	16	16	20	20	16	20	16	20	20	16
29	13.28	13	13	12	12	12	12	12	12	12	12	12	12	12	12
30	8.87	9	9	8	8	8	8	8	8	8	8	8	8	8	8
31	4.44	4	4	4	4	4	4	4	4	4	4	4	4	4	4
o_{rc}	0.0000	0.0037	0.0029	0.0566	0.0439	0.0442	0.0566	0.0439	0.0566	0.0439	0.0518	0.0566	0.0391	0.0518	0.0439
m_{rc}	0.0000	0.0077	0.0213	0.1282	0.1218	0.1218	0.1282	0.1218	0.1218	0.1282	0.1282	0.1282	0.1218	0.1218	0.1282
n_{rc}	0.0000	0.0109	0.0013	0.0605	0.0605	0.0605	0.0605	0.0605	0.0605	0.0605	0.0605	0.0605	0.0605	0.0605	0.0605

[a] T4, T7, T8, T11, T12, T14, T19, T20, T21, and T22 resulted in the same values as T3

4.2.2 Degrees of Approximation

By performing integer approximations to the floating DCT or HEVC core transform, some of the DCT properties (Table 3.1) may have been compromised. In order to measure the degrees of approximation for a few of these properties, the same measurements as applied in (Budagavi, Fuldseth and Bjøntegaard, 2014) were adopted. These are namely the orthogonality measure, o_{rc}, closeness to the original DCT, m_{rc}, and norm measure, n_r, as given in (4.1)–(4.3) for an N-point integer DCT approximation. In these equations, t_{rc} represents the matrix elements with r and c = 0, 1 … N – 1, basis vector rows are equal to t_r, t_c is the transpose of t_r, and d_{rc} are again the real DCT matrix elements (Section 3.2).

$$o_{rc} = t_r t_c \, / \, t_{r=0} t_{r=0}^T \ , \qquad r \neq c \tag{4.1}$$

$$m_{rc} = |\propto d_{rc} - t_{rc}| \, / \, t_{00} \tag{4.2}$$

$$n_r = |1 - t_r t_c \, / t_{r=0} t_{r=0}^T| \tag{4.3}$$

The worst-case values of o_{rc}, m_{rc}, and n_r for different transform alternatives are given at the bottom of Table 4.4, with the value of zero implying a perfect achievement. For comparison purposes, the respective measures of scaled and floating DCT, D_{sf}, scaled and rounded DCT, D_{sr}, and HEVC matrix elements are also included in the second, third, and fourth columns of Table 4.4, respectively.

HEVC core transform matrix elements are further from D_{sf} than D_{sr} (m_{rc}(HEVC) = 0.0213) due to the rounding and hand tuning operations. Nevertheless, they hold better orthogonality (o_{rc}(HEVC) = 0.0029) and norm (n_{rc}(HEVC) = 0.0013) properties than D_{sr} (Budagavi, Fuldseth and Bjøntegaard, 2014). On the contrary, all the approximation alternatives analysed in this work possess far worse o_{rc}, m_{rc}, and n_{rc} values indicating coarser approximations than HEVC. All 22 alternatives have the same value of n_{rc} (0.0605). The worst alternative transforms are T1, T5, and T15 having the largest o_{rc} (0.0566) and m_{rc} (0.1282) values as highlighted in red in Table 4.4. The best alternative approximated is identified to be T16 with o_{rc} and m_{rc} values of 0.0391 and 0.1218, respectively. For this reason, T16 was chosen to be the approximated transform matrix to be further analysed as the replacement of 32 × 32 HEVC core matrix (including the three embedded smaller matrices).

4.2.3 Arithmetic Complexity Analysis

In order to compare the number of bit shifts and additions/subtractions required by the chosen approximated transform matrix, T16, the same approaches of multiplier-free implementation, even–odd decomposition, and MCM are applied. The even part of the 4-point transform is the same as the original HEVC transform, thus it can be realised in the same way as (3.5) costing two shifts and two additions. The odd part of 4-point transform can be executed by (4.4a)–(4.4b) involving four shifts and two additions, where *d* is again the odd intermediate data after the add/sub part.

$$80d = (64 +^{1} 16)d \tag{4.4a}$$

$$36d = (32 +^{2} 4)d \tag{4.4b}$$

Similarly, the odd part of the 8-point transform is performed using (4.5a)–(4.5d) incurring five shifts and four additions.

$$96d = (64 +^{1} 32)d \tag{4.5a}$$

$$72d = (64 +^{2} 8)d \tag{4.5b}$$

$$48d = (64 - 16)d = (32 +^{3} 16)d \tag{4.5c}$$

$$20d = (16 +^{4} 4)d \tag{4.5d}$$

The odd-part of the 16-point transform is implemented as shown by (4.6a)–(4.6g) costing five shifts and six additions. Note that in (4.6d), although 56*d* can equally be implemented by 40*d* + 16*d* utilising the already calculated 40*d* (4.6e), it would incur a two-stage adder (as 40*d* + 16*d* would be calculated sequentially after (4.6e)), and could introduce a larger path delay in the data flow between the input, *d*, and the required output, 56*d*.

$$96d = (64 +^{1} 32)d \tag{4.6a}$$

$$80d = (64 +^{2} 16)d \tag{4.6b}$$

$$68d = (64 +^{3} 4)d \tag{4.6c}$$

$$56d = (64 -^{4} 8)d \tag{4.6d}$$

$$40d = (32 +^{5} 8)d \tag{4.6e}$$

$$28d = (32 -^{6} 4)d \tag{4.6f}$$

$$8d = d << 3 \tag{4.6g}$$

The odd-part of the 32-point transform implemented using (4.7a)–(4.7l) requires five shifts and ten additions/subtractions. Implementing the transform with (4.7a)–(4.7f) using the most right-hand side of these equations would involve only four shifts and ten additions. However, this would incur a seven-stage adder as the additions in (4.7a)–(4.7f) have to be performed serially after calculating (4.7g), thus increasing the critical path delay (cpd) and reducing the applicable maximum clock frequency. Although this seven-stage of adders can be implemented in a two- or three-pipeline fashion to reduce the cpd, this would complicate the overall control operation as the other transforms (4-/8-/16-point) are executable in only a single cc. Furthermore, the savings attainable by the latter configuration in this 32-point case is too small (only one shift) as opposed to the first configuration.

$$96d = (64 +^{1} 32)d = 80d + 16d \tag{4.7a}$$

$$80d = (64 +^{2} 16)d = 72d + 8d \tag{4.7b}$$

$$72d = (64 +^{3} 8)d = 68d + 4d \tag{4.7c}$$

$$68d = (64 +^{4} 4)d = 60d + 8d \tag{4.7d}$$

$$60d = (64 -^{5} 4)d = 56d + 4d \tag{4.7e}$$

$$56d = (64 -^{6} 8)d = 48d + 8d \tag{4.7f}$$

$$48d = (32 +^{7} 16)d \tag{4.7g}$$

$$40d = (32 +^{8} 8)d \tag{4.7h}$$

$$32d = d << 5 \tag{4.7i}$$

$$24d = (16 +^{9} 8)d \tag{4.7j}$$

$$12d = (8 +^{10} 4)d \tag{4.7k}$$

$$4d = d << 2 \tag{4.7l}$$

Table 4.5 shows the arithmetic complexity in a multiplier-free implementation of N-point/$N \times N$ 1-D approximated core transform using even–odd decomposition and MCM. When compared with Table 3.7, the numbers of adders/subtractors in the adder tree and add/sub part are the same. The only differences are in the numbers of shifts and adders/subtractors in the multiplier replacement, thus affecting the total number of adders/subtractors. Table 4.6 shows that about 13.8% savings in the total number of shifts and around 27.2% in the total number of adders/subtractors could be achieved by the chosen approximated core transform in comparison to the original HEVC transform matrices.

Table 4.5 Complexity in multiplier-free N-point/$N \times N$ 1-D approximated core transform using even–odd decomposition and Multiple-Constant Multiplication (MCM)

Size	**Multipliers**		**Shifts**	**Adders/Subtractors**			
	Element	**Quantity**		**Multiplier Replacement**	**Adder Tree**[1]	**Add/Sub part**	**Total**
4-point	64	2	2 * 1	0	-	-	-
	80	2	2 * 4	2 * 2	-	-	-
	36	2			-	-	-
	Total	6	**10**	4	4	4	**12**
8-point (odd rows)	96	4	4 * 5	4 * 4	-	-	-
	72	4			-	-	-
	48	4			-	-	-
	20	4			-	-	-
	Total	16	**20**	16	12	8	**36**

	90	8			-	-	-
	87	8			-	-	-
	80	8			-	-	-
16-point (odd rows)	70	8	8 * 5	8 * 6	-	-	-
	57	8			-	-	-
	43	8			-	-	-
	25	8			-	-	-
	9	8			-	-	-
	Total	64	**40**	48	56	16	**120**
	90	32			-	-	-
	88	16			-	-	-
	85	16			-	-	-
	82	16			-	-	-
	78	16			-	-	-
	73	16			-	-	-
	67	16			-	-	-
32-point (odd rows)	61	16	16 * 5	16 * 10	-	-	-
	54	16			-	-	-
	46	16			-	-	-
	38	16			-	-	-
	31	16			-	-	-
	22	16			-	-	-
	13	16			-	-	-
	4	16			-	-	-
	Total	256	**80**	160	240	32	**432**
Total		342	**150**	232	312	60	**600**

Table 4.6 Computational savings in multiplier-free N-point/$N \times N$ 1-D approximated core transform using even–odd decomposition and Multiple-Constant Multiplication (MCM)

Size	Forward Transform			
	HEVC		Approximated, T16	
	Shifts	Adds	Shifts (Savings)	Adds (Savings)
4-point	10	16	10 (0.00%)	12 (25.0%)
8-point (odd rows)	20	48	20 (0.00%)	36 (25.0%)
16-point (odd rows)	48	168	40 (16.7%)	120 (28.6%)
32-point (odd rows)	96	592	80 (16.7%)	432 (27.0%)
Total	**174**	**824**	**150 (13.8%)**	**600 (27.2%)**

4.2.4 Transform and Intermediate Scaling

As one of the criteria of the chosen integers in the approximated transform matrices is multiples of four, these integers are scaled down by four and this common factor is absorbed in the subsequent intermediate scaling. Therefore, the odd part of the 4-point approximated and scaled transform is implemented as (4.8a)–(4.8b) involving three shifts and two additions.

$$20d = (16 +^{1} 4)d \tag{4.8a}$$

$$9d = (8 +^{2} 1)d \tag{4.8b}$$

Similarly, the odd-part of the 8-point transform is calculated using (4.9a)–(4.9d) incurring four shifts and four additions.

$$24d = (16 +^{1} 8)d \tag{4.9a}$$

$$18d = (16 +^{2} 2)d \tag{4.9b}$$

$$12d = (8 +^{3} 4)d \tag{4.9c}$$

$$5d = (4 +^{4} 1)d \tag{4.9d}$$

The odd part of the 16-point transform is executed as shown by (4.10a)–(4.10g) costing four shifts and six additions.

$$24d = (16 +^{1} 8)d \tag{4.10a}$$

$$20d = (16 +^{2} 4)d \tag{4.10b}$$

$$17d = (16 +^{3} 1)d \tag{4.10c}$$

$$14d = (16 -^{4} 2)d \tag{4.10d}$$

$$10d = (8 +^{5} 2)d \tag{4.10e}$$

$$7d = (8 -^{6} 1)d \tag{4.10f}$$

$$2d = d \ll 1 \tag{4.10g}$$

The odd part of the 32-point transform is implemented using (4.11a)–(4.11l) requiring four shifts and ten additions/subtractions.

$$24d = (16 +^{1} 8)d \tag{4.11a}$$

$$20d = (16 +^{2} 4)d \tag{4.11b}$$

$$18d = (16 +^{3} 2)d \tag{4.11c}$$

$$17d = (16 +^{4} 1)d \tag{4.11d}$$

$$15d = (16 -^{5} 1)d \tag{4.11e}$$

$$14d = (16 -^{6} 2)d \tag{4.11f}$$

$$12d = (8 +^{7} 4)d \tag{4.11g}$$

$$10d = (8 +^{8} 2)d \tag{4.11h}$$

$$8d = d \ll 3 \tag{4.11i}$$

$$6d = (4 +^{9} 2)d \tag{4.11j}$$

$$3d = (2 +^{10} 1)d \tag{4.11k}$$

$$1d = d \tag{4.11l}$$

Table 4.7 shows the arithmetic complexity in a multiplier-free implementation of N-point/$N \times N$ 1-D approximated and scaled core transform using even–odd decomposition and MCM. When compared with Table 4.5, the numbers of adders/subtractors are the same. The only difference is in the number of shifts. Table 4.8 shows that a further 20.0% savings could be attained in the total number of shifts by the approximated and scaled core transform, namely in this thesis as ST16, in comparison to the approximated transform matrices, namely T16. ST16 requires 31.0% fewer shifts and 27.2% fewer additions/subtractions when compared with the original HEVC core transform.

Table 4.7 Complexity in multiplier-free N-point/$N \times N$ 1-D approximated and scaled core transform using even–odd decomposition and Multiple-Constant Multiplication (MCM)

Size	**Multipliers**		**Shifts**	**Adders/Subtractors**			
	Element	**Quantity**		**Multiplier Replacement**	**Adder Tree**[1]	**Add/Sub part**	**Total**
4-point	16	2	2 * 1	0	-	-	-
	20	2	2 * 3	2 * 2	-	-	-
	9	2			-	-	-
	Total	6	**8**	4	4	4	**12**
8-point (odd rows)	24	4	4 * 4	4 * 4	-	-	-
	18	4			-	-	-
	12	4			-	-	-
	5	4			-	-	-
	Total	16	**16**	16	12	8	**36**

	24	8			-	-	-
	20	8			-	-	-
	20	8			-	-	-
16-point (odd rows)	17	8	8 * 4	8 * 6	-	-	-
	14	8			-	-	-
	10	8			-	-	-
	7	8			-	-	-
	2	8			-	-	-
	Total	64	**32**	48	56	16	**120**
	24	32			-	-	-
	20	16			-	-	-
	20	16			-	-	-
	20	16			-	-	-
	20	16			-	-	-
	18	16			-	-	-
	17	16			-	-	-
32-point (odd rows)	15	16	16 * 4	16 * 10	-	-	-
	14	16			-	-	-
	12	16			-	-	-
	10	16			-	-	-
	8	16			-	-	-
	6	16			-	-	-
	3	16			-	-	-
	1	16			-	-	-
	Total	256	**64**	160	240	32	**432**
Total		342	**120**	232	312	60	**600**

Table 4.8 Computational savings in multiplier-free N-point/$N \times N$ 1-D approximated and scaled core transform using even–odd decomposition and Multiple-Constant Multiplication (MCM)

Size	**Forward Transform**					
	HEVC		**Approximated, T16**		**Approximated and Scaled, ST16**	
	Shifts	**Adds**	**Shifts (Savings1[a])**	**Adds (Savings1[a])**	**Shifts (Savings1[a]) (Savings2[b])**	**Adds (Savings1[a])**
4-point	10	16	10 (0.0%)	12 (25.0%)	8 (20.0%) (20.0%)	12 (25.0%)
8-point (odd rows)	20	48	20 (0.0%)	36 (25.0%)	16 (20.0%) (20.0%)	36 (25.0%)
16-point (odd rows)	48	168	40 (16.7%)	120 (28.6%)	32 (33.3%) (20.0%)	120 (28.6%)
32-point (odd rows)	96	592	80 (16.7%)	432 (27.0%)	64 (33.3%) (20.0%)	432 (27.0%)
Total	**174**	**824**	**150 (13.8%)**	**600 (27.2%)**	**120 (31.0%) (20.0%)**	**600 (27.2%)**

[a] Savings against HEVC transform

[b] Savings against approximated transform, T16

As the approximated transform matrices are scaled down by four (2^2), the intermediate scaling factors after each 1-D transform operation can be reduced by two bits, as shown in Table 4.9. The corresponding scaling factors in the inverse transform are not affected.

Table 4.9 Intermediate scaling factors in 2-D forward approximated and scaled transform

Stage	**Scaling Factor**
First forward transform	$2^{(4 + M/2)}$
After the first forward transform, S_{T1}	$2^{-(B + M - 7)}$
Second forward transform	$2^{(4 + M/2)}$
After the second forward transform, S_{T2}	$2^{-(M + 4)}$
Total scaling for forward transform	$2^{(11 - B - M)}$

4.3 Approximated Forward Quantisation

From Eq. (3.26), HEVC forward quantisation proposed by Budagavi *et al.* (Budagavi, Fuldseth and Bjøntegaard, 2014) comprises:

i. Multiplications by quantiser multipliers, f_i, and a ratio of frequency-dependent scaling, $16/w[r][c]$, at the specific transform coefficient location (r, c);
ii. Addition of an offset;
iii. Divisions by QP/6 and quantisation scale factor, S_Q.

The addition requires an *n*-bit adder, depending on the required precision, while the two divisions are performed by means of right bit shifts (opposite of left bit shifts for a multiplication), and therefore are not too resource demanding. On the other hand, multiplications using *n*-bit multipliers would be area consuming in particular when implemented on hardware. For this reason, the multiplication operation in the forward quantisation was selected to be simplified to reduce its complexity. In addition, only the quantiser multipliers, f_i, are considered in this thesis, as these multipliers are 15-bit numbers. The multiplication with the ratio of quantisation weight involves a calculation with the value of one or less, as $w[r][c]$ is equal to 16 or more, and therefore not considered in this work.

Following the same approach of multiplier-free implementation as in the transform stage, the multiplication of a transform coefficient, *coeff*, by f_i can be performed as shown in (4.12a)–(4.12f), costing 14 bit shifts and 22 adds/subs.

$$26214\mathit{coeff} = (((2^{14} + 2^{13}) + (2^{10} + 2^{9})) + (((2^{6} + 2^{5}) + (2^{2} + 2^{1}))\mathit{coeff}$$

$$= (((2^{14} +^{1} 2^{13}) +^{15} (2^{10} +^{5} 2^{9})) +^{20} (((2^{6} +^{10} 2^{5}) +^{12} 2^{2}) +^{14} 2^{1}))\mathit{coeff} \tag{4.12a}$$

$$23302\mathit{coeff} = (((2^{14} +^{2} 2^{12}) +^{16} (2^{11} +^{6} 2^{9})) +^{21} ((2^{8} +^{11} 2^{2}) +^{13} 2^{1}))\mathit{coeff} \tag{4.12b}$$

$$20560\mathit{coeff} = ((2^{14} +^{2} 2^{12}) +^{17} (2^{6} +^{7} 2^{4}))\mathit{coeff} \tag{4.12c}$$

$$18396\mathit{coeff} = ((((2^{14} + 2^{10}) + (2^{9} + 2^{8})) + ((2^{7} + 2^{6}) + (2^{4} + 2^{3}))) + 2^{2})\mathit{coeff}$$

$$= ((2^{14} +^{3} 2^{11}) -^{18} (2^{5} +^{8} 2^{2}))\mathit{coeff} \tag{4.12d}$$

$$16384\mathit{coeff} = (2^{14})\mathit{coeff} \tag{4.12e}$$

$$14564\mathit{coeff} = (((2^{13} +^{4} 2^{12}) +^{19} (2^{11} +^{9} 2^{7})) +^{22} ((2^{6} +^{10} 2^{5}) +^{12} 2^{2}))\mathit{coeff} \tag{4.12f}$$

Although the MCM technique could as well be applied sharing the resources and costing 13 bit shifts and 22 adders per transform coefficient, the longest cpd would be a three-stage adder tree to execute (4.12a), (4.12b), and (4.12f). A two-stage adder tree is required each for (4.12c) and (4.12d), while 16384*coeff* (4.12e) would simply need a single 14-bit left shift. One way to reduce the cpd is by implementing these multiplications in a two-stage pipeline consisting of a maximum of two adders per pipeline stage.

As a minor contribution, this thesis aims to search for alternative quantiser multipliers requiring fewer resources. These multipliers were approximated based on the following criteria:

i. A maximum of two-stage adder tree
ii. Yielding the minimum difference with respect to the original multiplication

Consequently, only (4.12a), (4.12b), and (4.12f) were selected to be replaced by alternative multiplier values, and (4.12c)–(4.12e) were retained as each of these multipliers requires a two-stage adder tree or less. Table 4.10 shows several of the considered alternatives. Although the search set may not be exhaustive, based on these alternatives, this thesis proposes a set of approximated quantiser multipliers, namely Q (4.13).

Table 4.10 Several alternative quantiser multipliers

Original Multiplier	**Equivalent bit shift – add operation**	**Difference (× *coeff*)**
26214 (4.12a)	$(2^{14} + 2^{13}) + (2^{10} + 2^{9}) = 26112$	–102
	$(2^{14} + 2^{13}) + (2^{11} + 2^{10}) = 27648$	1434
	$(2^{14} + 2^{13}) + 2^{11} = 26624$	410
	$(2^{14} + 2^{13}) = 24576$	–1638
	$(2^{15}) = 32768$	6554
23302 (4.12b)	$(2^{14} + 2^{12}) + (2^{11} + 2^{9}) = 23040$	–262
	$(2^{14} + 2^{12}) + (2^{11} + 2^{10}) = 23552$	250
	$(2^{14} + 2^{12}) = 20480$	–2822
	$(2^{14} + 2^{13}) = 24576$	1274
	$(2^{14} + 2^{13}) - (2^{10} + 2^{8}) = 23296$	–6
14564 (4.12f)	$(2^{13} + 2^{12}) + (2^{11} + 2^{7}) = 14464$	–100
	$(2^{13} + 2^{12}) + (2^{11} + 2^{8}) = 14592$	28
	$(2^{13} + 2^{12}) = 12288$	–2276
	$(2^{14} - 2^{9}) - (2^{8} + 2^{4}) = 15600$	1036

$$Q = [26112, 23296, 20560, 18396, 16384, 14592]^{\mathrm{T}} \tag{4.13}$$

Multiplications with Q using combinations of bit shifts and additions, as well as adopting the MCM technique as shown by (4.14a)–(4.14f) would cost 11 bit shifts and 14 adders/subtractors per transform coefficient, i.e., savings of 21.4% and 36.4% in the number of bit shifts and additions/subtractions, respectively. Although these savings are obtained only in the quantiser multipliers and may not indicate the overall resource reductions in the whole quantisation process, little contributions collectively could yield a bigger impact.

$$26112 \mathit{coeff} = (((2^{14} +^{1} 2^{13}) +^{10} (2^{10} +^{5} 2^{9}))\mathit{coeff} \tag{4.14a}$$

$$23296 \mathit{coeff} = (((2^{14} +^{2} 2^{13}) -^{11} (2^{10} +^{6} 2^{8}))\mathit{coeff} \tag{4.14b}$$

$$20560 \mathit{coeff} = ((2^{14} +^{2} 2^{12}) +^{12} (2^{6} +^{7} 2^{4}))\mathit{coeff} \tag{4.14c}$$

$$18396 \mathit{coeff} = ((2^{14} +^{3} 2^{11}) -^{13} (2^{5} +^{8} 2^{2}))\mathit{coeff} \tag{4.14d}$$

$$16384 \mathit{coeff} = (2^{14})\mathit{coeff} \tag{4.14e}$$

$$14592 \mathit{coeff} = (((2^{13} +^{4} 2^{12}) +^{14} (2^{11} +^{9} 2^{8}))\mathit{coeff} \tag{4.14f}$$

4.4 Summary

An approximated and complexity-reduced 32 × 32 forward core transform matrix, namely T16, for an HEVC encoder was first described in this chapter. In a multiplier-free, even–odd decomposition, and MCM implementation, 13.8% and 27.2% savings could be achieved in the number of bit shifts and additions/subtractions, respectively, by this approximated transform matrix in comparison to the original HEVC transform matrix. This matrix can as well be scaled down by two-bit, namely ST16, providing a further 20.0% saving in the number of bit shifts. Finally, a set of approximated quantiser multipliers were also introduced, namely Q, offering savings of 21.4% and 36.4% in the number of bit shifts and additions/subtractions, respectively, when compared with the quantiser multipliers suggested in (Budagavi, Fuldseth and Bjøntegaard, 2014). The next chapter presents the experimental results using these approximated values in HEVC reference software.

Chapter 5

Software-based Performance Evaluation of Approximated Forward Transform and Quantisation

Abstract This chapter evaluates the experimental results obtained by applying the approximated forward transform and approximated quantisation previously described, in HEVC reference software. The results are compared with the original HEVC algorithms in terms of objective quality metrics (Peak Signal to Noise Ratio (PSNR) and Bjøntegaard-Delta Bitrate (BD-rate)), and subjective observations.

5.1 Pilot Study

Prior to assessing the coding performance of the chosen approximated transform matrix, T16, a pilot study was conducted using HEVC reference software version 13.0 (HM–13.0) (JCT-VC, 2014). This was the latest version released by JCT-VC when the work presented in this thesis kick-started. For consistency, HM–13.0 was used throughout this thesis. The software was considered mature when an older version 10.0 (HM–10.0) was made available.

The approximated 32 × 32 transform matrix (and consequently including the internal smaller matrices) used in the pilot study was not T16 as described in Section 4.3, but with a slight difference in three of the entries (27th, 28th, and 29th entries). Table 5.1 compares this matrix, labelled as V, with T16 as well as HEVC, scaled and floating DCT (D_{sf}), and scaled and rounded DCT (D_{sr}) as references. V has a poorer worst-case orthogonality measure, o_{rc}, (0.0442) as opposed to T16 (0.0391), but has the same worst-case m_{rc} (0.1218) and n_{rc} (0.0605) measures as T16. It is assumed that the arithmetic savings of V and T16 over HEVC are similar (Table 4.6).

In this pilot study, 24 test video sequences were encoded. These sequences were progressively-scanned videos, formatted as YUV 4:2:0 colour space, and categorised into six classes, A–F, mainly based on their spatial resolutions. Sequences in classes A–D are natural videos of resolutions 2560 × 1600, 1920 ×

1080, 832 × 480, and 416 × 240, respectively. Class E samples are 1280 × 720 video conferencing sequences, while class F comprises a video of resolution 832 × 480, one with 1024 × 768, and two with 1280 × 720 containing graphical screen contents (Table 5.2).

Table 5.1 Comparison of approximated transform matrix in pilot study, V with T16, HEVC, D_{sf} and D_{sr}

First column	**Transform Alternatives**				
	D_{sf}	**D_{sr}**	**HEVC**	**V**	**T16**
0	64.00	64	64	64	64
1	90.40	90	90	96	96
2	90.07	90	90	96	96
3	89.53	90	90	96	96
4	88.77	89	89	96	96
5	87.80	88	88	80	80
6	86.61	87	87	80	80
7	85.22	85	85	80	80
8	83.62	84	83	80	80
9	81.82	82	82	80	80
10	79.82	80	80	80	80
11	77.63	78	78	80	80
12	75.26	75	75	72	72
13	72.70	73	73	72	72
14	69.96	70	70	68	68
15	67.06	67	67	68	68
16	64.00	64	64	64	64
17	60.78	61	61	60	60
18	57.42	57	57	56	56
19	53.92	54	54	56	56
20	50.28	50	50	48	48
21	46.53	47	46	48	48
22	42.67	43	43	40	40
23	38.70	39	38	40	40
24	34.64	35	36	36	36
25	30.49	30	31	32	32
26	26.27	26	25	24	28
27	21.99	22	22	20	24
28	17.66	18	18	16	20
29	13.28	13	13	12	12
30	8.87	9	9	8	8
31	4.44	4	4	4	4
o_{rc}	0.0000	0.0037	0.0029	0.0442	0.0391
m_{rc}	0.0000	0.0077	0.0213	0.1218	0.1218
n_{rc}	0.0000	0.0109	0.0013	0.0605	0.0605

Two coding profiles supported by HEVC version 1 – Main and Main 10 profiles – were applied in this pilot study. The encoding structures Random Access (RA) and Low delay with bidirectional inter-predicted (B)-frames (LB) were employed in order to simulate the entertainment and interactive applications, respectively (Ohm *et al.*, 2012). In RA, intra-predicted (I)-frames are inserted every 24, 32, 48, or 64 frames for sequences with rates of 24, 30, 50, and 60 fps, respectively. On the other hand, in LB, only the first frame is an I-frame while the rest are B-frames.

A Group of Pictures (GOP) of eight was applied, and four base quantisation parameter (QP) values were considered: 22, 27, 32, and 37, for the I-frames encoding. The chosen QP values represent normal quantisation within the whole range of supported QPs (Bossen, 2013). Hierarchical bidirectional-coding was also enabled, with a QP offset value of one between each temporal level. These settings are according to the common test conditions (CTC) set by JCT-VC (Bossen, 2013) and similar to the ones applied in (Grois *et al.*, 2013). Table 5.3 summarises the experimental settings applied in this pilot study.

Table 5.2 Test video sequences used in pilot study on approximated transform, V

Class	Sequence	Resolution	Frame rate (fps)
A	*A1 – Traffic*	2560 × 1600	30
	A2 – PeopleOnStreet		30
	A3 – Nebuta		60
	A4 – SteamLocomotive		60
B	*B1 – Kimono*	1920 × 1080	24
	B2 – Parkscene		24
	B3 – Cactus		50
	B4 – BasketballDrive		50
	B5 – BQTerrace		60
C	*C1 – BasketballDrill*	832 × 480	50
	C2 – BQMall		60
	C3 – PartyScene		50
	C4 – RaceHorses		30
D	*D1 – BasketballPass*	416 × 240	50
	D2 – BQSquare		60
	D3 – BlowingBubbles		50
	D4 – RaceHorses		30

Class	Sequence	Resolution	Frame rate (fps)
E	*E1 – FourPeople*	1280 × 720	60
	E2 – Johnny		60
	E3 – KristenAndSara		60
F	*F1 – BasketballDrillText*	832 × 480	50
	F2 – ChinaSpeed	1024 × 768	30
	F3 – SlideEditing	1280 × 720	30
	F4 – SlideShow	1280 × 720	20

Table 5.3 Experimental settings in pilot study on approximated transform, V

HEVC Reference Software Version	13.0 (HM–13.0)
Profiles	Main and Main 10
Encoding Structures	Random Access (RA) and Low Delay with B-frames (LB)
Test Video Sequences	24 (Table 5.2)
Intra-period for Random Access	24, 32, 48, or 64
Group of Pictures (GOP)	8
Hierarchical Bidirectional Coding	Enabled
Base Quantisation Parameters (QP)	22, 27, 32, 37
QP offset between temporal level	+1

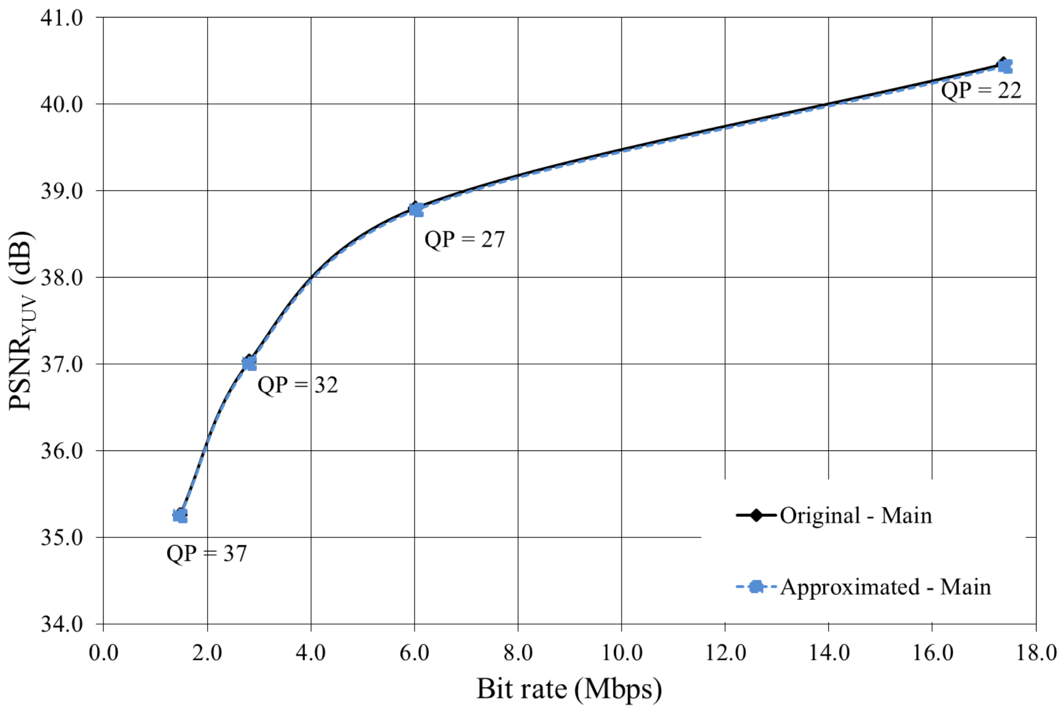
5.1.1 Peak Signal to Noise Ratio (PSNR)

For every test video sequence and QP value, the average Peak Signal to Noise ratio, PSNR_{YUV} was calculated as a weighted sum of the individual component average PSNR_{Y}, PSNR_{U}, and PSNR_{V} according to (5.1) (Ohm *et al.*, 2012).

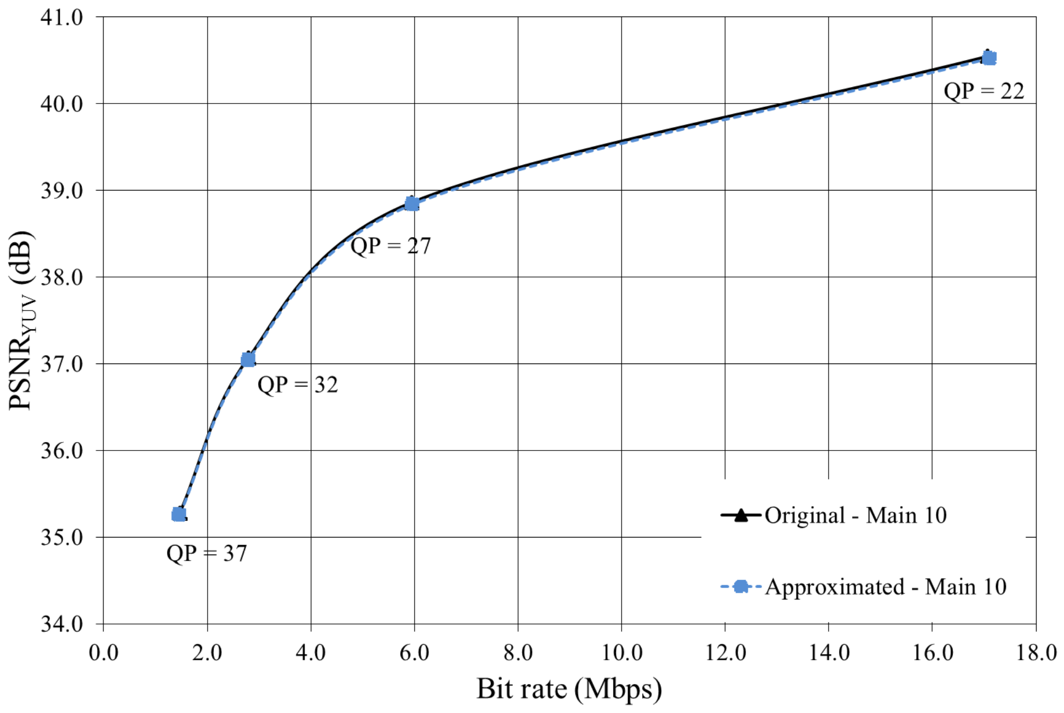
$$\text{PSNR}_{\text{YUV}} = (6 \cdot \text{PSNR}_{\text{Y}} + \text{PSNR}_{\text{U}} + \text{PSNR}_{\text{V}}) / 8 \tag{5.1}$$

Figs. 5.1 and 5.2 present the PSNR-based rate-distortion (R-D) curves under the RA and LB configurations, respectively, for a Class B sequence, *B4 – BasketballDrive*. A difference between two corresponding R-D_{PSNR} curves using the original HEVC transform and approximated pilot transform, V, is hardly noticeable as the two R-D curves almost overlap each other for this sequence in both Main and

Main 10 profiles. This is a typical observation of the tested video sequences in this pilot study, suggesting that in the presence of a normal quantisation, the four simplified transform matrices, V, could provide an equivalent quality-coding performance to the original HEVC set in the entertainment and interactive scenarios.

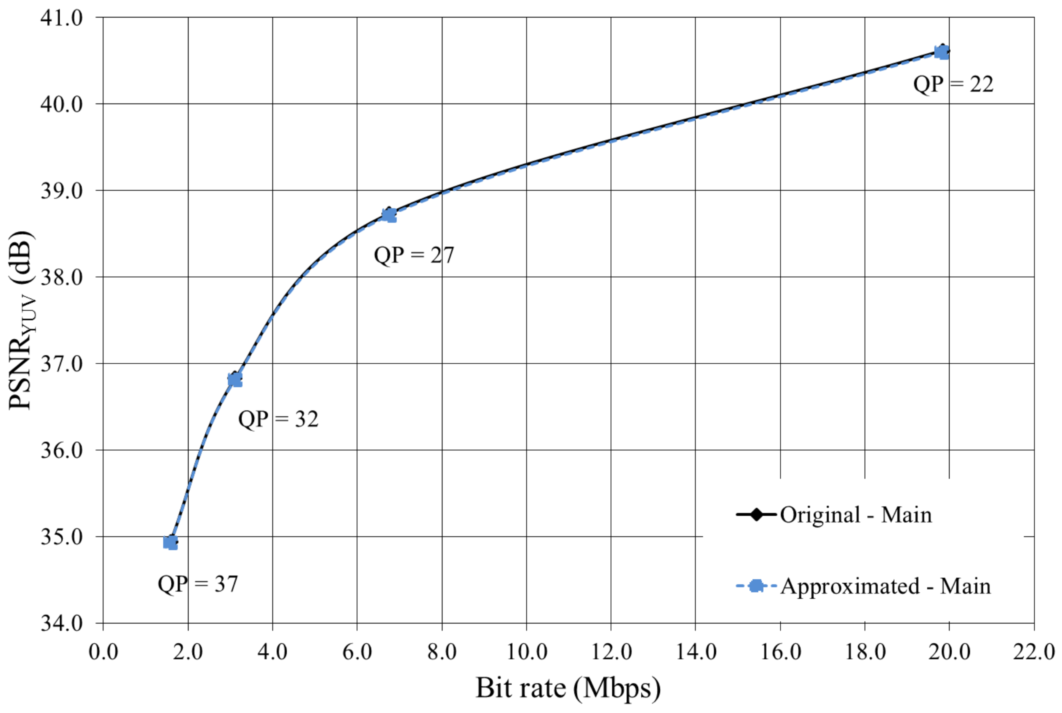


(a)

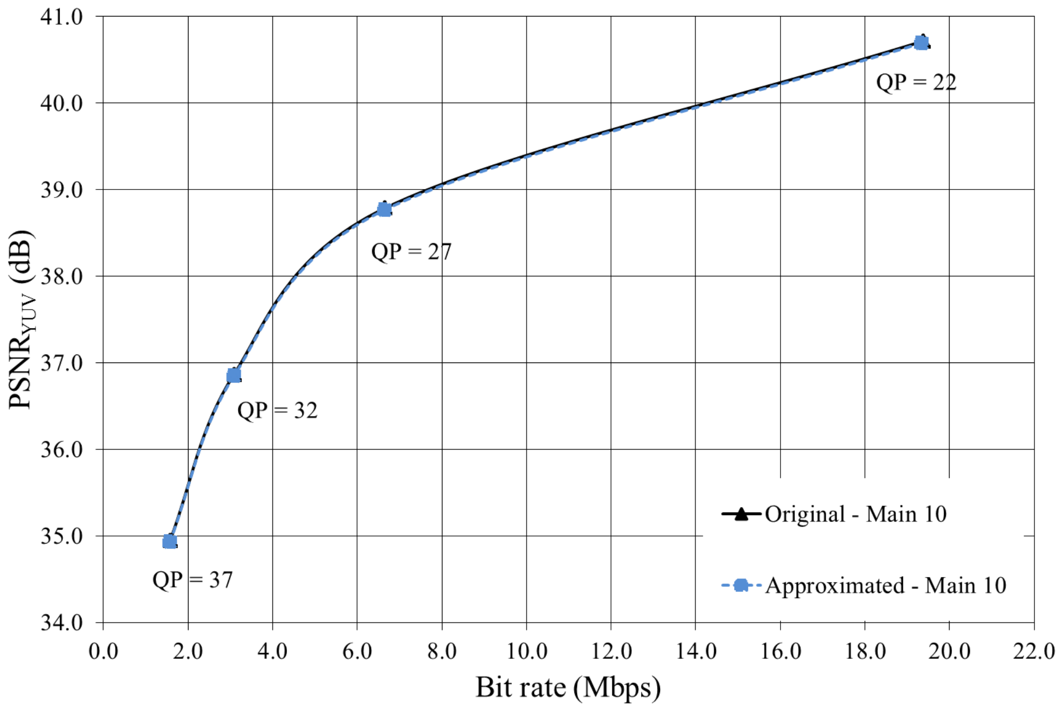


(b)

Fig. 5.1 R-D$_{\text{PSNR}}$ curves of *B4 – BasketballDrive* sequence using original (HEVC) and approximated (V) transform matrices under RA configuration and in (a) Main and (b) Main 10 profiles



(a)



(b)

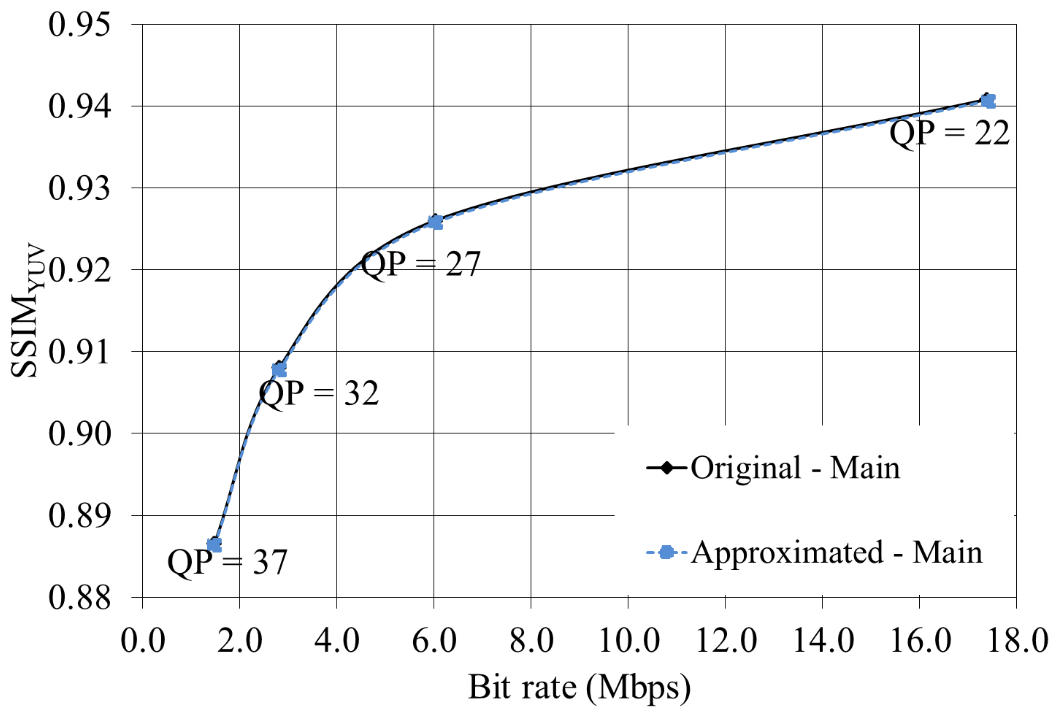
Fig. 5.2 R-D$_{PSNR}$ curves of *B4 – BasketballDrive* sequence using original (HEVC) and approximated (V) transform matrices under LB configuration and in (a) Main and (b) Main 10 profiles

5.1.2 Structural Similarity (SSIM) Index

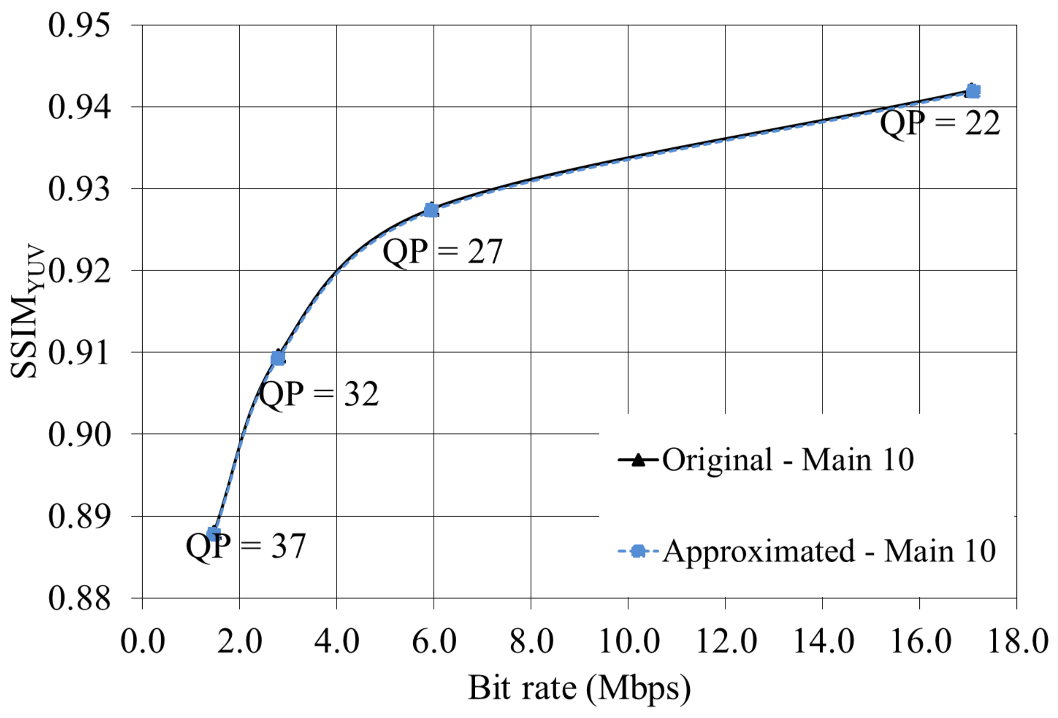
To complement PSNR, structural similarity (SSIM) index measurement was also included in this study. SSIM is a combination of luminance, contrast, and structure comparison functions (Wang *et al.*, 2004). It has a scale between zero and one, where readings that are closer to one indicate high-quality samples. It has been shown that SSIM has better correlations with subjective Mean Opinion Score (MOS) than PSNR (Wang *et al.*, 2004).

Similar to $PSNR_{YUV}$, the mean $SSIM_{YUV}$ was calculated as a weighted sum of individual $SSIM_Y$, $SSIM_U$, and $SSIM_V$ means for each test video sequence and QP value (5.2). Figs. 5.3 and 5.4 display the corresponding R-D curves for *B4 – BasketballDrive* sequence in terms of $SSIM_{YUV}$. Again, a difference between two R-D_{SSIM} curves using HEVC and V is not obvious for this sequence as both R-D_{SSIM} curves almost overlap each other in both Main and Main 10 profiles.

$$SSIM_{YUV} = (6 \cdot SSIM_Y + SSIM_U + SSIM_V) / 8 \qquad (5.2)$$

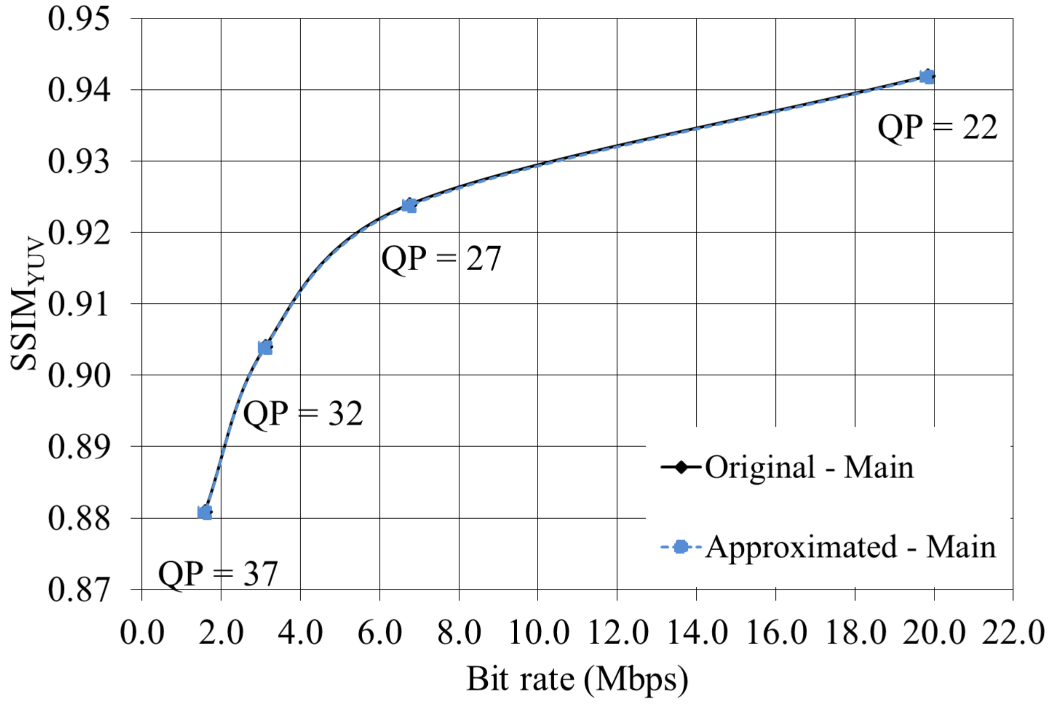


(a)

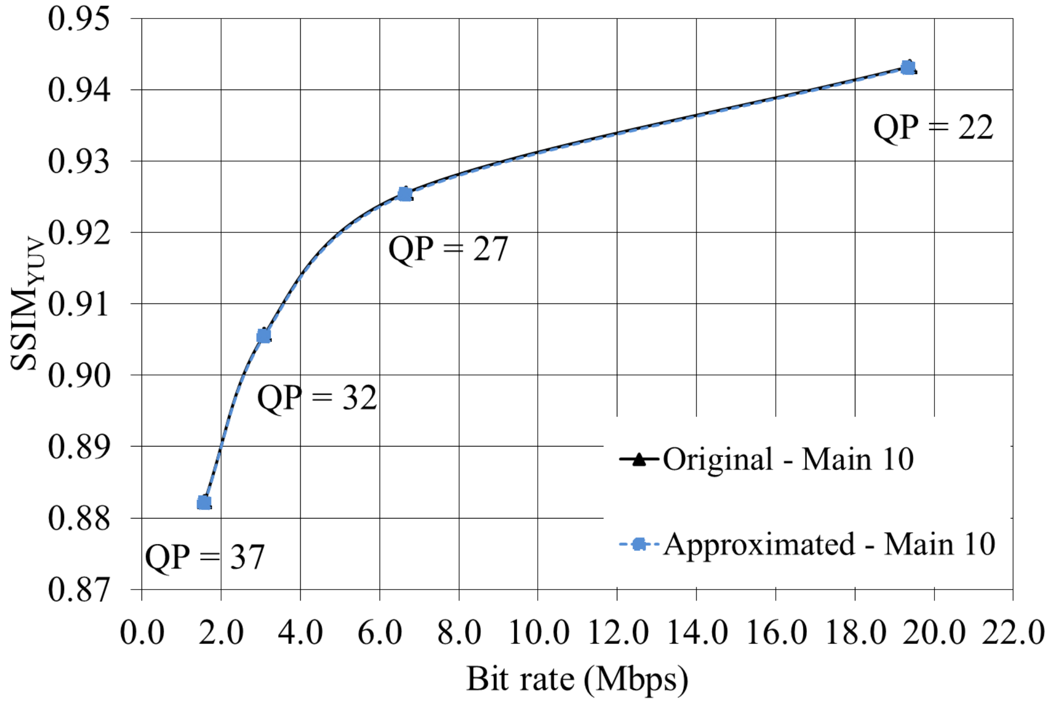


(b)

Fig. 5.3 R-D_{SSIM} curves of *B4 – BasketballDrive* sequence using original (HEVC) and approximated (V) transform matrices under RA configuration and in (a) Main and (b) Main 10 profiles



(a)



(b)

Fig. 5.4 R-D$_{SSIM}$ curves of *B4 – BasketballDrive* sequence using original (HEVC) and approximated (V) transform matrices under LB configuration and in (a) Main and (b) Main 10 profiles

5.1.3 Bjøntegaard-Delta Bitrate (BD-rate)

Bjøntegaard-delta bitrate (BD-rate) measurement (Bjøntegaard, 2008) was subsequently used to calculate the average bitrate difference between two R-D curves at the same objective $PSNR_{YUV}$ and $SSIM_{YUV}$ of the reference R-D curve. A positive BD-rate value indicates an increase in the bitrate to achieve the same $PSNR_{YUV}$ or $SSIM_{YUV}$ quality, while a negative BD-rate value means a saving in the bitrate and therefore favourable.

A summary of average BD-rate levels for each class is presented in Table 5.4 for RA and LB encoding structures in Main profile. There are positive BD-rate values in all classes, with the overall average BD-rate values of 1.1% and 0.6% for the RA and LB case, respectively. Similar observations could be seen in Main 10 profile. Although these increases may not be regarded as small penalties from the video coding perspective, they are far lower than the complexity savings anticipated by the approximated matrices as presented earlier in Table 4.6.

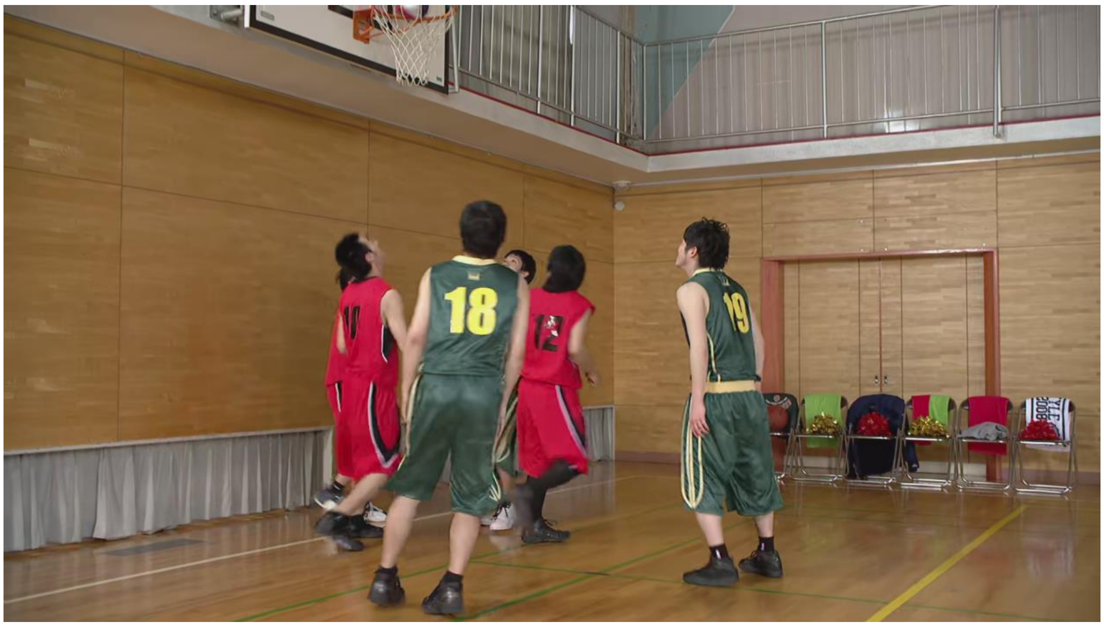
Table 5.4 Average BD-rate values (%) for equal $PSNR_{YUV}$ and $SSIM_{YUV}$ between the original (HEVC) and approximated (V) transform matrices in Main profile

Class	**Random Access (RA)**		**Low Delay (LB)**	
	$PSNR_{YUV}$	**$SSIM_{YUV}$**	**$PSNR_{YUV}$**	**$SSIM_{YUV}$**
A	1.8	1.6	-	-
B	1.5	1.7	0.7	0.8
C	0.7	0.7	0.5	0.5
D	0.7	0.6	0.4	0.4
E	-	-	0.6	0.8
F	0.5	0.6	0.6	0.5
Overall	1.1	1.1	0.6	0.6

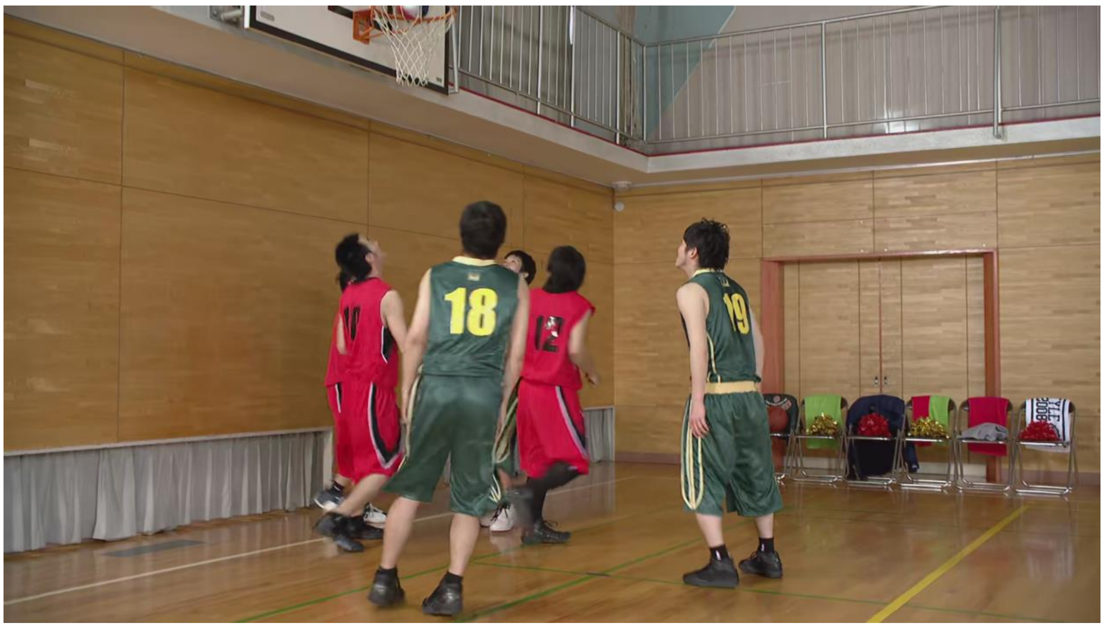
5.1.4 Visual Observations

Fig. 5.5 provides the snapshots of the last frame of *B4 – BasketballDrive* sequence encoded and reconstructed using (a) the original (HEVC) and (b) the approximated (V) transform matrices under RA configuration in Main profile. The base QP for intra-frame coding in this example was 32, and due to the applied QP offsets in the hierarchical bidirectional-coding settings, the QP value of the last frame of this sequence was increased by +4, i.e., QP(frame 499) = 36. The visual qualities of both snapshots appear to be identical, and this observation is supported by the corresponding objective values of $PSNR_{YUV}$ and $SSIM_{YUV}$ as provided in the caption of Fig. 5.5. A slight bitrate increment was obtained using V transform matrices, as indirectly reflected by Table 5.4 earlier.

In order to better evaluate the quality difference between the two sample frames in Fig. 5.5, Fig. 5.6 displays (a) the image difference and (b) the histogram of pixel differences of the last frame encoded and reconstructed using V transform matrices over HEVC. Most differences appear to be around object edges with the vast majority (96.93%) of them having values of 25 or less, i.e., not exceeding 10% of the dynamic range (256 for 8-bit image or video). The pixel differences near these edges are primarily attributed to transform matrices of lower sizes (4×4 and 8×8) whereas transforms of higher sizes (16×16 and 32×32) are more applicable in large homogeneous areas of a picture.

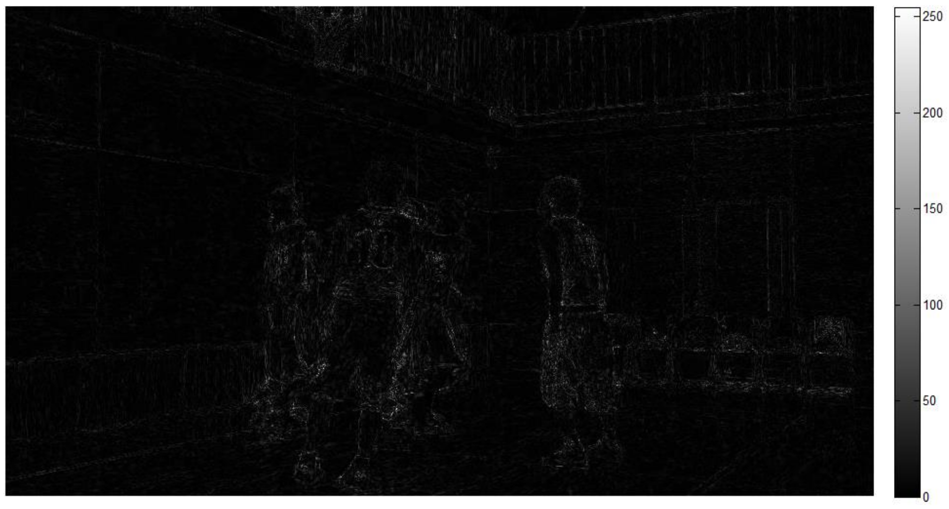


(a)

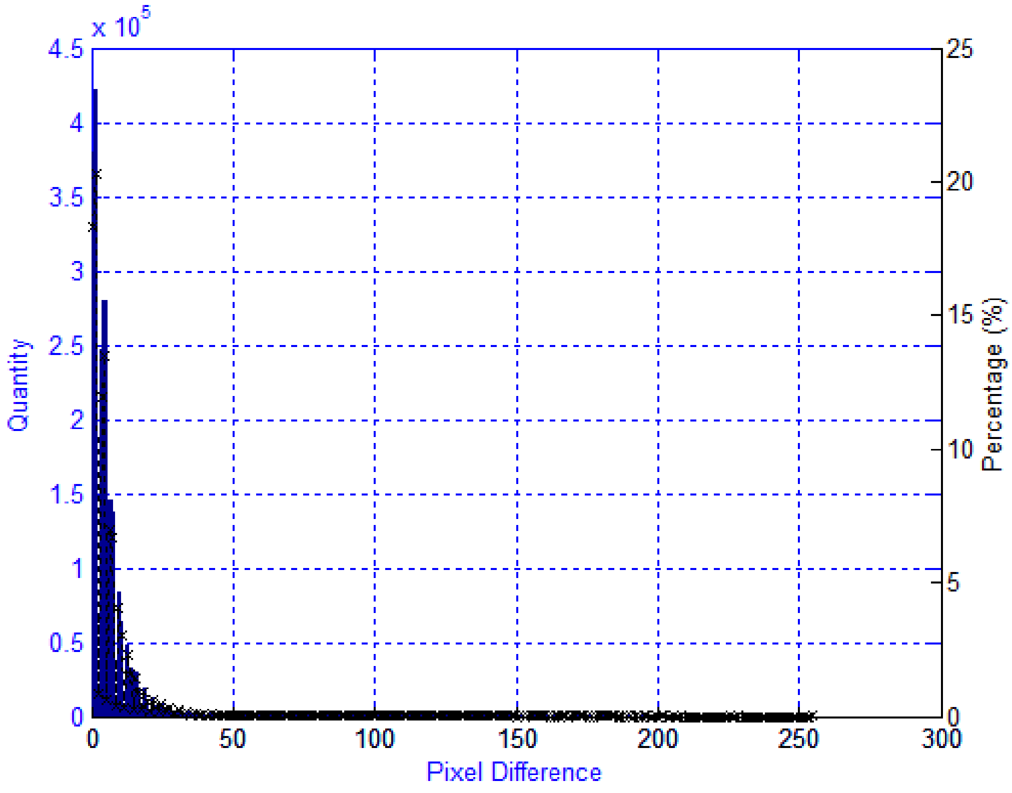


(b)

Fig. 5.5 Snapshots of *B4 – BasketballDrive* sequence (frame 499, RA, Main, QP = 36) using (a) original (HEVC) (17552 bits, $PSNR_{YUV}$ = 35.839 dB, $SSIM_{YUV}$ = 0.9041) and (b) approximated (V) (17560 bits, $PSNR_{YUV}$ = 35.822 dB, $SSIM_{YUV}$ = 0.9041) transform matrices



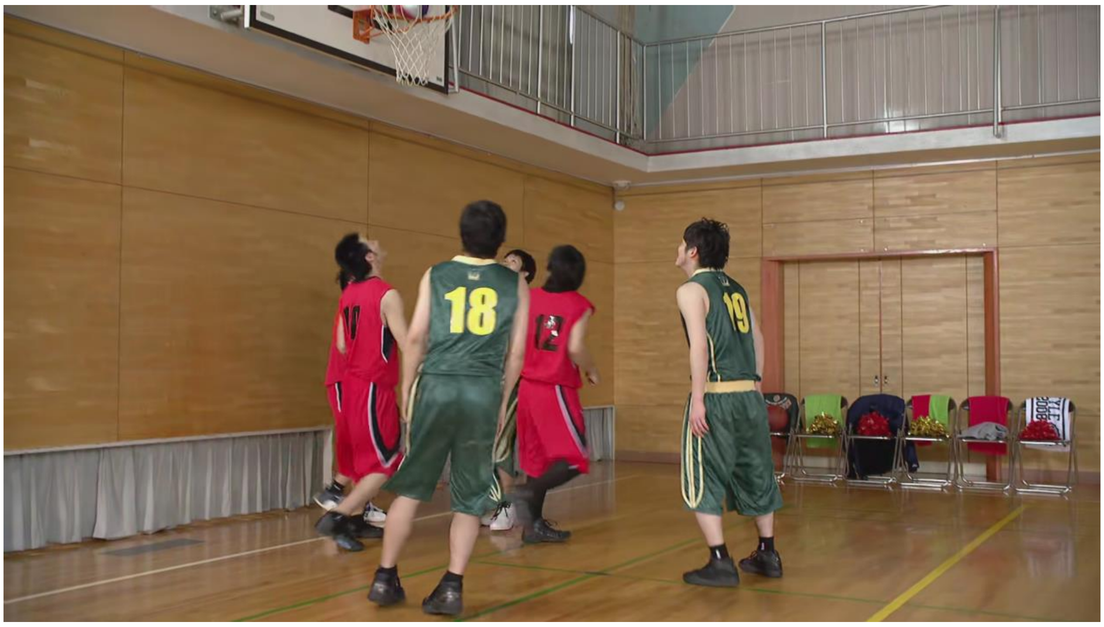
(a)



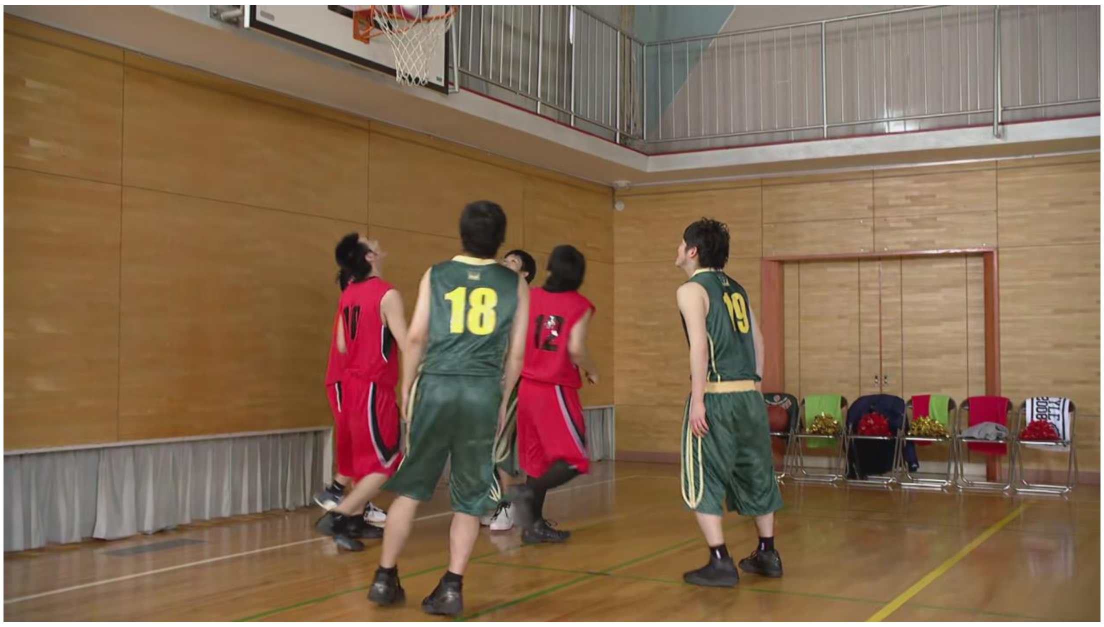
(b)

Fig. 5.6 (a) Image difference and (b) histogram of pixel differences of the last frame (frame 499, RA, Main, QP = 36) of *B4 – BasketballDrive* sequence using approximated (V) transform matrices over HEVC

Figs. 5.7 and 5.8 provide the corresponding diagrams for the LB configuration, with the QP value of the last frame in this configuration increasing by +3 due to the hierarchical bidirectional-coding settings, i.e., QP(frame 499) = 35. The qualities of the frames also appear to be visually identical, with the objective PSNR_{YUV} and SSIM_{YUV} values supporting this remark (as provided in the caption of Fig. 5.7). As the LB configuration involves only a single I-frame at the beginning of the coding sequence, the bitrates are higher than in the RA case. Using approximated V transform matrices under the LB configuration also yields a higher bitrate increment than in the RA scenario due to the carried over noise. From Fig. 5.8, most pixel differences also concentrate near object edges with the vast majority (97.21%) of them having values of 25 or lower. Similar observations could be observed in Main 10 profile for both LB and RA configurations.

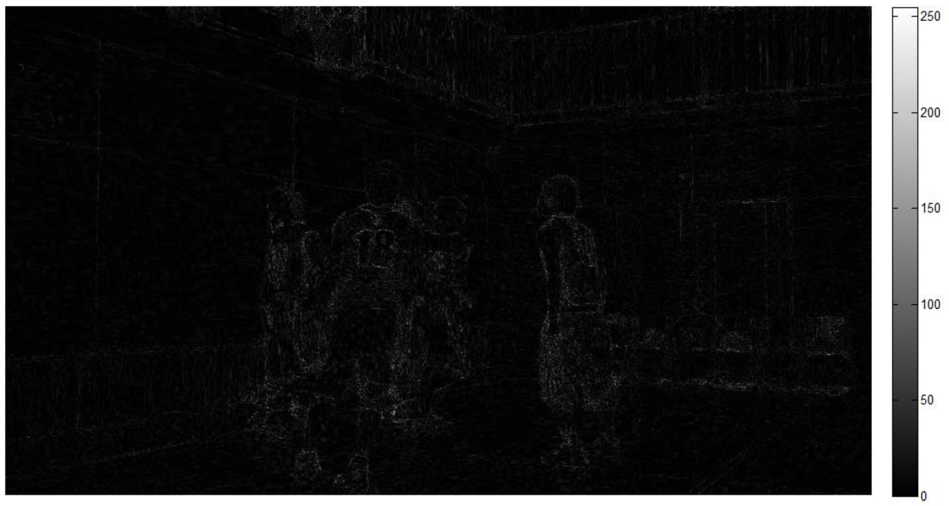


(a)

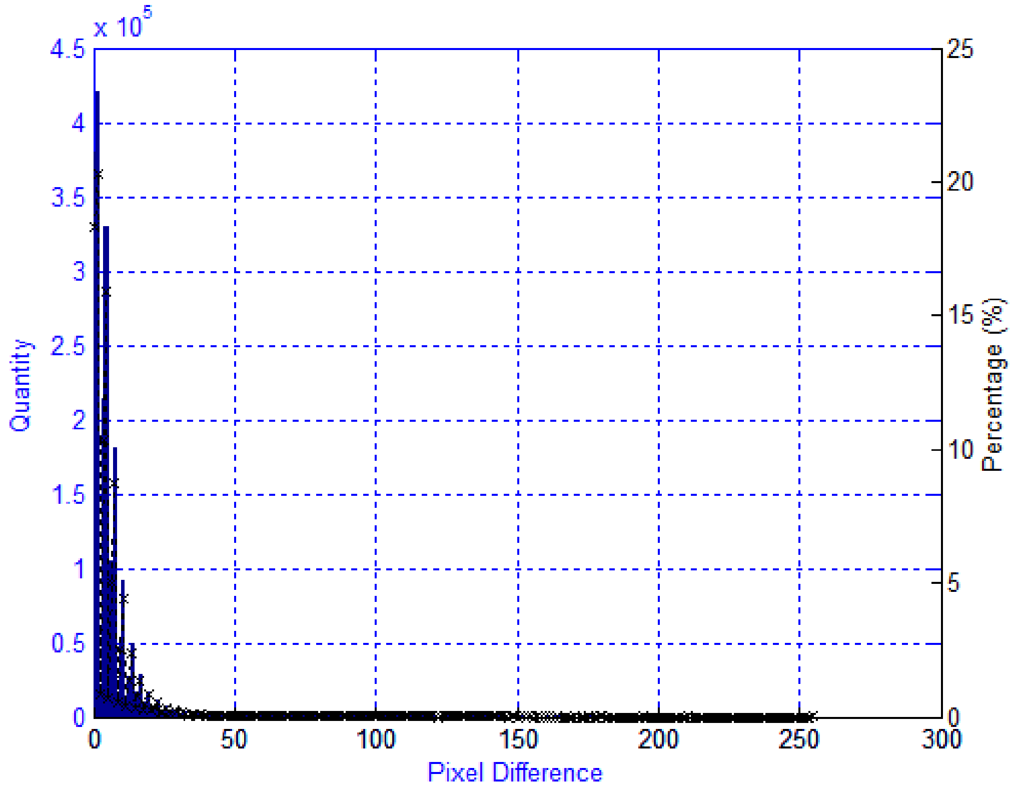


(b)

Fig. 5.7 Snapshots of *B4 – BasketballDrive* sequence (frame 499, LB, Main, QP = 35) using (a) original (HEVC) (31504 bits, $\mathrm{PSNR_{YUV}}$ = 36.186 dB, $\mathrm{SSIM_{YUV}}$ = 0.9038) and (b) approximated (V) (31824 bits, $\mathrm{PSNR_{YUV}}$ = 36.159 dB, $\mathrm{SSIM_{YUV}}$ = 0.9036) transform matrices



(a)



(b)

Fig. 5.8 (a) Image difference and (b) histogram of pixel differences of the last frame (frame 499, LB, Main, QP = 35) of *B4 – BasketballDrive* sequence using approximated (V) transform matrices over HEVC

5.1.5 Encoder-Decoder Compatibility

A major change in the forward transform matrices in the encoder requires the corresponding inverse transform matrices to be integrated into the decoder to avoid any mismatch issue. To see the negative impacts of having an incompatible forward-inverse transform operation, all bitstreams encoded with the approximated matrices, V, in Main profile for QPs of 22 and 37 were decoded using the original HM-13.0 decoder.

Table 5.5 presents the average PSNR differences (dB) of individual Y-, U-, and V-components for all classes in both RA and LB configurations for QP 22. In all cases, the decoded videos suffer PSNR drops in all three components. The most severe PSNR drops were observed in classes A, B, and E, i.e., high-resolution videos. The average drops decrease with decreasing spatial resolution with the least difference in the smallest resolution group, class D. The chrominance U- and V-components appear to be affected more, but as the HVS is more sensitive to luminance, a lower PSNR drop in the Y-component would still render an annoying viewing experience. On average, the PSNR difference of Y-component is about -5dB in both RA and LB configurations, while the drops of U- and V-components are higher in the RA configuration, about -10dB, than in the LB configuration, more than -7dB. These drops are more severe in QP 22 than QP 37. It is, therefore, inevitable to have compatible transform processing blocks in both encoder and decoder.

Table 5.5 Average PSNR differences (dB) for proposed bitstreams in Main Profile and QP 22 decoded with original HEVC decoder (HM-13.0)

Class	**Random Access (RA)**			**Low Delay (LB)**		
	Y	**U**	**V**	**Y**	**U**	**V**
A	-7.87	-19.44	-18.69	-	-	-
B	-6.21	-14.40	-14.56	-6.76	-11.36	-11.08
C	-3.12	-7.57	-7.84	-3.83	-7.67	-7.52
D	-1.66	-4.73	-4.12	-2.88	-3.02	-5.89
E	-	-	-	-7.46	-11.93	-8.76
F	-5.51	-6.46	-5.39	-4.26	-4.89	-5.11
Mean	-4.87	-10.52	-10.12	-5.03	-7.77	-7.67

5.1.6 Conclusions

From the conducted pilot study on the approximated transform for HEVC using V matrices, the following conclusions could be drawn. Positive BD-rate increments were seen on average in all six classes of test sequences, with an average of +1.1% in RA and 0.6% in LB. These positive BD-rate increments are larger in high-resolution videos (Classes A, B, and E) in comparison to low-resolution videos (Classes C, D, and F). These BD-rate differences also appear to be larger in the RA configuration as opposed to LB for natural videos (Classes B, C, and D) but about the same for class F with computer graphical contents. Similar results were obtained using PSNR and SSIM objective quality metrics. Considering natural videos, the approximated transform matrices appear to carry more bits for I-frames than B-frames with respect to the original HEVC matrices. No significant differences were seen in Main and Main 10 profiles in terms of objective metrics (PSNR and BD-rate) and visual observations of the reconstructed videos. From sample video frames, most pixel differences obtained using V transform matrices against HEVC lie near object edges with the vast majority (over 96% in both configurations) of them having values of 25 or lower. Finally, employing a different set of transform matrices in the encoder would require the corresponding inverse transform matrices in the decoder to avoid highly distorted decoded videos.

5.2 Approximated Transforms

This section evaluates the coding performance of the approximated core transform matrix, T16, and its scaled version, ST16, with HEVC core transform.

5.2.1 Experimental Settings

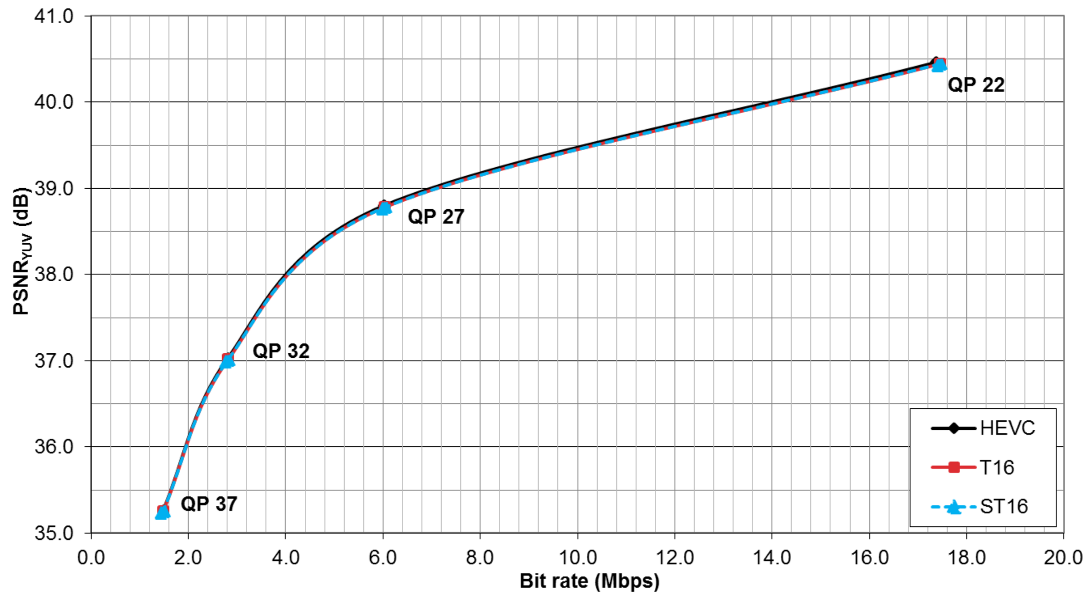
Based on the results of the pilot study, it was decided to conduct the experiments on T16 and ST16 only in Main profile. It is assumed that similar results would be obtained in Main 10 profile. In addition, this thesis puts more emphasis on high-resolution videos because the coding performance impact seen from the pilot study was higher in the high-resolution classes. Furthermore, there is a growing tendency to embed high-resolution capability in video-enabled devices due to technological advances and increasing demands, and low-resolution videos are slowly phasing out. Therefore, only Classes A (cropped UHD) and B (Full-HD 1080p) test video sequences were evaluated under RA coding structure, and Classes B and E (HD-ready 720p) videos under LB. Low-resolution videos should exhibit better coding performance results. In addition, only PSNR-based BD-rate coding performance metric was used from this experiment onwards. Similar results were expected to be obtained using SSIM. Table 5.6 shows the updated experimental settings used in this experiment.

Table 5.6 Experimental settings on approximated transform

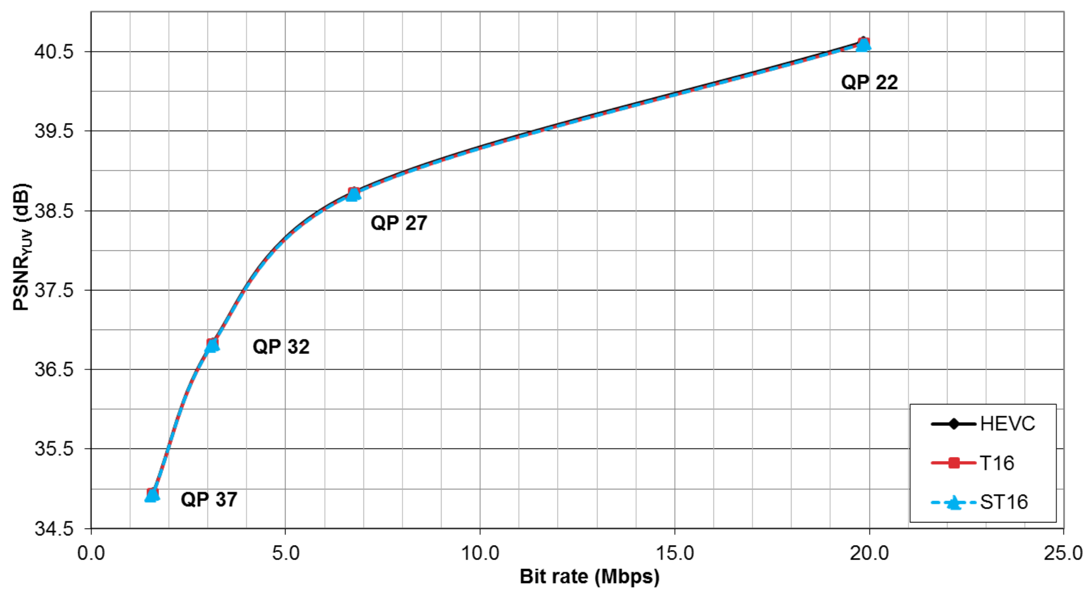
HEVC Reference Software Version	13.0 (HM–13.0)
Profiles	Main
Encoding Structures	Random Access (RA) and Low Delay with B-frames (LB)
Test Video Sequences	Classes A and B for RA Classes B and E for LB
Intra-period for RA	24, 32, 48, or 64
Group of Pictures (GOP)	8
Hierarchical Bidirectional Coding	Enabled
Base Quantisation Parameters (QP)	22, 27, 32, 37
QP offset between temporal level	+1

5.2.2 Peak Signal to Noise Ratio (PSNR)

Fig. 5.9 presents the PSNR-based rate-distortion (R-D) curves under the RA and LB configurations for a Class B sequence, *B4 – BasketballDrive*. Similar to the pilot study, the three R-D curves using the original (HEVC) and approximated (T16 and ST16) transform matrices appear to almost overlap each other in both RA and LB configurations. Similar observations could as well be seen with most other videos. The corresponding R-D curves for the other tested video sequences are provided in Appendix A. These R-D curves suggest that using a normal quantisation value (QP = 22–32), the four approximated transform matrices of T16 and ST16 may not introduce significant quality-coding performance degradations in the entertainment and interactive scenarios when compared with using the original HEVC transform set.



(a)



(b)

Fig. 5.9 R-D$_{PSNR}$ curves of *B4 – BasketballDrive* sequence using original HEVC and approximated transform matrices, T16 and ST16, under (a) RA and (b) LB configurations in Main profile

5.2.3 Bjøntegaard-Delta Bitrate (BD-rate)

A summary of average BD-rate levels for each class is presented in Table 5.7 for RA and LB encoding structures in Main profile. Again, there are unfavourable positive BD-rate values in all involved classes, with the overall average BD-rate values of 1.7% and 0.7% in the RA and LB case, respectively. The highest BD-rate differences in RA from Class A were obtained from *A3 – NebutaFestival* and *A4 – SteamLocomotive*, possibly due to these sequences being in 10-bit depth prior to conversion to 8-bit depth to be encoded in Main profile. From Class B, the largest BD-rate increases were obtained from *B1 – Kimono1* in both RA and LB configurations, possibly because there is a scene change in this sequence. Although these increases may not be regarded as small penalties from the video coding perspective, they are considerably lower than the complexity savings anticipated by the approximated transform matrices as presented earlier in Table 4.8.

Table 5.7 Average BD-rate values (%) for equal $PSNR_{YUV}$ between HEVC and approximated, T16 and ST16, transform matrices in Main profile

Class	**Sequence**	**Random Access (RA)**		**Low Delay (LB)**	
		T16	**ST16**	**T16**	**ST16**
A (2560 × 1600) Cropped UHD	*A1*	1.5	1.5	-	-
	A2	1.3	1.3	-	-
	A3	2.3	2.3	-	-
	A4	2.3	2.4	-	-
	Average	1.8	1.9	-	-
B (1920 × 1080) Full HD	*B1*	2.7	2.7	1.1	1.1
	B2	1.3	1.2	0.5	0.5
	B3	1.5	1.5	0.8	0.7
	B4	1.1	1.1	0.6	0.7
	B5	1.1	1.1	0.6	0.7
	Average	1.5	1.5	0.7	0.7

E (1280 × 720) HD-ready	*E1*	-	-	0.8	0.8
	E2	-	-	0.6	0.3
	E3	-	-	0.8	0.9
	Average	-	-	0.8	0.7
Overall[a]		1.7	1.7	0.7	0.7

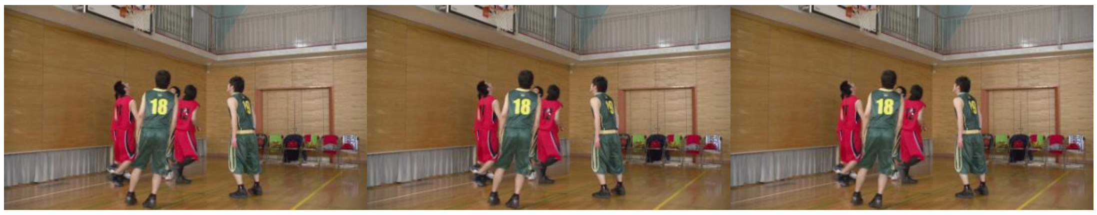
[a] Overall average of Classes A and B for RA and Classes B and E for LB

5.2.4 Visual Observations

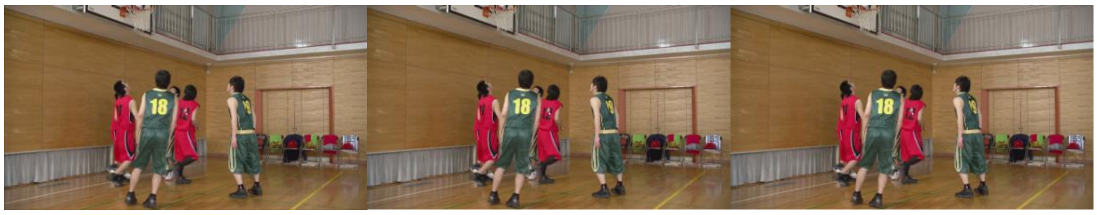
Fig. 5.10 provides the snapshots of the last frame of *B4 – BasketballDrive* sequence under the RA configuration in Main profile, encoded and reconstructed with HEVC and approximated, T16 and ST16 transform matrices, using the four base QP values (22–37). Table 5.8 equips Fig. 5.10 with the corresponding bitrate and PSNR values of the frames. The visual qualities of these snapshots look identical as supported by the objective PSNR values of all three YUV channels. Again, bitrate increments could be seen in most of the cases using T16 and ST16 approximated transform matrices.

Fig. 5.11 displays the image differences of the respective frames shown in Fig. 5.10 using T16 and ST16 matrices over HEVC. In general, the pixel differences are mainly near object edges, and these differences seem to be less visible as the QP value increases. Fig. 5.12 plots the histograms of pixel differences and Table 5.9 groups the percentages of these differences into three categories: ≤ 25; 26 – 50; > 50. For both T16 and ST16, the percentages of pixel differences having a value of 25 or lower increase with increasing QP. Consequently, pixel differences in the other two categories decrease in percentage as the QP increases. For the ≤ 25 category, while the differences between T16 and ST16 are not too obvious for QPs 27, 32, and 37, ST16 is better than T16 at QP 22. Against HEVC, both transform matrices give pixel differences not exceeding 25 in at least 88% of cases under the normal QP range being studied (22–37) and the RA configuration.

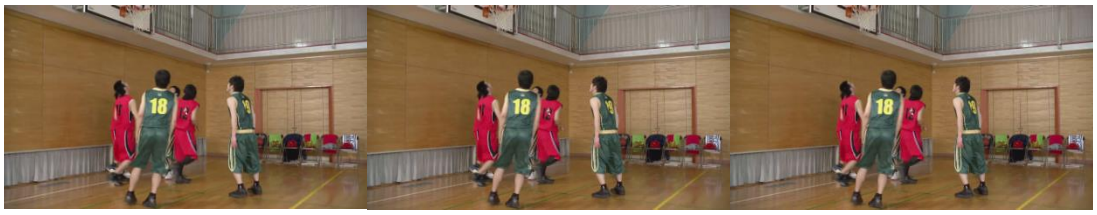
(a)



(b)



(c)



(d)

Fig. 5.10 Snapshots of the last frame of *B4 – BasketballDrive* sequence (frame 499, RA, Main) using HEVC (left column), T16 (middle column), and ST16 (right column) transform matrices and base QP values of (a) 22, (b) 27, (c) 32, and (d) 37

Table 5.8 Number of bits and PSNR values of last frame of *B4 – BasketballDrive* sequence under RA configuration using HEVC and approximated transform matrices

Base QP		**HEVC**	**T16**	**ST16**
22	Bits	126624	128336	128096
	PSNR	Y: 38.27 dB U: 42.64 dB V: 44.00 dB	Y: 38.25 dB U: 42.65 dB V: 44.02 dB	Y: 38.26 dB U: 42.67 dB V: 44.01 dB
27	Bits	43544	43888	44552
	PSNR	Y: 36.48 dB U: 41.54 dB V: 42.19 dB	Y: 36.41 dB U: 41.50 dB V: 42.21 dB	Y: 36.48 dB U: 41.53 dB V: 42.18 dB
32	Bits	17552	17776	17416
	PSNR	Y: 34.32 dB U: 40.37 dB V: 40.43 dB	Y: 34.34 dB U: 40.36 dB V: 40.42 dB	Y: 34.32 dB U: 40.37 dB V: 40.39 dB
37	Bits	8280	8120	8344
	PSNR	Y: 32.16 dB U: 39.56 dB V: 39.17 dB	Y: 32.24 dB U: 39.45 dB V: 39.17 dB	Y: 32.24 dB U: 39.43 dB V: 39.19 dB

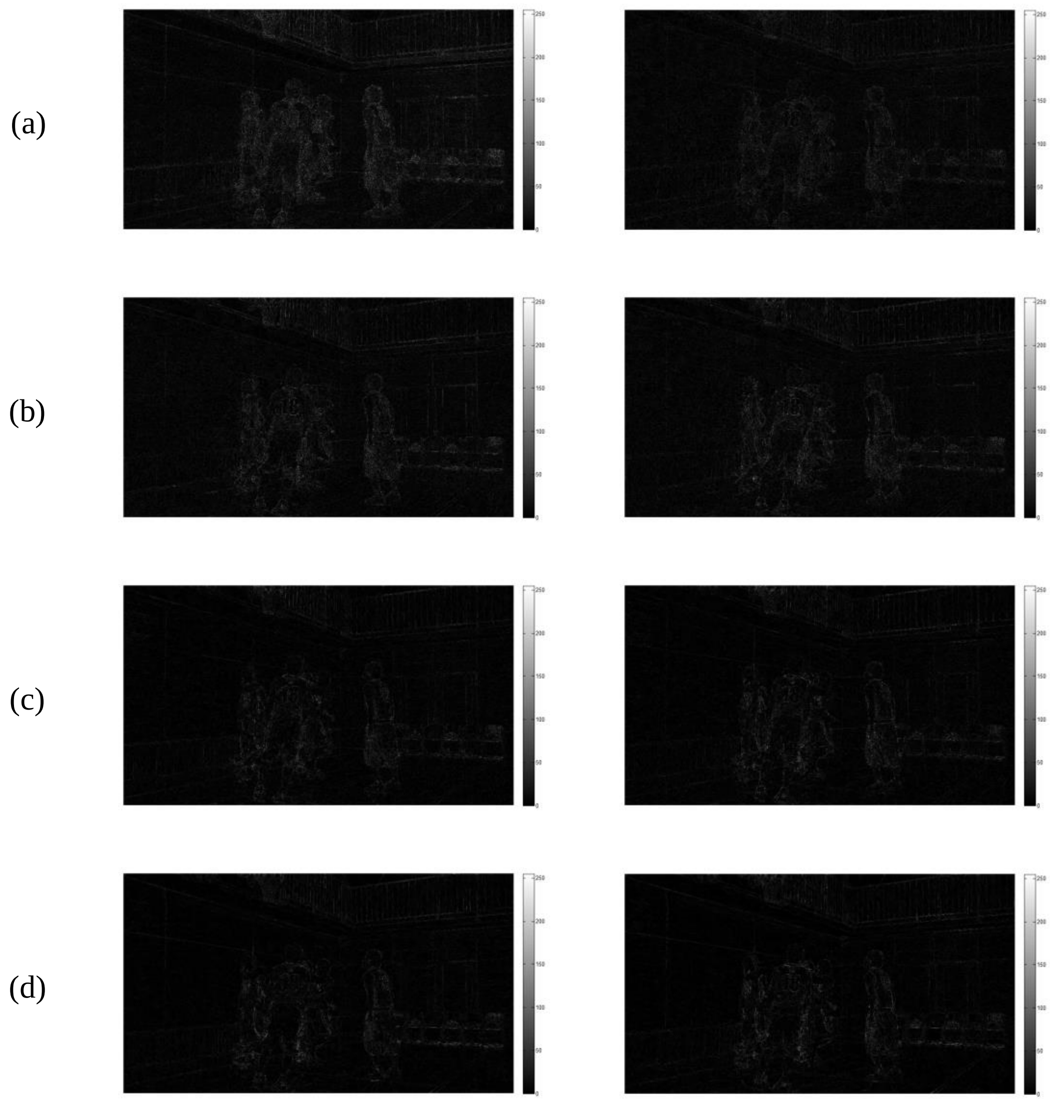


Fig. 5.11 Image differences of the last frame of *B4 – BasketballDrive* sequence (frame 499, RA, Main) using T16 (left column) and ST16 (right column) transform matrices over HEVC with base QP values of (a) 22, (b) 27, (c) 32, and (d) 37

(a)

(b)

(c)

(d)

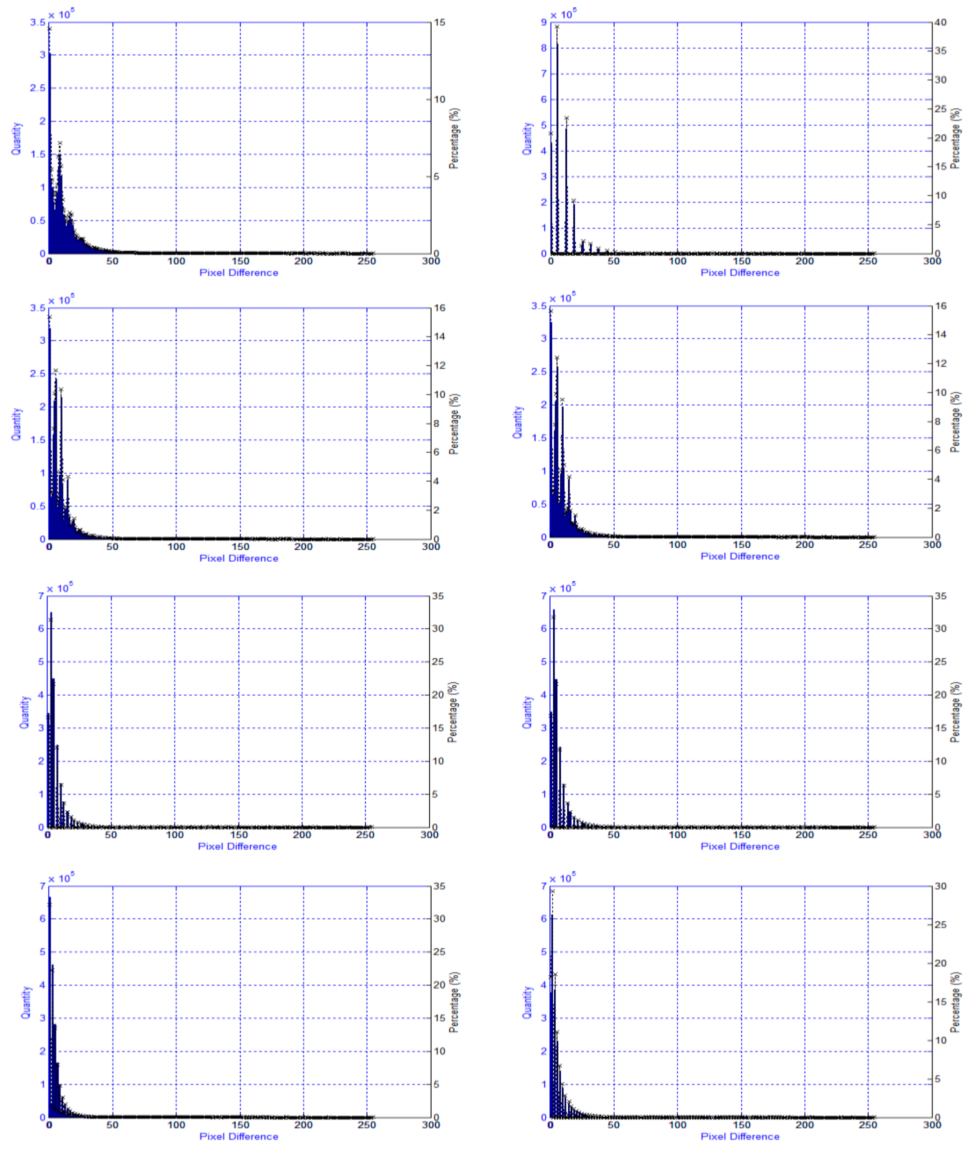


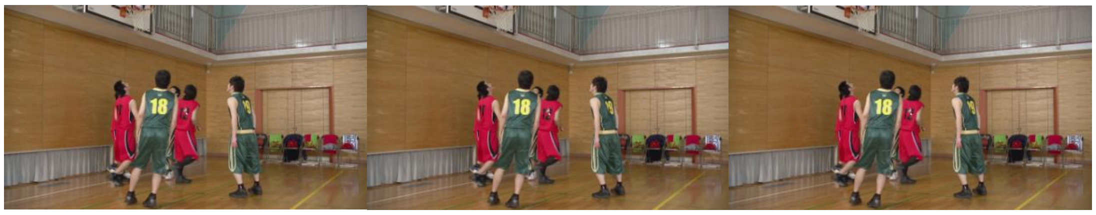
Fig. 5.12 Histograms of pixel differences of the last frame of *B4 – BasketballDrive* sequence (frame 499, RA, Main) using T16 (left column) and ST16 (right column) transform matrices over HEVC with base QP values of (a) 22, (b) 27, (c) 32, and (d) 37

Table 5.9 Percentage (%) of pixel differences of the last frame of *B4 – BasketballDrive* sequence using T16 and ST16 transform matrices over HEVC under RA configuration in Main profile

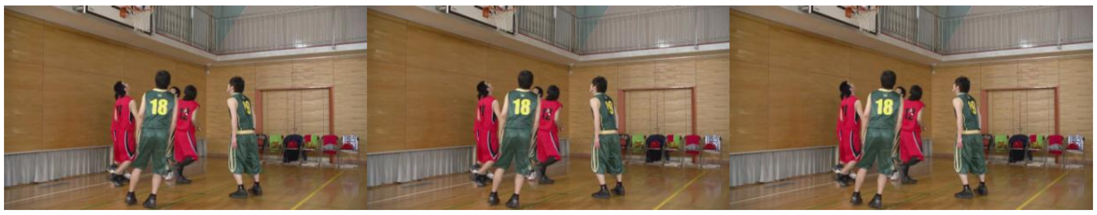
Base QP	**Transform Pixel Difference**					
	T16			**ST16**		
	≤ 25	**26 – 50**	**> 50**	**≤ 25**	**26 – 50**	**> 50**
22	88.63	9.40	1.97	93.98	5.57	0.45
27	94.97	4.24	0.80	95.02	4.21	0.77
32	97.37	2.31	0.32	97.35	2.23	0.42
37	97.74	1.94	0.33	97.58	2.03	0.39

Fig. 5.13 displays the snapshots of the last frame of *B4 – BasketballDrive* sequence under the LB configuration in Main profile with Table 5.10 showing the corresponding bitrate and PSNR values of the frames. The closeness in the obtained objective PSNR levels offer some support to the visual quality of these frames. Fig. 5.14 provides the image differences of the respective frames using T16 and ST16 matrices against HEVC shown in Fig. 5.13. As previously remarked for the RA configuration, most differences appear around object edges and these differences look less visible with increasing QPs. Fig. 5.15 plots the histograms of pixel differences and Table 5.11 shows the corresponding percentages of pixel differences grouped under the same three categories: ≤ 25; 26 – 50; > 50. The majority of these differences are also in the ≤ 25 category; with increasing percentages as the QP is increased for both T16 and ST16 transform matrices. There is a slight ambiguity with ST16 at QP27, where its percentage in the ≤ 25 category (89.57%) is lower than T16 (94.72%) while the percentages are higher at the other three QP values. More importantly against HEVC, both approximated transform matrices yield pixel differences of 25 or lower in at least 82% of cases under the studied QP values (22–37) and the LB configuration.

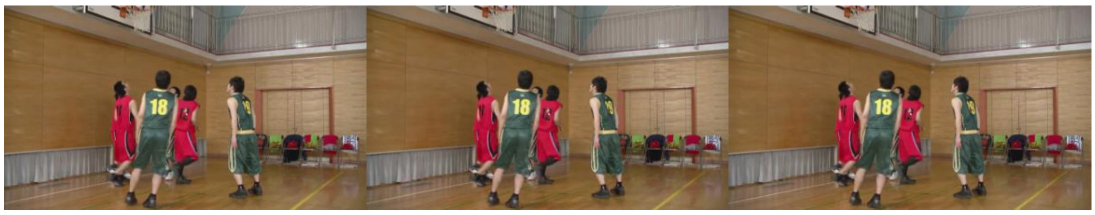
(a)



(b)



(c)



(d)

Fig. 5.13 Snapshots of *B4 – BasketballDrive* sequence (frame 499, LB, Main) using HEVC (left column), T16 (middle column), and ST16 (right column) transform matrices and base QP values of (a) 22, (b) 27, (c) 32, and (d) 37

Table 5.10 Number of bits and PSNR values of last frame of *B4 – BasketballDrive* sequence under LB configuration using HEVC and approximated transform matrices

Base QP		**HEVC**	**T16**	**ST16**
22	Bits	232448	231424	234440
	PSNR	Y: 38.92 dB U: 42.70 dB V: 44.24 dB	Y: 38.92 dB U: 42.67 dB V: 44.19 dB	Y: 38.92 dB U: 42.66 dB V: 44.21 dB
27	Bits	80568	80064	80632
	PSNR	Y: 37.08 dB U: 41.38 dB V: 42.19 dB	Y: 37.06 dB U: 41.37 dB V: 42.20 dB	Y: 37.06 dB U: 41.36 dB V: 42.19 dB
32	Bits	31504	31032	30640
	PSNR	Y: 34.85 dB U: 40.10 dB V: 40.27 dB	Y: 34.78 dB U: 40.08 dB V: 40.19 dB	Y: 34.80 dB U: 40.11 dB V: 40.25 dB
37	Bits	14592	14248	14608
	PSNR	Y: 32.54 dB U: 39.23 dB V: 38.94 dB	Y: 32.51 dB U: 39.24 dB V: 39.02 dB	Y: 32.55 dB U: 39.21 dB V: 39.00 dB

(a)

(b)

(c)

(d)

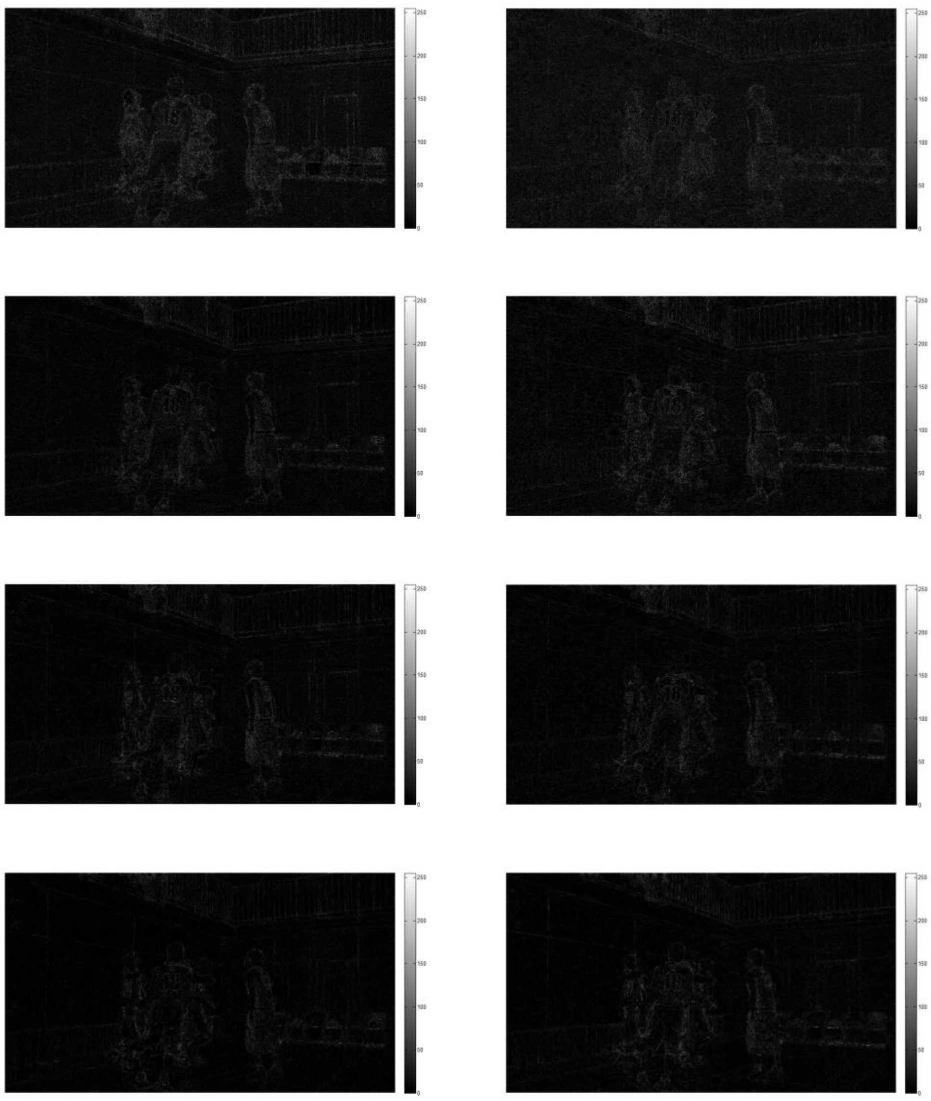


Fig. 5.14 Image differences of the last frame of *B4 – BasketballDrive* sequence (frame 499, LB, Main) using T16 (left column), and ST16 (right column) transform matrices over HEVC with base QP values of (a) 22, (b) 27, (c) 32, and (d) 37

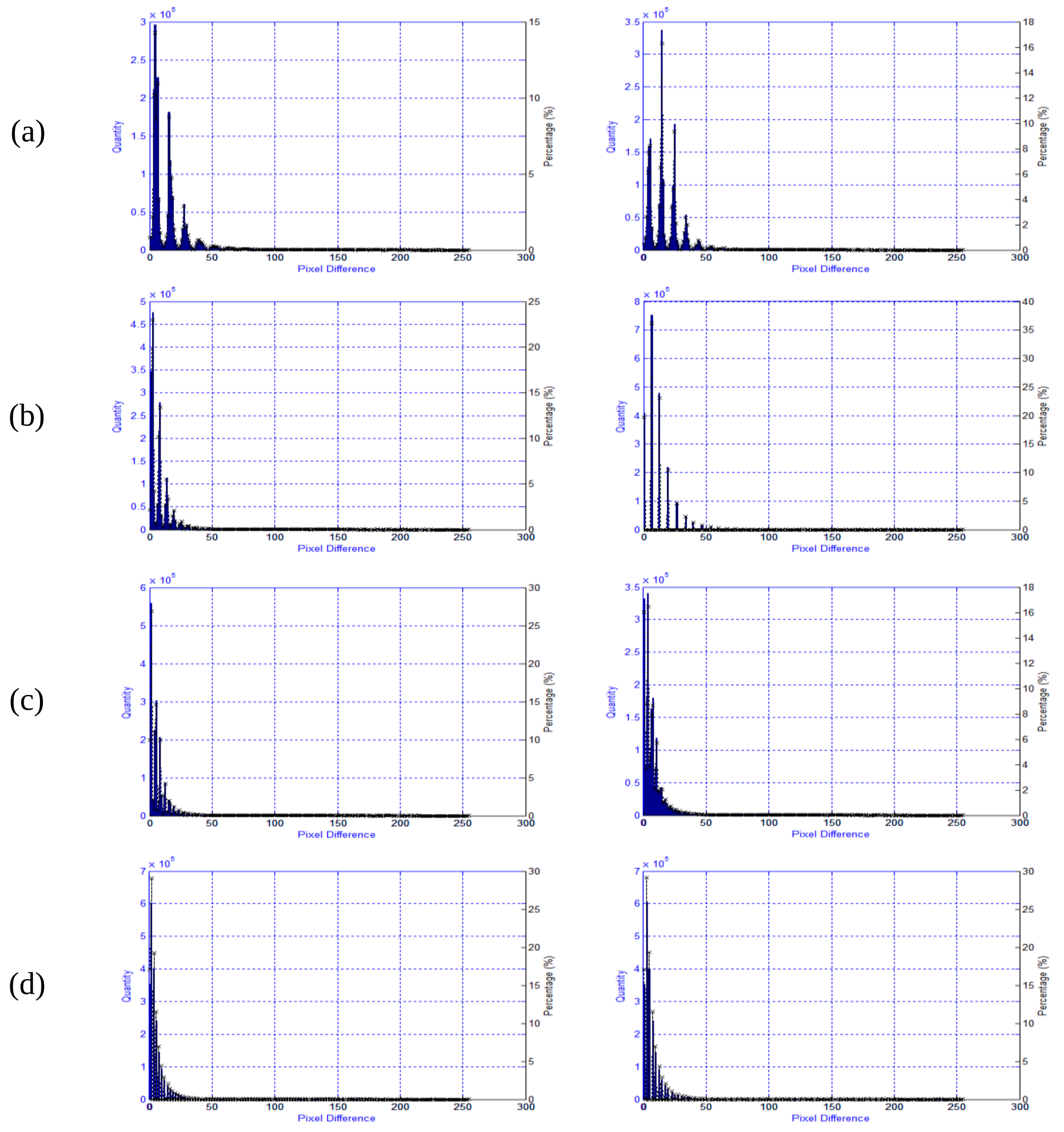


Fig. 5.15 Histograms of pixel differences of the last frame of *B4 – BasketballDrive* sequence (frame 499, LB, Main) using T16 (left column) and ST16 (right column) transform matrices over HEVC with base QP values of (a) 22, (b) 27, (c) 32, and (d) 37

Table 5.11 Percentage (%) of pixel differences of the last frame of *B4 – BasketballDrive* sequence using T16 and ST16 transform matrices over HEVC under LB configuration in Main profile

Base QP	**Transform Pixel Difference**					
	T16			**ST16**		
	≤ 25	**26 – 50**	**> 50**	**≤ 25**	**26 – 50**	**> 50**
22	82.23	14.72	3.05	82.86	14.62	2.53
27	94.72	4.51	0.76	89.57	8.93	1.51
32	95.94	3.41	0.65	96.01	3.41	0.58
37	97.89	1.81	0.30	96.67	2.82	0.51

5.2.5 Conclusions

Both approximated transforms, T16 and ST16, provide a similar average BD-rate coding performance of +1.7% in RA and +0.7% in LB configurations, respectively, for videos of HD-quality or beyond. From reconstructed video frame samples, identical visual and objective quality levels were obtained against HEVC, with at least 88% and 82% of pixel differences not exceeding 25 under RA and LB configurations, respectively. The increments in bitrate in order to achieve a similar objective quality may not be regarded as small, but the potential hardware complexity savings as shown earlier in Table 4.8 could outweigh these penalties in bitrate.

5.3 Approximated Quantisation

This sub-section evaluates the coding performance of the approximated quantisation multiplier set, Q, against the original HEVC quantisation.

5.3.1 Experimental Settings

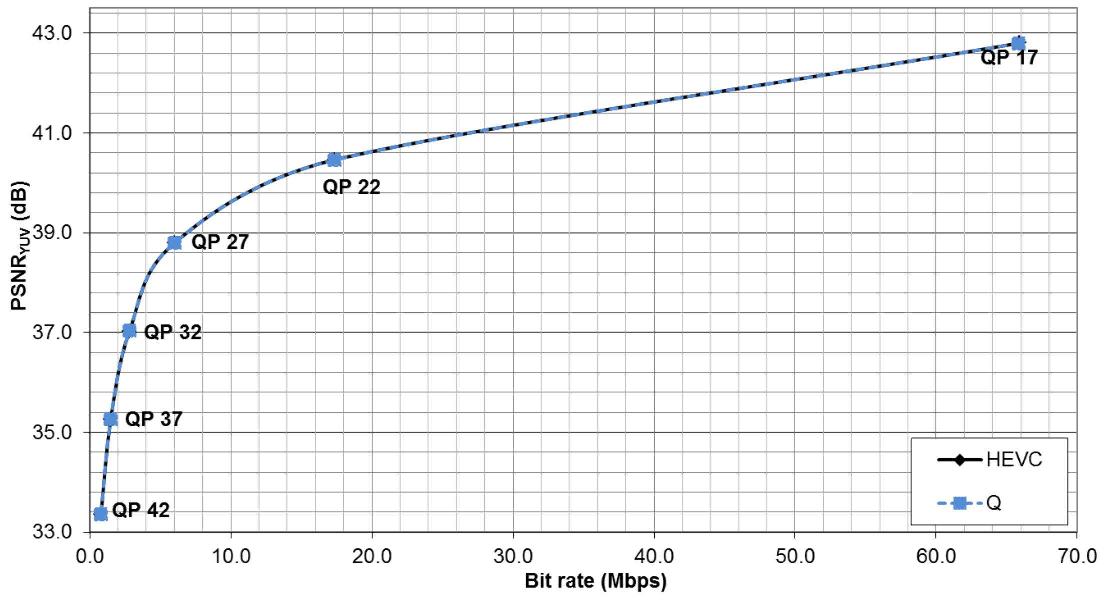
The following experiment on the approximated quantisation multiplier set, Q, follows the experimental settings applied for the approximated transform matrices, T16 and ST16. Therefore, only Classes A (cropped UHD) and B (Full-HD 1080p) test video sequences were evaluated under RA coding structure, and Classes B and E (HD-ready 720p) videos under LB. It is assumed that low-resolution videos would exhibit better coding performance results. On top of that, as there are six multipliers in quantisation set, the number of base QP values was extended from four to six to include 17 and 42. Table 5.12 summarises the experimental settings used in this section.

Table 5.12 Experimental settings on approximated quantisation

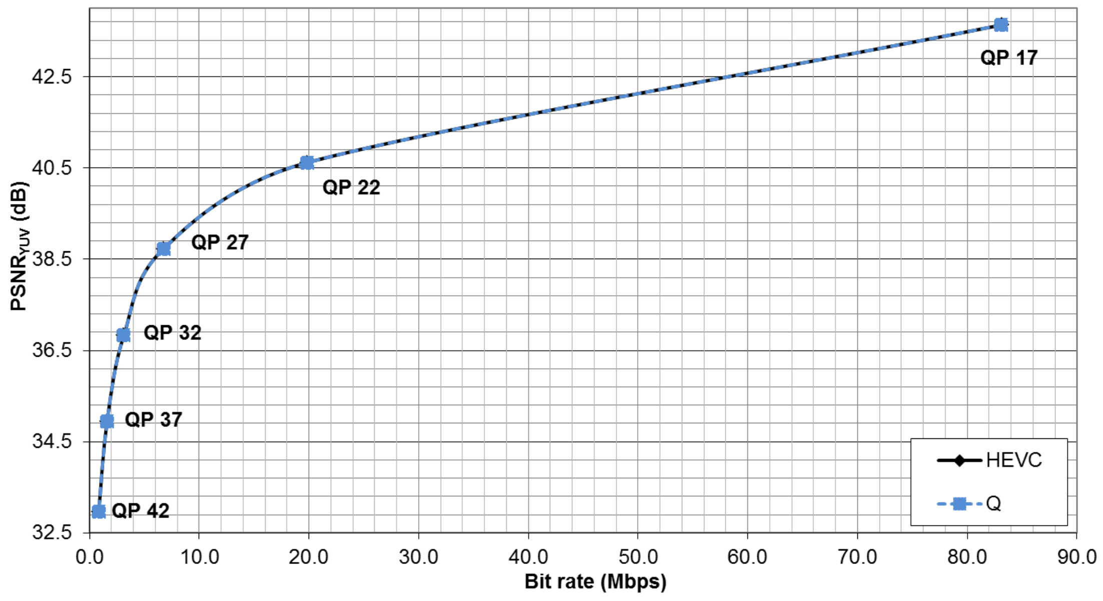
HEVC Reference Software Version	13.0 (HM–13.0)
Profiles	Main
Encoding Structures	Random Access (RA) and Low Delay with B-frames (LB)
Test Video Sequences	Classes A and B for RA Classes B and E for LB
Intra-period for RA	24, 32, 48, or 64
Group of Pictures (GOP)	8
Hierarchical Bidirectional Coding	Enabled
Base Quantisation Parameters (QP)	17, 22, 27, 32, 37, 42
QP offset between temporal level	+1

5.3.2 Peak Signal to Noise Ratio (PSNR)

Fig. 5.16 presents the $PSNR_{YUV}$-based R-D curves under RA and LB configurations for *B4 – BasketballDrive* sequence. Again, a difference between two corresponding R-D curves using the original HEVC and approximated quantisation is hardly noticeable for this sequence in both entertainment and interactive scenarios. The corresponding R-D curves for the other tested video sequences are provided in Appendix B. In most of these sequences, both R-D curves are almost aligned with each other, indicating the closeness of the quality-bitrate performance achieved by the approximated quantisation multiplier set, Q, in videos of HD-quality or beyond.



(a)



(b)

Fig. 5.16 R-D$_{PSNR}$ curves of *B4* – *BasketballDrive* sequence using original HEVC and approximated quantisation multiplier set, Q, under (a) RA and (b) LB configurations and in Main profile

5.3.3 Bjøntegaard-Delta Bitrate (BD-rate)

A summary of average BD-rate levels (YUV-based) for each class under study is presented in Table 5.13 for RA and LB encoding structures in Main profile. In RA, The average BD-rate value for classes A and B is both 0.0%, suggesting there is a negligible effect of the approximated quantisation multiplier set, Q, over HEVC. In LB, the same average value of 0.0% was obtained for class B videos, and slight bitrate savings were achievable for class E videos with an average of -0.1%. The most savings was seen from *E2 – Johnny,* where the background is most stagnant among the test conference video sequences, with a -0.3% average BD-rate value.

Table 5.13 Average BD-rate values (%) for equal PSNR_{YUV} between HEVC and approximated quantisation multipliers in Main profile

Class	Sequence	Random Access (RA)	Low Delay (LB)
A (2560 × 1600) Cropped UHD	*A1*	0.0	-
	A2	0.0	-
	A3	0.0	-
	A4	0.1	-
	Average	0.0	-
B (1920 × 1080) Full HD	*B1*	0.0	0.0
	B2	0.0	0.0
	B3	0.0	0.0
	B4	0.0	0.0
	B5	0.0	0.0
	Average	0.0	0.0
E (1280 × 720) HD-ready	*E1*	-	-0.1
	E2	-	-0.3
	E3	-	0.0
	Average	-	-0.1
Overall[a]		0.0	-0.1

[a] Overall average of Classes A and B for RA and Classes B and E for LB

5.3.4 Visual Observations

Fig. 5.17 shows the last frame of *B4 – BasketballDrive* sequence under the RA configuration. In each base QP value, obvious video degradations or improvements are not apparent using the approximated quantisation multipliers, Q, over HEVC. This observation is reflected in the closeness of the PSNR levels of individual Y-, U-, and V-components of the frame as shown in Table 5.14. Fig. 5.18 displays the image differences and the histograms of pixel differences between the two sets of quantisation multipliers. Pixel differences appear to be less noticeable as QP increases from 17 to 42.

Fig. 5.19 shows the corresponding snapshots of the last frame of *B4 – BasketballDrive* sequence under the LB configuration. Table 5.15 equips these snapshots with the respective PSNR values. As seen in the RA configuration, the visual and objective qualities of the reconstructed frames using Q and HEVC at most QP settings seem indistinguishable. Fig. 5.20 provides the image differences and histograms of pixel differences, where pixel differences are the most apparent at QP equals 17 and become less visible as QP increases.

Table 5.16 groups the percentages of pixel differences in both RA and LB configurations into three categories: ≤ 25; 26 – 50; > 50. It can be seen that periodical insertions of I-frames as applied in RA achieves more pixel differences having values of 25 or lower as opposed to LB, especially at QPs 17 and 22. At these two base QPs, due to the hierarchical bidirectional settings, the QP values for the last frame of *B4 – BasketballDrive* sequence are respectively increased to 21 and 26 in RA and 20 and 25 in LB. Recalling the approximated quantisation multiplier set from (4.13), Q = $[26112, 23296, 20560, 18396, 16384, 14592]^T$ where Q(2), Q(3), and Q(4) are maintained as in the original HEVC quantisation multipliers. At QPs 20 and 21 (i.e., base QP = 17), the involved quantisation multipliers are Q(20%6) = 20560 and Q(21%6) = 18396, respectively, i.e., as in the original HEVC set. Similarly, Q(26%6) = 20560. Only QP 25 involves one of the three approximated quantisation multipliers, i.e., Q(25%6) = 23296. At this QP under LB configuration, the total percentage of pixel differences of 50 or lower is 97.65% (Table 5.16), which is a great improvement relative to QP 20, where the total percentage of the first two categories is 80.95%.

As previously described in Chapter 4, the first base QP of 17, as well as the last QP of 42, were added to the normal QP range recommended by the CTC (Bossen, 2013) to ensure that all six quantisation multipliers were covered in this experiment. Low range of QP values is useful in producing videos of cinema quality. Thus, although Q may appear to be less suitable to be coupled with QPs below 22 especially in low delay applications, such a high quality encoded video signal is typically unnecessary in video communications, for instance in live broadcasts and videoconferencing meetings which require a transmission with small delays.

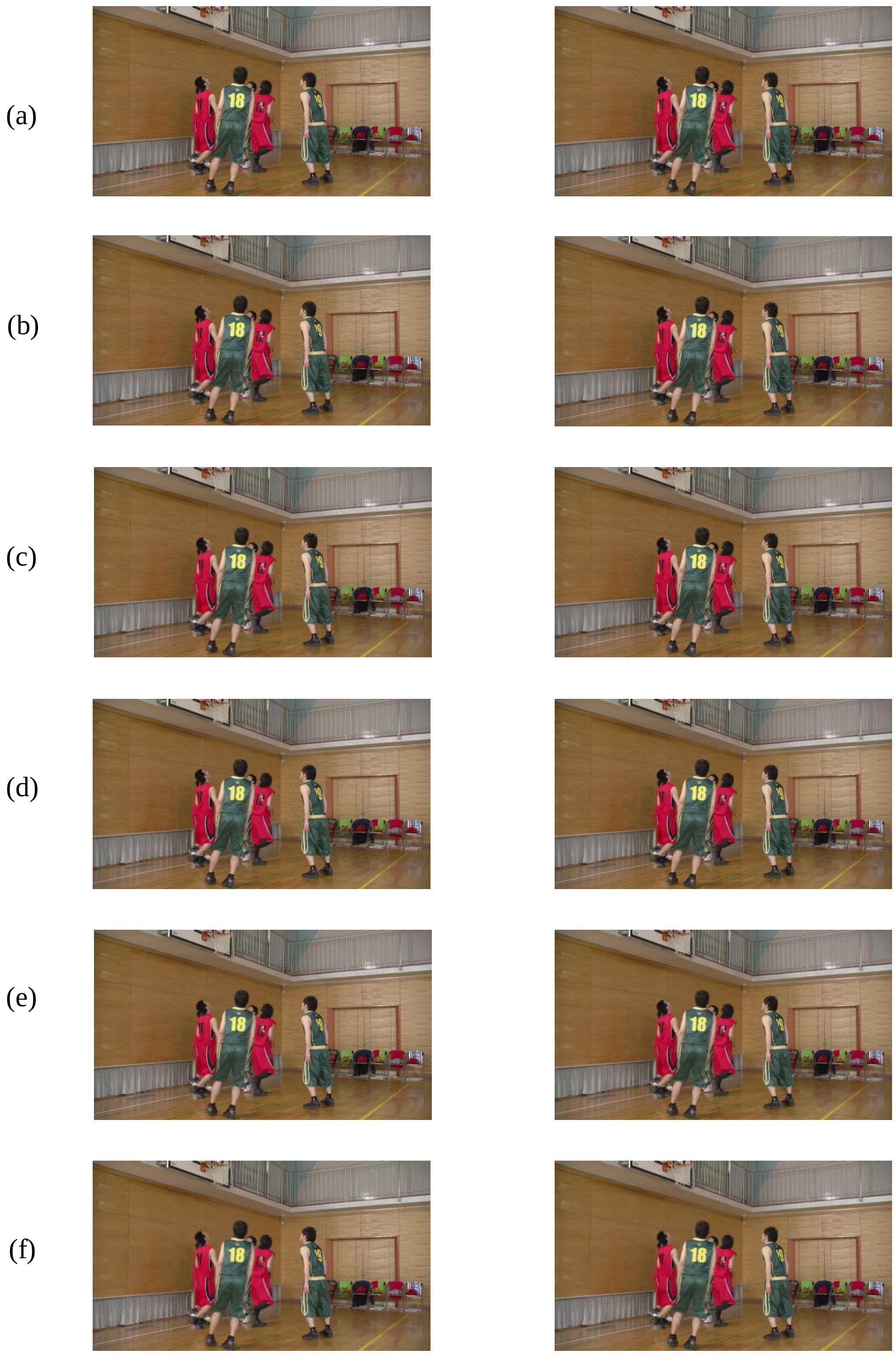


Fig. 5.17 Snapshots of the last frame of *B4 – BasketballDrive* sequence (frame 499, RA, Main) using HEVC (left column) and approximated quantisation multipliers, Q (right column) and QP values of (a) 17, (b) 22, (c) 27, (d) 32, (e) 37, and (f) 42

Table 5.14 Number of bits and PSNR values of last frame of *B4 – BasketballDrive* sequence under RA configuration using HEVC and approximated, Q, quantisation multipliers

Base QP		**HEVC**	**Q**
17	Bits	510176	506048
	PSNR	Y: 39.51 dB U: 43.47 dB V: 45.63 dB	Y: 39.52 dB U: 43.48 dB V: 45.64 dB
22	Bits	126624	126760
	PSNR	Y: 38.27 dB U: 42.64 dB V: 44.00 dB	Y: 38.28 dB U: 42.68 dB V: 44.07 dB
27	Bits	43544	44832
	PSNR	Y: 36.48 dB U: 41.54 dB V: 42.19 dB	Y: 36.51 dB U: 41.56 dB V: 42.20 dB
32	Bits	17552	17784
	PSNR	Y: 34.32 dB U: 40.37 dB V: 40.43 dB	Y: 34.35 dB U: 40.41 dB V: 40.47 dB
37	Bits	8280	8264
	PSNR	Y: 32.16 dB U: 39.56 dB V: 39.17 dB	Y: 32.22 dB U: 39.53 dB V: 39.08 dB
42	Bits	3680	3896
	PSNR	Y: 30.11 dB U: 38.67 dB V: 37.77 dB	Y: 30.15 dB U: 38.61 dB V: 37.78 dB

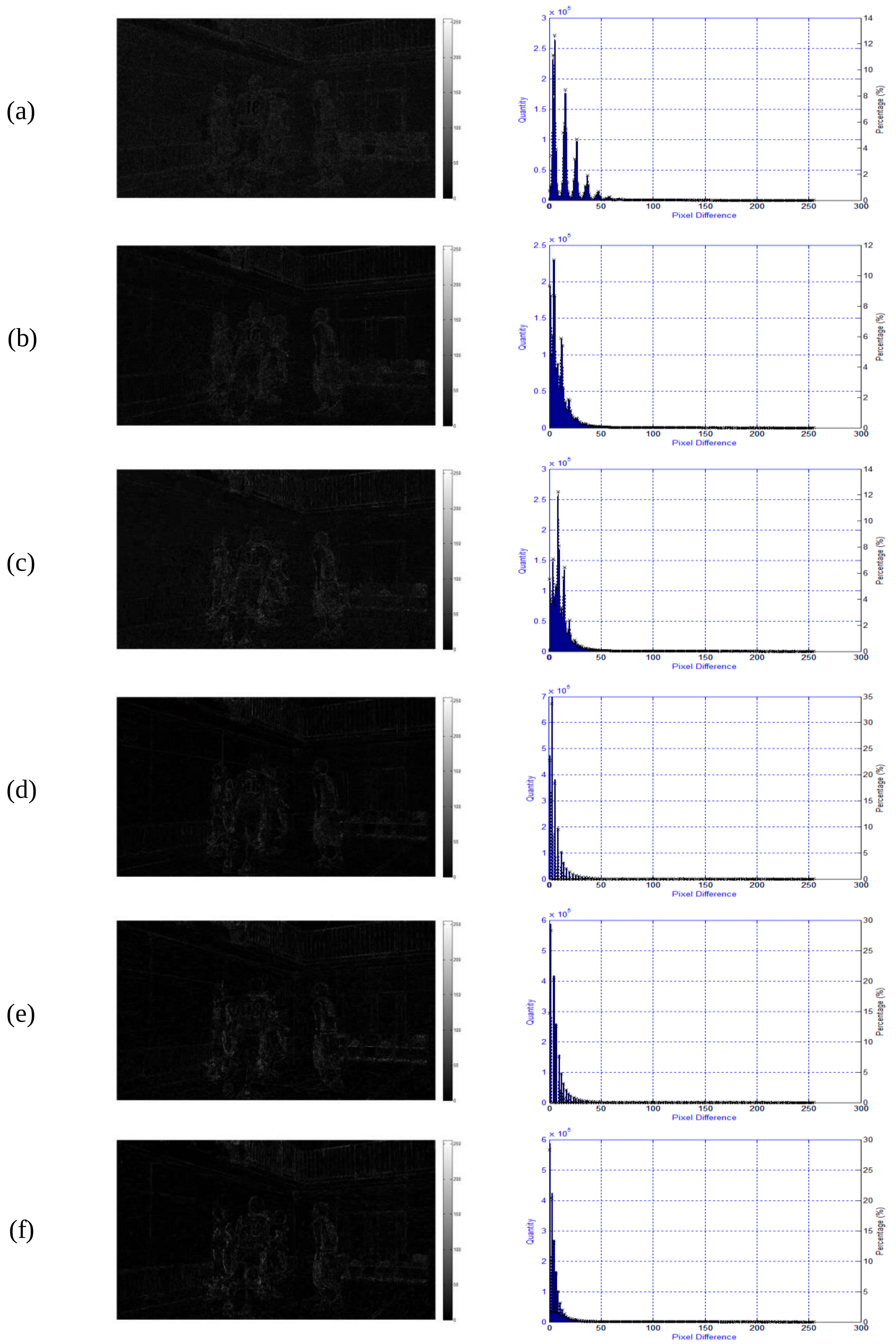


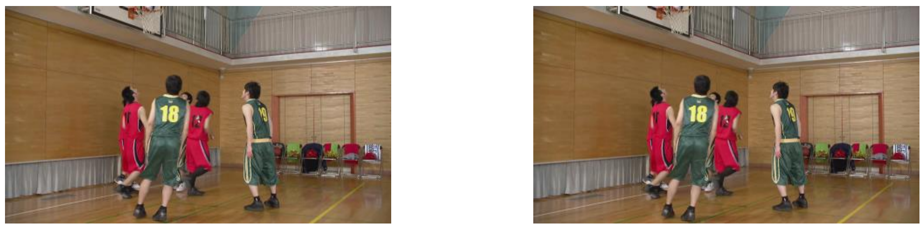
Fig. 5.18 Image differences and histograms of pixel differences of the last frame of *B4 – BasketballDrive* sequence (frame 499, RA, Main) using approximated quantisation multipliers, Q over HEVC at QP values of (a) 17, (b) 22, (c) 27, (d) 32, (e) 37, and (f) 42

(a)

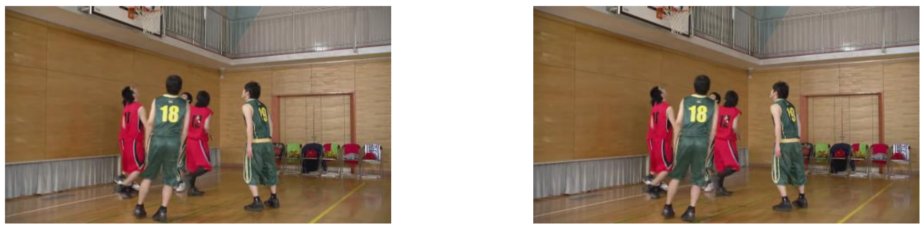
(b)

(c)

(d)



(e)



(f)

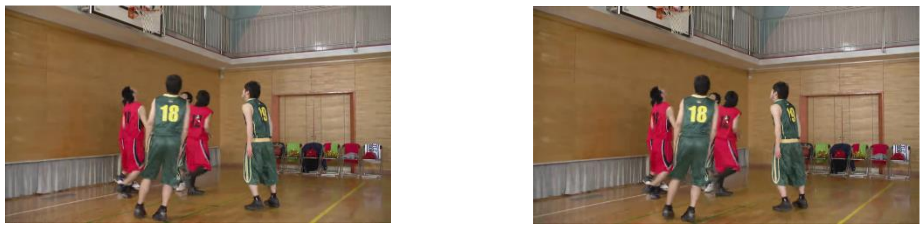
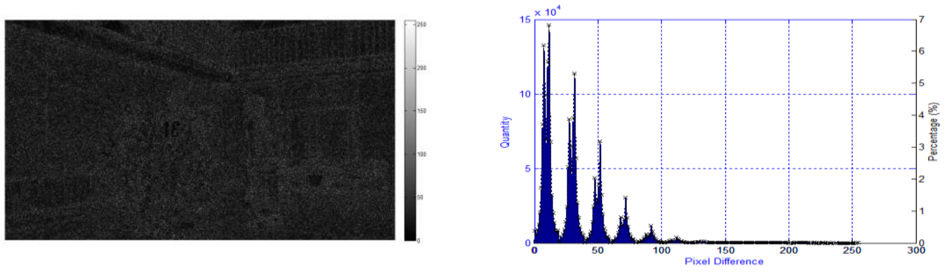


Fig. 5.19 Snapshots of the last frame of *B4 – BasketballDrive* sequence (frame 499, LB, Main) using HEVC (left column) and approximated quantisation multipliers, Q (right column) and QP values of (a) 17, (b) 22, (c) 27, (d) 32, (e) 37, and (f) 42

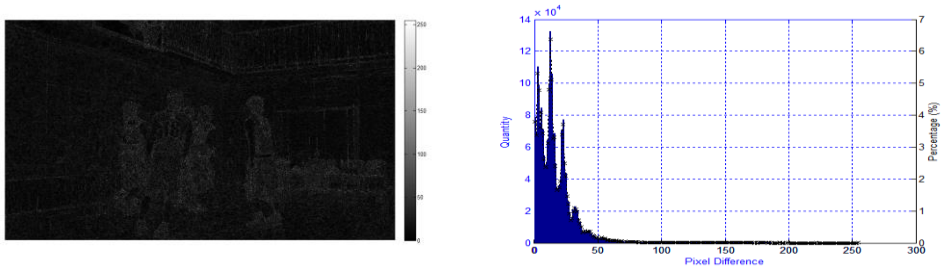
Table 5.15 Number of bits and PSNR values of last frame of *B4 – BasketballDrive* sequence under LB configuration using HEVC and approximated, Q, quantisation multipliers

Base QP		**HEVC**	**Q**
17	Bits	1289808	1288472
	PSNR	Y: 41.77 dB U: 43.75 dB V: 45.94 dB	Y: 41.78 dB U: 43.74 dB V: 45.93 dB
22	Bits	232448	231848
	PSNR	Y: 38.92 dB U: 42.70 dB V: 44.24 dB	Y: 38.91 dB U: 42.67 dB V: 44.18 dB
27	Bits	80568	80416
	PSNR	Y: 37.08 dB U: 41.38 dB V: 42.19 dB	Y: 37.09 dB U: 41.38 dB V: 42.19 dB
32	Bits	31504	31536
	PSNR	Y: 34.85 dB U: 40.10 dB V: 40.27 dB	Y: 34.82 dB U: 40.06 dB V: 40.27 dB
37	Bits	14592	13792
	PSNR	Y: 32.54 dB U: 39.23 dB V: 38.94 dB	Y: 32.55 dB U: 39.22 dB V: 38.94 dB
42	Bits	6688	7000
	PSNR	Y: 30.28 dB U: 38.55 dB V: 37.81 dB	Y: 30.27 dB U: 38.49 dB V: 37.84 dB

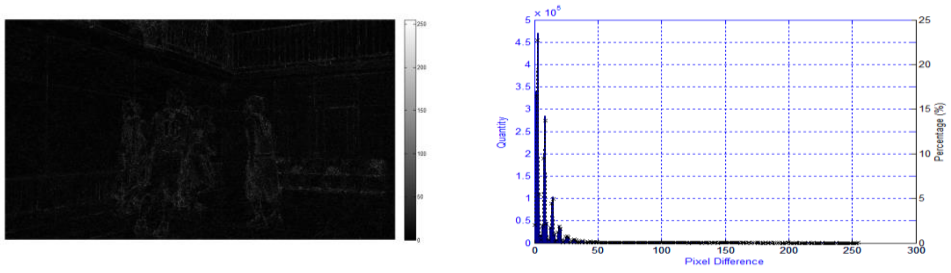
(a)



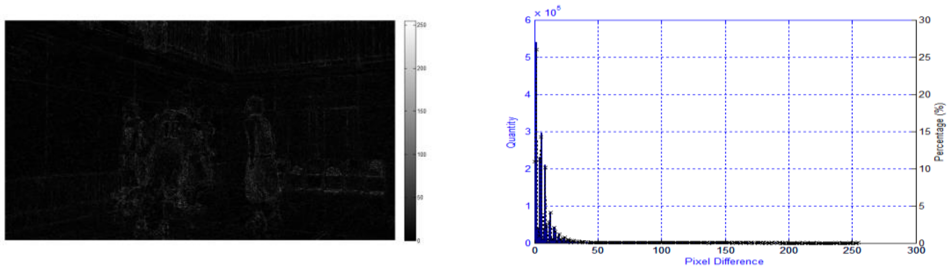
(b)



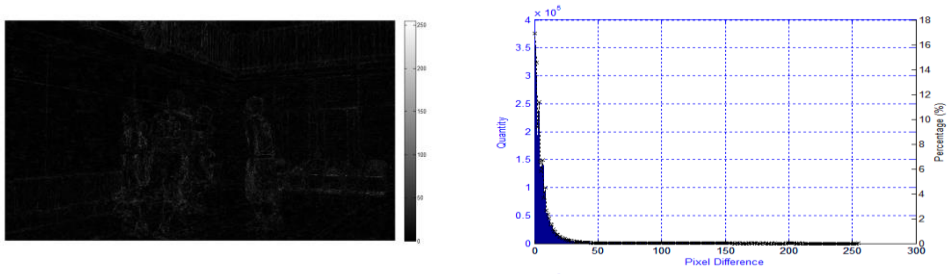
(c)



(d)



(e)



(f)

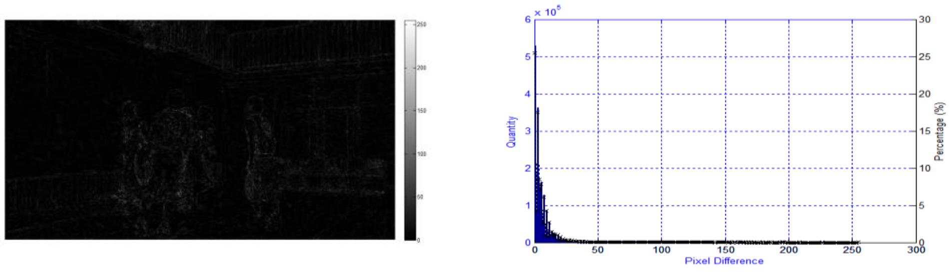


Fig. 5.20 Image differences and histograms of pixel differences of the last frame of *B4 – BasketballDrive* sequence (frame 499, LB, Main) using approximated quantisation multipliers, Q over HEVC at QP values of (a) 17, (b) 22, (c) 27, (d) 32, (e) 37, and (f) 42

Table 5.16 Percentage (%) of pixel differences of the last frame of *B4 – BasketballDrive* sequence using approximated quantisation multipliers, Q, in Main profile under RA and LB configurations

Base QP	**Configuration Pixel Differences**					
	RA			**LB**		
	≤ 25	**26 – 50**	**> 50**	**≤ 25**	**26 – 50**	**> 50**
17	78.46	19.48	2.06	43.73	37.22	19.05
22	93.47	5.73	0.80	82.80	14.85	2.35
27	93.07	5.86	1.07	94.41	4.81	0.78
32	96.85	2.72	0.43	95.87	3.47	0.66
37	96.40	2.99	0.61	96.67	2.88	0.44
42	96.40	3.03	0.57	96.93	2.67	0.40

5.3.5 Conclusions

The approximated quantisation multiplier set, Q, does not provide a significant difference in terms of coding performance on encoded HD-quality videos or larger, over the original quantisation multipliers in HEVC. Average BD-rate values of 0.0% and -0.1% obtained in RA and LB configurations, respectively, suggest that Q could be employed in encoders producing HEVC-compliant bitstreams, providing some complexity reductions in this processing block without requiring any changes in the decoders. From reconstructed video frame samples, at least 93% and 82% of pixel differences do not exceed the value of 25 in RA and LB, respectively, in base QP values of 22 and above, further suggesting that the approximated quantisation multipliers are practically applicable in entertainment and interactive applications at mid to high range of QP settings. Although lower performances obtained at the base QP of 17 in both configurations; with only around 78% and 43% of pixel differences having the values of 25 or lower in RA and LB, respectively, make Q to be less favourable for low values of QP, i.e., very fine quantisation levels, such a high-quality encoded video signal is unnecessary in typical use cases especially involving video transmissions.

5.4 Approximated Transform and Quantisation

After evaluating the coding performances of approximated transform matrix sets, T16 and ST16, as well as approximated quantisation multiplier set, Q, over the original HEVC encoding, this section assesses the combined performance of an approximated transform matrix with the approximated quantisation multipliers. As ST16 offers more arithmetic savings than T16 over HEVC transform matrices, as well as it is the eventual transform matrix intended to be applied in the HEVC encoder (and the corresponding modified inverse transform in the decoder), ST16 has therefore been selected to be combined with Q.

5.4.1 Experimental Settings

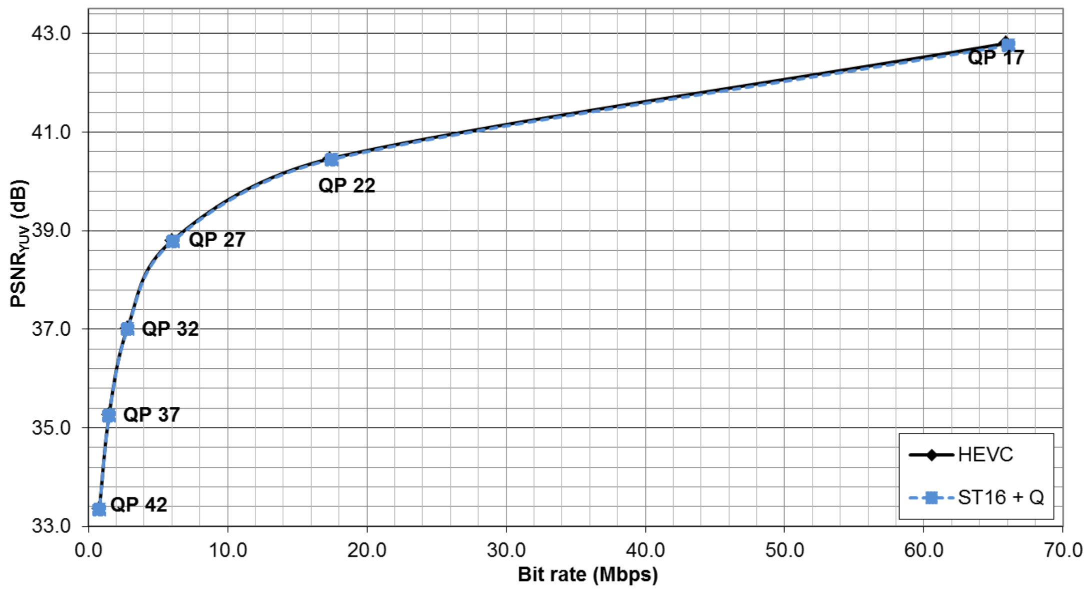
This experiment follows exactly the same settings as applied in the previous section on the approximated quantisation multiplier set, Q. Table 5.17 summarises the experimental settings used in this section (same as Table 5.12).

Table 5.17 Experimental settings on approximated transform and quantisation

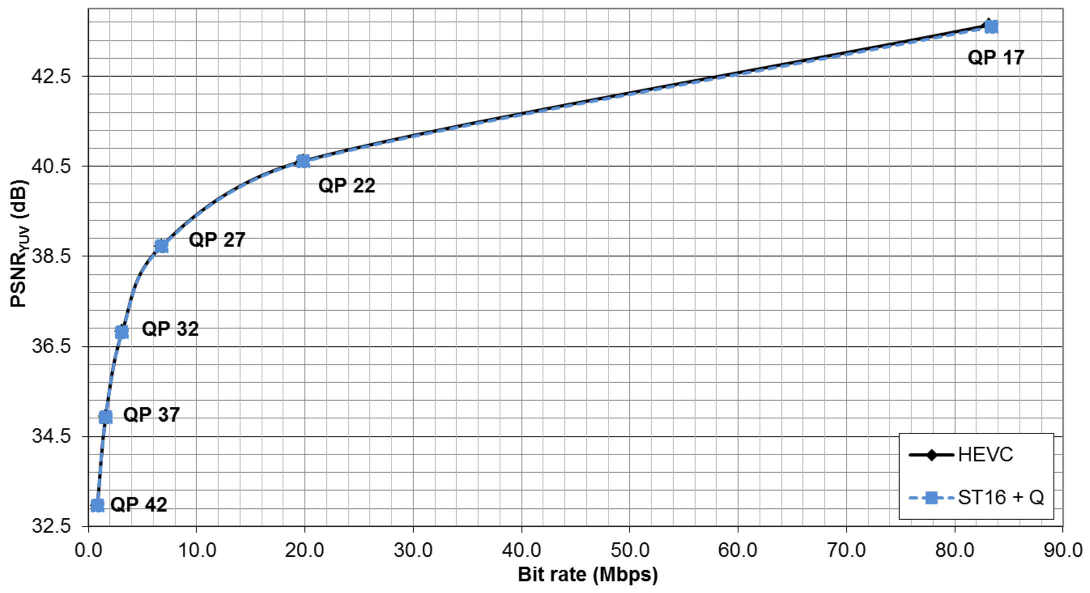
HEVC Reference Software Version	13.0 (HM–13.0)
Profiles	Main
Encoding Structures	Random Access (RA) and Low Delay with B-frames (LB)
Test Video Sequences	Classes A and B for RA Classes B and E for LB
Intra-period for RA	24, 32, 48, or 64
Group of Pictures (GOP)	8
Hierarchical Bidirectional Coding	Enabled
Base Quantisation Parameters (QP)	17, 22, 27, 32, 37, 42
QP offset between temporal level	+1

5.4.2 Peak Signal to Noise Ratio (PSNR)

Fig. 5.21 presents the $PSNR_{YUV}$-based R-D curves under RA and LB configurations for *B4 – BasketballDrive* sequence. Again, a difference between two corresponding R-D curves using the original HEVC and the combination of approximated transform and quantisation, ST16 + Q, is hardly noticeable for this sequence in both entertainment and interactive scenarios, as both R-D curves almost overlap each other. The corresponding R-D curves for the other tested video sequences are provided in Appendix C. In most of these sequences, both R-D curves almost overlap each other, indicating the closeness of the quality-bitrate performance achieved by the combination of the approximated transform matrices, ST16, and approximated quantisation multipliers, Q.



(a)



(b)

Fig. 5.21 R-D$_{\text{PSNR}}$ curves of *B4 – BasketballDrive* sequence using original HEVC and combination of approximated transform matrix and quantisation multiplier sets, ST16 + Q, under (a) RA and (b) LB configurations and in Main profile

5.4.3 Bjøntegaard-Delta Bitrate (BD-rate)

A summary of average BD-rate levels for each class is presented in Table 5.18 for RA and LB encoding structures in Main profile. Comparing with Table 5.7 earlier, there are no or little improvements of around 0.1% or 0.2% in most sequences such as *A1 – Traffic*, *B3 – Cactus*, and *B5 – BQTerrace* as opposed to using ST16 approximated transform matrices only. The biggest improvements could be seen from *A4 – SteamLocomotive* and *B1 – Kimono1* sequences, with a 0.5% BD-rate drop each using ST16 + Q combination. However, *A3 – NebutaFestival* sequence remains a problematic one with a 4.0% BD-rate in comparison to 2.3% when using either ST16 or T16 alone, thus raising the average of Class A to 2.1% rather than 1.9% seen earlier in Table 5.7. Besides these remarks, the other BD-rate values of ST16 + Q and ST16 appear to be similar in both RA and LB configurations.

Table 5.18 Average BD-rate values (%) for equal $PSNR_{YUV}$ between HEVC and combination of approximated transform and quantisation multipliers, ST16 + Q, in Main profile

Class	Sequence	Random Access (RA)	Low Delay (LB)
A (2560 × 1600) Cropped UHD	*A1*	1.3	-
	A2	1.3	-
	A3	4.0	-
	A4	1.9	-
	Average	2.1	-
B (1920 × 1080) Full HD	*B1*	2.2	1.2
	B2	1.3	0.6
	B3	1.3	0.7
	B4	1.1	0.7
	B5	1.0	0.7
	Average	1.4	0.8
E (1280 × 720) HD-ready	*E1*	-	0.6
	E2	-	0.6
	E3	-	0.6
	Average	-	0.6

Overall[a]	1.7	0.7

[a] Overall average of Classes A and B for RA and Classes B and E for LB

5.4.4 Visual Observations

Fig. 5.22 displays the last reconstructed frame of *B4 – BasketballDrive* sequence under the RA configuration using the combination of ST16 and Q. The same remarks can be made on the perceptual quality of these frames as in the approximated transforms, T16 and ST16, as well as the approximated quantisation multipliers, Q, i.e., the corresponding frames at each base QP settings appear visually identical. A difference in objective quality levels, as shown in Table 5.19, is also hardly noticeable in all six QP values under investigation. Fig. 5.23 provides the image differences and histograms of pixel differences between ST16 + Q combination and HEVC in the last reconstructed frames. As seen in the Q experiment, the worst differences were obtained at QP 17, where pixel differences were also present in large homogeneous areas in addition to object edges. As QP increases to 42, pixel differences generally improves.

Fig. 5.24 shows the corresponding snapshots of the last frame of *B4 – BasketballDrive* sequence under the LB configuration, with the respective objective PSNR values provided in Table 5.20. Again, the visual and objective quality levels between the two sets of transform and quantisation multipliers, HEVC and ST16 + Q, appear identical in most QP settings. Fig. 5.25 provides the image differences and histograms of pixel differences in the last frame of *B4 – BasketballDrive* sequence using ST16 + Q against HEVC. As seen in the Q case, the pixel differences are the most apparent at the base QP of 17 and generally improve as QP grows.

Finally, Table 5.21 groups the percentages of pixel differences in RA and LB configurations into three categories: ≤ 25; 26 – 50; > 50. Similar to the observation in the Q case, RA generally achieves better percentages of pixel differences having values of 25 or smaller compared to LB, in particular at QP values of 17 and 22. At QP 17, the introduction of ST16 to Q seems to further decrease the achieved percentages of the first category of pixel differences, with only around 53% in RA and 37% in LB as opposed to 78% and 43% respectively obtained earlier using only Q. Nevertheless, low range of QP values such as below 22 produce cinematic production-quality videos, and is a surplus to most general applications such as home cinema even at UHD-quality. Low values of QP are also unlikely to be used in transmissions of video signals, where the speed and bandwidth are of paramount importance. QP 17 was included in this experiment for additional coverage, as well as to have six QP settings to ensure the inclusiveness of all six quantisation multipliers. Still, at this QP, the total percentages of pixel differences of 50 or lower from the video frame samples are 89.96% in RA and 73.34% in LB, which are not too low. In summary, the combination of ST16 + Q is promising for QP values of 22 or greater in both RA and LB configurations, i.e., for general entertainment and interactive applications, but may be less favourable for lower QPs, i.e., for high-quality, cinema-like video productions and transmissions.

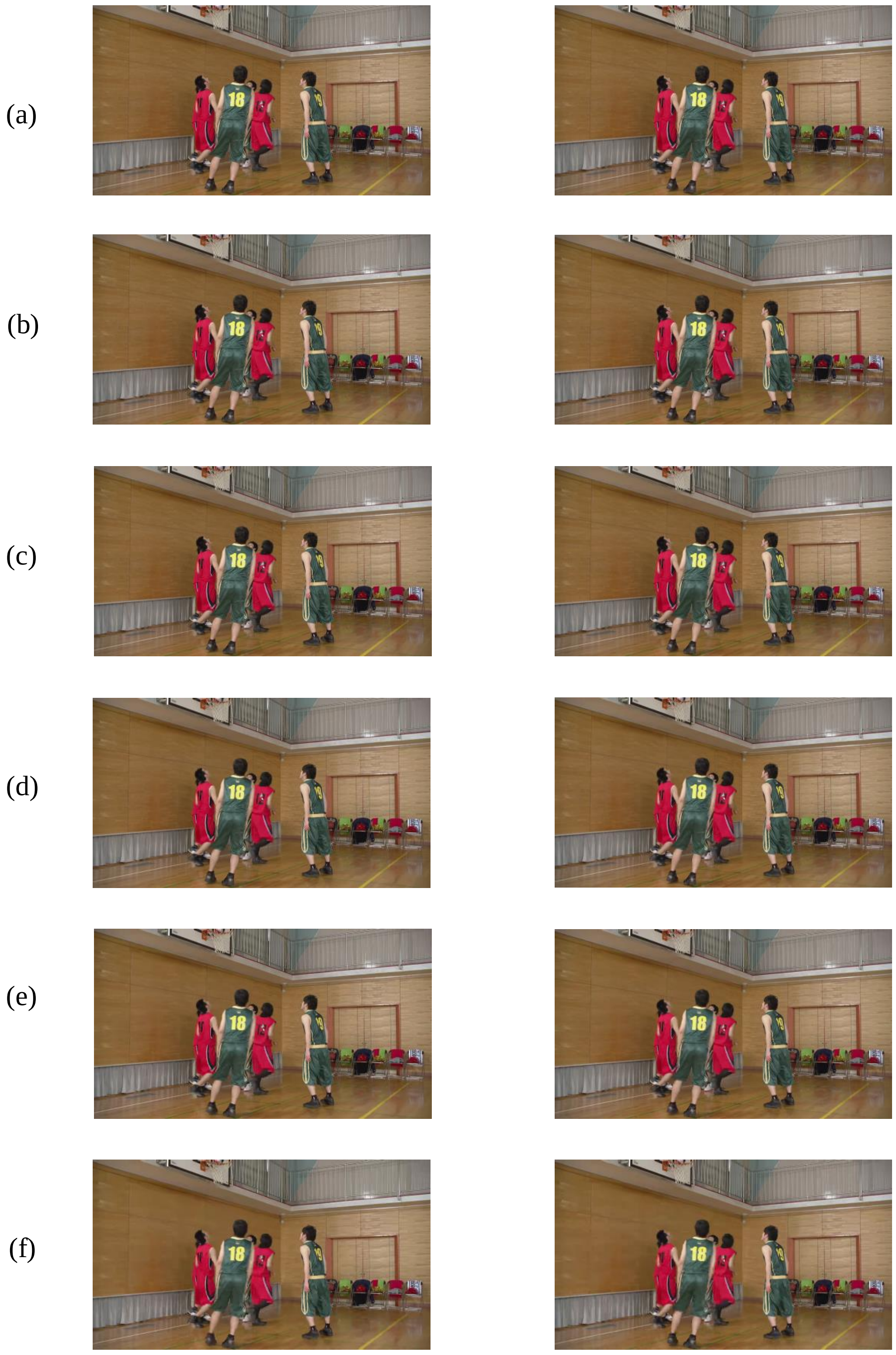


Fig. 5.22 Snapshot of the last frame of *B4 – BasketballDrive* sequence (frame 499, RA, Main) using HEVC (left column) and a combination of approximated transform and quantisation multipliers, ST16 + Q (right column), and QP values of (a) 17, (b) 22, (c) 27, (d) 32, (e) 37, and (f) 42

Table 5.19 Number of bits and PSNR values of last frame of *B4 – BasketballDrive* sequence under RA configuration using HEVC and approximated transform and quantisation multipliers, ST16 + Q in Main profile

Base QP		**HEVC**	**ST16 + Q**
17	Bits	510176	508960
	PSNR	Y: 39.51 dB U: 43.47 dB V: 45.63 dB	Y: 39.50 dB U: 43.48 dB V: 45.62 dB
22	Bits	126624	126048
	PSNR	Y: 38.27 dB U: 42.64 dB V: 44.00 dB	Y: 38.24 dB U: 42.64 dB V: 44.00 dB
27	Bits	43544	43816
	PSNR	Y: 36.48 dB U: 41.54 dB V: 42.19 dB	Y: 36.46 dB U: 41.55 dB V: 42.18 dB
32	Bits	17552	17544
	PSNR	Y: 34.32 dB U: 40.37 dB V: 40.43 dB	Y: 34.32 dB U: 40.36 dB V: 40.49 dB
37	Bits	8280	8400
	PSNR	Y: 32.16 dB U: 39.56 dB V: 39.17 dB	Y: 32.20 dB U: 39.47 dB V: 39.16 dB
42	Bits	3680	4016
	PSNR	Y: 30.11 dB U: 38.67 dB V: 37.77 dB	Y: 30.11 dB U: 38.60 dB V: 37.70 dB

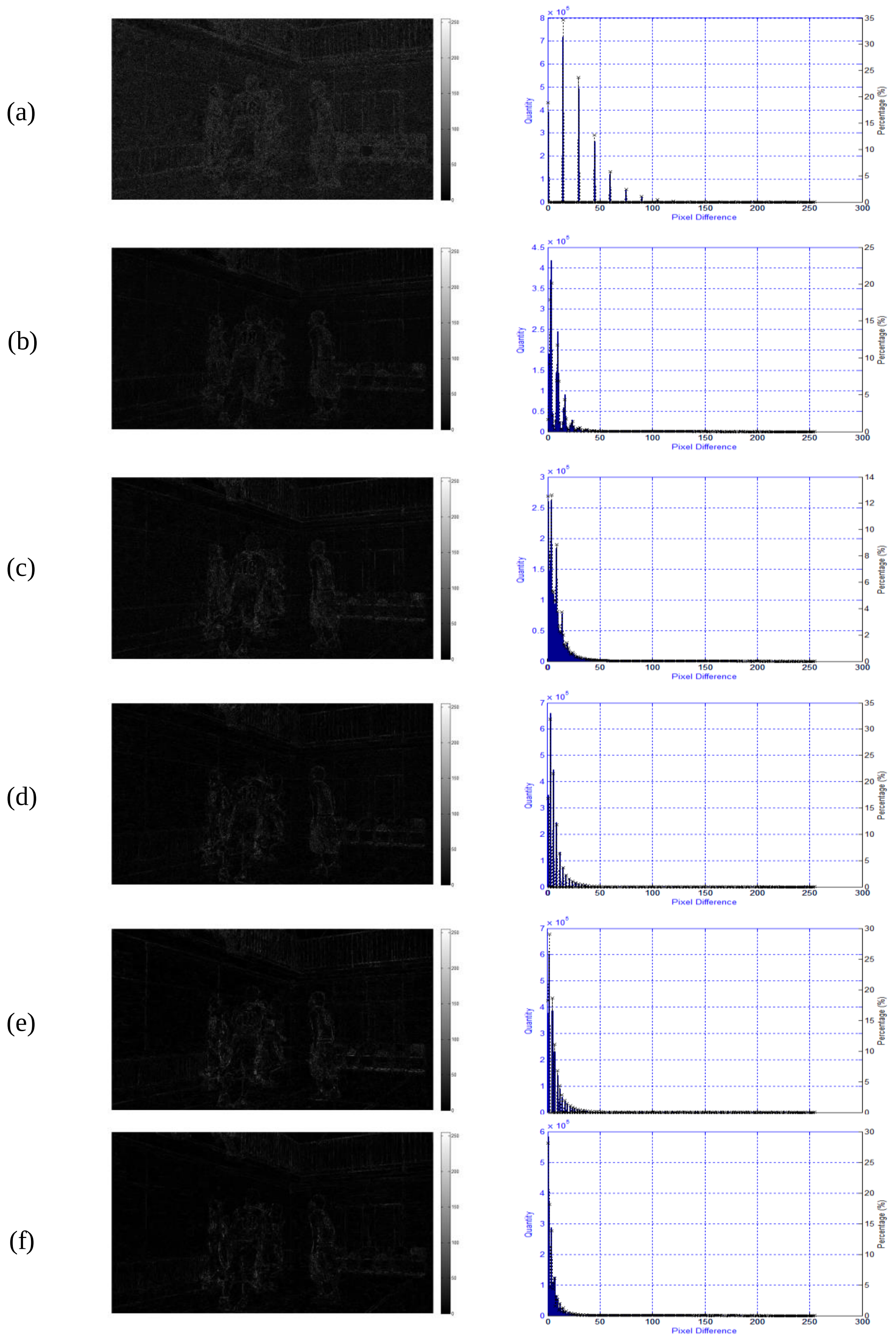


Fig. 5.23 Image differences and histograms of pixel differences of the last frame of *B4 – BasketballDrive* sequence (frame 499, RA, Main) using approximated transform and quantisation multipliers, ST16 + Q over HEVC at QP values of (a) 17, (b) 22, (c) 27, (d) 32, (e) 37, and (f) 42

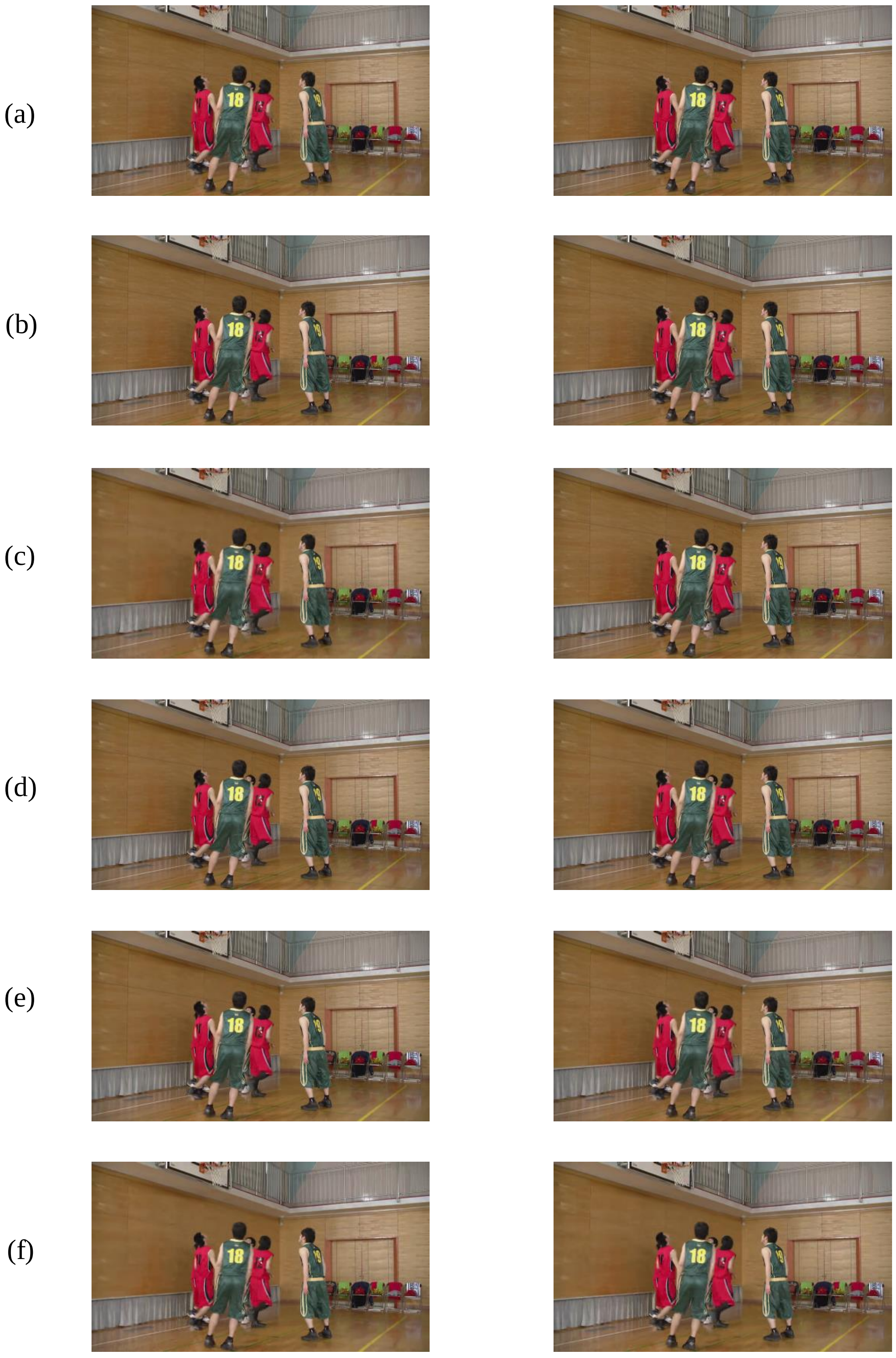
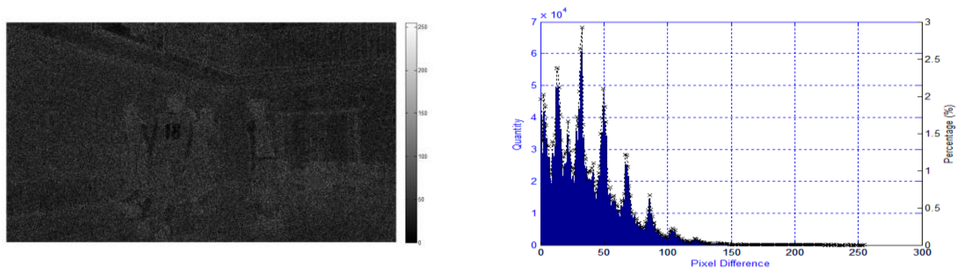


Fig. 5.24 Snapshot of the last frame of *B4 – BasketballDrive* sequence (frame 499, LB, Main) using HEVC (left column) and a combination of approximated transform and quantisation multipliers, ST16 + Q (right column), and QP values of (a) 17, (b) 22, (c) 27, (d) 32, (e) 37, and (f) 42

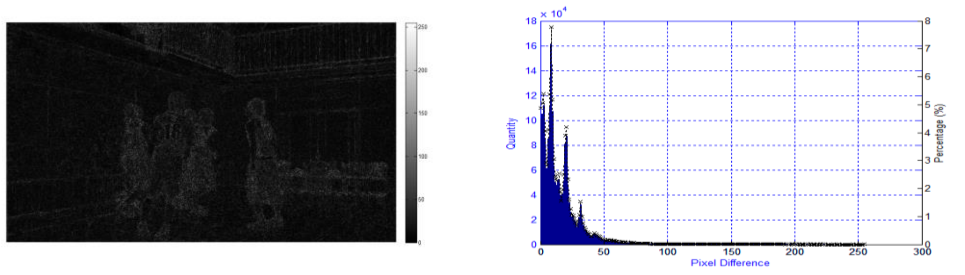
Table 5.20 Number of bits and PSNR values of last frame of *B4 – BasketballDrive* sequence under LB configuration using HEVC and approximated transform and quantisation multipliers, ST16 + Q in Main profile

Base QP		**HEVC**	**ST16 + Q**
17	Bits	1289808	1290096
	PSNR	Y: 41.77 dB U: 43.75 dB V: 45.94 dB	Y: 41.75 dB U: 43.73 dB V: 45.90 dB
22	Bits	232448	232392
	PSNR	Y: 38.92 dB U: 42.70 dB V: 44.24 dB	Y: 38.92 dB U: 42.69 dB V: 44.22 dB
27	Bits	80568	80392
	PSNR	Y: 37.08 dB U: 41.38 dB V: 42.19 dB	Y: 37.07 dB U: 41.36 dB V: 42.18 dB
32	Bits	31504	31072
	PSNR	Y: 34.85 dB U: 40.10 dB V: 40.27 dB	Y: 34.79 dB U: 40.05 dB V: 40.19 dB
37	Bits	14592	13960
	PSNR	Y: 32.54 dB U: 39.23 dB V: 38.94 dB	Y: 32.51 dB U: 39.20 dB V: 39.02 dB
42	Bits	6688	7048
	PSNR	Y: 30.28 dB U: 38.55 dB V: 37.81 dB	Y: 30.26 dB U: 38.52 dB V: 37.89 dB

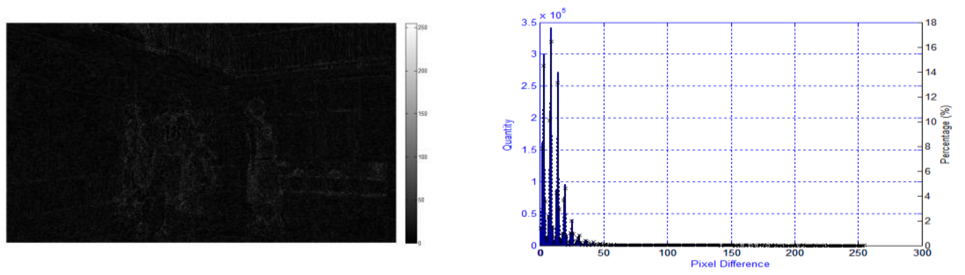
(a)



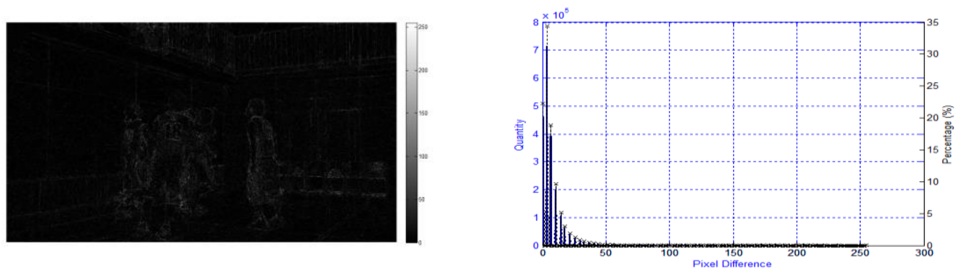
(b)



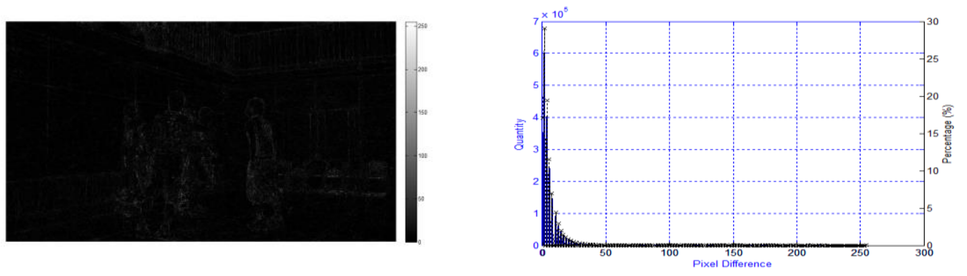
(c)



(d)



(e)



(f)

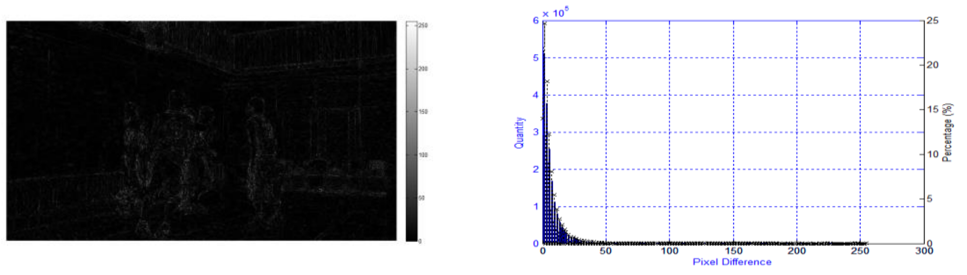


Fig. 5.25 Image differences and histograms of pixel differences of the last frame of *B4 – BasketballDrive* sequence (frame 499, LB, Main) using approximated transform and quantisation multipliers, ST16 + Q over HEVC at QP values of (a) 17, (b) 22, (c) 27, (d) 32, (e) 37, and (f) 42

Table 5.21 Percentage (%) of pixel differences of the last frame of *B4 – BasketballDrive* sequence using approximated transform and quantisation multipliers, ST16 + Q in Main Profile under LB configuration

Base QP	**Configuration / Pixel Differences**					
	RA			**LB**		
	≤ 25	**26 – 50**	**> 50**	**≤ 25**	**26 – 50**	**> 50**
17	53.49	36.47	10.04	37.70	35.64	26.65
22	95.34	4.05	0.61	82.59	14.28	3.13
27	94.38	4.66	0.96	93.65	5.54	0.80
32	96.67	2.86	0.47	95.26	4.01	0.72
37	96.32	3.09	0.59	97.87	1.85	0.28
42	97.67	2.03	0.30	96.57	2.91	0.52

5.4.5 Conclusions

Coding performances obtained by the combination of ST16 + Q in the HEVC encoder are in general similar to those obtained using only ST16 or T16 in both RA and LB configurations, with a BD-rate average value of 2.1% and 1.4% in each case, respectively. In some sequences, some BD-rate improvements could as well be seen. As noted earlier, these BD-rate increments may not be regarded as small penalties, but much higher resource savings could potentially be gained as a trade-off point. Reconstructed video frame samples achieved at least 94% and 82% of pixel differences not exceeding the value of 25 in RA and LB configuration, respectively, in base QP values of 22 and above, suggesting the suitability of ST16 + Q in general entertainment and interactive applications. However, with only around 53% and 37% of pixel differences of 25 or lower obtained in both RA and LB respectively at the base QP of 17, this combination of approximated transform and quantisation multipliers may less be desirable for encoding and transmitting superior, cinema-like videos.

5.5 Summary

In this chapter, four experiments were presented regarding approximated transforms and quantisation for HEVC. The first experiment was a pilot study on a set of approximated transform matrices, V, to see its effects across all test video classes in Main and Main 10 profiles conducted under RA and LB configurations to simulate the entertainment and interactive application scenarios, respectively. From the similar or better results obtained in both profiles and low-resolution videos, it was decided to conduct the subsequent three experiments only in Main profile and using video sequences of HD-quality or higher.

The second experiment was conducted on better-approximated transform matrices, T16 and ST16, yielding an average BD-rate difference of 1.7% and 0.7% each in RA and LB encoding structures, respectively, over four normal QP values following the CTC (Bossen, 2013). The third experiment conducted on an approximated quantisation multiplier set, Q, over six QP values provided 0.0% and -0.1% BD-rate difference on average in RA and LB case, respectively, suggesting that there is no significant difference in the coding performance despite offering some complexity savings. Nevertheless, further analysis using pixel differences from reconstructed video frame samples revealed that the suitability of Q may less be preferable in transmitting cinema-quality, big screen video signals by incorporating low QP values, if the need ever arises.

Finally, the last experiment was conducted by combining an approximated transform with the approximated quantisation, ST16 + Q, over the same six QP values yielding an average BD-rate difference of 1.7% in RA and 0.7% in LB, i.e., similar to those obtained in the second experiment using ST16 or T16 only. Additionally, pixel differences analysis supports the applicability of ST16 + Q in general entertainment and interactive use cases, i.e., when coupled with QP values of 22 or greater, but less suitable with lower QP values such as for motion picture productions and transmissions.

Even though the bitrate increments in order to reach the same objective quality levels as the original HEVC transform and quantisation cannot be regarded as small penalties, the potential resource savings could be more favourable in a complexity-reduced encoder to produce HEVC-like bitstreams.

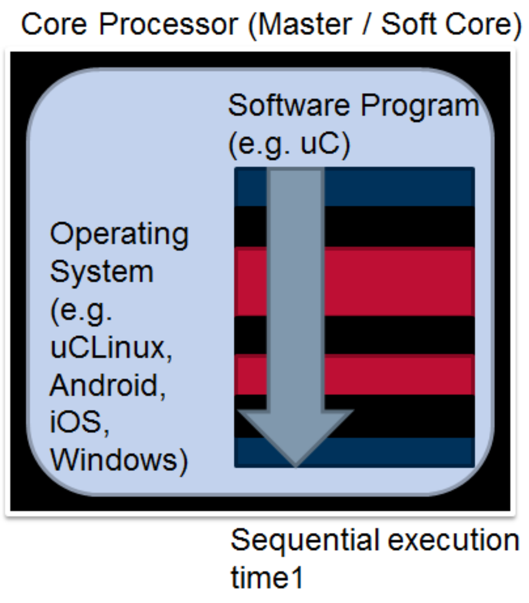
Chapter 6

Dedicated Hardware Architecture Designs for Approximated Transform, Intermediate Scaling, and Approximated Quantisation

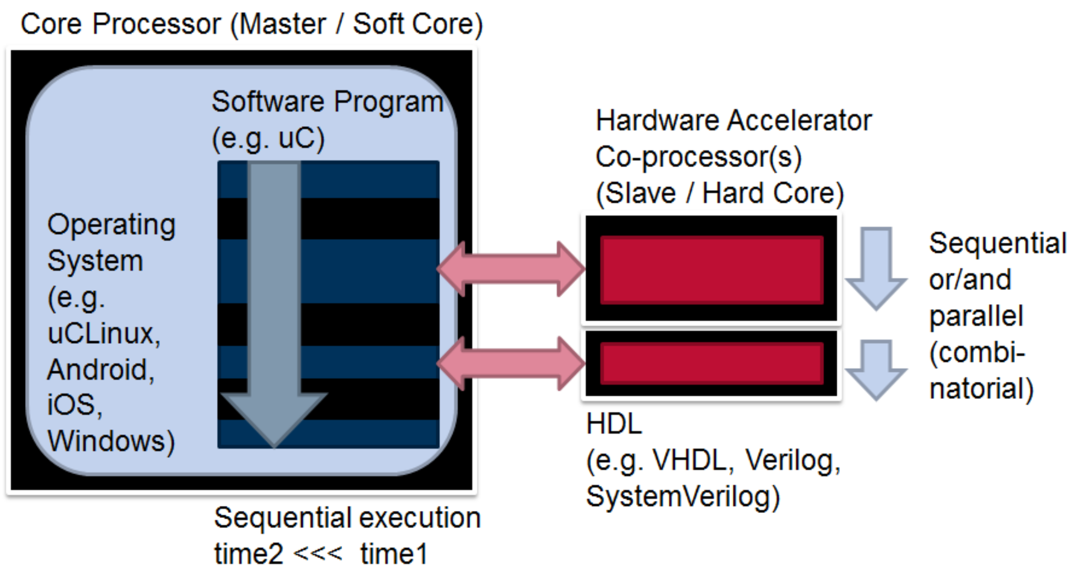
Abstract In order to estimate the potential hardware savings offered by the approximated transform matrix sets, T16 and ST16, as well as the approximated quantisation, Q, described thus far in this thesis, over the original HEVC algorithms, this chapter presents architecture designs developed in this work for hardware comparison purposes targeted on FPGA technology. Some care was taken to maintain the necessary similarities between the designs for a fair comparison and differ only in the core transform (including intermediate scaling) and/or quantisation processing blocks.

6.1 Hardware-Software Co-design Methodology

A computer program written in a High-Level Language (HLL) such as C++ can normally be run on a processor IC. This processor has an operating system (OS) residing in it such as Linux, Windows, iOS, Android, etc. An HLL program is executed in a sequential manner from the top to the bottom. Some programs such as a video encoder have many computationally intensive functions which are considerably more time consuming and resource demanding than the rest of the program. Executing a large program only on software basis would possibly be too impractical especially for applications requiring a real-time performance. A solution to this problem is hardware acceleration or also known as hardware-software co-design, where the most complex functions are offloaded from the main or master core processor to slave hardware co-processor(s) or accelerator(s). These functions are implemented by designing dedicated hardware architectures written in a Hardware Description Language (HDL) such as VHDL, Verilog, SystemVerilog, etc. (Fig. 6.1). An HDL program can be executed in sequential, parallel or concurrent, and combinational (combination of sequential and parallel). Having dedicated hardware designs speed-up the operations of complex functions, thus allowing higher data throughputs necessary for many applications such as video processing. Even running these hardware designs in a sequential manner would yield a much smaller execution time in comparison to a fully software-based implementation.


Core Processor (Master / Soft Core)
Software Program (e.g. uC)
Operating System (e.g. uCLinux, Android, iOS, Windows)
Sequential execution time1

(a)


Core Processor (Master / Soft Core)
Software Program (e.g. uC)
Operating System (e.g. uCLinux, Android, iOS, Windows)
Sequential execution time2 <<< time1
Hardware Accelerator Co-processor(s) (Slave / Hard Core)
Sequential or/and parallel (combi-natorial)
HDL (e.g. VHDL, Verilog, SystemVerilog)

(b)

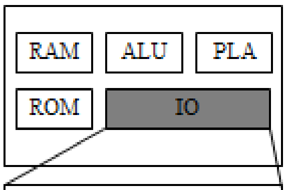
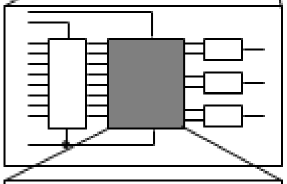
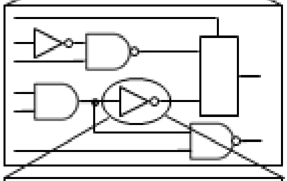
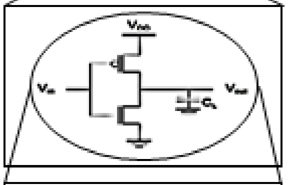
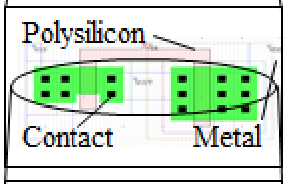
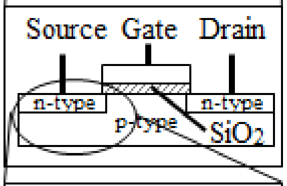
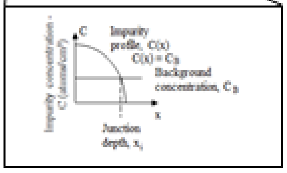
Fig. 6.1 Hardware acceleration concept with (a) fully software-based implementation and (b) hardware-software co-design

Complexity and time-to-market are among crucial factors influencing the success of digital circuits. Two out of many developed techniques to handle these issues are design abstraction and hierarchical modular design. Typically in the design of digital circuits, design abstraction levels in the increasing order are the device, circuit, gate, functional module and architectural or system level (Table 6.1). At each abstraction level, the internal details of a complex digital system may be represented by a black box model, where this model contains all the necessary information required by the module(s) at the immediate lower level of the design hierarchy. These abstraction levels usually involve different design teams and could as well be located at multiple sites. Having such black box models may not only substantially reduce the design complexity, but also reduce design lead times especially in meeting performance goals of Very Large Scale Integrated Circuits (VLSI) (Hani, 2011).

Modularisation applies the concept of “divide and conquer”, which is attributed to King Philip II of Macedon (382–336 BC), to efficiently design any complex system. The complexity of a design is conquered in such a way that a high-level module is broken down or divided into a hierarchy of simpler modules, i.e., from the general and conceptual at the top, to the details at the bottom. By employing a hierarchical modular design approach, the focus can be given to a single module at a time, without being hindered by the complexity of the entire circuit. On top of that, reuse of primitive or customised low-level modules can be made without the need to redesigning the same modules each time. A smaller amount of effort would also be necessary if these modules require minor future improvements (Hani, 2011).

A hierarchical modular implementation involves two approaches which are normally used together: 1) top-down and 2) bottom-up (Fig. 6.2). The top-down approach decomposes a system into subsystems, where these subsystems can also be further decomposed into simpler subsystems until a low enough level is reached such that modules at this level can easily be implemented. Conversely, the bottom-up approach connects the available or already developed modules to form subsystems, and these subsystems can be further connected to form larger and more complex subsystems until the complete operation is achieved.

Table 6.1 Typical digital design abstraction levels (Hani, 2011)

Graphical view	Level	Primitive units	Parameters of concern
	System / Architectural Level	Behavioural modules	Silicon area
	Register Transfer Level (RTL)	Functional modules	Timing
	Logic Level	Gates, Bits	Delays (transitions / propagations)
	Circuit Level	Transistors	Voltage, Currents
	Layout / Physical Level	Layout layers	Topology, Dimensions
 Source Gate Drain n-type n-type p-type SiO2	Device Level	MOSFET models	Current-Voltage characteristics
	Technology Level	Process models	Impurity profiles

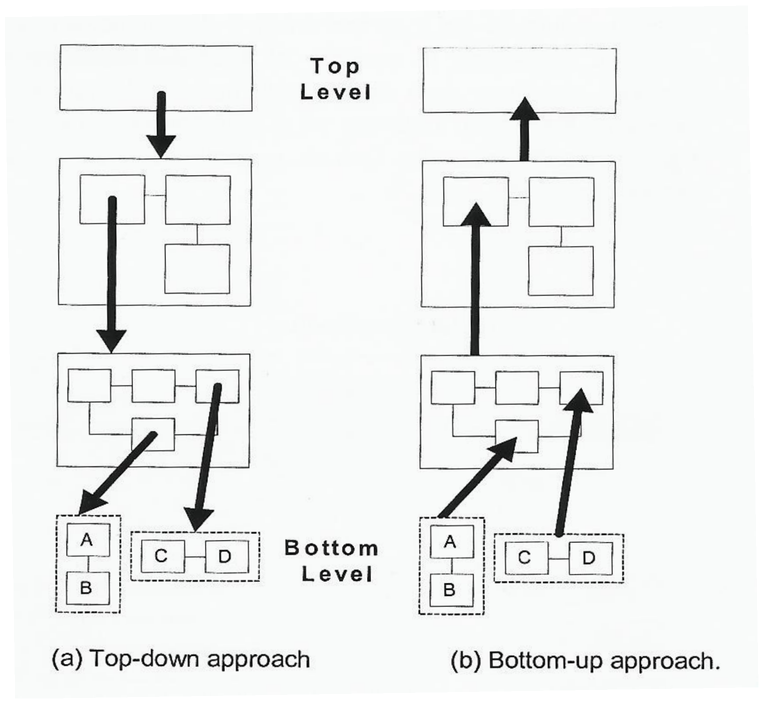


Fig. 6.2 Hierarchical modular design approach (Hani, 2011)

6.2 Hardware Architecture Designs for Approximated Transform and Intermediate Scaling

This section briefly describes the 2-D transform hardware architecture designed in this work. While some details have been left out, the walkthrough information provided here is expected to facilitate the understanding of the designs. These designs are intended as hardware slave co-processors. However, the master processor and necessary interface modules are not developed in this work.

6.2.1 Top-level Transform Module (TM)

By utilising the retained separable property of the transform core of HEVC and the approximated sets, i.e., the 2-D transform can effectively be implemented as two 1-D transforms with an intermediate transpose operation between them, the hardware architecture designs herein presented adopt the column-row decomposition approach. Furthermore, it was assumed that the input samples (prediction residuals) to the transform stage arrive serially at a rate of one pixel/cycle. The output 2-D transform coefficients were designed to be transferred in parallel at N pixels/cycle, where N = 4, 8, 16, or 32.

Fig. 6.3 depicts the top-level functional block diagram of 2-D transform hardware architecture developed in this work. This base architecture design consists of a data path module (DM) and a control module (CM). The DM comprises a serial-to-parallel (S2P) block, a 1-D transform block, a rounding and scaling (RS) block, a transpose buffer, a second 1-D transform block, and a second RS block. For each transform matrix (HEVC, T16, or ST16), the two 1-D transform blocks are exactly the same modules used twice. On the other hand, the two RS blocks slightly differ from each other depending on the necessary rounding and scaling for the first or second stage of the transform operation.

The CM schedules the flow of operations by providing a control vector (a 5-bit signal, *CV*) to the DM depending on the transform size to be executed. The CM was designed based on a Mealy Finite State Machine (FSM). Unlike DM, the CM operates on the falling edge of the master clock (*clock*).

The input signals of the architecture are *n*-bit prediction residual (X_{rc}), 2-bit *size*, *start*, *reset*, and *clock* signals. The outputs are 32 16-bit 2-D transform coefficients (T_{rc}), *valid* signal to indicate when the outputs are ready for the next process, and *done* signal to notify the last column of outputs. To maintain precision, the internal bit width after the first 1-D transform core, *m*, varies depending on the bit width of the input, X_{rc}.

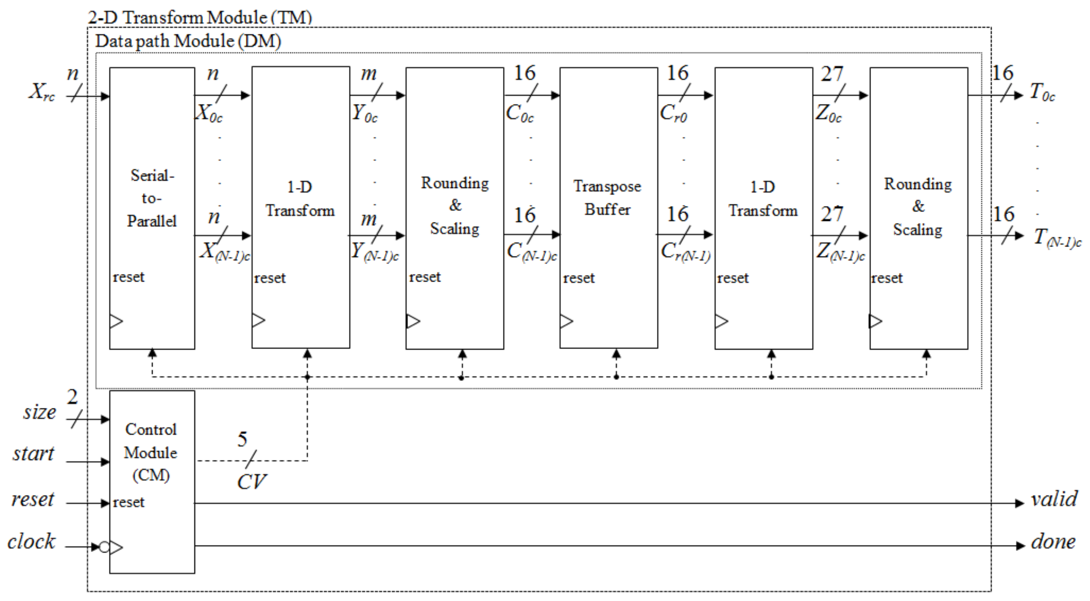


Fig. 6.3 Top-level functional block diagram of 2-D transform architecture

6.2.2 Data path Module (DM)

Serial-to-parallel (S2P) block: The serial-to-parallel (S2P) block was designed to transfer in parallel the input samples to the following 1-D transform block for all four sizes supported by HEVC. It consists of 32 × 32 *n*-bit registers (Fig. 6.4). It was also designed to ensure a continuous operation of the transform block on all columns of the input block without stalling for the next complete column. For an input block of size $N \times N$, the first column is transferred in the first N clock cycles. At the next cycle ($N + 1$), this whole column is transferred to the next register column of the S2P. This is repeated for the next ($N – 1$) columns. The latency of this block is, therefore, ($N^2 + 1$) cycles. For the transform size of 4, 8, 16, and 32, the latencies are 17, 65, 257, and 1025 cycles, respectively. The horizontal and vertical flows of the input data are controlled by the ld_0 and ld_1 signals, respectively.

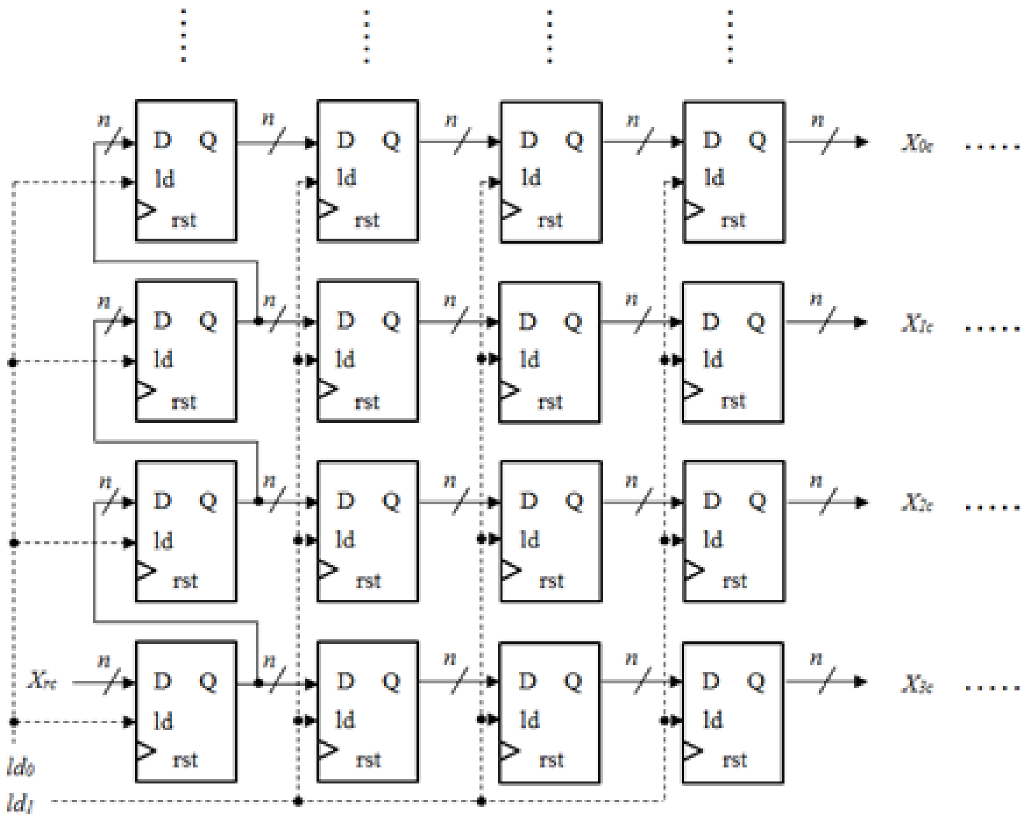
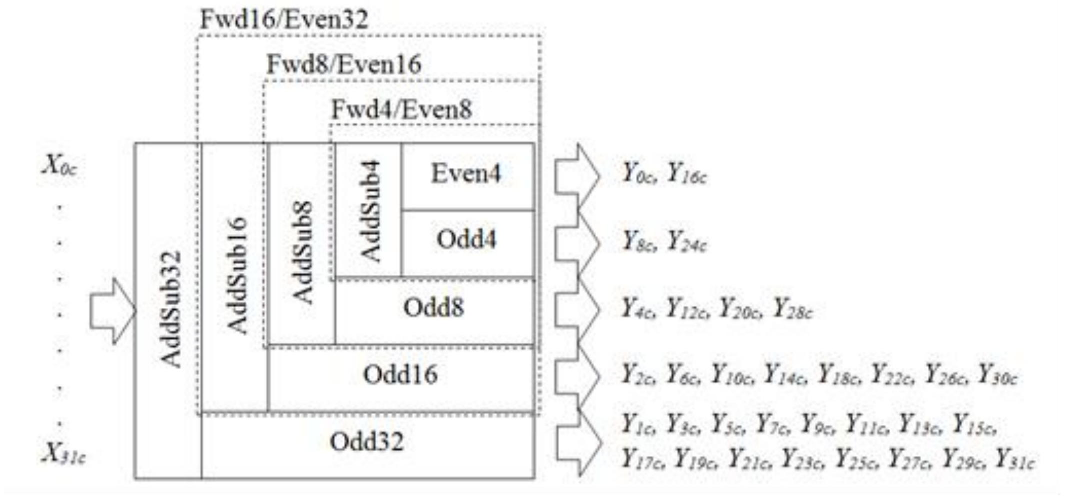
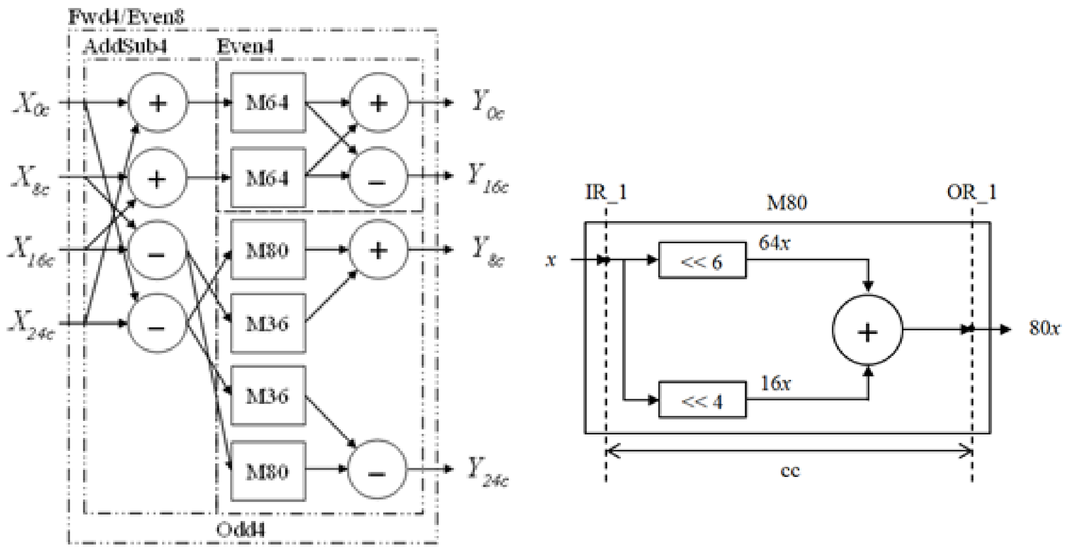


Fig. 6.4 Functional block diagram of serial-to-parallel block (for clarity reason, the clock and reset signals are not explicitly shown)

1-D transform block: The two 1-D transform blocks were designed using the even-odd decomposition approach as depicted in Fig. 6.5 (a). Each transform size consists of three basic modules: 1) AddSubN; 2) EvenN; 3) OddN. For *N* larger than four (4), the EvenN part is made up of the three modules of *N*/2 transform. The 4-point forward approximated transform, T16, is depicted in Fig. 6.5 (b), where multiplier-free multiplications of 64, 80, and 36 are implemented as illustrated in Fig. 6.5 (c), where IR_1 and OR_1 represent the internal input and output register, respectively. Similar designs were applied for ST16, where the multiplication blocks are down-scaled by four (16, 20, and 9). For HEVC transform blocks, the corresponding multiplications are implemented as previously shown in Fig. 4.1. The total clock cycle taken in the 4/8/16/32-point 1-D transform block is 3/4/5/6 cycles.


Fwd16/Even32
Fwd8/Even16
Fwd4/Even8
X_{0c}
X_{31c}
AddSub32
AddSub16
AddSub8
AddSub4
Even4
Odd4
Odd8
Odd16
Odd32
Y_{0c}, Y_{16c}
Y_{8c}, Y_{24c}
Y_{4c}, Y_{12c}, Y_{20c}, Y_{28c}
Y_{2c}, Y_{6c}, Y_{10c}, Y_{14c}, Y_{18c}, Y_{22c}, Y_{26c}, Y_{30c}
Y_{1c}, Y_{3c}, Y_{5c}, Y_{7c}, Y_{9c}, Y_{11c}, Y_{13c}, Y_{15c},
Y_{17c}, Y_{19c}, Y_{21c}, Y_{23c}, Y_{25c}, Y_{27c}, Y_{29c}, Y_{31c}

(a)



(b) (c)

Fig. 6.5 Functional block diagram of (a) 1-D forward transform block using even-odd decomposition, (b) 4-point approximated transform (T16), and (c) multiplier-free multiplication by 80

Rounding and Scaling (RS) block: The rounding and scaling (RS) blocks perform (6.1)–(6.2) to ensure that the intermediate transform coefficients stay within 16-bit width, where the values of *offset* and *shift* are summarised in Table 6.2 for 8-bit input bit width.

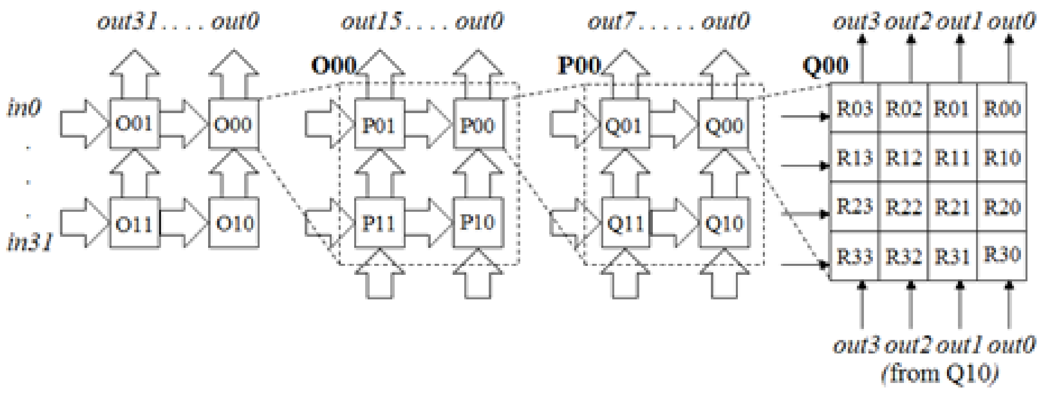
$$R_{rc} = Y_{rc} + offset \tag{6.1}$$

$$S_{rc} = R_{rc} >> shift \tag{6.2}$$

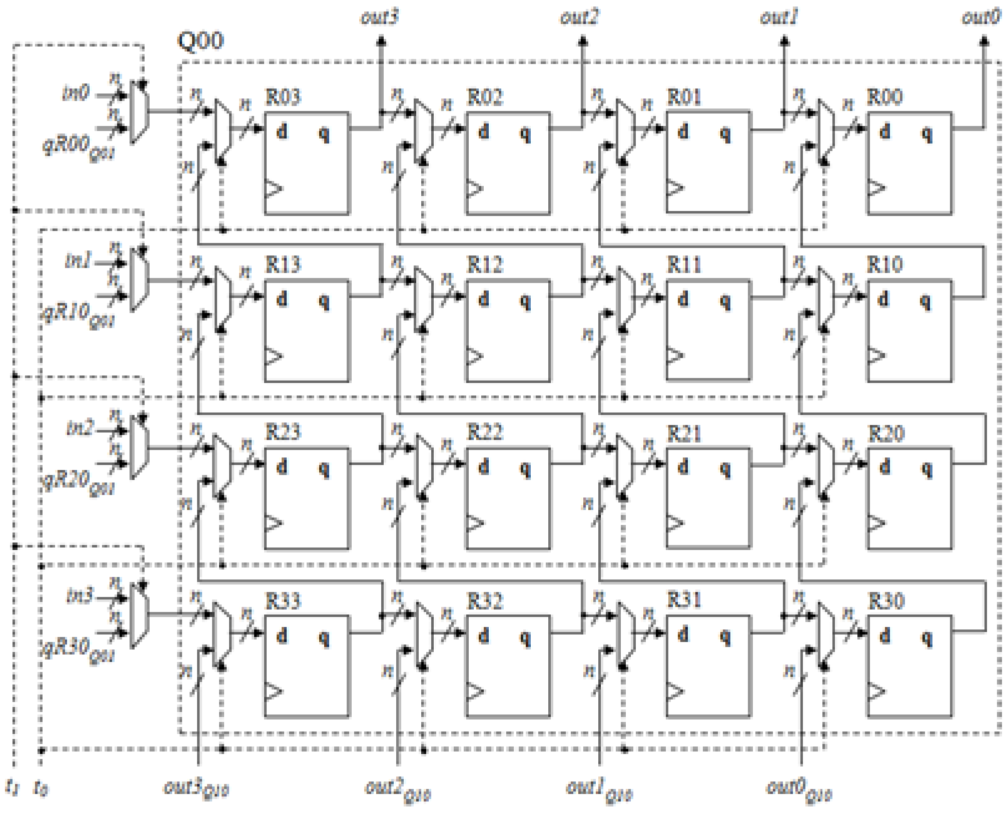
Table 6.2 Offset and shift values in RS stage (for 8-bit input bit width)

Size	First 1-D Transform		Second 1-D Transform	
	Shift	***Offset***	***Shift***	***Offset***
4 × 4	1	2^0	8	2^7
8 × 8	2	2^1	9	2^8
16 × 16	3	2^2	10	2^9
32 × 32	4	2^3	11	2^{10}

Transpose buffer: The transpose buffer designed in this architecture has a basic structure of four levels of 4-by-4 register array as illustrated in Fig. 6.6 (a). The horizontal or vertical flow of data inside this block is controlled internally by the 2-bit t signal (Fig. 6.6 (b)).



(a)



(b)

Fig. 6.6 Functional block diagram of (a) transpose buffer and (b) basic 4-by-4 register array (for clarity reason the *clock* and *reset* signals are not explicitly shown)

6.2.3 Control Module (CM)

The Control Module (CM) in this architecture consists of a Next State (NS) logic module, a Present State (PS) register, an Output Logic module, and two 5-bit up-counters (Fig. 6.7). The output of the CM is a 9-bit control signal, namely *Control Vectors* (*CV*[8:0]). Five Most Significant Bits (MSBs) of *CV*, *CV*[8:4], are delivered to the DM to schedule its operations according to the transform size (*size*) to be performed. *CV*[8:7] respectively provide ld_1 and ld_0 signals to control the vertical and horizontal data flow in the S2P module, *CV*[6] provides t_0 to the transpose buffer to transfer data horizontally or vertically for the transpose operation, and C*V*[5:4] provide the *valid* and *done* signals respectively, to notify the master device on the status of the whole transform operation. The four Least Significant Bits (LSBs), *CV*[3:0] provide internal clear (*clr*0 and *clr*1) and load (*ld*0 and *ld*1) signals to the two counters: Counter0 and Counter1. The design is based on a Mealy Finite State Machine (FSM) model as shown in Fig. 6.8. The two counters provide internal count signals, *i* and *j*, respectively, to execute nested loops and perform the FSM accordingly. The *start* and *clear* input signals instruct the CM to begin its operation and reset all the internal registers or counters, respectively.

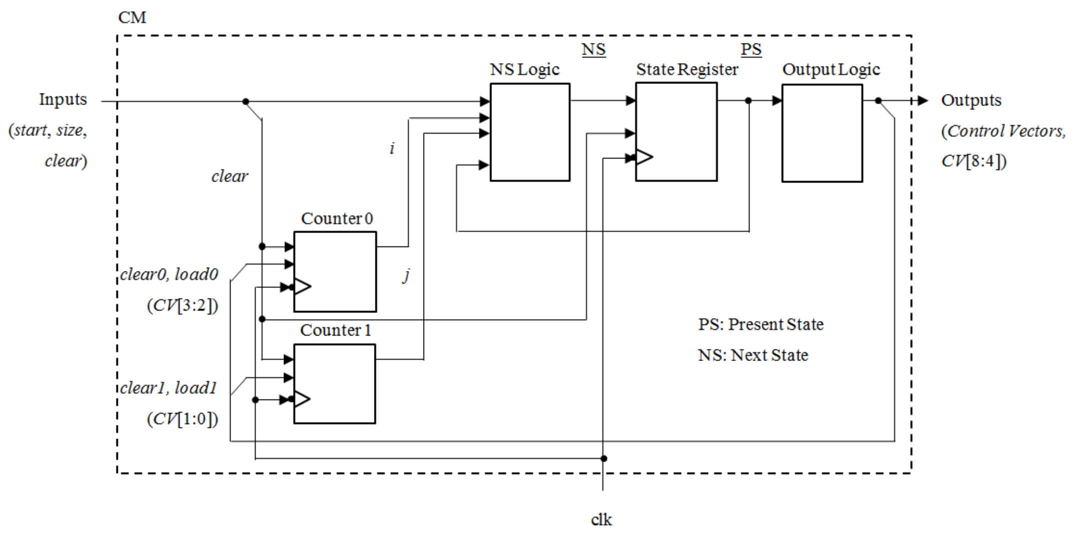


Fig. 6.7 Functional block diagram of Control Module (CM)

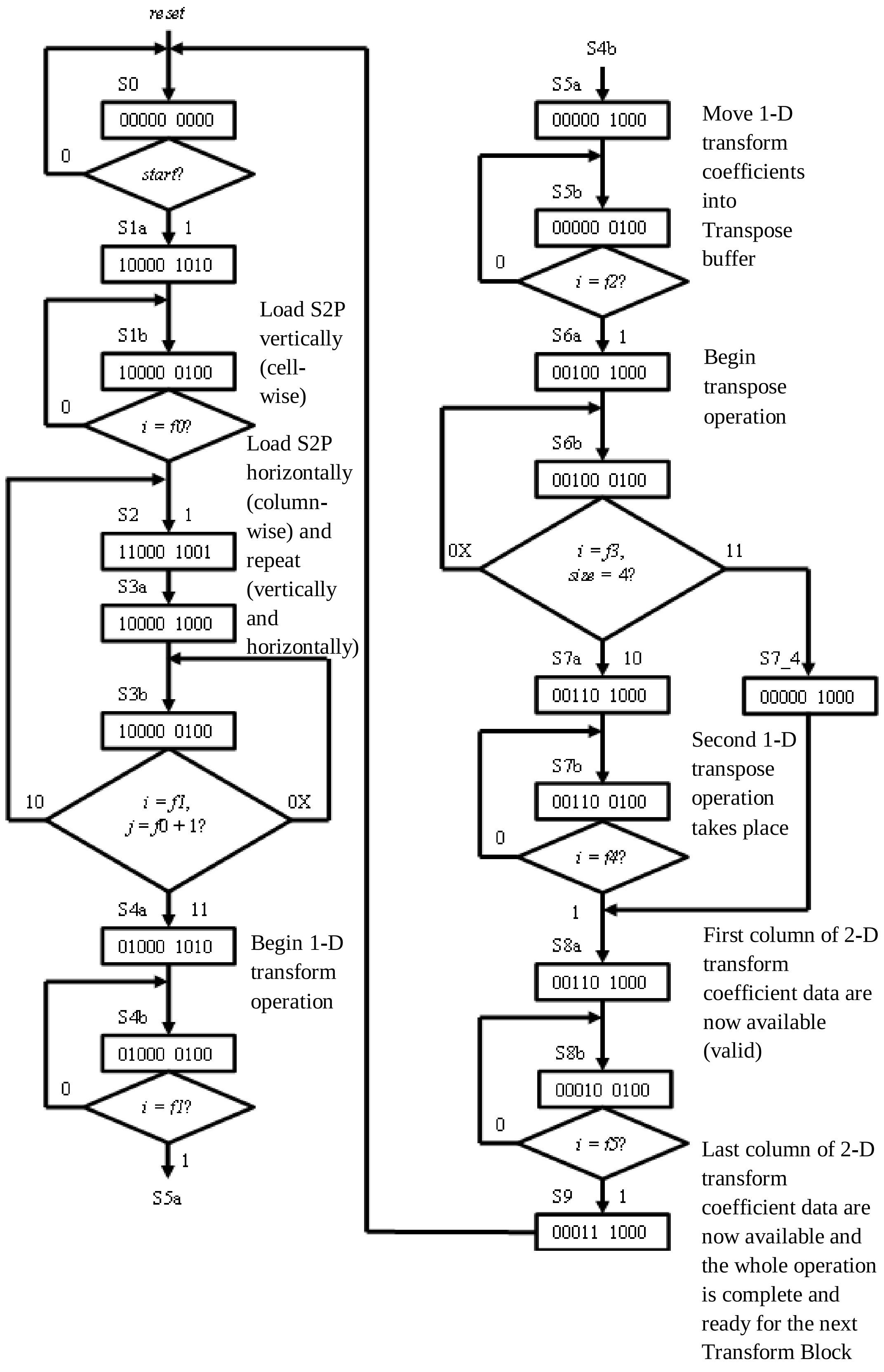

reset
S0
00000 0000
start?
0
1
S1a
10000 1010
S1b
10000 0100
i = f0?
0
1
Load S2P vertically (cell-wise)
S2
11000 1001
Load S2P horizontally (column-wise) and repeat (vertically and horizontally)
S3a
10000 1000
S3b
10000 0100
i = f1, j = f0 + 1?
10
0X
11
S4a
01000 1010
Begin 1-D transform operation
S4b
01000 0100
i = f1?
0
1
S5a
S4b
S5a
00000 1000
Move 1-D transform coefficients into Transpose buffer
S5b
00000 0100
i = f2?
0
1
S6a
00100 1000
Begin transpose operation
S6b
00100 0100
i = f3, size = 4?
0X
11
10
S7a
00110 1000
S7_4
00000 1000
S7b
00110 0100
Second 1-D transpose operation takes place
i = f4?
0
1
S8a
00110 1000
First column of 2-D transform coefficient data are now available (valid)
S8b
00010 0100
i = f5?
0
1
S9
00011 1000
Last column of 2-D transform coefficient data are now available and the whole operation is complete and ready for the next Transform Block

Fig. 6.8 Finite State Machine of 2-D transform architecture designed in this work (with some rough descriptions included)

6.2.4 Functional Verification

The architecture designs of the 2-D HEVC and approximated core transforms, T16 and ST16, were described in parametric IEEE-VHDL and routed to a Xilinx Virtex-6 xc6vl550t-2ff1760 FPGA as the target device using Xilinx ISE 14.7 tool chain. Functional simulations were performed for all three designs using test benches and test vectors in the Xilinx ISim environment and the results were verified with MATLAB software. Due to the complexity of the whole transform operation, the validation was performed only at transform block (TB) level using random *n*-bit signed residual values for all four transform sizes. The 2-D transform coefficients obtained were then compared to ensure behavioural correctness.

More thorough validations would certainly be more convincing, such as at slice, frame, or sequence level. Considering all three YUV components by evaluating at transform unit (TU) level would certainly be better. However, these recommendations would further complicate the design stage. Finally, a real physical implementation on hardware such as an FPGA board would clear many doubts about the usefulness of the designs. Unfortunately, all these tasks were not feasible to be realised in this work due to some practical constraints.

6.2.5 Results and Discussions

Table 6.3 provides the latencies and execution times of these designs. For a fair comparison, the designs for HEVC and approximated T16 and ST16 transforms were made similar such that they would yield the same latencies and execution times. Table 6.4 summarises the resource utilisation of HEVC and approximated transforms. From this table, in the case of 9-bit input signals, T16 and ST16 matrices utilise 38,507 and 38,383 slice registers respectively as opposed to 46,172 slice registers consumed by HEVC transform matrices, i.e., a reduction of 16.6% and 16.9% respectively. Likewise, savings of 19.6% and 21.7% were observed in the number of slice LUTs to implement the bit shifts and addition operations in the T16 and ST16 transform matrices when compared with HEVC transform matrices. Eight registers and four LUTs are contained in a Xilinx Virtex-6 slice (Xilinx, 2015). T16 and ST16 hardware designs consume 16.9% and 20.8% fewer slices than HEVC transform, respectively.

Table 6.3 Latency and execution times (clock cycles) of 2-D transform architecture designs

Stage	**Size**			
	4 × 4	**8 × 8**	**16 × 16**	**32 × 32**
S2P	17	65	257	1025
1-D Transform	3	4	5	6
RS	2	2	2	2
Transpose	5	9	17	33
1-D Transform	3	4	5	6
RS	2	2	2	2
Total Latency	32	86	288	1074
Execution Time	36	94	304	1106

Table 6.4 Resource utilisation of 2-D HEVC and approximated transform architecture designs

Parameter	**Transform**		
	HEVC	**T16**	**ST16**
Input bit width (*n*)	9	9	9
Internal bit width (*m*)	20	20	20

Slice registers (Savings %)	46,172	38,507 (16.6%)	38,383 (16.9%)
Slice LUTs (Savings %)	51,448	41,342 (19.6%)	40,291 (21.7%)
Slices (Savings %)	14,023	11,649 (16.9%)	11,100 (20.8%)
Operating Freq. (MHz)		200	
Output rate (per cycle)		4 / 8 / 16 / 32	
Throughput (×10^9 samples per second)		0.8 / 1.6 / 3.2 / 6.4	

Table 6.5 compares the implementations of the approximated transforms with other FPGA implementations. The works by Conceição *et al.* (2013) and Zhao and Onoye (2012) described earlier in sub-section 3.7.1 were implemented in Altera FPGAs, thus a direct comparison with our work is not so feasible. This thesis used the same FPGA device as Kalali *et al.* (2014), Xilinx Virtex-6 xc6vl550t-2, thus the comparison is more appropriate although their work was on the inverse transform. Additionally, although the exact FPGA type is not specified in (Kalali, Mert and Hamzaoglu, 2016), their results are also included in Table 6.5 as the FPGA used is also fabricated in 40 nm CMOS technology, which is the case for Xilinx Virtex-6. The recent work by da Silveira *et al.* (2017) also used a Xilinx Virtex-6 FPGA, but their work only covered a 1-D 16-point transform architecture. The recent work by Chen, Zhang and Lu (2017) used more advanced Xilinx Virtex-7 and Zynq FPGAs as well as Altera FPGAs, while Jridi and Meher (2016) used an older Xilinx Spartan 6 LX45T FPGA.

Both T16 and ST16 designs use about triple slice registers more than reported in (Kalali *et al.*, 2014; Kalali, Mert and Hamzaoglu, 2016). One probable reason is due to the transpose buffer implementation using the register array instead of on-chip memory, utilising 16-bit × 32 × 32, i.e., 16,384 registers instead of 32 BRAMs or 2 KB of memory. Other reasons are possibly the implementation of the RS block instead of only the clipping block in (Kalali *et al.*, 2014; Kalali, Mert and Hamzaoglu, 2016), in addition to the internal bit-width of 27-bit used on a few stages. An obvious advantage of excluding any use of on-chip memory is the higher operating frequency achieved by T16 and ST16 designs (200 MHz) than all designs in (Kalali *et al.*, 2014; Kalali, Mert and Hamzaoglu, 2016). Consequently, at a minimum throughput of 800 mega samples per second, T16 and ST16 designs are able to process more QFHD frames per second (60 fps) than (Kalali *et al.*, 2014) (48 fps) and the LU hardware in (Kalali, Mert and Hamzaoglu, 2016) (48 or 56 fps). As a result, the minimum hardware efficiencies of T16 (0.0687 mega samples/sec/slice) and ST16 (0.0721 mega samples/sec/slice) are higher than those designs (0.0397 – 0.0529 mega samples/sec/slice). The HU hardware in (Kalali, Mert and Hamzaoglu, 2016) have been well developed to sustain the processing of UHD videos of more than 50 fps or QFHD @ 120 fps despite operating at lower frequencies (111 or 117 MHz) than T16 and ST16. Their HU designs are roughly twice more hardware efficient (0.1660 and 0.1397 mega samples/sec/slice) than T16 and ST16.

Nevertheless, the initial aim of this work was to demonstrate the potential savings from the approximated core transforms, T16 and ST16, when compared with HEVC core transforms on the same hardware platform and using the same design principles as depicted earlier in Table 6.4. At a 200 MHz operating frequency and a minimum throughput of four samples per cycle, the design is capable of encoding a 4:2:0 QFHD @ 60 fps video ($3840 \times 2160 \times 60 \times 1.5 = 0.746496 \times 10^9$ samples per second).

Table 6.5 Resource utilisation of 2-D transform architecture designs

Parameter	**Design**			
	Kalali 2014 (Kalali *et al.*, 2014)	**Kalali 2016** (Kalali, Mert and Hamzaoglu, 2016)	**This work (T16)**	**This work (ST16)**
FPGA Technology	Xilinx Virtex-6	40 nm CMOS	Xilinx Virtex-6	Xilinx Virtex-6
Transform	2-D Inv.	2-D Inv.	2-D Fwd.	2-D Fwd.
Size	4/8/16/32	4/8/16/32	4/8/16/32	4/8/16/32
Internal bit width	n.a.	n.a.	≤ 27	≤ 27
Slice registers	11,762[a] 11,763[b]	11,110[c] 11,230[d] 12,025[e] 12,200[f]	38,507	38,383
Slice LUTs	38,790[a] 38,821[b]	33,376[c] 35,555[d] 38,006[e] 41,905[f]	41,342	40,291
Slices	11,343[a] 11,397[b]	9,797[c] 10,080[d] 11,279[e] 12,712[f]	11,649	11,100
Memory (BRAMs)	32	4/8/16/32[c & d] 8/8/16/32[e & f]	-	-
Multipliers	No	No	No	No
Others	Clipping	Clipping	Round and Scale	Round and Scale
Operating freq. (MHz)	150	116[c] 100[d] 117[e] 111[f]	200	200
Throughput ($\times 10^9$ samples per second)	0.6/1.2/2.4/4.8	0.464/0.928/1.856/3.712[c] 0.4/0.8/1.6/3.2[d] 1.872/1.872/1.872/3.744[e] 1.776/1.776/1.776/3.552[f]	0.8/1.6/3.2/6.4	0.8/1.6/3.2/6.4

Supported resolution @ fps	QFHD @ 48	QFHD @ 56[c] QFHD @ 48[d] UHD @ 56[e] UHD @ 53[f]	QFHD @ 60	QFHD @ 60
Hardware efficiency[g] (×10^6 samples/sec/slice)	0.0529[a] 0.0526[b]	0.0474[c] 0.0397[d] 0.1660[e] 0.1397[f]	0.0687	0.0721

[a] MCM hardware
[b] MCM + Energy hardware
[c] MCM LU hardware
[d] MCM + Energy LU hardware
[e] MCM HU Hardware
[f] MCM + Energy HU Hardware
[g] Hardware efficiency = Min Throughput (×10^6 samples/second)/Slices

6.2.6 Conclusions

The architecture designs of the 2-D HEVC and approximated core transforms, T16 and ST16, were presented in this section. The designs were implemented on a Xilinx Virtex-6 FPGA device adopting the even-odd decomposition, multiplier-free, and MCM approaches. Savings of 16.9% and 20.8% in the number of slices were obtained by T16 and ST16 designs respectively over HEVC transform (Table 6.4). Comparing with similar works in the literature (Kalali *et al.*, 2014; Kalali, Mert and Hamzaoglu, 2016), the T16 design utilises slightly more slices (11,649) and the ST16 design uses slightly fewer slices (11,100) than (Kalali *et al.*, 2014) (11,343 or 11,397) (Table 6.5), while the number of slices in (Kalali, Mert and Hamzaoglu, 2016) vary between 9,797 and 12,712. Despite using approximated transform matrices, not much difference could be seen in the number of slices in both T16 and ST16 designs over (Kalali *et al.*, 2014; Kalali, Mert and Hamzaoglu, 2016), mainly due to the implementation of the transpose buffer using register arrays instead of BRAMs as applied in (Kalali *et al.*, 2014; Kalali, Mert and Hamzaoglu, 2016). However, both T16 and ST16 designs could operate at a higher frequency (200 MHz compared to 100 – 150 MHz) and are capable of processing more QFHD frames per second than (Kalali *et al.*, 2014) and the LU hardware designs in (Kalali, Mert and Hamzaoglu, 2016). Higher hardware efficiencies could also be achieved by T16 (0.0687 mega samples/sec/slice) and ST16 (0.0721 mega samples/sec/slice) than (Kalali *et al.*, 2014) (0.0529 or 0.0526 mega samples/sec/slice) and the LU designs in (Kalali, Mert and Hamzaoglu, 2016) (0.0397 or 0.0474 mega samples/sec/slice). However, the HU designs in (Kalali, Mert and Hamzaoglu, 2016) are twice more hardware efficient (0.1660 and 0.1397 mega samples/sec/slice) than T16 and ST16, and capable of supporting UHD videos of more than 50 fps. In summary, the approximated transform schemes as adopted by T16 and ST16 hardware architecture designs may yield a slightly better performance than HEVC-compliant hardware architecture and considerable for a complexity-reduced HEVC-like encoder hardware implementation.

6.3 Hardware Architecture Designs for Approximated Quantisation

This section briefly describes the quantisation hardware architecture design developed in this work. Similar to the transform design described previously, the quantisation module (QM) is intended as a hardware slave or co-processor. The master core processor and necessary interface modules were not developed.

6.3.1 Top-level Quantisation Module (QM)

Unlike the transform module (TM), the developed quantisation module (QM) only consists of data path module (DM) and does not include a control module (CM) due to its relatively simpler algorithm compared to the 2-D transform. Besides the *clock* and *reset* signals, the inputs to the module are 16-bit transform coefficients (T_{0c}, …, T_{31c}) coming from the TM, 3-bit *sel*, 5-bit *total_scale*, *n*-bit *offset*, and *load* signals (Fig. 6.9). The *sel* signal selects which quantiser value to be performed depending on the QP value as summarised in Table 6.6. The *total_scale* signal performs the right bit shift operation according to *total_scale* = QP/6 + *shift2* as described in Section 4.3. Similarly, the *offset* can be added as desired. The *sel*, *total_scale*, and *offset* signals are expected to be provided by the master core processor and not included in the hardware design of QM. Finally, the *load* signal enables the quantised transform coefficients or levels (Q_{0c}, …, Q_{31c}) to be loaded into the corresponding internal registers as the output of QM and takes the *valid* signal from the TM. This signal is also carried over as the *validQ* output signal to indicate to the master processor or another processing module that the quantised levels are available. The Quantisation unit inside QM performs the multiplier-free quantisation operation as described in Section 4.3.

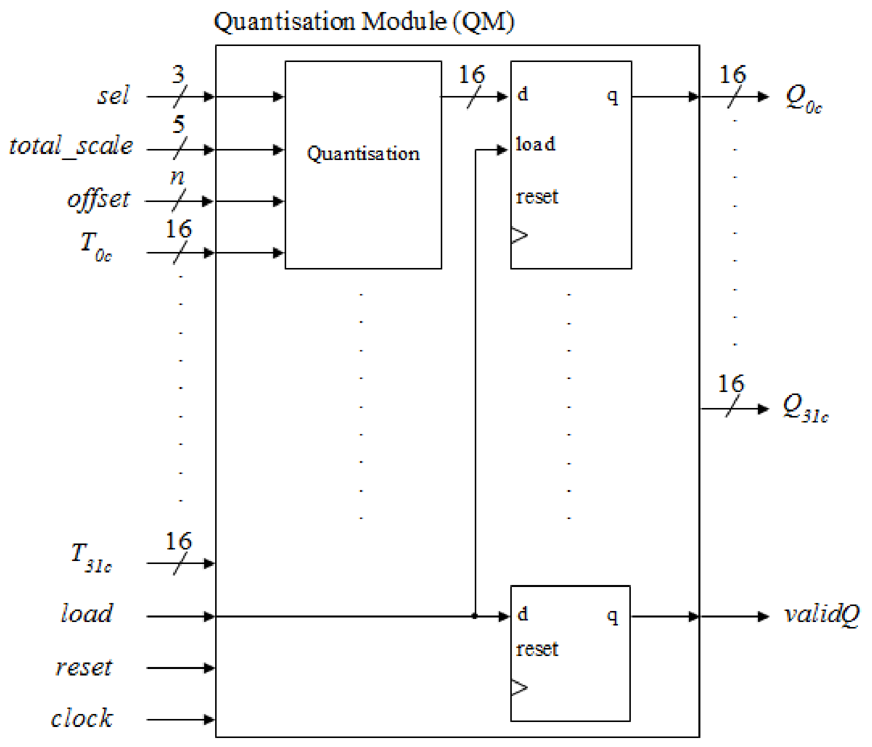


Fig. 6.9 Functional block diagram of quantisation module (QM)

Table 6.6 Quantiser value for HEVC and approximated quantisation (Q) modules

sel[2:0]	**HEVC**	**Q**
QP%6 = 0	26214	26112
QP%6 = 1	23302	23296
QP%6 = 2	20560	20560
QP%6 = 3	18396	18396
QP%6 = 4	16384	16384

QP%6 = 5	14564	14592
Others	0	0

6.3.2 Functional Verification

Similar to the TM designs, the QM hardware architecture designs were also described in IEEE-VHDL and routed to a Xilinx Virtex-6 xc6vl550t-2ff1760 FPGA device using Xilinx ISE 14.7 tool chain. Functional simulations were performed for both HEVC and Q designs using test benches and test vectors in the Xilinx ISim environment.

6.3.3 Results and Discussions

Table 6.7 summarises the resource utilisation of the developed QM designs. Although only three out of six quantiser values were approximated (Section 4.3), when implemented on the selected Xilinx Virtex-6 FPGA device, a saving of around 21% in the number of slices could be achieved. Additionally, as the critical path delay (cpd) in the original HEVC quantisation multiplier is a three-stage adder tree to perform the multiplications by 26,214, 23,302, and 14,564, this cpd is higher than in the approximated quantisation module and cannot operate at 200 MHz frequency. With an operating frequency of 166.67 MHz, QFHD videos could still potentially be processed but at a lower 50 fps as opposed to 60 fps achievable by the approximated quantisation module. The hardware efficiency of Q is also almost double (0.212 mega samples/sec/slice) from HEVC (0.122 mega samples/sec/slice).

Table 6.7 Resource utilisation of HEVC and approximated quantisation designs

Parameter	Quantisation	
	HEVC	Q
Slice registers (Savings %)	6,176	4,096 (33.67%)
Slice LUTs (Savings %)	14,032	10,560 (24.79%)
Slices (Savings %)	4,916	3,768 (21.17%)
Operating freq. (MHz)	166.67	200
Output rate (per cycle)	4/8/16/32	4/8/16/32
Throughput ($\times 10^9$ samples per second)	0.6/1.3/2.6/5.3	0.8/1.6/3.2/6.4
Supported resolution @ fps	QFHD @ 50	QFHD @ 60
Hardware efficiency[a] ($\times 10^6$ samples/sec/slice)	0.122	0.212

[a] Hardware efficiency = Min Throughput ($\times 10^6$ samples/second)/Slices

6.3.4 Conclusions

More than 20% hardware savings (in the number of Virtex-6 slices) could be yielded by the approximated quantisation (Q) hardware design when compared to using the original HEVC quantiser multipliers. Both designs were developed using the multiplier-free technique and MCM approach as applied in the transform designs. Having a cpd of a three-stage adder tree required in three of the HEVC quantisers (26,214, 23,302, and 14,564) results in a lower operating frequency (166.67 MHz) compared to Q (200 MHz). With a higher operating frequency, more QFHD frames can be processed (60 fps compared to 50 fps) and a greater hardware efficiency value could be obtained (0.212 mega samples/sec/slice compared to 0.122 mega samples/sec/slice) by the approximated quantisation design.

6.4 Hardware Architecture Designs for Approximated and Scaled Transform and Quantisation

This section briefly describes the combined transform and quantisation hardware architecture design developed in this work. For the approximated transform, only ST16 transform was considered and T16 was not performed as ST16 was the eventual objective of the complexity-reduced transform. Similar to the transform and quantisation designs described in the previous two sections, the transform and quantisation module (TQM) is designed as a hardware co-processor and the corresponding master processor and associated interface modules were not developed in this work.

6.4.1 Top-level Transform and Quantisation Module (TQM)

The transform and quantisation module (TQM) consists of the TM (Section 6.2) and QM (Section 6.3) (Fig. 6.10). The inputs to this module are the same as for the TM (residual signal (X_{rc}), transform size (*size*), *start*, *reset*, and *clock*) plus three parameter signals (*sel*, *total_scale*, *offset*) for the QM. The outputs of TQM module are the quantised levels (Q_{0c}, …, Q_{31c}) and *validQ* signal to indicate the validity of the outputs. Internally, the transform coefficients (T_{0c}, …, T_{31c}) from TM are input to QM, as well as the *valid* signal acting as the *load* signal for the internal registers in QM.

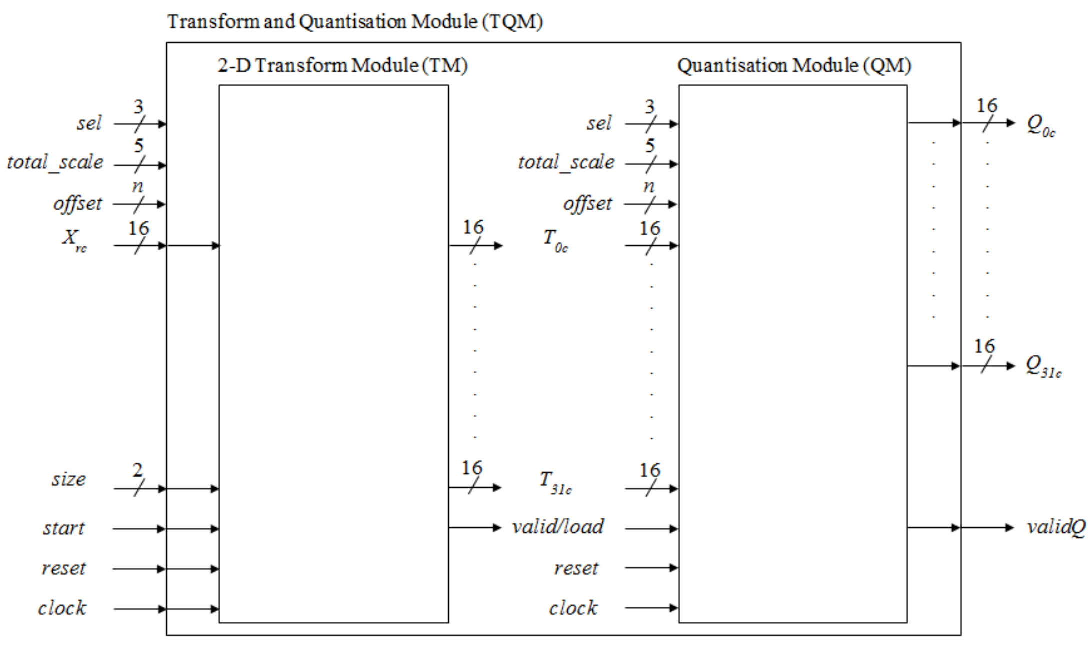


Fig. 6.10 Functional block diagram of transform and quantisation module (TQM)

6.4.2 Results and Discussions

Similar to the TM and QM designs, the TQM hardware architecture designs were also described in IEEE-VHDL and routed to a Xilinx Virtex-6 xc6vl550t-2ff1760 FPGA device using Xilinx ISE 14.7 tool chain. Table 6.8 summarises the resource utilisation of the developed TQM designs. When implemented on the selected Xilinx Virtex-6 FPGA device, a saving of more than 25% in the number of slices could be obtained by the combination of ST16 and Q designs relative to original HEVC transform and quantisation designs. Additionally, as the original HEVC quantisation can be operated at a frequency lower than 200 MHz, fewer QFHD frames (50 compared to 60) could be processed by the HEVC TQM and at a lower hardware efficiency (0.034 mega samples/sec/slice compared to 0.055 mega samples/sec/slice).

Table 6.8 Resource utilisation of HEVC and approximated transform and quantisation designs

Parameter	**Transform and Quantisation**	
	HEVC	**ST16 + Q**
Slice registers (Savings %)	52,348	42,478 (18.85%)
Slice LUTs (Savings %)	67,261	50,834 (24.42%)
Slices (Savings %)	19,783	14,667 (25.86%)
Operating freq. (MHz)	166.67	200
Output rate (per cycle)	4/8/16/32	4/8/16/32
Throughput ($\times 10^9$ samples per second)	0.6/1.3/2.6/5.3	0.8/1.6/3.2/6.4
Supported resolution @ fps	QFHD @ 50	QFHD @ 60
Hardware efficiency[a] ($\times 10^6$ samples/sec/slice)	0.034	0.055

[a] Hardware efficiency = Min Throughput ($\times 10^6$ samples/second)/Slices

6.4.3 Conclusions

More than 25% hardware savings (in the number of Virtex-6 slices) could possibly be attained by the approximated and scaled transform and quantisation ST16 + Q TQM over the original HEVC TQM. With a lower resource utilisation, the approximated TQM can still operate at a higher 200 MHz frequency and capable of processing QFHD @ 60 fps videos yielding a better hardware efficiency of 0.055 mega samples/sec/slice when compared with the HEVC TQM developed in this work (166.67 MHz capable of QFHD @ 50 fps and a hardware efficiency of 0.034 mega samples/sec/slice).

6.5 Summary

In this chapter, HEVC and approximated 2-D transform and quantisation hardware architecture designs developed in this work were described. Two approximated core transforms (T16 and ST16) were implemented on a Xilinx Virtex-6 FPGA device adopting the even-odd decomposition, multiplier-free, and MCM approaches, yielding savings of 16.9% and 20.8% in the number of slices over HEVC transform design. Both T16 and ST16 designs could as well be operated at a higher execution frequency (200 MHz) than (Kalali *et al.*, 2014) (150 MHz) and (Kalali, Mert and Hamzaoglu, 2016) (100 – 117 MHz), enabling these designs to encode more QFHD frames per second (60 fps) than (Kalali *et al.*, 2014) (48 fps) and the LU hardware in (Kalali, Mert and Hamzaoglu, 2016) (48 or 56 fps), and possessing slightly better hardware efficiencies (0.0687 and 0.0721 mega samples/sec/slice, respectively) than (Kalali *et al.*, 2014) (0.0529 or 0.0526 mega samples/sec/slice) and LU hardware in (Kalali, Mert and Hamzaoglu, 2016) (0.0397 or 0.0474 mega samples/sec/slice). However, the HU designs in (Kalali, Mert and Hamzaoglu, 2016) outperformed T16 and ST16 with twofold hardware efficiencies (0.1660 and 0.1397 mega samples/sec/slice) and capable of supporting more than 50 UHD frames per second.

One approximated quantisation (Q) design was also implemented and targeted to the same Xilinx Virtex-6 FPGA yielding more than 20% savings in the number of slices over a quantisation scheme suggested for HEVC in (Budagavi, Fuldseth and Bjøntegaard, 2014). Having a larger cpd of a three-stage adder tree required in three of the HEVC quantisers (26,214, 23,302, and 14,564) results in a lower operating frequency (166.67 MHz) compared to Q (200 MHz). With a higher operating frequency, more QFHD frames can be processed (60 fps compared to 50 fps) and a greater hardware efficiency value could be obtained (0.212 mega samples/sec/slice compared to 0.122 mega samples/sec/slice) by the approximated quantisation design over HEVC.

Finally, more than 25% hardware savings (in the number of Virtex-6 slices) could possibly be attained by combining the approximated and scaled transform and quantisation ST16 + Q TQM over the original HEVC TQM. Utilising fewer resources, the approximated TQM could also operate at a higher 200 MHz frequency and capable of processing QFHD @ 60 fps videos and yielding a better hardware efficiency of 0.055 mega samples/sec/slice when compared with the HEVC TQM developed in this work (166.67 MHz capable of QFHD @ 50 fps and a hardware efficiency of 0.034 mega samples/sec/slice).

In summary, the hardware implementations of the approximated transforms and quantisation presented in this chapter lay some support to the software-based coding performance results presented in Chapter 5 such that these transform and quantisation approximation schemes could be considered for a complexity-reduced HEVC encoder. Although some coding performance degradations were previously seen in terms of BD-rate (up to 2.1% on average), the hardware savings may outweigh these bitrate increments.

Chapter 7

Conclusions and Future Work

Abstract This is the final chapter of this thesis, summarising the work and results presented thus far and suggesting possible directions as a continuation of this study.

7.1 Conclusions

Two approximated transform matrices, T16 and ST16, and one approximated quantisation, Q, were presented in this thesis as alternatives to the original 2-D transform and quantisation of the most recent video coding standard, HEVC. These approximated transforms and quantisation were developed aiming to reduce the complexity of the respective operations in HEVC without too severely affecting the coding performance.

T16 was chosen among several approximated transform alternatives developed in this work due to its lowest orthogonality measure. These alternatives were aimed at a multiplier-free implementation using combinations of bit shifts and additions or subtractions, and derived by imposing three approximation criteria. The first criterion was that their matrix elements must be multiples of four, the second criterion being the maximum number of two bit shifts and one addition or subtraction for each element multiplication, and the final criterion was all multiplications are executable in one clock cycle of 5 ns or faster (i.e., operating frequency of 200 MHz or higher). ST16 is the down-scaled version of all elements of T16 by four, with necessary changes in the subsequent intermediate scaling operations in the 2-D transform operation. Q was developed using similar approaches, i.e., by approximating the original HEVC quantisation multipliers with a maximum of two additions or subtractions per one quantisation multiplier, also aimed at a multiplier-free implementation.

These approximated transforms and quantisation were evaluated using reference model software for HEVC, HM-13.0. The main experiments were conducted in Main profile of HEVC under RA and LB configurations, to simulate the entertainment and interactive application scenarios, respectively. The experiments were also conducted on test video sequences of HD quality or better. The experimental settings were subsets of the CTC established by JCT-VC, the experts committee responsible in developing the standard. Based on the conducted experiments, both T16 and ST16 provided similar average BD-rate differences of 1.7% in RA and 0.7% in LB configurations over HEVC. On the other hand, Q provided no significant difference against HEVC quantisation, with 0.0% and -0.1% average BD-rate differences in RA and LB, respectively. Finally, a combination of ST16 and Q in the encoder yielded on average BD-rate differences of 1.7% and 0.7% in RA and LB, respectively. These bitrate increments may not be viewed as small from video coding perspective, but are necessary penalties as the result of the approximations made in the transform and quantisation of HEVC.

Hardware architecture designs were then developed for T16, ST16, Q, and ST16 + Q in order to estimate potential resource savings against HEVC transform and quantisation algorithms. Methods used are MCM and multiplier-free implementation as previously mentioned, as well as even–odd decomposition for the transform algorithms, exploiting the embedded and symmetry properties inherited from the well-known DCT. When implemented on a Xilinx Virtex-6 FPGA device, savings of around 16.9%, 20.8%, 21.2%, and 25.9%, respectively, in the number of slices could be achieved when compared with HEVC transform and quantisation algorithms. The developed architecture designs could have a maximum operating frequency of more than 200 MHz, allowing them to support the encoding of QFHD @ 60 fps videos. When T16 and ST16 were compared with similar works in the literature, these algorithms have better hardware efficiency (0.0687 and 0.0721, respectively, in 10^6 sample rate per slice) than (Kalali *et al.*, 2014) (0.0529 or 0.0526 $\times 10^6$ sample rate per slice) and the LU designs in (Kalali, Mert and Hamzaoglu, 2016) (0.0397 or 0.0474 $\times 10^6$ sample rate per slice). Nevertheless, the HU designs in (Kalali, Mert and Hamzaoglu, 2016) are superior than T16 and ST16 with double hardware efficiencies (0.1660 and 0.1397 mega samples/sec/slice) and capable of sustaining UHD videos of more than 50 frames per second.

7.2 Future Work

The work presented in this thesis is anything but complete. The following weaknesses identified need to be addressed or improved before any algorithm presented in this work could actually be considered in practice. The quickest improvement to the work is by conducting the software-based experiments using the latest version of HEVC reference software. At the point of writing, the latest revision is HM-16.12 (JCT-VC, 2016), where this revision supports all three versions of HEVC, i.e., including 3D-HEVC, MV-HEVC, SHVC, and RExt extensions. It is important to see the effects of the approximated transforms and quantisation in these HEVC extensions. It would also be highly useful to conduct the experiments using more test video sequences of HD quality and beyond, as well as test suites suitable for the evaluation of range extension, scalable coding, multiview, and 3-D applications. A high-performance computer such as a supercomputer would be very useful in generating much faster encoding and decoding results, especially in these extended applications.

To provide more credentials to the work done, more quality metrics need as well be included on top of PSNR. The full versions of commercial video quality analysers such as sold by Elecard (Elecard, 2017) and Moscow State University (MSU, 2017) could be considered. The free versions of these analysers usually have limitations such as the maximum video resolution and number of frames that can be analysed.

Another important point is the validation of the architecture designs on a real hardware platform. This platform such as an FPGA device must have a high number of resources on its chip to be able to implement highly complex designs of video coding algorithms. A more useful realisation approach would be to synthesise the designs on CMOS fabrication technology software such as by Mentor Graphics (Mentor Graphics, 2017) and Synopsys (Synopsys, 2017), as eventually any digital circuits are intended to be implemented on a CMOS IC. The resource savings would also be more meaningful if analysed on a complete encoder system rather than only the involved processing blocks such as transformation and quantisation. The effects on power dissipation would further add value to the work conducted. It would also be useful to describe the relationship between hardware complexity (e.g., in a number of slices) and coding performance (e.g., BD-rate). This would require other alternative transform and quantisation algorithms be evaluated in HEVC reference software and designed for hardware implementations.

Most of these recommendations require a good source of research funding, as video and IC design technologies are highly-expensive know-hows.

References

Ahmed, A., Shahid, M. U. and Rehman, A. U. (2012) ‘N point DCT VLSI architecture for emerging HEVC standard’, *VLSI Design*, 2012, pp. 1–13. doi: 10.1155/2012/752024.

Ahmed, N., Natarajan, T. and Rao, K. R. (1974) ‘Discrete Cosine Transform’, *IEEE Transactions on Computers*, 1974(January), pp. 90–93.

Arayacheeppreecha, P., Pumrin, S. and Supmonchai, B. (2015) ‘Flexible Input Transform Architecture for HEVC Encoder on FPGA’, in *12th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*. Hua Hin, Thailand, 24-27 June 2015, pp. 1–6. doi: 10.1109/ECTICon.2015.7206947.

Bayer, F. M. and Cintra, R. J. (2012) ‘DCT-like transform for image compression requires 14 additions only’, *Electronics Letters*, 48(15), pp. 919–920. doi: 10.1049/el.2012.1148.

Bayer, F. M., Cintra, R. J., Edirisuriya, A. and Madanayake, A. (2012) ‘A digital hardware fast algorithm and FPGA-based prototype for a novel 16-point approximate DCT for image compression applications’, *Measurement Science and Technology*, 23(11), pp. 114–123. doi: 10.1088/0957-0233/23/11/114010.

Belghith, F., Loukil, H. and Masmoudi, N. (2013a) ‘Efficient Hardware Architecture of the Direct 2-D Transform for the HEVC Standard’, *World Academy of Science, Engineering and Technology*, 74, pp. 1290–1294.

Belghith, F., Loukil, H. and Masmoudi, N. (2013b) ‘Free Multiplication Integer Transformation For The HEVC Standard’, in *10th International Multi-Conference on Systems, Signals & Devices (SSD)*. Hammamet, Tunisia, 18-21 Mar. 2013, pp. 1–5. doi: 10.1109/SSD.2013.6564002.

Bjøntegaard, G. (2001) *Calculation of average PSNR differences between RD-curves*. VCEG-M33. ITU-T SG 16/Q 6. Austin, TX, USA, pp. 1-4.

Bjøntegaard, G. (2008) *Improvements of the BD-PSNR model*. VCEG-AI11. ITU-T SG 16/Q 6. Berlin, Germany, pp 1-2.

Bolaños-Jojoa, J. D. and Velasco-Medina, J. (2015) ‘Efficient Hardware Design of N-point 1D-DCT for HEVC’, in *20th Symposium on Signal Processing, Images and Computer Vision (STSIVA)*. Bogota, Colombia, 2-4 Sept. 2015, pp. 1–6. doi: 10.1109/STSIVA.2015.7330449.

Bossen, F. (2013) *Common test conditions and software reference configurations*. JCTVC-L1100. JCT-VC. Geneva, Switzerland, pp. 1-4.

Bouguezel, S., Ahmad, M. O. and Swamy, M. N. S. (2010) ‘A Novel Transform for Image Compression’, in *53rd IEEE International Midwest Symposium on Circuits and Systems*. Seattle, WA, USA, 1-4 Aug. 2010, pp. 509–512. doi: 10.1109/MWSCAS.2010.5548745.

Budagavi, M., Fuldseth, A. and Bjøntegaard, G. (2014) ‘Chapter 6 HEVC Transform and Quantization', in Sze, V., Budagavi, M., and Sullivan, G. J. (eds) *High Efficiency Video Coding (HEVC): Algorithms and Architectures*. Cham: Springer International Publishing, pp. 141–169. doi: 10.1007/978-3-319-06895-4_6.

Budagavi, M., Fuldseth, A., Bjøntegaard, G., Sze, V. and Sadafale, M. (2013) 'Core Transform Design in the High Efficiency Video Coding (HEVC) Standard', *IEEE Journal on Selected Topics in Signal Processing*, 7(6), pp. 1029–1041. doi: 10.1109/JSTSP.2013.2270429.

Budagavi, M. and Sze, V. (2012) 'Unified Forward + Inverse Transform Architecture for Hevc Inverse Transforms', in *19th IEEE International Conference on Image Processing (ICIP)*. Orlando, FL, USA, 30 Sept.-3 Oct. 2012, pp. 209–212. doi: 10.1109/ICIP.2012.6466832.

Cham, W. K. and Chan, Y. T. (1991) 'An Order-16 Integer Cosine Transform', *IEEE Transactions on Signal Processing*, 39(5), pp. 1205–1208. doi: 10.1109/78.80974.

Chang, C.-W., Hsu, H.-F., Fan, C.-P., Wu, C.-B. and Chang, R. C.-H. (2016) 'A Fast Algorithm-Based Cost-Effective and Hardware-Efficient Unified Architecture Design of 4 × 4, 8 × 8, 16 × 16, and 32 × 32 Inverse Core Transforms for HEVC', *Journal of Signal Processing Systems*, 82(1), pp. 69–89. doi: 10.1007/s11265-015-0982-8.

Chen, M., Zhang, Y. and Lu, C. (2017) 'Efficient architecture of variable size HEVC 2D-DCT for FPGA platforms', *AEUE - International Journal of Electronics and Communications*. Elsevier GmbH, 73, pp. 1–8. doi: 10.1016/j.aeue.2016.12.024.

Chen, W.-H., Smith, C. and Fralick, S. (1977) 'A Fast Computational Algorithm for the Discrete Cosine Transform', *IEEE Transactions on Communications*, 25(9), pp. 1004–1009. doi: 10.1109/TCOM.1977.1093941.

Cintra, R. J., Bayer, F. M. and Tablada, C. J. (2014) 'Low-complexity 8-point DCT approximations based on integer functions', *Signal Processing*. Elsevier, 99, pp. 201–214. doi: 10.1016/j.sigpro.2013.12.027.

Cisco (2016) *Cisco Visual Networking Index : Global Mobile Data Traffic Forecast Update , 2015 – 2020*. Available at: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html (Accessed: 11 April 2016).

Conceição, R., Souza, J. C., Jeske, R., Porto, M., Mattos, J. and Agostini, L. (2013) 'Hardware Design for the 32x32 IDCT of the HEVC Video Coding Standard', in *26th Symposium on Integrated Circuits and Systems Design (SBCCI)*. Curitiba, Brazil, 2-6 Sept. 2013, pp. 1–6. doi: 10.1109/SBCCI.2013.6644881.

Coutinho, A., Cintra, R. J., Bayer, F. M., Kulasekera, S. and Madanayake, A. (2016) 'A multiplierless pruned DCT-like transformation for image and video compression that requires ten additions only', *Journal of Real-Time Image Processing*, 12, pp. 247–255. doi: 10.1007/s11554-015-0492-8.

Darji, A. D. and Makwana, R. P. (2015) 'High-Performance Multiplierless DCT architecture for HEVC', in *19th International Symposium on VLSI Design and Test*.

Ahmedabad, India, 26-29 June 2015, pp. 1–5. doi: 10.1109/ISVDAT.2015.7208051.

Dias, T., Roma, N. and Sousa, L. (2013) ‘High Performance Multi-Standard Architecture for DCT Computation in H.264 / AVC High Profile and HEVC Codecs’, in *Conference on Design and Architectures for Signal and Image Processing*. Cagliari, Italy, 8-10 Oct. 2013, pp. 14–21.

Dias, T., Roma, N. and Sousa, L. (2014) ‘Unified transform architecture for AVC, AVS, VC-1 and HEVC high-performance codecs’, *EURASIP Journal on Advances in Signal Processing*, 2014(1), pp. 108–122. doi: 10.1186/1687-6180-2014-108.

Dias, T., Roma, N. and Sousa, L. (2015) ‘High performance IP core for HEVC quantization’, in *IEEE International Symposium on Circuits and Systems (ISCAS)*. Lisbon, Portugal, 24-27 May 2015, pp. 2828–2831. doi: 10.1109/ISCAS.2015.7169275.

Dong, J., Ngan, K. N., Fong, C.-K. and Cham, W.-K. (2009) ‘2-D order-16 integer transforms for HD video coding’, *IEEE Transactions on Circuits and Systems for Video Technology*, 19(10), pp. 1462–1474. doi: 10.1109/TCSVT.2009.2026792.

Elecard (2017) *Elecard Video Compression Guru*. Available at: http://www.elecard.com/en/index.html (Accessed: 14 March 2017).

Grois, D., Marpe, D., Mulayoff, A., Itzhaky, B. and Hadar, O. (2013) ‘Performance comparison of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC encoders’, in *Picture Coding Symposium (PCS)*. San Jose, CA, USA, 8-11 Dec. 2013, pp. 394–397. doi: 10.1109/PCS.2013.6737766.

Gweon, R. and Lee, Y. L. (2012) ‘N-level quantization in HEVC’, in *IEEE international Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. Seoul, South Korea, 27-29 June 2012, pp. 1–5. doi: 10.1109/BMSB.2012.6264318.

Haggag, M. N., El-Sharkawy, M. and Fahmy, G. (2010) ‘Efficient Fast Multiplication-Free Integer Transformation for the 2-D DCT H.265 Standard’, in *IEEE 17th International Conference on Image Processing (ICIP)*. Hong Kong, 26-29 Sept. 2010, pp. 3769–3772. doi: 10.1109/ICIP.2010.5653484.

Haggag, M. N., El-Sharkawy, M., Fahmy, G. and Rizkalla, M. (2010) ‘Efficient Fast Multiplication Free Integer Transformation for the 1-D DCT of the H.265 Standard’, *Journal of Software Engineering & Applications*, 3, pp. 784–795. doi: 10.4236/jsea.2010.38091.

Hanhart, P. and Ebrahimi, T. (2014) ‘Calculation of average coding efficiency based on subjective quality scores’, *Journal of Visual Communication and Image Representation*. Elsevier Inc., 25(3), pp. 555–564. doi: 10.1016/j.jvcir.2013.11.008.

Hanhart, P., Rerabek, M., De Simone, F. and Ebrahimi, T. (2012) ‘Subjective quality evaluation of the upcoming HEVC video compression standard’, *Proc. SPIE Appl. Digital Image Process. XXXV*, 8499, pp. 1–13. doi: 10.1117/12.946036.

Hani, M. K. (2011) *Starter’s Guide to Digital Systems VHDL & Verilog Design*. 2nd edn. (rev. edn. 2.4). Malaysia: Pearson Prentice Hall.

HHI (2016) *Video Coding & Analytics H.265/HEVC*. Available at: http://www.hhi.fraunhofer.de/en/departments/video-coding-analytics/products-technologies/coding-communication/h265hevc.html (Accessed: 23 June 2016).

Huynh-Thu, Q. and Ghanbari, M. (2012) 'The accuracy of PSNR in predicting video quality for different video scenes and frame rates', *Telecommunication Systems*, (49), pp. 35–48. doi: 10.1007/s11235-010-9351-x.

ISO (1993) *Information technology -- Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s -- Part 2: Video*. ISO/IEC 11172-2:1993. ISO/IEC JTC 1/SC 29.

ISO (2004) *Information technology -- Coding of audio-visual objects -- Part 2: Visual*. ISO/IEC 14496-2:2004. ISO/IEC JTC 1/SC 29.

ISO (2013a) *Information technology -- Generic coding of moving pictures and associated audio information -- Part 2: Video*. ISO/IEC 13818-2:2013. ISO/IEC JTC 1/SC 29.

ISO (2013b) *Information technology -- High efficiency coding and media delivery in heterogeneous environments -- Part 2: High efficiency video coding*. ISO/IEC 23008-2:2013. ISO/IEC JTC 1/SC 29.

ISO (2014) *Information technology -- Coding of audio-visual objects -- Part 10: Advanced Video Coding*. ISO/IEC 14496-10:2014. ISO/IEC JTC 1/SC 29.

ITU (1993a) *Codecs for videoconferencing using primary digital group transmission*. Recommendation ITU-T H.120. ITU-T SG 16.

ITU (1993b) *Video codec for audiovisual services at p x 64 kbit/s*. Recommendation ITU-T H.261. ITU-T SG 16.

ITU (2005) *Video coding for low bit rate communication*. Recommendation ITU-T H.263. ITU-T SG 16.

ITU (2011) *Studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios*. Recommendation ITU-R BT.601. ITU-R.

ITU (2012) *Information technology - Generic coding of moving pictures and associated audio information: Video*. Amd. 4. Recommendation ITU-T H.262. ITU-T SG 16.

ITU (2013) *High Efficiency Video Coding*. Recommendation ITU-T H.265. ITU-T SG 16.

ITU (2014) *Advanced video coding for generic audiovisual services*. 9th edn. Recommendation ITU-T H.264. ITU-T SG 16.

Jarboe, G. (2015) *Vidcon 2015 Haul: Trends, Strategic Insights, Critical Data, and Tactical Advice*. Available at: http://tubularinsights.com/vidcon-2015-strategic-insights-tactical-advice/ (Accessed: 19 April 2016).

JCT-VC (2014) *HEVC Test Model HM-13.0*. Available at: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-13.0/.

JCT-VC (2016) *HEVC Test Model HM-16.12*. Available at: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/trunk/.

Jridi, M. and Meher, P. (2016) ‘A Scalable Approximate DCT Architectures for Efficient HEVC Compliant Video Coding’, *IEEE Transactions on Circuits and Systems for Video Technology*, 8215(c), pp. 1–10. doi: 10.1109/TCSVT.2016.2556578.

Kalali, E., Mert, A. C. and Hamzaoglu, I. (2016) ‘A Computation and Energy Reduction Technique for HEVC Discrete Cosine Transform’, *IEEE Transactions on Consumer Electronics*, 62(2), pp. 166–174. doi: 10.1109/TCE.2016.7514716.

Kalali, E., Ozcan, E., Yalcinkaya, O. M. and Hamzaoglu, I. (2014) ‘A low energy HEVC Inverse Transform hardware’, *IEEE Transactions on Consumer Electronics*, 60(4), pp. 754–761. doi: 10.1109/ICCE-Berlin.2013.6698021.

Li, S., Ma, L. and Ngan, K. N. (2011) ‘Video quality assessment by decoupling additive impairments and detail losses’, in *3rd International Workshop on Quality of Multimedia Experience, QoMEX 2011*. Mechelen, Belgium, 7-9 Sept. 2011, pp. 90–95. doi: 10.1109/QoMEX.2011.6065719.

Martuza, M. and Wahid, K. A. (2012a) ‘A cost effective implementation of 8×8 transform of HEVC from H.264/AVC’, in *25th IEEE Canadian Conference on Electrical & Computer Engineering (CCECE)*. Montreal, Canada, 29 April-2 May 2012, pp. 1–4. doi: 10.1109/CCECE.2012.6334911.

Martuza, M. and Wahid, K. A. (2012b) ‘Low cost design of a hybrid architecture of integer inverse DCT for H.264, VC-1, AVS, and HEVC’, *VLSI Design*, 2012, pp. 1–10. doi: 10.1155/2012/242989.

Martuza, M. and Wahid, K. A. (2015) ‘Implementation of a cost-shared transform architecture for multiple video codecs’, *Journal of Real-Time Image Processing*, 10(1), pp. 151–162. doi: 10.1007/s11554-012-0266-5.

Meher, P. K., Park, S. Y., Mohanty, B. K., Lim, K. S. and Yeo, C. (2014) ‘Efficient integer DCT architectures for HEVC’, *IEEE Transactions on Circuits and Systems for Video Technology*, 24(1), pp. 168–178. doi: 10.1109/TCSVT.2013.2276862.

Mentor Graphics (2017) *IC Design*. Available at: https://www.mentor.com/products/ic_nanometer_design/ (Accessed: 14 March 2017).

MSU (2017) *MSU Video Quality Measurement tools*. Available at: http://www.compression.ru/video/quality_measure/index_en.html (Accessed: 14 March 2017).

Nam, J., Sim, D. and Bajić, I. V (2012) ‘HEVC-based Adaptive Quantization for Screen Content Videos’, in *IEEE international Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. Seoul, South Korea, 27-29 June 2012, pp. 1–4. doi: 10.1109/BMSB.2012.6264265.

Ohm, J., Sullivan, G. J., Schwarz, H., Tan, T. K. and Wiegand, T. (2012) ‘Comparison of the Coding Efficiency of Video Coding Standards — Including High

Efficiency Video Coding (HEVC)’, *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12), pp. 1669–1684. doi: 10.1109/TCSVT.2012.2221192.

Park, J. S., Nam, W. J., Han, S. M. and Lee, S. (2012) ‘2-D large inverse transform (16x16, 32x32) for HEVC (High Efficiency Video Coding)’, *Journal of Semiconductor Technology and Science*, 12(2), pp. 203–211. doi: 10.5573/JSTS.2012.12.2.203.

Pastuszak, G. (2014) ‘FPGA Architectures of the Quantization and the Dequantization for Video Encoders’, in *17th International Symposium on Design and Diagnostics of Electronic Circuits & Systems*. Warsaw, Poland, 23-25 April 2014, pp. 290–293. doi: 10.1109/DDECS.2014.6868812.

Raguraman, M. T. and Saravanan, D. S. (2016) ‘FPGA Implementation of Approximate 2D Discrete Cosine Transforms’, *Circuits and Systems*, 7, pp. 434–445. doi: 10.4236/cs.2016.74037.

Richardson, I. E. (2012) *The H.264 Advanced Video Compression Standard*. 2nd edn. Croydon: John Wiley & Sons, Ltd.

Robertson, M. R. (2015) *500 Hours of Video Uploaded to YouTube Every Minute [Forecast]*. Available at: http://tubularinsights.com/hours-minute-uploaded-youtube/ (Accessed: 19 April 2016).

Saxena, A. and Fernandes, F. C. (2013) ‘DCT/DST-based transform coding for intra prediction in image/video coding’, *IEEE Transactions on Image Processing*, 22(10), pp. 3974–3981. doi: 10.1109/TIP.2013.2265882.

Shen, S., Shen, W., Fan, Y. and Zeng, X. (2012) ‘A unified 4/8/16/32-point integer IDCT architecture for multiple video coding standards’, in *IEEE International Conference on Multimedia and Expo (ICME)*. Melbourne, Australia, 9-13 July 2012, pp. 788–793. doi: 10.1109/ICME.2012.7.

da Silveira, T. L. T., Oliveira, R. S., Bayer, F. M., Cintra, R. J. and Madanayake, A. (2017) ‘Multiplierless 16-point DCT approximation for low-complexity image and video coding’, *Signal, Image and Video Processing*. Springer London, 11(2), pp. 227–233. doi: 10.1007/s11760-016-0923-4.

Solsman, J. E. (2014) *Where do the most people go for TV online ? YouTube*. Available at: https://www.cnet.com/news/where-do-the-most-people-go-for-tv-online-youtube/ (Accessed: 18 April 2016).

Stankowski, J., Korzeniewski, C., Domanski, M. and Grajek, T. (2015) ‘Rate-distortion optimized quantization in HEVC: Performance limitations’, in *Picture Coding Symposium (PCS)*. Cairns, QLD, Australia, 31 May-3 June 2015, pp. 85–89. doi: 10.1109/PCS.2015.7170052.

Statista (2016) *Hours of video uploaded to YouTube every minute as of July 2015, The Statistics Portal*. Available at: http://www.statista.com/statistics/259477/hours-of-video-uploaded-to-youtube-every-minute/ (Accessed: 19 April 2016).

Sullivan, G. J. (2014) ‘Chapter 1 Introduction’, in Sze, V., Budagavi, M., and Sullivan, G. J. (eds) *High Efficiency Video Coding (HEVC): Algorithms and*

Architectures. Cham: Springer International Publishing, pp. 1–12. doi: 10.1007/978-3-319-06895-4_1.

Sullivan, G. J., Ohm, J., Han, W. and Wiegand, T. (2012) ‘Overview of the High Efficiency Video Coding’, *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12), pp. 1649–1668. doi: 10.1109/TCSVT.2012.2221191.

Sun, L., Au, O. C., Li, J., Zou, R. and Dai, W. (2012) ‘Hardware Oriented Re-design and Matrix Approximation Analysis for Transform in High Efficiency Video Coding (HEVC)’, in *IEEE Signal Info. Process. Assoc. Annual Summit Conf. (APSIPA ASC),*. Hollywood, CA, USA, 3-6 Dec. 2012, pp. 1–5.

Synopsys (2017) *Tools*. Available at: http://www.synopsys.com/Tools/Pages/default.aspx (Accessed: 14 March 2017).

Tabatabai, A., Suzuki, T., Hanhart, P., Korshunov, P., Ebrahimi, T., Horowitz, M., Kossentini, F. and Tmar, H. (2014) ‘Chapter 9 Compression Performance Analysis in HEVC’, in Sze, V., Budagavi, M., and Sullivan, G. J. (eds) *High Efficiency Video Coding (HEVC): Algorithms and Architectures*. Cham: Springer International Publishing, pp. 275–302. doi: 10.1007/978-3-319-06895-4_9.

Tan, T. K., Weerakkody, R., Mrak, M., Ramzan, N., Baroncini, V., Ohm, J. and Sullivan, G. J. (2016) ‘Video Quality Evaluation Methodology and Verification Testing of HEVC Compression Performance’, *IEEE Transactions on Circuits and Systems for Video Technology,* 26(1), pp. 76–90. doi: 10.1109/TCSVT.2015.2477916.

Vanne, J., Viitanen, M., Hamalainen, T. D. and Hallapuro, A. (2012) ‘Comparative rate-distortion-complexity analysis of HEVC and AVC video codecs’, *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12), pp. 1885–1898. doi: 10.1109/TCSVT.2012.2223013.

Video Clarity (2016) *Understanding MOS, JND and PSNR*. Available at: http://videoclarity.com/wpunderstandingjnddmospsnr/ (Accessed: 23 June 2016).

Wang, Q., Ji, X., Sun, M. T., Sullivan, G. J., Li, J. and Dai, Q. (2013) ‘Complexity reduction and performance improvement for geometry partitioning in video coding’, *IEEE Transactions on Circuits and Systems for Video Technology*, 23(2), pp. 338–352. doi: 10.1109/TCSVT.2012.2203743.

Wang, Z., Bovik, A. C., Sheikh, H. R. and Simoncelli, E. P. (2004) ‘Image quality assessment: From error visibility to structural similarity’, *IEEE Transactions on Image Processing*, 13(4), pp. 600–612. doi: 10.1109/TIP.2003.819861.

Weerakkody, R., Mrak, M., Baroncini, V., Ohm, J.-R., Tan, T. K. and Sullivan, G. J. (2014) ‘Verification Testing of HEVC Compression Performance for UHD Video’, in *GlobalSIP: Perception Inspired Multimedia Signal Processing Techniques*. Atlanta, GA, USA, 3-5 Dec. 2014, pp. 1083–1087. doi: 10.1109/GlobalSIP.2014.7032288.

Xilinx (2015) *Virtex-6 Family Overview*. DS150 (v2.5), August 20, 2015. Available at: https://www.xilinx.com/support/documentation/data_sheets/ds150.pdf.

Zeng, K., Rehman, A., Wang, J. and Wang, Z. (2013) ‘From H.264 to HEVC : coding gain predicted by objective video quality assessment models’, in *Seventh International Workshop on Video Processing and Quality Metrics for Consumer Electronics (VPQM2013)*. Scottsdale, AZ, USA, 30 Jan.-1 Feb. 2013, pp. 1–6.

Zhao, W. and Onoye, T. (2012) *A High-Performance Multiplierless Hardware Architecture of the Transform Applied to H.265/HEVC Emerging Video Coding Standard, IEICE*. Available at: http://ci.nii.ac.jp/naid/110009455690/en/.

Appendix A

R-D Curves of HEVC and Approximated Transforms

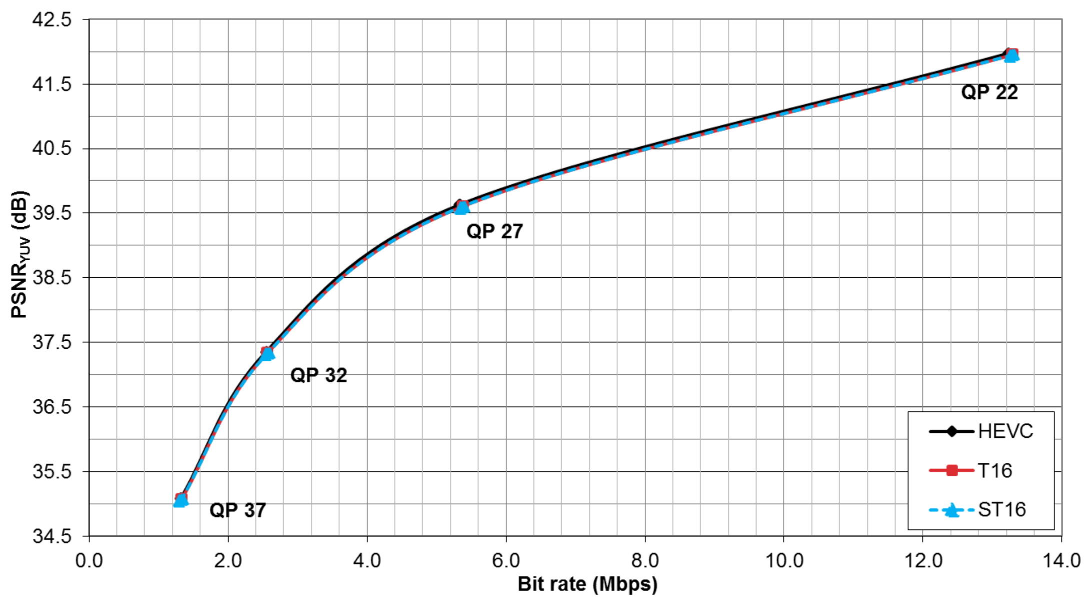


Fig. A.1 R-D$_{PSNR}$ curves of *A1 – Traffic* sequence using original HEVC and approximated transform matrices, T16 and ST16, under RA configuration in Main profile

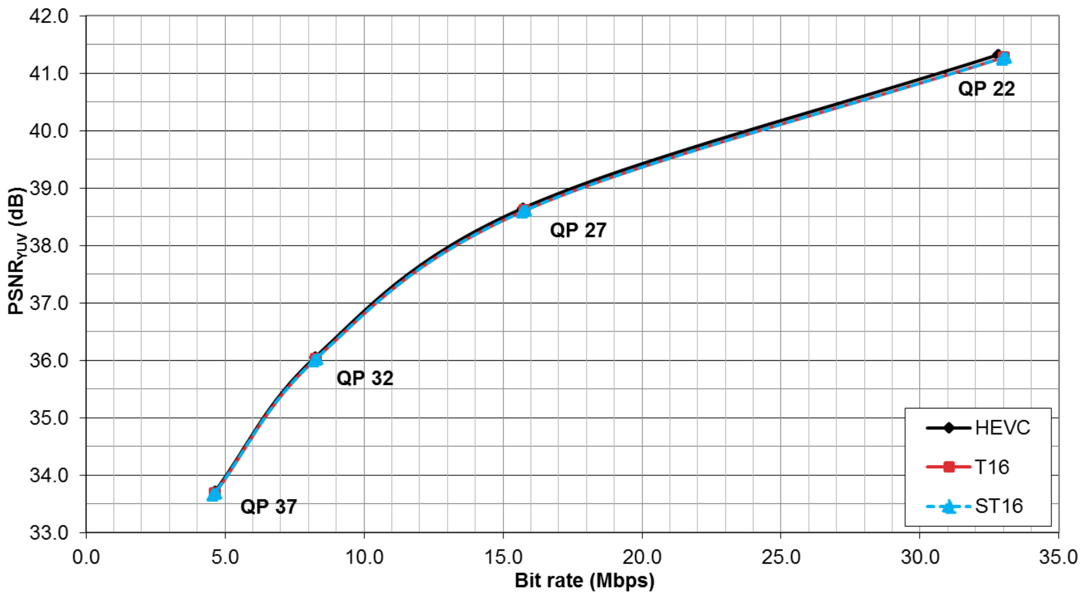


Fig. A.2 R-D$_{PSNR}$ curves of *A2 – PeopleOnStreet* sequence using original HEVC and approximated transform matrices, T16 and ST16, under RA configuration in Main profile

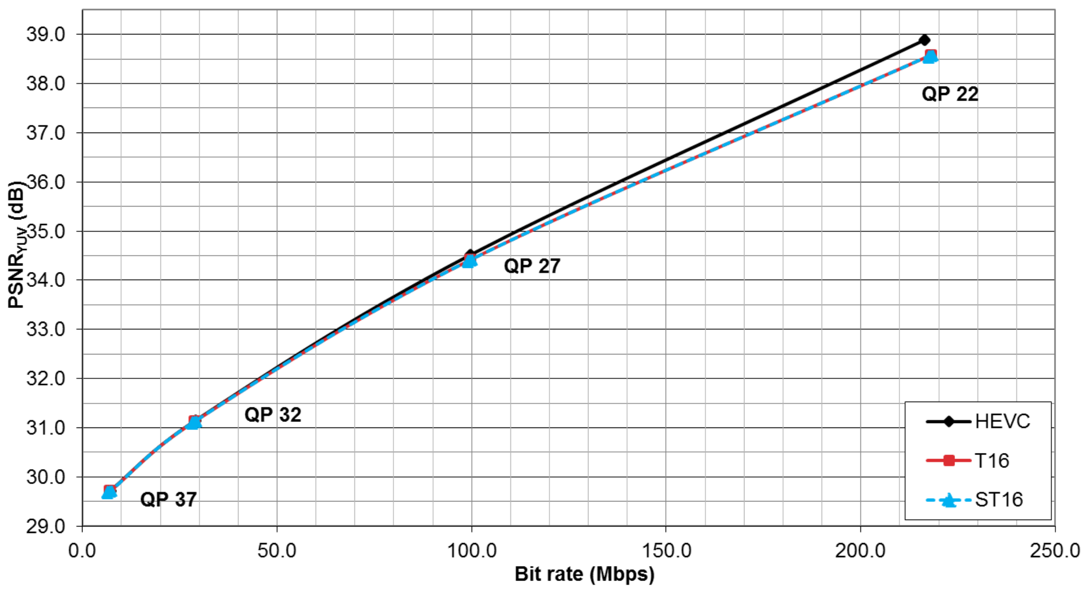


Fig. A.3 R-D$_{PSNR}$ curves of *A3 – Nebuta* sequence using original HEVC and approximated transform matrices, T16 and ST16, under RA configuration in Main profile

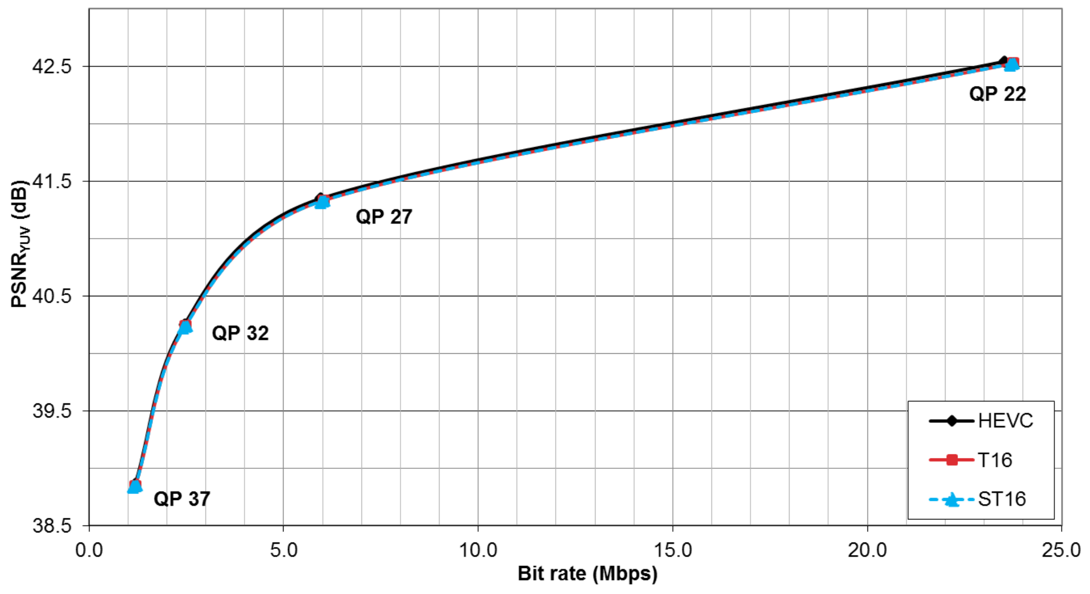
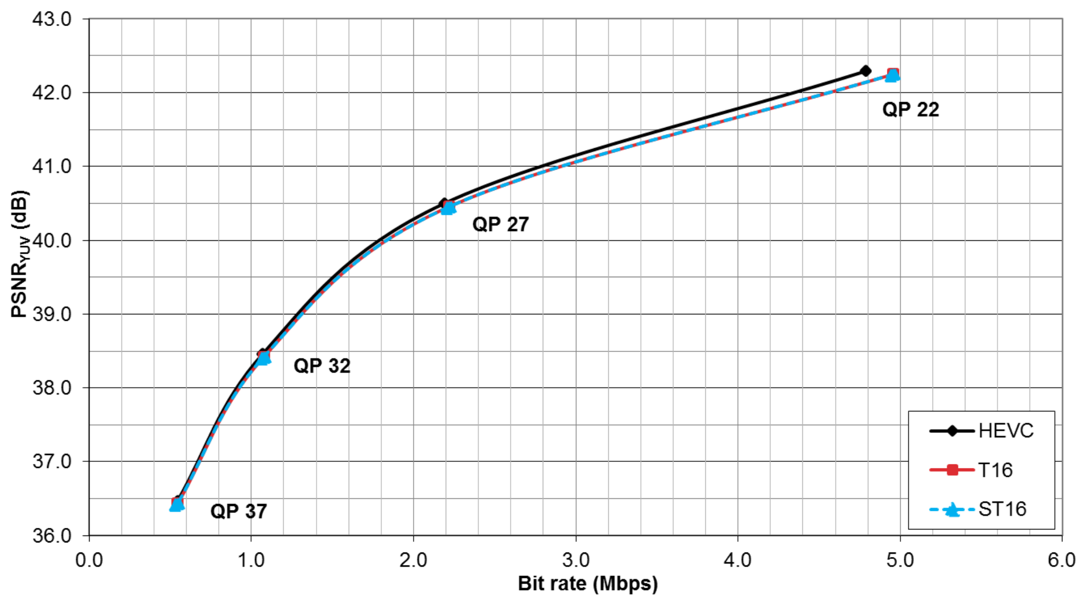
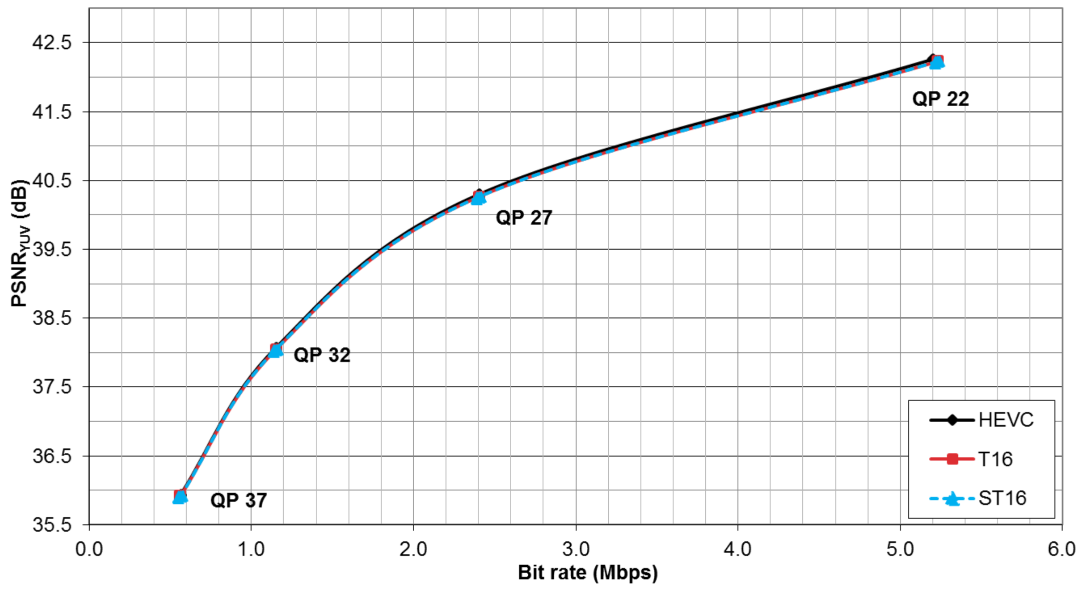


Fig. A.4 R-D$_{PSNR}$ curves of *A4* – *SteamLocomotive* sequence using original HEVC and approximated transform matrices, T16 and ST16, under RA configuration in Main profile

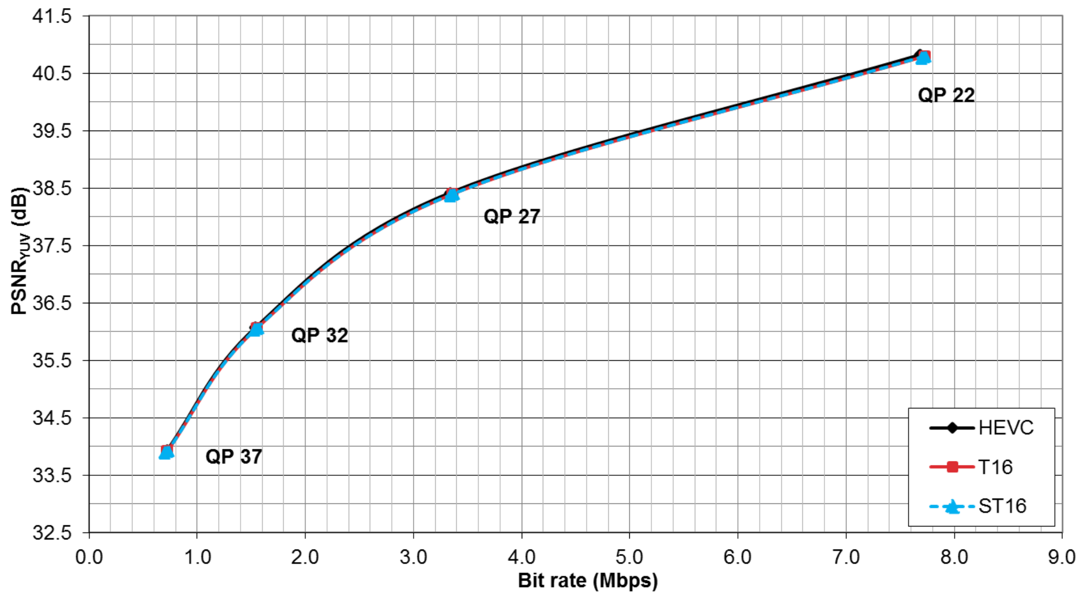


(a)

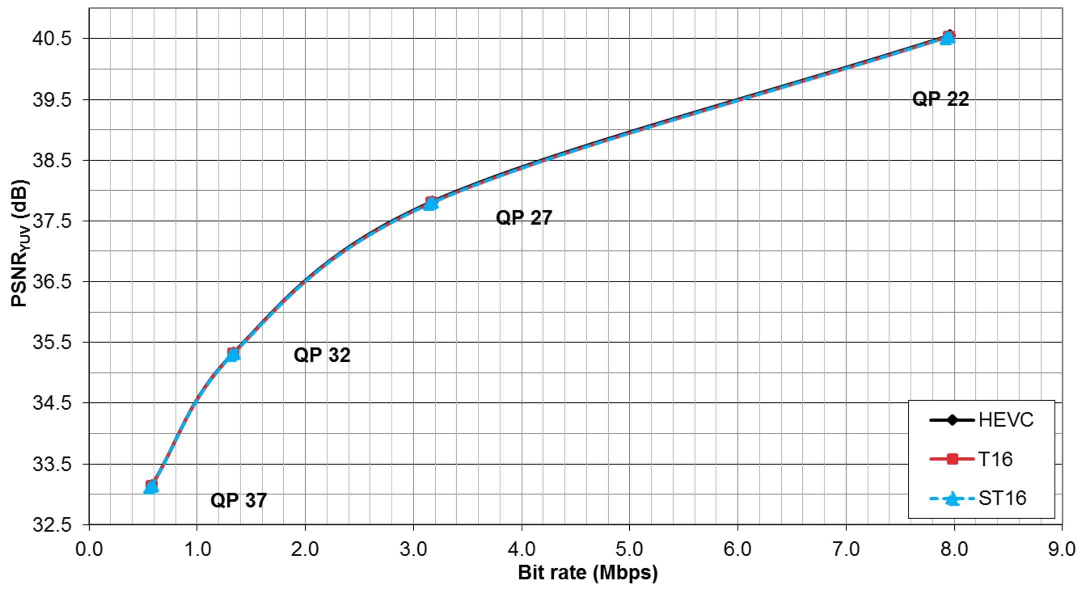


(b)

Fig. A.5 R-D$_{PSNR}$ curves of *B1 – Kimono* sequence using original HEVC and approximated transform matrices, T16 and ST16, under (a) RA and (b) LB configurations in Main profile

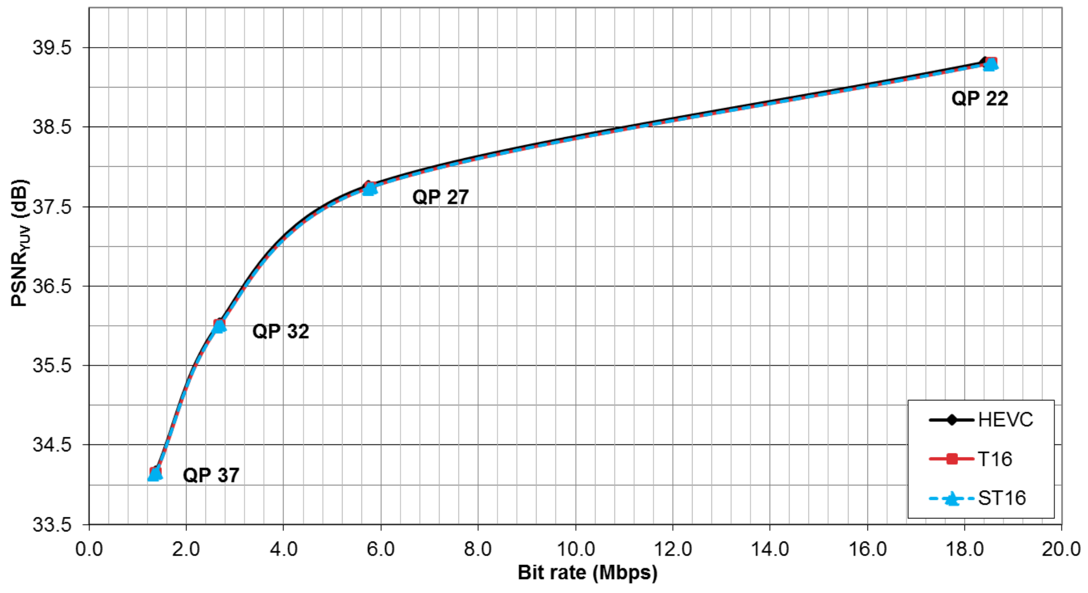


(a)

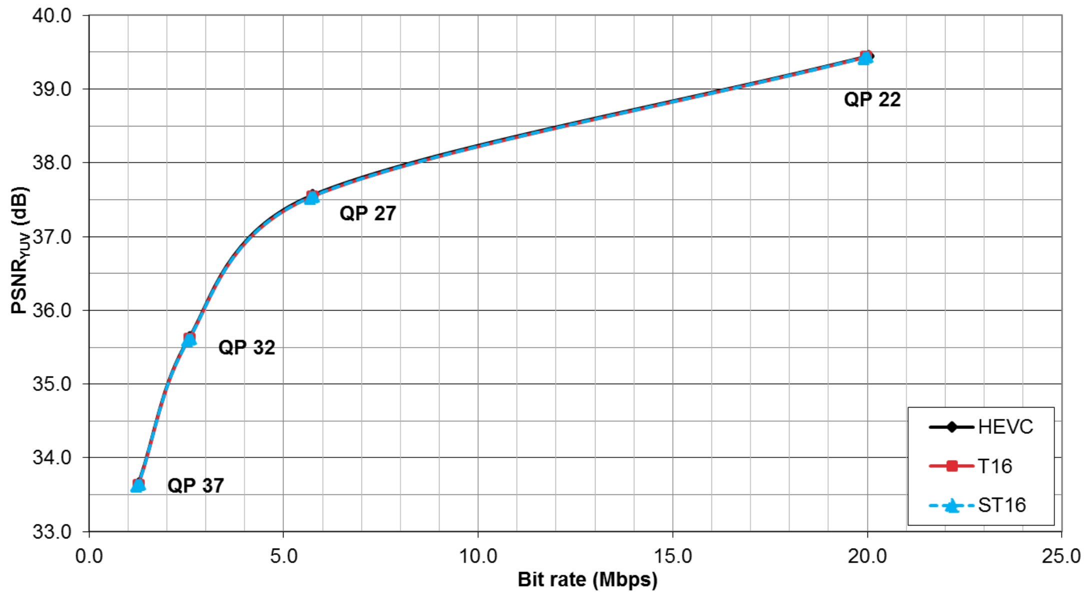


(b)

Fig. A.6 R-D$_{\text{PSNR}}$ curves of *B2 – ParkScene* sequence using original HEVC and approximated transform matrices, T16 and ST16, under (a) RA and (b) LB configurations in Main profile

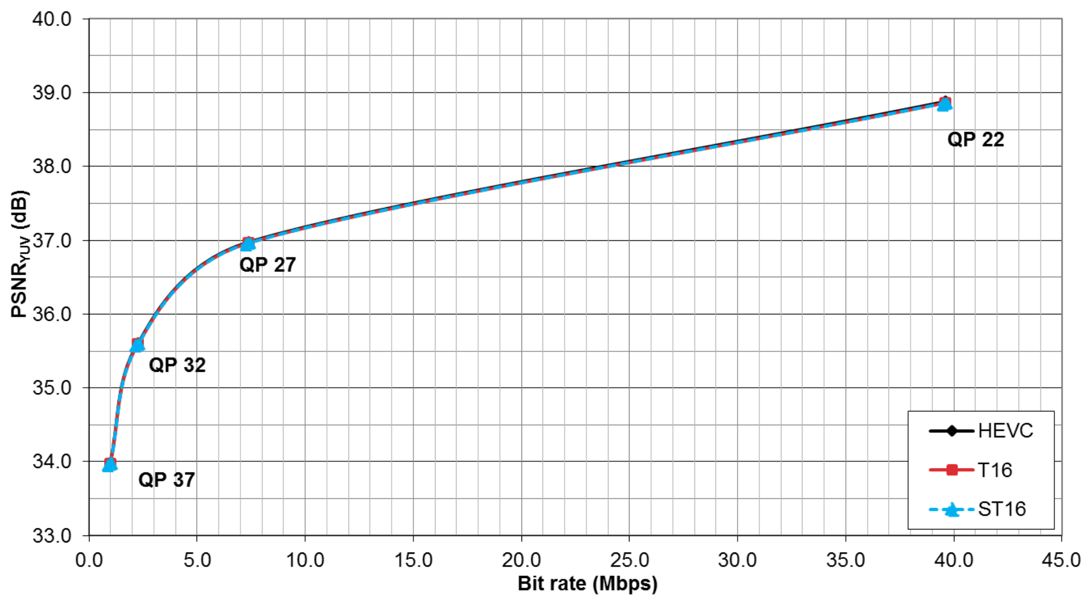


(a)

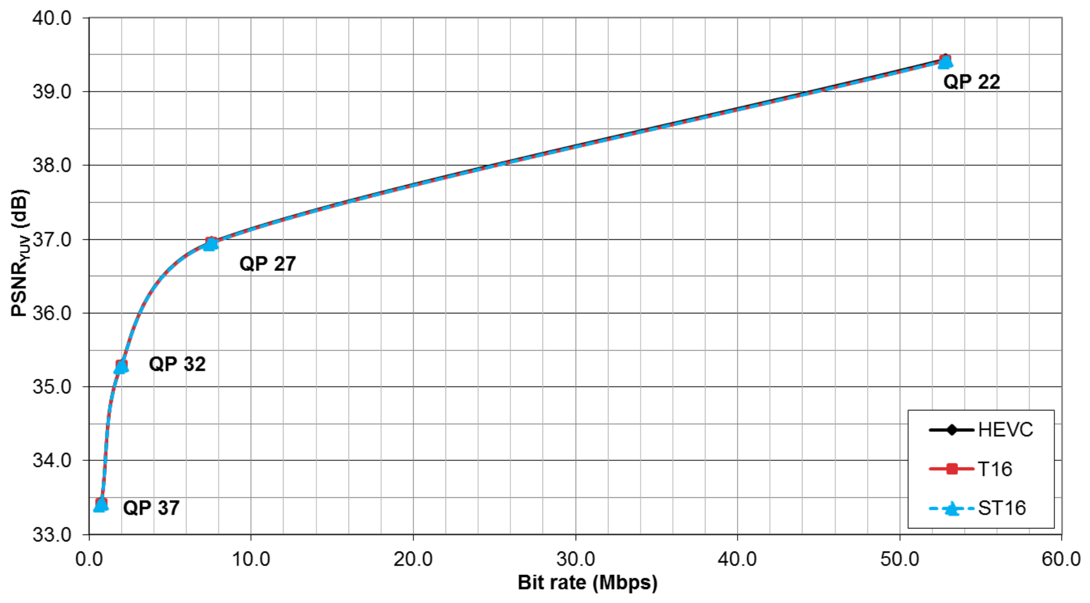


(b)

Fig. A.7 R-D$_{\text{PSNR}}$ curves of *B3 – Cactus* sequence using original HEVC and approximated transform matrices, T16 and ST16, under (a) RA and (b) LB configurations in Main profile



(a)



(b)

Fig. A.8 R-D$_{PSNR}$ curves of *B5 – BQTerrace* sequence using original HEVC and approximated transform matrices, T16 and ST16, under (a) RA and (b) LB configurations in Main profile

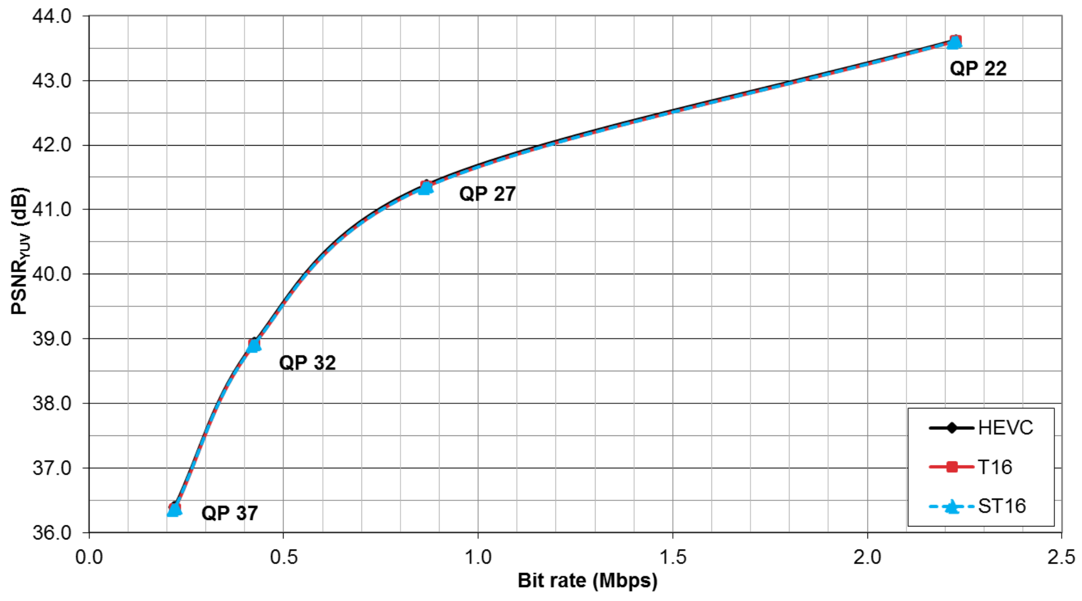


Fig. A.9 R-D$_{\text{PSNR}}$ curves of *E1 – FourPeople* sequence using original HEVC and approximated transform matrices, T16 and ST16, under LB configuration in Main profile

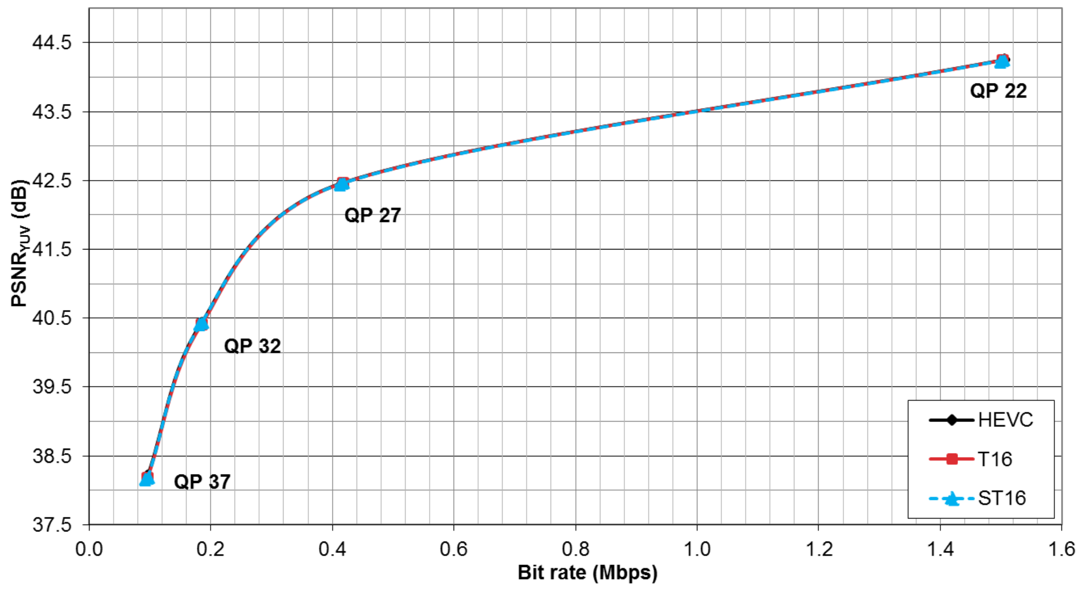


Fig. A.10 R-D$_{\text{PSNR}}$ curves of *E2 – Johnny* sequence using original HEVC and approximated transform matrices, T16 and ST16, under LB configuration in Main profile

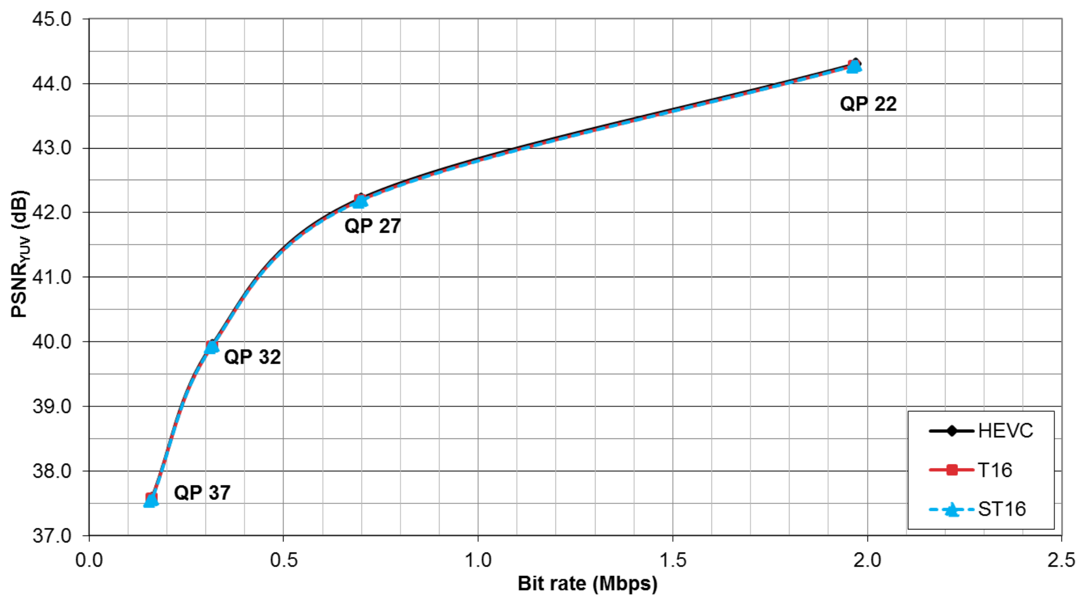


Fig. A.11 R-D$_{PSNR}$ curves of *E3 – KristenAndSara* sequence using original HEVC and approximated transform matrices, T16 and ST16, under LB configuration in Main profile

Appendix B

R-D Curves of HEVC and Approximated Quantisation

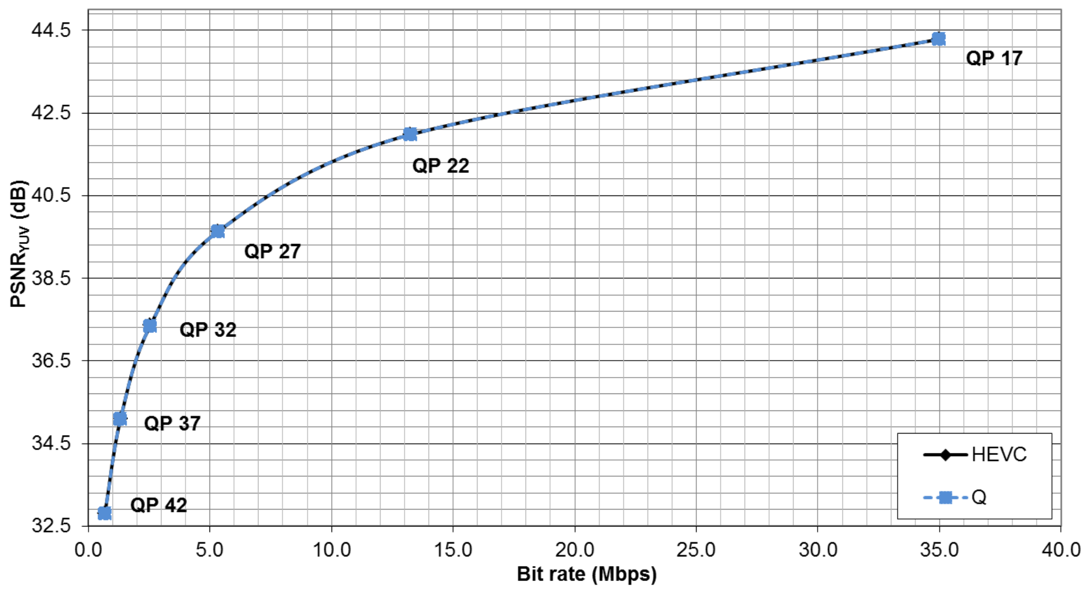


Fig. B.1 R-D$_{PSNR}$ curves of *A1 – Traffic* sequence using original HEVC and approximated quantisation multiplier set, Q, under RA configuration in Main profile

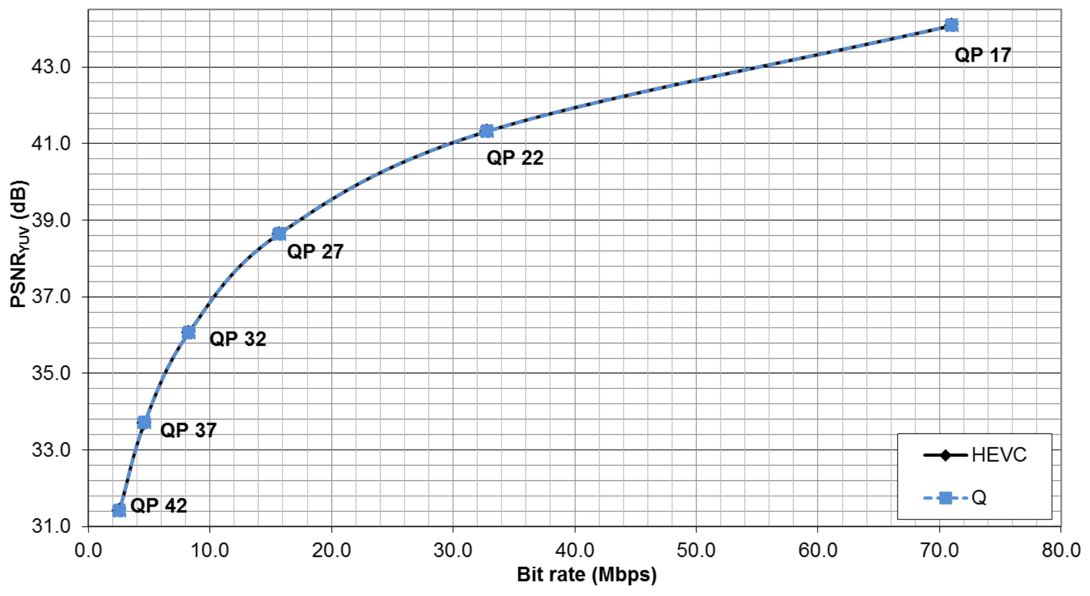


Fig. B.2 R-D$_{PSNR}$ curves of *A2* – *PeopleOnStreet* sequence using original HEVC and approximated quantisation multiplier set, Q, under RA configuration in Main profile

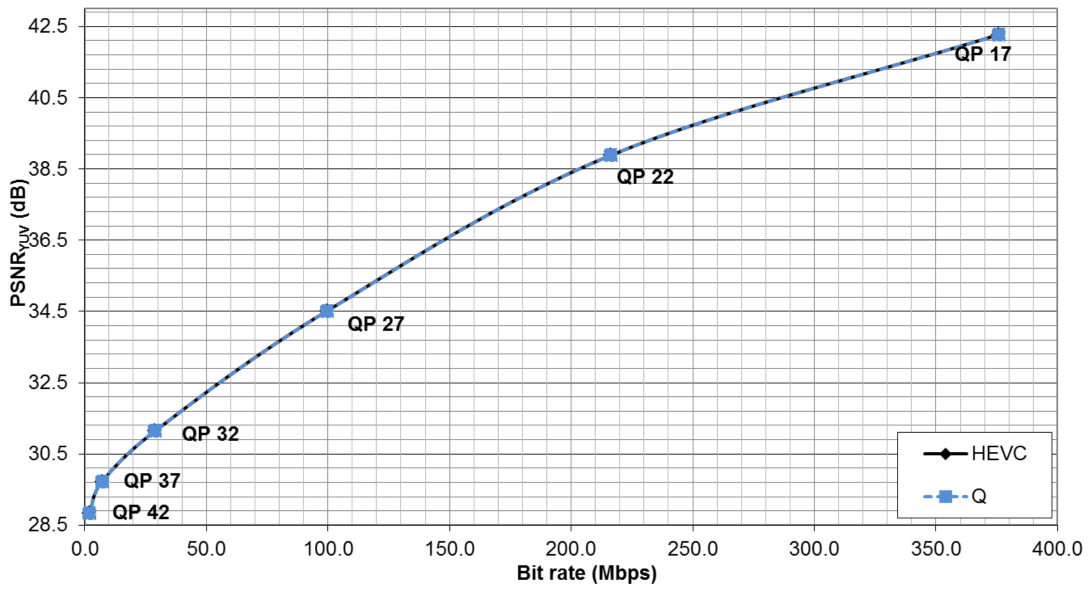


Fig. B.3 R-D$_{PSNR}$ curves of *A3* – *Nebuta* sequence using original HEVC and approximated quantisation multiplier set, Q, under RA configuration in Main profile

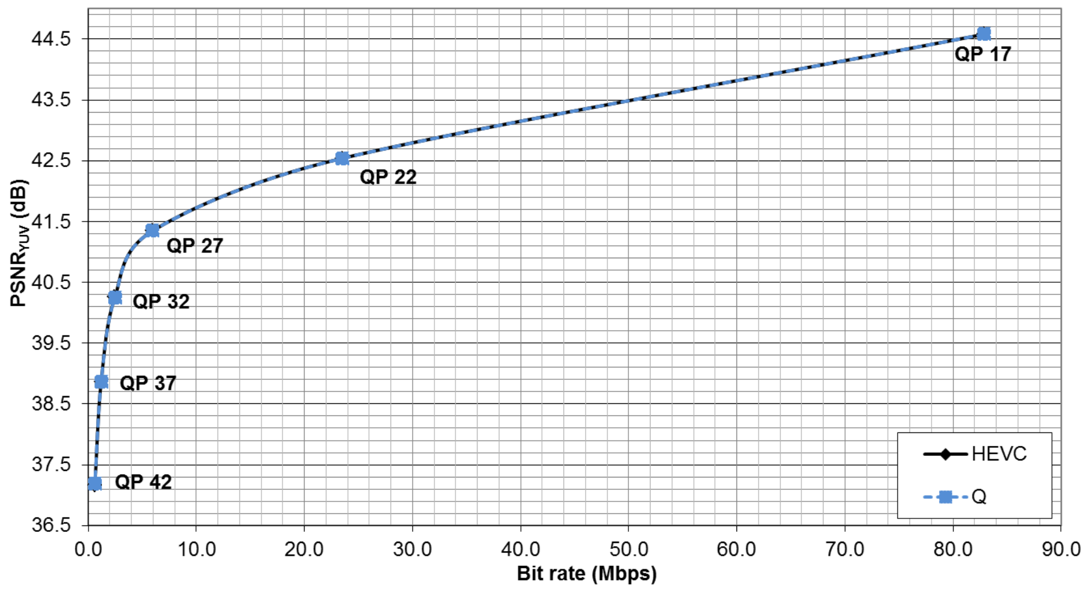
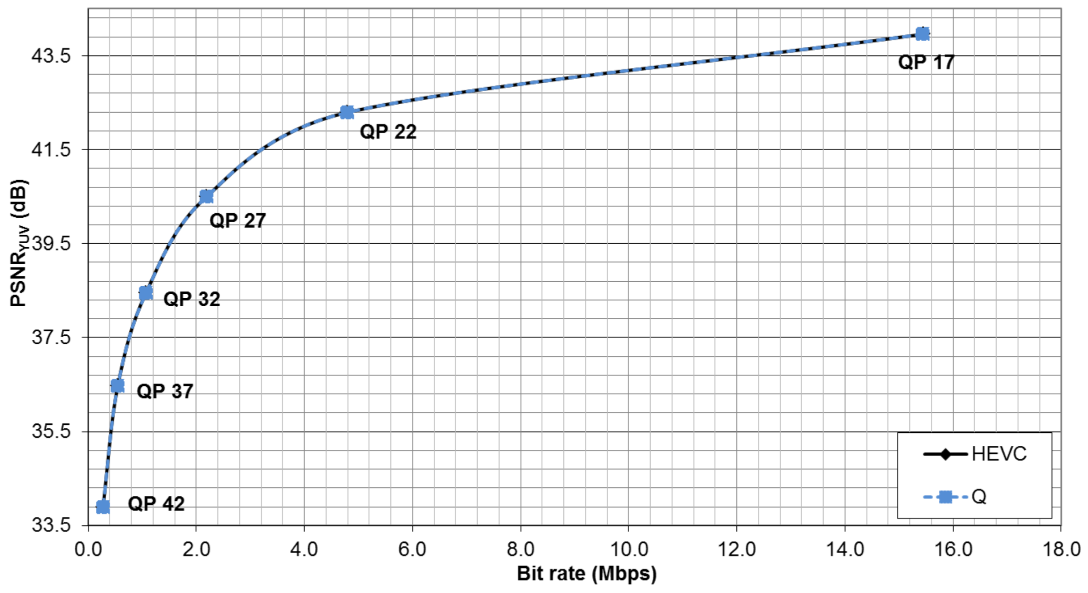
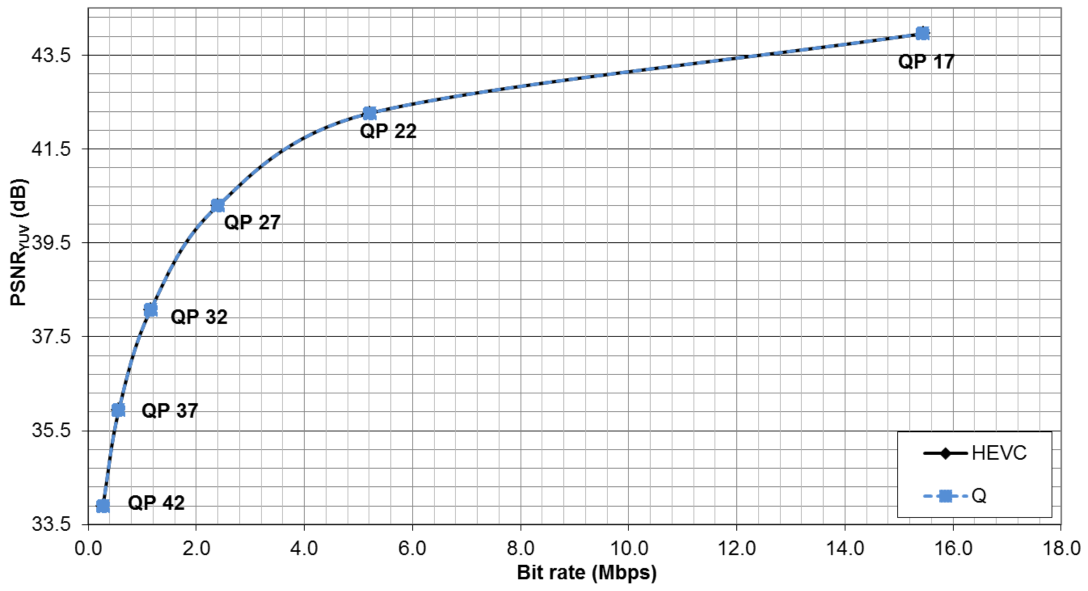


Fig. B.4 R-D$_{PSNR}$ curves of *A4 – SteamLocomotive* sequence using original HEVC and approximated quantisation multiplier set, Q, under RA configuration in Main profile

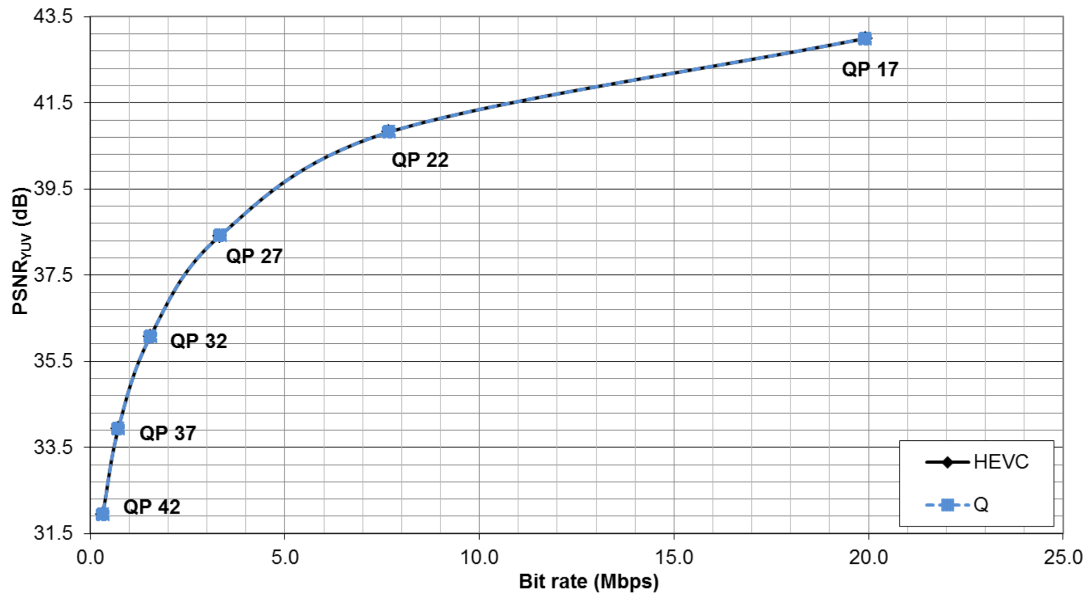


(a)

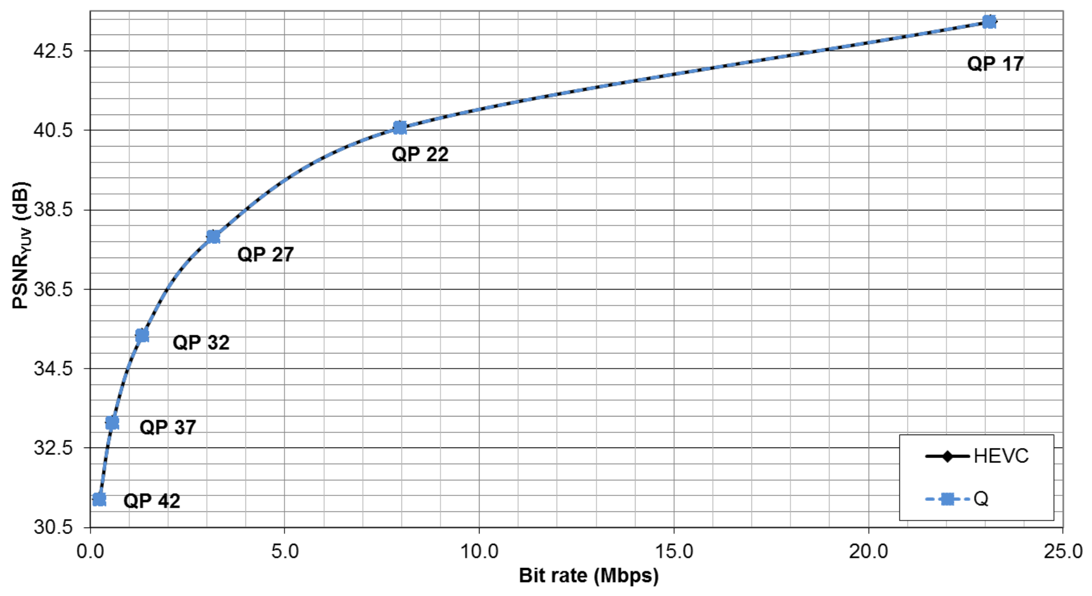


(b)

Fig. B.5 R-D$_{\text{PSNR}}$ curves of *B1 – Kimono* sequence using original HEVC and approximated quantisation multiplier set, Q, under (a) RA and (b) LB configurations in Main profile

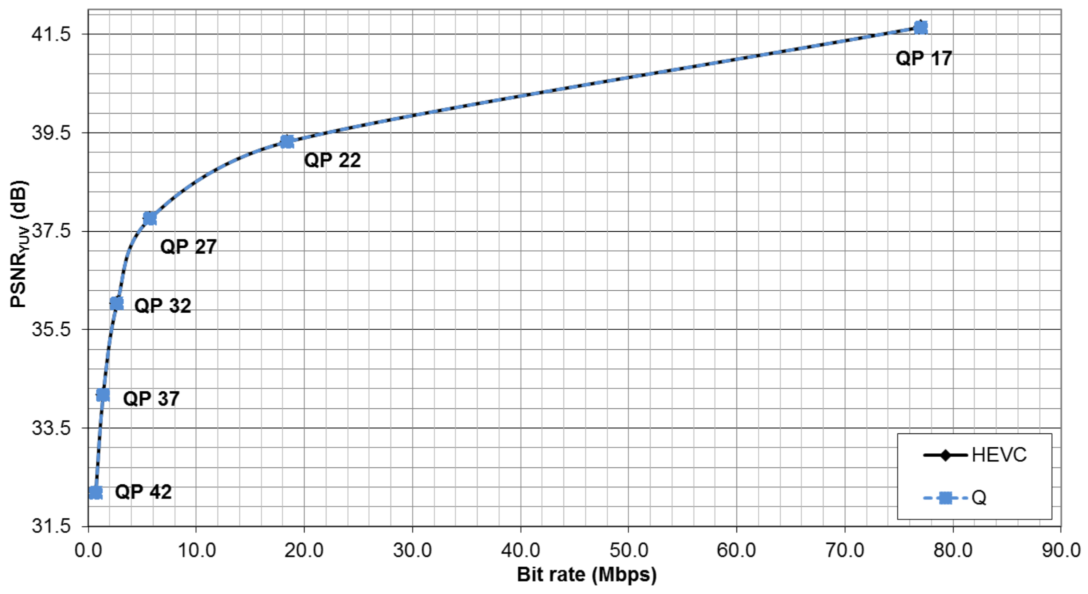


(a)

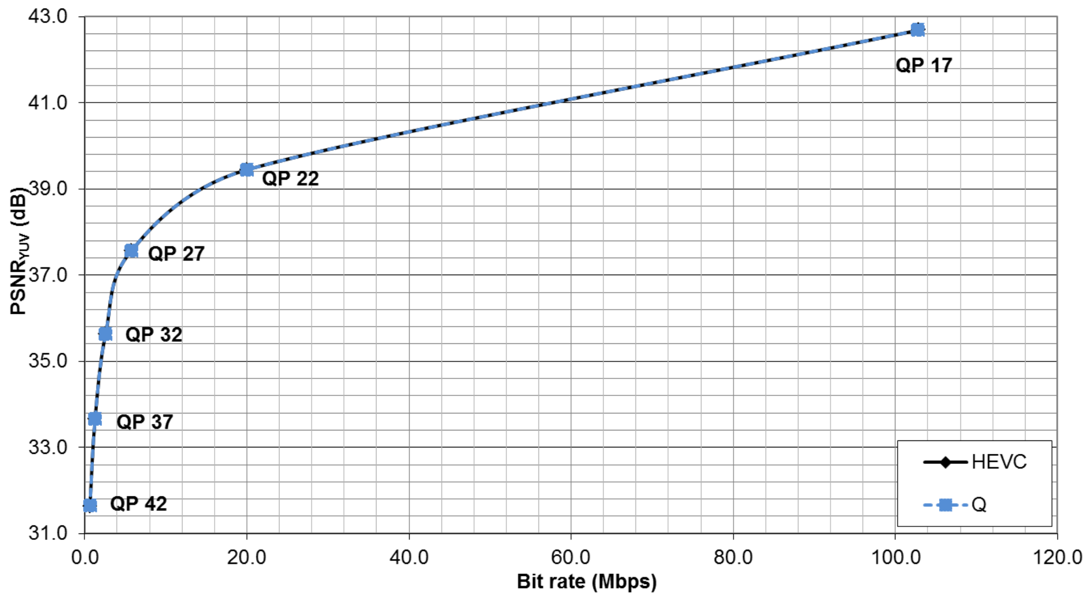


(b)

Fig. B.6 R-D$_{PSNR}$ curves of *B2 – ParkScene* sequence using original HEVC and approximated quantisation multiplier set, Q, under (a) RA and (b) LB configurations in Main profile

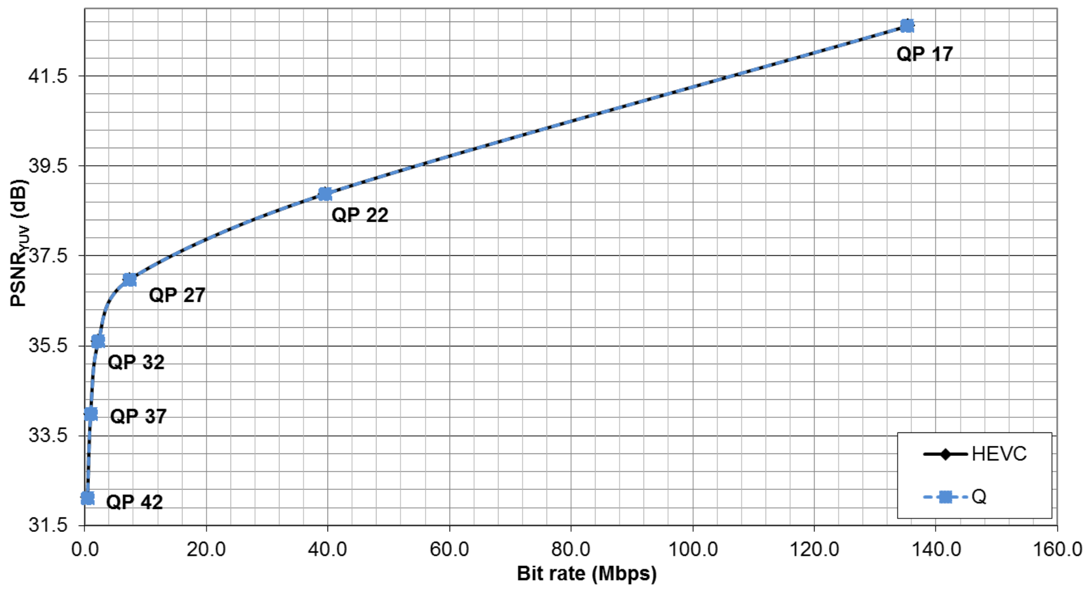


(a)

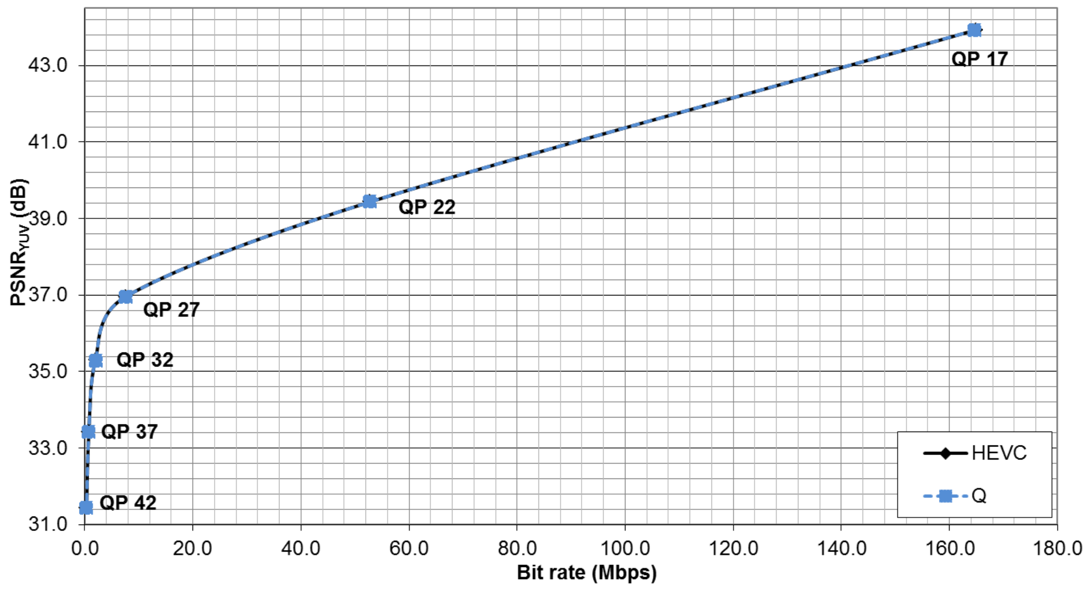


(b)

Fig. B.7 R-D$_{PSNR}$ curves of *B3* – *Cactus* sequence using original HEVC and approximated quantisation multiplier set, Q, under (a) RA and (b) LB configurations in Main profile



(a)



(b)

Fig. B.8 R-D$_{\text{PSNR}}$ curves of *B5 – BQTerrace* sequence using original HEVC and approximated quantisation multiplier set, Q, under (a) RA and (b) LB configurations in Main profile

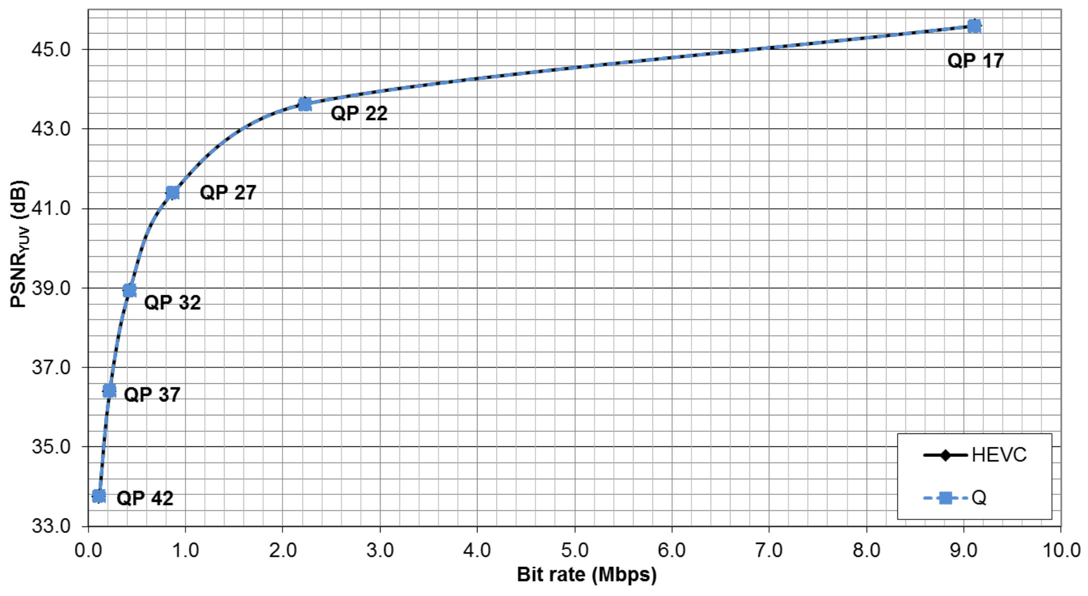


Fig. B.9 R-D$_{PSNR}$ curves of *E1 – FourPeople* sequence using original HEVC and approximated quantisation multiplier set, Q, under LB configuration in Main profile

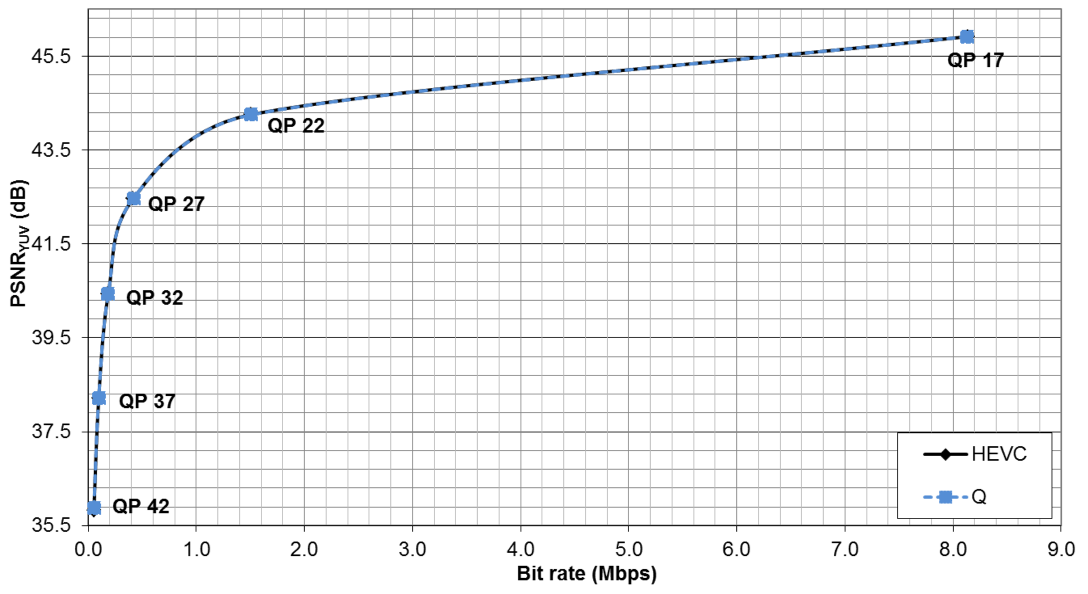


Fig. B.10 R-D$_{PSNR}$ curves of *E2 – Johnny* sequence using original HEVC and approximated quantisation multiplier set, Q, under LB configuration in Main profile

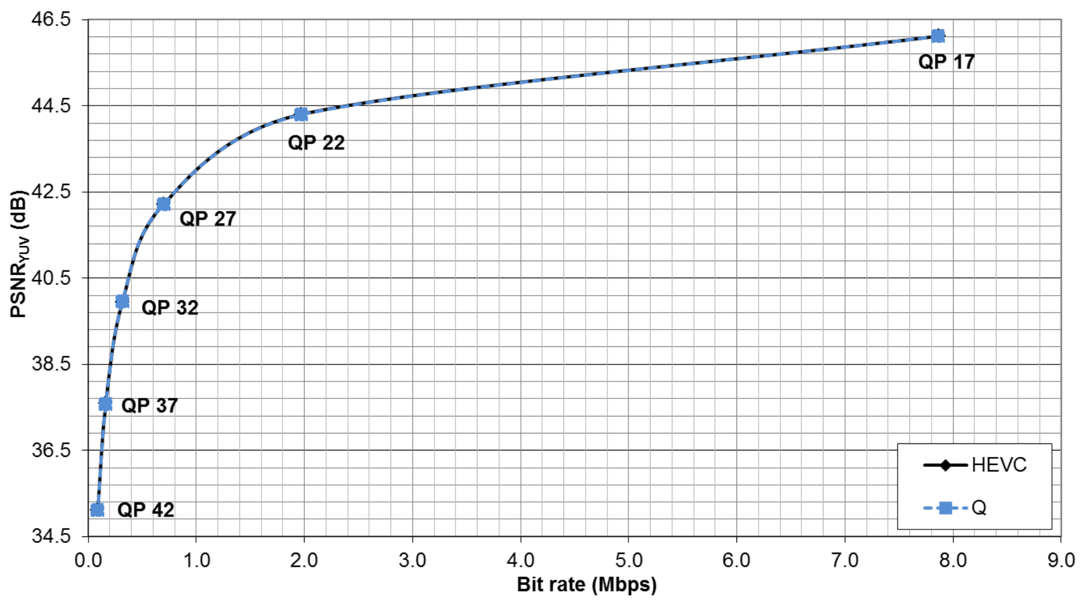


Fig. B.11 R-D$_{PSNR}$ curves of *E3 – KristenAndSara* sequence using original HEVC and approximated quantisation multiplier set, Q, under LB configuration in Main profile

Appendix C

R-D Curves of HEVC and Approximated Transform and Quantisation

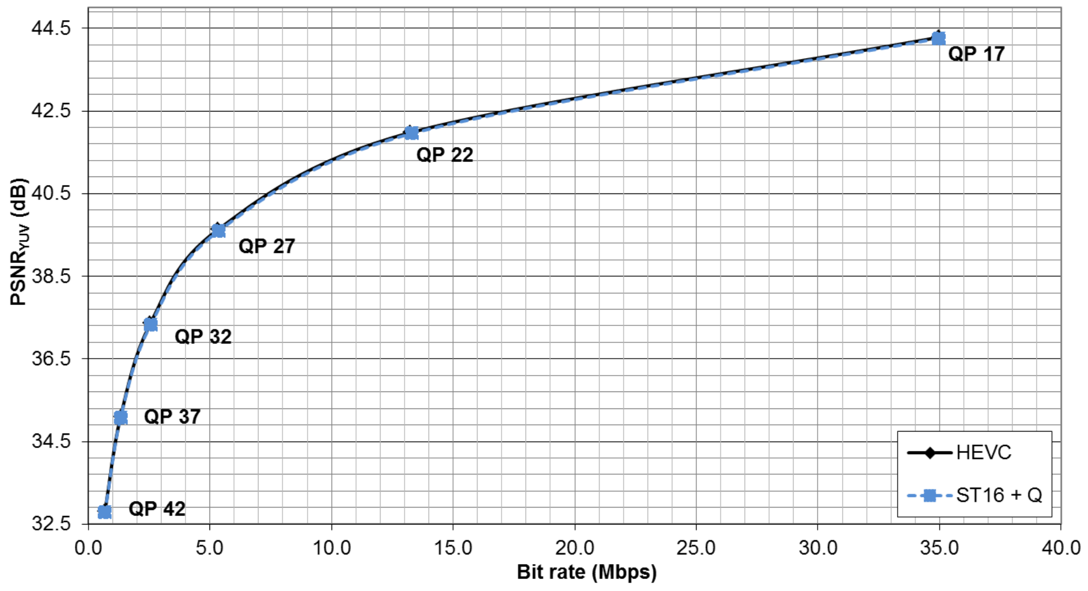


Fig. C.1 R-D$_{\text{PSNR}}$ curves of *A1 – Traffic* sequence using original HEVC and combination of approximated transform matrix and quantisation multiplier sets, ST16 + Q, under RA configuration in Main profile

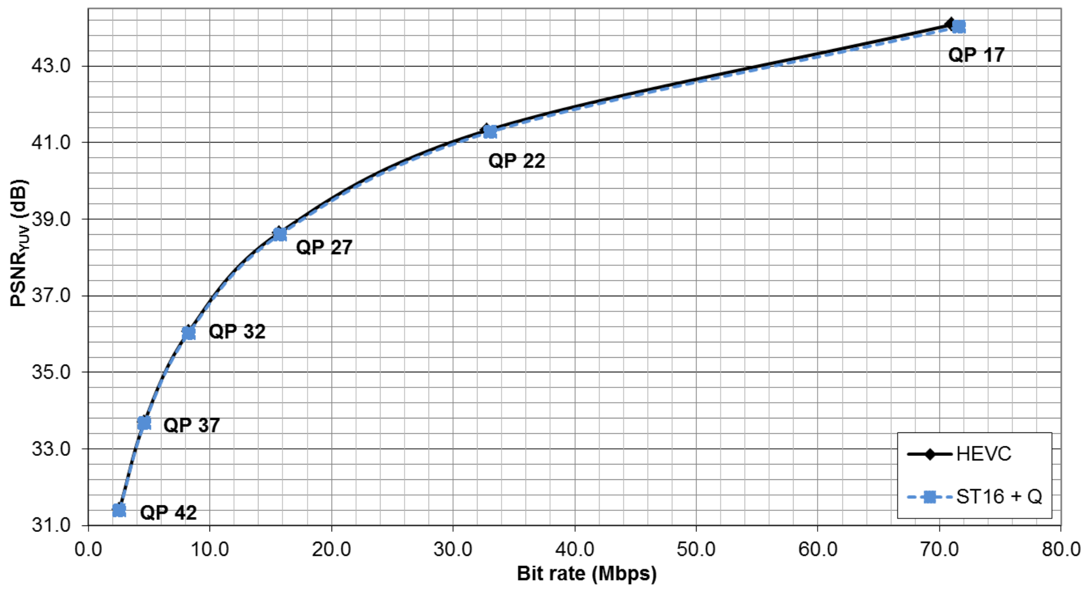


Fig. C.2 R-D$_{PSNR}$ curves of *A2* – *PeopleOnStreet* sequence using original HEVC and combination of approximated transform matrix and quantisation multiplier sets, ST16 + Q, under RA configuration in Main profile

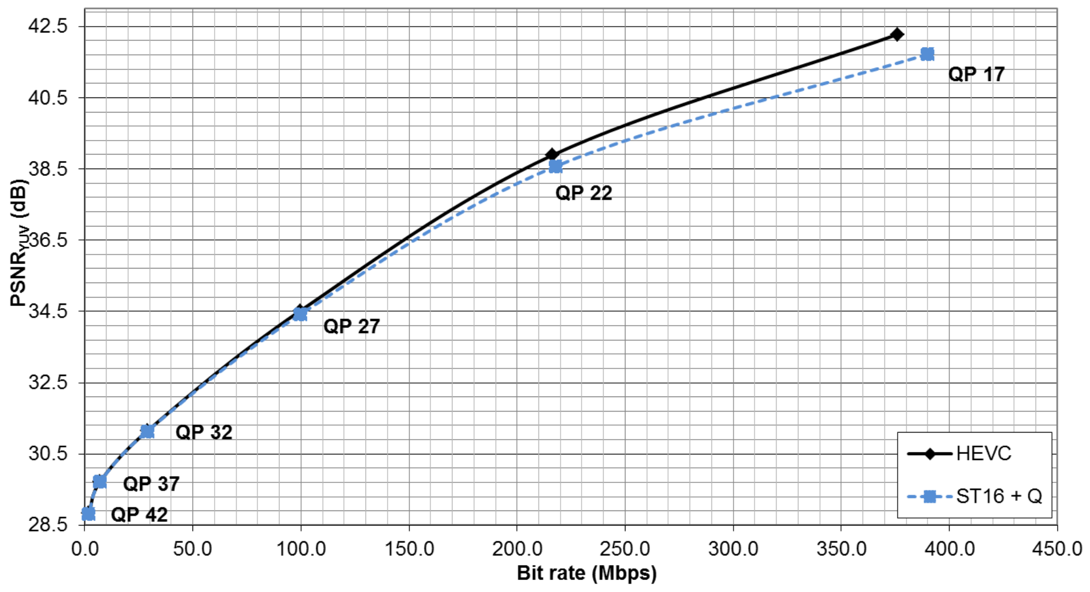


Fig. C.3 R-D$_{PSNR}$ curves of *A3* – *Nebuta* sequence using original HEVC and combination of approximated transform matrix and quantisation multiplier sets, ST16 + Q, under RA configuration in Main profile

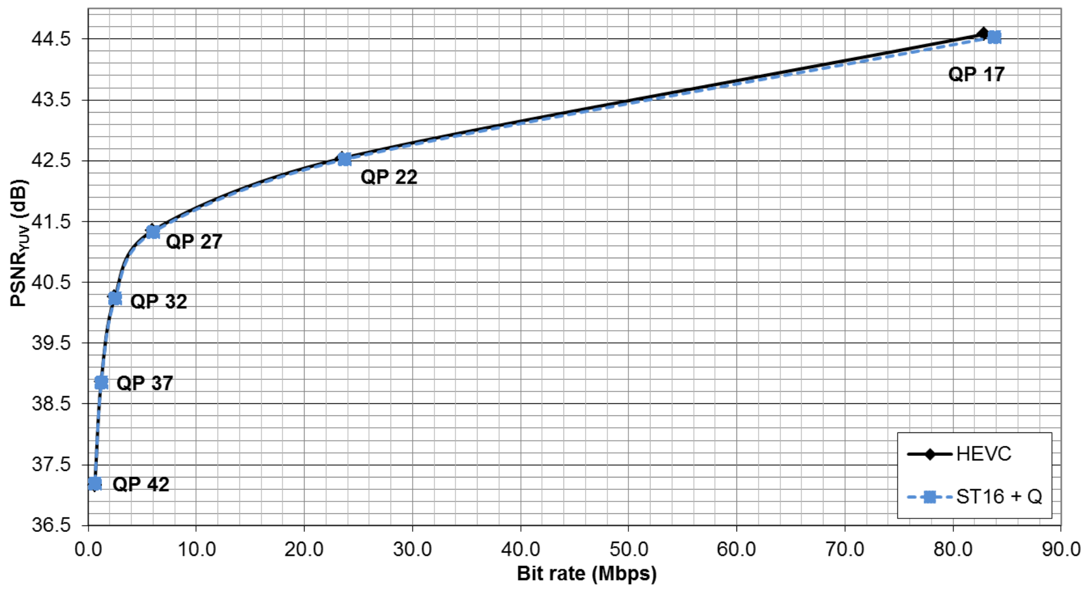
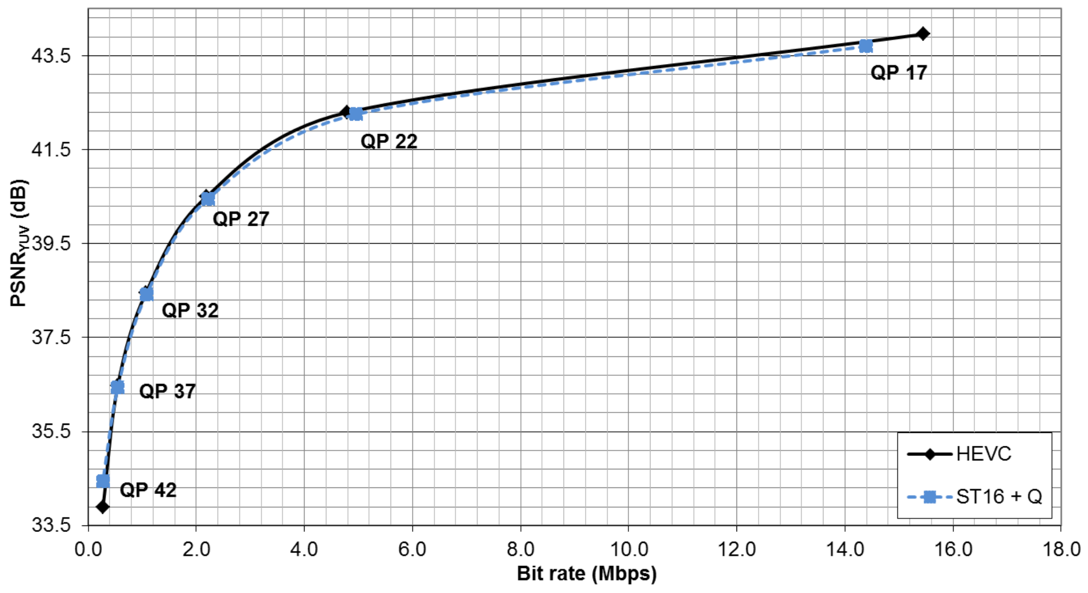
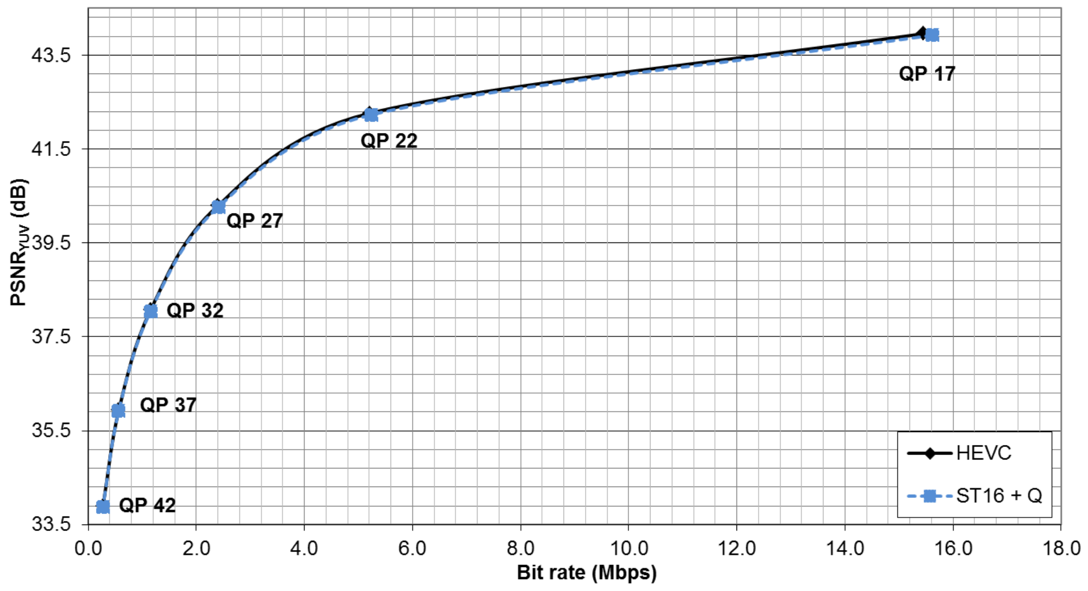


Fig. C.4 R-D$_{PSNR}$ curves of *A4 – SteamLocomotive* sequence using original HEVC and combination of approximated transform matrix and quantisation multiplier sets, ST16 + Q, under RA configuration in Main profile

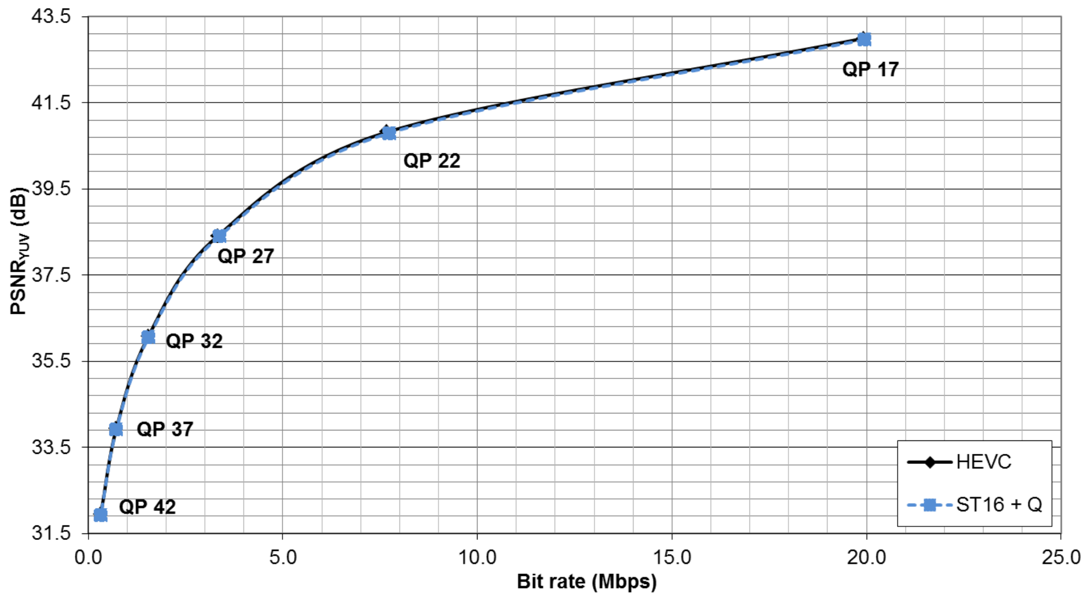


(a)

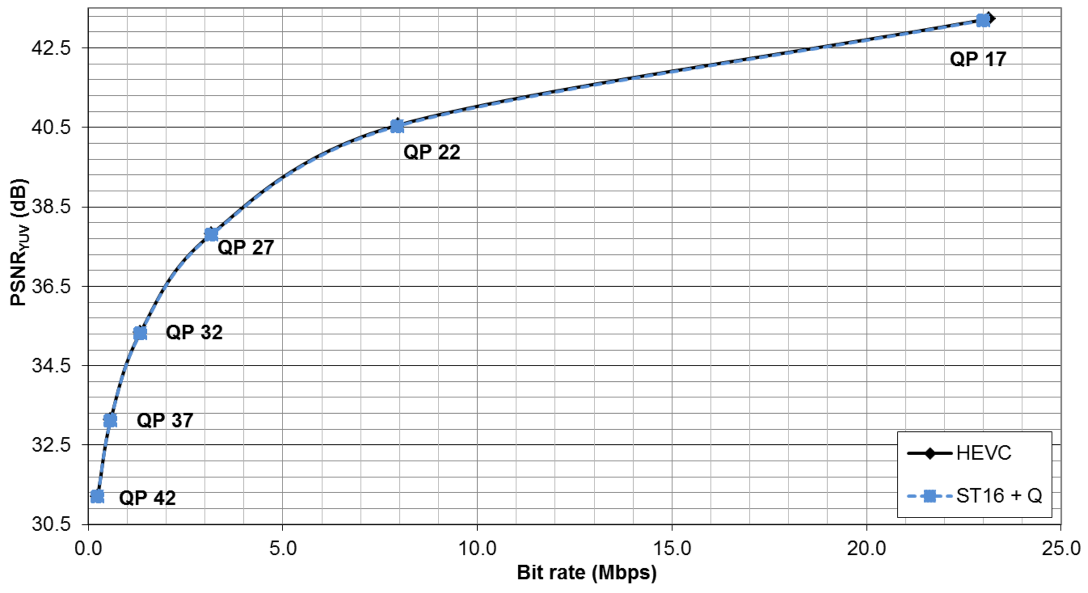


(b)

Fig. C.5 R-D$_{PSNR}$ curves of *B1 – Kimono* sequence using original HEVC and combination of approximated transform matrix and quantisation multiplier sets, ST16 + Q, under (a) RA and (b) LB configurations in Main profile

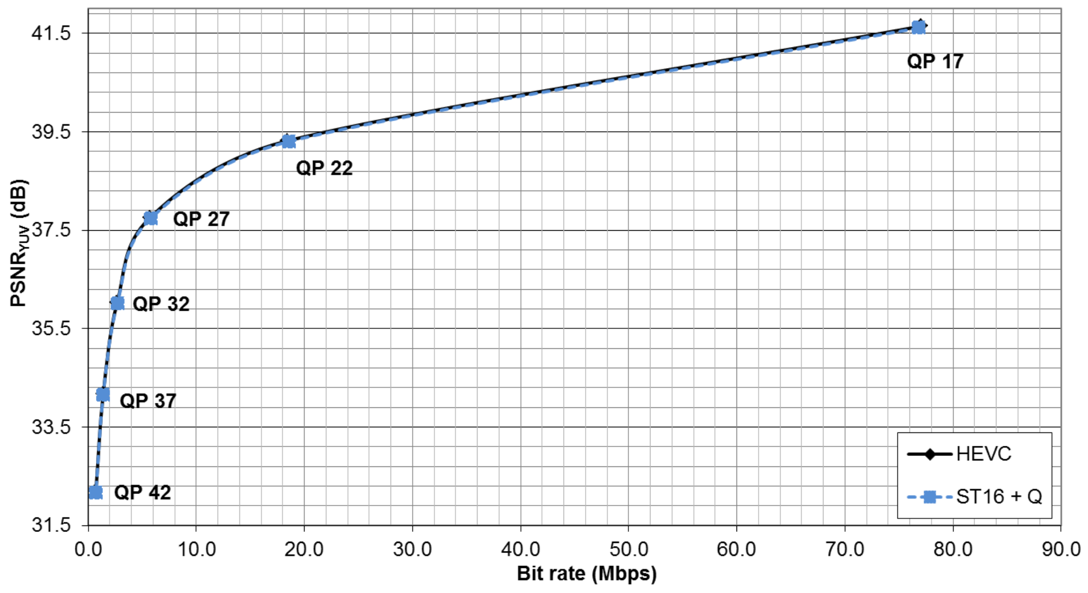


(a)

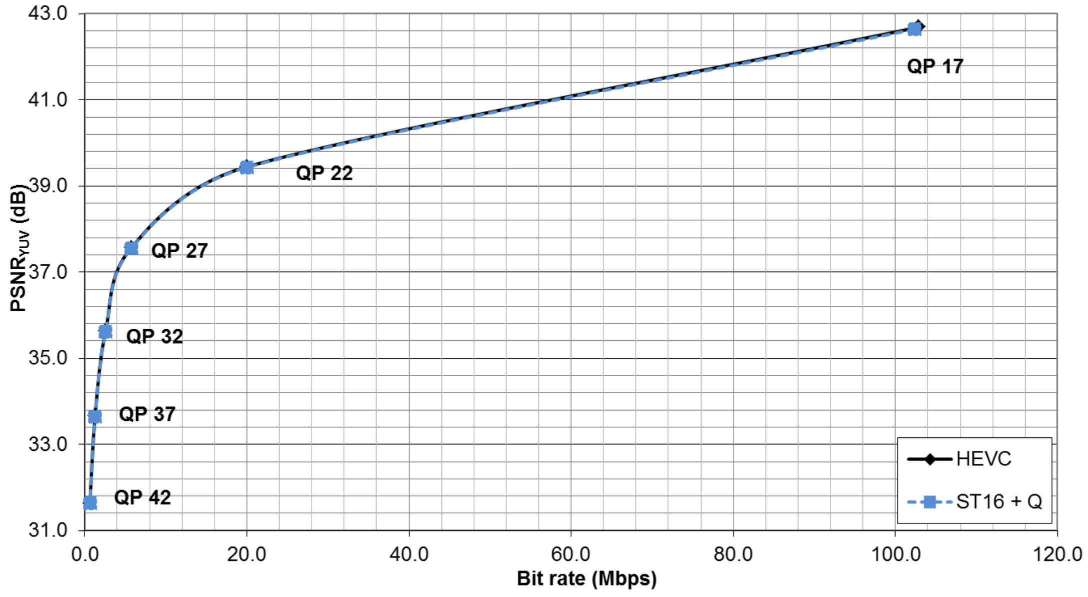


(b)

Fig. C.6 R-D$_{PSNR}$ curves of *B2 – ParkScene* sequence using original HEVC and combination of approximated transform matrix and quantisation multiplier sets, ST16 + Q, under (a) RA and (b) LB configurations in Main profile

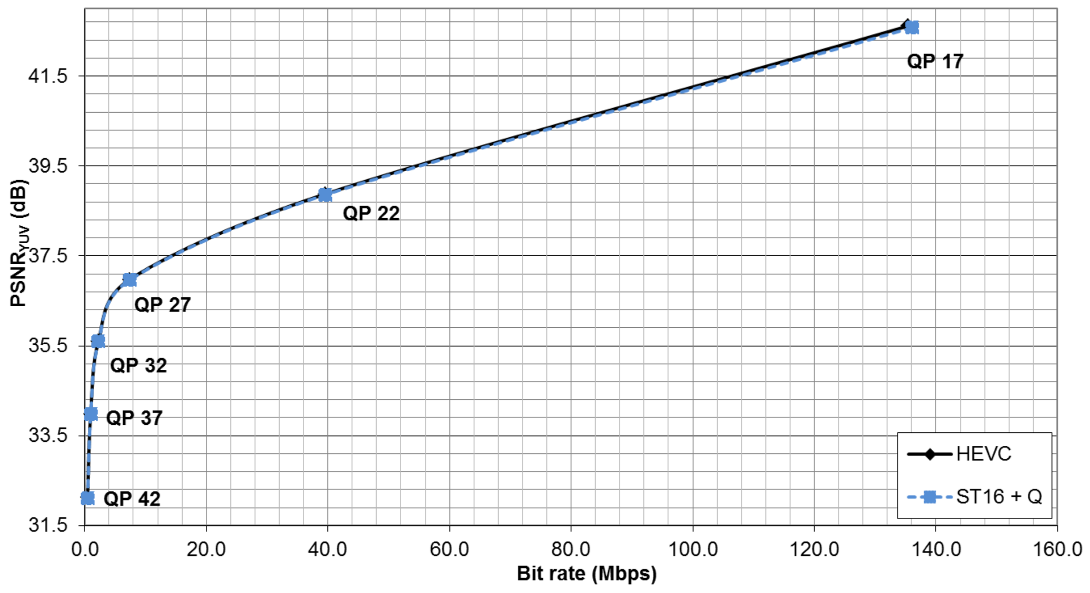


(a)

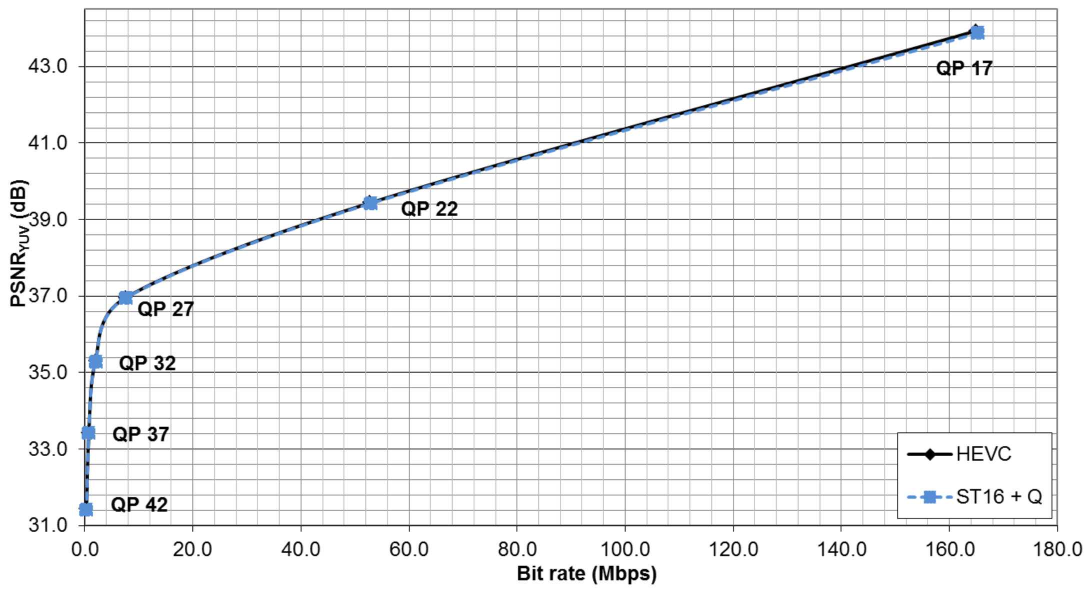


(b)

Fig. C.7 R-D$_{PSNR}$ curves of *B3 – Cactus* sequence using original HEVC and combination of approximated transform matrix and quantisation multiplier sets, ST16 + Q, under (a) RA and (b) LB configurations in Main profile



(a)



(b)

Fig. C.8 R-D$_{PSNR}$ curves of *B5 – BQTerrace* sequence using original HEVC and combination of approximated transform matrix and quantisation multiplier sets, ST16 + Q, under (a) RA and (b) LB configurations in Main profile

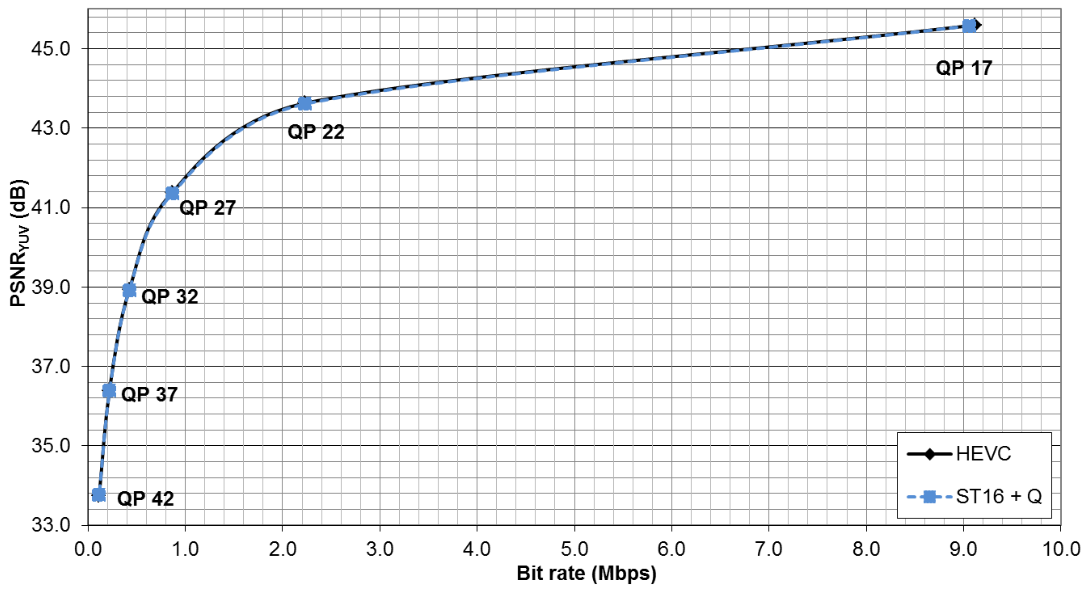


Fig. C.9 R-D$_{PSNR}$ curves of *E1 – FourPeople* sequence using original HEVC and combination of approximated transform matrix and quantisation multiplier sets, ST16 + Q, under LB configuration in Main profile

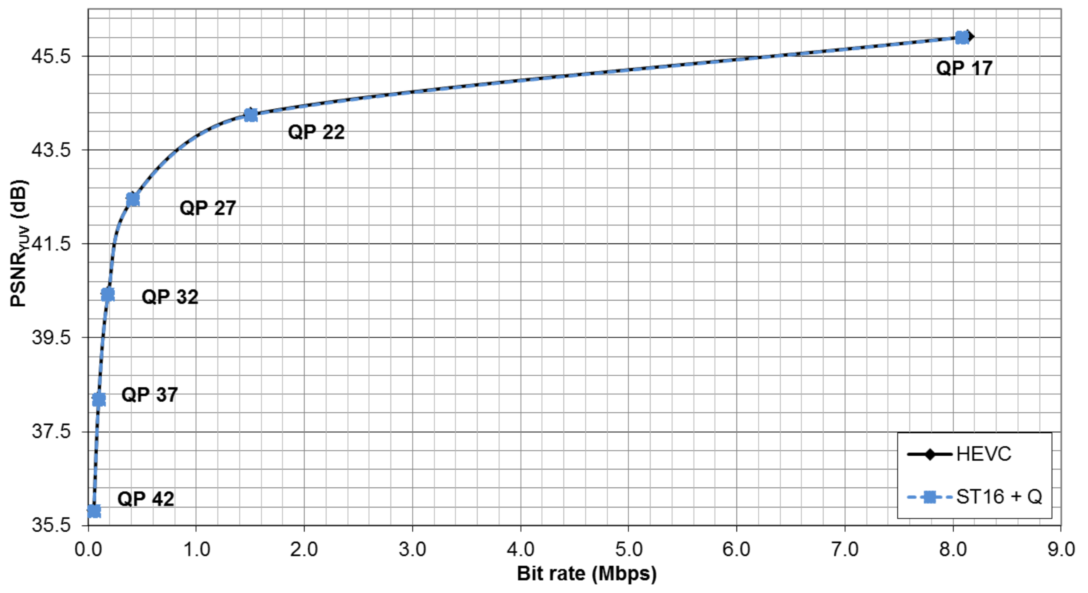


Fig. C.10 R-D$_{PSNR}$ curves of *E2 – Johnny* sequence using original HEVC and combination of approximated transform matrix and quantisation multiplier sets, ST16 + Q, under LB configuration in Main profile

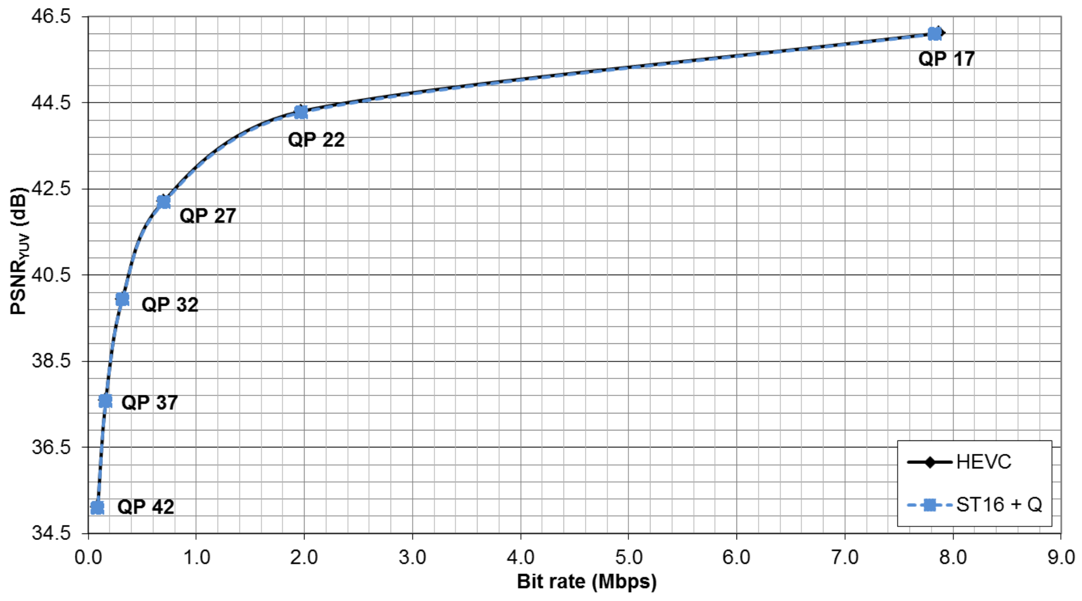


Fig. C.11 R-D$_{\text{PSNR}}$ curves of *E3* – *KristenAndSara* sequence using original HEVC and combination of approximated transform matrix and quantisation multiplier sets, ST16 + Q, under LB configuration in Main profile