TILE-LEVEL PARALLELISM FOR H.264/AVC CODEC USING PARALLEL DOMAIN DECOMPOSITION ALGORITHM ON SHARED MEMORY ARCHITECTURE

MOHAMMED F. EESSA

UNIVERSITI SAINS MALAYSIA

2015

TILE-LEVEL PARALLELISM FOR H.264/AVC CODEC USING PARALLEL DOMAIN DECOMPOSITION ALGORITHM ON SHARED MEMORY ARCHITECTURE

By

MOHAMMED F. EESSA

Thesis submitted in fulfilment of the requirements
for the degree of
Doctor of Philosophy

October 2015

ACKNOWLEDGMENTS

بســم الله الرحمن الرحيــم

فأما الزبد فيذهب جفاء وأما ما ينفع الناس فيمكث في الأرض كذلك يضرب الله الأمثال

صدق الله العظيم

All Praise is to Allah for giving me the courage to complete this thesis. I will like to express my gratitude to my parents, my wife Dr. Khansaa and all members of my family for their support during my study. I would also like to thank my son Elias and my daughter Misk, since their smile gives me that raging energy to keep going forward with my study to reach that wonderful goal.

I especially want to thank my supervisor, Prof Rosni Abdullah, for her guidance during my research at the **National Advanced IPv6 Centre (NAV6)/ Universti Sains Malaysia** (**USM**). Her perpetual energy and enthusiasm in research has motivated all her students, including me. In addition, she was always accessible and willing to help her students with their research. As a result, research life became smooth and rewarding for me

This thesis would not be possible without the support of my co-supervisor Dr. Ali Kattan. For that I say "thank you very much" for your help and encouragement and I should confess that it has been a joyful experience working with you.

I would like also to thank my Malaysian parents Syed Mohd Bakar and Natrah bt. Mohd Nazir for their support during my study and for their helps also in many situations here in Malaysia. It is really an amazing thing to be close or part from such a wonderful family.

Moreover, I am very grateful to the **Institute of Postgraduate Studies (IPS)** for offering the USM Fellowship as the financial support for my study

Last but not least, I am so grateful to **Universti Sains Malaysia** (**USM**) for the support, if it was not for their support this research would not have seen the light.

TABLE OF CONTENTS

Page

ACKNOWLEDGMENTS…….. ii

TABLE OF CONTENTS…… iv

LIST OF TABLES…….. x

LIST OF FIGURES…… xi

LIST OF ABBREVIATIONS……. xvi

LIST OF PUBLICATIONS…… xviii

ABSTRAK…… xix

ABSTRACT……. xxi

CHAPTER 1: INTRODUCTION

1.1 Introduction……. 1

1.2 Parallel Approaches of the H.264/AVC, a Brief……..…….. 3

1.2.1 Task-Level Approach……. 3

1.2.2 Data-Level Approach……. 4

1.3 Problem Statement…….. 6

1.4 Motivation……... 8

1.5 Research Objectives…… 8

1.6 Thesis Contribution……. 9

1.7 Scope and Limitation…….. 10

1.8 Thesis Organization……. 10

CHAPTER 2: BACKGROUND AND LITERATURES REVIEW

2.1 Introduction……. 12

2.2 Digital Video……... 12

2.3 The Importance of Video Compression…….. 13

2.4 Block-Based Hybrid Video Coding…… 14

2.5 The H.264/AVC Standard…….…….. 15

2.5.1 Prediction……... 16

2.5.2 Transform and Quantisation……... 17

2.5.3 Rate-Distortion Optimisation (RDO)……. 18

2.5.4 Deblocking Filter…… 19

2.5.5 Entropy Coding…….. 20

2.5.6 Profiles and Levels of the H.264/AVC Standard…….. 21

2.5.7 Dependencies of H.264/AVC Standard……. 21

2.5.8 Common H.264/AVC Coding Software…… 24

2.5.8.1 Overview to the JM Reference Encoder……. 25

2.5.8.2 JM Encoder Configuration File (*encoder.cfg*)…… 27

2.5.8.3 JM Encoder Output File…….. 27

2.6 Parallel Computing…….. 28

2.6.1 Level of Parallelism……... 29

2.6.2 Types of Parallelism……... 29

2.6.2.1 Task Decomposition…….. 29

2.6.2.2 Domain Decomposition…….. 30

2.6.3 Parallel Hardware Architectures…… 33

2.6.4 Parallel Programming languages…… 36

2.6.4.1 POSIX Threads…….…….... 35

2.6.4.2 Open Multiprocessing…… 36

2.6.4.3 Message Passing Interface…….…… 39

2.6.4.4 CUDA and OpenCL…….…….. 40

2.7 Parallel Approaches for H.264/AVC…….……. 41

2.7.1 GOP-Level…….…… 42

2.7.2 Frame-Level……..…….…… 44

2.7.3 Slice-Level…….…… 45

2.7.4 MB-Level…..…… 49

2.7.5 Tile-Level of other Video Encoders…….. 51

2.7.6 Summery of the Related Work…….. 53

2.8 Chapter Summary…… 56

CHAPTER 3: METHODOLOGY

3.1 Introduction……. 57

3.2 Research Framework..…… 57

3.3 Key Points of Native Parallel Algorithms…….. 59

3.4 The Design of the Natively-Parallel Domain Decomposition Algorithm... 59

3.5 Tile-Level Parallel H.264/AVC Encoder…… 60

3.6 Implementation, Results, and Evaluation…… 60

3.6.1 Setting up The Encoding Parameters……. 61

3.6.2 Experimental Setup…… 61

3.6.3 Video Test Sequence…….. 63

3.6.4 Video Quality Measurements……. 66

3.6.5 Evaluation Criteria for Parallel Video Encoders…… 67

3.7 Chapter Summary…… 69

CHAPTER 4: THE NATIVELY-PARALLEL DOMAIN DECOMPOSITION ALGORITHM

4.1 Introduction……. 70

4.2 The Inherently-Parallel 2D Domain Decomposition Algorithm……. 71

4.2.1 Preliminaries……... 72

4.2.2 Formal Problem Definition of Serial UBD …….. 73

4.2.3 Block Size (Dimensions)…… 73

4.2.4 Explicit Parallel Decomposition…….…… 76

4.2.4.1 One-Dimensional Domain…….……..…… 77

4.2.4.2 Two-Dimensional Domain……. 78

4.2.4.2 Three-Dimensional Domain……... 80

4.3 Algorithmic Optimisations.....…… 81

4.4 The Generalised Mathematical Formula for the Parallel UBD Algorithm. 82

4.5 Time Complexity of the Natively-Parallel UBD Algorithm…….. 86

4.6 Working Examples…….. 90

4.7 Chapter Summary…… 91

CHAPTER 5: THE PARALLEL TILE-LEVEL H.264/AVC ENCODER

5.1 Introduction……. 93

5.2 Data Flow of the Tile-level Parallel H.264/AVC Encoder……. 94

5.3 Framework of the Tile-level Parallel H.264/AVC Encoder…… 95

5.3.1 Defining Tiles (Blocks)…….. 97

5.3.2 Tile-Level H.264/AVC Parallel Encoding……. 101

5.3.2.1 Intra-Prediction…….. 101

5.3.2.2 Inter-Prediction…….. 103

5.3.2.3 Other Encoding Stages……... 106

5.4 Pros and Cons of the Parallel Tile-Level…….. 107

5.5 High-Level Parallelism and the Future of Multicore…….. 109

5.6 Chapter Summary…… 110

CHAPTER 6: IMPLEMENTATIONS, RESULTS, AND EVALUATIONS

6.1 Introduction……. 112

6.2 Implementation Details of: Parallel Slice-Level and Parallel Tile-Level H.264/AVC Encoders..…….. 113

6.2.1 Parallel Tile-Level (without Overlapping)…… 114

6.2.2 Parallel Tile-Level (with Overlapping)……. 115

6.2.3 Parallel Slice-Level……... 116

6.2.4 Code Verification (Conformance Testing)…… 116

6.3 Results and Evaluations…….. 117

6.3.1 Encoding Time……... 118

6.3.2 Parallel Speedup……. 122

6.3.3 Parallel Efficiency…….. 127

6.3.4 Parallel Scalability…….. 131

6.3.5 PSNR…….. 133

6.3.6 Bit Rate……..... 137

6.4 Comparisons with other H.264/AVC Parallel Approaches……. 141

6.5 Chapter Summary…… 142

CHAPTER 7: CONCLUSION AND FUTURE WORKS

7.1 Conclusion……... 144

7.2 Future Work…… 145

REFERENCES…… 146

APPENDICES……..... 160

APPENDIX A: INDEXES OF BLOCKS DIMENSIONS……... 161

APPENDIX B: NUMERICAL RESULTS FOR PARALLEL AND VIDEO QUALITY METRICS…… 164

LIST OF TABLES

		Page
Table 2.1	Evaluation of the related work	55
Table 3.1	Encoding parameters	61
Table 3.2	Full HD Video test sequences (1080p)	64
Table 3.3	HD Video test sequences (720p)	64
Table 4.1	Scenarios for one, two, and three dimensions parallel partitioning	91
Table 5.1	MB-equivalent of tile sizes	100
Table 6.1	Average speedup of overlapping and non-overlapping	132
Table 6.2	Speedup comparison with previous parallel approaches	142

LIST OF FIGURES

		Page
Figure 1.1	H.264/AVC data structure (Gu, J. and Sun, Y., 2011)	5
Figure 2.1	Block-based hybrid encoder block diagram (Ziyi, H. et al., 2011)	15
Figure 2.2	Variant block sizes of inter & intra predictions	17
Figure 2.3	4 x 4 H.264/AVC transform matrix	18
Figure 2.4	Horizontal and vertical edges filtering in a MB	20
Figure 2.5	Inter frame dependency (IBBPBBP sequence)	23
Figure 2.6	Inter frame dependency (IBPBP sequence)	23
Figure 2.7	Deblocking filter across the slice boundaries	24
Figure 2.8	JM encoder software operation (Richardson, I. E., 2010b)	26
Figure 2.9	JM reference encoder's output (Hameed, 2013)	28
Figure 2.10	Generalised block distribution (GBD) approaches	31
Figure 2.11	Non-overlapping versus overlapping domain decomposition	32
Figure 2.12	Distributed/shared memory architecture	34
Figure 2.13	Threads interaction (Barney, B., 2014)	36
Figure 2.14	Fork-join model of OpenMP	37
Figure 2.15	Syntax of parallelism of OpenMP	37
Figure 2.16	Syntax of *for* loop parallelism using OpenMP	38
Figure 2.17	CUDA grid, blocks and threads.	40
Figure 2.18	Hierarchical H.264/AVC parallel encoder	43
Figure 2.19	Adaptive slice number selection for parallel H.264/AVC encoding	47
Figure 2.20	Strip-wise parallel H.264/AVC encoding	48

Figure 2.21	MB region partitioning of a frame	50
Figure 2.12	MB-level parallelism using wavefront method	51
Figure 3.1	Research Framework	58
Figure 3.2	Sequence diagram of the evaluation	63
Figure 3.3	Full HD Video test sequences (1080p)	65
Figure 3.4	HD Video test sequences (720p)	65
Figure 4.1	Block diagram of the natively parallel UBD algorithm	72
Figure 4.2	Determining the size of subdomains (blocks)	75
Figure 4.3	One-dimensional decomposition	77
Figure 4.4	Parallel UBD of one-dimensional space	77
Figure 4.5	Thread accessing memory beyond the array size	78
Figure 4.6	Parallel UBD of one-dimensional space with memory accessing treatment	78
Figure 4.7	Two-dimensional decomposition	79
Figure 4.8	Parallel UBD of two-dimensional space with memory accessing treatment	80
Figure 4.9	Three-dimensional decomposition	81
Figure 4.10	Parallel UBD of three-dimensional space with memory accessing treatment	81
Figure 4.11	Different scenarios of 2D space partitioning	82
Figure 4.12	Ranking of nested loops	83
Figure 4.13	Pseudocode of the Serial UBD Algorithm	88
Figure 4.14	Pseudocode of the Natively-Parallel UBD Algorithm	89
Figure 5.1	Data flow of the tile-level parallel H.264/AVC encoder	94
Figure 5.2	Tile-level H.264/AVC parallel framework	96
Figure 5.3	Different video resolutions (approximated)	99
Figure 5.4	4-tile scenario	99

Figure 5.5	8-tile scenario	99
Figure 5.6	Parallel intra encoding, 4-tile scenario (HD resolution)	102
Figure 5.7	Parallel intra encoding, 8-tile scenario (HD resolution)	102
Figure 5.8	Inter and intra prediction switching	103
Figure 5.9	2 x 2 Overlapped Tiles	105
Figure 5.10	2 x 4 Overlapped tiles	105
Figure 5.11	Motion estimation beyond the tile boundaries	106
Figure 5.12	Slice-level versus tile-level	108
Figure 6.1	Bitstream conformance testing	116
Figure 6.2	Encoding time of 720p video sequences *(p = 2)*	118
Figure 6.3	Encoding time of 720p video sequences *(p = 4)*	118
Figure 6.4	Encoding time of 720p video sequences *(p = 6)*	119
Figure 6.5	Encoding time of 720p video sequences *(p = 8)*	119
Figure 6.6	Average encoding time of 720p video sequences	120
Figure 6.7	Encoding time of 1080p video sequences *(p = 2)*	120
Figure 6.8	Encoding time of 1080p video sequences *(p = 4)*	121
Figure 6.9	Encoding time of 1080p video sequences *(p = 6)*	121
Figure 6.10	Encoding time of 1080p video sequences *(p = 8)*	121
Figure 6.11	Average encoding time of 1080p video sequences	122
Figure 6.12	Parallel speedup of 720p video sequences *(p = 2)*	123
Figure 6.13	Parallel speedup of 720p video sequences *(p = 4)*	124
Figure 6.14	Parallel speedup of 720p video sequences *(p = 6)*	124
Figure 6.15	Parallel speedup of 720p video sequences *(p = 8)*	124
Figure 6.16	Parallel Speedup of 1080p video sequences *(p = 2)*	125
Figure 6.17	Parallel Speedup of 1080p video sequences *(p = 4)*	125
Figure 6.18	Parallel Speedup of 1080p video sequences *(p = 6)*	125

Figure 6.19	Parallel Speedup of 1080p video sequences *(p = 8)*	126
Figure 6.20	Average parallel Speedup of 720p video sequences	126
Figure 6.21	Average parallel Speedup of 1080p video sequences	127
Figure 6.22	Parallel efficiency of 720p video sequences *(p = 2)*	128
Figure 6.23	Parallel efficiency of 720p video sequences *(p = 4)*	128
Figure 6.24	Parallel efficiency of 720p video sequences *(p = 6)*	128
Figure 6.25	Parallel efficiency of 720p video sequences *(p = 8)*	129
Figure 6.26	Parallel efficiency of 1080p video sequences *(p = 2)*	129
Figure 6.27	Parallel efficiency of 1080p video sequences *(p = 4)*	129
Figure 6.28	Parallel efficiency of 1080p video sequences *(p = 6)*	130
Figure 6.29	Parallel efficiency of 1080p video sequences *(p = 8)*	130
Figure 6.30	Average parallel efficiency of 720p video sequences	131
Figure 6.31	Average parallel efficiency of 1080p video sequences	131
Figure 6.32	Predicted speedup of tile non-overlapping (*p = 16*)	132
Figure 6.33	Predicted speedup of tile overlapping (*p = 16*)	133
Figure 6.34	PSNR of 720p video sequences *(p = 2)*	134
Figure 6.35	PSNR of 720p video sequences *(p = 4)*	134
Figure 6.36	PSNR of 720p video sequences *(p = 6)*	134
Figure 6.37	PSNR of 720p video sequences *(p = 8)*	135
Figure 6.38	PSNR of 1080p video sequences *(p = 2)*	135
Figure 6.39	PSNR of 1080p video sequences *(p = 4)*	135
Figure 6.40	PSNR of 1080p video sequences *(p = 6)*	136
Figure 6.41	PSNR of 1080p video sequences *(p = 8)*	136
Figure 6.42	Average PSNR of 720p video sequences	136
Figure 6.43	Average PSNR of 1080p video sequences	137
Figure 6.44	Bit rate of 720p video sequences (*p = 2*)	138

Figure 6.45	Bit rate of 720p video sequences ($p = 4$)	138
Figure 6.46	Bit rate of 720p video sequences ($p = 6$)	139
Figure 6.47	Bit rate of 720p video sequences ($p = 8$)	139
Figure 6.48	Bit rate of 1080p video sequences *(p = 2)*	139
Figure 6.49	Bit rate of 1080p video sequences *(p = 4)*	140
Figure 6.50	Bit rate of 1080p video sequences *(p = 6)*	140
Figure 6.51	Bit rate of 1080p video sequences *(p = 8)*	140
Figure 6.52	Average bit rate of 720p video sequences	141
Figure 6.53	Average bit rate of 1080p video sequences	141

LIST OF ABBREVIATIONS

API	Application programming interface
ASIC	Application specific integrated circuit
AVC	Advanced video codec
B-frame	Bidirectional predicted frame
CABAC	Context-based adaptive binary arithmetic coding
CAVLC	Context-adaptively switched sets of variable length codes
CEAA	Co-exploration between algorithm and architecture
CIF	Common intermediate format
CPU	Central processing unit
CUDA	Compute unified device architecture
dB	Logarithmic decibel
DCT	Discrete cosine transform
FPGA	Field programmable gate array
FPS	Frames per second
GBD	Generalised block distribution
GHz	Gigahertz
GOP	Group-of-picture
GPGPU	General-purpose computing on graphics processing units
GPU	Graphical processing unit
HD	High definition
HEVC	High efficiency video coding
HPC	High performance computing
HVS	Human visual system
IDE	Integrated development environment
I-frame	Intra-coded frame

IPTV	Internet protocol television
JM	Joint model
JPEG	Joint photographic experts group
JVT	Joint video team
MB	Macroblock
ME	Motion estimation
MPEG	Moving picture experts group
MPI	Message passing interface
MV	Motion vector
NP	Nondeterministic polynomial time
OpenCL	Open computing language
OpenMP	Open multiprocessing
OS	Operation system
P-frame	Predicted frame
PMV	Predicted motion vector
PSNR	Peak signal-to-noise ratio
Pthreads	POSIX threads
QCIF	Quarter common intermediate format
QP	Quantisation parameter
RDO	Rate-distortion optimisation
RGB	Red, green, and blue
SD	Standard definition
thread ID	Thread identification
UBD	Uniform block distribution
VCEG	Video coding experts group

LIST OF PUBLICATIONS

Published Mohammed Faiz Aboalmaaly, Adel Nadhem Naeem, Hala A. Albaroodi, Sureswaran Ramadass, "Parallel H.264/AVC Encoder: a Survey", *International Journal of Advancements in Computing Technology (IJACT)*, Vol 5, No. 9, pp. 334-341, 2013 (**Scopus**)

Published Mohammed Faiz Aboalmaaly, Ali abdulrazzaq khudher, Hala A. Albaroodi, Sureswaran Ramadass, "Performance Analysis Between Explicit Scheduling And Implicit Scheduling Of Parallel Array-Based Domain Decomposition Using Openmp", *Journal of Engineering Science and Technology*, Vol. 9, No. 5, pp. 522-532, 2014 (**Scopus**)

Published Mohammed Faiz Aboalmaaly, Rosni Abdullah, Ali Kattan. "New Domain Decomposition Method for Quality-Aware Parallel H.264 Video Coding", *Malaysian Journal of Computer Science*, Vol. 27, issue 3, 2014 (**ISI Impact Factor 0.5**)

Published Mohammed Faiz Aboalmaaly, Rosni Abdullah, Ali Kattan, "Data-Level Parallel Approaches for the H.264 Coding: A Review", *First International Engineering Conference (IEC2014)*, Ishik University, Erbil, Kurdistan, Iraq, November 24-26, (2014)

Accepted Jun 2014 Mohammed Faiz Aboalmaaly, Sureswaran Ramadass, Mohammed Anbar, Hala A. Albaroodi "A New Parallel Algorithmic Design for a Uniform Block Distribution (UBD) Problem", *INFORMATION*, (**Scopus**)

Presented Aug 2014 Mohammed F Eessa, Rosni Abdullah, Ali Kattan, "H.264 Tiling: A New Parallel and Quality-Aware Approach For Parallelizing H.264 Codec", *Poster Presentation in the Computer Science Postgraduate Colloquium*, 26-27 August, Pusat Latihan Zakat (PULAZA) Balik Pulau, Pulau Penang, Malaysia, (2014)

Accepted Jan 2015 Mohammed Faiz Aboalmaaly, Rosni Abdullah, Ali Kattan, "Data-Level Parallel Approaches for the H.264/AVC: A Review From Encoder and Decoder Perspectives", *International Journal of Informatics and Communication Technology (IJ-ICT).*

PENSELARIAN PERINGKAT-JUBIN UNTUK KODEKS H.264/AVC MENGGUNAKAN ALGORITMA PENGURAIAN DOMAIN SELARI PADA SENI BINA MEMORI YANG DIKONGSI

ABSTRAK

Tema tesis ini adalah berdasarkan kepada penggunaan ciri-ciri model selari dalam fasa reka bentuk algoritma untuk mengurangkan kerumitan pengiraan dalam perbandingan dengan algoritma bersiri. Dengan menganggap bahawa seni bina selari membentuk majoriti pengiraan nod dalam peranti digital, cadangan bagi algoritma selari-inheren adalah sesuai. Dalam karya ini, proses atau pengenalan bebenang didaftar dalam satu formula matematik untuk mengurai domain satu, dua, dan domain tiga dimensi. Penyelesaian senario ruang dua dimensi seterusnya disesuaikan sebagai tahap baru keselarian untuk pengekodan piawaian H.264/AVC kerana kerumitan pengiraan yang lebih tinggi daripada pengekodan video ini berbanding dengan piawaian sebelumnya. Tahap baru keselarian untuk pengekod H.264 / AVC ini telah direka untuk mempertimbangkan beberapa metrik pengekodean video dan berorientasikan selari. Kaedah selari peringkat-jubin H.264/AVC yang dicadangkan dibandingkan dengan pendekatan selari tahap kepingan dan tahap blok makro. Perbandingan dibuat berhubungkait dengan pelaksanaan garis dasar (bersiri). Kecepatan, kecekapan selari, kadar bit, dan kadar signal puncak kepada ganggu (PSNR) digunakan sebagai metrik untuk semua pendekatan. Hasil kajian pada selari peringkat-jubin H.264 /AVC yang dicadangkan mengatasi tahap kepingan yang sebelum ini dikenali sebagai pendekatan yang paling sesuai untuk seni bina memori yang dikongsi. Berbanding dengan kaedah pendekatan selari tahap kepingan,

cadangan kaedah tahap jubin mencapai kelajuan yang lebih sebanyak 14%, pengurangan PSNR sebanyak 67% dan pengurangan kadar bit sebanyak 56.5%.

TILE-LEVEL PARALLELISM FOR H.264/AVC CODEC USING PARALLEL DOMAIN DECOMPOSITION ALGORITHM ON SHARED MEMORY ARCHITECTURE

ABSTRACT

The theme of this thesis is based on the utilisation of features of the parallel model in the design phase of an algorithm in order to reduce the computational complexity in comparison with the serial algorithm. By assuming that parallel architectures are forming the vast majority of computing nodes in digital devises, proposing inherently-parallel algorithms are no more an overstatement. In this work, the process or thread identification is used in a mathematical formulation to decompose a one-, two, and a three-dimensional domain. Then, the solution of the scenario of two-dimensional space is further customized to serve as a new level of parallelism for the H.264/AVC coding standard due to the higher computational complexity of this video coding in comparison with previous standards. This new level of parallelism for the H.264/AVC encoder has been designed in a way to consider several video coding and parallel- oriented metrics. As a further step, the proposed tile-level parallel H.264/AVC is compared with the slice-level and the macroblock-level parallel approaches. Comparisons are made with regards to the baseline implementation (serial). Speedup, parallel efficiency, bitrate, and peak-signal-to-noise-ratio (PSNR) are used as metrics for all of the approaches. Empirical results of the proposed tile-level parallel H.264/AVC encoder outperformed the slice-level which is previously known to be the most suitable approach for shared memory architecture. In comparison with the slice-level parallel encoder, the proposed tile-

level method achieved speedup of 14%, PSNR reduction of 67% and bit rate reduction of 56.5%.

CHAPTER 1

INTRODUCTION

1.1 Introduction

Due to the ongoing revolution in the digital era, the spectrum of multimedia services is continuously expanding. Digital video, for instance, is being used in a wide range of applications areas including education and training, leisure and entertainments, virtual reality and simulations, and many more. The delivered quality of these digital video applications over networks, such as the Internet, relies on the advances in the computing and communication technologies as well as the efficiency of the video compression algorithms (Woods, 2012).

With an increasing number of multimedia services and a growing popularity of high definition HD video contents and beyond, the need for a video compression standard, capable of a higher coding efficiency by occupying lower data rates became evident. The standard MPEG-4 Part10 (also known as H.264/AVC) has, compared to former standards, achieved such a need (Dhanani and Parker, 2013). The H.264/AVC is one of the most popular video codec today (Ozer, 2015). The standard H.264/AVC is an outcome of a joint work made by the Moving Picture Experts Group (MPEG) and the Video Coding Experts Group (VCEG) (Ostermann et al., 2004). When compared against previous standards, the H.264/AVC is capable of encoding a video sequence with higher quality using the same data rate or same quality using significant lower data rate due to the utilisation of sophisticated techniques that reduce video size by combining prediction, transform coding, and statistical compression (Puri et al., 2004). As a consequence, the computational

complexity of the encoder has grown significantly in comparison with older standards. Thus, encoding real time video, with 30 frames per second (fps), using standard quality options and normal resolution has become very difficult to achieve with traditional uniprocessor platforms.

Due to the high computational complexity of the H.264/AVC video encoding H.264/AVC standard, complexity reduction algorithms as well as parallel computing approaches have been employed to lessen the encoding time of the uncompressed videos (Horowitz et al., 2003; Choi and Jang, 2012). As the names imply, complexity reduction algorithms are done by skipping or by early terminating of some of the encoding features of the video compression algorithms which could subjectively deemed as redundant. As a result, the complexity of video compression will be reduced. On the other hand, the parallel computing approaches are fulfilled on a video codec, such as H.264/AVC standards, by the simultaneous processing of the standard's video compression components using a number of computing resources.

However, the preference of the parallelised video encoding algorithms over the complexity reduction video encoding algorithms is possible to be emphasised by the wide-spreading of parallel models such as shared, distributed, and data-parallel memory models. Therefore, the use of parallel computing has become no more optional but actually a necessity for resource demanded applications such as video coding (Chi et al., 2012).

Parallel video encoder implementations, however, are expected to take advantage of the full potential of the parallel hardware architectures. Yet, special care must be taken that a parallelised video encoder does not compromise low encoding delay, quality of video, accuracy of bit rate control, and error resilience due to modifications introduced by the parallelisation approach. Thus, there is a strong demand for research work addressing design and implementation issues related to low latency and high quality parallel video encoding (Lehtoranta, 2007).

1.2 Parallel Approaches of the H.264/AVC, a Brief

In general, parallel video codecs are possible to be categorised according to the flavours of the parallel computing itself. Hence, data (domain) and task (functional) decompositions are two types of video coding parallelisation methods. In the following sections (**section 1.2.1** and **section 1.2.2**), a brief review to the H.264/AVC video codec parallelisation techniques based on these two types is presented.

1.2.1 Task-Level Approach

In the task-level approach, the functional stages of the video compression are assigned to different processing units at the same time. Thus, these stages have to be independent in order to achieve parallelism. However, pipelining is an alternative, but less efficient approach (Feng et al., 2009), among dependent stages. Generally, Task-level decomposition requires significant communication between tasks in order to move the data from one processing stage to the other, and this could become a performance bottleneck.

However, in terms of the H.264/AVC standard, the main drawbacks of task-level decomposition are the non-scalability and load imbalance (Jo et al., 2012) Scalability is hard to achieve in terms of H.264/AVC due to limited number of independent tasks, wherein the different computational load of each task results in load imbalance among processing node. Moreover, in terms of H.264/AVC, pipelining is hard to achieve scalability as the number of stages is limited to few.

1.2.2 Data-Level Approach

Generally, video sequences can be expressed as a series of two dimensional arrays where each frame is one single two dimensional array (Choi and Jang, 2012). In a holistic view, paralleling H.264/AVC encoder based on the data-level approach has featured into different types based on the relative size of the parallel unit (see **Figure. 1.1**). From coarsest to the finest, Group-of-Picture (shortly GOP-level), frame-level, slice-level, macroblock-level (shortly MB-level), and block-level, are different possible granularities that can be chosen to parallelise H.264/AVC encoder (Fan, 2012).

Typically, GOPs are used for synchronisation purposes because there are no temporal dependencies among them. Each GOP is composed of a set of frames. These frames are possibly having temporal dependencies based on their types due to the motion prediction among frames. Each frame is further divided up into one or more slices. The slice is a standalone unit for encoding and decoding and there are no spatial dependencies between slices. Moreover, each slice is further composed of a set of MBs. MBs are the basic units of prediction. H.264/AVC allows variable sizes

of each MB. Additionally, MBs are composed of few blocks wherein each block is composed of picture samples, and these samples can be processed in parallel.

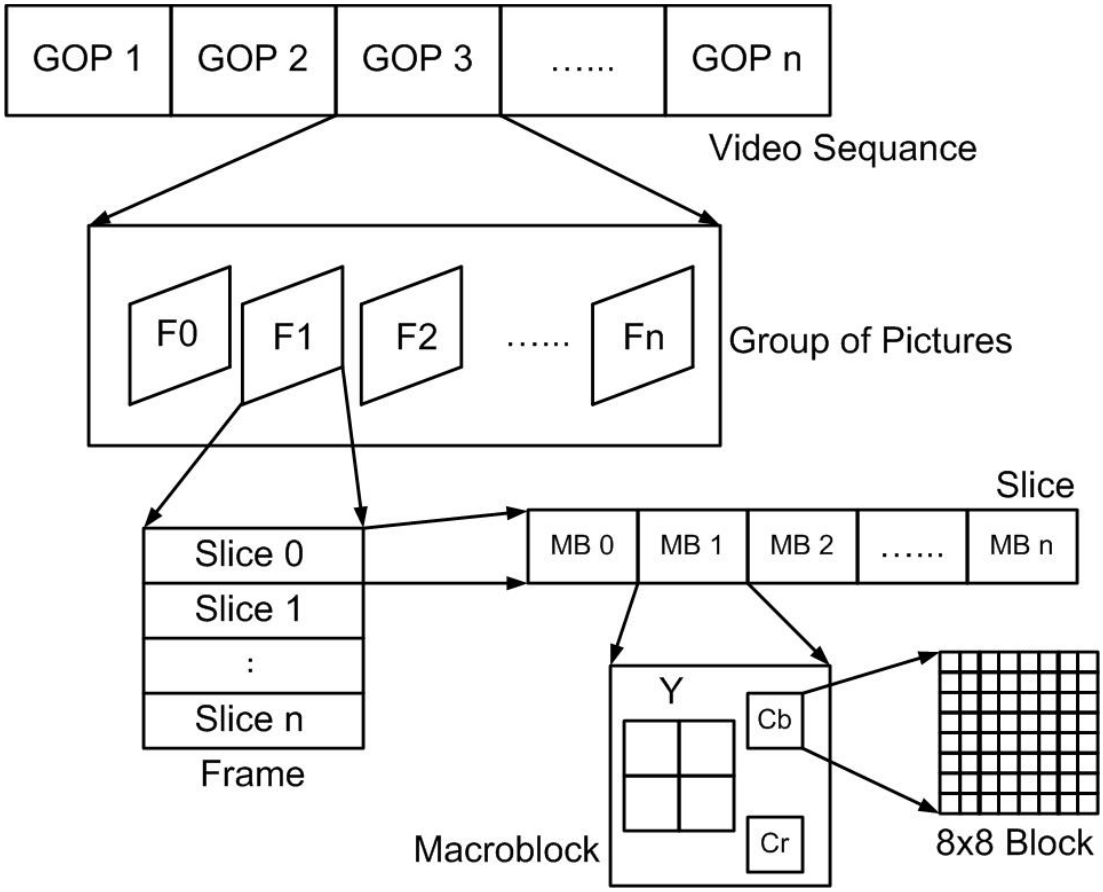

Figure 1.1: H.264/AVC data structure (Gu and Sun, 2011)

GOPs are a coding-independent unit. Therefore, the GOP level is easy to implement; however, it has long latency (Fernandez and Malumbres, 2002) and large memory requirements (Jo et al., 2012). Thus, paralleling the GOP level is inappropriate for shared memory architecture because of limited on-chip memory (Zrida et al., 2011). Frame-level coding does not increase bit rate. However, the complex interdependencies in the H.264/AVC standard, which are caused by very flexible usage of reference pictures, limit its parallel scalability (Yen-Kuang et al., 2004; Jung and Jeon, 2008; Roitzsch, 2007b). Moreover, this level of coding is

associated with large memory requirements. Slice-level coding has been associated with minimal synchronization cost, normal memory requirements, and good performance scalability (Jo et al., 2012). The only drawbacks associated with this level are the increasing bit rate and degradation of visual quality when the number of slices increases (Yen-Kuang et al., 2004). MB-level and block-level coding incur no bit rate degradation; nevertheless, both are associated with high synchronization costs because of the small-sized parallel unit, dependency among them (Lili et al., 2012), and poor scalability (Jo et al., 2012), which render them incompatible with the current trend of multicore.

In general, each one of these granularities has different constraints, would be suitable for particular platform, and could require different parallelisation methodologies.

1.3 Problem Statement

Technically, for the same video sequence, the H.264/AVC encoder requires computations that are about one order of magnitude more compared to previous video encoding standards and about two to four times more computations compared to earlier video decoding standard, due to the higher computations of its inter and intra prediction processes (Saponara et al., 2004; Tu et al., 2012). This remarkable increase has motivated the adoption of parallelism.

However, due to the diversity of the parallel memory models such as shared, distributed, and data-parallel, it is clear now, that the hardware models should play a decisive role in the decision making of the suitable parallel methodology for a video

codec. For instance, the co-exploration between algorithm and architecture (CEAA) (Choi and Jang, 2012), is a new trend in computing which takes into consideration the architecture features during the design phase of an algorithm in order to significantly utilise the full potential of that architecture with no or minimum compromise on the purpose of the algorithm. The H.264/AVC lacks to adopt such a trend. However, the upcoming high efficiency video coding (HEVC) standard has addressed such a remark by its support to parallelisation, but because of the *scope of standard* (bit stream and the decoder processes) addressing such a limitation with a new video compression algorithm cannot be directly backward compatible and a dedicated research work need to be conducted. Moreover, instant moving to a newer video coding standard is not always possible. This explains the existence of several video transcoders (Peixoto and Izquierdo, 2012; Peixoto et al., 2013; Shen et al., 2013), which implicitly motivate research works as the one in this thesis regardless of the technologies achieved with other video coding.

Considering the shared memory architecture from a CEAA point of view for current and new algorithms is vital, due to the considerable horizontal scaling in the number of cores per a single processing die as well as its affordability and wide-spread. In terms of parallel efficiency, the slice-level parallelism is the most suitable level for such architecture and it is a trade-off level based on its associated granularity. Moreover, it is the most universal parallelisation method employed to parallelise the H.264/AVC codec (Lili et al., 2012). Hence, it is the preference selection for parallelism for this parallel architecture. Unfortunately, this level, with respect to its parallel suitability, has been associated with few limitations. Technically, increase in bit rate, degradation in visual quality, and possibly load

imbalance are noticed upon the employment of this parallel method (Franche and Coulombe, 2012).

Considering the above, the introduction of an alternative level which has the same suitability level as the slice-level but with fewer disadvantages is remained as unanswered question for the H.264/AVC standard.

1.4 Motivation

Unfortunately, as it is happening for the parallel slice-level H.264/AVC encoder, lessening the complexity of algorithms by using parallelism has not to be at the expense of the purpose of these algorithms. This drawback has to be strongly avoided if the expense is getting higher as the number of parallel partitions is increased. However, as we have entered the multicore era, ignoring parallel computing as an effective solution in computing cannot be overlooked. Thus, a new bridge need to be established in a way that ensures a better utilisation of parallel computing along with no or little expense to the purpose of the algorithms seeking parallelisation.

1.5 Research Objectives

1. To design a natively parallel domain decomposition algorithm for uniform multi-dimensional domains.
2. To customise the proposed algorithm to serve as a new level of parallelism for the H.264/AVC tailored for shared memory architectures, which would has the same parallel suitability as the slice-level approach, but with minimal penalties on the bit rate and the visual quality.

3. To validate and evaluate the proposed parallel approach for the H.264/AVC video coding.

1.6 Thesis Contribution

In order to achieve the objectives of this thesis, a new method to seamlessly employ parallel computing in encoding videos using H.264/AVC without compromising the objectives that this video codec was designed for is proposed. In particular, a new parallel granularity is proposed in this thesis. This new granularity is based on decomposing the video frame using a 2D domain decomposition algorithm instead of the 1D domain decomposition method that is typically used in the slice-level parallelism.

Adding to the proposed parallel granularity, named tile-level, is a new data-level parallelism in terms of H.264/AVC codec; the algorithmic design is also new. We have considered using the facilitation of the parallel libraries in the design phase of the algorithm (natively-parallel) rather than making the utilisation as an optimisation or a post stage since parallel hardware is forming the vast majority, if not all, of the processing units in all digital devises. This enrolment has significantly simplified the mathematical formula for the proposed 2D domain decomposition algorithm when compared to other general-purpose 2D domain decomposition algorithms.

Further, a generalised mathematical modelling has been made to the algorithmic design of the proposed parallel domain decomposition algorithm.

Finally, real implementation test results for the proposed mathematical-based method show improvement in the parallel-oriented metric such as speedup. Moreover, on the same pace, video quality metrics has been remarkably improved compared to the slice-level parallelism on shared memory architectures.

1.7 Scope and Limitations

In the folds of this work, the scope is identified from the architecture and software perspectives. In terms of architecture, shared-memory architecture has been selected as an environment, while other types of parallel architecture have not been considered. Further, the H.264/AVC is selected as an example of array-based application for the new designed algorithm.

1.8 Thesis Organisation

This thesis is organised into seven chapters. The content is arranged to emphasise the flow of the presented knowledge. In **Chapter 1 (Introduction)**, a brief introduction to the thesis's area of research, problem statement, objectives, motivation, and the thesis contribution are stated.

In **Chapter 2 (Background and Literature Review),** a general background to the video compression is given. This is followed by detailed description of the H.264/AVC video encoding components. Then, an informational background to the parallel computing was also added. Finally a review of previous works that have been done on making video encoding faster, by using parallelism, is presented at the end of the chapter.

In **Chapter 3 (Methodology),** the research procedure of this study is described. **Chapter 3** is the part where the research framework, the used data set, and the experimental environments are specified.

Chapter 4 (Natively-Parallel Domain Decomposition Algorithm) is discussing the algorithmic description of the proposed natively-parallel domain decomposition algorithm. In particular, it shows how this algorithm was designed and defines the assumptions behind the introduction of this algorithm. Moreover, a proper logical testing for the algorithm to inspect it correctness is also made.

As the parallel algorithm proposed in **Chapter 4** is general in its purpose, application-oriented customisations to the new algorithm to serve as a new parallel-level for the H.264/AVC encoder are presented in **Chapter 5** (**The Parallel Tile-Level H.264/AVC Encoder**) in order to cope with this video coding standard. Issues related on how the tiles are added to the standard as a new syntax element have also been given.

Chapter 6 (Implementations, Results and Evaluations) is where the implementation of the parallel H.264/AVC tile-level and its evaluation with regard to the serial and other H.264/AVC parallel encoders is placed.

Finally, **Chapter 7 (Conclusion and Future Works)** covers the conclusions of the thesis, as well as recommendations for further research directions.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

2.1 Introduction

This chapter introduces some of the background information related to the areas of digital video and video compression. Colour spaces, underlying motivation for video compression, measuring the quality of compressed videos, and the theme behind hybrid video coding algorithms are presented and explained. Moreover, the standard H.264/AVC, as an example of hybrid video coding and as a part of the work presented in this thesis is explained in detail. A part, a background of parallel computing is found to be necessary followed by a presentation and a discussion of several related works that have adopted parallel computing to lessen the complexity of the H.264/AVC standard.

2.2 Digital Video

Digital video refers to the capturing, manipulation, and storing of moving images that can be displayed on computer screens. This requires that the moving images be digitally handled by the computer. The word digital refers to a system based on discontinuous events (sampling), as opposed to analogue, a continuous event. Visual pixels are the basic unit in digital video, where each colour component sorted digitally in each pixel.

Visual information at each sample pixel is representing by the values of three basic colour components: Red (R), Green (G), and Blue (B). This is called the RGB colour space (Tkalcic and Tasic, 2003). Each value is stored in a few bits number.

For example, an 8-bit number can store 256 levels to represent each colour component. In the RGB colour space, the light intensity (luminance) of each component is stored correspondingly in each of the three colour components. However, it has been proved that the human visual system (HVS) has less sensitivity to colour information (chromosomes) than luminance information. Therefore, with the separation of luminance (aka luma) information from the chromosomes (aka chroma) information, it is possible to represent chromosomes information with a less resolution than the luminance information, and hence less number of bits will be needed to represent each pixel.

This separation has been achieved by the introduction of the YCrCb colour space (Tkalcic and Tasic, 2003). YCrCb is another widely used colour space to represent digital visual contents. The luminance component 'Y' is extracted using mathematical equations of the three colour components R, G and B. The components Cr and Cb are the chrominance (or colour difference) components. Cr is the red chrominance component and the blue chrominance component is Cb. H.264/AVC standard uses the YCrCb colour space.

2.3 The Importance of Video Compression

Image and video compression is an area with much ongoing research. New demands for higher quality and higher resolution video have increased the needs for better compression. One reason is that bandwidth capacity has not scaled with the new demands for HD-video. In order to better understand how huge the data rate of videos of different resolution; two standard video resolutions are compared. In the standard definition (SD, 720 x 480) video size, the uncompressed form (raw) size of

this resolution of one second requires number of bits that can be obtained by multiplying the dimension of frame, frame rate, and the bits per pixel all together. By proposing the frame rate to be 30 frame per second (fps), and the bits per pixel is 16 (YCrCb format) the uncompressed size will be equalled to 16 x 720 x 480 x 30 = 165888000 bits (for one second video clip only), which is approximately equivalent to 158 megabits. The total size of one hour video will be equal to 568800 megabits, which approximately equals to 555 gigabits (about 69 GB) of storage. However, when the resolution further increases to Full HD (1920 x 1080), the uncompressed video requires about 949 megabits to play one second of video. An hour of video using this resolution requires about 3336 gigabits (about 417 GB).

Considering current standards, the average user has nowhere near this kind of storage space for watching neither a full length movie nor enough bandwidth to stream such size of videos across Internet in uncompressed form. Thus, with the continuing trend towards higher resolution and higher quality video, compression is still needed, perhaps more than ever.

2.4 Block-Based Hybrid Video Coding

In the block-based hybrid video coding (see **Figure 2.1**) that uses the YCrCb colour space, the basic unit of coding are blocks of *n x n* (e.g. 16 x 16) array size of luma sample and corresponding chroma samples. The frame is divided into a number of blocks based on the size of the frame and processed sequentially in raster scan order (Jian-Wen et al., 2006). It is hybrid video coding because the particular video coding system involves prediction as well as transformation stages. Generally, in such video coding system, the encoder has two data flow paths; forward and reverse.

The forward path represents the encoding process of coding units and the reverse path (decoder path) shows the decoding (reconstruction) of the coded units within the encoder that used for motion estimation. In general, major components of block based encoding are inter and intra prediction, transformation and quantisation, and entropy coding processes. The entropy coding process produces the bit stream which can be used for transmission or storage.

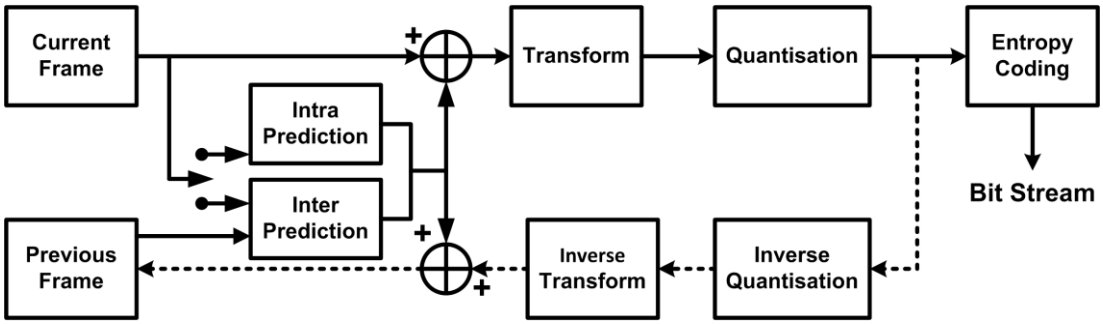

Figure 2.1: Block-based hybrid encoder block diagram (Ziyi et al., 2011)

2.5 The H.264/AVC Standard

Similar to several formers video encoders, the H.264/AVC video encoder carries out prediction, transformation and entropy encoding processes to produce a compressed H.264/AVC bit stream. While the H.264/AVC video decoder carries out the complementary processes of entropy decoding, inverse transformation and reconstruction to produce a decoded playable video sequence. Better compression efficiency and network-friendliness were the two goals behind the introduction of the H.264/AVC (Zrida et al., 2009).

Although H.264/AVC has similar coding features captured from earlier video coding standards, it has also introduced several new features such as variable block size, multiple reference frames and quarter-pixel accuracy (Jian-Wen et al., 2006).

These main processes along with other features of the H.264/AVC video coding standard are explained in the sections **(2.6.1-2.6.5)**. Moreover, the profiles and levels of the H.264/AVC, the common dependencies of the standard, and an overview to a number of industry and academic-based H.264/AVC software are stated in sections (**2.6.6**-**2.6.8**) respectively.

2.5.1 Prediction

The prediction in the encoder is formed of a current MB based on previously-coded MBs, either from the same frame (intra prediction) or from other previously coded frames (inter prediction). The encoder subtracts the prediction from the current MB to form a residual.

Prediction models of the H.264/AVC are more sophisticated compared to previous video coding standards as they enable accurate prediction. In terms of intra prediction, two block sizes are supported: 16 x 16 and 4 x 4 to predict the MB from surrounding and previously coded MBs within the same frame. For the 4 x 4 intra prediction, nine different prediction modes are supported while four different prediction modes are supported for the 16 x 16 block (Ostermann et al., 2004).

On the other hand, more variable block sizes are supported for the inter prediction mode, 16 x 16, 16 x 8, ….., down to 4 x 4 block sizes can be used to predict current MB from similar regions of previously coded MBs of previous coded frames (You and Jeong, 2010). It is worth mentioning that the coding MB in inter prediction mode can be predicted from a frame which is after the current frame in terms of the display order. This leads that in terms of H.264/AVC the display order

differs from the encoding order of frames. **Figure 2.2** shows all of the possible modes for inter and intra predictions supported by the H.264/AVC standard (Wu et al., 2013). In terms of the computational complexity, this process is empirically proved to occupy most of the computation of the H.264/AVC video encoder (Milicevic and Bojkovic, 2011a).

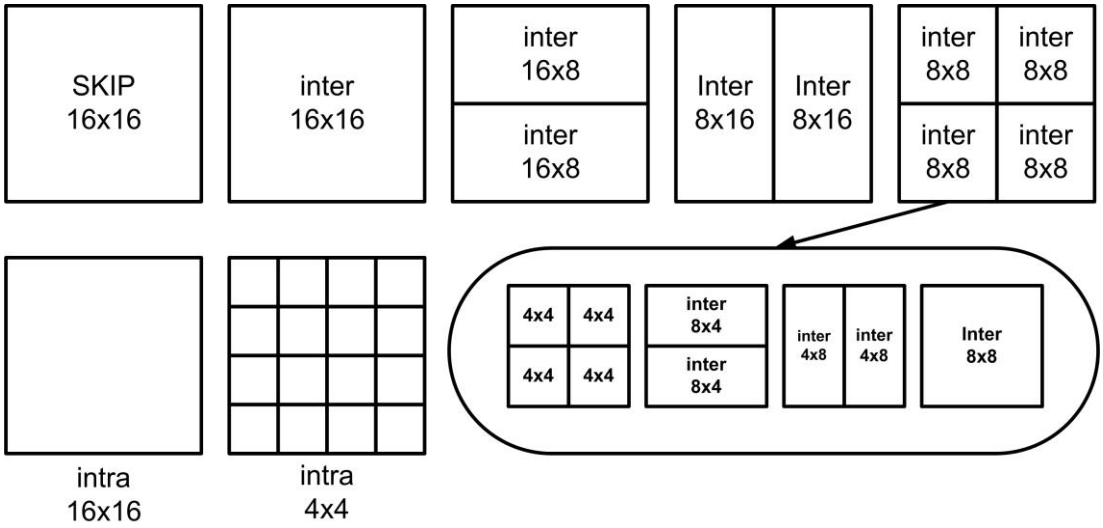

Figure 2.2: Variant block sizes of inter & intra predictions

2.5.2 Transform and Quantisation

A block of residual samples is then transformed using a 4 × 4 integer transform (8 x 8 in limited scenarios), which is a simplified version of the well-known Discrete Cosine Transform (DCT) (Ahmed et al., 1974) used in most of the former video coding standards. The transform results in a set of coefficients, each of which is a weighting value for a standard basis pattern. When combined, the weighted basis patterns recreate the block of residual sample. **Figure 2.3** shows the 4x4 transformation matrix mostly used by the H.264/AVC standard.

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}$$

Figure 2.3: 4 x 4 H.264/AVC transform matrix

Each block of transform coefficients, is further quantised. Quantisation divides the transform coefficients by an integer value (0-51). To achieve the targeted bitrate, this step reduces the precision of the transform coefficients according to a quantisation parameter (QP). However, in general, the higher value of QP, the better compression efficiency, but the poorer image quality (lower bit rate). While selecting lower value of QP, leads to better image quality (high bit rate) but also less efficiency in compression.

2.5.3 Rate-Distortion Optimisation (RDO)

The RDO is a technique of improving video quality during the video compression. This method refers to the optimisation of the amount of distortion (the loss of video quality) to the amount of data required to encode the video (the video rate) (Li-Chuan et al., 2011). The typical method of making encoding decisions for the video encoder is to choose the result which yields to the highest quality output image. However, this selection would be associated with a disadvantage represented by more bits while giving comparatively little quality benefit. For instance, in motion estimation, adding the extra precision (half and quarter pixel accuracy) to the motion

of a block during motion estimation might increase quality. But in some cases that extra quality isn't worth the extra bits necessary to encode the motion vector to a higher precision. Hence, the role of RDO would be probably to neglect such further precision and settle for normal pixel accuracy. It is worth mentioning that RDO works at the MB-level and examine the candidates with regard to their availability to the encoder (Zhou and Yuan, 2012).

2.5.4 Deblocking Filter

Deblocking Filter plays a vital role in block-based video coding systems. Since The H.264/AVC video coding standard uses blocks DCT-like coding techniques, this propagates blocking artefacts. Blocking artefacts can be defined as discontinuities occurring at the block boundaries. Hence, in such scenario, it is preferable to eliminate as much as possible of such visual annoying artefact at the boundaries of the MBs to enhance the quality of the video. H.264/AVC applies the deblocking filter at the encoder and decoder sides. The filtering is done first from left to right vertically and then from top to bottom on the horizontal boundaries of each block. Moreover, luma and chroma components are separately processed. The deblocking filter of the H.264/AVC can achieve substantial objective and subjective quality improvements (Choi and Ho, 2008). **Figure 2.4** illustrates the process of deblocking filter which ordered alphabetically.

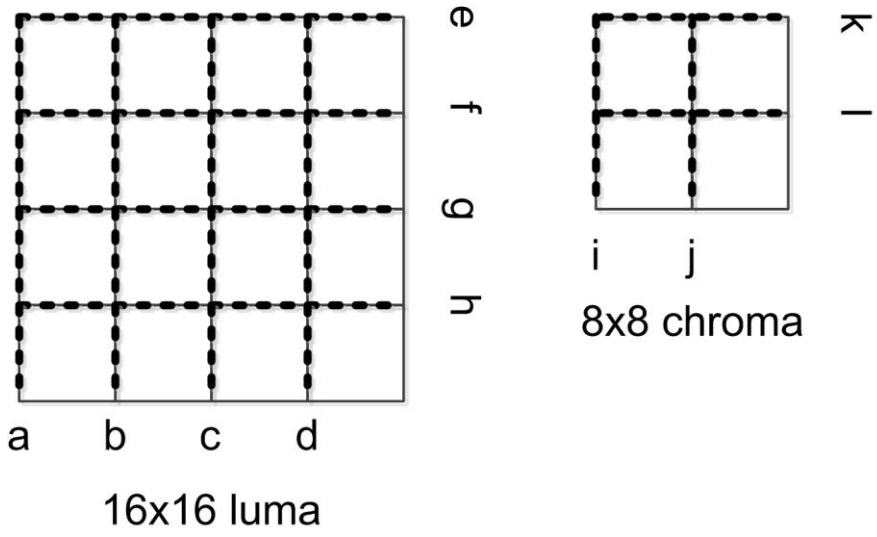

Figure 2.4: Horizontal and vertical edges filtering in a MB

2.5.5 Entropy Coding

The video coding process produces a number of values that must be encoded to form the compressed bit stream. These values along with the encoding parameters and the syntax elements are converted into binary codes using variable length coding or arithmetic coding. Each of these encoding methods produces an efficient, compact binary representation of the information for transmission or storage. H.264/AVC uses two methods of entropy coding: a low-complexity technique based on the usage of context-adaptively switched sets of variable length codes (CAVLC), and the computationally more demanding algorithm of context-based adaptive binary arithmetic coding (CABAC) (Zhan et al., 2008). Both methods represent major improvements in terms of coding efficiency compared to the techniques of statistical coding that are traditionally used in former video coding standards (Ostermann et al., 2004). This stage is typically known to be serial, as it scans in raster-scan order in a frame basis.

2.5.6 Profiles and Levels of the H.264/AVC Standard

In the first release of the H.264/AVC standard, three profiles were defined; **baseline**, **main** and **extended** profile. A year later another group of profiles have been also introduced to form a total of seven profiles. The H.264/AVC intended to serve wide range of multimedia applications on numerous architectures. Moreover, in each profile there are levels to specify options and tools to suit a particular multimedia application on specific architecture such as maximum stored frames, maximum frame size and maximum video bit rate.

It is important to know that not all of the encoding/decoding features are supported in all profiles. As an example, the baseline profile which targets real-time conversational services does not support B-frame in its coding process. Additionally, as stated in the previous section, H.264/AVC defines two schemes of entropy coding which are Context-Adaptive Binary Arithmetic Coding (CABAC) and Context-Adaptive Variable Length Coding (CAVLC). These two schemes differ in terms of the complexity-performance trade-off. The CAVLC is associated with lower computational complexity than CABAC (Tung et al., 2012; Sze and Chandrakasan, 2012). Hence it is the preferable choice for real-time applications (Wiegand et al., 2003).

2.5.7 Dependencies of H.264/AVC Standard

When the H.264/AVC was designed, there was no realistic consideration to natively support parallelism. This limitation explains the various types of dependencies in H.264/AVC codec. This section discusses these dependencies with

regard to the different possible data-level parallel units discussed previously (**section 1.2.2**) and the encoding stages.

Since GOPs are the coarsest syntax element to the H.264/AVC standard, there is no data dependency between GOPs. However, with some exceptions, it is possible that a GOP can depend on a previous GOP. For example, slices of a key picture can be either intra predicted or inter predicted according to the H.264/AVC standard (Hsu-Feng and Chen-Tsang, 2013). If a key picture is inter-coded, its reference frame will be from the previous GOP. Therefore, the two GOPs are no longer coded independently. However, to eliminate the possible errors from such a dependency over error-prone networks, key pictures are usually intra predicted (Hsu-Feng and Chen-Tsang, 2013).

In terms of frames, the frame type determines the level of dependency. I-frame is like a conventional static compressed image. P-frame is a predicted frame which holds only the changes in the image from the previous frame, while B-frame is a bidirectional predicted frame where this picture holds only the changes from previous and successive frames. However, due to the flexibility of selecting the reference frame and the various possible sequencing of I, P and B frames in each GOP (see **Figure 2.5** & **Figure 2.6**), the dependency may vary from one video sequence to another (Franche and Coulombe, 2012).

.

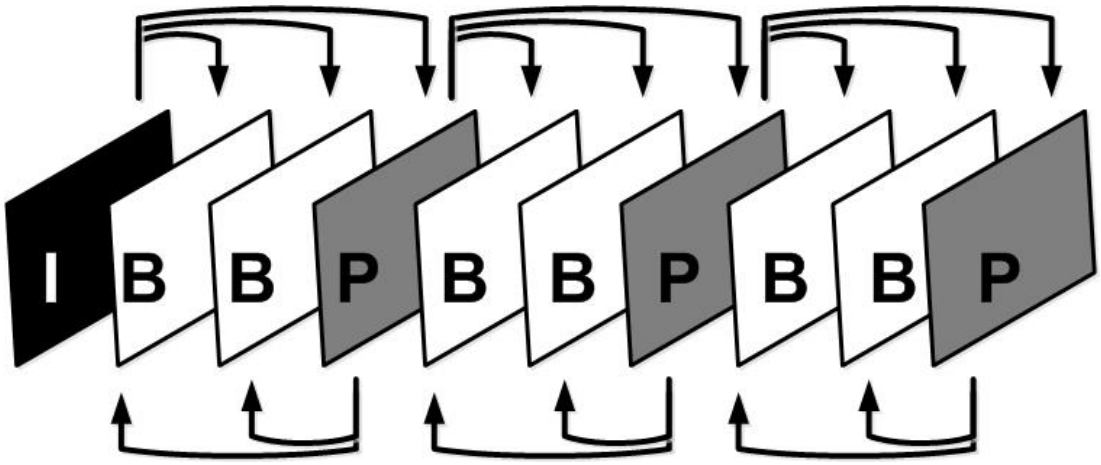

Figure 2.5: Inter frame dependency (IBBPBBP sequence)

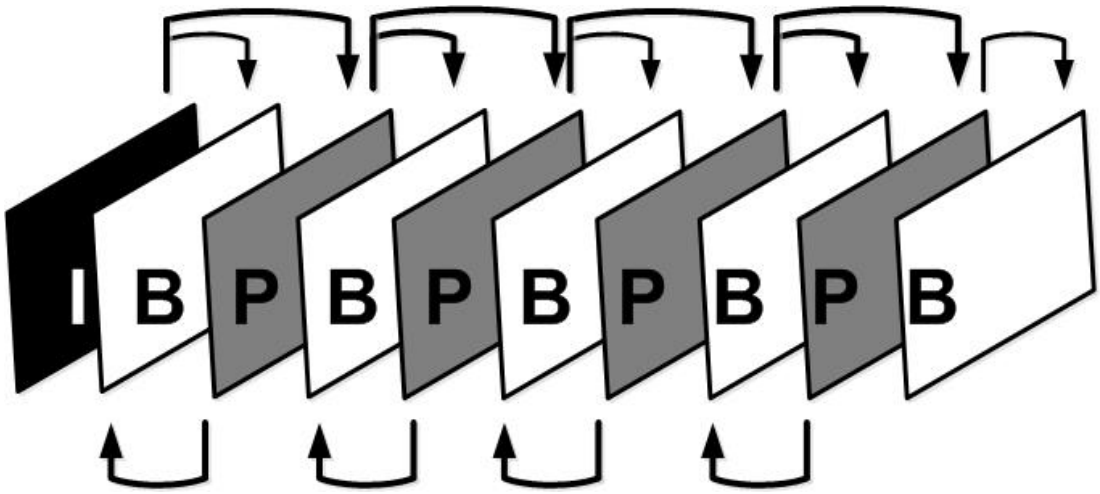

Figure 2.6: Inter frame dependency (IBPBP sequence)

Typically, slices of the same frame are coded independently. However, slices boundaries are subject for deblocking filter (Hiremath, 2010). Although this feature is optional, adopting it will incur dependency among slices. **Figure 2.7** shows the optional deblocking filter across the slice boundaries (dashed lines).

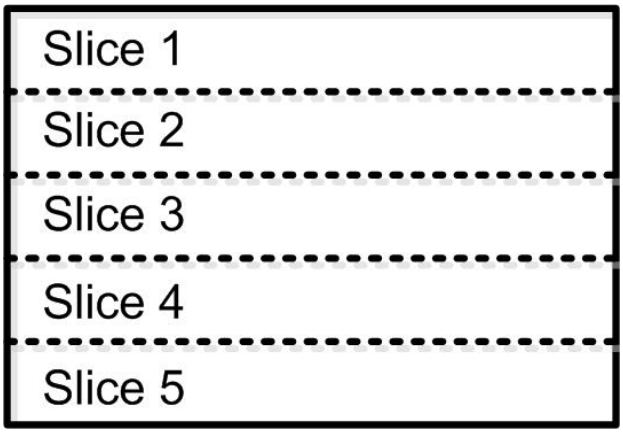

Figure 2.7: Deblocking filter across the slice boundaries

Dependencies at finer levels (MBs and blocks), are numerous. In terms of intra prediction, several modes of intra dependencies are required to predict an MB or a block in a frame. At the same pace, inter dependency of MB can be predicted by motion vectors of the same regions of prior coded frames. Moreover, deblocking filter is applied at the MB boundaries of flat areas of the image and at the block boundaries of the image for more detailed areas. In addition, the numbers of the supported prediction modes in H.264/AVC standards are numerous. Hence, the dependencies are varying from one mode to another.

Finally, the entropy coding stage is applied at the MB-level in a raster-scan order on a frame bases. This scan ordering limits the chance of parallelism accumulatively. Hence, in several studies, such as (Jo et al., 2012), this encoding stage is excluded from the parallelised loop.

2.5.8 Common H.264/AVC Coding Software

Currently, there are several available software programmes to encode/decode videos by using the H.264/AVC standard. Among these several solutions, three