UNIVERSIDADE DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

High performance and scalable unified architectures for transform and quantization in H.264/AVC codecs

Tiago Miguel Braga da Silva Dias

Supervisor: Doctor Nuno Filipe Valentim Roma
Co-Supervisor: Doctor Leonel Augusto Pires Seabra de Sousa

Thesis approved in public session to obtain the PhD Degree in
Electrical and Computer Engineering

Jury final classification: Pass with Merit

Jury

Chairperson: Chairman of the IST Scientific Board

Members of the Committee:

Doctor Horácio Cláudio de Campos Neto
Doctor Sebastian López Suárez
Doctor João Paulo de Castro Canas Ferreira
Doctor Nuno Filipe Valentim Roma
Doctor João Miguel Duarte Ascenso

2015

UNIVERSIDADE DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

High performance and scalable unified architectures for transform and quantization in H.264/AVC codecs

Tiago Miguel Braga da Silva Dias

Supervisor: Doctor Nuno Filipe Valentim Roma
Co-Supervisor: Doctor Leonel Augusto Pires Seabra de Sousa

Thesis approved in public session to obtain the PhD Degree in
Electrical and Computer Engineering

Jury final classification: Pass with Merit

Jury

Chairperson: Chairman of the IST Scientific Board

Members of the Committee:

Doctor Horácio Cláudio de Campos Neto, Professor Associado (com Agregação), Instituto Superior Técnico – Universidade de Lisboa

Doctor Sebastian López Suárez, Professor Associado, Instituto Universitario de Microelectrónica Aplicada – Universidad de Las Palmas de Gran Canaria, Espanha

Doctor João Paulo de Castro Canas Ferreira, Professor Auxiliar, Faculdade de Engenharia – Universidade do Porto

Doctor Nuno Filipe Valentim Roma, Professor Auxiliar, Instituto Superior Técnico – Universidade de Lisboa

Doctor João Miguel Duarte Ascenso, Professor Auxiliar, Instituto Superior Técnico – Universidade de Lisboa

2015

To mom, dad and Anabela,
for their love and unconditional support,
and to my son André,
for the final encouragement
to conclude this thesis.

Abstract

The H.264/AVC standard adopted improved transform and quantization procedures to enhance the compression performance. Such techniques also greatly increase the computational burden and the data processing rate requirements of video codecs, posing additional challenges when designing such systems. These issues are particularly relevant when high definition video contents must be processed or when real time operation is demanded.

To face these challenges for multiple application domains, this thesis addresses the design of efficient hardware structures for the fast computation of the H.264/AVC transform and quantization operations. First, a new high performance and scalable multi-transform architecture capable of supporting various sets of transforms, including the whole H.264/AVC transformation procedure, is proposed. Then, a novel class of high performance architectures with a reduced hardware cost is presented for the realization of H.264/AVC forward, inverse and unified quantizers. Finally, it is presented an integrated transform and quantization architecture that enables the combined and autonomous computation of all the H.264/AVC transform and quantization procedures.

The experimental evaluation conducted using a Xilinx Virtex-7 FPGA demonstrates the superior performance and hardware efficiency of the proposed architectures in comparison with the state of the art, which allow the processing of video sequences with resolutions up to the 4k UHDTV format in real time with a reduced hardware cost.

Keywords

Video coding, H.264/AVC standard, Discrete Cosine Transform, Quantization, Real time and embedded systems, Adaptable architecture, Systolic array, FPGA.

Resumo

A norma de vídeo H.264/AVC emprega novas técnicas de codificação de transformada e de quantização para aumentar ainda mais os seus fatores de compressão. A complexidade computacional e o ritmo de processamento destas ferramentas são elevados, o que impõe restrições adicionais no desenvolvimento de sistemas de codificação de vídeo. Esta problemática assume especial relevância no processamento de conteúdos de elevada definição ou em tempo real.

Para dar uma resposta efetiva a estes desafios, nesta tese apresentam-se estruturas de hardware especializadas e de elevado desempenho que permitem realizar estas duas operações da norma H.264/AVC, de uma forma rápida e eficiente, para diversos domínios da codificação de vídeo. Primeiramente, propõe-se uma arquitetura multi-transformada escalável que é capaz de suportar diferentes conjuntos de transformadas bidimensionais, incluindo todas as transformadas adotadas pela norma. Depois, apresenta-se uma classe de arquiteturas com custos de hardware reduzidos para a realização de quantizadores diretos, inversos e unificados. Finalmente, introduz-se uma arquitetura integrada de transformada e quantização que permite combinar e realizar de uma forma autónoma todas as operações de codificação de transformada e de quantização definidas na norma.

Os resultados experimentais relativos à implementação destas arquiteturas numa FPGA Virtex-7 da Xilinx comprovam os seus elevados níveis de desempenho e taxas de utilização de hardware, que possibilitam a sua utilização no processamento em tempo real de conteúdos vídeo com resoluções até ao formato 4k UHDTV.

Palavras-chave

Codificação de vídeo, Norma de vídeo H.264/AVC, Transformada Discreta do Co-seno, Quantização, Sistemas embebidos e de tempo real, Arquitetura configurável, Processador sistólico, FPGA.

Acknowledgments

The research work presented in this thesis is the result of several years of intense and hard work, to which several people and institutions have contributed and to whom I owe my most sincere gratitude and appreciation.

First of all, I want to thank my two supervisors, Prof. Nuno Roma and Prof. Leonel Sousa, for giving me the opportunity to perform this work. I am most grateful not only for their guidance, advices and support but also for their encouragement, infinite patience and friendship. It has been a privilege to learn and work with them!

I would also like to express my gratitude to Prof. Sebastian López for his confidence and the constant availability to help me when conducting a very important part of this work. His enthusiasm and motivation have definitely helped me pursuing this work. Thank you for your kind friendship!

A very special acknowledgement also to Nuno Sebastião, Artur Ferreira, Rui Ramalho, Mário Carrasqueira Simões, Pedro Sampaio and Pedro Miguéns for their friendship and constant encouragement, as well as for the many constructive and enlightened discussions and all the helpful suggestions, which altogether revealed to be most valuable contributions to improve this thesis.

I am also grateful to my colleagues at Instituto de Engenharia de Sistemas e Computadores: Investigação e Desenvolvimento em Lisboa (INESC-ID) for providing me a really nice and friendly working environment. I must extend this appreciation to the División de Diseño de Sistemas Integrados of the Instituto Universitario de Microelectrónica Aplicada (IUMA) at Universidad de Las Palmas de Gran Canaria (ULPGC), with a special thanks to Teresa Cervero and Prof. Roberto Sarmiento for all their generous assistance and kind friendship during my two stays in Las Palmas de Gran Canaria (LPGC). It was great to have the possibility to work with all of you!

I want to address a word of appreciation also to the following institutions, from which I received important support to perform this thesis:

- To INESC-ID, for providing me with the necessary working conditions along all these years;
- To both Instituto Politécnico de Lisboa (IPL) and Instituto Superior de Engenharia de Lisboa (ISEL), for their commitment in this work and for awarding me a Programa de apoio à formação avançada de docentes do Ensino Superior Politécnico (PROTEC) grant, with the

reference number SFRH/PROTEC/50152/2009, which supported this work from September 2009 to October 2012;

- To the European Cooperation in Science and Technology (COST) Action IC805 (Open Network for High-Performance Computing on Complex Environments), which provided the funding required to support my stay in LPGC in June 2012, under the Short Term Scientific Mission IC0805-010612-019048;
- To the 7th Framework Program of the HiPEAC European Network of Excellence, that financially supported my stay in LPGC from September 2010 to December 2010, under the Collaboration Grant EU ICT-217068;
- To the Fundação para a Ciência e a Tecnologia (FCT), for the financial support provided by the *Philosophiae Doctor* (PhD) grant with the reference number SFRH/BD/43639/2008 that was given to me from September 2008 to September 2009.

I must also express my deepest gratitude to my brother and to my dearest friends Mário Neves, Pedro Miranda and Rui Coutinho for their invaluable support and companionship during one of the most difficult periods of my life. Without your friendship this work would not have been possible!

Last but not least, I want to express my eternal gratitude to my parents and to my beloved Anabela for their constant support, encouragement, dedication, and unconditional love. A very special THANK YOU to Anabela for showing up in my life and bringing a whole new meaning and a much brighter future to it.

A final word goes to my son André, whose pregnancy and birth provided me the final encouragement to conclude this thesis.

I dedicate this thesis to you four!

Tiago Miguel Dias

July 31st, 2014

Contents

Abstract iii

Resumo v

Acknowledgments vii

Contents ix

List of Figures xiii

List of Tables xvii

List of Acronyms xxi

1 Introduction 1

1.1 A brief review of digital video coding 4

1.1.1 Exploiting data redundancy 5

1.1.2 Picture types 8

1.1.3 Typical architectures of video codecs 10

1.2 Overview of the H.264/AVC standard 12

1.2.1 H.264/AVC data flow and encoding procedures 12

1.2.2 The compression *vs* complexity dilemma 18

1.3 Motivation and objectives 21

1.4 Original contributions 24

1.5 Outline 27

2 Transform coding 29

2.1 Fundamentals of transform coding 30

2.1.1 Transform 31

2.1.2 Quantization 35

2.2 Transform coding in the H.264/Advanced Video Coding (AVC) standard 38

2.2.1 Hierarchical transformation procedure 38

2.2.2 Quantization procedure 40
2.3 Summary 45

3 State of the Art and Related Work **47**
3.1 Transform architectures 48
3.1.1 Dedicated Transform Architectures 48
3.1.2 Multi-Transform Architectures 55
3.1.3 Discussion 63
3.2 Architectures for quantization 66
3.2.1 Standard architectures for quantization 67
3.2.2 Optimized architectures for quantization 71
3.2.3 Discussion 75
3.3 Summary 75

4 Scalable multi-transform architecture **79**
4.1 Mapping of the transform algorithms into a systolic array 81
4.2 Proposed hardware structure 84
4.2.1 Transform array and Processing Elements (PEs) 85
4.2.2 PE architecture for the H.264/AVC standard 89
4.2.3 Transposition switch 91
4.2.4 Input buffer 92
4.2.5 Control unit 93
4.3 Dataflow 96
4.4 Scalability and parallelism 98
4.4.1 Scalability 99
4.4.2 Coarse-grain data-level parallelism 103
4.5 Summary 107

5 Configurable and low cost quantization architectures **109**
5.1 Forward Quantization Architecture 110
5.2 Inverse Quantization Architecture 112
5.3 Unified Quantization Architecture 114
5.4 Summary 118

6 Experimental evaluation **119**
6.1 Experimental setup and implementation considerations 120
6.2 Evaluation of the proposed MTA 122
6.2.1 FPGA implementation results 123
6.2.2 Comparative analysis and discussion 127

6.3 Evaluation of the proposed quantization architectures . . . 131
6.3.1 FPGA implementation results . . . 131
6.3.2 Comparative analysis and discussion . . . 134
6.4 Generalized design of integrated transform and quantization circuits . . . 138
6.5 Summary . . . 146

7 Conclusions **149**
7.1 Thesis summary and conclusions . . . 150
7.2 Future research directions . . . 152
7.3 List of publications . . . 153

Bibliography **157**

A Transform cores for other relevant digital video standards **171**
A.1 Overview of the VC-1, AVS and H.265/HEVC transform coding procedures . . . 172
A.1.1 VC-1 . . . 172
A.1.2 AVS . . . 173
A.1.3 H.265/HEVC . . . 174
A.2 PE architectures for the VC-1, AVS and H.265/HEVC standards . . . 175
A.2.1 PE for the VC-1 standard . . . 176
A.2.2 PE for the AVS standard . . . 178
A.2.3 PE for the H.265/HEVC standard . . . 179
A.3 Resource-shared architecture of the multi-standard PE . . . 182

B Test Conditions **187**

List of Figures

1.1 **Two neighbouring pictures in the *table tennis* video sequence.** 5

1.2 **Prediction error resulting from the encoding of the picture presented in Figure 1.1(b).** 7

1.3 **Example of inter-dependence among I-, P- and B-pictures in a video sequence.** 9

1.4 **Example of switching between two bit streams through SP-pictures.** 10

1.5 **Typical architecture of a video encoder.** 11

1.6 **Typical architecture of a video decoder.** 12

1.7 **Prediction directions for the $Intra_{4\times4}$ mode.** 14

1.8 **Motion-compensated prediction with multiple reference frames.** 15

1.9 **MB and block partitioning for MC prediction.** 16

1.10 **Processing rate requirements of an H.264/AVC codec for real time operation.** 23

1.11 **Computational rate requirements of an H.264/AVC codec for real time operation.** The amount of operations is specified as Reduced Instruction Set Computer (RISC)-like operations[7]. 23

1.12 **Bandwidth requirements of an H.264/AVC codec for real time operation.** . . . 23

2.1 **Transform-based picture coding procedure.** 30

2.2 8×8 **Discrete Cosine Transform (DCT) basis functions.** 34

2.3 **Generic architecture of a quantizer.** 35

2.4 **Input-output characteristic of a uniform quantizer.** 37

2.5 **Hierarchical transform paths defined in the H.264/AVC standard.** 39

2.6 **Reconstruction of the residue values.** 43

3.1 **Signal flow graph of the 8-points DCT (extracted from[92]).** 49

3.2 **Block diagram of the architecture proposed by Fan (extracted from[45]).** . . . 51

3.3 **two-dimensional (2-D) transform architectures based on the row-column decomposition.** 51

3.4 **Block diagram of the architecture proposed by Gu *et al* (extracted from[49]).** . 53

3.5 **Block diagram of the architecture proposed by Agostini *et al* (extracted from[1]).** 54

3.6 **Block diagram of the architecture proposed by Lo *et al* (extracted from[98]).** . 55

3.7 **Block diagram of the architecture proposed by Ho *et al* (extracted from [50]).** . 56
3.8 **Block diagram of the architecture proposed by Chen *et al* (extracted from [13]).** 58
3.9 **Block diagram of the architecture proposed by Chang *et al* (extracted from [11]).** 60
3.10 **Block diagram of the architecture proposed by Wang *et al* (extracted from [139]).** 62
3.11 **Block diagram of the architecture proposed by Kordasiewicz *et al* (extracted from [79]).** 68
3.12 **Block diagram of the architecture proposed by Husemann *et al* (extracted from [57]).** 69
3.13 **Block diagram of the architecture proposed by Lee *et al* (extracted from [87]).** 70
3.14 **Block diagram of the architecture proposed by Lin *et al* (extracted from [94]).** . 70
3.15 **Block diagram of the architecture proposed by Lee *et al* (extracted from [86]).** 71
3.16 **Block diagram of the architecture proposed by Peng *et al* (extracted from [117]).** 72
3.17 **Block diagram of the architecture proposed by Tran *et al* (extracted from [131]).** 73
3.18 **Block diagram of the architecture proposed by Ying *et al* (extracted from [86]).** 74

4.1 **Dependency Graphs (DGs) for the computation of the row-wise and column-wise transforms.** 82
4.2 **Signal Flow Graph (SFG) for the computation of a one-dimensional (1-D) transform.** 83
4.3 **Block diagram of a generic multi-transform PE.** 83
4.4 **Block diagram of the proposed Multi-Transform Architecture (MTA).** 84
4.5 **Block diagram of the devised PEs for the H.264/AVC standard.** 85
4.6 **Definition of a transform kernel value.** Definition of the $-a$ value located at position $(2, 2)$ of the generic 4×4 transform kernel shown in Equation 4.6, according to the proposed algorithm. 87
4.7 **Directed Acyclic Graph (DAG) of the mux-MCM used in the arithmetic module of the H.264/AVC PE.** 90
4.8 **Architecture of the time-multiplexed MCM (mux-MCM) used in the arithmetic module of the H.264/AVC PE.** 90
4.9 **Memory map of the ROM used in the H.264/AVC PE.** 91
4.10 **Block diagram of the devised Transposition Switch (TS).** 92
4.11 **Block diagram of the developed Input Buffer (IB) Parallel-In-Serial-Out (PISO) First-In-First-Outs (FIFOs).** 93
4.12 **Block diagram of the developed Control Unit (CU).** 94
4.13 **Algorithmic State Machine (ASM) chart [20] of the Control Unit's state machine.** 95
4.14 **Dataflow in a TA with $N \times N$ PEs for the processing of $N \times N$ data blocks.** . . 97
4.15 **Reprogramming of a TA with 4×4 PEs.** 98
4.16 **Alternative setups for the proposed MTA.** 101

4.17 **Architecture of the circular buffers.** 102
4.18 **Architecture of the Variable Delay Element (VDE).** 102
4.19 **Dataflow in a 8×8 TA for the simultaneous processing of four 2×2 data blocks.** The two data-sets depicted using a solid-line concern the residue values (or the transform coefficients, when inverse transforms are considered), while the data-sets represented using a dashed-line consist of the intermediate values of the row-column decomposition. 104
4.20 **Architecture of a transform core supporting coarse-grain data-level parallelism.** 105

5.1 **Block diagram of the proposed FQA.** 111
5.2 **Block diagram of the proposed IQA.** 113
5.3 **Block diagram of the proposed UQA.** 116

6.1 **Performance comparison of the several implemented transform cores, when operated at their maximum clock frequency.** For each transform core, the figure shows the amount of macroblocks (MBs) that can be processed when using the $Intra_{16\times16}$ prediction mode (lowest value), the default coding mode (middle value) and the 8×8 transforms (highest value). 125
6.2 **Performance comparison of the several implemented quantizers, when operated at their maximum clock frequency.** 133
6.3 **Comparison of the hardware cost of the several implemented quantizers.** . . 134
6.4 **Block diagram of the considered integrated transform and quantization circuit for the H.264/AVC standard.** 139
6.5 **Processing paths supported by the considered integrated transform and quantization circuit.** (Continued on the next page.) 142
6.5 **Processing paths supported by the considered integrated transform and quantization circuit.** 143
6.6 **Theoretical performance assessment of the considered integrated transform and quantization circuit.** 146

A.1 **DAG of the mux-MCM used in the arithmetic module of the VC-1 PE.** 176
A.2 **Architecture of the mux-MCM used in the arithmetic module of the VC-1 PE.** . 177
A.3 **Memory map of the ROM used in the VC-1 PE.** 177
A.4 **DAG of the mux-MCM used in the arithmetic module of the AVS PE.** 178
A.5 **Architecture of the mux-MCM used in the arithmetic module of the AVS PE.** . 179
A.6 **Memory map of the ROM used in the Audio Video coding Standard (AVS) PE.** 179
A.7 **DAG of the mux-MCM used in the arithmetic module of the H.265/High Efficiency Video Coding (HEVC) PE.** 180

A.8 **Architecture of the mux-MCM used in the arithmetic module of the H.265/HEVC PE.** 181
A.9 **Memory map of the ROM used in the H.265/HEVC PE.** 181
A.10 **DAG of the mux-MCM used in the arithmetic module of the multi-standard PE.** 183
A.11 **Architecture of the mux-MCM used in the arithmetic module of the multi-standard PE.** 183
A.12 **Memory map of the ROM used in the multi-standard PE.** 185

B.1 **Considered benchmark standard test video sequences.** 189

List of Tables

1.1 **Overview of the most important coding tools used in the Baseline Profile (BP), Main Profile (MP), Extended Profile (XP) and High Profiles (HiPs) of the H.264/AVC standard.** . . . 13

2.1 **Definition of the values for $m_4(QP, n)$.** . . . 42
2.2 **Definition of the values for $m_8(QP, n)$.** . . . 42
2.3 **Definition of the values for $\nu_4(QP, n)$.** . . . 44
2.4 **Definition of the values for $\nu_8(QP, n)$.** . . . 44

3.1 **Summary of the most relevant designs that have been presented for transform computation.** The letters *F* and *I* preceding the transform names in the column *Supported transform(s)* are used to denote a forward or an inverse transform, respectively. . . . 64
3.2 **Summary of the most relevant quantization architectures that have been presented in the literature.** In the column *Supported operation(s)*, *FQ* and *IQ* denote the forward and the inverse quantization procedures, respectively. . . . 76

4.1 **Encoding of the `Type_T` signal for the implementation of the proposed H.264/AVC multi-transform architecture.** . . . 86
4.2 **Multiplier control words for the H.264/AVC transform kernels (see Equations 2.12, 2.13, 2.14, 2.15 and 2.17).** . . . 90
4.3 **List of the most relevant signals for the operation of the CU's state machine.** 96

5.1 **Possible configurations of the proposed FQA.** . . . 112
5.2 **Functionality of the block η.** . . . 114
5.3 **Possible configurations of the proposed IQA.** . . . 114
5.4 **Functionality of the μ block.** . . . 117
5.5 **Possible configurations of the proposed UQA.** . . . 117

6.1 **Characteristics of the considered proof of concept transform cores.** . . . 121
6.2 **Characteristics of the considered proof of concept quantizers.** . . . 121

6.3 **Implementation results of the proof of concept $T_{8\times 8}$ transform core in a Xilinx Virtex-7 XC7VX485T-2FFG1761C FPGA device.** . . . 123
6.4 **Implementation results of the proof of concept $T_{8\times 4}$ transform core in a Xilinx Virtex-7 XC7VX485T-2FFG1761C FPGA device.** . . . 124
6.5 **Implementation results of the proof of concept $T_{8\times 2}$ transform core in a Xilinx Virtex-7 XC7VX485T-2FFG1761C FPGA device.** . . . 124
6.6 **Implementation results of the proof of concept $T_{8\times 8p}$ transform core in a Xilinx Virtex-7 XC7VX485T-2FFG1761C FPGA device.** . . . 126
6.7 **Comparison with other related transform architectures implemented in Field-Programmable Gate Array (FPGA) devices.** The letters *F* and *I* preceding the transform names in the column *Supported Transforms* are used to denote a forward or an inverse transform, respectively. The hardware cost is assessed in terms of the number of required Virtex slices for implementations based on Xilinx FPGA devices and in terms of Logic Elements (LEs) for the hardware realisations based on Altera FPGAs. . . . 129
6.8 **Comparison with other related transform architectures implemented as Application Specific Integrated Circuits (ASICs).** The letters *F* and *I* preceding the transform names in the column *Supported Transforms* are used to denote a forward or an inverse transform, respectively. The hardware cost is assessed in terms of the number of two input `NAND` gate equivalent gates. . . . 130
6.9 **Implementation results of the considered proof of concept quantizers in a Xilinx Virtex-7 XC7VX485T-2FFG1761C FPGA device.** . . . 132
6.10 **Comparison with other related quantization architectures implemented in FPGA devices.** . . . 135
6.11 **Comparison with other related quantization architectures implemented as ASICs.** . . . 135
6.12 **Operations supported by the several configurations of the considered integrated transform and quantization circuit for the H.264/AVC standard.** . . . 140
6.13 **Amount of Clock Cycles (CCs) that are required to encode one MB using the considered integrated transform and quantization circuit.** . . . 145

A.1 **Coefficients of the VC-1 transform kernels.** . . . 173
A.2 **Coefficients of the AVS transform kernels.** . . . 173
A.3 **Coefficients of the H.265/HEVC transform kernels.** . . . 174
A.4 **Encoding of the `Type_T` signal for the implementation of the VC-1 PE.** . . . 176
A.5 **Multiplier control words for the VC-1 transform kernels (see Equation 4.5, Equation 4.6 and Table A.1).** . . . 177
A.6 **Encoding of the `Type_T` signal for the implementation of the AVS PE.** . . . 178

A.7 **Multiplier control words for the AVS transform kernels (see Equation 4.5, Equation 4.6 and Table A.2).** 179
A.8 **Encoding of the `Type_T` signal for the implementation of the H.265/HEVC PE.** 180
A.9 **Multiplier control words for the H.265/HEVC DCT kernels (see[134], Equation 4.5, Equation 4.6 and Table A.3).** 181
A.10 **Encoding of the `Type_T` signal for the implementation of the multi-standard PE.** 182
A.11 **Multiplier control words for the H.264/AVC, VC-1, AVS and H.265/HEVC DCT kernels (see Equations 4.5, 4.6, 2.12, 2.13, 2.14, 2.15 and 2.17, Tables A.3, A.1, A.2 and A.3 and[134]).** 184

List of Acronyms

The acronyms most frequently used in this thesis are as follows.

1-D	one-dimensional
2-D	two-dimensional
3-D	three-dimensional
ASIC	Application Specific Integrated Circuit
ASIP	Application Specific Instruction Set Processor
ASM	Algorithmic State Machine
AVC	Advanced Video Coding
AVS	Audio Video coding Standard
BP	Baseline Profile
CABAC	Context-Adaptive Binary Coding
CAVLC	Context-Adaptive VLC
CC	Clock Cycle
CIF	Common Intermediate Format
CMOS	Complementary Metal-Oxide Semiconductor
CORDIC	Coordinate Rotation Digital Computer
COST	European Cooperation in Science and Technology
CPU	Central Processing Unit
CU	Control Unit
CSDA	Common Sharing Distributed Arithmetic
DA	Distributed Arithmetic

DAG Directed Acyclic Graph

DCI Digital Cinema Initiatives

DCT Discrete Cosine Transform

DFT Discrete Fourier Transform

DG Dependency Graph

DHT Discrete Hadamard Transform

DSP Digital Signal Processor

DST Discrete Sine Transform

DQA Dedicated Quantization Architecture

DST Discrete Sine Transform

DTA Dedicated Transform Architecture

DWT Discrete Wavelet Transform

DTUA Data Throughput per Unit of Area

FCT Fundação para a Ciência e a Tecnologia

FIFO First-In-First-Out

fps frames per second

FMO Flexible Macroblock Ordering

FPGA Field-Programmable Gate Array

FQA Forward Quantization Architecture

FRExt Fidelity Range Extension

FS Factor Sharing

GOP Group of Pictures

GOPS Giga Operations Per Second

GPP General-Purpose Processor

GPU Graphics Processing Unit

HD High Definition

HDTV HD Television

HEVC High Efficiency Video Coding

HiP High Profile

HVS Human Visual System

IB Input Buffer

IC Integrated Circuit

IEC International Electrotechnical Commission

INESC-ID Instituto de Engenharia de Sistemas e Computadores: Investigação e Desenvolvimento em Lisboa

IP Intellectual Property

IPL Instituto Politécnico de Lisboa

IQA Inverse Quantization Architecture

ISDN Integrated Services Digital Network

ISEL Instituto Superior de Engenharia de Lisboa

ISO International Organization for Standardization

ITQA Integrated Transform and Quantization Architecture

ITU International Telecommunications Union

ITU-T Telecommunication Standardization Sector of the ITU

IUMA Instituto Universitario de Microelectrónica Aplicada

KLT Karhunen-Loève Transform

LPGC Las Palmas de Gran Canaria

LE Logic Element

LTE Long Term Evolution

LUT Look-Up Table

MAC multiply-and-accumulate

MB macroblock

MC Motion Compensation

MCM	Multiple Constant Multiplier
ME	Motion Estimation
MP	Main Profile
MPEG	Moving Pictures Experts Group
MSc	Master of Science
MST	Multi-Standard Transform
MTA	Multi-Transform Architecture
mux-MCM	time-multiplexed MCM
MV	Motion Vector
MVC	Multi-View Video Coding
NAL	Network Adaptation Layer
NEDA	New DA
PE	Processing Element
pel	picture element
PF	Pre-scaling Factor
PIT	Pre-Scaled Integer Transform
PhD	*Philosophiae Doctor*
PISO	Parallel-In-Serial-Out
PROTEC	Programa de apoio à formação avançada de docentes do Ensino Superior Politécnico
PSNR	Peak Signal-to-Noise Ratio
QFHD	Quad Full HD
$Qstep$	Quantization Step
QP	Quantization Parameter
QE	Quantization Engine
RAM	Random Access Memory
RISC	Reduced Instruction Set Computer

RNS Residue Number System

ROM Read Only Memory

S/s Samples per Second

SD Standard Definition

SF Scaling Factor

SFG Signal Flow Graph

SMTPE Society of Motion Picture and Television Engineers

SoC System-on-Chip

SVC Scalable Video Coding

TA Transform Array

TE Transform Engine

TS Transposition Switch

TU Transform Unit

TV Television

UHDTV Ultra High Definition Television

ULPGC Universidad de Las Palmas de Gran Canaria

UMTS Universal Mobile Telecommunications System

UQA Unified Quantization Architecture

UVLC Universal VLC

UTA Unified Transform Architecture

VCEG Video Coding Experts Group

VCL Video Coding Layer

VDE Variable Delay Element

VHDL VHSIC Hardware Description Language

VHSIC Very High Speed Integrated Circuit

VLC Variable Length Code

VLSI Very Large Scale Integration

XP Extended Profile

1

Introduction

Contents

1.1 A brief review of digital video coding 4
1.2 Overview of the H.264/AVC standard 12
1.3 Motivation and objectives 21
1.4 Original contributions 24
1.5 Outline 27

The technological progresses that were achieved in the last decade, both for Very Large Scale Integration (VLSI) and network access technologies, have allowed the development of several new and different services and applications that people for long had been awaiting, most of them dealing with multimedia content. In this scope, video coding has widespread through multiple applications and networks, as a result of the constantly growing user demand for more, innovative and better quality multimedia services and equipment. These applications range from conversational services, such as bidirectional and real time videotelephony or videoconferencing, to non-conversational services, like broadcast of Standard Definition (SD) and High Definition (HD) Television (TV), video streaming over the Internet, delivery of high definition DVD and Blu-ray contents, video surveillance and the highest quality video for digital cinemas, which all together cover a wide range of bit rates and video resolutions. Simultaneously, the transmission media has diversified from the classical broadcast channels and Integrated Services Digital Networks (ISDNs) to embrace new network access technologies with improved bandwidth, like cable modem, xDSL, Universal Mobile Telecommunications System (UMTS) and 4G Long Term Evolution (LTE). Although many of these new networks offer modest data rates, they also present quite significantly different loss/error robustness characteristics that have been used to support the expansion of multimedia applications and services to the wireless and mobile domains.

One direct consequence of such advances is the recent proliferation of portable and handheld devices, which mostly focus on multimedia services and applications. This includes portable video and audio players and recorders, PDAs and 3G/4G mobile telephones, smartphones and tablets with Internet access, or even video surveillance and telephony equipment to be used over the Internet (i.e. IP cameras and videophones). At the same time, the use of other classes of computational (embedded) systems that target the home consumer and entertainment markets (e.g. DVD, HD-DVD and Blue-ray players or set-top boxes for three-dimensional (3-D), SD and HD digital TV) has also experienced a tremendous growth. However, while the enabling technologies to support efficient implementations of applications for speech, data, text and audio in modern computational systems are already available today, the management of video information represents the ultimate design challenge for these systems, due to its inherently high data rates and storage burdens. To overcome such troubling constraints, new and improved video standards have been proposed along the last few years, such as the H.264/Advanced Video Coding (AVC)[111], Audio Video coding Standard (AVS)[150], VC-1[128] or, more recently, the H.265/High Efficiency Video Coding (HEVC) standard[129]. Among these, H.264/AVC is currently still considered to be the *de facto* standard for digital video applications[1], mostly due to its high coding efficiency and increased flexibility to address multiple application domains.

The H.264/AVC standard was jointly developed by the International Organization for Standard-

[1] Although the first version of the H.265/HEVC standard was completed and published in June 2013, this new and promising technology will take at least 6-8 years to mature and replace H.264/AVC. In fact, AVC is only hitting its stride at the time of writing this thesis, nearly a decade after it was established.

ization (ISO)/International Electrotechnical Commission (IEC) Moving Pictures Experts Group (MPEG) and the Telecommunication Standardization Sector of the ITU (ITU-T) Video Coding Experts Group (VCEG) and its first draft, as an international standard, was approved by the ITU-T in May 2003. This initial version of the standard was designed to provide high quality coding of the video content at very low bit rates, but soon was adapted to address the full range of video applications by means of three amendments: Multi-View Video Coding (MVC), Fidelity Range Extension (FREext)[103] and Scalable Video Coding (SVC)[121]. As a result, the widespread use of the H.264/AVC standard has proved to provide increased coding efficiency in most video coding applications, regarding to prior successful and popular standards. In fact, for the same reconstructed picture quality, it can save up to 65%, 50% and 40% in the bit rate, when compared with that of MPEG-2, H.263++ HLP and MPEG-4 ASP, respectively[111,143]. This improved coding efficiency, which makes H.264/AVC one of the best video coding standards in terms of compression and quality, is owed to the addition of several new features that include, among others, variable block sizes and fractional Motion Estimation (ME) using multiple reference frames, several Intra prediction modes, in-loop de-blocking filtering and context adaptive entropy coding, as it is briefly presented in subsection 1.2.

Despite all these improvements and innovations, H.264/AVC is still a block-based motion compensated transform coding scheme, just like the former ITU-T (H.261[66] and H.263[67]) and the ISO/IEC MPEG video standards (MPEG-1[61], MPEG-2[68] and MPEG-4[62]). However, its computational complexity is much higher when compared with these predecessors, as it is discussed in subsection 1.2.2. In fact, it has been estimated that the complexity of a H.264/AVC encoder is about 5 – 10 times greater than that of a MPEG-4 encoder, while the complexity of the decoder has been estimated to be 2 – 4 times greater than that of a MPEG-4 decoder[18]. To make matters worse, several of the new coding techniques that were introduced in H.264/AVC involve massive amounts of data, therefore requiring a huge memory bandwidth and thus significantly increasing the storage complexity of this standard. It has been reported[93] that such bandwidth requirement can be as high as 528 MB/s or 878 MB/s for the decoding of 4096×2160 contents in the H.264/AVC Baseline Profile (BP) and Main Profile (MP), respectively[2]. All these constraints are also common to the other existing latest generation video standards (i.e. AVS[150], VC-1[128] and H.265/HEVC[129]), since these standards are all based on a similar set (in some cases, just a subset) of the coding tools adopted in H.264/AVC.

Implementing a state-of-the-art real time video codec therefore represents a quite difficult task for system designers, which simultaneously face four distinct challenges: *i*) increase the application performance, in order to exploit all the available coding tools and thus obtain the highest compression possible, while still achieving real time operation; *ii*) reducing the power consumption to provide long time operation, a crucial problem especially in portable and handheld devices; *iii*)

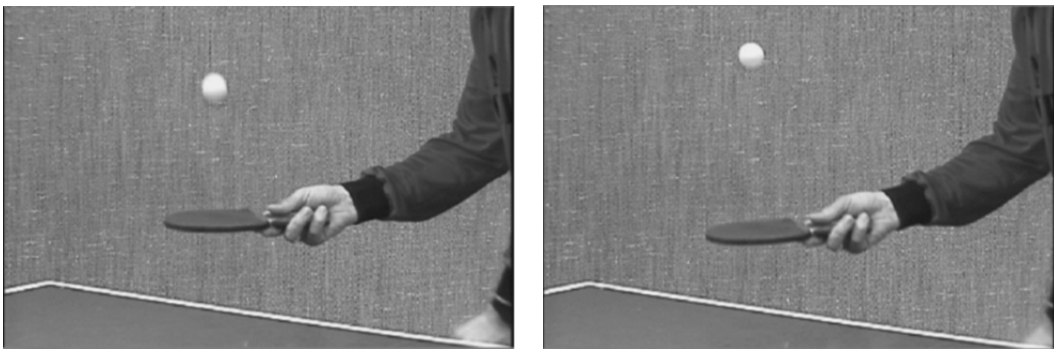
[2]Note that smarthphone devices nowadays already support 2048×1024 @ 30 fps video streams.

reducing the chip area, to optimize the implementation costs; and *iv*) shortening time-to-market metrics, in which the existence of configurable and multi-functionality design solutions are of the utmost importance. To achieve such goals, several different modules of both the video encoding and decoding algorithms are commonly mapped into specialized hardware processing structures, for which it is quite useful to have an in-depth understanding of the video encoding and decoding procedures. Furthermore, it is also very important to successfully exploit the compression efficiency *vs* computational complexity characteristic of the considered video standards. Such approach allows to identify the computational hotspots of the coding procedures that must be accelerated and thus to overcome undesired implementation bottlenecks. In the next subsections, it is presented an overview of all these issues, in order to better introduce video coding and the H.264/AVC standard to the reader.

1.1 A brief review of digital video coding

From the most simplistic point of view, a digital video signal can be seen as a sequence of bi-dimensional pictures that are taken at fixed time intervals. Each of these pictures is composed of a set of points, denominated as pixels, which represent the visual luminescence of the captured scene at that specific picture location. Hence, a monochromatic digital picture can be seen as a two-dimensional (2-D) matrix of pixel values, which are usually encoded using an 8-bits representation. On the other hand, since most polychromatic pictures are composed of three different components, each of them concerning one of the three components of the RGB colour space, i.e. the red (R), green (G) and blue (B) colours, such pictures typically require 24-bits per pixel to represent the scene luminescence. Consequently, by considering a video sequence captured at 30 frames per second (fps) with a spatial resolution of 1920×1080 pixels, which are nowadays common values for digital television (i.e. the HD Television (HDTV) resolution), this implies a signal bit rate of 1.49 Gbits/s, which is quite high for most communication channels. As a result, digital video signals need to be compressed so that they can be used in most practical applications.

The main goal of video coding is therefore to minimize the bit rate of digital video signals, while keeping the picture subjective quality at the highest quality level as possible. Hence, efficient compression techniques must be used in video coding to make most practical applications feasible. Fortunately, video signals are highly amenable to compression, due to two major factors. Firstly, because there is a considerable amount of information in the signal that is irrelevant from the human perceptual point of view. Secondly, due to the high degree of data correlation that exists in these sequences, whether among spatial neighbouring pixels within the same picture (spatial correlation), or among pixels corresponding to different pictures captured at distinct time instants (temporal correlation), as it is illustrated in Figure 1.1. Moreover, the RGB colour space, which is frequently used to capture and display polychromatic video sequences, adds an extra degree of



(a) Picture at time instant t.

(b) Picture at time instant $t + \Delta\ t$.

Figure 1.1: Two neighbouring pictures in the *table tennis* video sequence.

redundant information that can be further exploited to achieve maximum compression.

1.1.1 Exploiting data redundancy

One of the techniques that are commonly first used to reduce the data redundancy in video sequences consists in the transformation of the colour space. Almost all video capturing devices operate using the RGB colour space, in which every colour is defined by means of a linear combination of the red (R), green (G) and blue (B) colours. However, this colour space presents some drawbacks for video coding: the component values are not only redundant but also intensity and application dependent[91]. In order to overcome these disadvantages, the alternative YUV colour space is commonly used in digital video coding. This colour space is obtained from the RGB colour space by using Equation 1.1 [65] (CCIR-601 standard).

$$\begin{cases} Y = 0.299 \times R + 0.587 \times G + 0.114 \times B \\ U = 0.492 \times (B - Y) \\ V = 0.878 \times (R - Y) \end{cases} \tag{1.1}$$

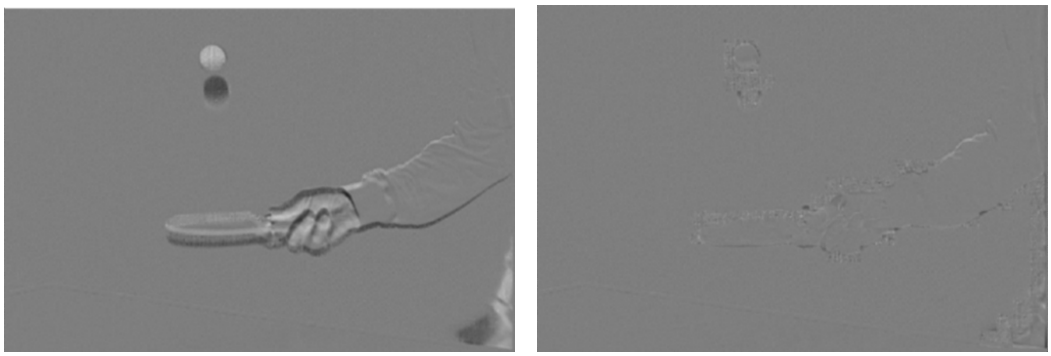
In the YUV colour space, the Y component (also known as gamma-corrected luminance, or simply as luminance) describes the range of gray values between light and dark. On the other hand, the U and V components (known as chrominances) refer to the differences between the red and blue colours and a reference white at the same luminance. These U and V components substantially reduce the amount of redundancy in the colour information, by subtracting the luminance values from the red and blue components of the RBG colour space. Hence, in some extent, this straightforward approach allows to decrease the bit rate of the resulting encoded signal. Moreover, since in this new colour space the luminance information of the video signal is completely uncorrelated to the chrominance information, the inherent limitations of the Human Visual System (HVS) can be better exploited to achieve higher compression ratios. More specifically, due to the HVS being less sensitive to the chrominance information rather than to the luminance information[91], the two chrominance channels can be sub-sampled in both the horizontal and vertical

dimensions, in order to reduce the number of pixels required to represent each picture in the video sequence. Note that by removing this irrelevant information from the video signal, i.e. the one that is imperceptible to the human viewer, the reconstructed data will no longer be identical (in value) to the original data. Nevertheless, the subjective quality of the resulting video sequence is kept mostly unchanged. Consequently, some amount of loss is generally tolerated in the resulting encoded video signal, which is already a result of the application of a lossy compression procedure[7] for the most common digital video applications.

Although there are two distinct classes of lossy compression schemes that can be used to compress image data, i.e. *sample-based coding* and *block-based coding*, it is the later one that is usually exploited in video coding. This is mostly owed to the fact that block-based coding schemes yield far better compression ratios for the same level of distortion than sample-based coding schemes[7]. Hence, several different compression techniques have been developed to operate in a blockwise manner. Such compression techniques are mostly used to implement predictive transform block coding schemes, because the resulting encoders and decoders present similar levels of complexity. In these coding schemes, the pixels from the input picture are grouped into blocks that are first predicted and then transformed into the frequency domain. This allows to obtain a more compact and uncorrelated representation of the data, which can be more efficiently compressed.

According to the established signal processing theory, predictive coding schemes are the most efficient to compress strongly correlated data[110]. In conventional predictive coding, the difference between the current sample and a predicted one, which is based on a previously encoded sample, is computed and encoded. Consequently, the better the prediction the smaller the prediction error, which results in lower bit rates for the encoded signal. In what concerns the encoding of video sequences, the search procedure for the best prediction of a particular pixel in the current picture is quite straightforward for still scenes, i.e. it is the same pixel in the previous picture of the scene. However, finding good predictions in sequences with motion is slightly more complicated, since motion is a major source of temporal variations. In such cases, the displacement of moving objects from one picture to another must be compensated, in order to guarantee the maximum compression in the video coding procedure. Consequently, one can expect that a good prediction for a particular pixel in the current picture will be the pixel that is located at the same area of the moving object in a previous picture. Hence, in order to efficiently encode sequences with motion, not only will the prediction pixels have to be *found* in the previous picture but also the corresponding displacement of moving objects from one picture to another must be *compensated*. These two procedures are usually known as ME and Motion Compensation (MC), respectively.

Following the previous discussion, it can be concluded that a compression process that uses both the motion compensation and the predictive coding techniques can provide optimum compression of video sequences, since the prediction error is minimized. This can be seen in Fig-



(a) Using predictive coding. **(b)** Using predictive coding after MC.

Figure 1.2: Prediction error resulting from the encoding of the picture presented in Figure 1.1(b).

ure 1.2, which presents the prediction error obtained for the encoding of the picture presented in Figure 1.1(b) using only the predictive coding technique (Figure 1.2(a)) and using both techniques (Figure 1.2(b)). Nevertheless, the pictures obtained using these predictive motion compensated schemes still have a high degree of irrelevant information for a human viewer. On the one hand, neighbouring pixels within the same picture have very similar values and thus present a high degree of data correlation. On the other hand, the predicted pictures still have significant amounts of information that are not perceptible to the human viewer. Therefore, by efficiently exploiting the characteristics of the HVS (i.e. its more reduced sensitivity to higher frequency content), even higher compression ratios for digital video signals can be achieved. In this scope, a linear transform of the prediction error is usually computed to obtain much more uncorrelated data values. The resulting data consists of a set of transform coefficients that can be quantized independently of each other and thus more efficiently encoded, by taking into account the perceptual characteristics of the HVS.

Several functions can be used for the transformation of a $N \times N$ image block from the spatial domain to the transform domain[3,7,8], such as the Karhunen-Loève Transform (KLT), the Discrete Fourier Transform (DFT), the Discrete Cosine Transform (DCT), the Discrete Sine Transform (DST), the Discrete Hadamard Transform (DHT), the Discrete Wavelet Transform (DWT) or integer approximations of these transforms[128,129,143,150]. Although the KLT has been proven to be the most efficient one in terms of energy compactness efficiency, this transform is also image dependent, which is a major drawback in image coding. Consequently, an image independent transform, with a performance very close to the KLT has been chosen as the basis for the majority of image and video compression standards: the DCT. As it is more thoroughly discussed in section 2.1, this transform presents several important properties and offers many benefits to video coding schemes.

It is worth noting that the use of a transform, by itself, does not provide any kind of data

compression. In order to compress the data, the transform domain coefficients that are obtained through the use of the transform must be quantized. In this procedure, higher compression ratios can be achieved by using coarser quantization steps. However, such coarser quantization steps may cause the loss of more information, which lowers the quality of the encoded video. Consequently, the characteristics of the HVS should also be taken into account in such quantization procedure. Accordingly, the transform coefficients corresponding to lower frequencies, to which the HVS is much more sensitive, are usually quantized with lower quantization steps. Conversely, the higher frequency transform coefficients are quantized using greater quantization steps. As a result, when these characteristics of the HVS are taken into account, the quantization procedure not only does not introduces a significant degradation in the quality of the encoded video but also provides higher data compression ratios.

By jointly using all the previously described compression techniques, most data redundancies and perceptual irrelevant information can be removed from the video signals, originating digital video signals with much lower bit rates. Nonetheless, the resulting bit rates can still be further reduced by using entropy coding techniques[7]. In these compression schemes, the symbols that are generated by the video encoder to represent the encoded digital video are mapped into codewords, according to the probability model of the signal. These probability models can be defined either from the input image data or from *a priori* assumptions related to such data. Nevertheless, the key idea is to use small codewords for symbols that occur with higher probability and longer codewords for symbols occurring with lower probability. Therefore, by using a Variable Length Code (VLC)[7] to map symbols into codewords, higher compression ratios can be achieved without any degradation of the picture quality. Some examples of the application of VLC techniques to encode digital video signals can be found in the encoding of the coordinates of a Motion Vector (MV), of the DCT coefficients and of the quantization step size.

1.1.2 Picture types

Almost all video standards define several different picture types, which are organized in classes according to the compression techniques that are used to exploit the redundant information contained in those pictures. Each picture may alternatively be split into one or several regions (usually denoted as ”slices”), which represent the basic spatial coding elements of the encoding procedure. Slices consist of groups of block-shaped units of the associated luma (Y) and chroma (U and V) samples called macroblocks (MBs)[3], which can be decoded independently of other slices.

Slices (or whole pictures, when they are composed of a single slice) that are encoded using techniques only exploiting the spatial redundancies, i.e. without referencing to data from any other neighbouring picture, are defined to be of type Intra. Thus, Intra slices (I-slices) only provide

[3]The terms luma and chroma are used herein rather than the terms luminance and chrominance, in order to avoid the implication of the use of linear transfer characteristics that is often associated with such terms.

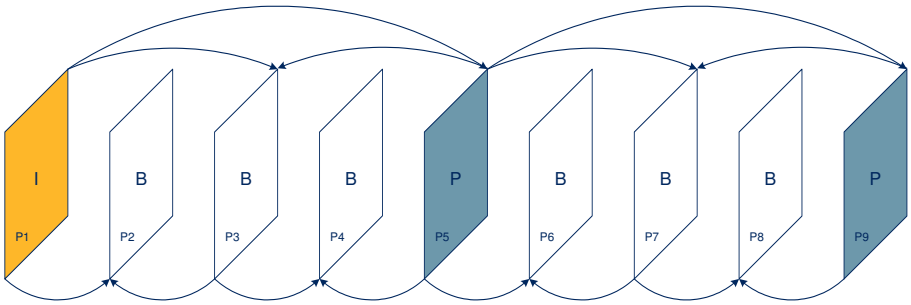


Figure 1.3: Example of inter-dependence among I-, P- and B-pictures in a video sequence.

moderate compression ratios. Nevertheless, pictures compressed using this Intra-frame coding allow for fast random access, which is very important in many digital video applications, such as video recording, editing, interactive television and surveillance systems.

Conversely, a slice is said to be of the Inter type if its encoding procedure employs motion-compensated prediction techniques. Depending on the MC scheme that is applied to encode the data, different compression ratios can be achieved. If the slice is encoded by using data from either past and/or future pictures (B-slices), the compression ratios are higher than those achieved by exploiting a temporal prediction using only data from past pictures (P-slices). Consequently, Inter type slices provide much higher compression ratios than Intra type slices.

Figure 1.3 depicts the relationship between the three main slice/picture types in a video sequence composed of nine pictures, each one consisting only of a single slice. Pictures P5 and P9 are P-pictures whereas P1 is a I-picture. The remaining ones are B-pictures. As it is shown in this figure, P-pictures are predicted using only past I- and P-pictures. In contrast, B-pictures can be predicted using both past and future I- and P-pictures, so that the temporal redundancy can be better exploited. This can be clearly seen in this example, in which P3 is encoded using motion compensation prediction from P1 and P5.

A new type of picture is also defined in the most recent video standards, the so-called switching P and I pictures (SP- and SI-pictures, respectively). This approach allows exact synchronization through replacing I pictures, which enables a decoder to switch between representations of the video content using different data rates (bit stream switching-splicing), recover from data losses or errors, and support *trick* modes (e.g. fast forward and fast reverse) and random access as well. Consequently, SP-pictures make use of motion compensated predictive coding to exploit temporal redundancy in a given sequence (similar to P-pictures). However, they differ from these pictures by allowing identical frames to be reconstructed even when they are predicted using different reference frames. Conversely, SI-pictures use only spatial prediction (just as I-pictures), but still allow to identically reconstruct the corresponding SP-picture.

To better illustrate the use of this type of pictures, a general application scenario is illustrated in Figure 1.4. At time t, S1 and S2 are at a switching point, which is provided for switching from bit stream 1 to bit stream 2 and vice versa. S1, S2 and S12 are compressed as SP-pictures with a

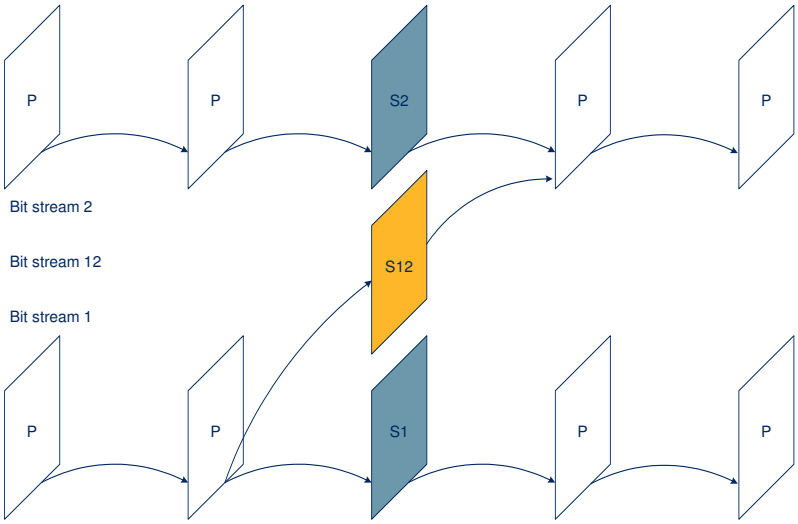


Figure 1.4: Example of switching between two bit streams through SP-pictures.

slight difference. Due to this difference, S1 and S2 are referred to as primary SP-pictures, whereas S12 is referred to as a secondary SP-picture. Assume that bit stream 1 is being transmitted to the user. When there is a switch to bit stream 2, S12 is transmitted at time t instead of S1. By decoding S12, the decoder obtains exactly the same reference as the one obtained by decoding S2 at time t. As a result, it can seamlessly continue the decoding of bit stream 2 at time $t + 1$.

1.1.3 Typical architectures of video codecs

The typical architecture of a video encoder that uses all the coding techniques described in the previous sections is presented in Figure 1.5. It should be noted that since video coding standards only define the syntax and the semantics of the coded bit streams, as well as the video decoding process, manufacturers of video encoders are not compelled to implement any specific architecture and, in particular, the one depicted in Figure 1.5.

As it can be seen, the encoding process usually begins with some pre-processing of the video signal. Colour conversion to the YUV colour space, pre-filtering, sub-sampling and format translation (e.g. from an interlaced to a progressive picture format[91]) are some of the tasks that may be done in this stage. In the next stage of the video encoding procedure, the input picture is split into MBs and the MBs are associated in slices. Furthermore, the encoder also selects the coding type for each input slice at this stage, according to any pre-defined criteria. Hence, a slice can be encoded either as an I-, P- or B-slice.

While for I-slices the encoder directly processes the pixels of each MB by applying the transform to the prediction error that is obtained by considering a prediction block in the same picture, for P- and B-slices the transform is applied to the prediction error resulting from the MC operation. Consequently, for each MB in the current picture, the motion estimator must determine the coordinates of the MB that best matches its characteristics in a search picture. For P-slices, this

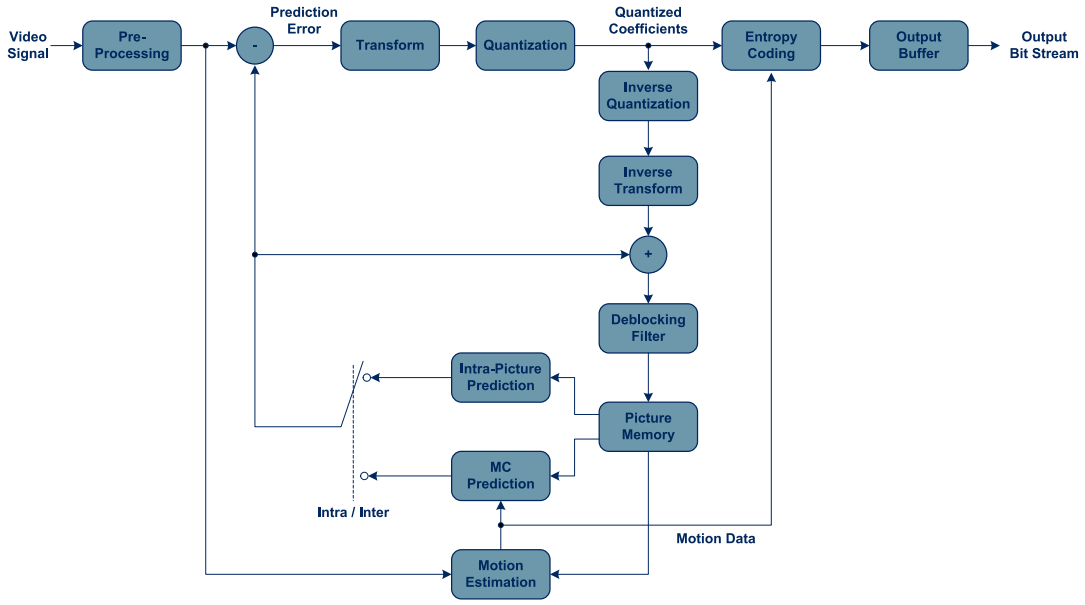


Figure 1.5: Typical architecture of a video encoder.

ME procedure is done using data only from past pictures. In contrast, this process is done twice for B-slices: firstly for a past picture and lastly for a future picture. As a result, the prediction error for P-slices is obtained by using a single candidate block, while for B-slices the prediction error is computed either from one of the two candidate blocks or from their average. After the transform is applied to this prediction error, the obtained transform coefficients are quantized, encoded using entropy coding techniques and stored in the output buffer. Furthermore, in applications that require a constant output bit rate, a buffer regulator is also used to adjust the quantization step that is applied to the transform coefficients, so that only slight variations exist in the output bit rate of the compressed bit stream.

The inherent lossy nature of this compression scheme makes it necessary to embed part of the decoding modules in the encoder architecture, in order to guarantee that the quality of the encoded pictures does not significantly diverge from the original ones. This is implemented by including a feedback loop in the encoder, where the quantized transform coefficients are inverse quantized, inverse transformed and then processed by a deblocking filter to reduce the block-artifacts. As a consequence, a copy of the encoded picture, as seen by the decoder, is used for future predictive coding.

In what concerns the video decoding process, it follows the opposite scheme of the encoder. Therefore, the block diagram of a typical video decoder is very similar to the reconstruction path in the feedback loop of the encoder, as it can be seen in Figure 1.6. First, the decoder applies entropy decoding to the received bit stream and determines the slice type from the header information. Then, the transform coefficients of all the MBs are inverse quantized and transformed into the spatial (pixel) domain, by applying the inverse transform of the one used in the encoder. If the decoded slice is of type I, the reconstructed picture data is directly stored in the output buffer.

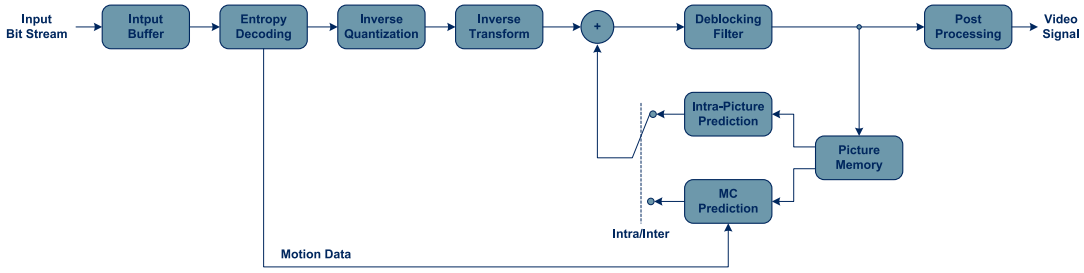


Figure 1.6: Typical architecture of a video decoder.

On the contrary, if the decoded slice is of types P or B, motion compensation must be performed before this operation can be realized. Such task consists in adding the area(s) of the reference picture(s) pointed by the MV(s) that were received in the bit stream to the decoded data.

1.2 Overview of the H.264/AVC standard

As with all popular video standards, H.264/AVC defines the syntax and the semantics of the encoded video bit stream, as well as the processing that the decoder needs to perform when converting such bit stream back into a video signal. In addition, H.264/AVC is structured in two different conceptual layers, the Video Coding Layer (VCL) and the Network Adaptation Layer (NAL), in order to provide the flexibility and customization options required to efficiently encode and transmit digital video for a large variety of applications and networks, as discussed in the previous sections. In this partitioning, the VCL defines the actual representation of the video, while the NAL formats the VCL data and provides header information for specific network transport layers or storage media. Although the NAL is of crucial importance to provide ”network friendliness”, therefore allowing for the most efficient use of H.264/AVC bit streams in almost all types of transmission and storage media, it is the VCL that offers all the new coding tools and techniques responsible for the high coding efficiency of the H.264/AVC standard. Moreover, it is also the VCL that presents the most critical computational constraints for the implementation of H.264/AVC video codecs in both software and hardware platforms. Consequently, in this thesis only the VCL is addressed and some of its encoding tools are discussed. The interested reader is referred to [111,143] and [69] for more detailed descriptions of the NAL and of the H.264/AVC standard.

1.2.1 H.264/AVC data flow and encoding procedures

Just like all prior video standards defined by the ITU-T and the ISO/IEC Moving Pictures Experts Group (MPEG), H.264/AVC also adopts the block-based hybrid video coding approach for the VCL. Each encoded picture, which is usually denominated as frame, is represented by using the YUV color space and, typically, the 4:2:0 sub-sampling scheme[7]. In this image format, each MB consists of a rectangular area of 16×16 luma samples and of two 8×8 chroma samples, corresponding to each chroma component. Such MBs are processed in partitions, called

Table 1.1: Overview of the most important coding tools used in the Baseline Profile (BP), Main Profile (MP), Extended Profile (XP) and High Profiles (HiPs) of the H.264/AVC standard.

Coding tool	Baseline Profile	Main Profile	Extended Profile	High Profiles
Picture types	I, P	I, P, B	I, P, B, SI, SP	I, P, B
FMO	Yes	No	Yes	No
Motion block size	16×16, 16×8, 8×16, 8×8, 8×4, 4×8, 4×4	16×16, 16×8, 8×16, 8×8, 8×4, 4×8, 4×4	16×16, 16×8, 8×16, 8×8, 8×4, 4×8, 4×4	16×16, 16×8, 8×16, 8×8, 8×4, 4×8, 4×4
Multiple reference frames	Yes	Yes	Yes	Yes
Motion pel accuracy	1, $\frac{1}{2}$, $\frac{1}{4}$	1, $\frac{1}{2}$, $\frac{1}{4}$	1, $\frac{1}{2}$, $\frac{1}{4}$	1, $\frac{1}{2}$, $\frac{1}{4}$
Weighted prediction	No	Yes	Yes	Yes
Transform	4×4	4×4	4×4	4×4, 8×8
Deblocking filer	Yes	Yes	Yes	Yes
Entropy coding	UVLC, CAVLC	UVLC, CAVLC, CABAC	UVLC, CAVLC	UVLC, CAVLC, CABAC

slices, usually in raster scan order (from left to right and from top to down), although the Flexible Macroblock Ordering (FMO) feature can also be used in some coding modes[69].

The considered source-coding algorithm follows the classical hybrid association of Inter-picture prediction and transform coding of the prediction residual, in order to exploit the temporal redundancy and the spatial irrelevancy, respectively. There is no single coding element in the VCL of H.264/AVC that makes this standard such a successful video coding scheme. In fact, its high coding efficiency results from the combination of several new techniques and tools. However, the joint application of all these tools also significantly increases the complexity of the encoders and of the decoders. Therefore, such complexity constraints are not compatible with most existing applications. This is why the H.264/AVC standard defines several *profiles* and *levels* that allow to specifically address the requirements of every single application, while still guaranteeing the interoperability.

By following this approach, the profiles define the set of coding tools that can be used by an encoder to provide compliant H.264/AVC bit streams, while the levels place constraints on the actual parameters of such bit streams. Currently, the H.264/AVC standard defines 21 different profiles and 17 levels, which are used for all the profiles[69]. Table 1.1 provides a summarized overview of the coding tools used in the four most important profiles: the Baseline Profile (BP), representing the simplest profile and being mainly used for video conferencing and mobile applications; the Main Profile (MP), intended to be used for consumer broadcast and storage applications; the XP, intended for streaming video and including special capabilities to improve robustness; and the High Profile (HiP), intended for high definition broadcast and disc storage (used in HD-DVD and Blue-ray). In the following subsections, the most important coding tools that were introduced by H.264/AVC and used in these profiles are briefly discussed. For further details, the interested reader is referred to[69,111,143].

A – Intra prediction

Intra prediction consists of a coding technique in which the samples of each MB are predicted by only using the data of neighbouring pixel values in the same slice that were already decoded and reconstructed, i.e. already transmitted MBs. In the H.264/AVC standard, all slice-coding types support three types of Intra prediction: $Intra_{4\times4}$, $Intra_{16\times16}$ and $Intra_{8\times8}$.

The $Intra_{4\times4}$ prediction mode is well suited to encode parts of the image with significant amounts of detail. In this mode, each MB is divided in blocks of 4×4 luma samples and a prediction for each block is computed. The H.264/AVC standard defines nine distinct prediction modes: a DC prediction mode, in which all pixels of a given block are predicted using the average of all neighbouring pixels to the left and to the top of the current block; and eight directional prediction modes, that are suited to predict textures with patterns in specific directions. Figure 1.7 shows all the possible prediction directions and the corresponding prediction mode identifications.

The $Intra_{16\times16}$ prediction mode is more suited to encode very smooth areas of a picture. Hence, in this mode only one prediction direction is applied to the whole 16×16 pixels luma block. Even so, four different modes are supported: a DC prediction mode; two directional prediction modes (vertical and horizontal), both of them with an operation entirely similar to the 4×4 prediction modes, but using 16 neighbour pixels on each side for the prediction; and a plane prediction mode. This last mode is intended for areas of gently changing luminance and uses a linear function for the prediction of the pixel values. Since chrominance signals are very smooth in most cases, a quite similar mode to the $Intra_{16\times16}$ is used to predict the chroma samples of a MB. However, in these cases the DC, vertical, horizontal and plane prediction modes are applied on blocks of 8×8 pixels.

Lastly, an extra $Intra_{8\times8}$ prediction mode is also supported for luma blocks in the H.264/AVC High Profile (HiP). A more detailed description of all the Intra prediction modes can be found in[69].

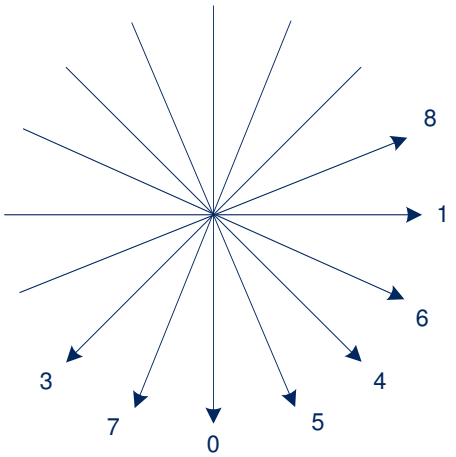


Figure 1.7: Prediction directions for the $Intra_{4\times4}$ mode.

B – Inter prediction

In addition to the Intra coding modes, various predictive or motion compensated coding modes can also be used for P- and B-slices/frames. In such cases, MBs are predicted with pixel data corresponding to already encoded reference frames, by displacing an area of the reference picture using a translational MV. In P-slices, only past pictures can be used as references. Nevertheless, more than one reference picture can be used for each motion-compensated picture, which is known as motion-compensated prediction with multiple reference pictures. Similarly, B-slices can also use multiple reference pictures. However, in this case both past and future pictures can be used as references to build the prediction signal. With such scheme, the prediction signal for a given MB is built by using a weighted average of two distinct motion-compensated prediction values. Figure 1.8 illustrates this concept.

To improve the accuracy of the prediction values, in H.264/AVC each MB can be divided into smaller luma partitions with 16×8, 8×16 and 8×8 pixels, and different MVs are estimated for each partition. Moreover, the 8×8 blocks in P-slices can be further partitioned in subblocks of 8×4, 4×8 and 4×4 pixels to increase even more the MC efficiency, as shown in Figure 1.9. Hence, a maximum of 16 MVs can be transmitted for a single luma MB, by using this so-called variable block size motion-compensated prediction mode.

To further increase the efficiency of the motion-compensated prediction, H.264/AVC also defines the accuracy of the MVs for luma blocks as a quarter of a picture element (pel), i.e. a quarter of the distance between luma samples. Consequently, in order to estimate MVs with fractional-pel resolution and to compensate the corresponding displacements, the pixel data corresponding to the reference image frequently has to be interpolated to sub-pel positions. Usually, the prediction values at half-sample positions are obtained using an interpolation filter that is based on a 6-tap windowed *sinc* function[120]. Conversely, quarter-sample interpolated positions are generated by averaging two integer or half-sample position values. In what concerns the chroma components, the corresponding MVs are one-eightth sample accurate, due to the lower resolution of the chroma sampling grid. The corresponding prediction values are always obtained by bilinear interpolation. The H.264/AVC motion-compensated prediction scheme is explained in detail in[69].

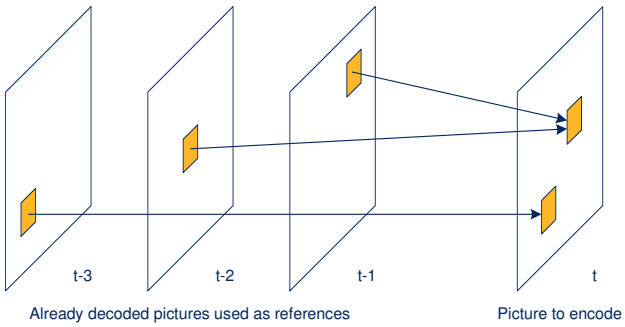


Figure 1.8: Motion-compensated prediction with multiple reference frames.

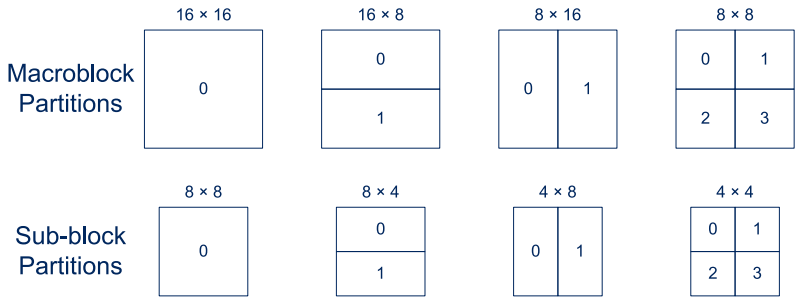


Figure 1.9: MB and block partitioning for MC prediction.

C – Transforms

Likewise the former video standards, the H.264/AVC standard also applies transform coding techniques to the prediction residue, in order to reduce its spatial redundancy. However, a separable 4×4 integer DCT is used to process the luma component, instead of the classical 8×8 DCT. Furthermore, a 2×2 integer transform is also used to encode the two chroma components. When compared with the real DCT, such integer transforms not only minimize the inherent distortion resulting from arithmetic and rounding mismatches but also allow its computation using 16-bits arithmetic, by using only low complexity operations (shift, add and subtract). Moreover, this transform coding tool has proved to be as efficient as the real 8×8 DCT, due to the reduced spatial correlation obtained in the prediction residual after the application of the improved MC technique.

For MB predictions obtained using the $Intra_{16\times16}$ prediction mode, H.264/AVC also defines that a 4×4 Hadamard transform should be further applied over all the 16 DC coefficients of the already transformed luma 4×4 blocks. This procedure allows to better exploit the redundancies in smoother areas of the picture. Conversely, in the High Profiles of the H.264/AVC standard, which allow the processing of MBs composed of four 8×8 blocks, an 8×8 integer transform can alternatively be used to compute the 64 coefficients of the luma MBs that are encoded using either the $Intra_{8\times8}$ or the $Inter$ prediction modes.

The H.264/AVC transform coding procedure is reviewed more thoroughly in subsection 2.2.1 and explained in detail in [69].

D – Entropy coding

The H.264/AVC standard introduced two classes of entropy coding techniques that represent major improvements in the coding efficiency, when compared to the entropy coding tools that were employed in prior video standards, i.e. Variable Length Codes (VLCs) and Context-Adaptive Binary Coding (CABAC).

VLC is the simplest entropy coding method and applies Universal VLC (UVLC) [69] for all syntax elements, except for the quantized transform coefficients. In UVLC, all the syntax elements are mapped into a single codeword table. This table is based on an Exp-Golomb code with very

simple and regular decoding properties, which is customized according to the statistics of the data being encoded. For transmitting the quantized transform coefficients, a more sophisticated and efficient method is used: Context-Adaptive VLC (CAVLC). In this entropy coding mode, several VLC tables are used to encode various syntax elements, by taking into consideration the data already transmitted. These tables implement a total of 32 different VLCs, as a result of having been designed to match the conditioned statistics of all the syntax elements. Consequently, the complexity of the CAVLC scheme is also significantly higher than that of UVLC. Nevertheless, for the typical coding conditions, CAVLC provides bit rate reductions between 2% and 7% regarding the UVLC scheme.

The efficiency of the entropy coding procedure can be further improved by using the CABAC technique, which is based on three key elements: binarization, context modelling and binary arithmetic coding. Binarization provides an efficient mapping of non-binary syntax elements to a sequence of bits, the so-called *bin-string*. Each element of the bin-string is then context modelled and arithmetic encoded to achieve the highest coding efficiency. Such improved performance is owed to the following properties. Firstly, arithmetic coding allows to assign a non-integer number of bits to each encoded symbol of the alphabet, which significantly increases the compression for symbol probabilities that are higher than 0.5. Secondly, context modelling permits to switch between several probability models based on conditional probabilities that are estimated from the statistics of already encoded syntax elements. As a result, CABAC typically provides a reduction in the bit rate between 5% and 15% when compared to CAVLC, but at the cost of a much higher computational complexity. More details on the VLC and CABAC entropy coding tools can be found in[69].

E – Deblocking filter

Block-based coding schemes tend to produce blocking artifacts, because block edges are often reconstructed with less accuracy than interior pixels. The introduction of these artifacts is often considered to be one of the most visible distortions in the decoded video. As a consequence, deblocking filters have been generally applied around the block edges as a post-filtering measure to reduce its visibility. In particular, the H.264/AVC standard defines a mandatory adaptive in-loop deblocking filter that is used not only to improve the visual quality of the coded video but also to enforce the decoder to deliver (approximately) the quality level that is intended by the producer.

The strength of this filtering process is controlled by several parameters and thresholds, as well as by the local characteristics of the picture itself[69]. Moreover, the filter strength can be adaptively controlled at three different levels: *i*) at the slice level, to adjust the filter strength to the characteristics of the video sequence; *ii*) at the block edge level, where the filter strength depends on the Inter/Intra prediction decision, motion differences, etc.; and *iii*) at the sample level, to distinguish the true image edges from those created by the quantization of the transform

coefficients. The application of the adaptive in-loop deblocking filter is explained in detail in[69].

1.2.2 The compression *vs* complexity dilemma

As mentioned in subsection 1.2.1, the global compression gains that are offered by the H.264/AVC standard result from the mutual and cumulative combination of its new coding features and not from the application of an individual or specific coding technique or tool. To increase the flexibility and optimize the coding efficiency for any given application, almost all the H.264/AVC coding techniques can be configured and adjusted, by using some coding parameters that are specified in the standard. As a result, such techniques and functionalities often entail additional complexity in the encoding and decoding procedures. Consequently, an appropriate and careful use of the H.264/AVC coding tools is highly required, in order to avoid compromising the development of commercially viable video solutions due to complexity issues. In fact, it has been shown that a proper selection and parameterization of specific subsets of H.264/AVC encoding tools roughly leads to the same compression performance that can be obtained by simultaneously using all the tools[143]. However, such approach enables a complexity reduction factor of about 6.5 for the encoder and of up to 1.5 for the decoder.

A complexity evaluation of the H.264/AVC coding tools is therefore highly required, so that the system designers are able to develop new processors capable of efficiently implementing real time H.264/AVC codecs. Such evaluation should provide the information necessary to optimize the hardware resources of a given processor for the specific requirements of the target applications. This analysis is also especially important to the design of power-aware multimedia devices, which typically consist of portable and mobile embedded systems that dynamically adapt the computational resources of the multimedia processing on the basis of the available power resources. In such cases, it should be possible to dynamically scale the computational complexity of these systems, by selecting the most adequate coding tools at any given time instant. This approach allows stretching the battery life of the system at the cost of a graceful degradation of the compression performance. However, such goals can only be achieved provided that the codecs are properly optimized to minimize the computational complexity in its two major components, i.e. the *time complexity* and the *spatial (or storage) complexity*.

In the following subsections, these two issues are briefly reviewed and the results of several studies addressing the complexity of the coding tools considered in the H.264/AVC standard are summarized.

A – Time complexity

Time complexity is a performance metric that relates the complexity of a given algorithm implementation to the amount of time required for its execution. Hence, when hardware implementations of a given algorithm are considered, time complexity can be assessed by measuring the

total time required by the hardware structure to produce the required results. Conversely, the evaluation of the time complexity for software implementations requires taking into account two different factors, i.e. *i*) the number of operations required to perform the algorithm implementation; and *ii*) the amount of hardware specific cycle counts for the realization of each of those operations. In addition, the memory bandwidth requirements of the algorithm must be considered in such assessments, since they reflect the time required to perform the data transfers between the processor and the memory system.

As a result, it can be concluded that the estimation of the time complexity of a video encoder or decoder is not a straightforward task. Firstly, because it depends on the intended use of such estimate. When considering hardware implementations, worst-case estimates are typically favoured so as to guarantee the correct circuit operation for the target application. On the other hand, average-case estimates are preferable for software implementations, due to the extra flexibility provided by such systems to recover from abnormal or less probable steps of an algorithm. Secondly, because the complexity estimates heavily depend on the characteristics of the platform to which the algorithm implementation is mapped into (General-Purpose Processor (GPP), Digital Signal Processor (DSP), Field-Programmable Gate Array (FPGA), etc.). Hence, two very similar algorithm implementations generating the same encoding results can present significantly different time complexity values, provided that they are assessed using platforms with quite different characteristics. Lastly, because time complexity is also greatly influenced by external factors, such as the source video content, resolution or bit rate. As a result, a given algorithm implementation may provide low execution times for some specific video contents or resolutions (low time complexity), but fail to do it under different conditions.

B – Storage complexity

Storage complexity evaluates the memory requirements of a given algorithm implementation. In general, the greater the amount of memory an algorithm implementation uses, the more complex it is. As a consequence, storage complexity is often traded by time complexity in many software implementations, so as to reduce the system cost at the expense of a modest increase in the program execution time. This is especially true for platforms based on DSPs and media processors, where the memory cost is very high.

In the most recent video standards, the storage complexity of a given codec can be evaluated based on two different storage classes: the memory that is used to store the constant data, which includes the variable-length decoding tables, the Intra prediction probability tables and other small constant tables; and the memory that is required for the data processing. This latter class can be further divided in three different subclasses, reflecting the different levels of the encoder data structure: *i)* the memory that is needed to accommodate a whole frame, which encompasses both the reconstructed and the reference frame memories; *ii)* the memory that is used to store a line

of MBs, which includes the tables employed in the loop filtering and Intra prediction; and *iii)* the memory that is required to process each MB, which typically covers the temporary buffers used to store the transform coefficients, prediction values and pixel values.

C – Complexity analysis

As it was previously mentioned, assessing the complexity of a video encoder (or even the individual contributions of its several encoding tools to the global encoder complexity) is not a straightforward task. This mostly results from the fact that such evaluation depends on several different factors: the involved algorithms, the encoding options, the input video sequences and the implementation platform (e.g. GPP, DSP, FPGA, etc). This is why multiple studies have been presented in the later years concerning the complexity analysis of several distinct implementations of H.264/AVC encoders and decoders using quite distinct platforms[52,100,108,124,143,155].

In what concerns the H.264/AVC encoder, the results of such studies have revealed that the inherent complexity is more seriously influenced by the following features: use of multiple reference frames, which causes a 25% increase in the complexity for each added frame; and the ME search range that, when combined with the use of multiple reference frames, can increase the complexity up to 60 times whenever large search ranges are considered. Nevertheless, several other tools also significantly contribute to increase the complexity of the encoder. The computation of the transform and quantization operations augments the time complexity between 7% and 20%. Furthermore, it also represents a memory access stress of about 20%. The CABAC entropy coding scheme typically rises the complexity requirements of the encoder by about 10%, when compared to methods using a single reversible VLC table for all syntax elements. The use of multiple modes for variable block size ME also increases the encoder time complexity by 2.5% for each additional mode that is employed. On the other hand, reducing the MV resolution to only half-pel accuracy allows a reduction of about 10% in the time complexity of the whole encoder. As for Intra-prediction and deblocking filtering, their effect in the overall complexity greatly depends on the image content to be encoded. Nevertheless, each of these tools represents a time complexity increase of less than 10%[143].

Unlike the encoder, the complexity of the decoder has been reported to be more significantly influenced by four different tools: inverse transform and reconstruction (including inverse quantization), MC, deblocking filter and entropy decoding. The complexity of the inverse transform and inverse quantization operations represents, on average, 20% of the total complexity of a decoder, both in terms of computation time and memory accesses. The use of B-frames in the MC procedure causes an increase in the decoder storage complexity that can vary between 10% and 30%. However, the extra time that is required to decode such frames only marginally increases the decoder complexity. Conversely, the use of the mandatory adaptive deblocking filter increases the decoder complexity by about 6%, while CABAC represents an increase in the memory accesses

of about 12%. Similarly to the encoder, the decoder time complexity can be reduced by 15% provided that MVs with quarter-pel resolution are not used[143].

1.3 Motivation and objectives

The popularity of multimedia centric devices and applications based on digital video is expected to continue increasing in the near future. Furthermore, the users demand for more, innovative and even better quality video services and applications, like mobile videotelephony, video streamming over the Internet or interactive and 3-D television, shall also continue to increase in the forthcoming years for the generality of consumer entertainment and professional video devices. With the latest advances in display and capture technologies, it is also established that such devices will be required to support even higher resolutions, such as 4096×2160 pixels (i.e. the 4k Ultra High Definition Television (UHDTV) format). For example, smarthphones supporting the 2560×1440 Quad-HD @ 30 fps format are already available today[89], while 4k Ultra HD (3840×2160) TV devices are expected to be commercially viable in the next couple of years. In addition to increased spatial resolutions, the temporal frame rates of the involved video sequences should also increase to around 60 fps and higher.

To enable digital video services with such promising characteristics, video standards employing efficient compression technologies will have to be generally adopted (e.g. the H.264/AVC standard) or new ones will have to be designed. In fact, at the time of writing this thesis, the ITU-T and ISO/IEC have already published the H.265/HEVC standard[70]. When compared with the currently established state-of-the-art H.264/AVC standard, H.265/HEVC is said not only to double the data compression ratio at the same level of video quality but also to efficiently support resolutions up to 8192×4320 pixels (i.e. the 8k UHDTV format)[129].

According to the analysis presented in subsection 1.2.2, this higher coding efficiency will come at the expense of greatly increased complexity, bandwidth and data processing rates. This poses several additional challenges in the design of the upcoming video coding systems, especially when real time operation and the processing of HD contents is demanded, or when mobile and other portable and handheld devices with limited computational, storage and energy resources are considered. These issues shall be even more problematic for the future generations of multimedia devices, which should have to simultaneously support several different video standards due to interoperability and compatibility issues. As a result, these systems will have to implement several distinct video codecs and thus execute multiple coding operations and algorithms, besides all the more general tasks that must be performed to ensure the correct system operation.

To face this restrictions, future video coding systems will be required to include even higher performance and cost effective processing structures, in order to being capable of competently realizing the computation of the most critical and complex operations of the newest (as well as of

the legacy) video standards, like ME, transform and quantization, deblocking filtering, etc. Such structures should also consist of *flexible and modular architectures*, so that they can be easily customised and adjusted (i.e. scaled or configured) to match the requirements of any given video codec, application or implementation platform. Furthermore, their *hardware and power efficiency* must also be relatively high, not only to reduce the implementation costs but also to guarantee a long operation time for the mobile and handheld equipments that include them.

Although ME is considered to be the most time complex operation of a video encoder, which is why many researchers have concentrated their work on this topic [4], the global performance and the efficiency of a video codec is also constrained by other operations, as it is discussed in section 1.1. For instance, transform and quantization are mandatory coding tools in all block-based motion compensated transform coding schemes, independently of the considered video standard, which greatly influence the performance of a video coding system both in terms of compression efficiency and processing requirements.

In what concerns the H.264/AVC standard, such data processing requirements can be incredibly high for video codecs capable of operating in real time or of processing HD video contents, as it can be seen in Figure 1.10. For example, the real time encoding of video sequences in the nowadays well established 1080p HDTV format (1920×1080 pixels @ 30 fps) requires high performance computational systems capable of sustaining processing rates as high as 94×10^6 samples (i.e. pixels) per second. To accomplish such rates, the transform and quantization modules of the video codecs must compute over 460 Giga Operations Per Second (GOPS) and 930 GOPS, respectively [5] . Moreover, they must support data transfers with the system memory of over 1 Gbps, which requires the use of high performance memory modules (e.g. DDR3-800). In what concerns the ”future” standard TV format (i.e. the 4k UHDTV format), which is already supported by many high end and expensive TV equipments, computers, smartphones and tablets, the involved processing requirements are even higher, as it can be seen in Figure 1.10, Figure 1.11 and Figure 1.12. The data processing requirements of video codecs supporting other modern standards are almost identical, since most of these standards have adopted quite similar coding procedures (including the transform and quantization algorithms).

For all these reasons, almost all the existing systems with multimedia capabilities make use of dedicated hardware structures embedded in some specialized processing cores (i.e. Graphics Processing Unit (GPU) and DSP) to perform these operations. In fact, this is also why transform

[4] The author of this document has also made some contributions to such research effort with the investigation he performed in the scope of his Master of Science (MSc) thesis [28–30] and as leading work for this *Philosophiae Doctor* (PhD) thesis [27,31,32,39,123].

[5] In video coding, a metric that is commonly used to assess the algorithm complexity is the amount of Reduced Instruction Set Computer (RISC)-like operations [7]. By using this metric, it can be estimated that the computation of an 8-points one-dimensional (1-D) DCT (see Equation 4.1) requires, at least, 8 image data loads, 8 coefficient data loads, 8 intermediate results data loads, 8 multiply-and-accumulate (MAC) operations and 8 data stores, for a total of 40 operations per image sample (i.e. pixel). As a consequence, $2 \times 40 \times 64 = 5120$ operations must be realized for the computation of the corresponding 2-D DCT, by using the row-column decomposition approach. The amount of RISC-like operations that are required for the computation of the H.264/AVC forward and inverse quantization procedures (see Equation 5.1 and Equation 5.4, respectively) can be estimated also by following this approach.

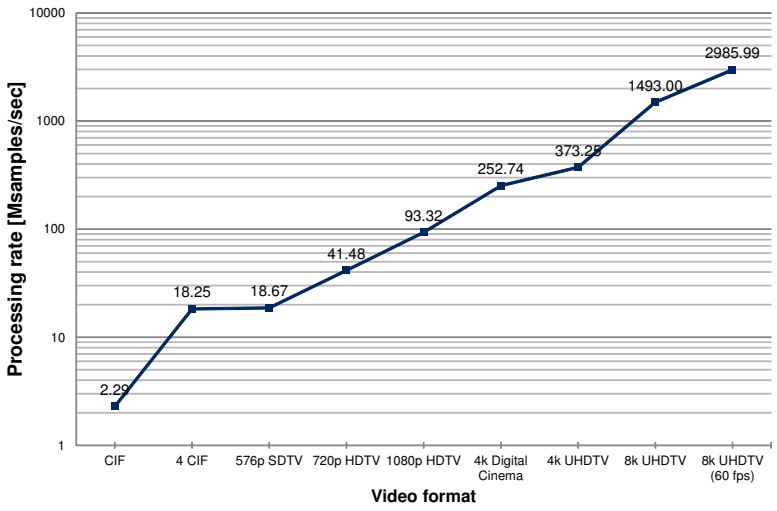
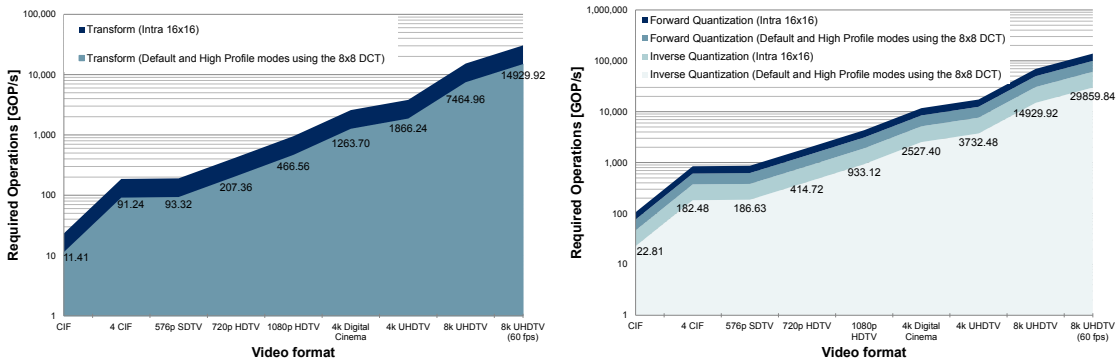
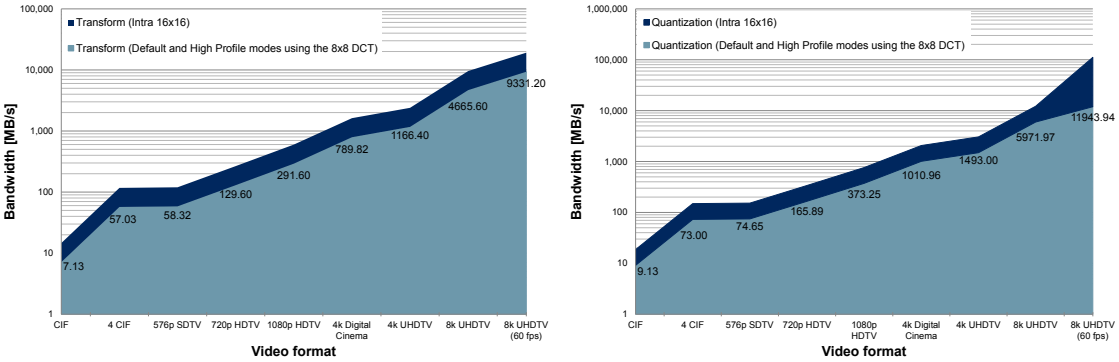


Figure 1.10: Processing rate requirements of an H.264/AVC codec for real time operation.



(a) Transform module. **(b)** Quantization module.

Figure 1.11: Computational rate requirements of an H.264/AVC codec for real time operation. The amount of operations is specified as RISC-like operations[7].



(a) Transform module. **(b)** Quantization module.

Figure 1.12: Bandwidth requirements of an H.264/AVC codec for real time operation.

coding has been an active research topic for several decades, with the publication of multiple studies addressing the definition of new transforms and the design of efficient algorithms and circuits for their computation. Nonetheless, up until now not many of these research works have resulted in the proposal of *high performance* and *hardware efficient* transform and quantization architectures that can be *easily customised or scaled*, in order to successfully develop efficient video encoders and decoders compliant with the requirements of the ITU-T H.264/AVC recommendation (and other recent digital video standards) for several different implementation platforms (e.g., FPGA and Application Specific Integrated Circuit (ASIC)), application domains or classes of video coding equipment.

1.4 Original contributions

In accordance with the previous analysis, the research work presented in this PhD thesis focuses on the development of new high performance and configurable architectures for the computation of the transform and quantization operations specified in the H.264/AVC standard.

In the following paragraphs it is presented a brief description of the main contributions of the conducted research work.

- **Development of a multi-transform architecture for the computation of the H.264/AVC transforms**

  One of the main contributions of this PhD research work is the proposal of an innovative high performance and scalable transform architecture that is capable of efficiently supporting all the transformation procedures defined in the H.264/AVC standard, even for the processing of high definition contents in real time (e.g. the 4k UHDTV format). This Multi-Transform Architecture (MTA) presents a modular and scalable hardware structure, which allows it to be easily configured to efficiently compute any of the 2×2, 4×4 and 8×8 H.264/AVC transforms. Furthermore, it can be configured to support various sets of transforms with distinct kernel values and sizes, making it highly suitable also for the realization of multi-standard transform cores. The flexible design that was adopted for this architecture provides the means required to better adapt it to the specific performance and hardware requirements of the target video coding systems and applications.

  The most relevant contributions of this study were published in the following scientific journals and conference proceedings:

  [23] T. Dias, S. López, N. Roma, and L. Sousa. Efficient and programmable processing unit for H.264/AVC systolic unified transform engines. In *VII Jornadas sobre Sistemas Reconfiguráveis (REC 2011)*, pages 13–19, Feb. 2011;

  [24] T. Dias, S. López, N. Roma, and L. Sousa. A flexible architecture for the computation of direct and inverse transforms in H.264/AVC video codecs. *IEEE Transactions on*

Consumer Electronics, 57(2):936–944, May 2011;

[25] T. Dias, S. López, N. Roma, and L. Sousa. High throughput and scalable architecture for unified transform coding in embedded H.264/AVC video coding systems. In *International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS 2011)*, pages 225–232, July 2011. Best Paper Award;

[26] T. Dias, S. López, N. Roma, and L. Sousa. Scalable unified transform architecture for advanced video coding embedded systems. *International Journal of Parallel Programming*, 41(2):236–260, Apr. 2013;

[36] T. Dias, N. Roma, and L. Sousa. High performance multi-standard architecture for DCT computation in H.264/AVC high profile and HEVC codecs. In *Conference on Design & Architectures for Signal and Image Processing (DASIP 2013)*, pages 14–21, Oct. 2013. Best Paper Award;

[35] T. Dias, N. Roma, and L. Sousa. Exploiting coarse-grained parallelism in multi-transform architectures for H.264/AVC high profile codecs. In *Conference on Electronics, Telecommunications and Computers (CETC 2013)*, pages CD–ROM, Oct. 2013. ISBN: 978-989-97531-3-6;

[37] T. Dias, N. Roma, and L. Sousa. Unified transform architecture for AVC, AVS, VC-1 and HEVC high-performance codecs. *EURASIP Journal on Advances in Signal Processing*, 2014(108), July 2014.

- **Development of quantization architectures for the H.264/AVC standard**

  Another very important contribution of this research work consists in the definition of a new class of high performance architectures with a reduced hardware cost for the computation of the H.264/AVC forward and inverse quantization operations. Such processing structures are based on a highly configurable design, which can be customised to realize both dedicated and unified quantization modules for H.264/AVC codecs. In addition, it supports multiple hardware configurations offering distinct performance *vs* hardware cost trade-offs. With this approach, it is possible to adjust the processing rate of the quantization modules not only to the performance of the remaining processing modules of the video codec but also to the requirements of the target application. Hence, such remarkable versatility allows this class of architectures to optimally address the requirements of any given quantizer.

  Most of these contributions were published in the following scientific conference proceedings:

  [33] T. Dias, N. Roma, and L. Sousa. Optimized forward/inverse quantization unit for H.264/AVC codecs. In *Conference on Electronics, Telecommunications and Computers (CETC 2011)*, pages CD–ROM, Nov. 2011. ISBN: 978-989-97531-0-5;

[34] T. Dias, N. Roma, and L. Sousa. Reconfigurable unified architecture for forward and inverse quantization in H.264/AVC. In *VIII Jornadas sobre Sistemas Reconfiguráveis (REC 2012)*, pages 75–82, Feb. 2012;

[40] T. Dias, L. Rosário, N. Roma, and L. Sousa. High performance unified architecture for forward and inverse quantization in H.264/AVC. In *15th Euromicro Conference on Digital System Design (DSD 2012)*, pages 632–639, Sept. 2012.

- **Proposal of an integrated transform and quantization architecture for the H.264/AVC standard**

  To reduce the impact of the transform coding module in the performance of video codecs compliant with the H.264/AVC standard, this research work also comprehended the investigation of a new integrated transform and quantization architecture for the H.264/AVC standard. The presented processing structure is capable of implementing not only the fundamental transform and quantization procedures but also the default forward and inverse transform and quantization coding paths, the $Intra_{16\times16}$ transform and quantization coding path and all the optional forward and inverse transform and quantization coding paths based on the 8×8 DCT that are defined in the high profiles of the H.264/AVC standard. In order to maximize the offered data processing rates, this architecture performs the transform and quantization operations in parallel. Such computations are realized by using the transform and quantization architectures that were also developed in the scope of this PhD thesis.
  Some preliminary and preparatory aspects of the study that was conducted to devise this architecture were already presented in three international scientific conferences:

  [122] N. Sebastião, T. Dias, N. Roma, and P. Flores. Integrated accelerator architecture for DNA sequences alignment with enhanced traceback phase. In *International Conference on High Performance Computing & Simulation (HPCS 2010)*, pages 16–23, June 2010;

  [31] T. Dias, N. Roma, and L. Sousa. H.264/AVC framework for multi-core embedded video encoders. In *International Symposium on System-on-Chip (SOC 2010)*, pages 89–92, Sept. 2010;

  [32] T. Dias, N. Roma, and L. Sousa. Hardware/software co-design of H.264/AVC encoders for multi-core embedded systems. In *International Conference on Design and Architectures for Signal and Image Processing (DASIP 2010)*, pages 231–238, Oct. 2010. Best Poster Award.

In addition, the research work that was developed in the scope of this thesis was also distinguished with the following awards:

- *Best Paper Award* in the 2010 Conference on Design and Architectures for Signal and

Image Processing (DASIP 2010), which was held in Edinburgh, United Kingdom, in October 2010;

- *The "Stamatis Vassiliadis" Best Paper Award* in the 11th International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS XI), which was held in the Greek island of Samos in July 2011;
- *Best Poster Award* in the 2013 Conference on Design and Architectures for Signal and Image Processing (DASIP 2013), which was held in Cagliari, Italy, in October 2013.

1.5 Outline

This thesis is organized in seven chapters. This introductory chapter not only presents the motivation and the objectives of the research work supporting this PhD thesis but also the most relevant background information concerning video coding and the H.264/AVC standard. Chapter 2 is devoted to the study of the transform coding modules used in digital video codecs, with a special emphasis on the transform and quantization techniques employed in the H.264/AVC standard. For such purpose, the main principles of transform coding are briefly reviewed, together with the presentation of an overview of the transform and quantization operations. In chapter 3 it is presented the state of the art concerning the most relevant architectures and systems that have been proposed in the literature to efficiently support the computation of these two operations. Then, chapter 4 introduces the scalable Multi-Transform Architecture (MTA) that was developed in the scope of this thesis, while chapter 5 discloses the proposed class of quantization architectures for the H.264/AVC standard. After these two chapters, which present the major contributions of this PhD research work, a comprehensive set of experimental results concerning the implementation of the proposed processing structures using FPGA technologies is provided in chapter 6. Such results not only are thoroughly analysed but also compared with some of the more relevant alternative designs described in the literature. Furthermore, an integrated transform and quantization architecture for the H.264/AVC standard is also proposed and discussed in this chapter. Finally, chapter 7 concludes the thesis with the presentation of a brief summary of the main contributions of the performed research work. In addition, some possible directions for future work in this research domain are also discussed.

2

Transform coding

Contents

2.1 Fundamentals of transform coding 30
2.2 Transform coding in the H.264/AVC standard 38
2.3 Summary 45

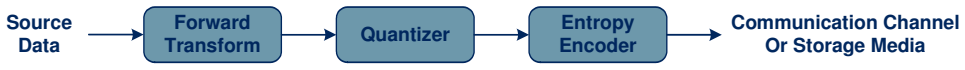
Transform coding is one of the basic tools that is used in almost all lossy audio, image and video coding applications to improve the compression ratio of the encoded data. Similarly to[95,142], in this thesis we consider that this tool comprehends three distinct operations that are performed in cascade, i.e. transform, quantization and entropy coding. The transform is employed to decorrelate the input data represented in the spatial (pixel) domain and compact its energy in a frequency domain. Quantization is subsequently applied to the obtained transform coefficients, the stage at which the loss of video detail is traded by the desired video compression ratio. Finally, the quantized coefficients are entropy coded using VLCs or arithmetic codes, in order to reduce the average number of bits that are used to represent the compressed video bit stream.

In the following sections, the typical transform coding procedure applied in block-based digital video coding is briefly reviewed, with focus on the transform and quantization operations. First, the basic concepts and principles of these two techniques are analysed in section 2.1. Then, the transform and quantization procedures adopted by the H.264/AVC standard are presented and discussed in more detail in subsections 2.2.1 and 2.2.2, respectively.

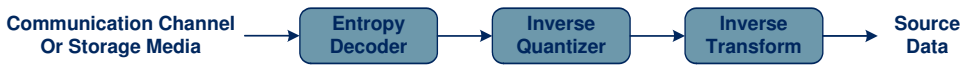
2.1 Fundamentals of transform coding

As discussed in section 1.1, transform coding is a fundamental tool used in digital video coding to obtain the level of data compression required for a given application. To achieve such goal, this tool exploits both the characteristics of the HVS and the correlations commonly found among pixels in a given picture neighbourhood [1], in order to reduce the entropy of the picture and thus minimize the amount of bits required to represent its data.

As it can be seen in Figure 2.1, such lossy compression procedure involves three distinct tasks, each one exploiting a different kind of redundancy in the picture data. In what concerns the operation of the encoder, the spatial redundancy in a picture is first exploited with the computation of a linear transform involving the input pixels. The resulting transform coefficients are



(a) Encoder.



(b) Decoder.

Figure 2.1: Transform-based picture coding procedure.

[1] The tendencies of pixels to be similar to their neighbours can be the result of the gradual light changes or the continuity, the texturing and the boundaries of the represented objects, or even due to the similarity of the multiple objects that are represented in the picture.

then individually quantized, by exploiting the characteristics of the HVS so as to discard the information perceptually irrelevant to the human eye, i.e. the psychovisual redundancy. Finally, the entropy encoder employs its knowledge of the transform and quantization processes to minimize the amount of bits required to encode the symbols generated by the quantizer, i.e. the statistical redundancy. Regarding to the decoder, it implements a very similar 3-steps procedure to reconstruct the picture, albeit the losses introduced by the quantization procedure. As it can be seen in Figure 2.1(b), this approach involves the inverse operations of the ones computed by the encoder, which are applied in the reverse order.

In the following subsections, the transform and quantization procedures commonly used in digital video coding are analysed more thoroughly. However, since the discussion about entropy coding is out of the scope of this thesis, for further information on this subject the interested reader is referred to[48,91].

2.1.1 Transform

In digital video coding, linear transforms are used to reduce the inter-pixel redundancy in the residual pictures resulting from the Intra- and Inter-prediction procedures (see subsection 1.1.1). This is achieved with the decorrelation and the representation of the original data in the transform domain, where the characteristics of the HVS can be more efficiently exploited to achieve data compression, as discussed in subsection 2.1.2. Such lossless operation consists in the computation of a 2-D linear transform, as shown in Equation 2.1, which mainly redistributes the picture energy into a small number of transform coefficients in a more compact manner. In this equation, x represents the data in the spatial (pixel) domain and y the corresponding coefficients in the transform domain, while c consists of the transform kernel representing the 2-D linear transformation.

$$y_{kl} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} x_{ij} c_{ik} c_{jl}, \quad k, l = 0, ..., N-1 \tag{2.1}$$

In order to maximize the coefficient decorrelation, this transformation procedure should ideally involve the whole pixels of the picture[91]. However, in block-based compression procedures, like the ones adopted by the ITU-T H.26x[66,67,69,70] and ISO/IEC MPEG-x[61,62,68] video standards, transforms are computed only for square blocks of pixels and on a block-by-block basis, with the block size (N) typically in the range of 2^1 to 2^5. This alternative computational approach is preferable for two specific reasons. On the one hand, it allows reducing the complexity of the transformation procedure, i.e. the computation and the storage requirements. On the other hand, the computation of larger transforms does not offer significantly better compression levels, since according to the rate distortion theory the pixel covariance function decays rapidly at long distances[7]. However, when high compression ratios are required, this simplified block-based approach can also introduce noticeable reconstruction errors at the block boundaries, i.e. blocking

artifacts[91]. In such cases, the boundaries between adjacent blocks become highly visible, due to the high frequency coefficients of the block being either discarded or more coarsely encoded by the quantizer, as discussed in subsection 2.1.2.

From the previous discussion, it can be concluded that the implementation of a good transform-domain block coding procedure requires the definition of a proper transform, as well as the optimal size of the picture blocks. From the compression point of view, a block with $N \times N$ pixels should be optimally processed by using a transform with size N that presents the following characteristics[48]:

- *Reversibility*, in order to guarantee that the input signal can be recovered to its original domain, and without errors, after applying the transform followed by its corresponding inverse transform. This is an essential feature, since video decoders must recover the original data in the spatial domain prior to displaying it;
- *Energy compaction*, so that the energy of the original picture can be concentrated, without any loss, in the smallest number of transform coefficients as possible;
- *Decorrelation*, to maximize the compression, by guaranteeing that each transform coefficient always conveys additional information that presents no or, at most, small repetition of the data;
- *Data independence*, in order to achieve a good compression efficiency for different picture types and with reduced complexity requirements;
- *Low complexity*, to enable the use of fast algorithms for its computation, which is desirable for both software and hardware implementations.

To comply with these requirements, the transforms most commonly adopted in digital video coding are unitary transforms[75]. This class of transforms is mainly characterized by including kernels with orthonormal basis functions, which presents several advantages for video and image compression. In fact, the kernel of a unitary transform not only provides good decorrelation, energy compaction and energy preservation representations of the transformed data but also is suitable to implement both the forward and the inverse transformation procedures (i.e. reversibility) with minor changes.

Among all the unitary transforms, the Karhunen-Loève Transform (KLT)[8] is an optimal transform in terms of decorrelation and energy compaction representation, since it minimizes the mean square error between the original and the transformed pictures. However, this transform presents several implementation-related problems, because its kernel is not composed of a fixed set of basis functions. In fact, the kernel basis functions are computed as functions of the input picture data under processing, which makes them data dependent. Consequently, the transform

kernel must be computed for each picture block, which greatly increases the computational requirements of video encoders. Since the transform kernels cannot be precomputed, the use of the KLT in video coding also requires a continuous transmission of the transform kernel coefficients to the decoder. This significantly affects the transmission bandwidth requirements of the video encoder, which quite often is a very serious problem in most transmission media, especially in wireless environments. Moreover, the transmission of both the kernel and the coefficient values, typically, does not pay-off in terms of compression ratio. Lastly, the computation of the KLT requires full matrix multiplication, therefore demanding unreasonable computation resources due to the fact that the KLT kernel is non-separable. Consequently, the use of the optimal KLT is rather uncommon in digital video coding, as it fails to fulfil the data independence and the low complexity requirements demanded for a "proper" and efficient transform. As a result, the most relevant legacy digital video and image standards (e.g. H.261/3, MPEG-1/2/4 and JPEG) have all adopted the type-II Discrete Cosine Transform (DCT)[119], or other DCT-like transforms, because it consists of a modest complexity and robust approximation to the optimal KLT, while still offering very similar performance results in terms of energy compaction, as well as fast implementations.

Mathematically, for a square picture block with $N \times N$ samples, the forward and inverse type-II DCT can be formulated as shown in Equations 2.2 and 2.3, respectively, with $k, l = 0, 1, ..., (N-1)$ and $\xi(.)$ given by Equation 2.4.

$$y_{kl} = \frac{2}{N}\xi(k)\xi(l)\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} x_{ij}\cos\left(k\frac{(2i+1)\pi}{2N}\right)\cos\left(l\frac{(2j+1)\pi}{2N}\right) \tag{2.2}$$

$$x_{ij} = \frac{2}{N}\xi(i)\xi(j)\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} y_{kl}\cos\left(i\frac{(2k+1)\pi}{2N}\right)\cos\left(j\frac{(2l+1)\pi}{2N}\right) \tag{2.3}$$

$$\xi(p) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{if } p\text{=0} \\ 1, & \text{otherwise} \end{cases} \tag{2.4}$$

It is clear from Equation 2.2 that the first basis function of the DCT kernel ($(i, j) = (0, 0)$) corresponds to a constant function, which represents the average value of the block under transformation. Consequently, this value is typically known as the DC coefficient, in analogy to what happens with the circuits analysis theory in electrical engineering. In accordance, all the other transform coefficients are known as AC coefficients and it can be noted that their corresponding basis functions exhibit a progressive increase in frequency, both in the vertical and horizontal direction. In Figure 2.2 it is depicted this behaviour of the DCT basis functions for $N = 8$, which has been extensively used in the legacy ISO/IEC and ITU-T image and video standards, i.e. the JPEG image standard and the MPEG-1/2/4 and H.261/3 video standards.

Several different algorithms for fast and efficient computation of the DCT have been developed, as a result of the wide application of this transform. Many of them take advantage of the relationships between the DCT and various existing fast transforms (e.g.[4,101,119,135,136]), while

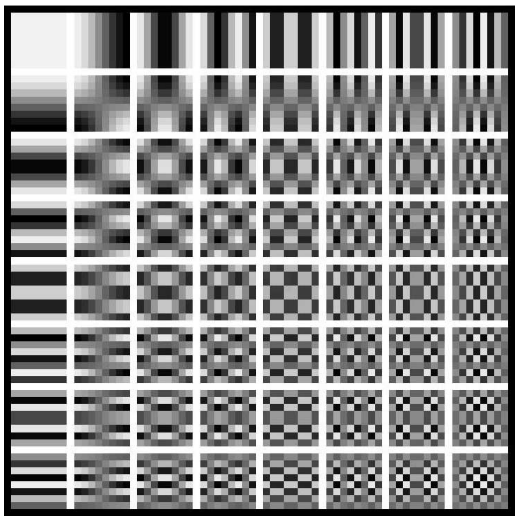


Figure 2.2: 8×8 **DCT basis functions.**

others are based on the sparse factorization of the DCT kernel[53,99]. Alternative proposals also consist of recursive implementations[15,82], which includes the row-column decomposition strategy as a result of the separability property that characterizes the 2-D DCT[119]. In such approach, a 2-D transform can be computed as two consecutive 1-D transforms operating on the rows and columns of the predicted blocks. The application of such methodology to Equation 2.2 is shown in Equations 2.5 and 2.6, respectively.

$$y'_{kj} = \sqrt{\frac{2}{N}}\xi(k) \sum_{i=0}^{N-1} x_{ij} \cos\left(k\frac{(2i+1)\pi}{2N}\right) \tag{2.5}$$

$$y_{kl} = \sqrt{\frac{2}{N}}\xi(l) \sum_{j=0}^{N-1} y'_{kj} \cos\left(l\frac{(2j+1)\pi}{2N}\right) \tag{2.6}$$

As it can be seen, the row-column decomposition strategy poses several advantages in terms of the involved computation time, since it allows to significantly reduce the computational complexity from $\mathcal{O}(N^4)$ to $\mathcal{O}(N^3)$[91]. Nevertheless, the implementation of all these DCT algorithms also presents several drawbacks, mostly resulting from the fact that all the entries in the transform kernels are irrational numbers. Hence, a straightforward computation of the DCT requires floating-point implementations involving expensive and slow arithmetic circuits. In addition, the use of such circuitry may cause drift (i.e. mismatch between the decoded data in the decoder and in the encoder), as a result of the forward and inverse transforms being implemented in different machines with distinct floating-point representations and rounding operations. Modern video standards like H.264/AVC[111] (or the AVS[150], VC-1[128] and H.265/HEVC[129] standards) are very sensitive to prediction drift, since they make extensive use of prediction to increase the offered compression efficiency. Consequently, in order to avoid all the problems mentioned above, state-of-the-art video standards have been adopting integer orthogonal transforms with the same symmetry and energy compaction properties of the DCT.

Although several different approaches can be used to obtain viable integer transform kernels[4,92], the theory of dyadic symmetry[10] has been considered in the development of almost all the newest video standards[111,128,129,150]. This technique allows to compute the transform not only by using reduced bit-width integer arithmetic (e.g. 16-bits) but also by exclusively considering integer multiplications and additions. However, the basis vectors of the resulting integer transform kernels frequently do not have the same norm[102]. Consequently, it becomes necessary to include an additional normalization stage in the video coding algorithm. Such arrangement is shown in Equation 2.7, which formulates the computation of the 2-D transform given in Equation 2.2 using the matrix notation in a factorized form. In this equation, A is a transform matrix whose entries correspond to the DCT basis functions and C is the corresponding normalized transform matrix, while E is a matrix of Scaling Factors (SFs).

$$Y = AXA^T = (CXC^T) \otimes E \tag{2.7}$$

As it can be seen, the original transformation procedure is factorized into two distinct operations: *i*) the computation of a 2-D transform based on an orthonormal kernel C and *ii*) an element-by-element matrix multiplication ($\otimes$) involving the SFs. In most video standards, these SFs are integrated into the quantization stage of the video codecs[128,129,143,150], in order to minimize the complexity of the coding procedure. In fact, that is the case of the H.264/AVC standard, as discussed in subsection 2.2.2.

2.1.2 Quantization

In general terms, quantization can be defined as the process of mapping a large set of input values y, described in a generic source alphabet Λ^N, to a smaller set of output values $\hat{y}$ that compose the reproduction codebook $C = \{\hat{y}_i\}_{i \in L} \subset \Lambda^N$. In this definition, L is a finite or countable infinite index set, while Λ can be a set of both real and integer scalars ($N = 1$) or vector values ($N > 1$). Since the cardinality of the reconstruction codebook is smaller than the one of the source alphabet, quantization can provide an effective means to achieve data compression. This is why quantization is a technique commonly used in lossy data compression schemes, like the ones implemented in the most relevant digital video standards (e.g. the ITU-T H.26x[66,67,69,70] and the ISO/IEC MPEG-x[61,62,68] standards), in order to reduce the entropy of the transform coefficients. In such systems, the implementation of the quantization algorithm (i.e. the quantizer) is decomposed in two distinct stages, as shown in Figure 2.3.

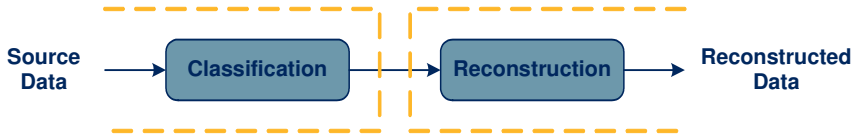


Figure 2.3: Generic architecture of a quantizer.

On the encoder side, the quantizer implements a *classification* procedure, in which the range of the source input is classified into K non-overlapping intervals I_i, with $i = 1, ..., K$ and $K \in L$. According to this procedure, commonly denominated as forward quantization, all the source inputs y that fall into a given quantization interval I_i are associated with the same quantizer index i. The complementary *reconstruction* procedure, also known as inverse quantization, is implemented on the decoder side, where each quantization interval is represented by an output (or reconstruction value) $\hat{y}_i$. Naturally, all the output values are specified in the reconstruction codebook C and can be obtained by implementing the mapping $y \in I_i \Rightarrow \hat{y} = \hat{y}_i$. Note that the quantization intervals are defined based on the quantizer decision boundary values b_i and thus the length of each interval, which is commonly known as the Quantization Step ($Qstep$), is given by $\Delta_i = [b_{i-1}, b_i[$. The two extreme limits b_0 and b_K correspond to the outer constraints of the source input signal, in the range $[Y_{min}, Y_{max}]$, and are defined as $b_0 = Y_{min}$ and $b_K = Y_{max}$.

From the previous discussion, it can be easily concluded that the quantization procedure can be used to compress source data, but at the cost of irreversibly introducing some amount of distortion (i.e. an error) in the output data. This results from the fact that the reconstructed data quite frequently differs from the original one, due to the loss of precision imposed by the implemented non-injective many-to-one mapping process. In general, the output data values $\hat{y}_i$ are at the midpoints $i\Delta_i - \frac{\Delta_i}{2}$. Therefore, they can present a quantization error in the range $[-\frac{\Delta_i}{2}, \frac{\Delta_i}{2}]$ with a variance of $\frac{\Delta_i^2}{12}$ [91], assuming equal length intervals. As a result, it can be concluded that the main goal when designing a successful quantizer is to minimize the distortion for a given source input, given a fixed number of output values, i.e. discretization intervals.

In video coding algorithms, this rate-distortion optimized quantization approach is highly relevant. In fact, it enables an encoder to insert a certain degree of distortion that is imperceptible to the HVS in the encoded video, in order to compress such data within the limits of the bit rate supported by the target communication channel or storage media. In fact, the most commonly used video standards adopt this approach, by employing scalar quantizers[91] to compress all the transform coefficients of both the Intra and Inter-predicted blocks. The involved quantization algorithms typically consist in the division of each of these frequency domain coefficients (y_{ij}) by an integer q_{ij} and then rounding the resulting value to the nearest integer, as shown in Equation 2.8, where $i, j = 1, ..., (N-1)$ and N is the block size.

$$z_{ij} = round\left(\frac{y_{ij}}{q_{ij}}\right) \tag{2.8}$$

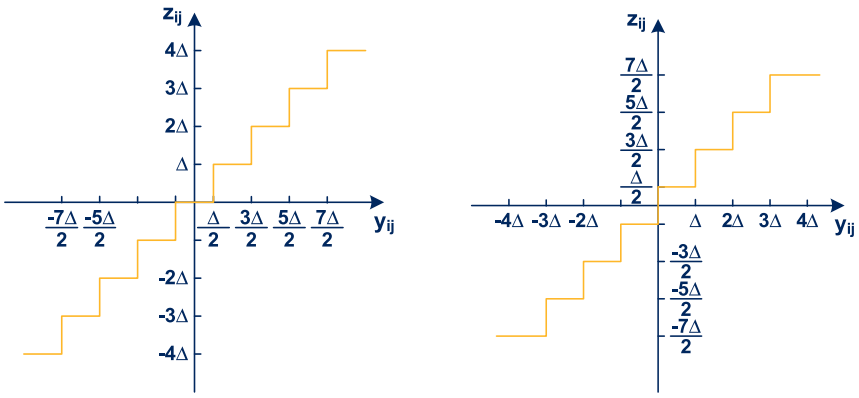
The quantization step values q_{ij} are typically obtained by using quantization tables $q_{ij} = Q(i, j)$ that are designed based on psychovisual studies, with the goal of maximizing the compression ratio while minimizing the perceptual losses in the encoded pictures. As a result, the entries in such tables tend to have larger values towards the higher frequency basis functions (i.e. the lower right corner of the transform blocks), to which the HVS is much less sensitive. Moreover, different quantization tables are also used in the Intra and Inter-prediction procedures of some video

standards, so that the amount of distortion that is introduced in the encoded data can be more efficiently managed before the quantization error becomes visible. As an example, in Equations 2.9 and 2.10 it is presented the quantization tables used in the ISO/IEC MPEG-1 standard for the Intra and Inter-coding procedures, respectively.

$$Q_{Intra} = \begin{bmatrix} 8 & 16 & 19 & 22 & 26 & 27 & 29 & 34 \\ 16 & 16 & 22 & 24 & 27 & 29 & 34 & 37 \\ 19 & 22 & 26 & 27 & 29 & 34 & 34 & 38 \\ 22 & 22 & 26 & 27 & 29 & 34 & 37 & 40 \\ 22 & 26 & 27 & 29 & 32 & 35 & 40 & 48 \\ 26 & 27 & 29 & 32 & 35 & 40 & 48 & 58 \\ 26 & 27 & 29 & 34 & 38 & 46 & 56 & 69 \\ 27 & 29 & 35 & 38 & 46 & 56 & 69 & 83 \end{bmatrix} \tag{2.9}$$

$$Q_{Inter} = \begin{bmatrix} 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \end{bmatrix} \tag{2.10}$$

For the previously mentioned reasons, different rounding operations are also commonly employed in the processing of the DC and AC coefficients of the transform blocks, as it is shown in Figure 2.4. While the AC coefficients are quantized using a mid-treat quantizer[91], which uses the rounding to the nearest integer operation shown in Equation 2.8, the DC coefficients are typically quantized using a mid-rise quantizer[91]. This type of quantizer, whose definition is presented in Equation 2.11, is based on the floor operator ($\lfloor . \rfloor$) and presents an average quantization error smaller than that of the mid-treat quantizer. Consequently, it can be used to encode the DC coefficients more efficiently, since they have a significant impact in the subjective quality of the encoded pictures and seldom are zero.



(a) Mid-tread quantizer. **(b)** Mid-rise quantizer.

Figure 2.4: Input-output characteristic of a uniform quantizer.

$$z_{ij} = \left\lfloor \frac{y_{ij}}{q_{ij}} \right\rfloor \tag{2.11}$$

In contrast, the mid-treat quantizer leaves a center ”dead zone” in the quantization space, where a large input range is mapped to zero. Such characteristic, which is depicted in Figure 2.4(a), is important when the input data fluctuates between small positive and integer values, since it can be used to represent the zero value for a much larger range of values and thus to improve the compression efficiency. This is the case of the AC transform coefficients, which usually present very small values, especially for the higher frequency basis functions of the considered transform, to which the HVS is much less sensitive.

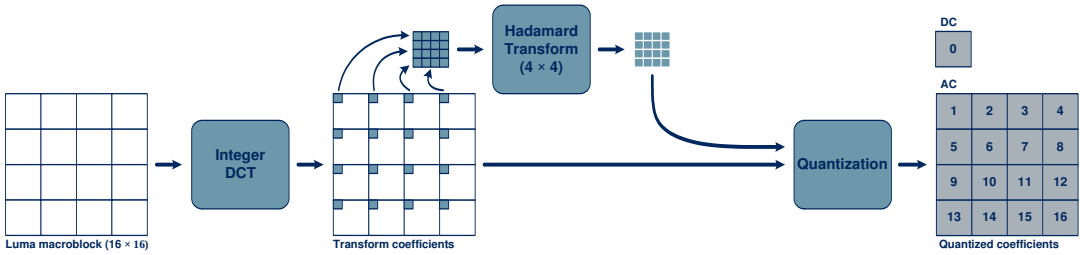
2.2 Transform coding in the H.264/AVC standard

The H.264/AVC standard also implements the classical block-based transform and quantization coding procedure. However, the involved transform and quantization operations significantly differ from the ones adopted in previous ITU-T H.26x and ISO/IEC MPEG-x standards, in order to maximize the trade-off between the offered compression efficiency and the resulting visual distortion for the encoded video sequences. As a result, in H.264/AVC these two operations not only present significant improvements to its classical realizations but also are much more complex and tightly coupled. In the following subsections, the H.264/AVC transform and quantization procedures are briefly reviewed.

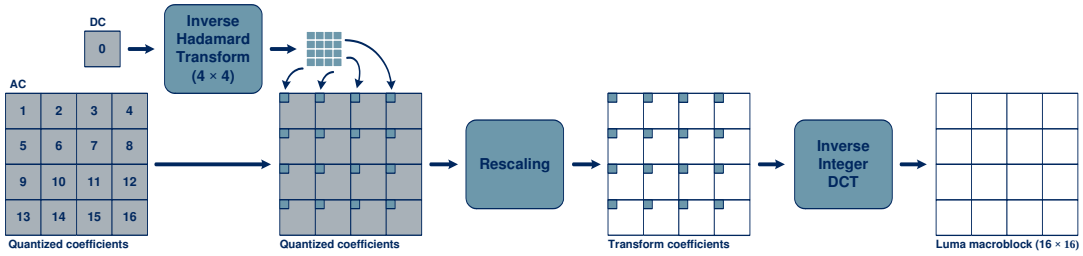
2.2.1 Hierarchical transformation procedure

The transformation procedure defined in the H.264/AVC standard consists of a two-level hierarchical transform path based on multiple integer transforms, namely, the forward and inverse 8×8 and 4×4 integer DCTs and the 4×4 and 2×2 Hadamard transforms[111]. This innovative approach not only allows increasing the compression performance offered by video encoders, due to the adoption of multiple transform block sizes, but also eliminating the propagation of rounding drift errors inside the encoder loop in the streaming direction, i.e. from the encoder to the decoder. Moreover, it alleviates the computation and the memory bandwidth requirements of the transform operation[102], which are very important features in the design of high performance and efficient transform computation modules, especially when video coding systems are required to operate in real time.

As it is shown in Figure 2.5, in the first transform level of the default coding mode of the H.264/AVC standard, a 4×4 ”core” transform is applied to all the 4×4 luma and chroma blocks of residual data resulting from either the motion-compensated Inter-prediction or the spatial Intra-prediction stages. This transform represents a simplified integer DCT, where the scaling factor was transferred to the subsequent quantization stage of the encoding algorithm[120]. As a conse-



(a) Forward transform coding path.



(b) Inverse transform coding path.

Figure 2.5: Hierarchical transform paths defined in the H.264/AVC standard.

quence, the considered forward and inverse transform kernels contain only four different power of 2 values each, as it can be seen in Equations 2.12 and 2.13, respectively. This characteristic greatly minimizes the complexity of these transforms, by allowing their computation using only addition and shift operations operated in 16-bits integer arithmetic.

$$C_{4\times 4_f} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \tag{2.12}$$

$$C_{4\times 4_i} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \frac{1}{2} & -\frac{1}{2} & -1 \\ 1 & -1 & -1 & 1 \\ \frac{1}{2} & -1 & 1 & -\frac{1}{2} \end{bmatrix} \tag{2.13}$$

The second level of the transform procedure is based on the Hadamard transform[8] and provides the means required to better exploit the redundancies in smoother areas of a picture. For such purpose, the Hadamard transform is used to process the DC coefficients of the transformed blocks, already computed in the previous level using the 4×4 integer DCT. Nonetheless, two distinct Hadamard transforms are used to process the DC coefficients corresponding to the luma and chroma blocks. For the MBs that are encoded using the $Intra_{16\times 16}$ prediction mode, a 4×4 Hadamard transform is used to process the sixteen DC coefficients of all the 4×4 luma blocks. Conversely, a 2×2 Hadamard transform is used to process the two sets of 4 chroma DC coefficients belonging to the four 4×4 chroma blocks of each MB in all the coding modes.

Likewise the 4×4 integer DCT, the 4×4 Hadamard transform is also an integer transform but with an even more reduced complexity. In fact, it can be implemented by using only integer

additions and subtractions, since its kernel contains only the 1 and -1 coefficients, as shown in Equation 2.14. Moreover, the same kernel can be used to compute both the forward and inverse transforms due to its symmetric nature. Regarding to the 2×2 Hadamard transform, its kernel consist of a sub-sampled version of the 4×4 Hadamard transform kernel and therefore contains only the first two entries of rows 0 and 2 of such kernel, as represented in Equation 2.15. Once again, the same kernel is used to compute the inverse and the forward 2×2 Hadamard transforms, as a result of its symmetric nature.

$$H_{4\times4} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \tag{2.14}$$

$$H_{2\times2} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2.15}$$

The high profiles defined in the FRExt amendment of the H.264/AVC standard also allow the processing of MBs composed of luma blocks with 8×8 samples[111]. In such cases, the transform process consists only of the first transform level and an 8×8 integer transform (based on the type-II DCT) is used to compute all the 64 coefficients of the four luma blocks composing the MBs that are encoded either in the $Intra_{8\times8}$ prediction mode or in any Inter prediction modes. Likewise the Hadamard transforms, the forward and inverse 8×8 integer transforms can share the same kernel, as shown in Equation 2.16. Nevertheless, since the involved transformed coefficients are of greater magnitude and more diverse, the complexity of these kernels is relatively higher, as it can be seen in Equation 2.17. Hence, the dynamic gain of these kernels is also relatively higher, which compromises the computation of the forward transform using only 16-bits arithmetic. As a result, in the computation of the forward 8×8 transform, a rescaling operation must be performed after the row-wise transform stage.

$$C_{8\times8_f} = \frac{1}{8} \times C_{8\times8_i} \tag{2.16}$$

$$C_{8\times8_i} = \begin{bmatrix} 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\ 12 & 10 & 6 & 3 & -3 & -6 & -10 & -12 \\ 8 & 4 & -4 & -8 & -8 & -4 & 4 & 8 \\ 10 & -3 & -12 & -6 & 6 & 12 & 3 & -10 \\ 8 & -8 & -8 & 8 & 8 & -8 & -8 & 8 \\ 6 & -12 & 3 & 10 & -10 & -3 & 12 & -6 \\ 4 & -8 & 8 & -4 & -4 & 8 & -8 & 4 \\ 3 & -6 & 10 & -12 & 12 & -10 & 6 & -3 \end{bmatrix} \tag{2.17}$$

2.2.2 Quantization procedure

Similarly to other video standards, the H.264/AVC quantization procedure is performed by using a scalar quantizer. However, the adopted quantizer significantly differs from the ones implemented in previous ITU-T and ISO/IEC MPEG video standards, implementing a more complex

non-linear quantization function. This allows the video encoders not only to maximize the trade-off between the bit rate and the picture quality but also to more accurately manage it. To achieve such goal, the H.264/AVC quantizer comprehends 52 different $Qstep$ values, in the range between 0.625 and 224, that are distanced apart by $\sqrt[6]{2}$, i.e. 12%. These steps are indexed by a Quantization Parameter (QP) that is common to all the coefficients of a transformed block[2]. Moreover, any value of the $Qstep$ function can be derived from its first 6 values using Equation 2.18. As a result, a wide range of quality levels can be efficiently addressed by this quantization process, since fine control is possible at the lowest quantization levels and coarse quantization is not burdened[120].

$$Qstep(QP) = Qstep(QP\%6) \times 2^{\lfloor \frac{QP}{6} \rfloor} \tag{2.18}$$

The very strong inter-dependencies that the H.264/AVC standard introduced between the transform and the quantization procedures also significantly influence the operation of the quantizer. In particular, the complexity of the H.264/AVC quantization operation was greatly increased, due to the incorporation of the SFs that were left over from the improved H.264/AVC transform path, in order to allow the computation of the transforms by solely using integer arithmetic[120]. Nonetheless, the benefits to the whole coding algorithm that resulted from realizing the transform coding operations in integer arithmetic greatly compensate such penalty.

By following the above considerations, Equation 2.19 represents the H.264/AVC quantization procedure, where z_{ij} is a quantized coefficient, y_{ij} is the corresponding transform coefficient, and i and j are the line and column indexes for the considered blocks of coefficients, respectively.

$$z_{ij} = round\left(y_{ij}\frac{SF_{ij}}{Qstep(QP)}\right) \tag{2.19}$$

As it can be seen, despite its simplicity, this formulation considers two relatively complex arithmetic operations: a multiplication and a division, both involving rational numbers. Nonetheless, an alternative representation based on integer arithmetic operations can also be adopted for the H.264/AVC quantization algorithm, as it is presented in[102] and discussed in chapter 5. Such formulation is shown in Equation 2.20 and replaces the division by an integer multiplication and an arithmetic shift-right operation, which greatly simplifies the computational complexity of the quantization operation.

$$z_{ij} = round\left[(y_{ij} \times MF(QP)_{ij} + f) \times \frac{1}{2^{15+\lfloor \frac{QP}{6} \rfloor}}\right] \tag{2.20}$$

In addition, the fusion of all the possible combinations for the ratio between SF and $Qstep$ in a single function (MF) not only allows to speed up the computation procedure but also to significantly reduce its memory requirements. In fact, Look-Up Table (LUT) implementations of this non-linear function MF for 4×4 and 8×8 blocks of coefficients include only either 18 or 36 distinct values, respectively, as it can be seen in Equation 2.21 and Equation 2.22. Note that

[2] In the High Profiles of H.264/AVC[120], it is possible to change the quantization step size within a block of coefficients when the *frequency-dependent quantization* tool is adopted.

$QP\%6$	$n=0$	$n=1$	$n=2$
0	13107	5243	8066
1	11916	4660	7490
2	10082	4194	6554
3	9362	3647	5825
4	8192	3355	5243
5	7282	2893	4559

Table 2.1: Definition of the values for $m_4(QP, n)$.

$QP\%6$	$n=0$	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$
0	13107	11428	20972	12222	16777	15481
1	11916	10826	19174	11058	14980	14290
2	10082	8943	15978	9675	12710	11985
3	9362	8228	14913	8931	11984	11259
4	8192	7346	13159	7740	10486	9777
5	7282	6428	11570	6830	9118	8640

Table 2.2: Definition of the values for $m_8(QP, n)$.

only the top-left quadrant is shown in Equation 2.22, because the distribution of the data in the other three quadrants are identical (likewise Equation 2.21). Such data consist of 14-bits positive integer values and are presented in Table 2.1 and Table 2.2.

$$MF_{4\times4}(QP)_{ij} = \begin{bmatrix} m_4(QP,0) & m_4(QP,2) & m_4(QP,0) & m_4(QP,2) \\ m_4(QP,2) & m_4(QP,1) & m_4(QP,2) & m_4(QP,1) \\ m_4(QP,0) & m_4(QP,2) & m_4(QP,0) & m_4(QP,2) \\ m_4(QP,2) & m_4(QP,1) & m_4(QP,2) & m_4(QP,1) \end{bmatrix} \tag{2.21}$$

$$MF_{8\times8}(QP)_{ij} = \begin{bmatrix} m_8(QP,0) & m_8(QP,3) & m_8(QP,4) & m_8(QP,3) & ... \\ m_8(QP,3) & m_8(QP,1) & m_8(QP,5) & m_8(QP,1) & ... \\ m_8(QP,4) & m_8(QP,5) & m_8(QP,2) & m_8(QP,5) & ... \\ m_8(QP,3) & m_8(QP,1) & m_8(QP,5) & m_8(QP,1) & ... \\ ... & ... & ... & ... & ... \end{bmatrix} \tag{2.22}$$

Finally, the formula presented in Equation 2.20 also includes the f parameter to provide a finer control of the quantization procedure near the origin ("the dead zone") for all types of MBs, as it is shown in Equation 2.23.

$$f = \begin{cases} \frac{2}{3}\left\lfloor \frac{QP}{6} \right\rfloor & \text{, if Intra block} \\ \frac{1}{3}\left\lfloor \frac{QP}{6} \right\rfloor & \text{, otherwise} \end{cases} \tag{2.23}$$

However, such formula is only used to quantize the transform coefficients computed in the first level of the H.264/AVC hierarchical transform path[120], i.e. the AC coefficients resulting from the application of the integer DCT to the 4×4 blocks of transform coefficients (see section 2.2.1). For the remaining coefficients of the MB, which consist of all the luma and chroma DC coefficients computed using either a 4×4 or a 2×2 Hadamard transform, the dead-zone control parameter f has to be adjusted to compensate their smaller dynamic range. Such correction factor is shown in Equation 2.24, which presents a general and complete formulation of the H.264/AVC quantization operation. This formulation is valid for the quantization of the AC ($h = 0$) and the DC ($h = 1$)

coefficients of both the Intra- and Inter-predicted MBs. The h value, represented in Equation 2.24, is defined in Equation 2.25, where $H_{4\times4}$ and $H_{2\times2}$ are the 4×4 and the 2×2 Hadamard transforms, respectively.

$$z_{ij} = round\left[\left(y_{ij} \times MF(QP)_{ij} + f \times 2^h\right) \times \frac{1}{2^{15+\left\lfloor\frac{QP}{6}\right\rfloor+h}}\right] \tag{2.24}$$

$$h = \begin{cases} 1 & \text{, if } H_{4\times4} \vee H_{2\times2} \\ 0 & \text{, otherwise} \end{cases} \tag{2.25}$$

In what concerns the inverse quantization operation, the algorithm considered in H.264/AVC also reflects the tighter coupling between the inverse transform and the inverse quantization processes and, naturally, with the improved formulation of the $Qstep$ values. Hence, similarly to what happened with quantization, the complexity of the inverse quantization operation is also higher than in previous video standards, which is mostly due to the same reasons. As it can be seen in Equation 2.26, which formulates this inverse quantization operation, the reconstruction of the quantized data (y) in H.264/AVC requires, fundamentally, two multiplications involving rational numbers: the $Qstep$ and a Pre-scaling Factor (PF) resulting from the inverse transformation procedure[120].

$$y_{ij}^S = z_{ij} \times Qstep(QP) \times PF_{ij} \times 64 \tag{2.26}$$

Naturally, the final multiplication by the constant value 64, which is used only to improve the accuracy of the inverse transforms computation, does not contribute to such complexity augment, since it can be easily implemented using a shift-left operation. Nonetheless, the resulting residue values computed by the inverse transform functions ($\hat{x}_{ij}^S$) reflect this constant value, and thus do not correspond to the desired reconstructed residue values $\hat{x}_{ij}$. Consequently, the $\hat{x}_{ij}^S$ scaled residue values must be adjusted in the last phase of the H.264/AVC decoding procedure, as shown in Figure 2.6, in order to obtain the final reconstructed pixel values. Such operation is defined in Equation 2.27 and consists in dividing by 64 and rounding towards zero the final results of the inverse transform operation.

$$\hat{x}_{ij} = \left\lfloor \frac{\hat{x}_{ij}^S + 32}{64} \right\rfloor \tag{2.27}$$

Equation 2.26 also evidences the quite similar characteristics of the operands involved in the computation of the quantization (see Equation 2.19) and inverse quantization procedures. Consequently, all the considerations mentioned above focusing on the optimization of the quantization operation can also be applied to Equation 2.26, in order to simplify its computation. In particular,

Figure 2.6: Reconstruction of the residue values.

all the possible combinations of the term $Qstep(QP) \times PF_{ij}$ can be precomputed and mapped into LUTs, while the multiplications can be realized in integer arithmetic (some can even be converted into shift-left operations). Such alternative formulation, which introduces no penalty in the Peak Signal-to-Noise Ratio (PSNR) performance[102], is shown in Equation 2.28.

$$y_{ij}^{S} = z_{ij} \times V(QP)_{ij} \times 2^{\lfloor \frac{QP}{6} \rfloor} \tag{2.28}$$

As it can be seen, the optimizations considered in this particular case allowed replacing two multiplications by a shift-left operation and an integer multiplication. In the multiplications, the second operand ($V(QP)_{ij}$) can be obtained from LUTs indexed by QP and the position of the coefficients within the block. Furthermore, similarly to what happens in the quantization procedure, these LUTs only contain either 18 or 36 different pre-computed positive integer values that are encoded using 5-bits, depending on the size of the block of coefficients. Equation 2.29 and Table 2.3 describe the content of the LUT used to process the 4×4 blocks, while Equation 2.30 and Table 2.4 present the corresponding values for the processing of the 8×8 blocks. In Equation 2.30 only the top-left quadrant is shown, since the other quadrants are identical.

$$V_{4\times4}(QP)_{ij} = \begin{bmatrix} \nu_4(QP,0) & \nu_4(QP,2) & \nu_4(QP,0) & \nu_4(QP,2) \\ \nu_4(QP,2) & \nu_4(QP,1) & \nu_4(QP,2) & \nu_4(QP,1) \\ \nu_4(QP,0) & \nu_4(QP,2) & \nu_4(QP,0) & \nu_4(QP,2) \\ \nu_4(QP,2) & \nu_4(QP,1) & \nu_4(QP,2) & \nu_4(QP,1) \end{bmatrix} \tag{2.29}$$

$$V_{8\times8}(QP)_{ij} = \begin{bmatrix} \nu_8(QP,0) & \nu_8(QP,3) & \nu_8(QP,4) & \nu_8(QP,3) & ... \\ \nu_8(QP,3) & \nu_8(QP,1) & \nu_8(QP,5) & \nu_8(QP,1) & ... \\ \nu_8(QP,4) & \nu_8(QP,5) & \nu_8(QP,2) & \nu_8(QP,5) & ... \\ \nu_8(QP,3) & \nu_8(QP,1) & \nu_8(QP,5) & \nu_8(QP,1) & ... \\ ... & ... & ... & ... & ... \end{bmatrix} \tag{2.30}$$

$QP\%6$	$n=0$	$n=1$	$n=2$
0	10	16	13
1	11	18	14
2	13	20	16
3	14	23	18
4	16	25	20
5	18	29	23

Table 2.3: Definition of the values for $\nu_4(QP,n)$.

$QP\%6$	$n=0$	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$
0	20	18	32	19	25	24
1	22	19	35	21	28	26
2	26	23	42	24	33	31
3	28	25	45	26	35	33
4	32	28	51	30	40	38
5	36	32	58	34	46	43

Table 2.4: Definition of the values for $\nu_8(QP,n)$.

It must be noted that the operation represented in Equation 2.28 only allows obtaining valid scaled AC coefficients, since the values depicted in Table 2.3 strictly concern to the pre-scaling factors required for the inverse integer DCT applied to these coefficients. To extend this new formulation to address the reconstruction of both the AC and the DC coefficients for all types of MBs, it is only required that: *i*) the dynamic range is maximized for all transform types; and *ii*) the differences in the quantization step sizes for luma and chroma blocks are processed accordingly. This more generic formula for the inverse quantization operation is presented in Equation 2.31, where the values of δ and τ are shown in Equation 2.32 and Equation 2.33, respectively.

$$y_{ij}^{S} = (z_{ij} \times V(QP)_{ij} + \delta) \times 2^{\lfloor \frac{QP}{6} \rfloor - \tau} \tag{2.31}$$

$$\delta = \begin{cases} 2^{1-\lfloor \frac{QP}{6} \rfloor} & \text{, if inverse } H_{4\times4} \wedge (QP < 12) \\ 0 & \text{, otherwise} \end{cases} \tag{2.32}$$

$$\tau = \begin{cases} 2 & \text{, if inverse } H_{4\times4} \\ 1 & \text{, if inverse } H_{2\times2} \\ 0 & \text{, otherwise} \end{cases} \tag{2.33}$$

2.3 Summary

The block-based transform coding procedure commonly employed in digital video is reviewed in this chapter, with a special emphasis on the transform and quantization operations used in the H.264/AVC standard. This presentation includes not only the motivation for the use of these two tools in lossy compression systems but also an overview of the most relevant approaches that are usually considered in digital video coding.

In what concerns the transformation procedure, different approaches to compute 2-D transforms are presented and the most important properties of a transform (from the compression point of view) are discussed. Furthermore, the set of transforms that are most frequently adopted by modern digital video standards, such as the integer DCT, are also reviewed in this chapter.

In a similar way, the overview of the quantization procedure that is provided in this chapter also presents not only the basic concepts and the principles of quantization but also the set of quantization techniques and quantizers most commonly used in video coding, with a particular emphasis on the H.264/AVC standard.

3

State of the Art and Related Work

Contents

3.1 Transform architectures 48
3.2 Architectures for quantization 66
3.3 Summary 75

Transform coding has been an active research topic in the definition of video codecs for several decades. As mentioned in the previous chapters, this is a result not only from the significant impact of this procedure in the compression efficiency and video quality that is offered by the video codecs but also from its multiple implications in the design of such systems (e.g. processing rate, hardware cost and power consumption). Consequently, this research effort has addressed the study of new transforms and quantizers to be included in novel video standards (e.g. H.264/AVC, AVS, VC-1 and H.265/HEVC), as well as the development of efficient algorithms and computational systems to support their realization. For all these reasons, the investigation of high performance and hardware/computational efficient transform and quantization architectures are very active research areas nowadays.

In this chapter, it is presented an overview of the most relevant architectures that have been presented in the literature to perform the transform and quantization operations adopted in the ITU-T H.264/AVC recommendation[69], as well as in other relevant digital video standards.

3.1 Transform architectures

The majority of the solutions that have been proposed for the computation of the transforms adopted in digital image and video standards consist of dedicated processing structures targeting efficient VLSI realizations in ASICs. However, some alternative proposals addressing the design of specialized programmable processors[83,84], Application Specific Instruction Set Processors (ASIPs)[109] and, more recently, efficient implementations in reconfigurable platforms[9,55,56,126,132] have also been presented. Independently of the considered platforms and technologies, these architectures generally implement fast and optimized algorithms and are based on high performance processing structures, in order to mitigate the involved complexity constraints and speed up the inherent computations. Such operations can concern the realization either of a single transform or multiple and distinct transforms, whenever multi-transform or multi-standard functionality are desired. As a consequence of this diversity of hardware implementations and functionalities, these designs are commonly classified either as *Dedicated Transform Architectures (DTAs)* or *Multi-Transform Architectures (MTAs)*.

3.1.1 Dedicated Transform Architectures

DTAs consist of specialized hardware structures capable of efficiently supporting the computation of a single transform. Consequently, its use in video coding is usually restricted to the design of a forward or an inverse transform core to be used by a given video codec. Typically, *direct 2-D* processing structures are employed when high performance implementations are desired. These circuits consist of hardware realizations of direct methods to compute 2-D transforms (e.g. based on polynomial transform techniques[42]) and are characterized by their relatively high paral-

lel processing capabilities. Such important property allows this class of designs to simultaneously process several different transform coefficients, and thus to greatly speed up the computation of the corresponding transforms. However, these architectures are also characterized by requiring a significant design effort, due to their irregularity. Such characteristic also greatly reduces their flexibility to support other transforms, and thus their reusability in the design of alternative video codecs. Moreover, they usually impose higher hardware costs and power consumption requirements, owing to the huge amount of hardware resources they involve. The typical cascaded processing design of these architectures (with long depth processing data paths) additionally characterizes them as being long latency processing structures. Altogether, and besides the offered processing throughput, these characteristics pose important constraints in the design of efficient and high performance video codecs, especially when real time operation is required. Consequently, not so many direct 2-D transform architectures have been proposed and the majority of them concern realizations of the classical 8-points type-II DCT[14,15,72,88,118].

The most well known architecture for the computation of the 8-points DCT consists of the butterfly-like computation structure proposed by Chen *et al*[15], which has been successfully used in codecs for the JPEG image standard and for the MPEG-1/2/4 and H.261/3 video standards. Such architecture results from the methodology proposed by Chen *et al* to decompose the $N \times N$ matrix multiplication into two sets of operations with a more reduced complexity, as it can be seen in Figure 3.1. The first set contains the even indexed terms, while the second set contains the odd indexed terms. When $N = 2m$, for any integer m, such decomposition can be continued until the 2-points DCT. As a result of this factorisation, the complexity of the DCT computation is lowered from N^2 to the order of $N \times \log N$. Such reduced amount of operations not only allows to speed up the computation but also to cut down the hardware complexity of the resulting hardware structure. For example, the computation of an 8-points DCT following this approach requires only

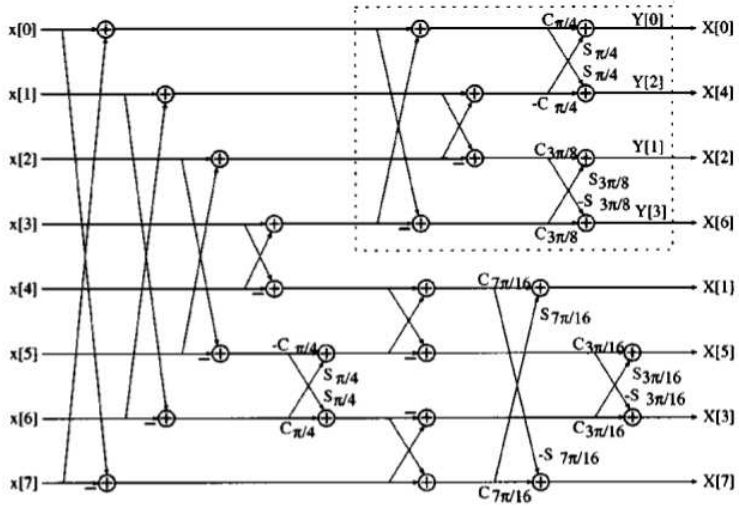


Figure 3.1: Signal flow graph of the 8-points DCT (extracted from[92]).

13 multiplications and 29 additions.

Besides the architecture proposed in[15], other direct 2-D designs targeting more recent standards have also been devised by using quite similar approaches. In what concerns the H.264/AVC standard, three high throughput architectures have been proposed in[19] for fast computation of its mandatory transforms, i.e. the forward and inverse 4×4 DCTs and the Hadamard transform. All these designs were directly derived from the formulations of the three transforms and consist of parallel architectures exclusively based on integer arithmetic adders/subtractors and hard-wired shifts. Since they are capable of computing a 2-D transform in only 2 clock cycles, they can achieve processing rates as high as 746 Msamples/sec when synthesized using a 0.35 μm TSMC Complementary Metal-Oxide Semiconductor (CMOS) technology and operated at about 100 MHz. Kordasiewicz *et al*[79] also used the same methodology to develop an optimized architecture for the computation of the H.264/AVC forward DCT. This highly parallel direct 2-D design computes the whole DCT in a single clock cycle, with a cycle duration closely equivalent to a carry propagation through a 64-bits adder. Therefore, it can process about 2.5 Gsamples/sec when synthesized using a 0.18 μm TSMC CMOS technology and operating with a clock frequency of 100 MHz. Nevertheless, the hardware requirements of such high performance architecture are about four times higher than those of the designs proposed in[19]. Finally, in[12] it is presented a more generic architecture that can be used to compute both the forward and the inverse 4×4 DCTs and Hadamard transforms. By offering a throughput of 8 samples per clock cycle and imposing only 1 cycle of latency, this structure can process the 1080i HDTV (1920×1080 @ 60 fps) and the Digital Cinema Initiatives (DCI) 4k (4928×2048 @ 30 fps) formats in real time, when synthesized using a 0.18 μm TSMC CMOS technology and operated at 50 MHz and 100 MHz, respectively.

In contrast with these architectures, the direct 2-D structures presented in[45,59] implement optimized algorithms for the computation of the 4×4 transforms defined in the H.264/AVC standard. Fan[45] considers matrix factorization techniques with Kronecker product[51] and direct sum to realize the forward 4×4 DCT. As it can be seen in Figure 3.2, such design consists of a 4-stages pipelined structure with 64 adders, which can sustain a throughput of 16 samples per clock cycle. Similarly, block multiplication and permutation matrices are employed in[59] to optimize the computation of the inverse DCT and Hadamard transforms. The resulting processing structure also consists of a direct 2-D design, which is partitioned into three components: a multiplication block and two permutation blocks, consisting only of pure hard-wired connections. The multiplication block can be configured to support the computation of either the inverse DCT or the inverse Hadamard transform, which requires a single clock cycle. When synthesized using a 0.18 μm Chartered CMOS technology, this design is able to process 3.25 Gsamples/sec by using a 200 MHz clock frequency.

Meanwhile, the technique introduced by Chen *et al*[15] has also been frequently considered to

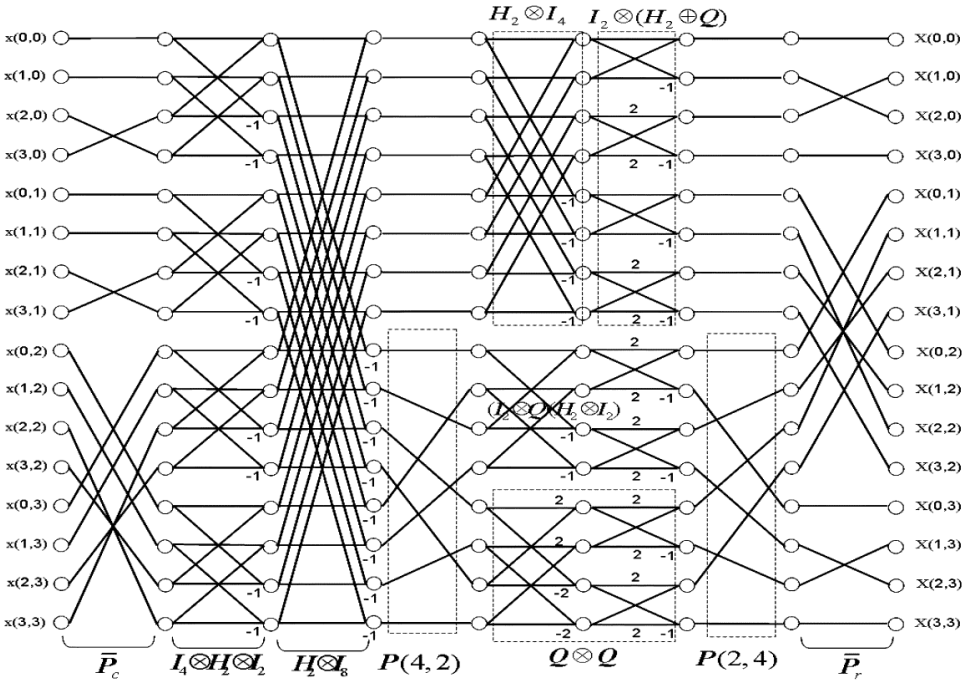
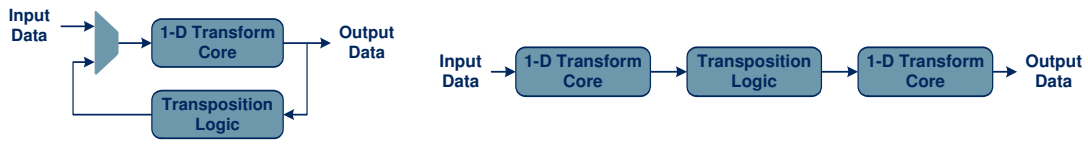


Figure 3.2: Block diagram of the architecture proposed by Fan (extracted from [45]).

develop direct 1-D architectures for the computation of 1-D transforms. In this approach, these simpler processing structures are employed to compute the 2-D transforms by using an indirect method, i.e. the row-column decomposition strategy[7] (see section 2.1.1). Therefore, the considered 2-D transforms are computed in a two steps process and by using either a single or two 1-D transform cores, together with some transposition logic (e.g. a transposition memory, a transposition register array or a transposition switch), as it is shown in Figure 3.3. Such additional logic is often used not only to transpose the 1-D intermediate results but also to feed them into the 1-D transform core responsible for the processing of the column transform, so as to obtain the final 2-D transform results. Therefore, the hardware cost of these designs is relatively lower, since the same 1-D transform processing circuit can be used twice in the computation of a 2-D transform (see Figure 3.3(a)). Moreover, this reduction in the hardware cost may also contribute to decrease the power consumption associated with this type of architectures. Nonetheless, in most practical cases such gains are usually modest, owing to the use of quite large and complex memory based circuits to realize the transposition operation[54]. In addition, the transposition circuit not only intro-



(a) Single core implementation. **(b)** Dual core implementation.

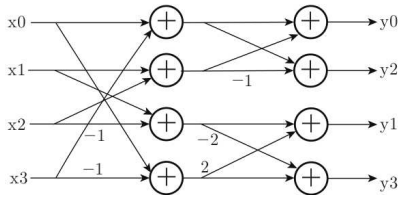
Figure 3.3: 2-D transform architectures based on the row-column decomposition.

duces some latency but also slightly restricts the data processing rate[12,54], which in some cases may compromise the implementation of real time operations. Still, many designs of direct 1-D transform architectures have been proposed for the computation of 2-D transforms.

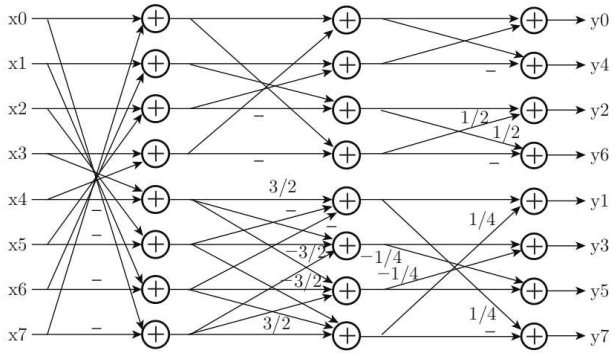
The architectures presented in[16,71,125,152] concern implementations of the 1-D 8-points DCT. Although they share the same functionality, these structures are quite distinct because of the different techniques that were used to improve their computational efficiency. In particular, Coordinate Rotation Digital Computer (CORDIC) arithmetic was used to design the architecture presented in[71], in order to avoid the computation of multiplications and improve the power consumption. Such 1-D transform core consists of a parallel design with 6 CORDIC units, which are combined with a couple of adders and subtractors to provide a throughput of 8 samples per clock cycle. When synthesized using a 0.18 μm Anam CMOS technology and operated at about 60 MHz, this design is capable of providing processing rates as high as 460 Msamples/sec and power savings of about 26%, when compared to the conventional CORDIC-based DCT designs.

The designs proposed in[16,125,152] also avoid the computation of multiplications, but by making use of the Distributed Arithmetic (DA) technique[116,154]. In this case, the multiplications consist of bit-serial computational operations that realize the inner product of a pair of vectors, i.e. the transform kernel values and the transform input data. Since the kernel values are constants, this operation can be performed very fast and by using only bit-level memories and adders/subtractors. For example, the design presented in[152] is able to compute a 2-D DCT in just 136 clock cycles, by using Read Only Memorys (ROMs) and shift-and-add accumulation. This allows to greatly improve the area efficiency, as well as to increase the maximum clock frequency up to 210 MHz for VLSI realizations based on a 0.13 μm TSMC CMOS technology. Similarly, the timing property of the DCT is also exploited in[16] to devise a highly optimized architecture that makes use of only 11 adders to compute a 1-D DCT. The direct 1-D design proposed by Shams *et al*[125], which is based on the New DA (NEDA) algorithm[114], also computes the DCT by exclusively using adders. In this 4-stages pipelined structure, 42 additions are required to compute an 8-points DCT in a single clock cycle. Since the clock cycle corresponds to the delay of a 3-level 4:2 compressor tree, high performance can be easily achieved. For the implementation based on a 0.35 μm technology process reported in[125], the achieved operating clock frequency of 1.5 GHz allows a processing rate of 108 Gbps for image/video sequence data.

Regarding the integer transforms adopted by the H.264/AVC standard, some direct 1-D structures have also been proposed for the computation of both the forward and inverse transformation procedures. In[73], a fast architecture is proposed for the computation of the forward 8×8 DCT defined in the H.264/AVC FRExt amendment. This design consists of a butterfly structure, which was derived directly from the formulation of the considered transform. As a result, it only requires 28 additions and 10 hard-wired shifts to compute the 1-D 8×8 DCT. With a processing rate of 8 samples per clock cycle, this architecture is able to fully satisfy the real time processing



(a) Butterfly structure of the 1-D 4×4 DCT.



(b) Butterfly structure of the 1-D 8×8 DCT.

Figure 3.4: Block diagram of the architecture proposed by Gu *et al* (extracted from [49]).

requirements of the 720p and 1080p HDTV formats when synthesized using a 0.18 μm SMIC CMOS technology and operated with a clock frequency of 300 MHz. This work was extended in [49], where two different pipelined architectures are presented for the computation of the forward 4×4 and 8×8 DCTs. Such designs are based on one 1-D transform core and one register array implementing the row-column transposition operation. As it can be seen in Figure 3.4, the proposed 4×4 and 8×8 transform cores also consist of direct 1-D structures based on the butterfly design. Since the transform coefficients are computed in pipelined processing, the 4×4 and 8×8 DCTs are processed in 5 clock cycles and 21 clock cycles, respectively. Consequently, these architecture are able to comply with the real time requirements of the 4k UHDTV format (i.e. 3840×2160 @ 30 fps) when synthesized using a 0.13 μm SMIC CMOS technology and operated at 300 MHz.

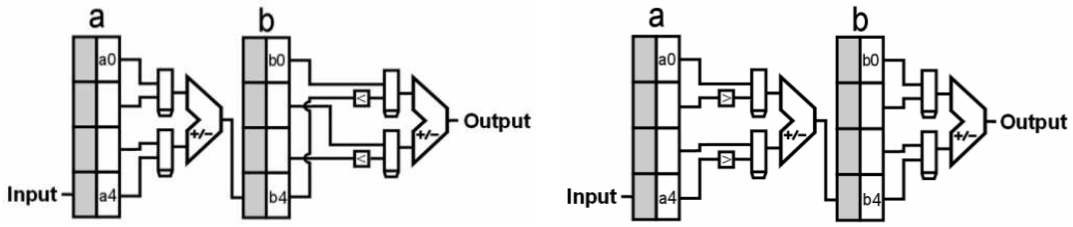
In contrast, the designs proposed by Husemann *et al* [58] include two 1-D transform cores and one register array interconnected in a cascade, in order to reduce the latency in the computation of the 2-D transforms. Moreover, the row-column transposition of the data is embedded in the horizontal transform core, to further reduce the latency of the circuit in about 20%. Due to all these optimizations, these pipelined processing structures require 8 clock cycles to compute the forward 4×4 DCT and 4×4 Hadamard transform: 2 clock cycles for the processing of each 1-D transform and 4 clock cycles to transpose the data. To further increase the processing rate of the architecture, a dual DCT module is also proposed in [58]. This alternative design parallelizes the computation of the 2-D transforms by duplicating the transform computation circuits. Hence, the

throughput is increased from 4 to 8 samples per clock cycle, which enables this parallel design to process about 1.9 Gsamples/sec when implemented in a Xilinx Virtex-II Pro FPGA device (V2P30FF896) and using a 250 MHz clock frequency.

Other alternative cost effective solutions are also described in[1,79], which propose area optimized transform computation units based on serial algorithms. In[79], a programmable serial multiplication module and an accumulator are used to compute the forward 1-D DCT. Although the shift-and-add multiplications and the result accumulation only require one clock cycle, the computation of all the 4×4 transform coefficients takes 112 clock cycles. The extra 6 clock cycles per sample are used for synchronization (2) and to fetch the input data (4). Obviously, such control overhead significantly degrades the performance of the architecture, which can only offer a throughput of 11 Msamples/sec when operated at about 150 MHz and implemented in both a Xilinx Virtex-II Pro FPGA device (XC2VP7) and as an ASIC using a 0.18 μm TSMC CMOS technology.

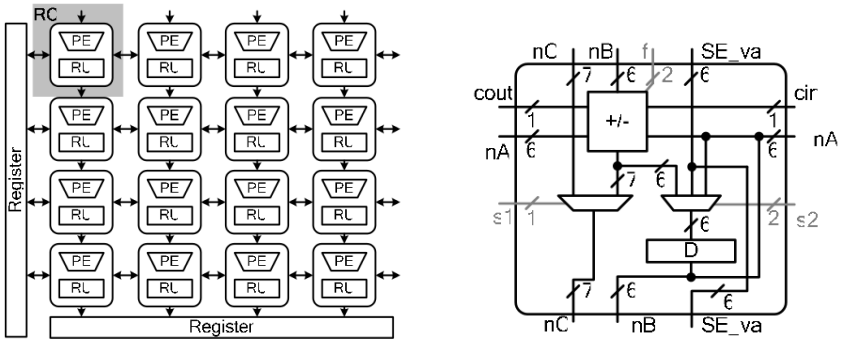
Agostini *et al*[1] also proposed serial 1-D architectures for the computation of the forward and inverse 4×4 DCTs. In these designs, the 2-D transforms are computed by using two 1-D transform units and one transpose buffer. The two transform units are identical and consist of a 2-stages pipelined structure, in which each pipeline stage is composed of one adder, one shifter and two multiplexers, as it can be seen in Figure 3.5. The multiplexers are used to select the correct adder inputs, while the shifter is used to implement the multiplications and the divisions. A finite state machine controls the circuit operation, which requires 8 clock cycles to produce a new output value. Consequently, the global latency for the processing of a 2-D transform is 32 clock cycles: 8 cycles for the computation of each 1-D transform and 16 cycles to perform the row-column transposition of the data. The performance of these designs is also quite modest, since the offered throughput is only of 1 sample per clock cycle. This results in a processing rate of 165 Msamples/sec for implementations on Altera Stratix FPGA devices (EP1S10F484C5) operated at about 200 MHz.

Residue Number System (RNS) based architectures have also been reported for the computation of the H.264/AVC transforms. For example, in[5] the $2^n - 1, 2^n, 2^n + 1$ *moduli set*, with $n = 5$, was considered to develop a 1-D transform architecture for the computation of the forward



(a) Forward 1-D DCT design. **(b)** Inverse 1-D DCT design.

Figure 3.5: Block diagram of the architecture proposed by Agostini *et al* (extracted from[1]).



(a) Reconfigurable cell array architecture. **(b)** Reconfigurable cell design.

Figure 3.6: Block diagram of the architecture proposed by Lo *et al* (extracted from[98]).

and inverse 4×4 DCTs. The throughput of this RNS based transform unit is 4 samples per clock cycle, which allows computing the DCT in just 4 clock cycles. Since all the involved transform computations are performed using 6-bits arithmetic, relatively high processing rates can be obtained with this approach by considering pipelined processing. This processing technique is highly required because of the complexity of the forward and reverse converters that are included in the architecture, in order to convert the data from the binary to the RNS format and vice-versa. Furthermore, these mandatory modules also greatly increase the hardware cost of the architecture, which compromises its use in many applications.

Lastly, systolic array structures for the computation of several different transforms have also been investigated[46,85,98,113,141]. In what concerns the H.264/AVC standard, the design introduced in[98] consists of a 4×4 reconfigurable array architecture that can be dynamically configured to realize either the inverse DCT or the inverse Hadamard transform, as well as to support the decoding procedures of CABAC and CAVLC. As it can be seen in Figure 3.6, the base reconfigurable cells are composed of a processing element and a routing unit, which is used to realize the data transfers with the neighbour processor elements. Each of the 16 processing elements computes one distinct transform operation, by using an accumulator and a set of multiplexers in cascade. Therefore, 64 clock cycles are required for the computation of an inverse 4×4 DCT or inverse Hadamard transform. Hence, the performance and the throughput of this design are relatively modest, despite its relatively high hardware efficiency. In fact, this architecture is only capable of carrying out real time processing of the inverse transform operations up to the H.264/AVC BP @ Level 3.0 (i.e. the 720p HDTV resolution - 720×480 pixels), when synthesized using a 0.18 μm TSMC CMOS technology and operated at 66 MHz.

3.1.2 Multi-Transform Architectures

In contrast with the dedicated transform designs, the majority of the hardware structures that have been proposed for the newest video standards consist of a distinct class of architectures,

denominated as *Multi-Transform Architectures (MTAs)*. When such more elaborated designs are capable of computing the set of transforms employed exclusively by a video standard and by using the same hardware structure (e.g. all the forward and inverse DCTs and Hadamard transforms defined in H.264/AVC), they are usually subclassified as *Unified Transform Architectures (UTAs)*. However, if they are designed to implement the several different transforms used by various standards, they are alternatively subclassified as *Multi-Standard Transform (MST)* architectures. In order to achieve both goals, two different approaches can be considered to devise this type of structures.

The simplest design strategy consists in implementing all the required transforms, by adopting a quite straightforward methodology to design an hybrid architecture: make use of a distinct, independent and specifically optimized hardware structure to compute each of the considered transforms and then select the proper results, according to the type of transform that is required. Ho *et al*[50] followed this approach to design a MST architecture capable of performing the inverse transforms of the H.264/AVC, VC-1, AVS, MPEG-2/4 and H.263 standards. All the 2-D transforms are computed by using the row-column decomposition method, for which a single transform core is employed. As it can be seen in Figure 3.7, such processing structure is composed of three distinct modules: a *separated part*, which includes distinct circuits to compute the multiplications between the input data and the transform coefficients for each of the considered video standards; a *common part* that is shared by all the standards and that is used to perform the multiple additions/subtractions of the transform algorithm; and a *miscellaneous part*, which is used to implement other functions, such as the Hadamard transform and the H.264/AVC inverse 4-points DCT. Due to the use of all these highly specialized circuits, this architecture can support the processing of the Quad Full HD (QFHD) (3840×2160 @ 60 fps) video format in real time when

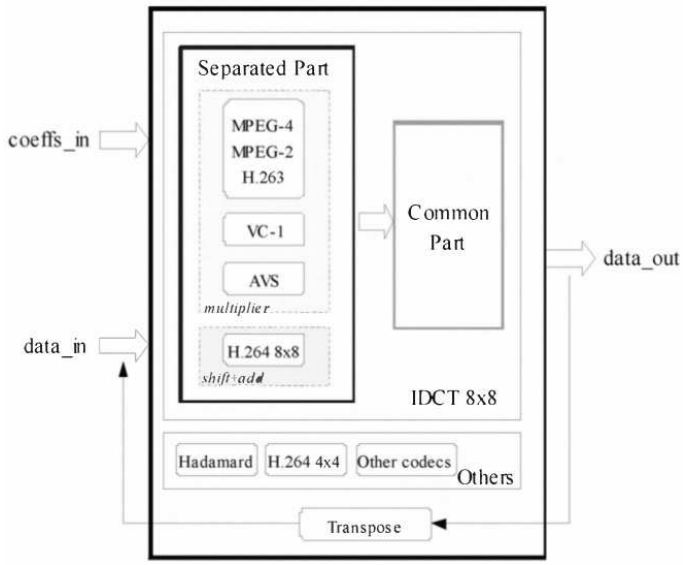


Figure 3.7: Block diagram of the architecture proposed by Ho *et al* (extracted from[50]).

synthesized using a 0.13 μm CMOS technology and operated at 384 MHz. However, such high performance comes at the expense of a relatively high hardware cost. In fact, this is what typically characterizes the MTAs adopting this straightforward design methodology: high hardware cost with increased memory bandwidth and power consumption requirements. These requirements are often prohibitive for implementations targeting platforms with more modest capabilities or stricter design constraints, such as mobile and low power video coding systems. This results from the fact that all the transforms are partially computed (in some cases, totally computed) in parallel and in simultaneous. Nonetheless, most video coding systems do not require the realization of multiple transforms in parallel, since only one transform needs to be computed for each block of data at a time. As a result, almost all the existing MTAs have been developed using the alternative *resource-shared design strategy*.

In this approach, several different techniques can be used to obtain more efficient multi-transform processing structures. For example, *configurable logic blocks* can be used to design reconfigurable architectures with the capability to process transforms with distinct kernels, but of the same size. Gu *et al* adopted this technique to design the UTA for the H.264/AVC standard that is also presented in[49]. Likewise the processing structures for the computation of the 4×4 and 8×8 DCTs, this architecture implements the same row-column decomposition method to process the 4×4 DCT and Hadamard transform. To achieve such goal, it makes use of a direct 1-D reconfigurable transform core, whose architecture consists of a butterfly that uses multiplexers to adaptively select the multiply parameters, and thus the transform being computed. With this straightforward approach, the proposed UTA can satisfy the requirements of real time QFHD (3840×2160 @ 30 fps) encoding when synthesized using a 0.13 μm SMIC CMOS technology and operated at 300 MHz. The high performance UTA for H.264/AVC presented in[13] is also based on this technique. However, this design consists of a direct 2-D processing structure with a throughput of 8 samples per clock cycle, as it can be seen in Figure 3.8. Consequently, it is able to sustain the processing of the 720p HD, 1080i HD and DCI 4k formats in real time when synthesized using a 0.18 μm TSMC CMOS technology and operated at 22, 50, and 100 MHz, respectively. Wei *et al*[140] also adopted a quite similar approach to develop an UTA for the computation of all the H.264/AVC 4×4 transforms. Such direct 2-D processing structure is the result of the combination of the Signal Flow Graphs (SFGs) of the three transforms in a single unified SFG. Hence, it is composed of 4 sets of 8 configurable processor elements, which are interconnected by means of 4 reconfigurable interconnection blocks. In order to support the computation of all the transforms, three types of processor elements with distinct functionalities are also used in the architecture. Moreover, two of these processor elements can have their functionality dynamically configured, according to the type of transform being computed. Despite this enormous flexibility, the proposed UTA computes the forward and inverse 4×4 DCTs and the 4×4 Hadamard transform in a single clock cycle. Consequently, its ASIC implementation using a 0.18 μm TSMC CMOS technology

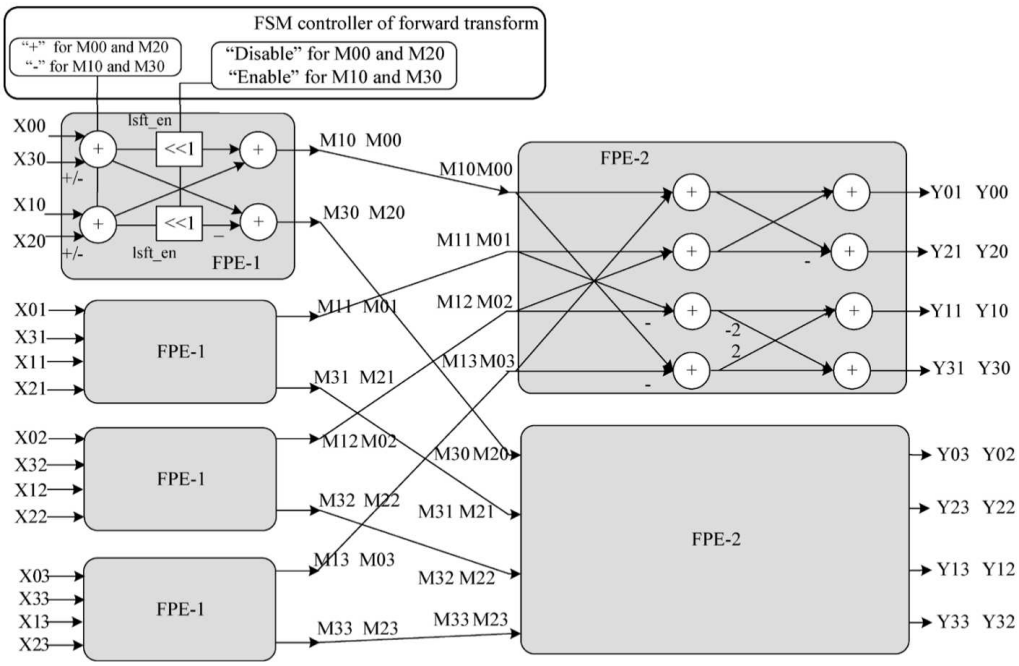


Figure 3.8: Block diagram of the architecture proposed by Chen *et al* (extracted from[13]).

allows to process the DCI 4k format (4096×2048 @ 60 fps) in real time using a clock frequency of 100 MHz.

Li *et al*[90] considered an alternative approach in the development of a parallel UTA supporting all the H.264/AVC 8×8 and 4×4 transforms. Their proposed algorithm is based on *matrix decompositions* and starts by decomposing the kernels of the 8×8 transforms into 4×4 bases. Then, a unified architecture is used to process all the 4×4 transforms. Such processing structure computes the 2-D transforms by following the row-column decomposition approach and by making use of a control block and two specialized units. The 1-D transform unit is capable of computing either one 8×8 or two 4×4 transforms in parallel, for which it includes one butterfly module and two transform operators. These modules strictly realize arithmetic shifts and integer additions. The other specialized unit consists of a linear shift addressing circuit that is applied to the coefficients memory buffer, in order to support parallel access both in the row and column directions. Consequently, the proposed UTA does not include a row-column transposition circuit. With this approach, each 8×8 block is processed in 16 cycles. When synthesized using a 0.18 μm UMC CMOS technology, the data throughput of this UTA can achieve 800 Msamples/s by using a clock frequency of 200 MHz, and thus comply with the real time requirements of the DCI 4k format.

A more elaborated technique based on *multiple matrix factorizations* and *matrix decompositions* was adopted in[44] to derive a cost-effective 1-D transform architecture for the MPEG-1/2/4, H.264/AVC, AVS and VC-1 standards. In each step of the developed algorithm, several heuristic strategies and two guidelines are used to select the proper matrix decompositions. First, a decom-

position form (i.e. the addition form of sparse matrices and/or the cascaded multiplication form of sparse matrices) is chosen to simultaneously maximize the hardware sharing and minimize the amount of mode selection multiplexers. The sparse matrices resulting from these decompositions must be composed of the -1, 0 and 1 elements, or other integers that are a combination of powers of two. Moreover, they should include as much zero elements as possible. With this approach, the proposed structure is also able to process all the transforms using only shift and additions, which further reduces its hardware cost. Regarding the performance, this multi-standard transform architecture supports the encoding and decoding of video in the 1080p HDTV format in real time. Hwangbo *et al*[60] also employed multiple permutations and matrix factorizations to achieve a multi-transform architecture targeting the H.264/AVC high profiles. By using these techniques, it was possible to derive not only the forward 4×4 DCT and the 4×4 Hadamard transforms from the inverse 4×4 DCT but also to integrate them in the computation of the 8×8 DCTs. The resulting hardware structure consists of a direct 1-D transform core based on four independent processing elements, which implement butterfly-adders with shift operations to calculate all the required multiplications. Such core is used twice in the realization of the considered 2-D transforms, which are computed using the row-column decomposition approach. The throughput of the proposed multi-transform architecture is 8 samples per clock cycle, which enables the processing of the 4k HDTV format (3840×2190 @ 50 fps) in real time for ASIC implementations based on a 0.18 μm UMC CMOS technology operated using a clock frequency of 200 MHz. More recently, Shen *et al*[127] also adopted these two techniques to design a unified VLSI architecture for the computation of the 4×4 and 8×8 inverse transforms adopted by the MPEG-2/4, H.264/AVC, AVS and VC-1 standards, as well as the inverse 16/32-points DCTs that are defined in the emerging H.265/HEVC recommendation. This multi-standard architecture implements a fast computational algorithm, which applies recursive division and matrix factorization to incorporate the smaller transforms in the computational circuit of the larger transforms. Such direct 1-D core consists of multiple permutation and butterfly modules composing a pipelined datapath with 5 stages, in order to maximize the clock frequency. To obtain a good area efficiency, regular integer multipliers are reused by the several different butterfly modules that are used in the computation of the different transforms. However, this approach limits the offered throughput to only 4 samples per clock cycle. Still, the proposed MST architecture is able to support the processing the 4k UHDTV format in real time, when synthesized using a 0.18 μm CMOS technology and operated using a clock frequency of about 200 MHz. Such scenario considers the row-column decomposition method, where two 1-D cores and a SRAM based transposition memory are interconnected in a cascade for the computation of the 2-D transforms.

The *delta matrix mapping* technique is employed in[138] to devise a resource-shared architecture for the computation of the inverse 8×8 DCTs defined in the JPEG, MPEG-2, H.264/AVC, AVS and VC-1 standards. The rationale behind this technique consists in computing all the trans-

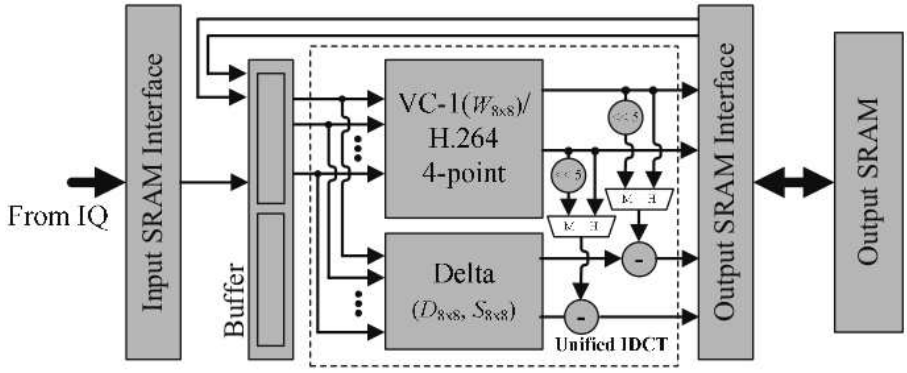


Figure 3.9: Block diagram of the architecture proposed by Chang *et al* (extracted from[11]).

form coefficients using a founding kernel, whose values are obtained by reducing the coefficients of all the transform kernels to a set of common factors, and then compensating the differences of the remaining transform kernels. Wahid *et al*[138] considered the AVS kernel as the founding matrix of all the transform computations and devised specific *delta* matrices to compensate the differences for the VC-1 and H.264/AC standards. Moreover, they developed scaling matrices to allow the computation of the JPEG and MPEG-2 transforms based on the same founding matrix and delta matrices for the VC-1 and H.264/AVC standards. To reduce the hardware cost of the proposed 1-D transform architecture, all the values of these matrices were properly and jointly factorized and the multiplications were designed to be performed using shift-and-add arithmetic. The decoding capabilities of this low cost design satisfy the real time requirements of the 1080p HD video format. Chang *et al*[11] also considered the delta matrix mapping technique to develop the multi-standard architecture depicted in Figure 3.9, which performs the inverse transforms defined in the MPEG-4, H.264/AVC and VC-1 standards. However, in this case those 2-D transforms are computed using the row-column decomposition strategy, for which a single 1-D unified IDCT core and a SRAM transposition circuit are employed. All the transforms are performed by first using the VC-1 computation circuit. Then, the MPEG-4 transform results can be obtained by shifting such data five bits to the left and subtracting the delta coefficients. Similarly, the H.264/AVC transform results are obtained by subtracting a different set of delta coefficients from the VC-1 results. The required transform results are collected after the completion of all the computations, which requires 798 clock cycles per macroblock. The offered processing rate is 50 fps for the 720p HD format, by considering a synthesis procedure using a 0.18 μm CMOS technology and a clock frequency of 150 MHz. Martuza *et al*[104] combined the delta matrix mapping technique with multiple matrix factorizations to design a MST architecture for the processing of the JPEG, MPEG-2, H.264/AVC, VC-1 and AVS inverse transforms. In this approach, the 8×8 transform kernels are first split into two small 4×4 matrices, by applying permutation techniques. Then, multiple matrix factorizations based on sparse matrices were employed to define two adaptable 4×4 founding kernels suitable for the computation of all the inverse transforms of the last three standards. Finally, a delta mapping scheme was derived to enable the computation of the JPEG and

MPEG-2 inverse transforms from the VC-1 processing circuit. To reduce the hardware complexity and maximize the hardware efficiency, all the multipliers were replaced by adders and shifters and the whole hardware resources are shared for the processing of all the transforms. Consequently, only the computational units associated to the realization of one inverse DCT are activated at any given time instant. When synthesized using a 0.18 μm CMOS technology and operated at about 200 MHz, this design is capable of decoding video bit streams in the WQXGA format (2560×1600) in real time.

Lai *et al*[81] employed another technique to develop a multi-standard architecture capable of supporting the computation of the inverse transforms defined in the MPEG-1/2/4, H.264/AVC and VC-1 standards. These 2-D transforms are computed using the row-column decomposition strategy, by employing two identical 1-D transform cores and a register-based transposition circuit. Such transform cores follow the NEDA architecture approach, in order to avoid the use of ROMs and multipliers, and thus optimize the hardware cost and reduce the power consumption. Moreover, their internal hardware structure can be configured to operate in two distinct modes: mode 1 is used to perform the inverse MPEG-1/2/4 8×8 DCT, the H.264/AVC inverse 8×8 DCT and the VC-1 inverse 8×8 and 8×4 transforms, while mode 2 supports the computation of the VC-1 inverse 4×8 and 4×4 transforms, as well as the H.264/AVC inverse 4×4 DCT and inverse Hadamard transform. In both coding modes, the architecture is capable of sustaining a throughput of 8 samples per clock cycle. Consequently, it can process the 720p HD, 1080p HD and DCI 4k formats in real time when synthesized using a 90 nm CMOS technology and operated at 6 MHz, 12 MHz and 48 MHz, respectively.

The DA technique was also used by Chen *et al*[17] to develop a new algorithm for the realization of the MPEG-1/2/4, H.264/AVC and VC-1 transforms. The proposed *Common Sharing Distributed Arithmetic (CSDA) algorithm* combines the Factor Sharing (FS) and the DA sharing techniques, in order to efficiently reduce the hardware cost. By expanding the coefficients matrix at the bit level, the FS method first shares the same factor in each coefficient. Then, the DA method is applied to share the same combination of the input among each coefficient position. This allows reducing the number of non zero elements in the multi-standard transform matrix, and thus to minimize the amount of adders in the circuit. The corresponding CSDA-MST architecture that was devised using this approach implements the row-column decomposition method, for which it includes two 1-D CSDA-MST cores and a transposition memory. To optimize the hardware cost, the two 1-D cores differ in their word lengths and the transposition memory is designed using 64 12-bits registers. Since each core has four pipeline stages and the transposition operation requires 8 clock cycles, the latency of the proposed CSDA-MST design is 16 clock cycles. In terms of performance, the 8 samples per clock cycle throughput enables the processing of the DCI 4k format when synthesized using a 0.18 μm TSMC CMOS technology and operated at 160 MHz.

The 1-D MST architecture proposed by Wang *et al*[139] follows a different approach, based on the reconfigurable Multiple Constant Multiplier (MCM) algorithm. This algorithm is used to generate multiplierless MCMs[137] with multiple and distinct outputs, which are used to realize the transform operations involved in all the considered video standards. To achieve such goal, the different standards are supported through static reconfiguration and by using two different fusing strategies, so that the hardware cost of the resulting circuits is optimized for all the video standards. In the proposed 1-D transform design, which addresses the MPEG-2/4, H.264/AVC, VC-1 and AVS standards, these reconfigurable shift-and-add MCMs are used to implement two 4×4 matrix computation blocks. One of these blocks is used to compute all the 4×4 transforms, while the two blocks are used together to compute the 8×8 transforms by adopting the matrix decomposition technique. As it can be seen in Figure 3.10, this architecture also includes a reconfigurable pre/post-processing block to realize the butterfly/permutation operations for the forward/inverse transforms, respectively, and an adder tree block to obtain the sum of the matrix product. Together with a couple of multiplexers, the reconfigurable pre/post-processing block allows to dynamically reconfigure the architecture for the processing of a forward transform or the corresponding inverse transform. Therefore, the offered throughputs are 8 samples per cycle for a forward/inverse 8-points transform and 4 samples per cycle for a forward/inverse 4-points transform. This MTA can decode 1080p @ 60 fps HD video bit streams when synthesized using a 0.13 μm SMIC CMOS technology and operated at 55 MHz.

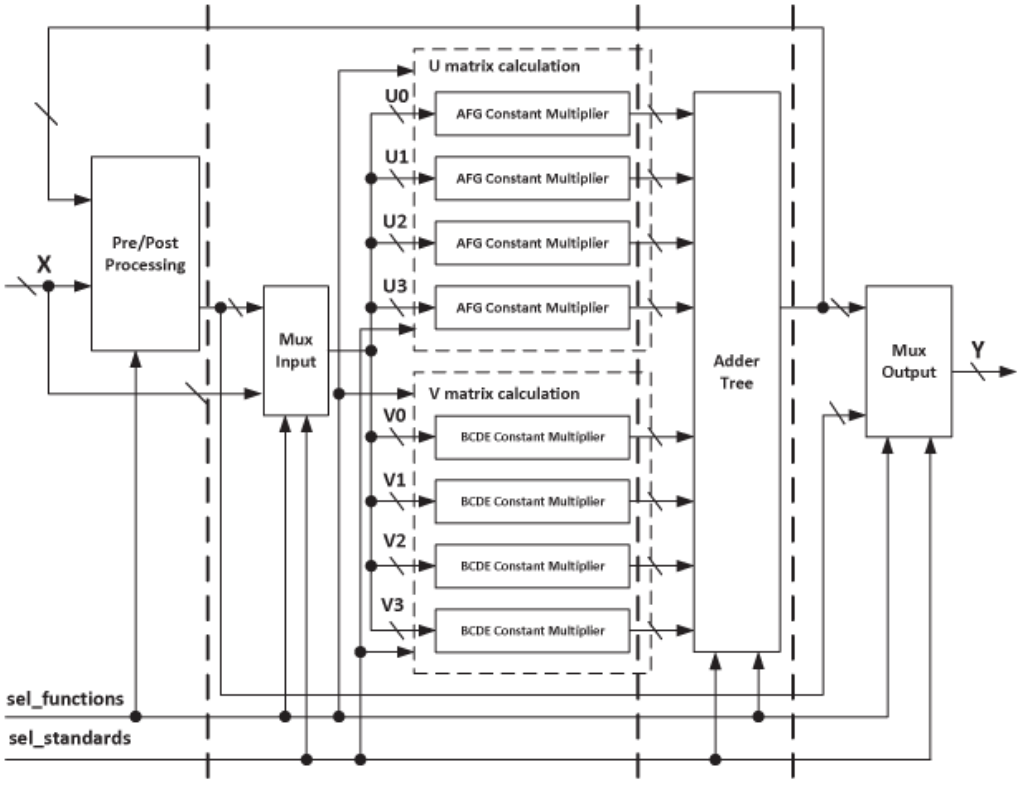


Figure 3.10: Block diagram of the architecture proposed by Wang *et al* (extracted from [139]).

3.1.3 Discussion

The description of the state of the art architectures that is presented in the previous subsections, and that is summarized in Table 3.1, clearly shows that the investigation of specialized and dedicated transform architectures has been a quite active and very important research area in video coding for a long time. In the last few years, such investigation mostly focused on the development of high performance structures capable of processing the long-awaited 720p and 1080p HD video formats for several different application domains, with a special emphasis on the digital TV broadcast and (Internet) video streaming markets. Therefore, the vast majority of the proposed transform architectures were developed and optimized to specifically address the requirements of the H.264/AVC profiles targeting such classes of applications, i.e. the BP, XP and MP.

However, the most recent and forthcoming video encoding and decoding systems contrast with such former trend by presenting significantly different and even more demanding requirements, since they are expected to support the processing of video contents with even higher resolutions (e.g. the 4k UHDTV format), they must be compliant also with other state-of-the-art video standards (e.g. VC-1, AVS and H.265/HEVC) and they are required to provide efficient realizations in new and alternative implementation platforms (e.g. multi-core GPPs, GPUs, or FPGAs). In addition, the low cost and portable nature that characterizes the vast majority of these systems (e.g. smartphones, tablets, video and IP cameras, etc) also calls for the use of more flexible transform architectures with improved hardware efficiency, and that can be easily configured (or even adapted in run-time) to better comply with the aimed performance and power consumption requirements of the considered application domains. Unfortunately, the vast majority of the existing transform architectures fail to simultaneously comply with all these requirements.

Most high throughput solutions offering the capability to process the higher resolution image formats are typically based on very optimized direct 2-D or highly parallel processing structures. Therefore, they usually present huge hardware costs and power consumption constraints, which often restricts their usefulness in the design of low cost and low power video coding systems, such as those typically developed for portable, handheld and other battery supplied devices. In addition, the rather rigid hardware structure of these architectures usually prevents the eventual introduction of modifications or adaptations, in order to allow them to support other video standards and thus to offer the demanded multi-standard functionality.

Conversely, the higher flexibility that is already offered by the existing multi-transform designs comes at the cost of a significant reduction in the provided performance levels. In addition, many of these architectures also duplicate some of their hardware resources to enable the processing of several different standards, which results in poor hardware and power efficiency rates, especially for the processing of UHDTV contents. Naturally, this greatly reduces their attractiveness for the set of applications previously mentioned.

Lastly, none of the reviewed transform architectures can be effectively scaled in run-time and

Table 3.1: Summary of the most relevant designs that have been presented for transform computation. The letters *F* and *I* preceding the transform names in the column *Supported transform(s)* are used to denote a forward or an inverse transform, respectively.

Design	Class	Computation method	Algorithm	Supported standard(s)	Supported transform(s)	Applications
[79] (area)	Dedicated	Direct 2-D	Shift-and-Add Serial	H.264/AVC	$FDCT_{4\times4}$	CIF
[79] (speed)	Dedicated	Direct 2-D	Parallel using butterflies	H.264/AVC	$FDCT_{4\times4}$	4k UHDTV
[45]	Dedicated	Direct 2-D	Parallel using matrix factorizations	H.264/AVC	$FDCT_{4\times4}$	–
[59]	Dedicated	Direct 2-D	Parallel using matrix permutations and block multiplications	H.264/AVC	$IDCT_{4\times4}$, $H_{4\times4}$	4k UHDTV
[1]	Dedicated	R-C decomposition	Serial	H.264/AVC	$FDCT_{4\times4}$, $IDCT_{4\times4}$	1080p HDTV
[73]	Dedicated	R-C decomposition	Parallel using butterflies	H.264/AVC	$FDCT_{8\times8}$	4k UHDTV
[49]	Dedicated	R-C decomposition	Parallel using butterflies	H.264/AVC	$FDCT_{8\times8}$	4k UHDTV
[49]	Dedicated	R-C decomposition	Parallel using butterflies	H.264/AVC	$FDCT_{4\times4}$	4k UHDTV
[58]	Dedicated	R-C decomposition	Parallel using butterflies	H.264/AVC	$FDCT_{4\times4}$, $H_{4\times4}$	4k UHDTV
[5]	Dedicated	R-C decomposition	RNS based	H.264/AVC	$FDCT_{4\times4}$, $IDCT_{4\times4}$	–
[98]	Dedicated	Direct 2-D	Systolic array	H.264/AVC	$FDCT_{4\times4}$, $H_{4\times4}$, $H_{2\times2}$	720p HDTV
[19]	UTA	Direct 2-D	Parallel using butterflies	H.264/AVC	$FDCT_{4\times4}$, $IDCT_{4\times4}$, $H_{4\times4}$	DCI 4k
[12]	UTA	Direct 2-D	Parallel using matrix factorizations	H.264/AVC	$FDCT_{4\times4}$, $IDCT_{4\times4}$, $H_{4\times4}$	DCI 4k
[13]	UTA	Direct 2-D	Parallel using matrix factorizations	H.264/AVC	$FDCT_{4\times4}$, $IDCT_{4\times4}$, $H_{4\times4}$	DCI 4k
[140]	UTA	Direct 2-D	Parallel using matrix factorizations	H.264/AVC	$FDCT_{4\times4}$, $IDCT_{4\times4}$, $H_{4\times4}$	DCI 4k
[90]	UTA	R-C decomposition	Parallel using matrix decompositions	H.264/AVC	$FDCT_{8\times8}$, $IDCT_{8\times8}$, $FDCT_{4\times4}$, $IDCT_{4\times4}$, $H_{4\times4}$	DCI 4k

Continued on the next page.

Table 3.1: Summary of the most relevant designs that have been presented for transform computation. The letters *F* and *I* preceding the transform names in the column *Supported transform(s)* are used to denote a forward or an inverse transform, respectively.

Design	Class	Computation method	Algorithm	Supported standard(s)	Supported transform(s)	Applications
[49]	UTA	R-C decomposition	Parallel using butterflies	H.264/AVC	FDCT$_{4\times4}$, H$_{4\times4}$	4k UHDTV
[60]	UTA	R-C decomposition	Parallel using matrix factorizations	H.264/AVC	FDCT$_{8\times8}$, IDCT$_{8\times8}$, FDCT$_{4\times4}$, IDCT$_{4\times4}$, H$_{4\times4}$	4k UHDTV
[44]	MST	R-C decomposition	Hybrid	MPEG-2/4, H.264/AVC, VC-1, AVS	IDCT$_{8\times8}$, IDCT$_{4\times4}$	1080p HDTV
[50]	MST	R-C decomposition	Hybrid	MPEG-2/4, H.264/AVC, H.263, VC-1, AVS	IDCT$_{8\times8}$, IDCT$_{4\times4}$	4k UHDTV
[127]	MST	R-C decomposition	Parallel using matrix factorizations	MPEG-2/4, H.264/AVC, H.265/HEVC, VC-1, AVS	IDCT$_{32\times32}$, IDCT$_{16\times16}$, IDCT$_{8\times8}$, IDCT$_{4\times4}$	4k UHDTV
[138]	MST	R-C decomposition	Delta matrix mapping	JPEG, MPEG-2/4, H.264/AVC, VC-1, AVS	IDCT$_{8\times8}$	1080p HDTV
[11]	MST	R-C decomposition	Delta matrix mapping	MPEG-2/4, H.264/AVC, VC-1	IDCT$_{8\times8}$, IDCT$_{4\times4}$, H$_{4\times4}$	720p HDTV
[104]	MST	R-C decomposition	Delta matrix mapping and matrix factorizations	JPEG, MPEG-2/4, H.264/AVC, VC-1, AVS	IDCT$_{8\times8}$	DCI 4k
[81]	MST	R-C decomposition	Parallel based on NEDA	MPEG-2/4, H.264/AVC, VC-1	IDCT$_{8\times8}$, IDCT$_{4\times4}$, H$_{4\times4}$	DCI 4k
[17]	MST	R-C decomposition	Parallel based on CSDA	MPEG-2/4, H.264/AVC, VC-1	FDCT$_{8\times8}$, FDCT$_{4\times4}$, H$_{4\times4}$	DCI 4k
[139]	MST	R-C decomposition	Parallel based on the RMCM algorithm	MPEG-2/4, H.264/AVC, VC-1, AVS	FDCT$_{8\times8}$, IDCT$_{8\times8}$, FDCT$_{4\times4}$, IDCT$_{4\times4}$	1080p HDTV

very few can be (re)configured, in order to match the requirements of a particular video coding application or system (e.g. performance, power consumption or the amount of available hardware resources in reconfigurable platforms). In practice, the majority of these processing structures mainly consist of pseudo-reconfigurable structures composed of fixed hardware functional blocks, whose functionality and interconnection can be reprogrammed to support the computation of different transforms. Consequently, the hardware efficiency and the performance that they are able to offer when implemented in FPGA and modern hybrid Central Processing Unit (CPU)+FPGA platforms (e.g. the Xilinx Extensible Processing Platform[22], the Altera Cyclone V System-on-Chip (SoC)[2] or the Intel/Altera Atom E6x5C processor series[21]) is usually very limited.

In conclusion, a new approach needs to be adopted when designing the hardware architectures to be used for the implementation of the transform modules in video encoding and decoding systems. In particular, innovative processing structures should be investigated and proposed, in order to guarantee the success of the next generation of digital video systems for the most common (but often rather demanding) application domains. Such high performance processing structures must be capable not only to efficiently support the transformation procedures defined in the H.264/AVC standard (as well as other upcoming video standards, e.g. H.265/HEVC) but also to process UHDTV contents in real time. Moreover, they should present a very flexible and highly configurable hardware structure, so that they can be easily used to obtain efficient realizations in several different implementation platforms and adapted to the performance, energy and hardware requirements of any video codec or application.

3.2 Architectures for quantization

The quantization procedures that are specified by the latest video standards, such as H.264/AVC, are of relatively high complexity, due to requiring numerous multiplications and divisions involving rational operands (see subsection 2.2.2). For hardware implementations, these computations greatly augment both the hardware cost and the power consumption of the involved processing structures. Moreover, they penalize the performance of such architectures, since the increased computation delay caused by the involved multiplication and division circuits reduces the efficiency of pipeline techniques. For software implementations, the computation of these complex operations also results in poorer quantizer performances, because more clock cycles are spent in their realization. To make matters worse, the very tight interconnection of the transform and quantization procedures further increases the complexity of the quantization module[120]. This is a consequence of the huge amount of memory handling operations and of the high processing rates that the two procedures involve. Altogether, this poses several difficult challenges in the design of modern video codecs, especially in those aiming at the processing of real time and HD video contents.

In order to overcome these constraints, several different specialized hardware structures have been proposed to realize both the forward and the inverse quantization procedures. Such processing structures are commonly classified either as *Dedicated Quantization Architectures (DQAs)*, *Unified Quantization Architectures (UQAs)* or *Integrated Transform and Quantization Architectures (ITQAs)*. *DQAs* consist of processing structures that are capable of implementing only the forward or the inverse quantization procedure. Conversely, *UQAs* support the realization of both procedures, by sharing some of the hardware resources in the architecture to perform the forward and inverse quantization operations. *ITQAs* comprise a completely different class of designs that are able not only to realize the forward and/or inverse quantization procedures but also some (or even all) of the transforms defined in a given set of video standards, by sharing a couple of the involved hardware resources.

Despite this multiplicity of classes, almost all the designs that have been proposed targeting the H.264/AVC standard present a very similar architecture, in which multipliers, adders, shifters and LUTs are combined in a quite straightforward manner to directly implement the desired quantizer. Nevertheless, a couple of alternative implementations have also been presented to simplify the realization of the H.264/AVC forward and inverse quantization procedures, and thus reduce the complexity of its implementation in hardware. In the following subsections, the most prominent H.264/AVC quantizers that are described in the literature are thoroughly reviewed.

3.2.1 Standard architectures for quantization

The design presented in[78] consists of a DQA targeting the H.264/AVC standard, which uses 16 purely combinational quantization units to compute the forward quantization procedure in a single clock cycle. Each quantization unit directly implements Equation 2.20 by interconnecting, in a cascade, one 16-bits binary multiplier, one 30-bits adder and one 31-bits barrel shifter. The constants involved in these computations are stored in two LUTs, whose outputs are shared between the 16 quantization units. This approach allows to slightly reduce the hardware cost of the architecture without compromising its performance. In fact, this high performance architecture is capable of processing the 4k UHDTV format in real time, when synthesized using a 0.18 μm TSMC CMOS technology and operated with a clock frequency of about 68 MHz. Kordasiewicz *et al* later extended their work to design an area optimized quantizer[79]. Such alternative circuit includes only one quantization unit, whose architecture consists of a 4 stages pipelined version of the one previously described. As it can be seen in Figure 3.11, one LUT is used in the first pipeline stage to store all the constant values that depend on QP. The multiplication factors (MF) are provided by another LUT in the second pipeline stage. All the computations are performed in the last two stages, by using the same three arithmetic/logic circuits that are described above. Namely, the multiplication is performed in stage 3, while the addition and shift operations are realized in stage 4. With this approach, Kordasiewicz *et al* proposed an area optimized architecture for the

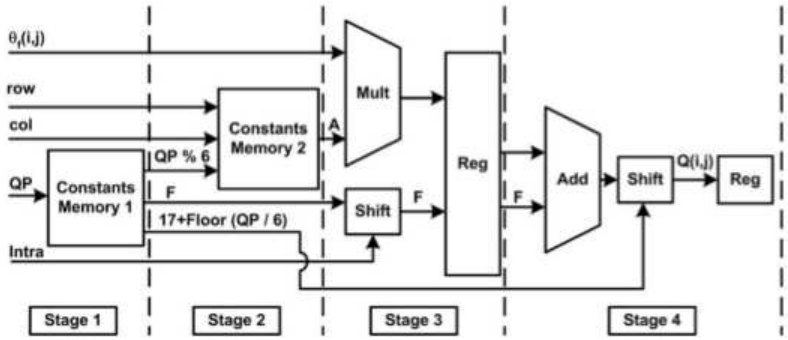


Figure 3.11: Block diagram of the architecture proposed by Kordasiewicz *et al* (extracted from [79]).

computation of the H.264/AVC forward quantization operation, whose maximum processing rate of 86 Msamples/sec (when synthesized using a 0.13 μm TSMC CMOS technology and operated at about 85 MHz) is suitable for low and medium performance encoders.

The parallel design strategy employed by Kordasiewicz *et al* in [78] has been frequently considered by many other researchers, in order to devise fast quantization architectures suitable to be used in high performance coding systems. For example, Elhaji *et al* [43] proposed a forward quantization architecture very similar to the one presented in [78], but using only eight parallel datapaths. Therefore, such structure is capable of simultaneously processing eight coefficients in a single clock cycle. When implemented in a Xilinx Virtex-2 Pro FPGA device, the obtained processing rates are fully compliant with the real time requirements of the 8k UHDTV format, by using a clock frequency of 300 MHz. The architecture presented in [57] consists of another parallel DTA, but that can be used to realize either the forward or the inverse H.264/AVC quantization procedures. Forward quantization is implemented by using four identical computational circuits operating in parallel, as it can be seen in Figure 3.12. Each of these circuits implements a three stages pipeline to compute Equation 2.20 for a different sample of a 4×4 sub-block of transform data. Such computation procedure involves not only the typical binary multiplier, adder, barrel shifter and LUT holding the multiplication factors (MF) but also two auxiliary units that compute the absolute value of the input data and assign the proper sign value to the output data. In addition, two other LUTs holding the offset values (f) and the constants that depend on QP are shared by the four parallel quantization datapaths. A similar approach was also used in the design of the inverse quantization circuit, which implements a pipeline with only two stages owing to the fact that the inverse quantization operation does not involve an offset value. Consequently, the two circuits are capable of sustaining identical throughputs, i.e. four samples per clock cycle, but imposing distinct latencies. Pastuszak *et al* [115] also followed the same parallel design strategy, in order to develop two high throughput architectures for the realization of the H.264/AVC forward and inverse quantization procedures. Both circuits are very similar and consist of 32 identical subunits working in parallel, each one embedding a multiplier, an adder and a shifter. Once again, the multiplicands

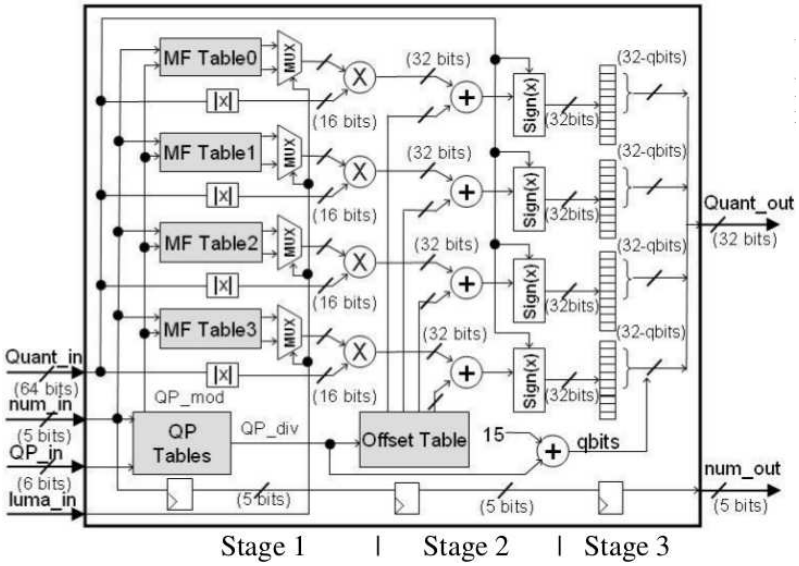


Figure 3.12: Block diagram of the architecture proposed by Husemann *et al* (extracted from[57]).

and the shift values are obtained from LUTs on the basis of the quantization parameter and of the transform type. Since 32 different data values are processed in each clock cycle, one MB can be fully processed in only 8 clock cycles. Consequently, these architectures can also process the 8k UHDTV format when synthesized using a 0.35 μm AMS CMOS technology and operated at about 80 MHz.

The design presented in[87] consists of another parallel processing structure, but that jointly realizes the H.264/AVC forward transform and quantization procedures. This ITQA implements the row-column decomposition strategy to compute any generic 4-points or 8-points 2-D DCT, owing to the fact that the transform kernel values are loaded from external sources. Moreover, the same circuit is used to compute both types of transforms, since matrix partitioning and permutation techniques are employed in the computation of the 8-points transforms. As it can be seen in Figure 3.13, the datapath of the proposed architecture is composed of one TQE array, two hardwired reordering modules implementing all the required pre/post matrix permutations and one multi-purpose buffer, which is used also as a transpose buffer for the computation of the 2-D transforms. Both the 4-points DCT operations and the quantization operations are realized within the TQE array, which includes four dedicated and quite similar computational units to process 16 samples at each clock cycle. Each of these computational units has several multiplexers, adders and binary multipliers. In order to minimize the circuit size, some of the adders are shared in the computation of the transform operations, while the multipliers are shared by the transform and quantization procedures. This time-sharing approach does not impose any extra clock cycles in the realization of the quantization operations, since they are performed using the spare cycles of the transform computation procedures. In fact, when processing the 1080p HDTV format, this architecture is able to process 70 frames per second for 8-points DCT and 121 fps for 4-points

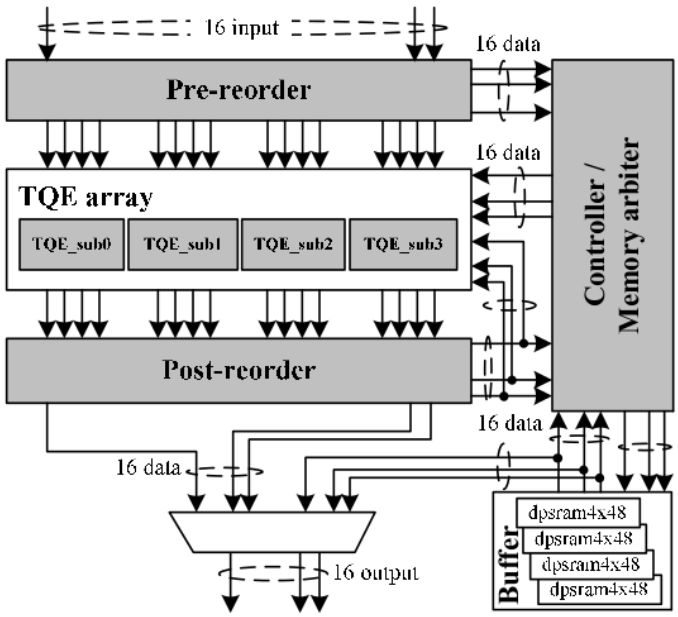


Figure 3.13: Block diagram of the architecture proposed by Lee *et al* (extracted from [87]).

DCT, when synthesized using a 0.13 μm CMOS technology and operated at 100 MHz.

Lin *et al* [94] also proposed combined transform and quantization architectures for H.264/AVC codecs. In these ITQAs, the forward and inverse 4×4 transforms are computed using a direct 2-D butterfly-like computation structure very similar to the one presented in [19]. Forward and inverse quantization are implemented by embedding 8 very similar quantizers into this structure, which is capable of processing 16 data values in parallel in only 2 clock cycles. This is a result of the considered optimized transform computation procedures, in which the input data in related positions are always added prior to the transform computation. Hence, only half of the quantizers are required to compute the whole set of 16 results in parallel. As it can be seen in Figure 3.14, such units were inserted after the first 1-D transform stage for forward quantization, while they precede the 2-D transform circuit for inverse quantization. In what concerns the architecture of the eight forward/inverse quantizers, it consists of a combinational circuit composed of only

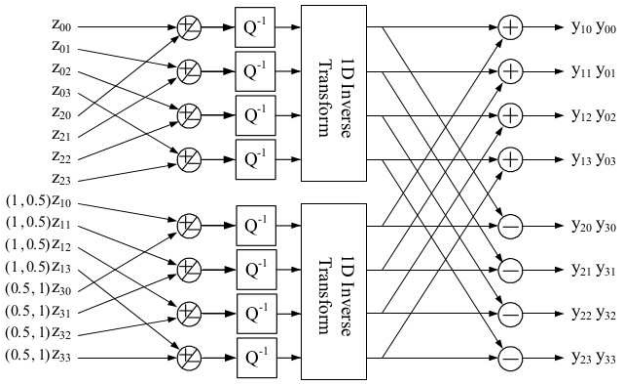


Figure 3.14: Block diagram of the architecture proposed by Lin *et al* (extracted from [94]).

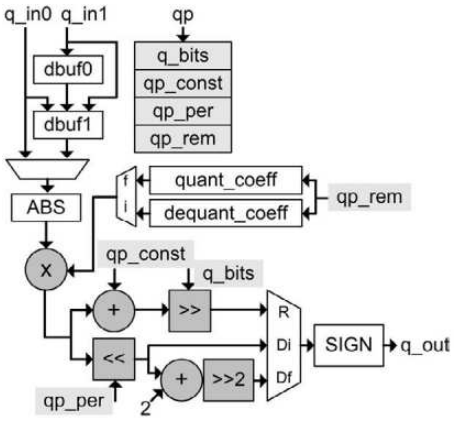


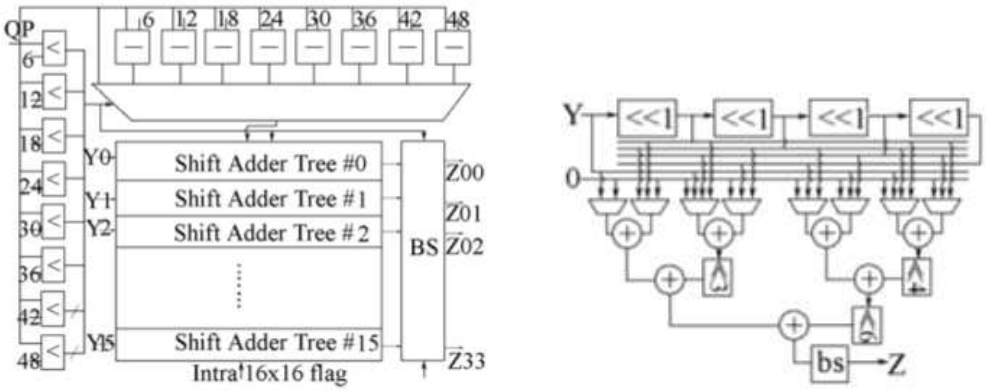
Figure 3.15: Block diagram of the architecture proposed by Lee *et al* (extracted from[86]).

one multiplier, one barrel-shifter and one LUT storing the multiplication factors (MF/V). These authors did not include an adder for the compensation of the offset values (the f parameter in Equation 2.23), in order to simplify the forward quantization procedure. To reduce the hardware cost, a single comparator is used by the 8 quantizers to select the proper multiplication factors and shift amounts for each coefficient.

Lee *et al*[86] adopted a different design strategy to devise a hardware efficient design for the computation of the H.264/AVC forward and inverse quantization operations. The proposed hardware structure consists of an UQA that is capable of processing one coefficient per clock cycle for a given quantization operation, which is specified at run-time by the control unit of the video coding system. As it can be seen in Figure 3.15, such flexible operation mode was achieved by embedding some multiplexers in the datapath of the architecture. These circuits not only allow to reconfigure the datapath to implement either the forward or the inverse quantization procedures at a given time instant but also to share some of the hardware resources in the computation of the involved operations. For example, distinct adders, shifters and LUTs holding the quantization coefficients are employed for the processing of the forward and inverse quantization procedures. However, both procedures share the same multiplier, the LUT used to obtain the constants that depend on QP and the auxiliary units computing the absolute value of the input data and assigning the proper sign value to the output data.

3.2.2 Optimized architectures for quantization

Korah *et al*[77] proposed an architecture for the computation of both the forward and the inverse quantization procedures of the H.264/AVC standard targeting video codecs with modest performance requirements. Such processing structure consists of a reduced complexity and low hardware cost design that only considers the subset of QP values for which $QP \ mod \ 6 \ = \ 4$. In this simplified approach, the coefficients to be quantized or rescaled are first left shifted by different amounts, in order to generate eight partial products. Then, these results are combined



(a) Architecture of the quantizer. **(b)** Shift-adder-tree for Y00, Y02, Y20 and Y22.

Figure 3.16: Block diagram of the architecture proposed by Peng *et al* (extracted from [117]).

in an adder tree to obtain the final product value, which is subsequently adjusted in a shifter. All the shift amounts are determined by a control module based on the value of QP.

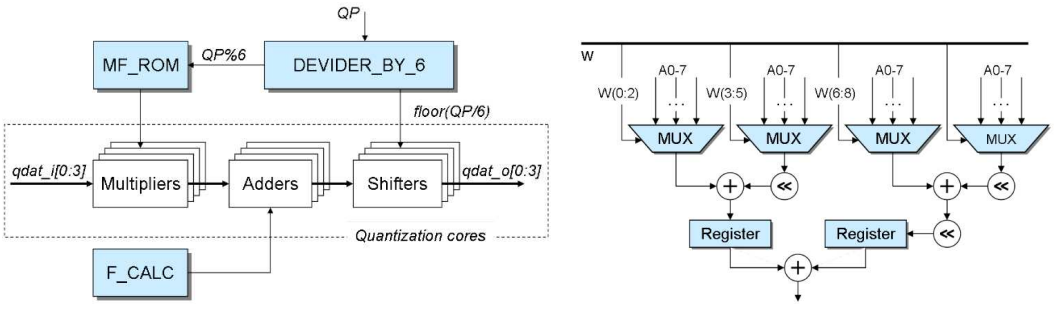
In[117] it is presented another design of a highly optimized forward quantizer that completely avoids multiplications and divisions to compute up to 16 coefficients in parallel. To achieve such goal, the authors devised a parallel architecture composed of 16 shift-adder-tree circuits and one very simple combinational control module. As it can be seen in Figure 3.16, each shift-adder-tree circuit computes a different coefficient value, by using only adders, shifters and multiplexers to perform the necessary multiplications and a barrel shifter to implement the divisions. To enable this processing mode in the proposed architecture, the 52 possible multiplication factors (MF) were encoded as a sum of 2's exponents in the control module. This module also includes several comparators and subtractors to decompose the QP value in two components: a *tail*, which is used to configure the datapath of the shift-adder-tree circuits performing the required operations, and a *head*, which is used to control the barrel shifter. As a result of this simplicity, the processing rates offered by this parallel architecture are compatible with the 4k UHDTV format, when synthesized using a 0.18 μm SMIC CMOS technology and operated at 156 MHz.

Zhang *et al*[153] also proposed an improved quantization architecture, in which the multiplications are computed by using an shift-and-add algorithm that only requires multiple adders interconnected in a cascade topology. To reduce the amount of adders, and thus minimize the hardware cost of the architecture, the authors also modified the values of MF. The proposed MF values are slightly smaller than the original ones, which reduces the necessary precision in their computations. The imposed deviation error is limited within $\pm 6.5\%$ of the original MF values, so that the proposed quantizer is capable of offering a high rate-distortion performance. Nevertheless, the proposed optimizations still introduce a minor mismatch error between the encoder and the decoder. This causes an average decrease in the quality of the encoded video that is less than 0.18 dB. Michael *et al*[107] later improved this technique to reduce the deviation error to a

range of only $\pm 2.4\%$. In addition, the improved MF values allow reducing the hardware cost and the power consumption requirements of the proposed architecture in about 8%, when compared to other similar designs.

Other modifications to the forward quantization equations are also proposed in[112], in order to simplify the hardware realization of this procedure. The suggested changes remove the absolute value and resign operations, by introducing a new set of offset values (f') for the negative transform coefficients. Such values consist of the 2's complement values of f, which allows to merge the offset compensation and rounding procedures in a single operation for all the transform coefficients (see Equation 2.20). The resulting hardware structure consists of a parallel design, which uses 16 multipliers, 16 adders and 16 shifters to simultaneously process 16 transform coefficients. The four possible offset values for a given QP are computed in parallel from a base value of $\frac{2}{3}^{15+\frac{QP}{6}}$ and by using only shift operations and additions. A $(2:1)$ multiplexer is used to selected the correct offset value to be used in the computation of each of the 16 quantized coefficients. Although this hardware redundancy increases the hardware cost of the design, it enables the architecture to achieve higher processing rates. For example, it is capable of fulfilling the requirements of the 4k UHDTV format when synthesized using a 0.13 μm UMC CMOS technology and operated at 140 MHz.

Tran *et al*[131] presented a different type of an area-efficient forward quantization architecture for the H.264/AVC standard, which is based on a specialized multiplication unit. As it can be seen in Figure 3.17, this architecture includes four quantization cores operating in parallel, each one composed of one multiplier, one adder and one shifter. All the combinational circuits and LUTs that are used to generate the required multiplicand and offset values are shared by the four quantization modules. Regarding the multiplier design, it consists of a fast processing structure implementing a conditional multiplier. By using this structure, a multiplication is computed following a 2-steps procedure. First, a pre-multiplier block is used to determine some partial results of the multiplication involving a given multiplication factor (MF). Then, those results are combined using a shift-and-add tree, in order to obtain the final multiplication result. With this strategy, the



(a) Architecture of the quantizer. **(b)** Architecture of the multipliers.

Figure 3.17: Block diagram of the architecture proposed by Tran *et al* (extracted from[131]).

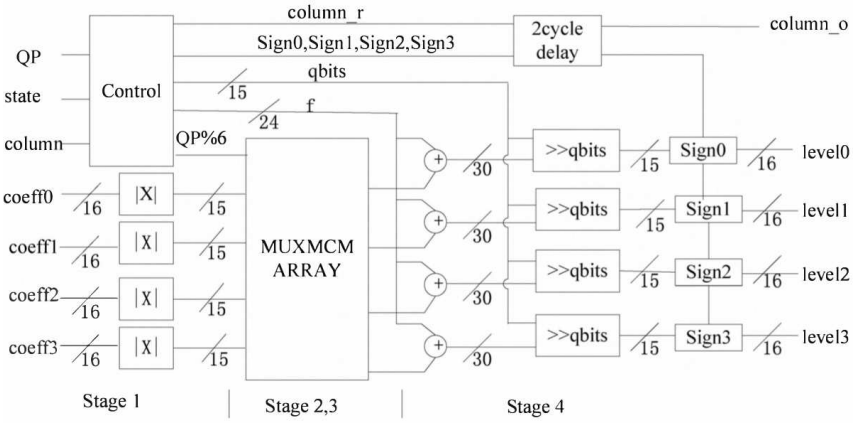


Figure 3.18: Block diagram of the architecture proposed by Ying *et al* (extracted from [86]).

authors claim to have reduced the latency and the hardware cost of the multiplication operation. In what concerns the attained performance, the architecture is capable of sustaining a maximum throughput of 445 Msamples/s that is compatible with the 4k UHDTV format, when synthesized using a 0.13 μm TSMC CMOS technology and operated at 250 MHz.

The quantization architecture proposed by Ying *et al* [149] is also capable of processing four different transform coefficients in parallel, but by following a design strategy based on time-multiplexed MCMs (mux-MCMs) [133]. In this approach, the H.264/AVC 4×4 transform blocks are processed column by column in a 4-stages pipelined datapath. Within this datapath, four identical and independent ways are used to process the four coefficients. As it can be seen in Figure 3.18, each way is composed of the typical absolute value, addition, shifting and resigning circuits. Furthermore, one mux-MCM array is also shared by the four ways to compute all the required multiplications in parallel using fewer hardware resources. This design strategy allows improving the hardware efficiency of the architecture, without compromising its performance. In fact, the proposed architecture is able to comply with the real time processing requirements of the 4k UHDTV format when synthesized using a 0.18 μm SMIC CMOS technology and operated using a clock frequency of 250 MHz.

The multi-standard ITQA proposed by Sun *et al* [130] consists of another multiplierless design, where CORDIC arithmetic was employed to realize the quantization procedures. It can perform the H.264/AVC forward 8×8 and 4×4 transform and quantization procedures, as well as a quantized 8×8 DCT. All the 2-D transforms are computed according to the row-column decomposition technique, for which the architecture includes two identical 1-D transform cores that are interconnected with a transpose memory. Each transform core is capable of processing 8 transform coefficients in parallel, by using only integer shift-and-add arithmetic. The outputs of the second transform core are applied to four CORDIC-Scalers, which scale and quantize the transform coefficients according to the considered type of transform, scaling factor and quantization level. A Post-Quantizer is subsequently used to perform the remaining compensations, i.e. the final

adjustments required for the implementation of the 8×8 DCT using integer arithmetic. In this computational procedure, a CORDIC-Scaler Configurator is also used to retrieve all the required configurations from a bank of LUTs, as well as to dynamically configure the internal architecture of the four CORDIC-Scalers, so that they compute the aimed operation in the desired pipelined computation mode.

3.2.3 Discussion

The conducted survey on state-of-the-art architectures for the computation of the H.264/AVC quantization operations, whose results are presented in the previous subsections and summarized in Table 3.2, revealed that most of the existing processing structures are capable of processing HD video contents in real time. In fact, the obtained results even show that some of these structures are able to comply with the real time requirements of the 4k UHDTV format. Nevertheless, such results also evidenced that very few of the reviewed designs can be used to realize both the forward and the inverse quantization procedures. In addition, they revealed that many of the designs do not share the hardware resources that are involved in the computation of the operations that are common to the two procedures. Consequently, the resulting hardware realizations not only exhibit an increased hardware cost but also a quite poor hardware efficiency.

This study also allowed to conclude that the vast majority of the existing quantization architectures presents a rather rigid hardware structure. This characteristic not only poses serious difficulties when trying to adjust the design to the requirements of a specific video coding system (e.g. in terms of performance, hardware cost or power consumption) but also prevents its adaptation to support other quantization operations defined by different video standards. Therefore, the attractiveness of these architectures for the design of modern video coding systems, where the multi-standard functionality is often a required requisite, is somewhat diminished.

From the presented survey results it can also be observed that not many ITQAs have been proposed, despite the very tight coupling between the transform and quantization operations. The integration of the two operations in a single architecture presents several important advantages in the design of video coding systems, especially for those addressing the most recent video standards, since it allows to optimize several aspects of the codec operation concerning time synchronism and data manipulation. In addition, it can greatly reduce (or even eliminate) the delays inherent to the encoding and decoding procedures, due to external control and memory concurrency issues.

3.3 Summary

This chapter provides an extensive overview of the most relevant dedicated and specialized architectures that were proposed in the last few years to support the computation of the transform and quantization procedures defined in the H.264/AVC standard, as well as in other relevant

Table 3.2: Summary of the most relevant quantization architectures that have been presented in the literature. In the column *Supported operation(s)*, *FQ* and *IQ* denote the forward and the inverse quantization procedures, respectively.

Design	Class	Computation technique(s)	Supported standard(s)	Supported operation(s)	Applications
1080p HDTV[78] (speed)	DTA	Integer arithmetic; Binary Multipliers	H.264/AVC	FQ	4k UHDTV
[79] (area)	DTA	Integer arithmetic; Binary Multipliers	H.264/AVC	FQ	720p HDTV
[43]	DTA	Integer arithmetic; Binary Multipliers	H.264/AVC	FQ	8k UHDTV
[57]	DTA	Integer arithmetic; Binary Multipliers	H.264/AVC	FQ, IQ	DCI 4k
[115]	DTA	Integer arithmetic; Binary Multipliers	H.264/AVC	FQ, IQ	8k UHDTV
[77]	DTA	Integer arithmetic; Shift-and-Add Multiplications	H.264/AVC	FQ, IQ	720p HDTV
[117]	DTA	Integer arithmetic; Shift-and-Add Multiplications	H.264/AVC	FQ	4k UHDTV
[153]	DTA	Algorithmic simplifications; Integer arithmetic; Shift-and-Add Multiplications	H.264/AVC	FQ	-
[107]	DTA	Algorithmic simplifications; Integer arithmetic; Shift-and-Add Multiplications	H.264/AVC	FQ	-
[112]	DTA	Algorithmic simplifications; Integer arithmetic; Binary Multipliers	H.264/AVC	FQ	4k UHDTV
[131]	DTA	Integer arithmetic; Conditional Multipliers	H.264/AVC	FQ	4k UHDTV
[149]	DTA	Integer arithmetic; Multiplications based on mux-MCMs	H.264/AVC	FQ	4k UHDTV
[86]	UQA	Integer arithmetic; Binary Multipliers	H.264/AVC	FQ, IQ	-
[94]	ITQA	Integer arithmetic; Binary Multipliers	H.264/AVC	$FDCT_{8\times8}$, $FDCT_{4\times4}$, FQ, $IDCT_{8\times8}$, $IDCT_{4\times4}$, IQ	1080p HDTV
[87]	ITQA	Integer arithmetic; Binary Multipliers	JPEG, MPEG-1/2/4, H.264/AVC, VC-1	$FDCT_{8\times8}$, $FDCT_{4\times4}$, FQ, $IDCT_{8\times8}$, $IDCT_{4\times4}$, IQ	1080p HDTV
[130]	ITQA	CORDIC arithmetic	MPEG-2/4, H.264/AVC	$FDCT_{8\times8}$, $FDCT_{4\times4}$, FQ	4k UHDTV

digital video standards. Such processing structures are herein analysed and classified according to the type of functionalities they offer. In addition, they are discussed in terms of the benefits and disadvantages that they present in the development of modern video coding systems.

Following this study, the requirements for the design of new high performance and cost effective architectures suitable for the realization of the transform and quantization modules of the next generation of video encoding and decoding systems were identified. Such processing structures should be capable not only to efficiently realize these procedures for the H.264/AVC standard (as well as for other upcoming video standards, e.g. H.265/HEVC) but also to process video contents with even higher resolutions (e.g. the 4k or 8k UHDTV formats) in real time. Moreover, they should present a very flexible and highly configurable hardware structure, so that they can be easily used to obtain efficient implementations in several different implementation platforms (e.g. multi-core GPPs, GPUs, or FPGAs) and adapted to the performance, energy and hardware requirements of any video codec or application.

4

Scalable multi-transform architecture

Contents

4.1 Mapping of the transform algorithms into a systolic array 81
4.2 Proposed hardware structure 84
4.3 Dataflow 96
4.4 Scalability and parallelism 98
4.5 Summary 107

As discussed in chapter 2, the computation of the transforms defined in digital video standards mostly consists of two consecutive $N \times N$ matrix multiplications, where N is the size of the considered 2-D transform. Typically, the value of N is a power of two ($N = 2^n$), which nowadays is in the range between 2 and 32[128,129,143,150]. Therefore, these computations can be performed by using regular algorithms that usually do not require very complex operations, due to the set of integer transform kernels that are generally used (see section 2.2 and appendix A).

Nonetheless, the characteristics of modern digital video applications impose significant restrictions in the design of video coding systems, since they are often required to deal with very high spatial and temporal resolutions in real time. This involves the processing of huge amounts of data in a reduced amount of time. Relevant examples of such systems are portable and handheld devices supporting 4G video telephony or displaying HDTV contents, among others. To make matters worse, currently these systems and applications should also support several different digital video standards (e.g. MPEG-2, H.263, H.264/AVC, VC-1 and AVS), which involves the computation of multiple transforms with distinct kernels, both in terms of their size and coefficient values. As a result of all these constraints and requirements, the design of modern video coding systems and applications still requires efficient implementations (in hardware) of the involved transform computation procedures. Such realizations must be capable of offering high data throughput, high computational rates and low latency, especially when real time operation is considered. In addition, they should preferably present a scalable architecture, as well as result from hardware efficient design approaches.

Systolic array processors[74] offer rather convenient solutions to overcome all of the requisites mentioned above. These hardware processing structures consist of locally connected Processing Elements (PEs) that compute, in a particularly efficient manner, a restricted and very well defined set of operations (e.g. the MAC operations required to compute the transforms employed in digital video coding systems), together with a communication infrastructure that allows exchanging the data under processing among the several PEs by using a pipelined dataflow. Such modular and massively parallel design approach usually requires a rather limited control and simple interconnection circuitry, which allows to obtain highly configurable and scalable hardware structures, often capable of operating using high clock frequencies. Consequently, the obtained designs not only can be easily resized to better adapt its amount of PEs to the computational requirements of the implemented algorithm but also are capable of providing high computational rates and high data throughputs with increased hardware usage efficiency.

In the following subsections, a highly configurable and scalable systolic array processor suitable for the realization of transform cores for H.264/AVC codecs is presented and its corresponding hardware structure is thoroughly described. Furthermore, the scalability and parallel processing capabilities of this innovative Multi-Transform Architecture (MTA) are also discussed, in terms of the advantages they offer to the computation of the H.264/AVC transforms.

4.1 Mapping of the transform algorithms into a systolic array

Although systolic array designs can be derived for most signal processing algorithms, not all the obtained architectures result in efficient, or even feasible, hardware realizations. In fact, due to hardware implementation restrictions, systolic array architectures are only of practical interest when the considered designs are defined using no more than two dimensions[76]. Consequently, designs defined over index spaces with higher dimensions are usually mapped into systolic arrays by applying the set of methodologies first described by Kung[80], which may include the algorithm decomposition into sub-parts, multiple projections, time scaling and delay transfer procedures, among other techniques. This is the case for the computation of most video transforms, whose algorithms are usually defined over a four dimensional indexed space, as it can be seen in Equation 2.1. Consequently, in order to obtain a viable, scalable and both performance and hardware efficient architecture for the computation of multiple 2-D transforms, three different techniques were considered in the development of the MTA herein proposed.

First, the separability property of the transform was applied to Equation 2.1, which represents the generic definition of a 2-D transform, in order to decompose the involved computational procedure in two parts, each one defined over an index space with only two dimensions. Such decomposition is illustrated in Equations 4.1 and 4.2, where x, y, z and c denote the input data, the output data, the intermediate results of the first 1-D transform and a $N \times N$ transform kernel, respectively.

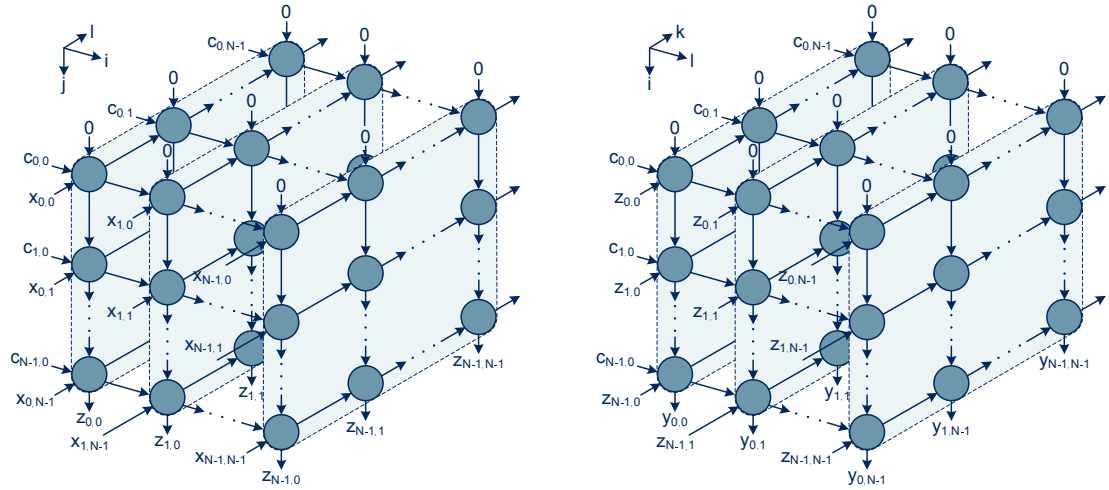
$$z_{il} = \sum_{j=0}^{N-1} x_{ij} c_{jl}, \quad i,l = 0, ..., N-1 \tag{4.1}$$

$$y_{kl} = \sum_{i=0}^{N-1} z_{il} c_{ik}, \quad k,l = 0, ..., N-1 \tag{4.2}$$

As it can bee seen, the first part (Equation 4.1) deals with indices j and l of Equation 2.1 and thus computes a 1-D row-wise transform, while the second part (Equation 4.2) is spawned across indices i and k of Equation 2.1 and computes a 1-D column-wise transform, by using the transposed results obtained from the first part. Hence, the considered simplification results in the well known *row-column decomposition approach*[7].

Then, the transpose property applied to matrix multiplications was considered to further optimize the design of the proposed MTA. Such property was exclusively applied to the computation of the column-wise transform (Equation 4.2), in order to allow the two simpler 1-D transforms to be computed by using exactly the same operations (i.e. MAC) and the same multiplier constants (i.e. the same transform kernel values). To better illustrate this simplification, the final formulations of these two operations using the matrix notation are presented in Equations 4.3 and 4.4, respectively, and depicted in the Dependency Graphs (DGs) shown in Figure 4.1.

$$Z = CX^T \tag{4.3}$$



(a) Row-wise transform (see Equation 4.1) **(b)** Column-wise transform (see Equation 4.2)

Figure 4.1: DGs for the computation of the row-wise and column-wise transforms.

$$Y = CXC^T = C\left\{\left(XC^T\right)^T\right\}^T = C\left(CX^T\right)^T = CZ^T \tag{4.4}$$

A projection and scheduling scheme was subsequently applied to the presented DGs, in order to attain a hardware efficient 2-D systolic array capable of efficiently supporting the computation of both the row-wise and column-wise transforms. Firstly, the processor spaces of the SFGs describing the computation of the two 1-D transforms were derived with two different goals in mind: *i*) to guarantee a similar structure for the two SFGs; and *ii*) to ensure that the function of all its nodes is identical. Accordingly, the DG represented in Figure 4.1(a) was projected in the i-direction, in order to assign all the nodes along the horizontal straight lines in the ij-planes into a single PE. A similar linear projection, using a projection vector in the l-direction, was applied to the DG shown in Figure 4.1(b) to achieve the same goal.

Then, a proper time schedule was specified, in order to break all the data broadcast lines in the SFGs. This not only allowed to obtain the desired systolic processing scheme but also to support the computation of 2-D transforms using the row-column decomposition approach, by using the same processing structure to compute both the row-wise and column-wise transforms and without requiring additional memory circuits to perform the involved row-column data transposition operation. In Figure 4.2 it is depicted the SFG of such hardware structure, which consists of a unified systolic array of the class AB2[76] for the computation of $N \times N$ transforms.

In the presented SFG, the nodes consist of the PEs realizing the operations shown in Equations 4.1 and 4.2 (or Equations 4.3 and 4.4, by using the alternative notation), while the timing is specified by the scheduling vector $\vec{s} = (1, 1)$. The numbers at the diagonal scheduling phases denote the time instant of computation, assuming a regular input dataflow. In such scenario, the processing of a 1-D transform requires $t_{AB2} = (N-1) + N + (N-1)$ time instances, where N time instances are used to effectively compute the results and the remaining $2 \times (N-1)$ time

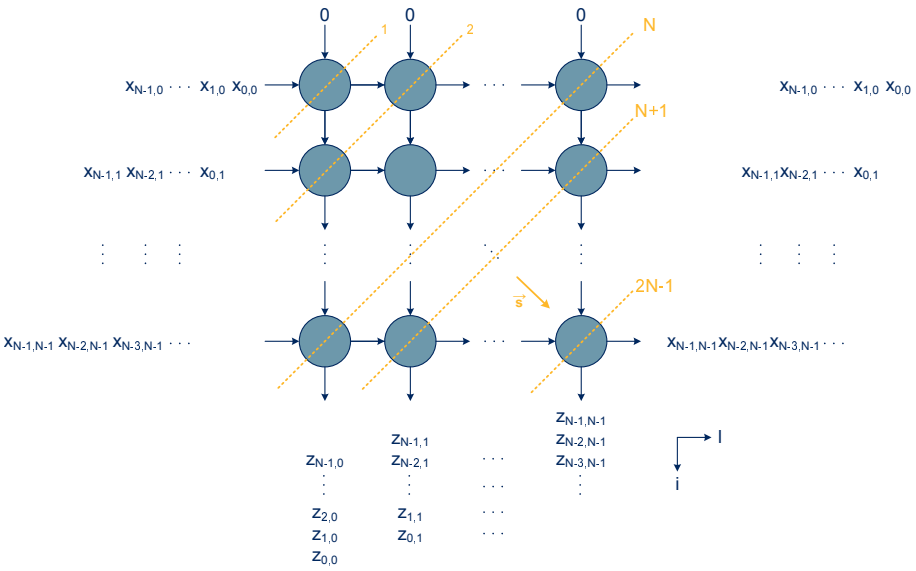


Figure 4.2: SFG for the computation of a 1-D transform.

instances are used to fill in and empty the processing structure. Note that the transform kernel values remain fixed in the computation nodes, although they must be set prior to the transform computation procedure starts.

Finally, a second projection was also considered to further improve the hardware efficiency of the intended architecture, as well as to increase its flexibility. Such enhancement consisted in enabling the processing structure to compute multiple and distinct transforms, by increasing the amount of operations that can be supported by the involved PEs. To achieve such goal, the considered projection allowed assigning to each PE located at coordinates (row, column) in the systolic array, the set of operations realized by all the desired transforms for that specific location. For example, in the H.264/AVC standard such set of transforms may consist of the forward and inverse 8×8 DCT, the forward and inverse 4×4 DCT, the 4×4 Hadamard transform or the 2×2 Hadamard transform [69]. As it can be seen in Figure 4.3, this procedure resulted in the definition of

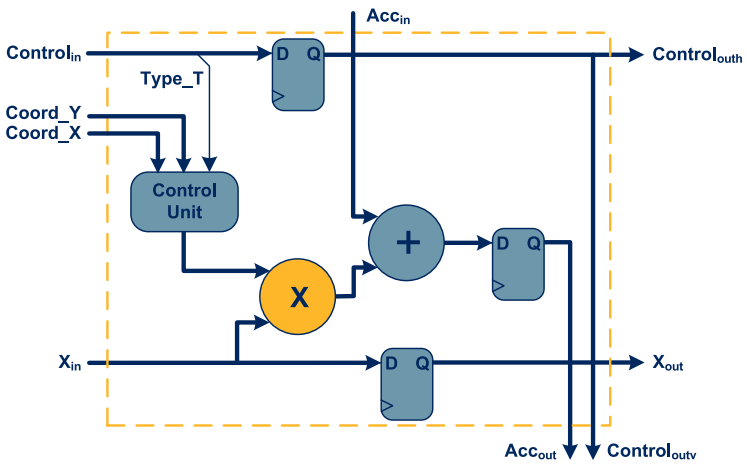


Figure 4.3: Block diagram of a generic multi-transform PE.

a dedicated architecture for the PEs, in which specialized and highly efficient arithmetic circuits are used to compute the required MAC operations and local control units are employed to dynamically select the transform kernel values required for the computation of the desired transform operation.

4.2 Proposed hardware structure

The architecture that was developed to efficiently support the computation of the eight transforms defined in the H.264/AVC standard, i.e. the forward and inverse 8×8 DCT, the forward and inverse 4×4 DCT and Hadamard transforms and the forward and inverse 2×2 Hadamard transforms, consists of a highly configurable and scalable processing structure composed of only four different functional modules, as it can be seen in Figure 4.4. The datapath of such specialized architecture is implemented using the *Input Buffer (IB)*, the *Transform Array (TA)* and the *Transposition Switch (TS)*, which altogether provide the necessary mechanisms to compute all the considered 2-D transforms using the row-column decomposition approach. Nevertheless, the core of the devised architecture is the TA, since it is in this structure that all the transform computations are performed.

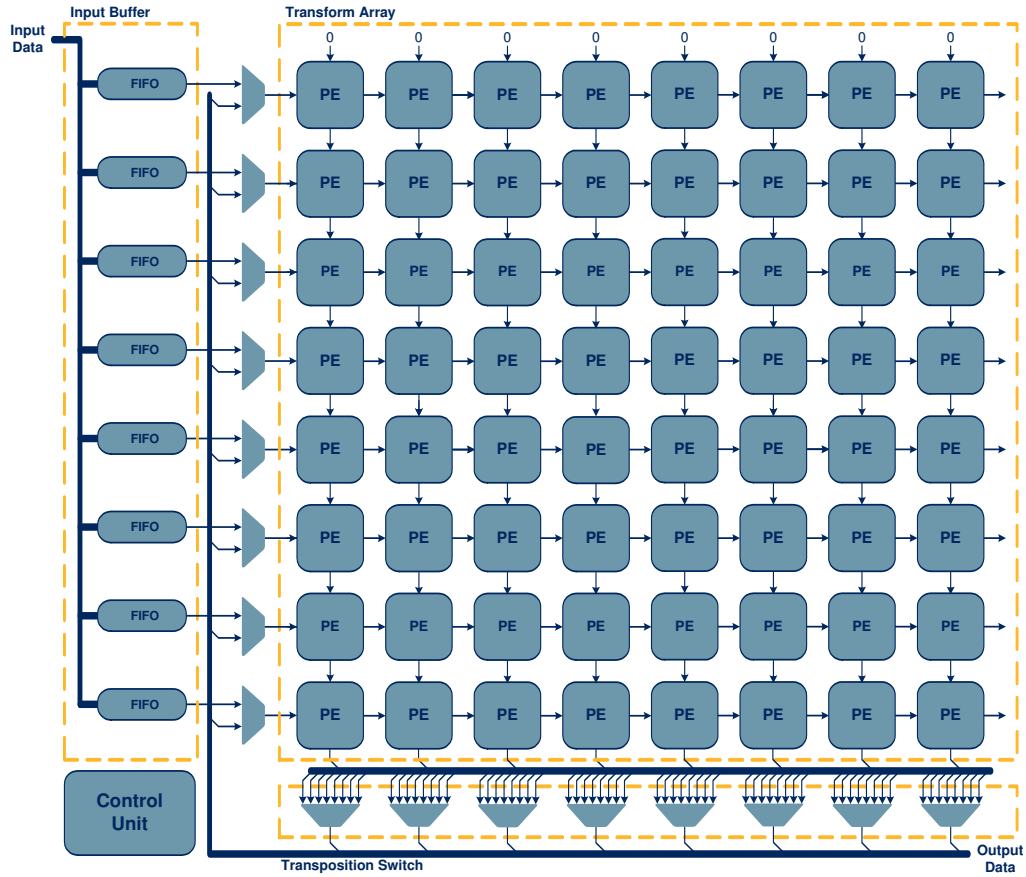


Figure 4.4: Block diagram of the proposed MTA.

4.2.1 Transform array and PEs

In its base configuration, depicted in Figure 4.4, the devised TA consists of an implementation of the 2-D systolic structure presented in subsection 4.1 with 8×8 PEs, which were specifically designed to address all the H.264/AVC transforms. Nonetheless, alternative configurations of this highly configurable and scalable 2-D systolic structure can also be easily obtained, in order to support the computation of different sets of transforms. For example, in order to only compute the set of transforms that are used in the H.264/AVC BP, MP and XP profiles, the TA can be composed of a distinct (and much simpler) class of PEs[24]. The devised TA can also be efficiently used to compute transforms with different sizes, by changing the amount of PEs that are included in any of its two dimensions so that they can match the size of the considered highest order transform. Such important characteristics of the architecture are explained in detail in section 4.4.

Independently of the configuration that is adopted for the TA, all the involved PEs perform the same operations and share an identical architecture, capable of supporting the calculations required by all the considered transforms. The PEs communicate with each other by using small point-to-point interconnections and a simple and reduced signal interface, as it can be seen in Figure 4.5. By using this approach, the local connections between the several PEs that compose the TA can be optimized to meet a target delay (and/or power consumption requirement), therefore allowing to increase the throughput offered by such processing units. Moreover, to minimize the delays resulting from the control operations, and thus to maximize the offered data processing rate, all the control logic that is required for the correct circuit operation was distributed and

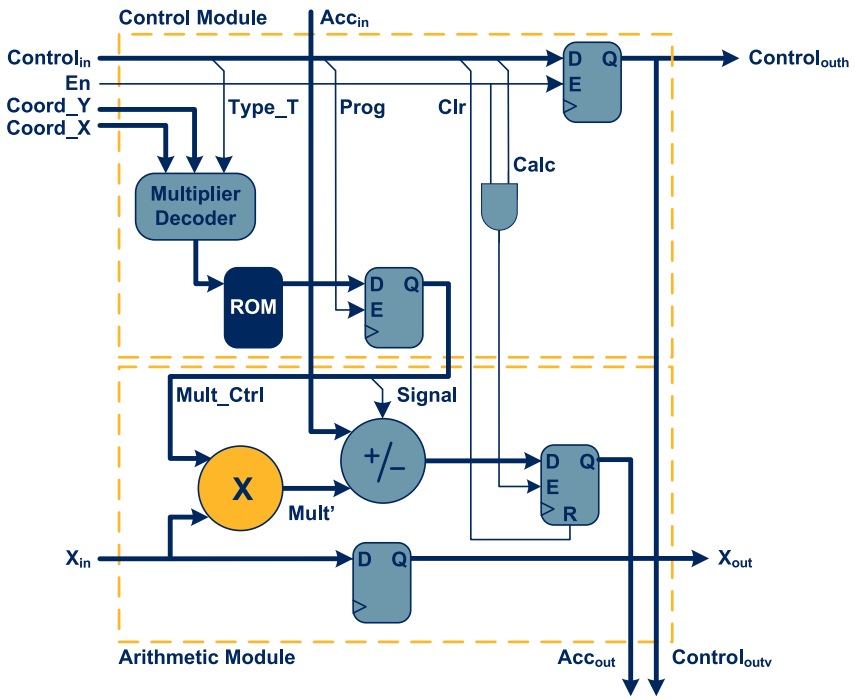


Figure 4.5: Block diagram of the devised PEs for the H.264/AVC standard.

Table 4.1: Encoding of the `Type_T` signal for the implementation of the proposed H.264/AVC multi-transform architecture.

Type_T	Transform
0	Forward 2×2 Hadamard transform
1	Forward 4×4 Hadamard transform
2	Forward 4×4 DCT
3	Forward 8×8 DCT
4	Inverse 2×2 Hadamard transform
5	Inverse 4×4 Hadamard transform
6	Inverse 4×4 DCT
7	Inverse 8×8 DCT

partially embedded in the architecture of the PEs. As a consequence, the internal structure of the PEs is divided in two main modules: the *control module* and the *arithmetic module*.

The *control module* is responsible for guaranteeing the correct flow of all the control signals of the architecture inside the TA, giving support to the desired systolic dataflow model. Furthermore, it is also responsible for generating the control signals that command the transform computation procedure inside the PE. One of the key operations that is realized by this module consists in the generation of the multiplier values (i.e. the transform kernel values) to be used in the MAC operations that are performed by the PE. Such values are generated by taking into consideration the horizontal and vertical coordinates of the PE inside the TA (identified by the `Coord_X` and `Coord_Y` signals), as well as the type of transform that must be computed. The transform type is specified by the `TYPE_T` signal, according to the encoding shown in Table 4.1.

The rationale behind the algorithm that was implemented to generate the kernel values for any given transform results from the observation that only $N-1$ different basis values exist in a $N \times N$ transform kernel[119]. As it can be seen in Equation 4.5 and Equation 4.6, which represent generic 8×8 and 4×4 transform kernels, respectively, such values consist of the first column-vector of the kernel. These basis values correspond to the subset of kernel values matching the angles in the range $\left[0, \frac{\pi}{2}\right]$. Consequently, they can also be used to generate the remaining $(N-1) \times (N-1)$ values of the transform kernel, provided that the symmetry and periodicity trigonometric properties of the cosine function are properly exploited. This is illustrated in Figure 4.6 for the case of the generic 4×4 transform kernel presented in Equation 4.6, where the value located at position (2,2) of the transform kernel $(-a)$ is obtained by reducing the corresponding angle (α) to an elementary angle (α') located in the first quadrant of the trigonometric circle.

$$K_{8\times8} = \begin{bmatrix} a & a & a & a & a & a & a & a \\ b & c & d & e & -e & -d & -c & -b \\ f & g & -g & -f & -f & -g & g & f \\ c & -e & -b & -d & d & b & e & -c \\ a & -a & -a & a & a & -a & -a & a \\ d & -b & e & c & -c & -e & b & -d \\ g & -f & d & -g & -g & f & -f & g \\ e & -d & c & -b & b & -c & d & -e \end{bmatrix} \tag{4.5}$$

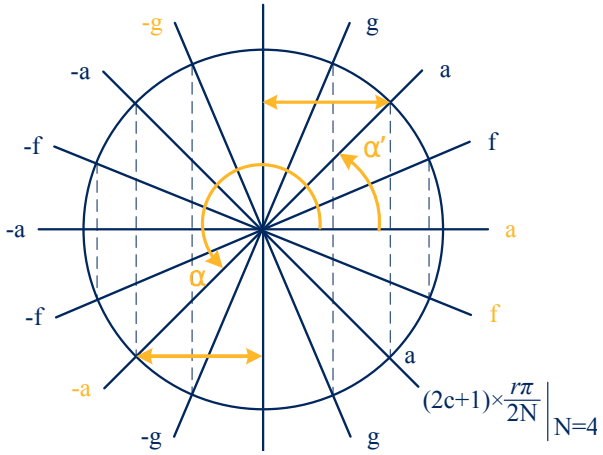


Figure 4.6: Definition of a transform kernel value. Definition of the $-a$ value located at position $(2, 2)$ of the generic 4×4 transform kernel shown in Equation 4.6, according to the proposed algorithm.

$$K_{4\times4} = \begin{bmatrix} a & a & a & a \\ f & g & -g & -f \\ a & -a & -a & a \\ g & -f & f & -g \end{bmatrix} \tag{4.6}$$

To implement the devised algorithm, the control module of the PEs makes use of a quite simple combinational circuit (identified as `Multiplier Decoder` in Figure 4.5) and of a small ROM. Together, they generate all the multiplier values corresponding to all the kernel values of the considered transforms. The operation of such circuit for a $N \times N$ transform kernel is the following.

First, the vertical ($r =$ `Coord_Y`) and the horizontal ($c =$ `Coord_X`) coordinates of the PE inside the TA are used to determine the amount (ω) of $\frac{\pi}{2N}$ angular segments of the angle corresponding to the considered kernel value (e.g. $\omega = (2c + 1) \times r$ for the computation of the row-wise transforms), as specified in the generic formulation of the 2-D DCT shown in Equation 2.2. Although such computations should be performed with integer arithmetic by using $2 \times \lceil \log_2 N \rceil + 1$ bits, only the $\lceil \log_2 N \rceil + 2$ least significant bits of ω are of practical interest, because they identify the corresponding angular value constrained to the range $[0, 2\pi[$. The remaining bits only specify the amount of times the 2π domain is exceeded, and therefore can be ignored. To exemplify this procedure, Figure 4.6 also illustrates the definition of all the angles corresponding to the entries of the generic order-4 transform kernel ($N = 4$) presented in Equation 4.6. In particular, the angles corresponding to all the values in the second column of this matrix ($c = 2; r = 0...3$) are also highlighted in yellow color ($(r, c) \rightarrow \alpha = \omega \times \frac{\pi}{2\times4}$).

The two most significant bits of the obtained constrained result ($\omega' = \omega_{<\log_2 N+1:0>}$) are then evaluated, in order to determine if the corresponding angle is in one of the following three ranges: $\left[\frac{\pi}{2}, \pi\right[$, $\left[\pi, \frac{3\pi}{2}\right[$ or $\left[\frac{3\pi}{2}, 2\pi\right[$. In such cases, the constrained value ω' is computed, which corresponds to a reduction of the original angle into the range $\left[0, \frac{\pi}{2}\right]$. This final result is used to address the ROM, in order to retrieve the required *multiplier control word*. Such data, which corresponds

to the absolute value of the considered transform kernel entry, is then used to control the operation of the specialized multiplication circuit that is embedded in the PE arithmetic module, as it is described below and is represented in Figure 4.5. In addition, the sign information of the kernel value is computed by considering the two most significant bits of ω'. If such data corresponds to an angle in the range $\left[\frac{\pi}{2}, \frac{3\pi}{2}\right]$ ($1 \leq \omega'_{<\log_2 N+1:\log_2 N>} \leq 2$), the considered transform kernel entry represents a negative number. Otherwise, it represents a positive number.

As it can be seen in Figure 4.5, the obtained multiplier control word (`Mult_Ctrl`) and the corresponding sign information bit (`Signal`) are not directly applied to the multiplication circuit embedded in the PE. Instead, such data is stored in an internal data-standing register of the PE, so that it can be used to compute the MAC operations in the subsequent clock cycles. This approach allows to significantly improve the processing rate of the PEs, since it greatly reduces the critical path of the circuit. This aspect is of the utmost importance in the proposed MTA, because all the computations that are performed in the arithmetic module of the PEs are realized using integer arithmetic circuits with a relatively high resolution. This is a result of the increased dynamic gains imposed by the higher order integer transform kernels adopted in the state-of-the-art video standards (e.g. the H.264/AVC forward and inverse 8×8 transform kernels). Nevertheless, this procedure also imposes an explicit PE configuration stage prior to the computation of a new transform, whenever the architecture is reprogrammed to make use of a different transform kernel. Such operation requires only one clock cycle and can be performed in a pipelined fashion within the TA, as explained in detail in section 4.3.

The *arithmetic module* of the PEs makes use of an accumulator and of a specialized multiplication circuit to perform all the required transform operations (see Figure 4.5). In this scope, the data values to be processed ($\mathtt{X_{in}}$) are placed at one of the inputs of the multiplier. Conversely, the partial value of the transform operation being computed (`ACC_in`), which was calculated by an adjacent PE in the previous clock cycle, is placed at one of the inputs of the accumulator. Then, this partial value is updated with the result of the multiplication involving $\mathtt{X_{in}}$ and the kernel value corresponding to the multiplier control word stored in the internal standing-data register of the PE. This operation also takes into consideration the sign information bit stored in the same internal standing-data register. The resulting value is stored in another internal standing-data register before being propagated to the following PEs in the array, in order to shorten the critical path of the architecture and to guarantee the systolic dataflow.

As it can be easily concluded, this highly flexible PE architecture allows a designer to efficiently adapt the functionality offered by the PEs, so that it can support several different combinations of transform kernels. In fact, in order to design a new PE structure that is capable of supporting a given set of transforms, it is only necessary to encode all the involved kernel values in the ROM and develop the corresponding multiplication circuit. In appendix A, this feature is jointly exploited with the modularity and reconfigurable properties of the proposed multi-transform architecture, in

order to to realize several different PEs addressing other state-of-the-art video standards, such as the AVS, VC-1 and H.265/HEVC standards. Furthermore, it is also presented a resource-shared multi-standard PE, which is capable of processing all the transforms defined in these standards, as well as the ones defined in H.264/AVC.

4.2.2 PE architecture for the H.264/AVC standard

In order to fully support the computation of all the transforms defined in the H.264/AVC standard, a PE for the proposed MTA must be capable of computing all the MAC operations required by 8 different transforms (see subsection 2.2.1). This involves 18 distinct transform kernel values in the range of -12 and $+12$. However, only one of these constants is used by the PE to perform the required MAC operation at any moment. Consequently, this opens the opportunity to make use of mux-MCM structures[133], in order to implement reduced-area and faster multipliers for the PE.

Although several different approaches can be considered to develop this class of multipliers based on the shift-and-add algorithm, the addition chain method[6] for multiplying by a constant and a Directed Acyclic Graph (DAG) fusion algorithm similar to the one presented in[133] are adopted in this work. The first technique is used to reduce the number of additions, by allowing the results of the intermediate operations to be shifted and reused in arbitrary subsequent additions. For the considered type of MCM problems, where an input value must be multiplied by one of p given preset constants, the addition chain method is able to provide some potential extra savings, since it allows the sharing of common subexpressions for the computation of the p constants. The considered DAG fusion algorithm is applied in a later development stage, in order to ”fuse” the addition chains corresponding to the several individual constants into a single network of adders, wired shifts and multiplexers, suitable for time-multiplexing. Furthermore, the application of this algorithm also guarantees that the resulting fused addition chain circuit only includes as many adders as the largest of its fundamental addition chains.

In what concerns the development of a PE for the H.264/AVC standard implementing the architecture presented in Figure 4.5, the first step in the design of its mux-MCM consisted in the definition of a DAG, representing an optimal addition chain, for each of the considered kernel values, i.e. the basis values of all the considered transform kernels presented in Equations 2.12, 2.13, 2.14, 2.15 and 2.17. The next design step aimed at finding and exploiting the similarities in all these graphs, in order to obtain the best composite DAG jointly representing the addition chains of all the individual DAGs. Such graph consists exclusively of additions, shifts and multiplexers, as it can be seen in Figure 4.7.

Then, a proper hardware structure was devised to implement the DAG that was obtained in the previous design step. As it can be seen in Figure 4.8, such circuit requires only three adders and four $(2:1)$ multiplexers, in order to perform all the MAC operations involved in the computation of

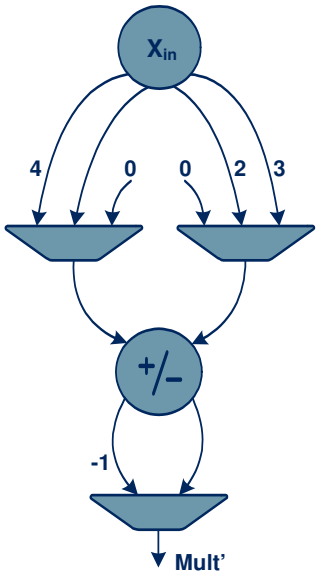


Figure 4.7: DAG of the mux-MCM used in the arithmetic module of the H.264/AVC PE.

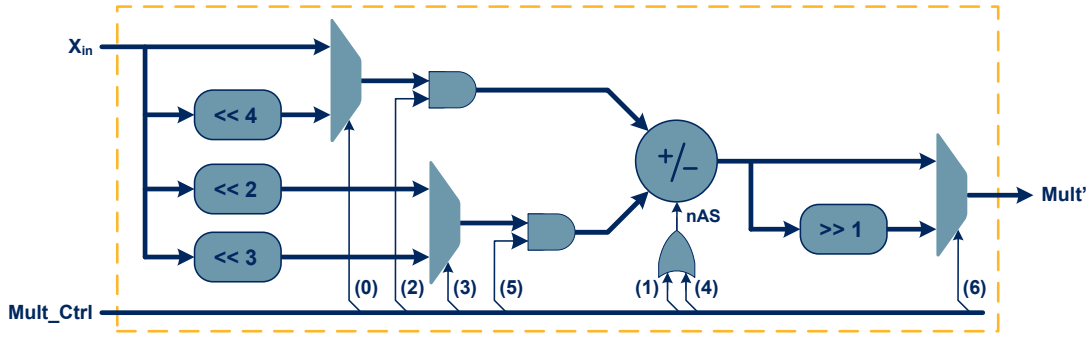


Figure 4.8: Architecture of the mux-MCM used in the arithmetic module of the H.264/AVC PE.

the eight H.264/AVC transforms.

In the following design step, a 7-bits control word was defined for each of the considered kernel values. These 7-bits are used to command the operation of all the multiplexers and addition/subtraction circuits composing the hardware structure obtained in the previous design step. Such control words are presented (in hexadecimal notation) in Table 4.2 for the set of kernel values shown in Equations 2.12, 2.13, 2.14, 2.15 and 2.17.

Finally, the contents of the ROM that is embedded in the control module of the proposed H.264/AVC PE (see Figure 4.5) were specified, by properly disposing all the multiplier control words in six different memory segments. As it can be seen in Figure 4.9, each of these memory segments concerns a distinct transform kernel of order-N, which can be addressed by using the

Table 4.2: Multiplier control words for the H.264/AVC transform kernels (see Equations 2.12, 2.13, 2.14, 2.15 and 2.17).

Kernel Value	$\frac{1}{2}$	1	2	3	4	6	8	10	12
ROM Word	$0x44$	$0x4$	$0x6$	$0x26$	$0x20$	$0x75$	$0x28$	$0x65$	$0x35$

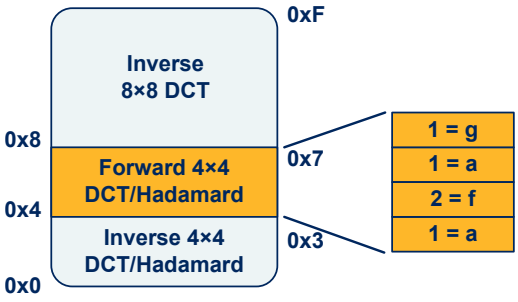


Figure 4.9: Memory map of the ROM used in the H.264/AVC PE.

previously mentioned `Type_T` signal, according to the data presented in Table 4.1. Within each memory segment, N consecutive memory positions are occupied with the multiplier control words corresponding to the N basis values of the considered transform kernel. This data is disposed in the same manner as its corresponding kernel values in the first column of the transform kernel matrix. As a result, in order to address and retrieve a given transform kernel value from the PE ROM, it is only necessary to combine the bits of the `Type_T` signal with those of the address value generated in the `Multiplier Decoder` block (ω'), which represents the offset inside the memory segment. Consequently, by using this memory layout to program the H.264/AVC PE ROM with all the required multiplier control words, it is possible to greatly reduce the complexity of the `Multiplier Decoder` in the PE.

4.2.3 Transposition switch

The *Transposition Switch (TS)* is used to implement a hardwired row-column transposition of the data that has been processed in the TA, which corresponds to the computation of a single 1-D transform. Hence, it provides the necessary mechanisms to support the computation of 2-D transforms using the considered row-column decomposition approach.

Unlike other transposition units that have been proposed[16,49,60,73,96,97,127], the devised TS does not include any Random Access Memory (RAM) modules. In fact, it mostly consists of a set of multiplexers that allow a fast and direct row-column transposition of the data. This alternative design allows saving significant hardware resources for the implementation of the transposition operation, since it avoids the use of the typical memory companion cells. In practice, the functionality of such cells is already implemented by the PEs within the TA, which store such data in their internal standing-data registers (see Figure 4.5). The proposed TS also does not impose any penalty in the performance on the whole system operation, owing both to the pipelined processing nature of the PE array and to the much lower propagation time of its switching circuitry, when compared to the latency of the PEs.

As it can be seen in Figure 4.10, this scalable switching circuit mostly consists of a set of N distinct $(M:1)$ multiplexers, where N and M correspond to the amount of PE lines and columns in the TA, respectively. Typically, $N = M = T$ and T corresponds to the size of the largest

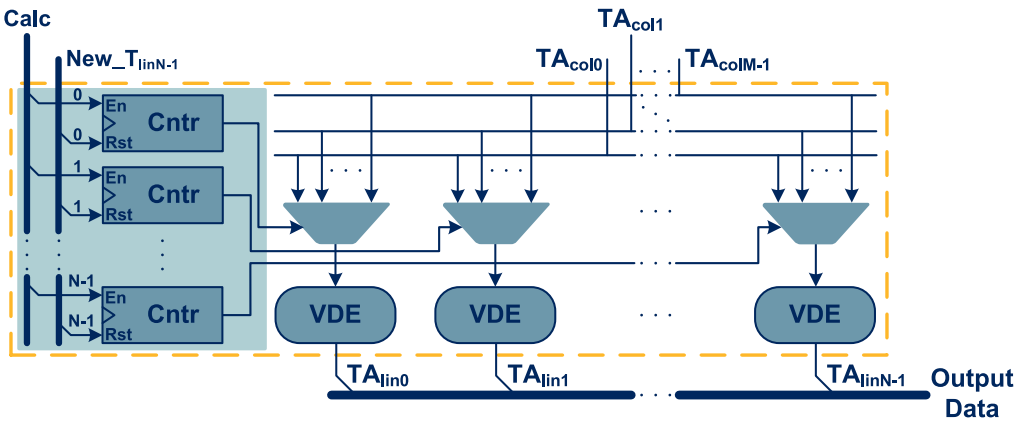


Figure 4.10: Block diagram of the devised Transposition Switch (TS).

transform kernel supported by the TA. This is why there are eight lines and eight columns of PEs ($N = M = 8$) in the base configuration of the proposed MTA targeting the H.264/AVC standard, as it is depicted in Figure 4.4. For configurations of the TA in which $M < N$, the TA additionally includes N distinct Variable Delay Element (VDE) blocks to guarantee the correct dataflow, as it is explained in subsection 4.4.1. In both cases, all the circuits operate in parallel to perform the required row-column transposition of the data, commanded by local control units. Hence, there is a control unit per multiplexer that is used to define the proper value for that multiplexer's selection input on each clock cycle: the i^{th} processed data value output at column j of the TA corresponds to the data input i of multiplexer j, where $i = 0, ..., N-1$ and $j = 0, ..., M-1$.

As a result of this quite simple operation mode, all the local control units can be implemented using only a $\lceil \log_2 M \rceil$-bits up counter. The operation of these counters is commanded by using the set of control signals that are propagated from the TA into the TS, namely, the `New_T` and `Calc` control signals. The `New_T` signal, which is asserted by the MTA control unit whenever the processing of a new 1-D transform is initiated, is used to reset the counter value to zero. Conversely, the `Calc` signal is used to allow the increment of the counter, since it indicates that a new data value is available at the TS input corresponding to that control unit. Note that the `Calc` signal is asserted by the MTA control unit only for the clock cycles in which a new data value is loaded from the IB into the TA. As it was mentioned above, this signal is subsequently propagated inside the TA, to enable the processing of such data by the PEs until it arrives at the TS.

4.2.4 Input buffer

The *Input Buffer (IB)* is used to feed the PEs of the TA either with the residue data resulting from the Intra- and Inter-predictions or the previously computed transform coefficients, according to the type of transform that is being computed, i.e. a forward or an inverse transform. This unit is highly required to minimize the delays when accessing the external data memories where such data is stored, as well as to guarantee the necessary regular dataflow within the TA (as it is

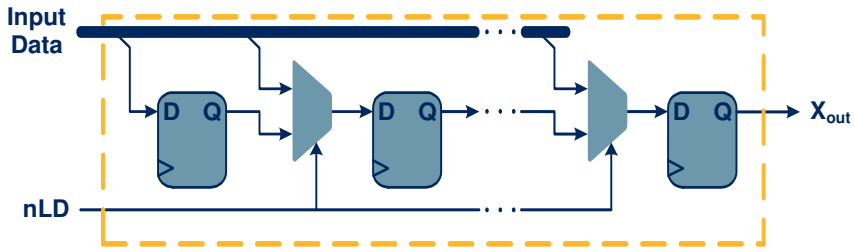


Figure 4.11: Block diagram of the developed Input Buffer (IB) PISO FIFOs.

discussed in section 4.1). As a consequence, the devised IB was designed to operate in parallel with the remaining computational circuits of the proposed MTA, therefore allowing to optimize the overall system data processing rate.

To achieve such goal, the developed IB consists of a scalable structure composed of N distinct First-In-First-Out (FIFO) buffering elements of depth N, which operate concurrently with each other. N corresponds to the size of the largest transform kernel supported by the TA, and thus to the amount of PE lines it includes. As it can be seen in Figure 4.4, with this approach the input data is loaded from the external memory into these FIFOs in a round-robin fashion through the input data port of the IB, which allows to immediately start (or resume) the computation of a transform as soon as new data values are loaded into the empty FIFOs. Furthermore, it also allows to efficiently exploit cache access patterns, since the input data is typically stored in the external memory using the raster scan format. These data values are then serially transferred to the several lines of the TA in the subsequent clock cycles, in order to comply with the implemented pipelined dataflow and in response to the commands issued by the MTA control unit.

To minimize the data loading time, the designed IB also offers two other important features. On the one hand, the size of its input data port can be configured to match the size of the external memory data bus, which allows to optimize the number of memory accesses that are required to efficiently retrieve the input data from the external memory. On the other hand, it is also possible to configure the amount of data values that are simultaneously loaded into the FIFO buffering elements, as a result of such circuits being implemented using Parallel-In-Serial-Out (PISO) shift-registers, as shown in Figure 4.11. Consequently, in the best case scenario, a full line of residue values of a given block (or transform coefficients, for the computation of the inverse transforms) can be loaded into the IB in a single clock cycle, which allows to compute a 1-D $N \times N$ transform in $4 \times N - 2$ clock cycles.

4.2.5 Control unit

The Control Unit (CU) is responsible for controlling the operation of the TA, as well as for commanding both the IB and the TS. It is also in charge of implementing the necessary synchronization mechanisms between the proposed MTA and the outer video coding system that is incorporating this dedicated processing structure. Consequently, in order to simplify the design

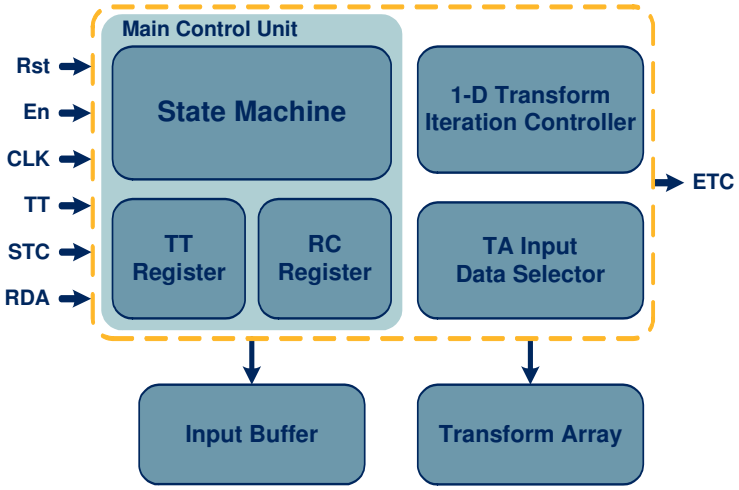


Figure 4.12: Block diagram of the developed Control Unit (CU).

of this controller and guarantee the desired multi-transform functionality of the proposed MTA (namely, support a scalable hardware structure), this design is decomposed into the system main controller and two simpler control circuits, as shown in Figure 4.12. The functionality of each of these controllers, which are synchronized through a quite restricted set of signals, is the following:

i) *the main control unit* manages the overall operation of the MTA and is composed of:

- a state machine, which coordinates the several tasks of the transform computation procedure and the synchronization mechanisms with the outer video coding system;
- the `TT` register, to store the type of the transform being computed during the whole 2-D transform computation procedure;
- the `RC` register, which holds the information regarding the nature of the 1-D transform being computed, i.e. a row-wise or a column-wise transform.

ii) *the 1-D Transform Iteration Controller* is used to assist the control of the computation of the 1-D row-wise and column-wise transforms, by monitoring the amount of transform computations that have been computed;

iii) *the TA Input Data Selector* is employed to control the transfer of the TA input data, by commanding the set of $(2:1)$ multiplexers that select the source of such data, i.e. the IB output when a row-wise transform is being computed or the TS output for the computation of the column-wise transforms.

Table 4.3 summarizes the most relevant signals for the operation of the CU's state machine, which consists of a Mealy sequential circuit with only six different states, as it can be seen in the Algorithmic State Machine (ASM) chart[20] depicted in Figure 4.13.

`Sstdby` is the initial state, which is used to initialize the whole system and to wait for the external command to start the computation of a new 2-D transform, i.e. the assertion of the `STC` signal. As soon as this signal is asserted, the code of the transform type is captured into the `TT`

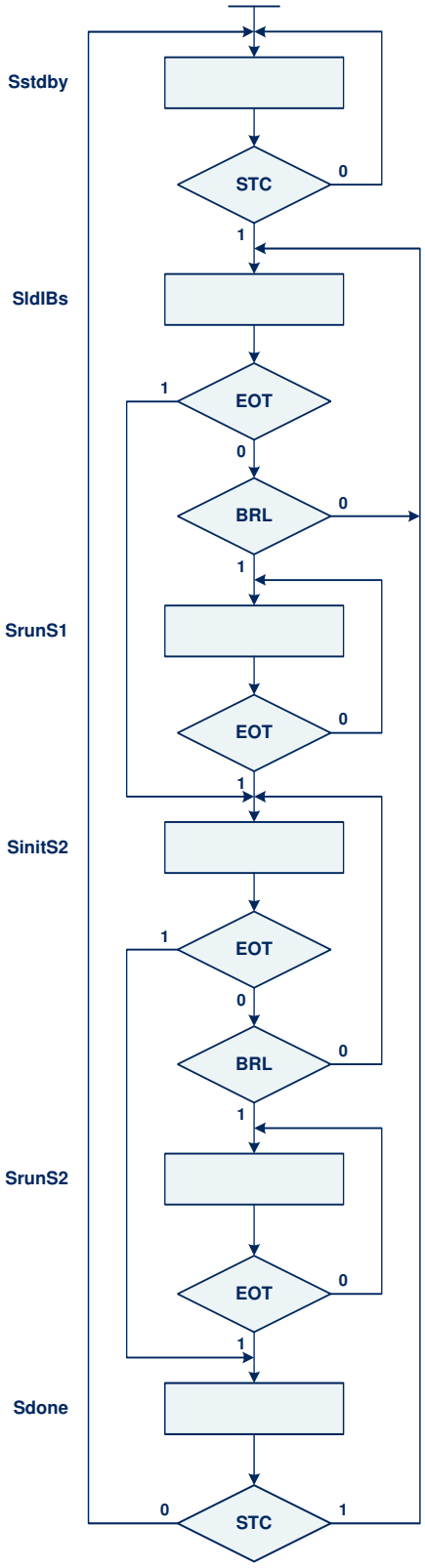


Figure 4.13: ASM chart[20] of the Control Unit's state machine.

Table 4.3: List of the most relevant signals for the operation of the CU's state machine.

Signal	Description
CLK	Clock signal.
En	Enables the normal circuit operation.
ETC	Signals the completion of the computation of a 2-D transform.
RDA	Signals that new data is available in the IB, ready to be loaded into the TA.
Rst	Sets all the circuits into their initial states.
STC	Triggers the computation of a new 2-D transform.
TT	Type of transform to be computed (see Table 4.1).

register and the circuit advances to state SldIBs. In this state, the FIFOs of the IB are loaded with data from the external memory. Moreover, the processing of the row-wise transform is also initiated, owing to the fact that data values are immediately transferred to the TA, as soon as they are available in the FIFOs (see subsection 4.2.1). After all the FIFOs are completely loaded, the circuit goes into state SrunS1, which is used to complete the computation of the first 1-D transform. The state SinitS2 is subsequently used to reinitialize the architecture for the computation of the column-wise transform. Such computations are performed in states SinitS2 and SrunS2. The completion of the 2-D transform procedure is signalled in state Sdone, with the assertion of the signal ETC. Such notification requires a single clock cycle, after which the state machine is put back into its initial state (Sstdby) waiting for a new command to start the computation of another 2-D transform.

It should be noted that this operation loop is slightly simplified whenever the size of the transform to be computed (k) is smaller than the height of the TA (N), where k and N are integer powers of two and $k = 2^i$ with $i = 1, ..., \log_2 N - 1$. In such cases, only k FIFOs must be loaded with input data from the external memory, which reduces the duration of state SldIBs. Moreover, since the 1-D row-wise transform is also simultaneously computed with the loading of such data, the EOT signal is asserted still in state SldIBs. This is also valid for the computation of the column-wise transform, but involving state SinitS2. As a result, the control flow for the computation of a 2-D transform in these particular cases involves neither the state SrunS1 nor the state SrunS2.

4.3 Dataflow

The dataflow model of the proposed MTA was designed not only to maximize the data processing rate within the TA but also the occupancy rate of the PEs. Consequently, it employs a simple and distributed control scheme, which enables the computation of all the transform coefficients in a pipelined fashion and without any stalls. Figure 4.14 illustrates such dataflow for the processing of a 2-D $N \times N$ transform using a TA with $N \times N$ PEs. The three represented data-sets correspond to the processing of two consecutive $N \times N$ blocks: the two data-sets of predicted residue values (or transform coefficients, for the computation of inverse transforms) are represented using a solid-line, while the data-set comprehending the intermediate values of the

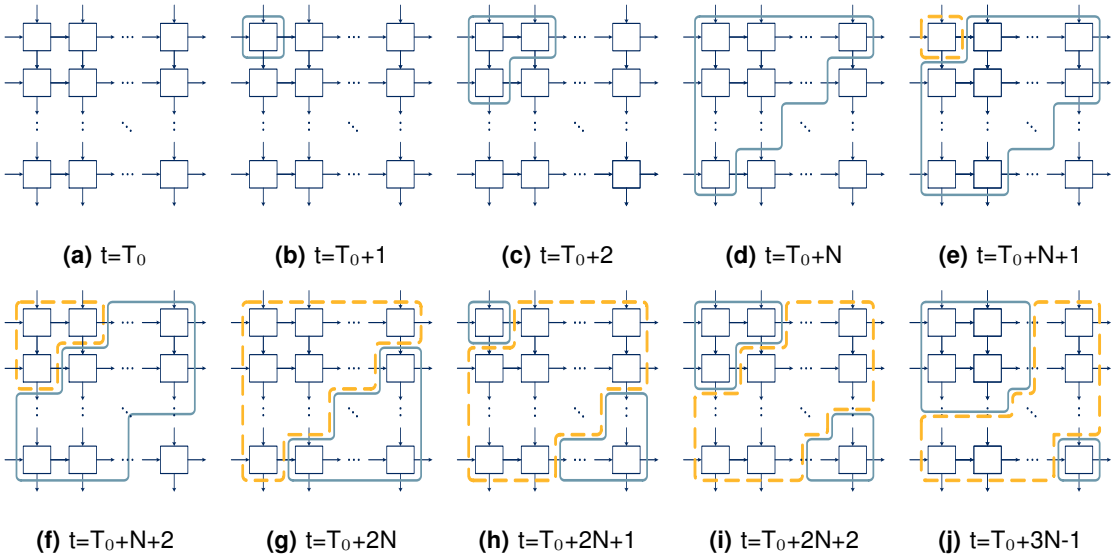


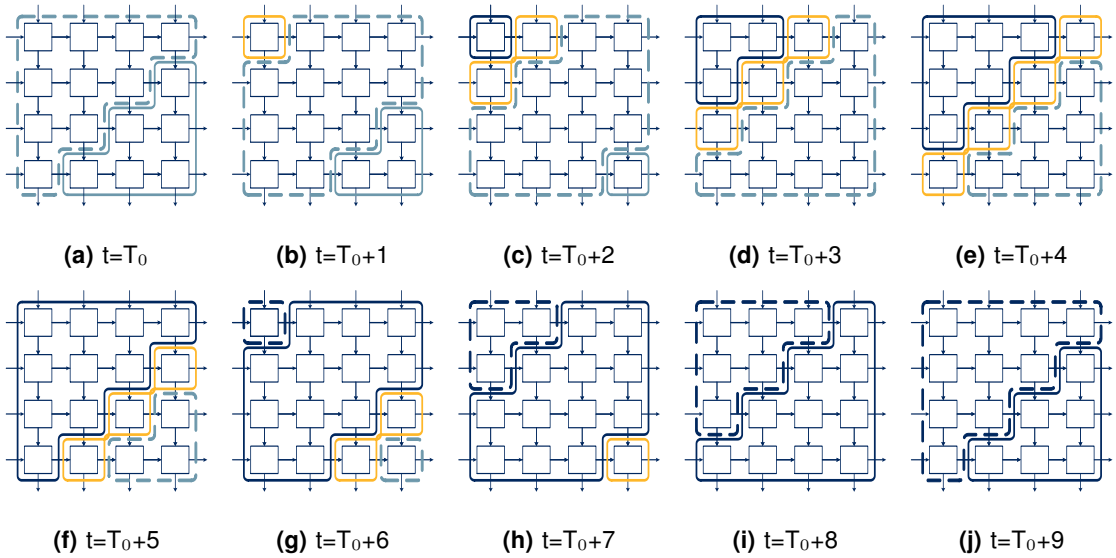
Figure 4.14: Dataflow in a TA with $N \times N$ PEs for the processing of $N \times N$ data blocks.

row-column decomposition is depicted using a dashed-line.

As it can be seen in Figure 4.14, the data values within the TA are processed by the PEs in a wavefront manner, by following a regular data streaming model. Accordingly, the data is fed into the TA rows through the input buffers in the left column of the array (see section 4.2). Then, it is processed by each PE (as explained in subsection 4.2.1) and subsequently propagated in the horizontal and vertical directions to the neighbour PEs inside the TA, advancing one PE level in both directions at each clock cycle. Conversely, the control signals for all the PEs enter the array through the top-left corner PE and are propagated to the other PEs also in both directions and synchronously with the data propagation. Such signals are also propagated into the TS, where they are used by the control logic to select the proper data to be feedback to each row of the TA, as discussed in section 4.2.3.

Therefore, the devised dataflow model makes it possible to start the computation of a different transform value in each row of the TA ay each clock cycle, provided that the IBs are not empty. Simultaneously, it also allows executing another iteration of the considered transform for all the values that are being processed in the remaining rows of PEs. Hence, this approach offers a maximization of the data processing rate within the TA, since the only PEs that will be stalled at any given time instant are those lacking some data to be processed, or the ones being reprogrammed to support the computations involving a distinct transform kernel in the subsequent clock cycles, as it is explained in section 4.2.1.

In such reprogramming stage, the PEs are also configured in a wavefront manner. The command that triggers such event is sent through the top-left corner PE, which then propagates it to the remaining PEs in both the horizontal and vertical directions. Since this operation requires only a single clock cycle and is performed in a pipelined fashion, it is possible to initiate the computation



(a) t=T_0 **(b)** t=T_0+1 **(c)** t=T_0+2 **(d)** t=T_0+3 **(e)** t=T_0+4

(f) t=T_0+5 **(g)** t=T_0+6 **(h)** t=T_0+7 **(i)** t=T_0+8 **(j)** t=T_0+9

Figure 4.15: Reprogramming of a TA with 4×4 PEs.

of the new transform on the clock cycle that immediately succeeds the reconfiguration of a PE. Consequently, the reprogrammable nature of the PEs only increases the latency of the proposed MTA in one single clock cycle. Still, the complete reprogramming of a TA with $N \times N$ PEs always requires $2N - 1$ clock cycles. This reprogramming procedure is illustrated in Figure 4.15 for a TA with 4×4 PEs, where the data-sets represented in light and dark blue colours correspond to the processing of two different blocks using distinct transform kernels and the reprogramming stage of the TA is represented in yellow colour.

From the previous discussion, it can be easily concluded that the reprogrammable characteristic of the proposed MTA does not affect its performance. Consequently, in steady conditions (i.e. the computation of the same transform for several consecutive blocks) the architecture is capable of sustaining a throughput of N samples per clock cycle, which enables the computation of 2-D $N \times N$ transforms in $2 \times N$ clock cycles, with a repetition interval of $2 \times N$ clock cycles. Naturally, $(N - 1)$ additional clock cycles are also required to fill in the TA with the data of the first block to be processed, so that it is possible to sustain this fully pipelined dataflow in the subsequent clock cycles. Similarly, retrieving all the computed data from the TA after the processing of the last data block requires another time period of $(N - 1)$ clock cycles.

4.4 Scalability and parallelism

When compared with other existing processing structures offering a similar functionality, the proposed MTA offers increased advantages also in terms of scalability and parallelism. Nowadays, these are two very important characteristics that are highly required by most transform architectures, in order to allow the realization of high performance and hardware efficient multi-transform cores capable of complying with the requirements of the H.264/AVC standard. Furthermore, these

features are also most relevant in the design of other transform computational circuits addressing not only alternative state-of-the-art video standards (e.g. VC-1 and AVS) but also the next generation of video standards (i.e. H.265/HEVC [129]), whose definitions include multiple and higher-order transforms and present optimizations regarding the application of parallel processing techniques.

4.4.1 Scalability

Due to the modular and highly flexible interconnection structure that was devised for the proposed MTA, this architecture can be easily scaled to realize transform cores that are capable of efficiently supporting the computation of transforms with distinct sizes. Although this scaling capability is common to all the modules of the architecture, it is applied differently to the TA, the TS, the IB and the CU.

In what concerns the TA, the offered scalability is used to resize the hardware structure of this module by changing the amount of PEs that it includes in both the vertical and horizontal dimensions, so that it can match the size of the considered transform kernel. As a result of the devised systolic dataflow model, such resizing operation only requires the extension of the control and data exchange signals that are used to interconnect all the PEs within the TA to a different amount of PEs.

Similarly, the scaling capabilities of the TS and of the IB allow adjusting the amount of multiplexers and buffers that compose these modules to the number of lines of the considered TA. In addition, the offered scalability is also used to adapt the internal structure of these circuits to the amount of columns of the TA, in order to maximize the performance *vs* hardware cost trade-off for any given architecture configuration. For example, employing multiplexers with only as much data inputs as the amount of columns (M) available in the considered TA to build the TS allows sustaining the maximum data processing rate offered by the TA when performing the row-column transposition operation (i.e. M samples per clock cycle) , as well as to minimize the hardware cost of the TS for such architecture configuration. In the same manner, it is possible to minimize the hardware cost of the IB without compromising the dataflow within the TA, by changing the depth of the buffers that compose the IB to match the amount of columns of the considered TA.

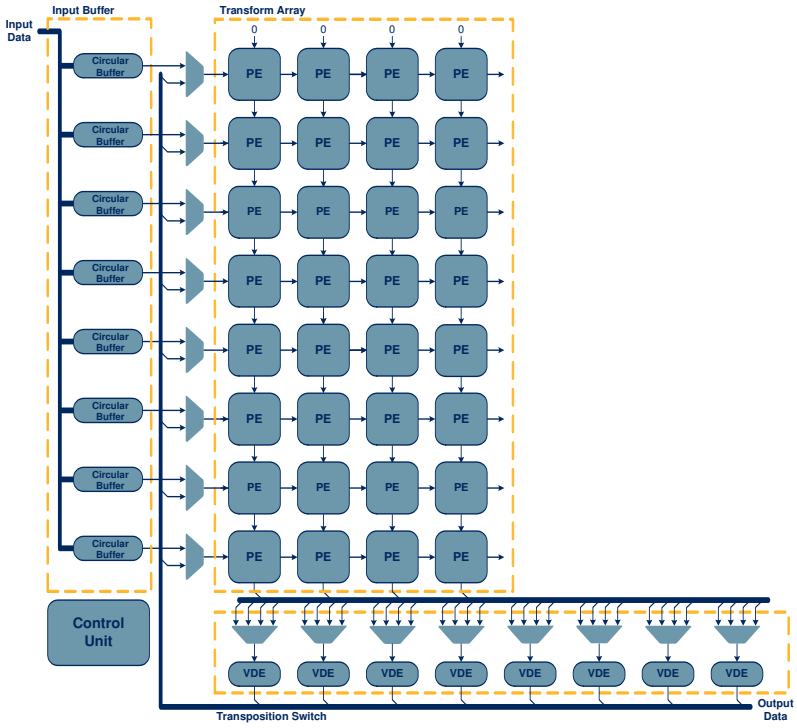
In contrast with all these modules, the architecture of the CU is not affected by the scalable nature of the proposed MTA. This is a result of its modular design (see section 4.2.5), since only the operation of the `1-D Transform Iteration Controller` depends on the size of the considered TA, i.e. the size of the transform kernel. Consequently, the CU can easily support the scalability of the architecture, by using the number of lines (N) and columns (M) of the TA as threshold values to control the operation of its auxiliary control circuits.

As it can be easily concluded, the enormous flexibility that is offered by the proposed MTA to realize transform cores with different sizes (scalability) and using distinct PEs (modularity) not only confers it the desired multi-transform capability but also provides significant advantages in

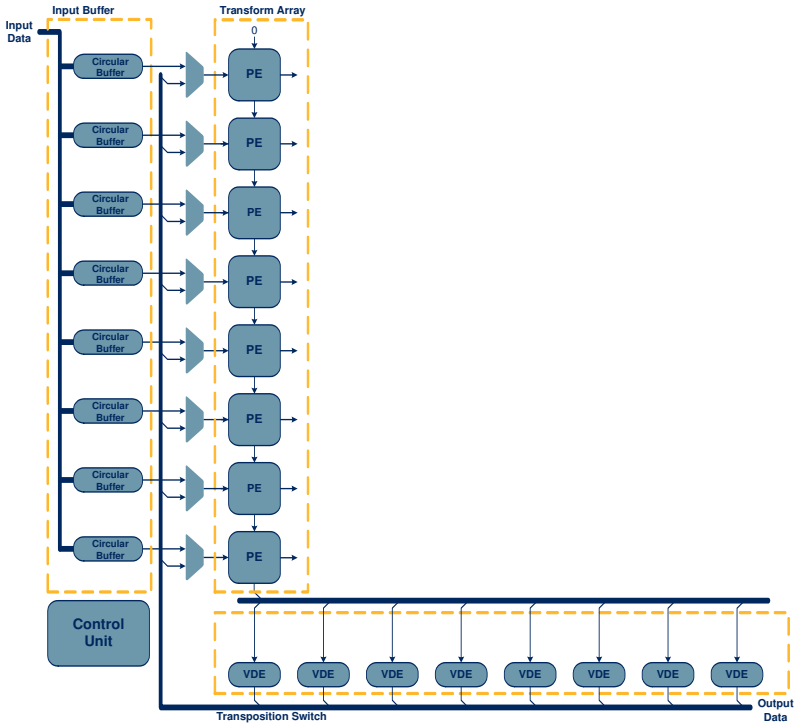
the design of hardware efficient and high performance multi-transform cores. Furthermore, these scalability and modularity characteristics also make it possible to successfully use the proposed MTA in the design of very efficient transform cores for alternative application domains, where the video codec performance might not be a highly critical issue. In such cases, the scalability of the architecture can also be exploited to adapt, in a different perspective, the hardware structure of the transform core to the specific requisites of the target application, i.e. the balance between the hardware cost and the desired performance. For example, designs with fewer PEs can be obtained by resizing the TA in its horizontal dimension and by considering setups where the number of columns in the TA (M) is a submultiple of the transform size to be computed (N). However, by using a smaller $N \times M$ array to compute a $N \times N$ transform, where N and M are integer powers of two and $M = 2^i$ with $i = 0, ..., \log_2 N - 1$, these designs require $\left(\frac{N}{M} - 1\right) \times N$ additional clock cycles to complete the computation procedure of the involved 1-D transforms. In these situations, the input data must go $\frac{N}{M} - 1$ additional times through the M columns of PEs in the TA, so that all the N^2 output results can be processed. Consequently, these transform cores reflect quite specific compromise solutions between the desired hardware savings and the resulting data processing capabilities of the design.

As an example, Figure 4.16 shows two possible lower cost designs that can be obtained by applying the scalability property to the transform core presented in Figure 4.4. Both designs are able to compute exactly the same set of transforms as the original transform core, i.e. the forward and inverse 8×8 and 4×4 DCTs and the forward and inverse 4×4 and 2×2 Hadamard transforms. The 8×4 PEs setup that is presented in Figure 4.16(a) comes as a compromise solution between performance and hardware cost. When compared with the base setup using 8×8 PEs, it offers a reduction of the required hardware resources by using only eight lines and four columns of PEs. As a consequence, it imposes a moderate penalty in the resulting throughput of the architecture. Nevertheless, the performance is only affected for the computation of the 8×8 transforms, which require twice as much clock cycles as in the 8×8 PEs setup. Conversely, the setup with 8×1 PEs depicted in Figure 4.16(b) offers the greatest savings in terms of hardware cost, due to the use of only a single column with eight lines of PEs. Nonetheless, this comes at the expense of a significant reduction in the computational performance of the resulting transform core, since this architecture configuration requires either 8, 4 or 2 times more clock cycles to process a 8×8, a 4×4 or a 2×2 transform, respectively. Consequently, it can be concluded that this setup is more suitable for applications that need to comply with strict hardware restrictions, or that do not require very high performance levels.

Despite the obvious advantages that can be obtained by efficiently exploiting the scalable nature of the proposed MTA to design low cost transform cores, such approach also has some minor drawbacks. On the one hand, it imposes some modifications in the architecture of the IB and of the TS, in order to ensure the computation of a $N \times N$ transform when using a smaller



(a) Setup with 8×4 PEs.



(b) Setup with 8×1 PEs.

Figure 4.16: Alternative setups for the proposed MTA.

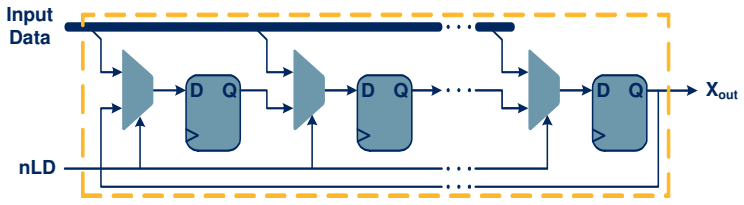


Figure 4.17: Architecture of the circular buffers.

$N \times M$ array. On the other hand, it slightly increases the complexity of the CU, which has to interface with the datapath in a different manner to avoid stalls in the processing of the input data.

To overcome such problems in a quite efficient manner, the IB offers an alternative operation mode in these smaller rectangular array configurations for which the FIFOs are configured to implement circular buffers, as shown in Figure 4.17. The resulting cyclic dataflow model provides the required means to feed the TA with the same input data (either corresponding to the first or the second 1-D transform) multiple times, as it is explained above. Consequently, this approach allows the computation of a $N \times N$ transform in $N - M$ independent iterations.

Regarding the TS, the necessary architecture modifications mostly consist in the embedding of M Variable Delay Elements (VDEs) in this processing structure, as shown in Figure 4.16. This results from the fact that by eliminating $N - M$ columns of PEs in the TA, it becomes impossible to implement the row-column transposition operation for transforms larger than $N \times M$ by solely using multiplexers. More specifically, in such scenarios the TA no longer contains enough registers to hold the $N \times N$ intermediate data values under processing, which must be transposed and feedback into the array for the computation of the column-wise transforms. Consequently, M VDEs are employed to store such temporary data in these rectangular TA configurations.

As it can be seen in Figure 4.18, which depicts the architecture of a generic VDE, these circuits mostly consist of a set of data registers and programmable bypass multiplexers. The multiplexers are used to compensate the lower or higher amount of PE columns in the TA, by including more (lower values of M) or less (higher values of M) registers in the TS datapath, respectively.

It should be highlighted that due to the interconnection scheme that was adopted for the proposed MTA, the VDEs mostly extend the TA pipeline into the TS, and therefore do not influence the overall system performance. Moreover, the hardware cost for the necessary transposition op-

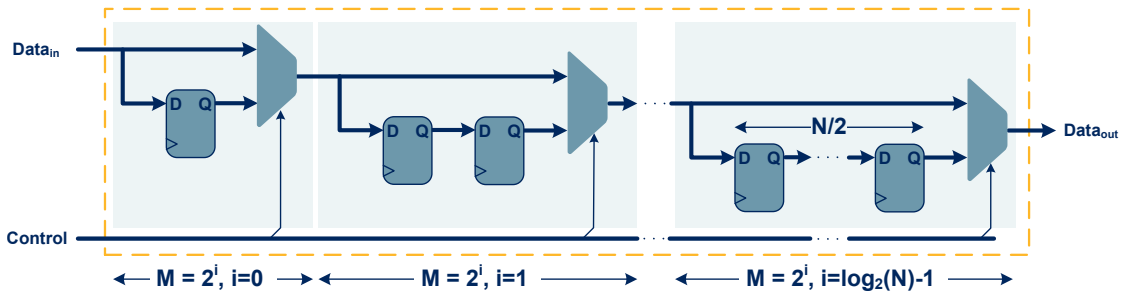


Figure 4.18: Architecture of the VDE.

eration also does not significantly increases in these smaller array configurations, since the total amount of registers used in the TA and in the TS is always the same.

4.4.2 Coarse-grain data-level parallelism

The scalability that is offered by the proposed MTA is most useful to improve both the functionality and the hardware efficiency of the devised architecture for the implementation of transform cores supporting the computation of any given transform. Nevertheless, by itself, such feature allows increasing the architecture's hardware efficiency only when the considered hardware realizations compute a single transform, or multiple transforms of the same size. In such situations, the TA includes as much PEs as the size of the largest transform to be computed, which allows to obtain the maximum throughput and PE occupation rate that can be provided by that specific architecture configuration, as it is explained in section 4.4.1. However, this is not the case of the hardware designs targeting modern video standards, which require the computation of several different transforms with distinct kernel sizes. In the H.264/AVC standard, it is necessary to compute the 8×8 DCTs, the 4×4 DCTs and the 4×4 and 2×2 Hadamard transforms[143].

In these scenarios, both the performance and the hardware efficiency of the implemented transform cores are greatly reduced when computing all the considered lower order transforms. This results from the fact that the involved TA, TS and IB permanently include all the hardware resources required to compute the highest order transforms, albeit only some of such resources are employed in the processing of the lower order transforms. Nonetheless, it is still possible to obtain hardware efficient and high performance multi-transform designs for these application domains, by exploiting the modular and flexible interconnection structure that characterizes the proposed MTA to implement a different processing mode involving *coarse-grain data-level parallelism*.

In this alternative processing mode, the proposed MTA is capable of efficiently and simultaneously computing several different transforms, by using the same hardware structure. More specifically, a transform core based on a TA with $N \times N$ PEs supporting this processing mode is capable of efficiently computing k transforms of size $\frac{N}{k} \times \frac{N}{k}$ in parallel, where $k = 2^i$ with $i = 0, ..., \log_2 N - 1$.

As it can be easily concluded, this parallel processing mode allows to greatly improve both the performance and the hardware efficiency of the proposed MTA for the implementation of transform cores supporting the computation of multiple transforms with distinct sizes. In what concerns the performance, this processing mode not only allows to accelerate the transform computation procedure of the $\frac{N}{k}$ lower order transforms by k times but also to guarantee the optimal throughput of N processed samples per clock cycle, as a consequence of k different transforms being simultaneously computed. Regarding the hardware efficiency, the simultaneous computation of all the $\frac{N}{k} \times \frac{N}{k}$ smaller transforms makes use of the whole TA, composed by $N \times N$ PEs, as well as of all the multiplexers and FIFOs available in the TS and IB, respectively. As a result, the archi-

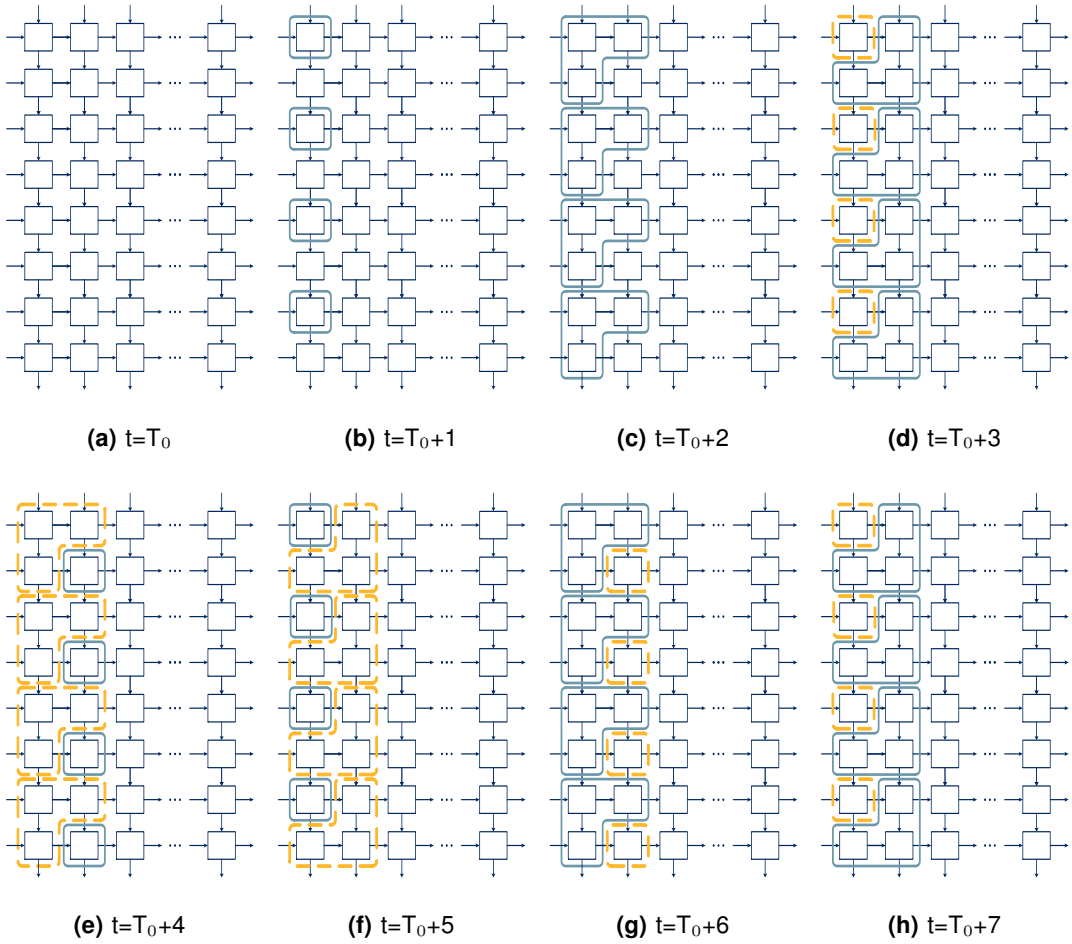


Figure 4.19: Dataflow in a 8×8 TA for the simultaneous processing of four 2×2 data blocks. The two data-sets depicted using a solid-line concern the residue values (or the transform coefficients, when inverse transforms are considered), while the data-sets represented using a dashed-line consist of the intermediate values of the row-column decomposition.

tecture's hardware efficiency increases about $(k-1) \times \frac{1}{k^2}$, since the global hardware cost of the MTA is mostly influenced by the TA. These gains can be clearly seen in Figure 4.19, which shows the dataflow for the parallel processing of four different 2×2 transforms, by using the transform core depicted in Figure 4.20. Such transform core, which consists of an implementation of the proposed MTA with added support for coarse-grain data-level parallelism, is based on a TA with 8×8 PEs ($N = 8$), which is capable of efficiently computing not only a single 8×8 transform in 30 clock cycles but also two 4×4 transforms ($k = 2$) and four 2×2 transforms ($k = 4$), in simultaneous, in just 14 and 6 clock cycles, respectively.

As it can be seen in Figure 4.20, the hardware structure of a transform core offering support for coarse-grain data-level parallelism is slightly more complex than the base design of the proposed MTA. In what concerns the TA, this module is rearranged as a set of k^2 independent *processing units*, which allows to simultaneously process up to k distinct $\frac{N}{k} \times \frac{N}{k}$ transforms. Each processing unit is composed of $\frac{N}{k} \times \frac{N}{k}$ PEs and is capable of fully processing an $\frac{N}{k} \times \frac{N}{k}$ transform. All the

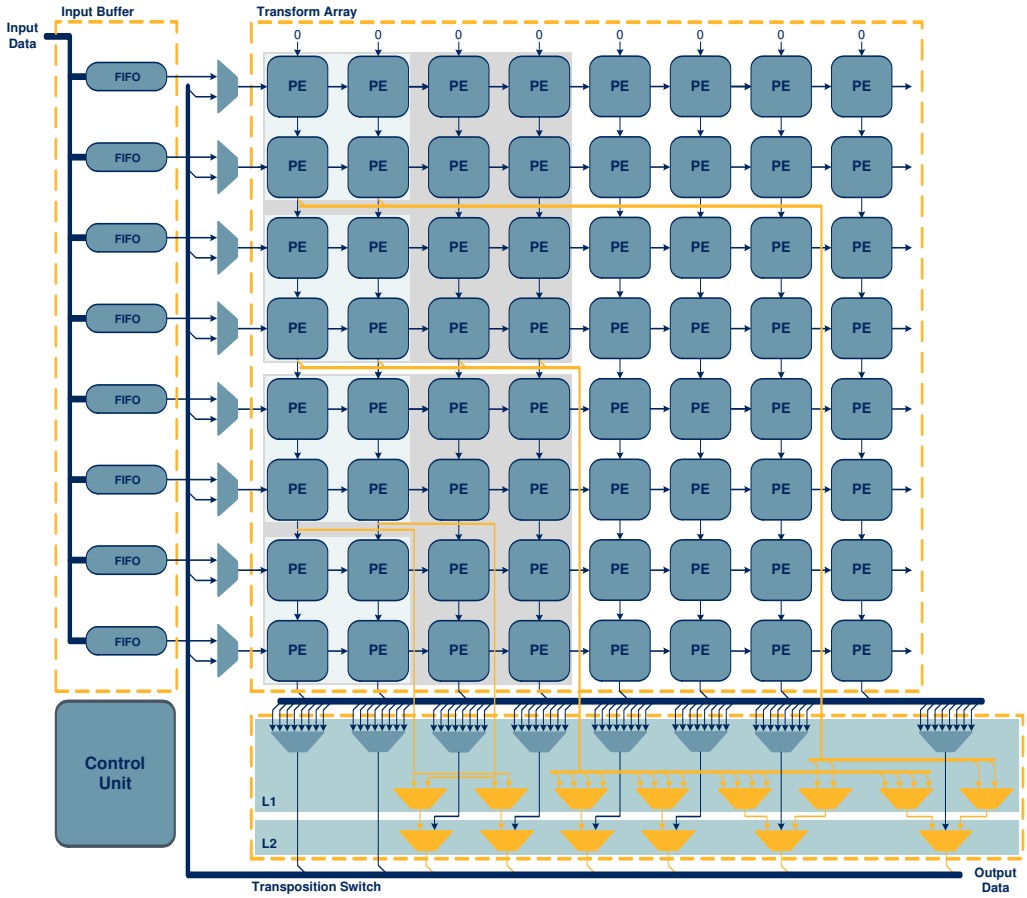


Figure 4.20: Architecture of a transform core supporting coarse-grain data-level parallelism.

k processing units available in the $(N \times N)$ TA have identical processing capabilities and can be interconnected with each other by using two different strategies, in order to make the best use of the available processing modules for the computation of any given transform size. In the first case, all the processing units are interconnected in cascade, just like in the base design of the proposed MTA, which provides the means required to support the computation of the larger $N \times N$ transforms. In the other case, all the processing units are left detached from one another and the k leftmost units of the TA are independently operated, so as to allow the computation of the several smaller transforms in parallel. Such flexible interconnection scheme is achieved by including a new set of multiplexers at the accumulated data inputs (`ACC_in`) of the topmost PEs of each one of the considered k processing units. As a consequence of this new PE interconnection scheme, the output interface of the TA also presents some minor differences regarding the base design of the proposed MTA. As it can be seen in Figure 4.20, such interface provides an additional set of output signals for the new data buses transporting the computation results of the smaller transforms into the TS.

The modifications that must be introduced in the TS also concern the interface and the architecture of this processing module. More specifically, the input interface of the TS has to be

extended to accommodate the new output data buses of the parallel TA. Conversely, the modifications that were introduced in the architecture of the TS provide the mechanisms required to parallelize the realization of the row-column data transposition operation, whenever k different transforms are processed in simultaneous within the TA. To achieve such goal, this enhanced architecture of the TS not only includes the N multiplexers that were already employed in its base design but also an additional set of $\sum_{j=1}^{\log_2 K} \left(2^j - 2^{j-1}\right) \times \frac{N}{2^j} + N - K$ multiplexers, where K represents the size of the smallest transform that can be processed. All these circuits are organized in several different units, which are properly combined to implement a hierarchical data switching structure with two selection levels, as it is illustrated in Figure 4.20.

In the first level of this structure, $N + \sum_{j=1}^{\log_2 K} \left(2^j - 2^{j-1}\right) \times \frac{N}{2^j}$ of such units are employed to perform, in parallel, the row-column data transposition operations concerning all the possible $\frac{N}{k} \times \frac{N}{k}$ transforms that can be computed in the TA. Each of these units, which is composed exclusively of $\left(\frac{N}{2^j} : 1\right)$ multiplexers, is used to process a single transform size. In particular, a unit composed of $\left(\frac{N}{2^j} : 1\right)$ multiplexers performs the transposition of the data blocks strictly resulting from the parallel computation of all the transforms with size $\frac{N}{k} \times \frac{N}{k}$. The output values provided by all these units are subsequently processed in the second level of the architecture, where another group of $N - K$ multiplexers selects the proper transposed data to be sent back to the TA, according to the size of the transforms under computation. As a result, by using this enhanced architecture of the TS it is possible to perform the transposition of the data generated by a $N \times N$ TA not only as a whole $N \times N$ block but also as k smaller $\frac{N}{k} \times \frac{N}{k}$ blocks.

Naturally, the extra multiplexers that are used to implement this hierarchical data switching structure have some influence in the maximum clock frequency of the devised transform cores, since they slightly increase the critical path of such designs. Nevertheless, the corresponding impact in the global performance of these transform cores should be almost negligible for two specific reasons. Firstly, because the critical path of the proposed MTA is only limited by the latency of its slowest functional module, as a result of the implemented pipelined processing scheme. Secondly, owing to the fact that the PEs are, usually, the slowest functional modules of the MTA, due to all the arithmetic circuits that they include.

Finally, the CU also comprises a couple of adjustments in the architecture of its three main modules, in order to being able to support the parallel processing mode offered by the proposed MTA. In what concerns the CU's main control unit, the required changes consist in the definition of an additional set of output signals for the state machine. Such control signals are used to command the extra multiplexers that must be embedded in the TA and in the TS to support this alternative processing mode, as it is explained above. Likewise the base design of the architecture, these signals enter the TA through the top-left corner PEs of each TA unit in the same clock cycle and are subsequently propagated to the other PEs within that same unit, synchronously with the data propagation. Regarding the other two controllers, i.e. the `1-D Transform`

`Iteration Controller` and the `TA Input Data Selector`, the necessary modifications consist only in the definition of a broader set of threshold values for the involved auxiliary control circuits (see section 4.2.5). For example, the `1-D Transform Iteration Controller` must take into consideration the value $\frac{N}{k}$ instead of N, when signalling the completion of the computation of a 1-D transform to the CU's main control unit in the parallel processing mode.

4.5 Summary

A novel high throughput Multi-Transform Architecture (MTA) for the computation of the multiple 2-D transforms defined in the H.264/AVC standard is presented in this chapter. Such processing structure is based on a 2-D systolic array and on a RAM free row-column data transposition circuit, which are used to compute the H.264/AVC forward and inverse 8×8 and 4×4 DCTs and the forward and inverse 4×4 and 2×2 Hadamard transforms by using the row-column decomposition approach. Due to its highly modular and flexible hardware structure, the proposed architecture can be easily scaled in terms of performance and hardware cost, in order to provide hardware realizations capable of complying with the specific requirements of any given video coding application. In addition, the proposed MTA also supports coarse-grain data-level parallelism. This allows it to simultaneously compute several different transforms, and thus to improve its hardware efficiency and speed up the transform computation procedure.

5

Configurable and low cost quantization architectures

Contents

5.1 Forward Quantization Architecture 110
5.2 Inverse Quantization Architecture 112
5.3 Unified Quantization Architecture 114
5.4 Summary 118

As discussed in chapter 2, the most recent video standards have been adopting improved prediction schemes and more elaborate compression techniques, which has resulted in the definition of considerably more complex quantization algorithms. In the H.264/AVC standard, such augment in complexity is mainly due to the inclusion of multiplications and divisions involving rational operands, as well as to the numerous memory accesses that the quantization procedure involves. In addition, the very tight interconnection between the transform and quantization procedures further increases such complexity requirements.

These forward and inverse quantization procedures significantly influence the performance of video coding systems, both in terms of throughput and latency (see section 1.1). Consequently, the increased complexity of the H.264/AVC quantization operations poses several additional and difficult challenges in the design and implementation of their corresponding video codecs, especially when HD contents must be supported or real time operation is demanded.

In order to overcome all these constraints, this chapter presents a new class of high performance architectures for the computation of the H.264/AVC quantization operations. Such processing structures can be used to realize not only forward and inverse quantizers but also unified quantization circuits with reduced hardware cost. Furthermore, they can be easily configured to provide implementations offering different trade-offs between performance and hardware cost, which makes them suitable to be used in multiple application domains with distinct requisites.

5.1 Forward Quantization Architecture

Although the quantization formulation presented in Equation 2.24 seems to be relatively straightforward, its direct implementation in hardware presents significant difficulties in terms of the demanded performance and involved implementation complexity. Such penalties are mostly owed to the inherent multiplication and division operations, whose computations require circuits imposing long latencies and high hardware costs, especially when high performance is demanded. Fortunately, the implementation complexity of Equation 2.24 can be greatly reduced, by exploiting the quite specific properties of the H.264/AVC quantization procedure[102].

On the one hand, the underlying quantization algorithm was developed to allow its implementation using 16-bits integer arithmetic, for integer input data encoded using 9-bits[1]. Nevertheless, the computation of the intermediate results requires arithmetic circuits with higher resolution. For example, the computation of the multiplication involving the transform coefficients (y_{ij}) and the multiplication factors ($MF(QP)_{ij}$) produces a 31-bits result. Still, all the quantized values are guaranteed to fall within the required 16-bits range, due to the final rescaling and rounding stage that is imposed by the H.264/AVC specification (see Equation 2.24).

On the other hand, the operands of almost all the operations that are specified in Equation 2.24

[1]The inputs to the transform process are the prediction residuals and thus they are encoded using 9-bits for 8-bits pixel data, which originates transform coefficient values that can be encoded using, at least, 16-bits.

(i.e. additions, multiplications and divisions) are integer power of 2 values. Therefore, the corresponding multiplications and divisions can be replaced by arithmetic shift-left/right operations, respectively, in order to reduce the associated implementation complexity. The quantization procedure can still be further simplified by applying the distributive property to the division shown in Equation 2.24. This allows using a single arithmetic shift-right circuit to implement the rescaling and rounding operations.

As a result of all these observations, the formulation presented in Equation 2.24 can be rewritten as Equation 5.1, where the $>>$ symbol denotes an arithmetic shift-right operation and h is given by Equation 2.25. The dead-zone control parameter (ρ) can be computed as shown in Equation 5.2, where the $<<$ symbol represents a shift-left operation and β is given by Equation 5.3.

$$z_{ij} = (y_{ij} \times MF(QP)_{ij} + \rho) >> \left(15 + \left\lfloor \frac{QP}{6} \right\rfloor + h\right) \tag{5.1}$$

$$\rho = f \times 2^h = \left(\frac{2}{3}^{\left\lfloor \frac{QP}{6} \right\rfloor} \times 2^{-\beta}\right) \times 2^h = \left(\frac{2}{3}^{\left\lfloor \frac{QP}{6} \right\rfloor} >> \beta\right) << h \tag{5.2}$$

$$\beta = \begin{cases} 0, & \text{if Intra block} \\ 1, & \text{otherwise} \end{cases} \tag{5.3}$$

This less complex formulation of the quantization operation evidences that the computation of a quantized coefficient consists of a four stages procedure, involving very few integer arithmetic operators. As it can be seen in Figure 5.1, the Forward Quantization Architecture (FQA) that was developed to support such computation procedure consist of a very efficient and purely combinational integer datapath, with five different processing phases and only three distinct computational circuits, i.e. a 16×15-bits signed multiplier, a 31-bits adder and a 31-bits arithmetic shift-right circuit. In addition, it includes other less complex logical elements for the computation of some intermediate values, such as a 4-bits adder to determine the shift-amount and a couple of ROMs to provide all the required constant values (i.e. $QP\%6$, $\left\lfloor \frac{QP}{6} \right\rfloor$, $\frac{2}{3}^{\left\lfloor \frac{QP}{6} \right\rfloor}$ and MF).

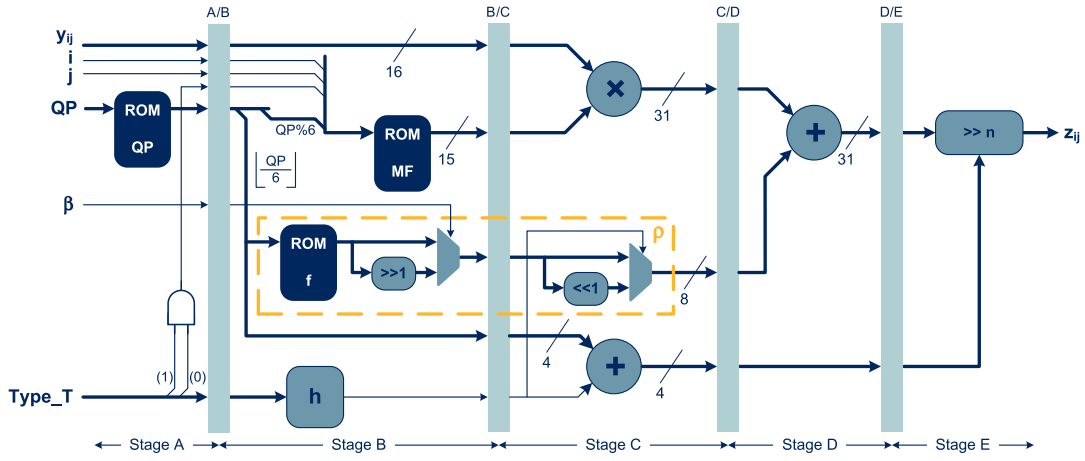


Figure 5.1: Block diagram of the proposed FQA.

Table 5.1: Possible configurations of the proposed FQA.

Architecture configuration	Pipeline registers			
	A/B	B/C	C/D	D/E
Non-pipelined	-	-	-	-
2 pipeline stages	-	-	√	-
3 pipeline stages	-	√	√	-
4 pipeline stages	-	√	√	√
5 pipeline stages	√	√	√	√

Figure 5.1 also evidences the very flexible structure of the proposed FQA, which not only is capable of providing high processing rates but also of supporting different configurations with distinct performance *vs* hardware cost characteristics. Such configurations consist of non-pipelined and pipelined versions of the proposed architecture, as it is shown in Table 5.1. The configuration with the most reduced hardware cost is also the one providing the longest critical path and consists of the non-pipelined version of the architecture, in which all the pipeline registers are replaced by direct point-to-point circuit interconnections. On the other hand, the highest performance levels are offered by the configuration implementing a fully pipelined architecture with the five stages `A`, `B`, `C`, `D` and `E`, as defined in the bottom of Figure 5.1. The remaining architecture configurations also implement pipelined designs, but with fewer pipeline stages. This allows to guarantee the desired trade-off between hardware cost and performance, which includes also the latency. For all the considered configurations, the obtained throughput is always one quantized transform coefficient per clock cycle. Moreover, the data processing rate is also maximized in each configuration, as a result of all the efforts that were devised to keep the pipeline stages properly balanced. This aspect is discussed in more detail in section 6.3.

In what concerns the functionality of the devised FQA, phases `A` and `B` are used to fetch all the data related to QP and the scaling factors (MF) from the ROMs. The shift-amount $\left(15 + \left\lfloor \frac{QP}{6} \right\rfloor + h\right)$ for the final adjustment of the quantized coefficient is computed in phase `C`, as well as the required rounding factors (ρ). Nevertheless, the preliminary steps in the computation of such value are realized in phase `B`, so as to keep all the pipeline stages as balanced as possible. The rounding operation is then realized in phase `D`, by using the scaled data values that are determined in phase `C`. Lastly, the ultimate value of the quantized coefficient is adjusted in phase `E`, by using the arithmetic right shifter. Such value is made available at the circuit's output port z_{ij}.

5.2 Inverse Quantization Architecture

As discussed in subsection 2.2.2, the H.264/AVC forward and inverse quantization operations are based on the same principles. Consequently, the set of observations and simplifications that are presented in section 5.1 can also be applied to Equation 2.31, in order to develop a

high performance and hardware efficient architecture for the realization of the H.264/AVC inverse quantization operation. Therefore, the following two optimizations were considered to develop the proposed Inverse Quantization Architecture (IQA): *i*) perform all the required computations using integer arithmetic; and *ii*) employ shift operations to realize the 22-bits signed multiplications involved in the rescaling and rounding steps of the algorithm. According to Equation 2.31, these shift operations may result either in an arithmetic shift-right or in a shift-left displacement, depending on the sign of the required shift-amount $\left(\left\lfloor \frac{QP}{6} \right\rfloor - \tau\right)$. As a result, the obtained alternative and less complex formulation of Equation 2.31 is the following:

$$y_{ij}^{S} = \begin{cases} (z_{ij} \times V(QP)_{ij} + \delta) >> \left(\tau - \left\lfloor \frac{QP}{6} \right\rfloor\right) & \text{, if } c1 \vee c2 \\ (z_{ij} \times V(QP)_{ij} + \delta) << \left(\left\lfloor \frac{QP}{6} \right\rfloor - \tau\right) & \text{, otherwise} \end{cases} \tag{5.4}$$

$$c1 : \text{inverse } H_{4\times4} \wedge (QP < 12)$$

$$c2 : \text{inverse } H_{2\times2} \wedge (QP < 6)$$

According to Equation 5.4, the computation of the scaled transform coefficients (y_{ij}^{S}) consists of a five stages procedure very similar to the one presented in section 5.1 for the computation of the forward quantization operation. As a consequence, the devised IQA is also based on a flexible hardware structure that can be configured to implement both pipelined and non-pipelined integer datapaths. As it can be seen in Figure 5.2, such processing structure is mostly composed of three computational circuits: a 16×6-bits signed multiplier, a 22-bits adder and one 22-bits arithmetic barrel shifter. In addition, it makes use of two ROMs to obtain all the required constant values (i.e. $QP\%6$, $\left\lfloor \frac{QP}{6} \right\rfloor$, $QP < 12$, $QP < 6$ and V), as well as of a couple of combinational circuits to generate the values of δ, τ and η. Table 5.2 describes the functionality of block η, which is used to determine the shift direction and the absolute value of the shift-amount to be considered by the barrel shifter. The encoding of the `Type_T` signal is shown in Table 4.1.

By comparing Figure 5.2 with Figure 5.1 it is possible to conclude that the proposed IQA and FQA consist of two very similar structures, with comparable operation modes. In fact, the IQA can

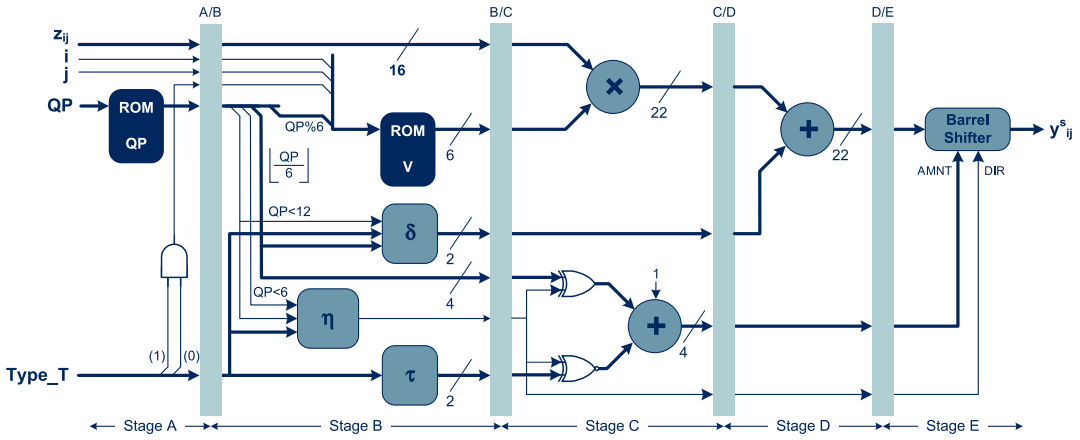


Figure 5.2: Block diagram of the proposed IQA.

Table 5.2: Functionality of the block η.

Type_T	$QP < 12$	$QP < 6$	η
0, 4	-	0	0
0, 4	-	1	1
1, 5	0	-	0
1, 5	1	-	1
2, 3, 6, 7	-	-	0

Table 5.3: Possible configurations of the proposed IQA.

Architecture configuration	Pipeline registers			
	A/B	B/C	C/D	D/E
Non-pipelined	-	-	-	-
2 pipeline stages	-	-	√	-
3 pipeline stages	-	√	√	-
4 pipeline stages	-	√	√	√
5 pipeline stages	√	√	√	√

also be configured to implement one out of five distinct configurations, whose characteristics and offered performance levels are identical to the corresponding configurations of the FQA described in section 5.1. Table 5.3 clearly demonstrates this property.

The similarities in the functionality of the two processing structures are also quite evident. In fact, the most significant difference in their operation modes concerns the computation of the rounding factor (i.e. ρ and δ). In the IQA, such computation takes place exclusively in phase B, while in the FQA it is computed throughout phases B and C to balance the pipeline stages. Consequently, the functionality of the proposed IQA can be easily extrapolated from the corresponding description presented in subsection 5.1.

5.3 Unified Quantization Architecture

A careful analysis of the formulations of the H.264/AVC forward and inverse quantization operations shown in Equations 2.24 and 2.31, respectively, reveals that they can be represented using a unique and more generic expression:

$$o_{ij} = (s_{ij} \times \sigma(QP)_{ij} + \varphi) \times 2^{\varepsilon} \tag{5.5}$$

In this formulation, σ represents a quantization function, while φ consists of a control parameter that can be used to improve the accuracy of the quantization procedure near the origin and ε is a rescaling/rounding factor. Hence, o_{ij} can be either the quantized or the scaled transform coefficient of line i and column j of the block of coefficients that is being forward or inverse quantized, respectively, while s_{ij} is the corresponding transform or quantized coefficient.

Equation 5.5 evidences that the realization of the two quantization procedures involves not only the same set of operations but also very similar operands. Therefore, it can be used to develop resource-shared architectures capable of computing both procedures. In terms of hardware

cost, such unified hardware structures offer some important advantages in the design of video encoders, since they enable the use of the same computational circuit to implement both the forward and the inverse quantization modules depicted in Figure 1.5. In addition, they are also very relevant for the implementation of Integrated Transform and Quantization Architectures (ITQAs) supporting the H.264/AVC encoding and decoding procedures. In such cases, the hardware cost and the efficiency of these processing structures can be greatly improved by using also a resource-shared Unified Quantization Architecture (UQA) to realize the quantization module supporting the necessary computations of the forward and inverse quantization procedures.

The hardware cost of such UQAs can be further reduced by recalling the set of considerations and simplifications that are presented in the previous subsections. Such approach not only allows realizing all the operations exclusively using integer arithmetic but also replacing most of the involved multiplications and divisions by arithmetic and logical shift operations. As a consequence, the performance of these processing structures can also be greatly improved, due to the use of faster and less complex circuits.

By following this methodology, Equation 5.5 can be applied to compute the forward quantization operation by using the MF function (see Equation 2.21 and Equation 2.22) as the adopted quantization function σ (i.e. $\sigma_Q = MF$), and by computing the control parameter φ as specified in Equation 5.5 (i.e. $\varphi_Q = \rho$). The multiplication involving ε can be realized with an arithmetic shift-right operation, where the shift-amount is given by Equation 5.6. Such important simplification results from the fact that this scaling factor always takes positive integer values (see Equation 2.24 and Equation 5.1).

$$\varepsilon_Q = 15 + \left\lfloor \frac{QP}{6} \right\rfloor + h \tag{5.6}$$

As a result of all these simplifications, Equation 5.5 can be rewritten for the quantization operation as shown in Equation 5.7.

$$o_{Q_{ij}} = (s_{ij} \times \sigma_Q(QP)_{ij} + \varphi_Q) >> \varepsilon_Q \tag{5.7}$$

Regarding the inverse quantization operation, the computation of the involved control parameter φ_{IQ} can be optimized by carefully analyzing Equation 2.32. Such observation reveals that φ_{IQ} can take only the three values shown in Equation 5.8.

$$\varphi_{IQ} = \begin{cases} 0 & \text{, if inverse } H_{4\times4} \wedge (QP \geq 12) \\ 1 & \text{, if inverse } H_{4\times4} \wedge (QP \geq 6) \wedge (QP < 12) \\ 2 & \text{, otherwise} \end{cases} \tag{5.8}$$

As it is discussed in section 5.2, the scaling factor for the inverse quantization operation (ε_{IQ}) can take both positive and negative integer values. Therefore, the corresponding multiplication can be implemented by using either an arithmetic shift-right or shift-left operation. The involved shift-amount is obtained by replacing Equation 2.33 in Equation 5.4, as shown in Equation 5.9.

$$\varepsilon_{IQ} = \begin{cases} -\left(2 - \left\lfloor \frac{QP}{6} \right\rfloor\right) & \text{, if } c1 \\ -\left(1 - \left\lfloor \frac{QP}{6} \right\rfloor\right) & \text{, if } c2 \\ \left\lfloor \frac{QP}{6} \right\rfloor & \text{, otherwise} \end{cases} \tag{5.9}$$

$$c1 : \text{inverse } H_{4\times4} \wedge (QP < 12)$$

$$c2 : \text{inverse } H_{2\times2} \wedge (QP < 6)$$

By considering all the above simplifications, Equation 5.5 is rewritten as Equation 5.10 for the implementation of the inverse quantization operation, where the quantization function σ consists of the V function (see Equation 2.29 and Equation 2.30), i.e. $\sigma_{IQ} = V$.

$$o_{IQ_{ij}} = \begin{cases} (s_{ij} \times \sigma_{IQ}(QP)_{ij} + \varphi_{IQ}) >> |\varepsilon_{IQ}|, & \text{if } c1 \vee c2 \\ (s_{ij} \times \sigma_{IQ}(QP)_{ij} + \varphi_{IQ}) << \varepsilon_{IQ}, & \text{otherwise} \end{cases} \tag{5.10}$$

$$c1 : \text{inverse } H_{4\times4} \wedge (QP < 12)$$

$$c2 : \text{inverse } H_{2\times2} \wedge (QP < 6)$$

Accordingly, the UQA that was developed to realize the forward and the inverse H.264/AVC quantization procedures implements the functionalities represented in Equation 5.7 and Equation 5.10 (or more generally, in Equation 5.5), by sharing the hardware resources that are common to both procedures. More specifically, the UQA depicted in Figure 5.3 makes use of one 16×15-bits signed multiplier, one 31-bits adder and one 31-bits barrel shifter to compute the product $(s_{ij} \times \sigma(QP)_{ij} + \varphi) \times 2^{\varepsilon}$ for the two operations. Moreover, a couple of ROMs are also shared in these computations. $\texttt{ROM}_{\sigma}$ is used to implement the MF and V quantization functions, while $\texttt{ROM}_{QP}$ provides all the constant values depending on QP, i.e. $QP\%6$, $\left\lfloor \frac{QP}{6} \right\rfloor$, $QP < 12$ and $QP < 6$. In addition, a 4-bis adder is used in conjunction with the combinational block μ and some logic gates to determine the required shift-amount for the two operations. Nonetheless, the

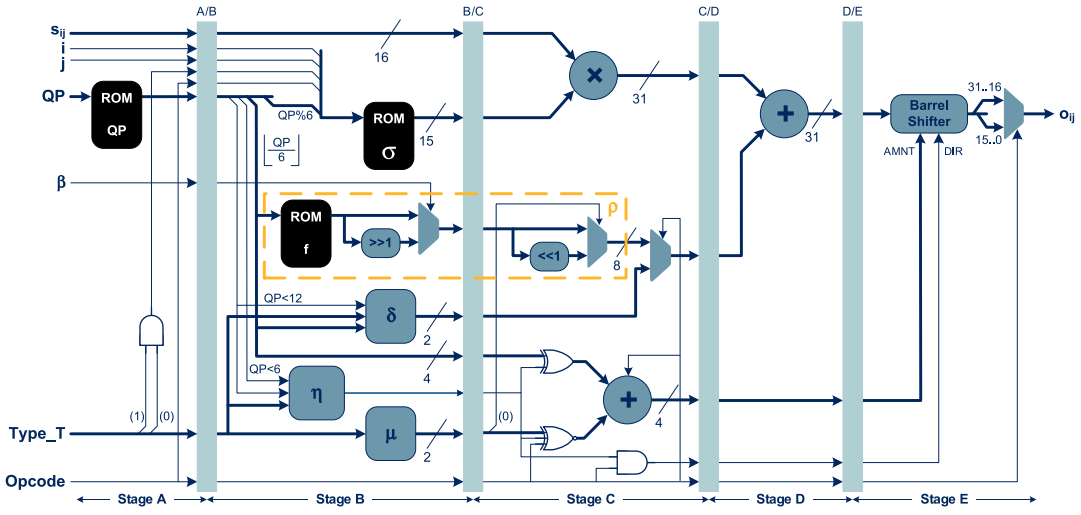


Figure 5.3: Block diagram of the proposed UQA.

Table 5.4: Functionality of the μ block.

Opcode	Type_T	μ
0	0, 1	1
0	2, 3	0
1	4	1
1	5	2
1	6, 7	0

Table 5.5: Possible configurations of the proposed UQA.

Architecture configuration	Pipeline registers			
	A/B	B/C	C/D	D/E
Non-pipelined	-	-	-	-
2 pipeline stages	-	-	$\surd$	-
3 pipeline stages	-	$\surd$	$\surd$	-
4 pipeline stages	-	$\surd$	$\surd$	$\surd$
5 pipeline stages	$\surd$	$\surd$	$\surd$	$\surd$

proposed UQA also includes other circuits that are exclusively used to realize only one of the two possible quantization procedures. For example, the ρ block is solely involved in the computation of the forward quantization operation, while the δ and η blocks are exclusively employed to perform the inverse quantization procedure. The functionality of these blocks is presented in Equations 2.25, 5.2, 2.32 and 2.33, respectively, while Table 5.4 describes the functionality of block μ. The Opcode signal defines the type of quantization to be performed, i.e. forward quantization (Opcode=0) or inverse quantization (Opcode=1).

As it can be seen in Figure 5.3, the proposed UQA also implements a purely combinational integer datapath with four different processing phases. Likewise the FQA and the IQA, this processing structure can be configured to provide one non-pipelined and several pipelined hardware realizations with distinct performance *vs* hardware cost characteristics, in order to optimally address the requirements of any given application. Table 5.5 lists the five possible architecture configurations, whose characteristics and offered performance levels are almost identical to the ones discussed in section 5.1 and section 5.2.

In what concerns the functionality of the proposed UQA, phase A is used to fetch all the constant coefficients dependent on QP from ROM_{QP}, while the quantization function values $MF(QP)_{ij}$ and $V(QP)_{ij}$ are retrieved from ROM_{σ} in phase B. The amount and direction of the shift for the final adjustment of the processed values are computed in phase C, together with the rounding control parameter (φ). Nevertheless, the preliminary steps in the computation of such values are realized in phase B, in order to keep all the processing phases as balanced as possible. The rounding operation is realized in phase D, by using the scaled data value that is selected in phase C. In phase E, the final values of the quantized/scaled transform coefficients are adjusted using the barrel shifter. The resulting 16-bits value of both the quantized and scaled transform coefficients are provided at the o_{ij} output port of the architecture.

5.4 Summary

A new class of architectures for the computation of the forward and inverse H.264/AVC quantization procedures is presented in this chapter. Such processing structures are based on a configurable integer datapath with reduce hardware cost, which can be used to realize not only forward and inverse quantizers but also resource-shared hardware structures capable of computing both operations. The highly flexible hardware structure that was devised for this datapath allows it to be easily configured to provide several different pipelined and non-pipelined hardware realizations, reflecting distinct hardware cost *vs* performance (both in terms of data processing rate and latency) optimization goals. Such enormous versatility allows these quantizers to optimally address the requirements of any given application. Consequently, the application scenarios of the proposed class of quantization architectures range from the implementation of hardware accelerators in modern SoCs to specialized functional units of ASIPs. In addition, the devised quantization architectures can also be integrated with other existing processing structures for the computation of the H.264/AVC transforms, in order to develop integrated transform and quantization specialized processors.

6

Experimental evaluation

Contents

6.1 Experimental setup and implementation considerations 120
6.2 Evaluation of the proposed MTA 122
6.3 Evaluation of the proposed quantization architectures 131
6.4 Generalized design of integrated transform and quantization circuits 138
6.5 Summary 146

The proposed multi-transform and quantization architectures are evaluated in this chapter, regarding their effectiveness and efficiency in the design of H.264/AVC compliant video encoding and decoding systems for multiple application domains, including their adequacy for the processing of ultra HD video formats in real time.

Accordingly, the considered analysis comprises the experimental assessment of the performance levels that can be attained by using such processing structures in the most common video coding scenarios, as well as the evaluation of the corresponding hardware costs and hardware efficiency levels. In addition, it comprehends the relative assessment of the proposed architectures, by considering the most related and prominent alternative designs that were proposed in the literature.

6.1 Experimental setup and implementation considerations

The conducted experimental procedures considered several different proof of concept transform cores and quantizers based on the architectures presented in chapters 4 and 5, respectively.

In what concerns the evaluation of the proposed MTA, four distinct transform cores were implemented and assessed. Each of these circuits supports the computation of the forward and inverse 8×8 and 4×4 DCTs, as well as the forward and inverse 4×4 and 2×2 Hadamard transforms. However, they reflect distinct architectural configurations that offer different trade-offs between performance and hardware cost. More specifically, three of the considered hardware realizations concern standard implementations of the proposed MTA that are based on specific setups of the TAs using 8×8, 8×4 and 8×2 PEs. The remaining hardware realization is also based on an 8×8 PEs TA, but with added support for coarse-grain data-level parallelism. Hence, this transform core is capable of simultaneously computing not only one 8×8 transform but also two 4×4 transforms or four 2×2 transforms.

The evaluation of the proposed class of quantization architectures is based on fifteen distinct forward, inverse and unified quantizers. Such circuits implement the non-pipelined and the four pipelined architecture configurations presented in Table 5.1, Table 5.3 and Table 5.5, thus offering distinct performance *vs* hardware cost characteristics. All these configurations are capable of processing the two types of quantization blocks defined in the H.264/AVC standard, i.e. the 4×4 and the 8×8 blocks of coefficients.

All the considered proof of concept processing structures, whose characteristics are summarized in Table 6.1 and Table 6.2, were experimentally assessed by using the Xilinx Virtex-7 XC7VX485T-2FFG1761C FPGA device[146] included in a Xilinx VC707 Evaluation Kit[148]. In addition, some structures were also synthesized for other Xilinx FPGA devices, so that the presented discussions concerning their relative assessment regarding alternative state-of-the-art designs can be as fair as possible. All these implementation and evaluation procedures were conducted

Table 6.1: Characteristics of the considered proof of concept transform cores.

Design	TA configuration	Data-level parallelism	Supported transforms
$T_{8\times2}$	8×2	○	$\text{FDCT}_{8\times8}$, $\text{IDCT}_{8\times8}$, $\text{FDCT}_{4\times4}$, $\text{IDCT}_{4\times4}$, $\text{H}_{4\times4}$, $\text{H}_{2\times2}$
$T_{8\times4}$	8×4	○	
$T_{8\times8}$	8×8	○	
$T_{8\times8p}$	8×8	●	

Table 6.2: Characteristics of the considered proof of concept quantizers.

Design	Architecture configuration	Quantization		Supported block sizes
		Forward	Inverse	
FQ_{NP}	Non-pipelined	●	○	8×8, 4×4
FQ_{P2}	2 pipeline stages	●	○	
FQ_{P3}	3 pipeline stages	●	○	
FQ_{P4}	4 pipeline stages	●	○	
FQ_{P5}	5 pipeline stages	●	○	
IQ_{NP}	Non-pipelined	○	●	
IQ_{P2}	2 pipeline stages	○	●	
IQ_{P3}	3 pipeline stages	○	●	
IQ_{P4}	4 pipeline stages	○	●	
IQ_{P5}	5 pipeline stages	●	○	
UQ_{NP}	Non-pipelined	●	●	
UQ_{P2}	2 pipeline stages	●	●	
UQ_{P3}	3 pipeline stages	●	●	
UQ_{P4}	4 pipeline stages	●	●	
UQ_{P5}	5 pipeline stages	●	○	

with the Xilinx ISE 13.2i and 10.1i development toolchains[144,145], by considering a slightly modified version of the standard synthesis and place and route strategies, in order to achieve a balanced optimization of the performance *vs* area occupation results.

To accomplish this evaluation, the proposed transform and quantization architectures were fully described using the VHSIC Hardware Description Language (VHDL). These descriptions follow a modular design approach based on independent and self-contained functional blocks, making an extensive use of `generic` configuration and parameterization inputs to better support all the architecture functionalities and their ample configuration options. Furthermore, such descriptions mostly adopt a quite generic coding style, in order to achieve efficient implementations not only when using FPGA devices but also when other alternative technologies might be considered (e.g. ASIC). Nonetheless, a special attention was given to the description of the most performance critical blocks (e.g. the PE in the MTA; the multiplier, the adders and the ROMs in the quantization architecture), so as to assist the synthesis tools in inferring the most efficient primitives for its implementation, according to the chosen design constraints, synthesis strategy and implementation technology. This design effort was especially relevant in the realization of all the proof of concept quantization circuits, since it allowed making a good use of the `DSP48E1` slices[147] available in the considered Virtex-7 FPGA device. In particular, this slice was used to ensure an efficient computation of the multiplication operation in the non-pipelined configurations

of all the implemented quantizers. In the pipelined configurations, the `DSP48E1` slice was used to implement the MAC operation required in processing phases `C` and `D` (see Figure 5.1, Figure 5.2 and Figure 5.3). The impact of these optimizations in terms of the resulting performance and hardware cost is more thoroughly discussed in section 6.3.1.

The functionality of all the considered hardware implementations was also properly and successfully validated with several test vectors that were generated using the H.264/AVC JM reference software[63]. Such data was obtained with the encoding of a collection of benchmark standard test video sequences of the MPEG-4 Video Verification Model[64], by considering not only the 8×8 DCT but also a wide range of quantization steps. Appendix B provides more details about the considered test conditions.

As it can be easily concluded from the previous presentation, the adopted experimental setup provides the means required to validate the functionality of all the considered transform cores and quantizers, as well as to perform the required evaluation of the proposed transform and quantization architectures regarding their effectiveness and efficiency in the design of H.264/AVC codecs for multiple application domains. Furthermore, it is worth noting that,

- All the considered proof of concept processing structures are able to perform the complete set of operations specified in the four most important profiles of the H.264/AVC standard, i.e. the BP, MP, XP and HiPs, therefore addressing all the intended application domains;
- The several different transform cores and quantizers that are considered in this evaluation offer distinct hardware cost *vs* performance trade-offs, which allows using them to build both low cost and high performance video coding systems and thus cover all the aimed classes of devices;
- The presented discussion is based on experimental results obtained using a Xilinx XC7VX485T-2FFG1761C Virtex-7 FPGA device, which has shown to be adequate for the realization of modern video coding systems (including the ones based on the H.264/AVC standard) due to offering high logic density and supporting relatively high clock frequencies. Naturally, better results (in terms of performance) could be obtained by implementing all the considered proof of concept transform cores and quantizers using an ASIC technology. However, such alternative approach is out of the scope of this PhD thesis, since it would require an extremely expensive and time consuming engineering effort that would not add much to the presented discussion, owing to the Intellectual Property (IP) core nature of the proposed multi-transform and quantization architectures.

6.2 Evaluation of the proposed MTA

The experimental results that were obtained with the prototyping of the considered four proof of concept transform cores in the adopted Xilinx Virtex-7 FPGA device are reported and anal-

ysed in this section. Relative assessment is also discussed, by considering the most related and prominent alternative designs that were proposed in the literature.

6.2.1 FPGA implementation results

Table 6.3 summarizes the implementation results that were obtained for the conducted $T_{8\times8}$ proof of concept hardware realization, which evidence the advantages offered by the proposed MTA for the design of high performance H.264/AVC compliant transform cores based on FPGA platforms.

On the one hand, the presented maximum clock frequency values reveal that the devised proof of concept structure not only is able to compute almost 18 GOPS but also to offer a sustained processing throughput of about 2.2×10^9 Samples per Second (S/s), by using 64 PEs operating in simultaneous. Such considerably high performance levels allow this processing structure to compute the whole set of H.264/AVC transforms in real time for video sequences with resolutions up to 3840×2160 pixels and with a frame rate of 30 fps (i.e., the 4k UHDTV video format). More specifically, by assuming the usual 4:2:0 chrominance subsampling format, this proof of concept transform core is capable of processing 1.40×10^6 MB/s in the default coding mode, 1.35×10^6 MB/s when the $Intra_{16\times16}$ prediction mode is adopted, or 2.06×10^6 MB/s when the 8×8 transform is adopted in the high profiles.

As it can be seen in Table 6.3, these results are due to the following three reasons: *i*) the highly optimized and very efficient architecture that was devised for the PEs, which can be operated using a clock frequency of almost 300 MHz despite supporting all the operations involved in the computation of all the H.264/AVC transforms; *ii*) the highly parallel and pipelined processing nature of the proposed MTA; and *iii*) the distribution of the architecture control circuits and signals among the main CU and the several PEs composing the TA, which allowed shortening its critical path.

On the other hand, the remaining data that is presented in Table 6.3 also demonstrates the reduced hardware cost of the $T_{8\times8}$ transform core, while offering a remarkable processing rate. As it can be seen, about 7% of the total hardware resources that are available in the adopted medium-

Table 6.3: Implementation results of the proof of concept $T_{8\times8}$ transform core in a Xilinx Virtex-7 XC7VX485T-2FFG1761C FPGA device.

Processing structure	FPGA Utilization				Max. Freq.
	Registers		LUTs		
⊟ Transform core	6420	1.06 %	21452	7.07 %	280.6 MHz
⊞ IB	1681	0.28 %	1254	0.41 %	344.8 MHz
⊟ TA (with 8×8 PEs)	1501	0.25 %	15727	5.18 %	296.5 MHz
⊞ PE	70	0.01 %	254	0.08 %	298.5 MHz
⊞ TS	1462	0.24 %	2161	0.71 %	360.0 MHz
⊞ CU	97	0.02 %	77	0.03 %	366.4 MHz

size FPGA device[1] were used in the realization of the considered transform core. Almost all of these hardware resources (about 73%) were employed to implement the 2-D systolic array, with each of its 8×8 PEs requiring only 44 slices to perform the transform computations involving 20 distinct kernel values. This corresponds to only about 1% of the total implementation resources.

From the previous discussion, it can be concluded that the hardware cost of the proposed MTA mostly results from the adopted configuration for its TA (i.e. the amount of PEs), while its maximum clock frequency is limited by the internal critical path of the PEs. Consequently, it can be stated that other optimized and efficient transform cores addressing alternative application domains, with distinct requirements in terms of performance and hardware cost, can be obtained by exploiting the highly configurable and scalable characteristics of the presented transform architecture. The results presented in Table 6.4 and Table 6.5, which concern the implementation of the $T_{8\times4}$ and $T_{8\times2}$ transform cores, sustain this observation and emphasize the advantages of the proposed MTA in terms of scalability.

As it can be seen, the use of TAs with different dimensions does not significantly constrain the maximum frequency of the clock signal that is applied to the architecture. However, it allows to efficiently trade-off the offered performance (in terms of throughput) for hardware savings. For example, the 8×4 configuration allows to reduce the hardware cost in about 37%, when compared with the base 8×8 PEs setup. For greater savings, the 8×2 configuration reduces the hardware requirements of the proposed MTA by 56%. This is a very important factor for reconfigurable

Table 6.4: Implementation results of the proof of concept $T_{8\times4}$ transform core in a Xilinx Virtex-7 XC7VX485T-2FFG1761C FPGA device.

Processing structure	FPGA Utilization				Max. Freq.
	Registers		LUTs		
⊟ Transform core	4068	0.67 %	13591	4.48 %	288.7 MHz
⊞ IB	1681	0.28 %	1254	0.41 %	344.8 MHz
⊟ TA (with 8×8 PEs)	2578	0.42 %	8614	2.84 %	297.3 MHz
⊞ PE	70	0.01 %	254	0.08 %	298.5 MHz
⊞ TS	1462	0.24 %	2161	0.71 %	360.0 MHz
⊞ CU	97	0.02 %	77	0.03 %	366.4 MHz

Table 6.5: Implementation results of the proof of concept $T_{8\times2}$ transform core in a Xilinx Virtex-7 XC7VX485T-2FFG1761C FPGA device.

Processing structure	FPGA Utilization				Max. Freq.
	Registers		LUTs		
⊟ Transform core	2852	0.47 %	9530	3.14 %	290.1 MHz
⊞ IB	1681	0.28 %	1254	0.41 %	344.8 MHz
⊟ TA (with 8×8 PEs)	1346	0.22 %	4497	1.48 %	297.8 MHz
⊞ PE	70	0.01 %	254	0.08 %	298.5 MHz
⊞ TS	1462	0.24 %	2161	0.71 %	360.0 MHz
⊞ CU	97	0.02 %	77	0.03 %	366.4 MHz

[1]The considered Xilinx Virtex-7 XC7VX485T-2FFG1761C FPGA device[146] contains a total of 75,900 slices. Each Virtex-7 slice contains four 6-input LUTs and eight D-type flip-flops (i.e. one bit registers).

implementation platforms (as well as for ASIC realizations), since the ability to choose different architecture configurations for the proposed MTA enables the implementation of transform cores in FPGA devices with both reduced and greater amounts of hardware resources.

It should be noted that the set of functionalities that are offered by the implemented transform cores are not diminished as a result of these hardware savings. In fact, such reductions only imply an increase in the amount of time required to perform all the necessary computations. Nevertheless, both the $T_{8\times4}$ and the $T_{8\times2}$ proof of concept transform cores are still capable of achieving real time operation for video sequences in the 1080p HDTV format (1920×1080 @ 60 fps), as it is shown in Figure 6.1. This results not only from the highly efficient pipeline structure that was devised for the TA, which is capable of providing the maximum achievable throughput rate (without any need for eventual data stalls) but also from the significantly high operating frequency value that is supported by the devised PE. Hence, it can be concluded that the scaling capabilities of the proposed MTA enables the design of transform cores with the most suitable architecture configuration for any given application. For example, smaller configurations can be adopted by applications with stricter restrictions in terms of hardware cost, but where the data processing rate can be slower. Conversely, larger configurations should be employed to support the more time critical applications, which demand the maximum performance and where the hardware cost is not a limitation.

In another direction, Table 6.6 shows the impact of the devised coarse-grain data-level parallel processing technique in the performance and hardware cost of the implemented proof of concept 8×8 transform cores. As it can be seen, the $T_{8\times8p}$ transform core can be operated using the

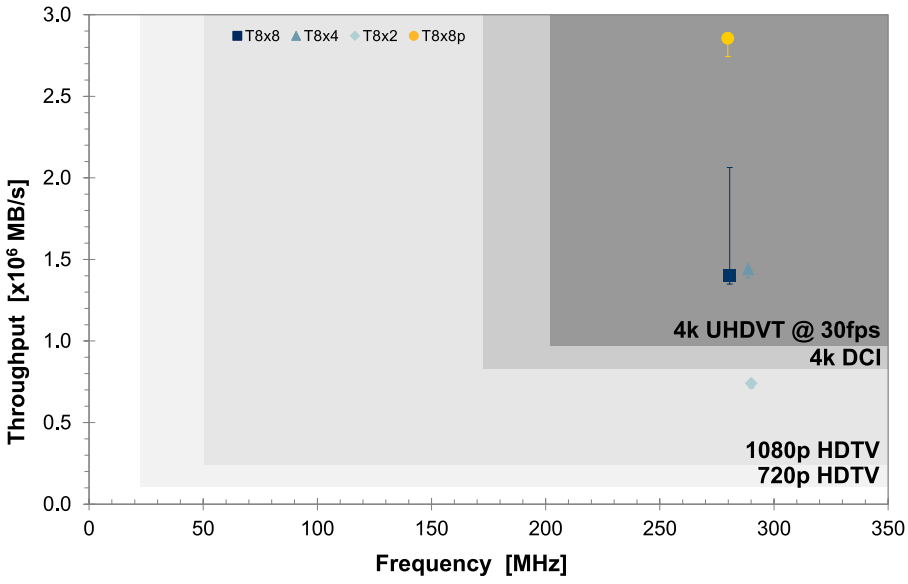


Figure 6.1: Performance comparison of the several implemented transform cores, when operated at their maximum clock frequency. For each transform core, the figure shows the amount of MBs that can be processed when using the $Intra_{16\times16}$ prediction mode (lowest value), the default coding mode (middle value) and the 8×8 transforms (highest value).

Table 6.6: Implementation results of the proof of concept $T_{8\times8p}$ transform core in a Xilinx Virtex-7 XC7VX485T-2FFG1761C FPGA device.

Processing structure	Registers		LUTs		Max. Freq.	
	#	$\Delta T_{8\times8}$	#	$\Delta T_{8\times8}$	MHz	$\Delta T_{8\times8}$
⊟ Transform core	6420	+0.0%	22972	+7.1%	279.8	-0.3%
⊞ IB	1681	+0.0%	1254	+0.0%	344.8	-0.0%
⊟ TA (with 8×8 PEs)	1501	+0.0%	15857	+0.8%	294.7	-0.6%
⊞ PE	70	+0.0%	254	+0.0%	298.5	-0.0%
⊞ TS	1462	+0.0%	3047	+41.0%	335.7	-6.8%
⊞ CU	97	+0.0%	86	+11.7%	365.2	-0.3%

maximum clock frequency of 279.8 MHz. This value is very similar to the one attained by its corresponding non-parallel design (i.e. $T_{8\times8}$), as shown in Table 6.3. Therefore, it can be concluded that the devised coarse-grain data-level parallelism technique has a negligible impact (less than 0.3%) in the clock frequency of the implemented transform cores. In line with the previous discussion, this was already expected owing to the fact that the maximum clock frequency of the proposed MTA is mostly limited by the critical path of its PEs, rather than by the critical path of the TS or CU.

As a consequence, the $T_{8\times8p}$ proof of concept transform core is able to compute up to 17.91 GOPS by using a clock frequency of 279.8 MHz, while offering a maximum throughput of 8 samples per clock cycle. Such values correspond to the computation of the 8×8 transforms, in which each of the 64 PEs that compose the TA computes one different transform operation at each clock cycle. For the computation of the smaller 4×4 and 2×2 transforms the resulting throughput is also 8 samples per clock cycle, since 2 different transforms (or 4 transforms in the last case) are simultaneously computed within the datapath of the implemented parallel architecture. However, only half of the PEs (a quarter of the PEs for the 2×2 transforms) are used in such computations, which constrains the offered computational rate to 8.95 GOPS and 4.48 GOPS, respectively. Nevertheless, by using the devised parallel processing structure it is possible to compute one 8×8 transform in 16 clock cycles, two 4×4 transforms in 8 clock cycles or four 2×2 transforms in 4 clock cycles, where the period of the clock signal is 3.57 ns. This results in speedup values of about 1, 2 and 4 times for the computation of the 8×8, 4×4 and 2×2 transforms regarding the $T_{8\times8}$ processing structure, whose clock cycle period is 3.56 ns (see Table 6.3). It should be noted that such proof of concept transform core is able to compute only one 8×8, 4×4 and 2×2 transform at a time, which requires also 16, 8 and 4 clock cycles, respectively.

The data corresponding to the used hardware resources that is presented in Table 6.6 also shows that the hardware requirements of the $T_{8\times8p}$ and $T_{8\times8}$ proof of concept processing structures are very similar and small. More specifically, less than 8% of the total hardware resources available in the considered medium-sized FPGA are used in the implementation of the $T_{8\times8p}$ transform core. The minor differences that can be observed between the two hardware structures

mostly concern to the resources required to implement the TS, as a result of all the extra multiplexers that the $T_{8\times8p}$ includes to support the implemented coarse-grain data-level parallelism technique. Although such extra resources cause an increase of about 40% in the hardware cost of the TS, it only augments the total hardware cost of the transform core in about 7%. This results from the fact that the majority of the consumed hardware resources are used in the implementation of the TA (about 74%). Conversely, the hardware cost of the 8×8 PEs TA and of the CU are almost identical in both structures, due to the quite negligible impact of the inherent modifications that were implemented in their hardware structures. Such modifications mostly consisted in *i*) the addition of some extra buses in the TA, in order to feed the TS with the output data of all the smaller transforms; and *ii*) the definition of earlier stop points in the control algorithm implemented by the CU (see section 4.4.2).

From the previous discussion it can be concluded that the coarse-grain data-level parallel processing capabilities improve the computational performance of an 8×8 transform core in about 2 times for the computation of the 4×4 transforms and in about 4 times for the computation of the 2×2 transforms. This allows the considered $T_{8\times8p}$ proof of concept transform core to compute the whole set of H.264/AVC transforms in real time for video sequences in the 4k UHDTV format (3840×2160 @ 60 fps), when operated using a clock frequency of 279.8 MHz. In addition, it can be observed that the hardware efficiency of this processing structure is also greatly enhanced, i.e. about 50% for the computation of the 4×4 transforms and about 25% for the computation of the 2×2 transforms. Such improvements are the result of the application of the considered coarse-grain data-level parallelism technique, which imposes a marginal increase of about 7% in the global hardware cost of the parallel architecture.

6.2.2 Comparative analysis and discussion

In order to further evaluate the advantages that are offered by the proposed MTA when compared with other alternative solutions for the design of video codecs supporting the H.264/AVC standard, several related designs described in the literature were thoroughly analysed. Besides offering different functionalities, such hardware realizations are based on a diverse set of implementation technologies and involve distinct design considerations. Consequently, the performed study is mostly focused on the assessment of the functionalities and processing capabilities offered by the various structures, in order to achieve a comparison as fair as possible. Accordingly, the presented comparative analysis uses as figures of merit the type and the amount of supported transforms, as well as the latency (L) (defined in Clock Cycles (CCs)), the throughput (T) (computed as the number of samples (S) processed in one CC), and the real time encoding and decoding capabilities of all the considered transform cores. Table 6.7 and Table 6.8 summarize the results of the conducted evaluation, which concerns only the implementation of the transform computation module(s) of all the reviewed designs.

A straightforward analysis of the collected data reveals that almost all the reviewed designs can be used to process the same video formats in real time. However, it also evidences that besides the four devised proof of concept transform cores only a few of such hardware structures[41,60] have the capability to compute all the transforms defined in the H.264/AVC standard. In fact, the vast majority of the analysed processing structures only support restricted subsets of the H.264/AVC transforms, since they were especially designed either to accelerate the computation of specific transforms (i.e. the 8×8 or the 4×4 DCTs) or optimized for a given transform coding path of a video encoder or decoder. Consequently, their application domain is rather limited, which makes them unsuitable to implement video encoders and decoders fully compliant with the H.264/AVC standard.

By analysing the presented results in terms of the offered performance levels, the comparison that is presented in Table 6.7 clearly shows that the proposed MTA allows designing transform cores capable of attaining some of the highest processing rates (2.2×10^9 S/s), despite being one of the few designs with the ability to compute all the H.264/AVC transforms. In fact, it can be observed that the implemented $T_{8\times8}$ and $T_{8\times8p}$ transform cores outperform almost all the other considered architectures by, at least, about 1.7 times. The only exception is the transform core presented in[115]. In addition, the designs described in[41,79] can also provide higher performance levels, since they consist of pipelined direct 2-D hardware structures requiring only half of the clock cycles to compute the considered transforms. Nevertheless, it is important to observe that none of these processing structures is able to compute all the H.264/AVC transforms and that they consist of specifically optimized and dedicated hardware structures, which means that their higher performance (and lower hardware cost) is the result of several algorithmic optimizations. As a consequence, and contrasting to the proposed MTA, all these optimizations considerably restrict the flexibility offered by these architectures and prevent the implementation of any changes or adaptations to their structure, in order to extend their functionality to fully support the H.264/AVC standard.

The previous observations are still valid when the devised proof of concept transform cores are compared with the considered alternative designs implemented using ASIC technologies. As it can be seen in Table 6.8, most of the enumerated processing structures offer performance levels similar to those presented by the $T_{8\times8}$ and $T_{8\times8p}$ transform cores. The only exceptions are the designs presented in[49,60,73]. Nevertheless, these faster processing structures can only compute a reduced subset of the transforms supported by the devised proof of concept transform cores (although many also offer multi-standard functionality), since they were either specifically designed to accelerate the inverse 8×8 DCT[73], optimized for the implementation of the forward transform coding path of a video encoder[49], or consist of a highly optimized and extremely hardware costly parallel design based on multiple matrix factorizations and permutations[60].

The results presented in Table 6.7 and in Table 6.8 also demonstrate that the latency that is

Table 6.7: Comparison with other related transform architectures implemented in FPGA devices. The letters *F* and *I* preceding the transform names in the column *Supported Transforms* are used to denote a forward or an inverse transform, respectively. The hardware cost is assessed in terms of the number of required Virtex slices for implementations based on Xilinx FPGA devices and in terms of LEs for the hardware realisations based on Altera FPGAs.

Design	Supported Standards	Supported Transforms	L (CCs)	T (S/CC)	Technology	HW Cost ($\times 10^3$)	Max. Freq. (MHz)	Proc. Rate ($\times 10^9$ S/s)	Target Applications
[79] (area)	H.264/AVC	$FDCT_{4\times4}$	7	0.14	Virtex-2 Pro	103	153.9	0.02	CIF
[79] (speed)	H.264/AVC	$FDCT_{4\times4}$	5	16	Virtex-2 Pro	644	107.5	1.72	4k UHDTV
[1]	H.264/AVC	$FDCT_{4\times4}$	32	1	Stratix	216	168.9	0.17	1080p HDTV
[1]	H.264/AVC	$IDCT_{4\times4}$	32	1	Stratix	465	168.9	0.16	1080p HDTV
[41]	H.264/AVC	$FDCT_{8\times8}$, $FDCT_{4\times4}$, $H_{4\times4}$	8	8	Virtex-4	2100	167.0	1.34	4k UHDTV
[41]	H.264/AVC	$IDCT_{8\times8}$, $IDCT_{4\times4}$, $H_{4\times4}$	8	8	Virtex-4	2100	133.5	1.07	4k UHDTV
[115]	H.264/AVC	$FDCT_{8\times8}$, $FDCT_{4\times4}$	1	64	Stratix	5143	100.0	6.40	4k UHDTV
[115]	H.264/AVC	$IDCT_{8\times8}$, $IDCT_{4\times4}$	1	64	Stratix	5258	100.0	6.40	4k UHDTV
[58]	H.264/AVC	$FDCT_{4\times4}$	8	4	Virtex-2 Pro	155	234.2	0.94	4k UHDTV
[58]	H.264/AVC	$IDCT_{4\times4}$	8	4	Virtex-2 Pro	259	190.1	0.76	4k UHDTV
[58]	H.264/AVC	$FH_{4\times4}$, $FH_{2\times2}$	8	8	Virtex-2 Pro	426	215.5	1.72	4k UHDTV
[58]	H.264/AVC	$IH_{4\times4}$, $IH_{2\times2}$	8	4	Virtex-2 Pro	276	242.1	0.97	4k UHDTV
[97]	H.264/AVC, VC-1, AVS, MPEG-4	real $IDCT_{8\times8}$, $IDCT_{4\times4}$, $H_{4\times4}$, $H_{2\times2}$	19 / 18 / 17	8	Virtex-4	1217	110.8	0.89	4k UHDTV
$T_{8\times8}$	H.264/AVC	$FDCT_{8\times8}$, $IDCT_{8\times8}$, $FDCT_{4\times4}$, $IDCT_{4\times4}$, $H_{4\times4}$, $H_{2\times2}$	16 / 8 / 4	8	Virtex-7	5363	280.6	2.24	4k UHDTV
$T_{8\times4}$	H.264/AVC	$FDCT_{8\times8}$, $IDCT_{8\times8}$, $FDCT_{4\times4}$, $IDCT_{4\times4}$, $H_{4\times4}$, $H_{2\times2}$	32 / 16 / 8	4	Virtex-7	3398	288.7	1.15	4k UHDTV
$T_{8\times2}$	H.264/AVC	$FDCT_{8\times8}$, $IDCT_{8\times8}$, $FDCT_{4\times4}$, $IDCT_{4\times4}$, $H_{4\times4}$, $H_{2\times2}$	64 / 32 / 16	2	Virtex-7	2383	290.1	0.58	1080p HDTV
$T_{8\times8p}$	H.264/AVC	$FDCT_{8\times8}$, $IDCT_{8\times8}$, $FDCT_{4\times4}$, $IDCT_{4\times4}$, $H_{4\times4}$, $H_{2\times2}$	16 / 8 / 4	8	Virtex-7	5743	279.8	2.24	4k UHDTV

Table 6.8: Comparison with other related transform architectures implemented as ASICs. The letters *F* and *I* preceding the transform names in the column *Supported Transforms* are used to denote a forward or an inverse transform, respectively. The hardware cost is assessed in terms of the number of two input NAND gate equivalent gates.

Design	Supported Standards	Supported Transforms	L (CCs)	T (S/CC)	Technology	HW Cost ($\times 10^3$)	Max. Freq. (MHz)	Proc. Rate ($\times 10^9$ S/s)	Target Applications
[13]	H.264/AVC	FDCT$_{4\times4}$, IDCT$_{4\times4}$, H$_{4\times4}$	2	8	180 nm	6.48	100.0	0.80	DCI 4k
[73]	H.264/AVC	IDCT$_{8\times8}$	1	8	180 nm	–	300.0	2.40	4k HDTV
[60]	H.264/AVC	FDCT$_{8\times8}$, IDCT$_{8\times8}$, FDCT$_{4\times4}$, IDCT$_{4\times4}$, H$_{4\times4}$, H$_{2\times2}$	2	20.5	180 nm	63.62	200.0	4.10	4k UHDTV
[49]	H.264/AVC	FDCT$_{4\times4}$, H$_{4\times4}$	2	16	130 nm	10.40	363.6	3.66	4k UHDTV
[49]	H.264/AVC	FDCT$_{8\times8}$	16	8	130 nm	8.46	367.7	0.91	4k UHDTV
[127]	H.264/AVC, VC-1, AVS, MPEG-2/4, HEVC	IDCT$_{4\times4}$, IDCT$_{8\times8}$, IDCT$_{16\times16}$, IDCT$_{32\times32}$	2 / 3 / 4 / 5	4	130 nm	10.92	350.0	1.40	4k UHDTV
[104]	H.264/AVC, VC-1, AVS, MPEG-2, JPEG	IDCT$_{8\times8}$	–	1	180 nm	45.90	202.8	0.20	DCI 4k
[105]	H.264/AVC, HEVC	IDCT$_{8\times8}$	–	1	180 nm	12.30	211.4	0.21	1080p HDTV
[50]	H.264/AVC, VC-1, AVS, MPEG-2/4, H.263	IDCT$_{4\times4}$, IDCT$_{8\times8}$	3	4 / 2	130 nm	56.99	384.0	1.54 / 0.77	4k UHDTV
[139]	H.264/AVC, VC-1, AVS, MPEG-2/4	FDCT$_{8\times8}$, IDCT$_{8\times8}$, FDCT$_{4\times4}$, IDCT$_{4\times4}$	3	8	130 nm	23.06	100.0	0.80	1080p HDTV
[138]	H.264/AVC, VC-1, AVS, MPEG-2, JPEG	IDCT$_{8\times8}$	8	1	180 nm	12.60	146.0	0.15	1080p HDTV

imposed by the implemented transform cores in the processing of the video bit streams is not only low but also similar to the one imposed by the majority of the considered alternative designs computing the same transforms. This is a very important aspect when real time operation is considered, since the overall delay that is imposed in the encoding of each video frame may compromise the operation of the codec when employed in interactive application domains (e.g. live broadcast or video conference).

6.3 Evaluation of the proposed quantization architectures

This section discusses the experimental results that were obtained by implementing the proof of concept quantizers listed in Table 6.2 in the considered Virtex-7 FPGA device. Relative assessment is also presented by considering the related state of the art and implementations of the devised hardware structures using a Xilinx Virtex II Pro FPGA.

6.3.1 FPGA implementation results

Table 6.9 summarizes the obtained performance and hardware cost results for the fifteen considered quantization circuits. Such data shows the rather small and similar hardware requirements of the forward, inverse and unified quantizers for any of its possible four architecture configurations, despite the offered flexibility in terms of functionality. In fact, the minor differences that can be observed among the five considered implementations of each quantizer result from two distinct factors: *i*) the hardware resources that are required to implement the rounding adder in the non-pipelined implementations, since the minimal latency offered by these architecture configurations prevents making use of the 48 bits adder that is embedded in the `DSP48E1` slices; *ii*) the extra resources that are used to partition the datapath into multiple stages in the pipelined configurations. Still, the implementation of the faster configurations of the devised proof of concept quantization circuits requires only a single `DSP48E1` slice (to jointly compute the multiplication and rounding operations) and between 135 and 301 ordinary Virtex-7 slices. This consists of less than 1% of the total capacity of the adopted medium size FPGA device[146]. Therefore, it can be expected that the hardware requirements of the proposed class of quantization architectures do not compromise its use in the design of H.264/AVC encoders or decoders.

In what concerns the performance, the maximum clock frequency values presented in Table 6.9 evidence the high processing rates that can be attained by using the proposed quantization architectures, i.e. up to 464.7×10^6 coefficients per second. According to such results, it can be concluded that the devised proof of concept quantizers are able to comply (at least) with the real time processing requirements of the 1080p HDTV format (1920×1080 @ 30fps). This is illustrated in Figure 6.2, which shows the upper bound limits for the processing rates that are offered by each configuration of the three quantizers, when operated using their maximum clock

Table 6.9: Implementation results of the considered proof of concept quantizers in a Xilinx Virtex-7 XC7VX485T-2FFG1761C FPGA device.

Design	L (CCs)	T (S/CC)	FPGA Utilization			Max Freq. (MHz)	Applications
			FFs	LUTs	DSP48E1s		
FQ_{NP}	1	1	0	165	1	126.0	1080p HDTV
FQ_{P2}	2	1	4	115	1	154.3	1080p HDTV
FQ_{P3}	3	1	18	125	1	249.6	1080p HDTV
FQ_{P4}	4	1	26	121	1	367.4	DCI 4k
FQ_{P5}	5	1	40	119	1	414.9	4k UHDTV
IQ_{NP}	1	1	0	170	1	125.3	1080p HDTV
IQ_{P2}	2	1	5	137	1	144.9	1080p HDTV
IQ_{P3}	3	1	20	145	1	224.7	1080p HDTV
IQ_{P4}	4	1	35	127	1	356.5	DCI 4k
IQ_{P5}	5	1	44	130	1	464.7	4k UHDTV
UQ_{NP}	1	1	0	310	1	119.0	1080p HDTV
UQ_{P2}	2	1	6	248	1	135.1	1080p HDTV
UQ_{P3}	3	1	24	256	1	201.3	1080p HDTV
UQ_{P4}	4	1	30	257	1	343.2	DCI 4k
UQ_{P5}	5	1	48	253	1	394.5	4k UHDTV

frequencies.

As it can be observed, the non-pipelined configurations allow the processing of more than 119×10^6 coefficients per second, while the throughput offered by the fastest pipelined configuration is about 400×10^6 coefficients per second and enough to support the processing in real time of the 4k UHDTV format (3840×2160 @ 30fps). The differences in the attained performance are owed not only to the application of the multi-stage pipeline technique but also to a better use of the `DSP48E1` slice in the pipelined configurations. As it was previously mentioned, the `DSP48E1` slices are exclusively employed to implement fast multipliers for the non-pipelined configurations, therefore avoiding the much slower and hardware costly LUT-based implementations. Conversely, the `DSP48E1` macrocells are used to implement a very optimized and fast MAC unit in the pipelined configurations, where some of the pipeline registers are pushed into the `DSP48E1` slice due to the synthesis tool being capable to successfully infer the MAC coding pattern. In such implementations, the observed performance differences between the 4 pipelined designs are owed to the different amounts of pipeline stages that can be fully implemented within a `DSP48E1` slice. Hence, by exploiting the internal registers of a `DSP48E1` slice to implement more pipeline registers, it becomes possible to have the `DSP48E1` MAC unit operating at much higher clock frequencies, and thus to significantly improve the global circuit performance without greatly increasing its hardware cost.

The implementation results presented in Table 6.9 also demonstrate that the five possible architecture configurations that can be used to realize forward, inverse and unified quantizers effectively allow to trade-off performance for hardware cost. As it can be seen in Figure 6.3, the amount of hardware resources that are required by such processing structures scales well with the increase of its maximum clock frequencies. In fact, while the 3-stages pipelined implementa-

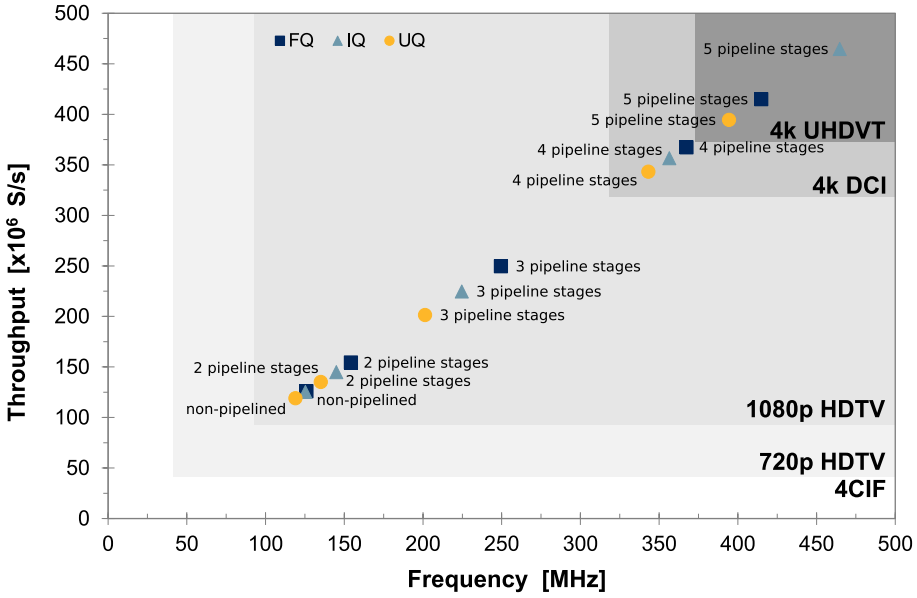


Figure 6.2: Performance comparison of the several implemented quantizers, when operated at their maximum clock frequency.

tions cost about 10 more slices than the 2-stages pipelined designs, the performance attained by these structures is also over 50% higher than that offered by their corresponding configurations with 2 pipeline stages. The comparison between the deeper pipelined designs reveals a similar tendency, but with not so high relative increases both in the performance and hardware cost.

The use of the proposed UQA in the design of a video coding system also allows to trade-off performance for hardware cost in a different perspective. In fact, an encoding system that includes a single instance of such unified quantization circuit is able to compute both the forward and the inverse quantization operations with some savings in terms of hardware cost, when compared to another system that uses two independent and dedicated functional units (FQ and IQ) to realize the same operations, i.e. up to 7.5% of the LUTs, between 33% and 50% of the flip-flops and 50% of the `DSP48E1` slices (see Table 6.9). This is especially relevant not only when such video coding systems are implemented using FPGA devices, in which designs with higher hardware costs usually demand the use of larger (and thus more expensive) devices, but also when realized as ASICs due to the final cost of the corresponding Integrated Circuits (ICs) being directly and significantly affected by the amount of hardware resources that are required to implement the quantizers.

Nonetheless, the advantages offered by the UQA in terms of hardware cost also result in some inherent degradation of the performance of the video coding systems. Such performance penalty is owed to the lower clock frequencies provided by the devised UQA, which are at least 3% lower than those offered by the proposed FQA and IQA, as well as to the time share use of the unified quantizer. In fact, it should be noted that a single design based on two dedicated functional units offers the added possibility of having the forward and inverse quantization modules

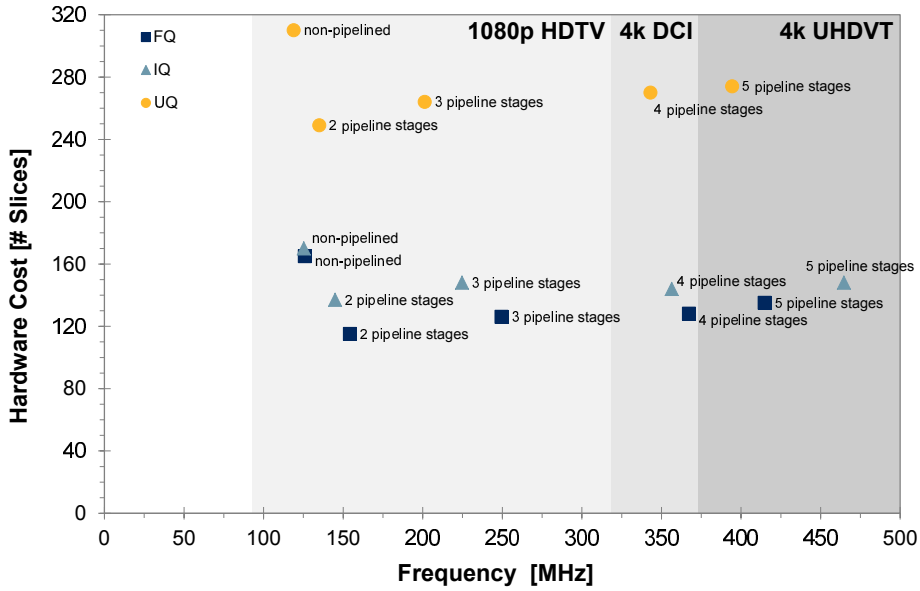


Figure 6.3: Comparison of the hardware cost of the several implemented quantizers.

working in parallel, while the one using a single instance of the proposed UQA can only process one of the two operations for a block of coefficients at a given time instant. Consequently, the data processing rate of this unified system is actually reduced by up to 60% regarding the more hardware costly one. Still, the high performance levels that can be attained by using the proposed UQA allow alternating its operation mode along the time, in order to compute the forward and inverse quantization operations for the most typical application scenarios, i.e. the processing in real time of video sequences up to the 1080p HDTV format. Hence, it can be concluded that this alternative use of the proposed UQA is most suitable for the implementation of reduced complexity and cost video coding systems with moderate requirements in terms of performance.

6.3.2 Comparative analysis and discussion

In order to evaluate the benefits of the proposed class of quantization architectures when compared with other alternative solutions, the most related and prominent designs described in the literature were revised and analysed. Such hardware structures are listed in Table 6.10 and Table 6.11, which summarize the results of the performed comparative study grouped by implementation technology, i.e. FPGA or ASIC, respectively.

As it can be seen, besides offering different functionalities, the reviewed quantization circuits were implemented using multiple FPGA and ASIC technologies. Furthermore, in most cases, such implementations were achieved by taking into account several different design considerations and optimizations. Consequently, an extra effort had to be devised to perform the presented discussion, in order to guarantee a comparison as fair as possible. More specifically, the conducted study not only assesses the functionalities and the processing rates that are offered by the various designs but also compares the hardware efficiency of those implemented in FPGA

Table 6.10: Comparison with other related quantization architectures implemented in FPGA devices.

Design	Supported Operations	Supported Block Sizes	L (CCs)	T (S/CC)	Technology	HW Cost				Max. Freq. (MHz)	DTUA ($\times 10^6$)		Applications
						FFs	LUTs	Slices	Other		S/LUT/s	S/Slice/s	
[79] (area)	FQ	4×4	4	0.25	Virtex-2 Pro	176	-	143	-	135.2	-	0.24	720p HDTV
[131]	FQ	4×4	8	4	Virtex-2 Pro	-	-	-	-	115.0	-	-	4k UHDTV
[57]	FQ	4×4	3	4	Virtex-2 Pro	-	965	-	-	107.6	0.45	-	DCI 4k
[43]	FQ	4×4, 8×8	1	8	Virtex-2 Pro	-	-	5104	-	297.0	-	0.47	8k UHDTV
[79] (speed)	FQ	4×4	1	16	Virtex-2 Pro	257	-	992	-	97.1	-	1.57	4k UHDTV
[115]	FQ	4×4, 8×8	1	32	Stratix II	-	2180	-	32 DSP	100.0	1.16	-	8k UHDTV
[57]	IQ	4×4	2	4	Virtex-2 Pro	-	443	-	-	136.1	0.45	-	DCI 4k
[115]	IQ	4×4, 8×8	1	32	Stratix II	-	2049	-	32 DSP	100.0	1.23	-	8k UHDTV
[86]	FQ, IQ	4×4	1	1	Virtex-2	-	-	-	-	38.0	-	-	4CIF
[77]	FQ, IQ	4×4	1	1	Virtex-2 Pro	-	-	-	-	82.0	-	-	720p HDTV
FQ_{NP}	FQ	4×4, 8×8	1	1	Virtex-2 Pro	0	311	161	1 MULT18X18	67.9	0.22	0.42	720p HDTV
FQ_{P2}	FQ	4×4, 8×8	1	2	Virtex-2 Pro	44	303	159	1 MULT18X18	94.3	0.31	0.59	1080p HDTV
FQ_{P3}	FQ	4×4, 8×8	1	3	Virtex-2 Pro	89	292	153	1 MULT18X18	144.8	0.50	0.95	1080p HDTV
FQ_{P4}	FQ	4×4, 8×8	1	4	Virtex-2 Pro	114	295	165	1 MULT18X18	174.6	0.59	1.06	1080p HDTV
FQ_{P5}	FQ	4×4, 8×8	1	5	Virtex-2 Pro	144	280	167	1 MULT18X18	174.6	0.62	1.05	1080p HDTV
IQ_{NP}	IQ	4×4, 8×8	1	1	Virtex-2 Pro	0	287	149	1 MULT18X18	73.4	0.26	0.49	720p HDTV
IQ_{P2}	IQ	4×4, 8×8	1	2	Virtex-2 Pro	30	265	139	1 MULT18X18	109.1	0.41	0.78	1080p HDTV
IQ_{P3}	IQ	4×4, 8×8	1	3	Virtex-2 Pro	71	257	135	1 MULT18X18	144.8	0.56	1.07	1080p HDTV
IQ_{P4}	IQ	4×4, 8×8	1	4	Virtex-2 Pro	124	268	145	1 MULT18X18	180.3	0.67	1.24	1080p HDTV
IQ_{P5}	IQ	4×4, 8×8	1	5	Virtex-2 Pro	139	251	143	1 MULT18X18	204.5	0.81	1.44	1080p HDTV
UQ_{NP}	FQ, IQ	4×4, 8×8	1	1	Virtex-2 Pro	0	638	329	1 MULT18X18	64.2	0.10	0.20	720p HDTV
UQ_{P2}	FQ, IQ	4×4, 8×8	1	2	Virtex-2 Pro	46	597	311	1 MULT18X18	92.3	0.15	0.30	1080p HDTV
UQ_{P3}	FQ, IQ	4×4, 8×8	1	3	Virtex-2 Pro	105	605	313	1 MULT18X18	126.6	0.21	0.40	1080p HDTV
UQ_{P4}	FQ, IQ	4×4, 8×8	1	4	Virtex-2 Pro	160	597	313	1 MULT18X18	154.4	0.26	0.49	1080p HDTV
UQ_{P5}	FQ, IQ	4×4, 8×8	1	5	Virtex-2 Pro	185	571	304	1 MULT18X18	174.6	0.31	0.57	1080p HDTV

Table 6.11: Comparison with other related quantization architectures implemented as ASICs.

Design	Supported Operation(s)	Supported Block Sizes	L (CC)	T (S/CC)	Technology	HW Cost ($\times 10^3$ gates)	Max. Freq. (MHz)	Applications
[149]	FQ	4×4	4	4	SMIC 0.18 μm	25.56	250.0	4k UHDTV
[112]	FQ	4×4	1	16	UMC 0.13 μm	-	156.5	4k UHDTV
[117]	FQ	4×4	2	16	SMIC 0.18 μm	20.96	160.5	4k UHDTV
[115]	FQ	4×4	1	32	AMS 0.35 μm	77.67	79.0	DCI 4k
[115]	IQ	4×4	1	32	AMS 0.35 μm	55.39	79.0	DCI 4k

devices with the devised proof of concept quantizers, by using the Data Throughput per Unit of Area (DTUA) metric.

The DTUA performance index is defined as the ratio between the data throughput, in processed samples (S) per second, and the hardware cost, in terms of unit of area. Although the number of slices is generally used to represent the unit of area in FPGA implementations[41], the number of LUTs was also adopted as the unit of area in the presented comparative analysis, due to the fact that this is the only data that was reported for some of the considered state-of-the-art designs. Moreover, since almost all the reviewed designs implemented using reconfigurable logic are based on Xilinx Virtex II FPGAs, whose slices and LUTs are rather different from the ones available in the the adopted Virtex-7 prototyping device, the considered proof of concept quantizers were also re-synthesized for a XC2VP30 FPGA. Such task was conducted using the Xilinx ISE 10.1i development toolchain[144] and by following the same synthesis strategy described in section 6.1, without any further or specific optimizations to the involved VHDL circuit descriptions. The obtained implementation results are presented in the bottom of Table 6.10.

A straightforward analysis of the data presented in Table 6.9, Table 6.10 and Table 6.11 shows that only the devised proof of concept quantizers and a few other hardware structures[43,115] are capable of processing the two types of quantization blocks defined in the H.264/AVC standard, and thus to comply with the requirements of its High Profiles. In addition, the obtained experimental data reveals that, despite the extra logic that is required to provide such enhanced functionality (which also imposes a small penalty in the offered computation rates), the proposed class of quantization architectures can be used to realize quantization circuits with a reduced hardware cost that are capable of processing in real time the same video formats as the reviewed designs, when implemented in a Xilinx Virtex-7 FPGA device. The only exception is the set of structures presented in[43,115], which can also be used to perform the encoding and the decoding of video sequences in the 8k UHDTV format. However, it is important to observe that these designs consist of highly parallel processing structures that share many of their hardware resources to compute a single quantization operation. Nevertheless, it should be observed that this same approach can also be adopted to design higher performance parallel forward, inverse and unified quantizers based on the proposed class of quantization architectures. To achieve such goal, it would only be required to combine several instances of the same quantization circuit in a single quantizer and have them operating in parallel, in order to simultaneously process more than one coefficient at each clock cycle. For example, the throughput of a forward quantizer using eight instances of FQ_{P3} can be as high as 2.0×10^6 S/s, while an inverse quantizer using eight instances of IQ_{P3} should be capable of processing up to 1.8×10^6 S/s. These processing rates are enough to fulfil the real time encoding and decoding requirements of the 8k UHDTV format.

Naturally, the hardware cost of such parallel processing structures would also be higher. However, this should not constitute a problem when fully implementing parallel forward, inverse and

unified quantizers using FPGA devices (or ASIC technologies), owing to the quite diminished hardware requirements of the proposed class of quantization architectures (even when its higher performance configurations are considered). For instance, the hardware cost of the two parallel quantizers mentioned above could be as high as 1008 slices or 1184 slices, respectively, when implemented using the considered Virtex-7 prototyping device. This consists of less than 2% of the total amount of hardware resources available in such device. Nevertheless, it is still possible to reduce the hardware cost of these parallel quantizers, by having many of their hardware resources shared among the multiple instances of the base quantization circuit (i.e. all the logic except the ROMs storing the MF and V scaling factors, the multiplier, the rounding adder and the barrel-shifter depicted in Figure 5.1, Figure 5.2 and Figure 5.3). Such optimizations would also allow improving their hardware efficiency, which is already relatively high when implemented in a XC7VX485T-2FFG1761C FPGA, i.e. between 0.8×10^6 S/slice/s and 3.1×10^6 S/slice/s for the forward and inverse proof of concept quantizers and between 0.4×10^6 S/slice/s and 1.5×10^6 S/slice/s for the unified proof of concept quantizers (see the DTUA columns in Table 6.9).

When prototyped using the same technology as the vast majority of the reviewed state-of-the-art quantizers, i.e. a Xilinx Virtex II Pro FPGA, the obtained DTUA results are (at least) as good as those provided by the alternative designs. The only exceptions are the structures presented in[58,79] that achieve relatively higher DTUAs values. Regarding the forward quantizer proposed in[79], it is important to observe that it consists of a highly parallel processing structure capable of processing sixteen different coefficients at each clock cycle. Furthermore, the reported data strictly concerns the datapath. By taking into consideration also the hardware requirements (86 slices) and the latency (3.2 ns) imposed by the two LUTs contained in the control module of the quantizer, the DTUA of this architecture is reduced to 1.1×10^6 S/slice/s. Conversely, the hardware structure introduced in[58] consists of a 4-ways parallel architecture for the computation of the inverse quantization operation. Such quantizer is based on an optimized pipelined datapath with two stages that avoids the computation of the rounding operation to improve its performance. This optimization not only allows reducing the circuit latency in about 1.9 ns but also saving about 50% of its original hardware requirements. However, it also introduces some error in the de-quantization procedure, which does not happen when using the proposed inverse quantization architecture. Also, these two architectures can only process one of the two types of quantization blocks defined in the H.264/AVC standard, i.e. the 4×4 block type. Consequently, their use is restricted to the development of H.264/AVC codecs that do not fully support the high profiles of the standard, which is not acceptable for the design of the current and upcoming video coding systems aiming at the processing of the UHDTV formats. In contrast, the performance levels that are already offered by the considered proof of concept quantizers can be easily increased to match such demanding requirements without degrading the hardware efficiency, by having several instances simultaneously operating in parallel in a single quantization circuit, as discussed above.

6.4 Generalized design of integrated transform and quantization circuits

The discussion presented in the previous sections shows that the proposed transform and quantization architectures can be successfully used to realize dedicated processing structures for fast and efficient computation of all the transform and quantization operations defined in the H.264/AVC standard. Possible application scenarios for such processing structures therefore comprehend their integration in video encoders and decoders not only as specialized functional units of GPPs or ASIPs but also as hardware accelerators in SoCs or specialized processing cores in heterogeneous multi-core systems, where they can implement the necessary transform and quantization modules.

Nonetheless, the application scope of the proposed transform and quantization architectures is much broader, since they can also be combined in a single circuit to develop high performance and hardware efficient integrated transform and quantization processing structures. As mentioned in chapter 3, these processing structures present several important advantages in the design of efficient video coding systems, since they provide the means required to optimize several aspects of the codec operation (e.g. time synchronism, data manipulation, memory concurrency issues, etc.).

To attest the above observation, an integrated transform and quantization circuit based on the multi-transform and quantization architectures presented in chapters 4 and 5, respectively, is now discussed and theoretically evaluated in terms of the offered data processing capabilities. Figure 6.4 depicts the block digram of such processing structure, which not only is capable of computing all the H.264/AVC transforms and quantization procedures but also of autonomously realizing all the operations defined in all the forward and inverse hierarchical transform coding paths specified by the standard, i.e. the default forward and inverse transform and quantization paths, the $Intra_{16\times16}$ forward and inverse transform and quantization paths, the optional forward and inverse transform and quantization paths based on the 8×8 DCT that can be employed in the H.264/AVC high profiles, and the corresponding complete coding loops. The set of operations computed in each one of the supported coding paths is detailed in Table 6.12.

As it can be seen in Figure 6.4, the proposed circuit consists of a dedicated processing structure based on two distinct computational modules, i.e. a `Transform Engine (TE)` and a `Quantization Engine (QE)`, which are interconnected using a quite flexible and configurable hardware structure. The `TE` is used to perform the computation of all the possible H.264/AVC transforms, while the `QE` is employed for the realization of the necessary forward and inverse quantization operations for the two possible block sizes. The `Control Unit` is responsible for commanding the operation of all the circuits, including the dynamic configuration of the interconnection structure to implement the several coding paths, as well as the establishment of an

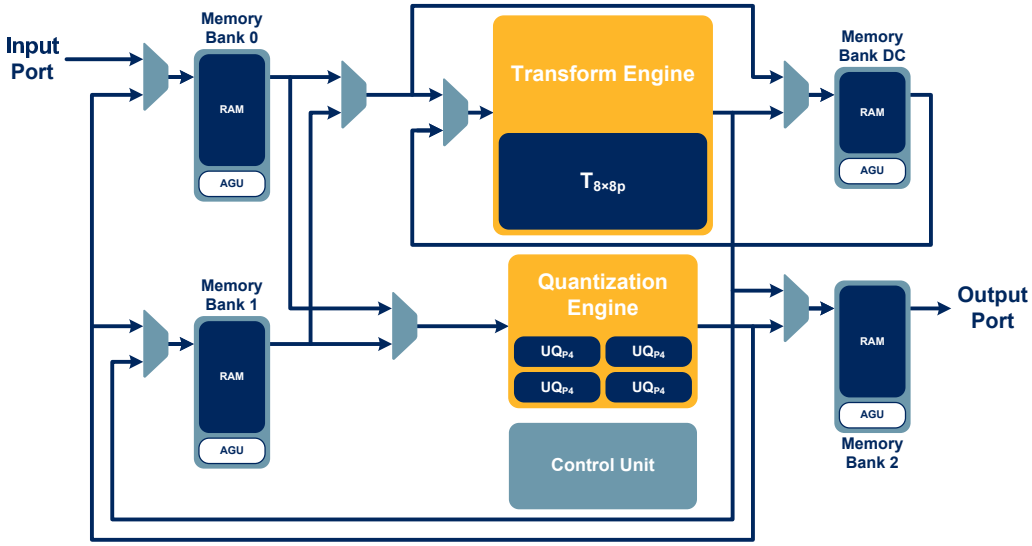


Figure 6.4: Block diagram of the considered integrated transform and quantization circuit for the H.264/AVC standard.

efficient data streaming interface between the integrated transform and quantization circuit and the remaining modules of the video codec.

Although any configuration of the proposed multi-transform and quantization architectures can be used to implement the TE and the QE, the suggested proof of concept structure is based on the $T_{8\times8p}$ and UQ_{P4} setups described in Tables 6.1 and 6.2, respectively. This approach allows obtaining the desired functionality, while maximizing the overall hardware efficiency and performance of the circuit. Nevertheless, the presented processing structure further exploits two distinct design strategies to achieve such goals.

To compensate the different latencies and the maximum processing rates offered by the $T_{8\times8p}$ and UQ_{P4} hardware structures, the TE is implemented using a single instance of the $T_{8\times8p}$ architecture whilst four instances of the UQ_{P4} architecture are used to realize the QE. As a consequence, both modules are able to fully process the luma and chroma components of a MB in the same amount of time (t_{MB}), by using a single clock signal (CLK). Such time frame can be determined by using Equation 6.1, where T_{CLK} denotes the period of the circuit's clock signal and the number of clock cycles required by each engine to process the luma and chroma components of a MB (i.e. $CPMB_{TE}$ and $CPMB_{QE}$) is given by Equation 6.2 and Equation 6.4, respectively. In these equations, B represents the number of $N \times N$ blocks composing a MB, P_{TE} expresses the amount of transforms that are computed in parallel (given by Equation 6.3), while L_{TE}, L_{QE}, T_{TE} and T_{QE} denote the latency and the throughput of the TE and QE modules, respectively. For this particular implementation, $L_{TE} = N - 1$, $T_{TE} = 8$, $L_{QE} = 4$ and $T_{QE} = 4$.

$$t_{MB} = \max\{CPMB_{TE}, CPMB_{QE}\} \times T_{CLK} \tag{6.1}$$

$$CPMB_{TE} = L_{TE} + 2N \times \max\left(1, \frac{B}{P_{TE}}\right) + L_{TE} \tag{6.2}$$

Table 6.12: Operations supported by the several configurations of the considered integrated transform and quantization circuit for the H.264/AVC standard.

Processing path	Transform operations								Quantization operations			
	$FDCT_{8\times8}$	$IDCT_{8\times8}$	$FDCT_{4\times4}$	$IDCT_{4\times4}$	$FH_{4\times4}$	$IH_{4\times4}$	$FH_{2\times2}$	$IH_{2\times2}$	$FQ_{8\times8}$	$IQ_{8\times8}$	$FQ_{4\times4}$	$IQ_{4\times4}$
Default / $Intra_{16\times16}$ forward transform	○	○	●	○	●	○	●	○	○	○	○	○
Default inverse transform	○	○	○	●	○	●	○	●	○	○	○	○
Default forward quantization	○	○	○	○	○	○	○	○	○	○	●	○
Default inverse quantization	○	○	○	○	○	○	○	○	○	○	○	●
Default / $Intra_{16\times16}$ forward transform and quantization	○	○	●	○	●	○	●	○	○	○	●	○
Default / $Intra_{16\times16}$ inverse quantization and transform	○	○	○	●	○	●	○	●	○	○	○	●
Default / $Intra_{16\times16}$ complete coding loop	○	○	●	●	●	●	●	●	○	○	●	●
8×8 forward transform	●	○	○	○	○	○	○	○	○	○	○	○
8×8 inverse transform	○	●	○	○	○	○	○	○	○	○	○	○
8×8 forward quantization	○	○	○	○	○	○	○	○	●	○	○	○
8×8 inverse quantization	○	○	○	○	○	○	○	○	○	●	○	○
8×8 forward transform and quantization	●	○	○	○	○	○	○	○	●	○	○	○
8×8 inverse quantization and transform	○	●	○	○	○	○	○	○	○	●	○	○
8×8 complete coding loop	●	●	○	○	○	○	○	○	●	●	○	○

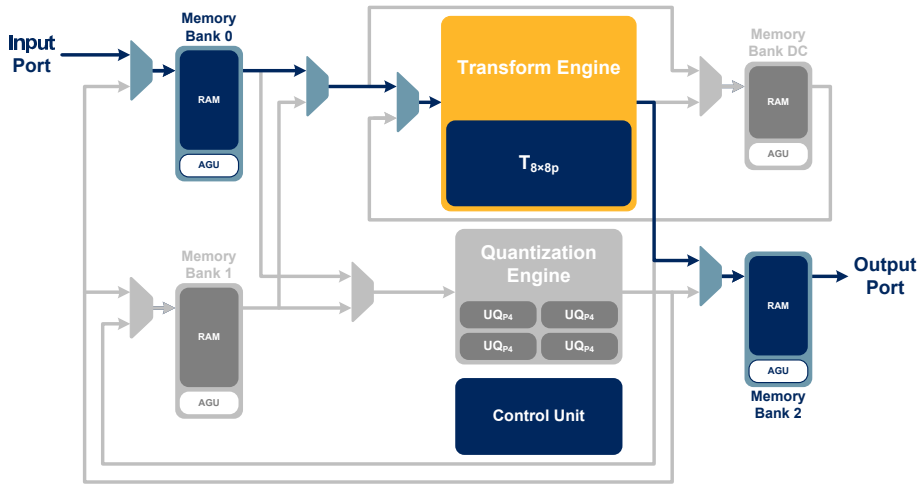
$$P_{TE} = \frac{T_{TE}}{N} \tag{6.3}$$

$$CPMB_{QE} = L_{QE} + \max\left(1, \frac{N^2}{T_{QE}}\right) \times B \tag{6.4}$$

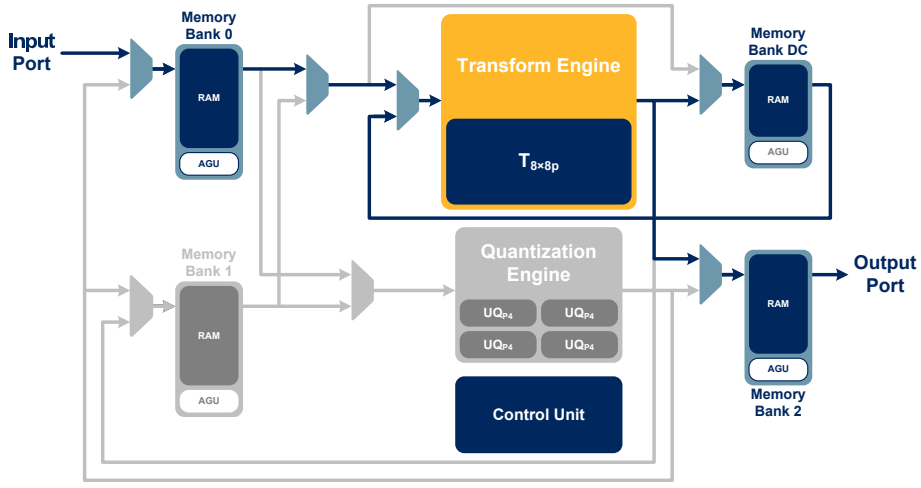
The transform and quantization engines are also interconnected in a fully configurable pipelined chain, by using four dual port memory banks with double buffering streaming capability and a set of multiplexers. Each of these buffers is capable of storing all the data corresponding to two consecutive MBs, which enables a high degree of parallel processing at the MB level for the following two reasons: *i*) the `TE` and the `QE` can be operated in parallel; *ii*) the processing of a MB can occur simultaneously with either the fetching of the data corresponding to the next MB to be processed, or with the output of the computation results corresponding to the previous MB.

To support the implementation of all the considered processing paths, `Memory Bank 0` can be connected either to the input data port of the circuit or to the output data port of the `QE`, as shown in Figure 6.5. In the first case, it is used to store the residual data of a MB when a forward coding path is configured, or the quantized transform coefficients when an inverse coding path is selected. In the second case, it initially holds the residual data of a MB and is later used to store the corresponding quantized transform coefficients that are produced by the `QE`, so as to support the implementation of the complete forward transform, forward quantization, inverse quantization and inverse transform loop of the codec. This consists of a two steps procedure in which the interconnection structure is initially configured to implement the forward coding path and is later reconfigured to implement the inverse coding path. In both cases, `Memory Bank 1` is used to buffer the data between the transform and quantization engines, as illustrated in Figure 6.5. Consequently, the data input port of `Memory Bank 1` can be connected to the output port of either the `TE` or the `QE`, depending on the processing path that has been selected. Conversely, `Memory Bank 2` is always connected to the circuit's output data port and is exclusively used to store the results generated in all the transform and quantization coding paths.

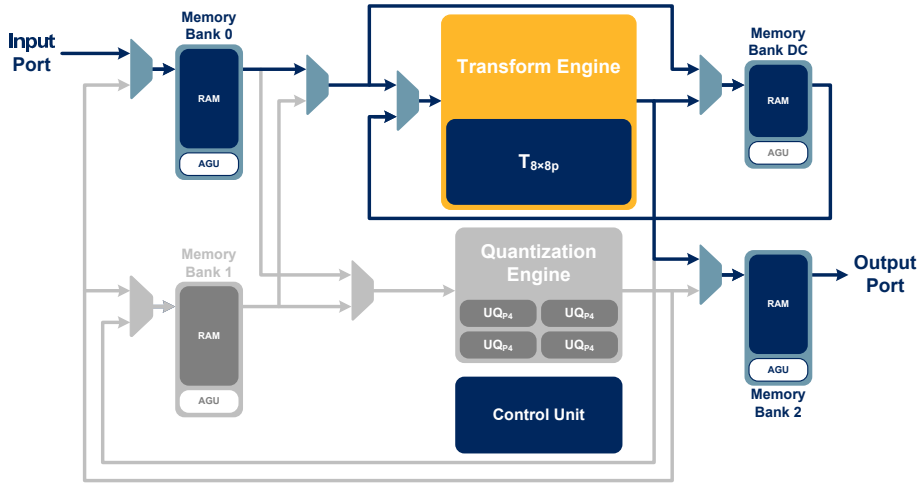
The smaller `Memory Bank DC` (capable of storing two 4×4 blocks of data) is only used in some of the considered dataflows, so that the second level Hadamard transforms can be computed more efficiently. Such improved efficiency mostly results from keeping the memory accesses local to the integrated transform and quantization circuit, and thus allowing the `TE` and the `Memory Bank 0` to operate in parallel. For the $Intra_{16\times16}$ forward processing paths, this approach allows to directly store in `Memory Bank DC` all the DC coefficients that are computed using the 4×4 DCT. The Hadamard transform coefficients can be immediately computed by using these values and the corresponding transform kernel, while the quantized transform coefficients for all the AC coefficients of the MB are also simultaneously computed by the quantization engine. As soon as both procedures are completed, the `QE` is reused to quantize the obtained Hadamard transform coefficients. Similarly, `Memory Bank DC` also stores the MB DC coefficients when the circuit is configured to implement the inverse transform and quantization path. However, such data is



(a) Forward and inverse transforms path.

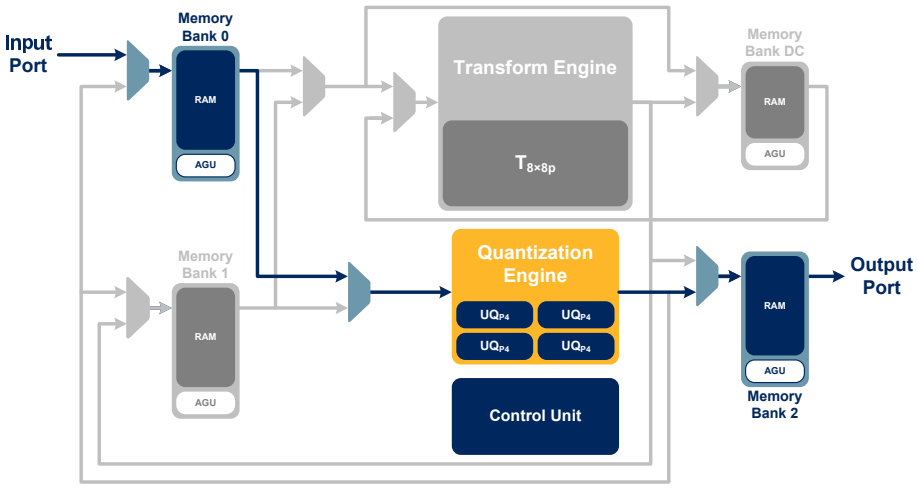


(b) $Intra_{16\times16}$ forward transform path.

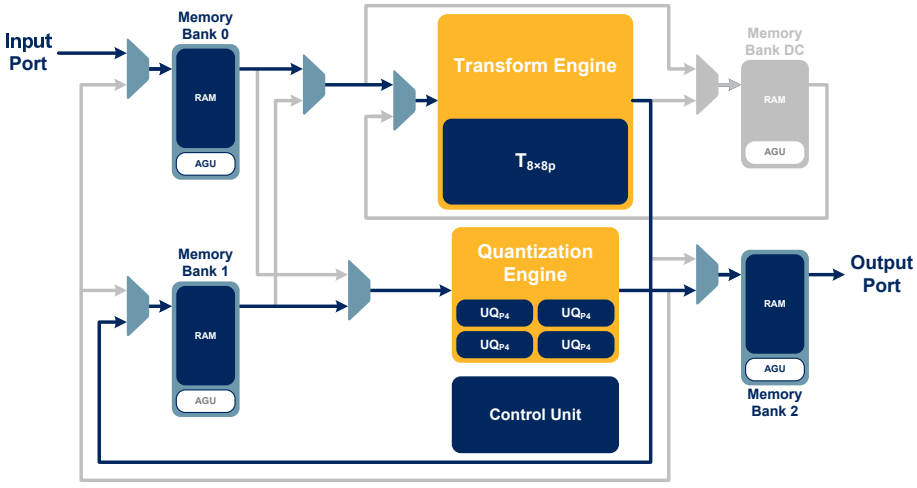


(c) $Intra_{16\times16}$ inverse transform path.

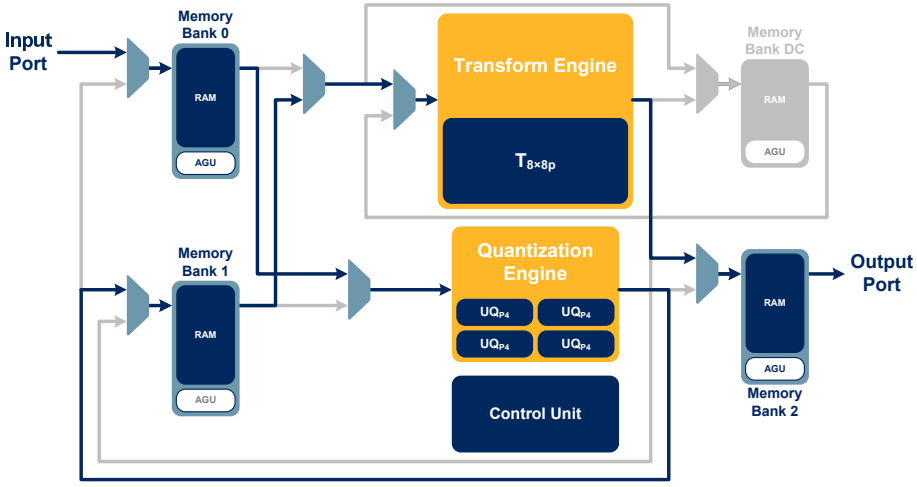
Figure 6.5: Processing paths supported by the considered integrated transform and quantization circuit. (Continued on the next page.)



(a) Forward and inverse quantization paths.



(b) Forward transform and quantization path.



(c) Inverse quantization and transform path.

Figure 6.5: Processing paths supported by the considered integrated transform and quantization circuit.

initially copied from `Memory Bank 0`, so that the computation of the inverse Hadamard transform can occur in parallel with the inverse quantization of the AC coefficients of the MB, which are also stored in `Memory Bank 0`. Immediately after the completion of the two procedures, the inverse 4×4 DCT is computed by combining the data stored in `Memory Bank 1` and `Memory Bank DC`, i.e. the scaled AC transform coefficients and the inverse transform DC coefficients.

By taking into account the fully pipelined processing scheme and the double buffering capabilities of the presented proof of concept integrated transform and quantization circuit, its performance can be theoretically evaluated based on the amount of Clock Cycles (CCs) that are required to encode and decode the several possible MB types defined in the H.264/AVC standard. Such amount of CCs can be evaluated by using Equation 6.5, where $CPMB_{TE}$ and $CPMB_{QE}$ are given by Equation 6.2 and Equation 6.4, respectively.

$$CPMB = \max\left\{CPMB_{TE_{luma}} + CPMB_{TE_{chroma}}, CPMB_{QE_{luma}} + CPMB_{QE_{chroma}}\right\} \quad (6.5)$$

Accordingly, Table 6.13 presents the amount of CCs that must be spent to process a single MB using the forward and inverse coding paths enumerated in Table 6.12, while Figure 6.6 depicts estimates of the data processing rates that can be attained by the proposed processing structure when it is operated using several different clock frequencies.

The performance analysis presented in Figure 6.6 suggests that the presented integrated transform and quantization circuit can be used not only to encode but also to decode video sequences in the 4k UHDTV format (3840×2160 @ 60 fps) in real time, by using a clock signal with a frequency of 240 MHz. Such value is lower than the maximum clock frequency values obtained for the implementation of the $T_{8\times8p}$ and UQ_{P4} proof of concept hardware structures in the considered prototyping platform, as it is shown in Table 6.6 and Table 6.9. Therefore, it can be expected that successful implementations of high performance integrated transform and quantization circuits based on the envisioned processing structure can also be obtained using Xilinx Virtex-7 XC7VX485T-2FFG1761C FPGA devices.

For the processing in real time of video sequences with even higher resolutions or frame rates, such as the 8k UHDTV format (7680×4320 @ 30 fps), the results depicted in Figure 6.6 instruct the use of clock frequencies higher than 300 MHz. In fact, the data presented in Table 6.13 suggests that the decoding of a whole MB using the $Intra_{16\times16}$ inverse transform and quantization path (i.e. the processing path that requires the highest amount of clock cycles to process a MB) demands a clock frequency of about 470 MHz. Nowadays, such relatively high clock frequency values can be easily attained by using ASIC implementations based on modern CMOS processes, which justifies the use of the proposed transform and quantization architectures in the development of ITQAs also for ultra high performance video coding systems.

Table 6.13: Amount of CCs that are required to encode one MB using the considered integrated transform and quantization circuit.

Processing path	N	B	TE			QE		$CPMB_{TE}$		$CPMB_{QE}$		Total
			L_{TE}	T_{TE}	P_{TE}	L_{QE}	T_{QE}	Luma	Chroma	Luma	Chroma	$CPMB$
Default forward /inverse transform	4	16	3	8	2	4	4	70	42	0	0	112
Default forward / inverse quantization	4	16	3	8	2	4	4	0	0	68	38	106
Default forward /inverse transform and quantization	4	16	3	8	2	4	4	70	42	68	38	112
$Intra_{16\times16}$ forward transform and quantization	4	16	3	8	2	4	4	78	42	68	38	120
$Intra_{16\times16}$ inverse transform and quantization	4	16	3	8	2	4	4	82	42	68	38	124
8×8 forward / inverse transform	8	8	7	8	1	4	4	78	0	0	0	78
8×8 forward / inverse quantization	8	8	7	8	1	4	4	0	42	68	38	106
8×8 forward / inverse transform and quantization	8	8	7	8	1	4	4	78	42	68	38	120

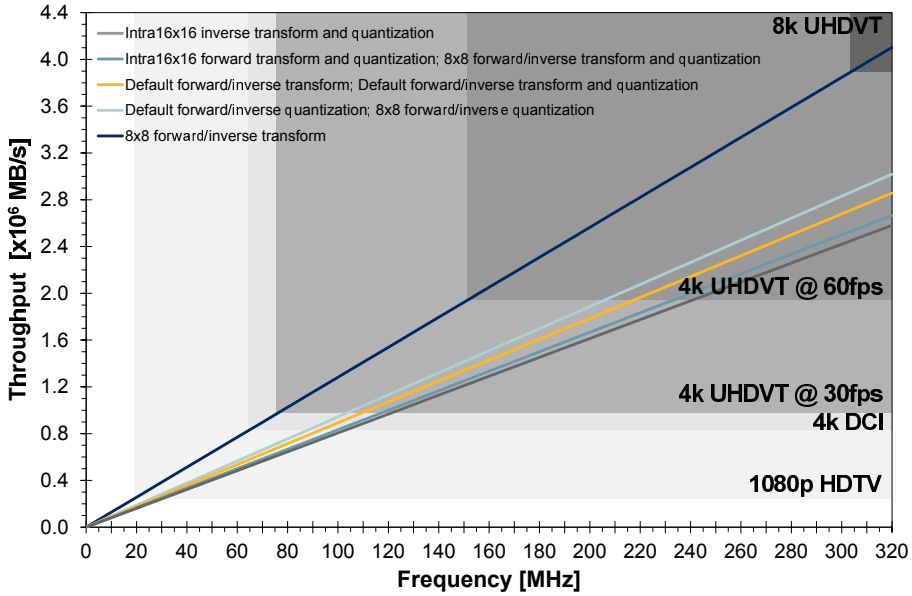


Figure 6.6: Theoretical performance assessment of the considered integrated transform and quantization circuit.

6.5 Summary

In this chapter, the proposed multi-transform and quantization architectures are evaluated based on experimental results that were attained with the implementation of several different proof of concept transform cores and quantizers compliant with the complete H.264/AVC standard in a Xilinx Virtex-7 XC7VX485T-2FFG1761C FPGA device.

The collected data shows that the proposed MTA can be successfully used to realize transform cores with a reduced hardware cost (i.e. less than 7% of the hardware resources available in the considered medium size FPGA device) and that are capable of computing almost 18 GOPS and sustaining processing throughputs of about 2.2×10^9 S/s, by using a clock frequency of 280.6 MHz. Such high performance levels are competitive regarding the state of the art and allow the processing of the whole set of H.264/AVC transforms in real time for video sequences with resolutions up to 3840×2160 pixels and with a frame rate of 30 fps (i.e. the 4k UHDTV video format).

The obtained results also suggest that the hardware requirements of the proposed MTA scale well with the configuration adopted for the TA, which accounts for about $\frac{2}{3}$ of the total hardware resources of the implemented transform cores albeit the quite reduced cost of the devised PE (i.e. about 44 Virtex-7 slices). This is a very important aspect, since it enables to effectively trade-off performance for hardware cost and thus to design processing structures suitable for both low cost and high performance video coding systems. Furthermore, the presented data shows that the coarse-grain data-level parallel processing capability offered by the proposed MTA not only allows to improve the hardware efficiency of a given transform core up to 50% but also to speed up the

transform computation procedure up to 4 times, with a negligible impact (less than 0.3%) in the clock frequency of the implemented 8×8 PEs transform cores.

In what concerns the proposed class of quantization architectures, the obtained experimental results show that it can be used to design high performance forward, inverse and unified quantizers capable of processing up to 464×10^6 S/s, and thus support the real time processing of video sequences in the 4k UHDTV format. Moreover, they also show that the hardware requirements of such quantizers not only are quite diminished (i.e. they require between 135 and 301 Virtex-7 slices and a single `DSP48E1` slice) but also scale well with the amount of implemented pipeline stages, suggesting that several different quantization circuits offering distinct performance *vs* hardware cost trade-offs can be obtained by using the proposed architectures. As mentioned above, this is a very important aspect because it allows the presented class of quantization architectures to be used in a much wider range of video coding systems, ranging from low-end systems to high performance codecs. The collected data further suggests that the proposed class of quantization architectures is competitive regarding the state of the art in all these application domains, not only owing to its improved hardware efficiency but also to the possibility to easily combine several instances of the same quantization circuit in a single quantizer, in order to obtain highly parallel processing structures capable of processing even higher resolution video formats in real time.

The obtained experimental results also suggest that the proposed multi-transform and quantization architectures can be successfully used to design high performance and autonomous H.264/AVC integrated transform and quantization circuits, highly suitable for the realization of the transform coding modules that mandatoroly integrate all video coding systems.

7

Conclusions

Contents

7.1 Thesis summary and conclusions 150
7.2 Future research directions 152
7.3 List of publications 153

This chapter presents an overview of the research work that was performed in the scope of this PhD thesis and summarizes the most important conclusions and original contributions of such investigation. In addition, it briefly discusses the most relevant open research issues and points out some directions for future work. Finally, it provides the list of all the scientific articles that were published in scientific journals and conference proceedings presenting the most relevant contributions and findings of the performed research work.

7.1 Thesis summary and conclusions

In the last years, the H.264/AVC standard emerged as one of the most important and widely used digital video standards, both for non-conversational and conversational services. This is mostly a result of the greatly increased compression performance that this standard is capable of offering. However, such improved coding efficiency comes at the expense of higher computational complexity, data processing rates and memory bandwidth requirements. These constraints pose several difficult challenges in the design of video encoding and decoding systems, especially when real time operation and the processing of HD contents is demanded, or computing systems with limited computational, storage and energy resources are considered (e.g. mobile and portable devices). In practice, such requirements can only be met by using highly specialized hardware structures to accelerate the realization of the most critical and time consuming operations of the video codecs.

Accordingly, this PhD thesis addresses the problem of efficiently computing the H.264/AVC mandatory transform and quantization procedures by exploiting the use of dedicated hardware structures in video coding systems implemented using VLSI and reconfigurable technologies. In particular, this thesis presents a novel high performance and scalable multi-transform architecture for the computation of all the transforms defined in the H.264/AVC standard. In addition, a new class of high performance architectures with reduced hardware cost is also presented for the realization of H.264/AVC quantizers. Furthermore, an integrated transform and quantization circuit based on these multi-transform and quantization architectures is also herein proposed to optimize the joint computation of the H.264/AVC transform and quantization procedures.

Regarding the computation of the H.264/AVC transforms, the key conclusion of this thesis is that systolic array processors can be successfully used to develop high performance, hardware efficient and scalable transform cores. In particular, this thesis demonstrates that by jointly exploiting the use of the row-column decomposition algorithm and 2-D systolic arrays it is possible to obtain highly parallel data-stream driven processing structures that are capable of efficiently computing all the DCTs defined in the H.264/AVC standard, i.e. the forward and inverse 8×8 and 4×4 DCTs and the forward and inverse 4×4 and 2×2 Hadamard transforms. Furthermore, it is herein shown that the modular and regular pipelined design style of systolic arrays can be effi-

ciently exploited in several different ways, in order to obtain hardware structures that are capable of supporting various transforms with distinct sizes and kernels.

More specifically, this thesis thoroughly discusses how to efficiently map the operations required by a given set of transforms into single- and multi-transform PE architectures. On top of that, it presents an innovative methodology to design highly specialized and parameterizable PEs for the computation of any given DCT, which exploits the use of mux-MCM structures to implement fast multipliers with a reduced hardware cost. Due to all the considered techniques and optimization procedures, the resulting PEs can be operated using high clock frequencies, thus offering enhanced computational and data processing rates.

In another direction, this thesis also shows how to exploit the modularity, regularity and pipelinability properties of systolic arrays, in order to efficiently adapt the hardware structure of a systolic transform core to the specific requirements of any given video coding system or application. In this scope, the presented work introduces three distinct techniques that can be used to adjust these transform architectures either to match the size of the considered transform kernels, to scale them according to the performance and hardware cost requirements of the target implementation, or implement coarse-grain data-level parallelism to enable the simultaneous computation of several different transforms.

All these conclusions are supported by multiple experimental results concerning the implementation and evaluation of several different H.264/AVC transform cores in a Xilinx Virtex-7 FPGA device. In fact, the obtained results not only prove that systolic arrays can be used to design transform cores capable of supporting the real time processing requirements of the 4k UHDTV format (3840×2160 pixels @ 30 fps) with a reduced hardware cost but also demonstrate the advantages offered by all the proposed techniques to effectively manage the balance between the performance and the hardware requirements of such transform cores.

In what concerns the realization of the H.264/AVC forward and inverse quantization schemes, two major conclusions can be drawn from the performed research work. On the one hand, this thesis demonstrates that both schemes can be represented by using a single and more generic formulation, since they follow very similar procedures involving operands with comparable characteristics and the same combination of operations. Moreover, it also shows that such alternative definition offers important advantages in the development of highly specialized architectures for the realization of the forward and inverse quantization procedures, as well as enables the design of resource-shared unified architectures that can be used to efficiently implement both procedures with a more reduced hardware cost.

On the other hand, this thesis shows that the non-linear quantization functions that are used by the H.264/AVC forward and inverse quantization schemes can be realized by using not only integer arithmetic but also relatively simple combinational circuits, comprising only three computational modules, i.e. one integer multiplier, one integer adder and one shifter. Based on

these observations, two dedicated architectures are herein proposed for the computation of the H.264/AVC forward and inverse quantization procedures and one very efficient resource-shared unified quantization architecture supporting the realization of both procedures is also presented. These high performance architectures exhibit low hardware cost and extensively exploit pipeline parallelism techniques, in order to attain the high computational and data processing rates that are required for the processing in real time of the HD and ultra-HD video formats. Furthermore, they present a very flexible and configurable hardware structure that allows them to be customized, in order to provide implementations offering different trade-offs between performance and hardware cost, thus making them highly suitable for multiple application domains.

The experimental results that were obtained with the implementation of several different forward, inverse and unified H.264/AVC quantization cores in a Xilinx Virtex-7 FPGA device show that such circuits can meet the real time processing requirements of the 4k UHDTV format (4096×2160 pixels @ 30 fps) when operated with a minimum clock frequency of 374 MHz.

With regard to the joint optimization of the H.264/AVC transform and quantization procedures, this thesis shows that the proposed multi-transform and quantization architectures can also be successfully combined to design ITQAs compliant with the H.264/AVC standard. Furthermore, it presents a high performance and hardware efficient ITQA that is capable of autonomously implementing all the hierarchical transform and quantization paths defined in the H.264/AVC standard, i.e. *i*) the computation of the forward transform and quantization procedures in cascade, employed in the encoding of video sequences; *ii*) the computation of the inverse quantization and transform procedures in cascade, required to decode the video bit streams; and *iii*) the complete forward transform, forward quantization, inverse quantization and inverse transform loop, so that a single hardware circuit can be used in the realization of H.264/AVC video codecs.

7.2 Future research directions

Considering the objectives of this PhD research plan, and by taking into account the work that was already developed and that is herein reported, several different topics can still be identified as open research questions. Consequently, the following tasks can be defined as highly relevant directions for future work:

- Implementation of the ITQA presented in section 6.4 as a configurable IP core supporting the autonomous computation of the H.264/AVC transform and quantization procedures. This processing structure should be described by using a parametrizable circuit description, which can be written in VHDL or in another alternative hardware description language (e.g. Verilog), and by considering a generic coding style, so that efficient hardware realizations can be obtained not only when using distinct technologies (i.e. FPGA and ASIC) but also different implementation processes.

- Investigation of efficient techniques to enable the dynamic reconfiguration of the proposed MTA, not only to implement TAs with distinct sizes but also to make use of different PEs. In reconfigurable platforms, this feature can be used to dynamically adjust the requirements (in terms of performance and hardware cost) and the functionality of the instantiated transform cores to the constraints and requisites of the considered video coding application in run time. Therefore, it should provide the means required to obtain a more efficient management of the hardware resources available in the FPGA, as well as of the usable energy stored in the batteries of mobile and portable video coding systems.

- Increase the versatility and functionality of the H.265/HEVC and multi-standard PEs presented in appendix A, so that they are also able to support the computation of the forward and inverse 4×4 DST defined in the H.265/HEVC standard.

- Extend the use of the devised scalable MTA to the MPEG-2, MJPEG an JPEG standards. This research should involve a comprehensive study of the forward and inverse 8×8 real DCT adopted by these video standards, in order to allow its computation using integer arithmetic with the required precision levels, as well as the definition of a new PE for the efficient computation of all the required operations. In addition, the results of such study could also be used to develop a new unified architecture for the PEs, suitable for the realization of multi-standard transform cores supporting the H.264/AVC, VC-1, AVS, H.265/HEVC, MPEG-2, MJEPG and JPEG standards.

- Enhance the processing capabilities of the proposed quantization architectures, by increasing the amount of coefficients that can be computed at each clock cycle. In particular, data-level parallelism techniques should be investigated and exploited to support the replication of the datapath in an efficient manner, so that several different coefficients can be simultaneously computed by using the same control data.

- Adaptation and extension of the proposed class of quantization architectures to support the quantization procedures defined in other video standards, such as VC-1, AVS, H.265/HEVC and MPEG-2. In this scope, alternative hardware structures based on the MCM paradigm should be investigated and special attention should be devoted to the H.265/HEVC standard, since at the time of writing this thesis it is emerging as the newest state-of-the-art video standard.

7.3 List of publications

In the realization of this PhD thesis, a special attention was given to the validation and the dissemination of the developed hardware structures and techniques within the related scientific

community. Accordingly, the most relevant topics, results and conclusions of the conducted research work resulted in the publication of three articles in international scientific journals and ten articles in the proceedings of several international (6) and national (4) scientific conferences. The following is a list of these publications, which are presented in reverse chronological order.

- International Scientific Journals:

  [37] T. Dias, N. Roma, and L. Sousa. Unified transform architecture for AVC, AVS, VC-1 and HEVC high-performance codecs. *EURASIP Journal on Advances in Signal Processing*, 2014(108), July 2014.

  [26] T. Dias, S. López, N. Roma, and L. Sousa. Scalable unified transform architecture for advanced video coding embedded systems. *International Journal of Parallel Programming*, 41(2):236–260, Apr. 2013;

  [24] T. Dias, S. López, N. Roma, and L. Sousa. A flexible architecture for the computation of direct and inverse transforms in H.264/AVC video codecs. *IEEE Transactions on Consumer Electronics*, 57(2):936–944, May 2011.

- Proceedings of International Scientific Conferences:

  [38] T. Dias, N. Roma, and L. Sousa. High performance quantization architectures for HEVC transform coding IP cores. In *IEEE International Symposium on Circuits and Systems (ISCAS 2015)*, May 2015. (To appear);

  [36] T. Dias, N. Roma, and L. Sousa. High performance multi-standard architecture for DCT computation in H.264/AVC high profile and HEVC codecs. In *Conference on Design & Architectures for Signal and Image Processing (DASIP 2013)*, pages 14–21, Oct. 2013. Best Paper Award;

  [40] T. Dias, L. Rosário, N. Roma, and L. Sousa. High performance unified architecture for forward and inverse quantization in H.264/AVC. In *15th Euromicro Conference on Digital System Design (DSD 2012)*, pages 632–639, Sept. 2012;

  [25] T. Dias, S. López, N. Roma, and L. Sousa. High throughput and scalable architecture for unified transform coding in embedded H.264/AVC video coding systems. In *International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS 2011)*, pages 225–232, July 2011. Best Paper Award;

  [31] T. Dias, N. Roma, and L. Sousa. H.264/AVC framework for multi-core embedded video encoders. In *International Symposium on System-on-Chip (SOC 2010)*, pages 89–92, Sept. 2010;

  [32] T. Dias, N. Roma, and L. Sousa. Hardware/software co-design of H.264/AVC encoders for multi-core embedded systems. In *International Conference on Design and*

Architectures for Signal and Image Processing (DASIP 2010), pages 231–238, Oct. 2010. Best Poster Award;

[122] N. Sebastião, T. Dias, N. Roma, and P. Flores. Integrated accelerator architecture for DNA sequences alignment with enhanced traceback phase. In *International Conference on High Performance Computing & Simulation (HPCS 2010)*, pages 16–23, June 2010.

- Proceedings of National Scientific Conferences:

[35] T. Dias, N. Roma, and L. Sousa. Exploiting coarse-grained parallelism in multi-transform architectures for H.264/AVC high profile codecs. In *Conference on Electronics, Telecommunications and Computers (CETC 2013)*, pages CD–ROM, Oct. 2013. ISBN: 978-989-97531-3-6;

[34] T. Dias, N. Roma, and L. Sousa. Reconfigurable unified architecture for forward and inverse quantization in H.264/AVC. In *VIII Jornadas sobre Sistemas Reconfiguráveis (REC 2012)*, pages 75–82, Feb. 2012;

[33] T. Dias, N. Roma, and L. Sousa. Optimized forward/inverse quantization unit for H.264/AVC codecs. In *Conference on Electronics, Telecommunications and Computers (CETC 2011)*, pages CD–ROM, Nov. 2011. ISBN: 978-989-97531-0-5;

[23] T. Dias, S. López, N. Roma, and L. Sousa. Efficient and programmable processing unit for H.264/AVC systolic unified transform engines. In *VII Jornadas sobre Sistemas Reconfiguráveis (REC 2011)*, pages 13–19, Feb. 2011.

In addition, the performed research work was also distinguished with the following three awards:

- *Best Paper Award* in the 2013 Conference on Design and Architectures for Signal and Image Processing (DASIP 2013), which was held in Cagliari, Italy, in October 2013.
- *The "Stamatis Vassiliadis" Best Paper Award* in the 11th International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS XI), which was held in the Greek island of Samos in July 2011;
- *Best Poster Award* in the 2010 Conference on Design and Architectures for Signal and Image Processing (DASIP 2010), which was held in Edinburgh, United Kingdom, in October 2010.

Bibliography

[1] L. Agostini, R. Porto, M. Porto, T. Silva, L. Rosa, J. Guntzel, I. Silva, and S. Bampi. Forward and inverse 2-D DCT architectures targeting HDTV for H.264/AVC video compression standard. *Latin American applied research*, 37:11–16, Jan. 2007.

[2] Altera Corporation. *Arria V Device Overview*, 2013.12.26 edition, Dec. 2013.

[3] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image coding using wavelet transform. *IEEE Transactions on Image Processing*, 1(2):205–220, Apr. 1992.

[4] Y. Arai, T. Agui, and M. Nakajima. A fast DCT-SQ scheme for images. *IEICE Transactions*, E71(11):1095–1097, 1988.

[5] R. B. Are and K. Rajan. An RNS based transform architecture for H.264/AVC. In *2008 IEEE Region 10 Conference TENCON*, pages 1–6, 2008.

[6] F. Bergeron, J. Berstel, and S. Brlek. Efficient computation of addition chains. *Journal de théorie des nombres de Bordeaux*, 6(1):21–38, 1994.

[7] V. Bhaskaran and K. Konstantinides. *Image and Video Compression Standards: Algorithms and Architectures*. Kluwer Academic Publishers, 2nd edition, June 1997.

[8] V. Britanak, P. Yip, and K. R. Rao. *Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations*. Academic Press, 1st edition, Sept. 2006.

[9] K. Bukhari, G. Kuzmanov, and S. Vassiliadis. DCT and IDCT implementations on different FPGA technologies. In *13th Annual Workshop on Circuits, Systems and Signal Processing*, Nov. 2002.

[10] W.-K. Cham. Development of integer cosine transforms by the principle of dyadic symmetry. *Communications, Speech and Vision, IEE Proceedings I*, 136(4):276–282, 1989.

[11] H. Chang, S. Kim, S. Lee, and K. Cho. Design of area-efficient unified transform circuit for multi-standard video decoder. In *International SoC Design Conference*, pages 369–372, 2009.

[12] K.-H. Chen, J.-I. Guo, and J.-S. Wang. An efficient direct 2-d transform coding IP design for MPEG-4 AVC/H.264. In *2005 IEEE International Symposium on Circuits and Systems*, volume 5, pages 4517–4520, 2005.

[13] K.-H. Chen, J.-I. Guo, and J.-S. Wang. A high-performance direct 2-D transform coding IP design for MPEG-4AVC/H.264. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(4):472–483, 2006.

[14] L.-G. Chen, J.-Y. Jiu, H.-C. Chang, Y.-P. Lee, and C.-W. Ku. A low power 2D DCT chip design using direct 2D algorithm. In *1998 Asia and South Pacific Design Automation Conference*, pages 145–150, 1998.

[15] W.-H. Chen, C. Smith, and S. Fralick. A fast computational algorithm for the discrete cosine transform. *Communications, IEEE Transactions on*, 25(9):1004–1009, 1977.

[16] Y. Chen, X. Cao, Q. Xie, and C. Peng. An area efficient high performance DCT distributed architecture for video compression. In *9th International Conference on Advanced Communication Technology*, volume 1, pages 238–241, Feb. 2007.

[17] Y.-H. Chen, J.-N. Chen, T.-Y. Chang, and C.-W. Lu. High-throughput multistandard transform core supporting MPEG/H.264/VC-1 using common sharing distributed arithmetic. *IEEE Transactions on Very Large Scale Integration Systems*, PP(99):1–1, 2013.

[18] Y. Cheng, Z. Wang, J. Guo, and K. Dai. Research on intra modes for inter-frame coding in H.264. In *Proc. of the 9th Int. Conf. on Computer Supported Cooperative Work in Design*, pages 740–744. IEEE, May 2005.

[19] Z.-Y. Cheng, C.-H. Chen, B.-D. Liu, and J.-F. Yang. High throughput 2-D transform architectures for H.264 advanced video coders. In *2004 IEEE Asia-Pacific Conference on Circuits and Systems*, volume 2, pages 1141–1144, 2004.

[20] C. R. Clare. *Designing Logic Systems Using State Machines*. McGraw-Hill, June 1973.

[21] I. Corporation. Intel® Atom™ processor E6x5C series-based platform for embedded computing. In *Platform Brief: Intel Atom Processor E6x5C Series*. Intel Corporation, 2010.

[22] K. DeHaven. Extensible processing platform - ideal solution for a wide range of embedded systems. In *White Paper: Extensible Processing Platform*. Xilinx, Inc., 2010.

[23] T. Dias, S. López, N. Roma, and L. Sousa. Efficient and programmable processing unit for H.264/AVC systolic unified transform engines. In *VII Jornadas sobre Sistemas Reconfiguráveis (REC 2011)*, pages 13–19, Feb. 2011.

[24] T. Dias, S. López, N. Roma, and L. Sousa. A flexible architecture for the computation of direct and inverse transforms in H.264/AVC video codecs. *IEEE Transactions on Consumer Electronics*, 57(2):936–944, May 2011.

[25] T. Dias, S. López, N. Roma, and L. Sousa. High throughput and scalable architecture for unified transform coding in embedded H.264/AVC video coding systems. In *International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS 2011)*, pages 225–232, July 2011. Best Paper Award.

[26] T. Dias, S. López, N. Roma, and L. Sousa. Scalable unified transform architecture for advanced video coding embedded systems. *International Journal of Parallel Programming*, 41(2):236–260, Apr. 2013.

[27] T. Dias, S. Momcilovic, N. Roma, and L. Sousa. Adaptive motion estimation processor for autonomous video devices. *EURASIP Journal on Embedded Systems – Special Issue on Embedded Systems for Portable and Mobile Video Platforms*, 2007(57234):1–10, May 2007.

[28] T. Dias, N. Roma, and L. Sousa. Customizable and reduced hardware motion estimation processors. In P. Lysaght and W. Rosenstiel, editors, *New Algorithms, Architectures, and Applications for Reconfigurable Computing*. Springer-Verlag, 2005.

[29] T. Dias, N. Roma, and L. Sousa. Efficient motion vector refinement architecture for sub-pixel motion estimation systems. In *IEEE Workshop on Signal Processing Systems*, pages 313–318, Nov. 2005.

[30] T. Dias, N. Roma, and L. Sousa. Low power distance measurement unit for real-time hardware motion estimators. In *International Workshop on Power and Timing Modeling, Optimization and Simulation*, volume 4148, pages 247–255, Sept. 2006.

[31] T. Dias, N. Roma, and L. Sousa. H.264/AVC framework for multi-core embedded video encoders. In *International Symposium on System-on-Chip (SOC 2010)*, pages 89–92, Sept. 2010.

[32] T. Dias, N. Roma, and L. Sousa. Hardware/software co-design of H.264/AVC encoders for multi-core embedded systems. In *International Conference on Design and Architectures for Signal and Image Processing (DASIP 2010)*, pages 231–238, Oct. 2010. Best Poster Award.

[33] T. Dias, N. Roma, and L. Sousa. Optimized forward/inverse quantization unit for H.264/AVC codecs. In *Conference on Electronics, Telecommunications and Computers (CETC 2011)*, pages CD–ROM, Nov. 2011. ISBN: 978-989-97531-0-5.

[34] T. Dias, N. Roma, and L. Sousa. Reconfigurable unified architecture for forward and inverse quantization in H.264/AVC. In *VIII Jornadas sobre Sistemas Reconfiguráveis (REC 2012)*, pages 75–82, Feb. 2012.

[35] T. Dias, N. Roma, and L. Sousa. Exploiting coarse-grained parallelism in multi-transform architectures for H.264/AVC high profile codecs. In *Conference on Electronics, Telecommunications and Computers (CETC 2013)*, pages CD–ROM, Oct. 2013. ISBN: 978-989-97531-3-6.

[36] T. Dias, N. Roma, and L. Sousa. High performance multi-standard architecture for DCT computation in H.264/AVC high profile and HEVC codecs. In *Conference on Design & Architectures for Signal and Image Processing (DASIP 2013)*, pages 14–21, Oct. 2013. Best Paper Award.

[37] T. Dias, N. Roma, and L. Sousa. Unified transform architecture for AVC, AVS, VC-1 and HEVC high-performance codecs. *EURASIP Journal on Advances in Signal Processing*, 2014(108), July 2014.

[38] T. Dias, N. Roma, and L. Sousa. High performance quantization architectures for HEVC transform coding IP cores. In *IEEE International Symposium on Circuits and Systems (ISCAS 2015)*, May 2015. (To appear).

[39] T. Dias, N. Roma, L. Sousa, and M. Ribeiro. Reconfigurable architectures and processors for real-time video motion estimation. *Journal of Real-Time Image Processing – Special issue on Field-Programmable Technology*, 2(4):191–205, Dec. 2007.

[40] T. Dias, L. Rosário, N. Roma, and L. Sousa. High performance unified architecture for forward and inverse quantization in H.264/AVC. In *15th Euromicro Conference on Digital System Design (DSD 2012)*, pages 632–639, Sept. 2012.

[41] T. Do and T. Le. High throughput area-efficient SoC-based forward/inverse integer transforms for H.264/AVC. In *2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 4113–4116, June 2010.

[42] P. Duhamel and C. Guillemot. Polynomial transform computation of the 2-D DCT. In *1990 International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages 1515–1518, 1990.

[43] M. Elhaji, A. Zitouni, S. Meftali, J. Dekeyser, and R. Tourki. A low power and highly parallel implementation of the H.264 8x8 transform and quantization. In *2010 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, pages 528–531, 2010.

[44] C.-P. Fan. Cost-effective hardware sharing architectures of fast 8x8 and 4x4 integer transforms for H.264/AVC. In *2006 IEEE Asia Pacific Conf. on Circuits and Syst. (APCCAS'06)*, pages 776–779, Dec. 2006.

[45] C.-P. Fan. Fast 2-dimensional 4x4 forward integer transform implementation for H.264/AVC. *IEEE Transactions on Circuits and Systems—Part II: Analog and Digital Signal Processing*, 53(3):174–177, Mar. 2006.

[46] W.-H. Fang and M.-L. Wu. An efficient unified systolic architecture for the computation of discrete trigonometric transforms. In *IEEE International Symposium on Circuits and Systems*, volume 3, pages 2092–2095, 1997.

[47] W. Gao, S. Ma, L. Zhang, L. Su, and D. Zhao. AVS video coding standard. In C. Chen, Z. Li, and S. Lian, editors, *Intelligent Multimedia Communication: Techniques and Applications*, volume 280 of *Studies in Computational Intelligence*, pages 125–166. Springer Berlin Heidelberg, 2010.

[48] V. K. Goyal. Theoretical foundations of transform coding. *IEEE Signal Processing Magazine*, 18(5):9–21, 2001.

[49] M. Gua, N. Yu, C. Jiang, and W. Lu. Hardware prototyping for various transforms in H.264 high profile. *Journal of Information and Computational Science*, 8(1):119–128, 2011.

[50] T. Ho, T. Le, K. Vu, S. Mochizuki, K. Iwata, K. Matsumoto, and H. Ueda. A 768 megapixels/sec inverse transform with hybrid architecture for multi-standard decoder. In *IEEE 9th Int. Conf. on ASIC (ASICON)*, pages 71–74, Oct. 2011.

[51] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University, June 1994.

[52] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro. H.264/AVC baseline profile decoder complexity analysis. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):704–716, 2003.

[53] H. S. Hou. A fast recursive algorithm for computing the discrete cosine transform. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 35(10):1455–1461, 1987.

[54] S.-C. Hsia, C.-F. Tsai, S.-H. Wang, and K.-C. Hung. Transposed-memory free implementation for cost-effective 2D-DCT processor. *Journal of Signal Processing Systems*, 58(2):161–172, 2010.

[55] J. Huang and J. Lee. A self-reconfigurable platform for scalable DCT computation using compressed partial bitstreams and blockRAM prefetching. *IEEE Transactions on Circuits and Systems for Video Technology*, 19(11):1623–1632, 2009.

[56] J. Huang, M. Parris, J. Lee, and R. F. Demara. Scalable FPGA-based architecture for DCT computation using dynamic partial reconfiguration. *ACM Transactions on Embedded Computer Systems*, 9(1):1–18, Oct. 2009.

[57] R. Husemann, M. Majolo, V. Guimaraes, A. Susin, V. Roesler, and J. V. Lima. Hardware integrated quantization solution for improvement of computational H.264 encoder module. In *18th IEEE/IFIP VLSI System on Chip Conference*, pages 316–321, 2010.

[58] R. Husemann, M. Majolo, A. Susin, V. Roesler, and J. Lima. Highly efficient transforms module solution for a H.264/SVC encoder. In *2010 IEEE Computer Society Annual Symp. on VLSI*, pages 86–91, July 2010.

[59] W. Hwangbo, J. Kim, and C.-M. Kyung. A high-performance 2-D inverse transform architecture for the H.264/AVC decoder. In *IEEE International Symposium on Circuits and Systems*, pages 1613–1616, 2007.

[60] W. Hwangbo and C.-M. Kyung. A multitransform architecture for H.264/AVC High-Profile coders. *IEEE Transactions on Multimedia*, 12(3):157–167, 2010.

[61] ISO/IEC. *ISO/IEC JTC1 CD 11172 'Coding of moving pictures and associated audio for digital storage media up to 1.5 Mbit/s'*. ISO/IEC, 1992.

[62] ISO/IEC. *ISO/IEC 14 496-2 'Coding of Audio-Visual Objects - Part 2: Visual', Version 1*. ISO/IEC, Apr. 1999. ISO/IEC 14 496-2 (MPEG-4 Visual Version 1).

[63] ISO/IEC and ITU-T. *JM H.264/AVC Reference Software - version 18.4*, Aug. 2012. http://iphome.hhi.de/suehring/tml.

[64] ISO/IEC JTC1/SC29/WG11 N3908. *MPEG-4 Video Verification Model - version 18.0*, Jan. 2001.

[65] ITU-R. *Recommendation ITU-R BT.601-7 'Studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios*. ITU-R, 2011. ITU-R Recommendation BT.601 Version 7.

[66] ITU-T. *ITU-T Recommendation H.261 'Video Codec for Audiovisual Services at px64 kbit/s', Version 1*. ITU-T, 1990. ITU-T Recommendation H.261 Version 1.

[67] ITU-T. *ITU-T Recommendation H.263 'Video Coding for Low Bit Rate Communication', Version 1*. ITU-T, 1995. ITU-T Recommendation H.263 version 1.

[68] ITU-T and ISO/IEC. *ITU-T Recommendation H.262 and ISO/IEC 13 818-2 (MPEG-2) 'Generic Coding of Moving Pictures and Associated Audio Information - Part 2: Video'*. ITU-T and ISO/IEC JTC 1, 1994. ITU-T Recommendation H.262 and ISO/IEC 13 818-2 (MPEG-2).

[69] ITU-T and ISO/IEC. *ITU-T Recommendation H.264 and ISO/IEC 14496-10 'Advanced video coding for generic audiovisual services'*. ITU-T and ISO/IEC JVT, Apr. 2013.

[70] ITU-T and ISO/IEC. *ITU-T Recommendation H.265 and ISO/IEC 23008-2 'High efficiency coding and media delivery in heterogeneous environments – Part 2: High efficiency video coding'*. ITU-T and ISO/IEC JVT, Apr. 2013.

[71] H. Jeong, J. Kim, and W. k. Cho. Low-power multiplierless DCT architecture using image correlation. *IEEE Transactions on Consumer Electronics*, 50(1):262–267, Feb. 2004.

[72] B. L. Jian, Z. Xuan, T. J. Rong, and L. Yue. An efficient VLSI architecture for 2D-DCT using direct method. In *4th International Conference on ASIC*, pages 393–396, 2001.

[73] C. Jiang, N. Yu, and M. Gu. A novel VLSI architecture of 8x8 integer DCT based on H.264/AVC FRext. In *3rd International Symposium on Knowledge Acquisition and Modeling (KAM)*, pages 59–62, Oct. 2010.

[74] K. T. Johnson, A. Hurson, and B. Shirazi. General-purpose systolic arrays. *IEEE Computer*, 26(11):20–31, 1993.

[75] H. B. Kekre and J. K. Solanki. Comparative performance of various trigonometric unitary transforms for transform image coding. *International Journal of Electronics*, 44(3):305–315, 1978.

[76] T. Komarek and P. Pirsch. Array architectures for block matching algorithms. *IEEE Transactions on Circuits and Systems*, 36(10):1301–1308, Oct. 1989.

[77] R. Korah and J. R. Perinbam. FPGA implementation of integer transform and quantizer for H.264 encoder. *Journal of Signal Processing Systems*, 53(3):261–269, 2008.

[78] R. Kordasiewicz and S. Shirani. Hardware implementation of the optimized transform and quantization blocks of H.264. In *2004 Canadian Conf. Elect. Comput. Eng.*, volume 2, pages 943–946, May 2004.

[79] R. C. Kordasiewicz and S. Shirani. ASIC and FPGA implementations of H.264 DCT and quantization blocks. In *2005 IEEE International Conference on Image Processing*, volume 3, pages III–1020–3, 2005.

[80] S. Y. Kung. *VLSI Array Processors*. Prentice Hall, 1988.

[81] Y.-K. Lai and Y.-F. Lai. A reconfigurable IDCT architecture for universal video decoders. *IEEE Transactions on Consumer Electronics*, 56(3):1872–1879, 2010.

[82] B. Lee. A new algorithm to compute the discrete cosine transform. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 32(6):1243–1245, 1984.

[83] J. Lee, S. Yang, S. Park, I. Heo, and Y. Paek. VLIW processor for H.264: Integer transform and quantization. In *International SoC Design Conference*, pages 178–181, 2010.

[84] J.-J. Lee, S. Park, and N. Eum. Design of application specific processor for H.264 inverse transform and quantization. In *International SoC Design Conference*, volume 2, pages 57–60, 2008.

[85] M. H. Lee and Y. Yasuda. Simple systolic array algorithm for Hadamard transform. *Electronics Letters*, 26(18):1478–1480, Aug. 1990.

[86] S. Lee and K. Cho. Implementation of an AMBA-compliant IP for H.264 transform and quantization. In *2006 IEEE Asia Pacific Conference on Circuits and Systems*, pages 1071–1074, 2006.

[87] S. Lee and K. Cho. Design of high-performance transform and quantization circuit for unified video CODEC. In *2008 IEEE Asia Pacific Conference on Circuits and Systems*, pages 1450–1453, 2008.

[88] Y.-P. Lee, T.-H. Chen, L.-G. Chen, M.-J. Chen, and C.-W. Ku. A cost-effective architecture for 8x8 two-dimensional DCT/IDCT using direct method. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(3):459–467, 1997.

[89] L. LG Display Co. Press release – LG display brings innovation in smart mobile panels to SID 2011. `http://lgdnewsroom.com/press_releases/601`, May 2011.

[90] Y. Li, Y. He, and S. Mei. A highly parallel joint VLSI architecture for transforms in H.264/AVC. *Journal of Signal Processing Systems*, 50(1):19–32, 2008.

[91] Z.-N. Li and M. S. Drew. *Fundamentals of Multimedia*. Prentice Hall, 1st edition, Nov. 2003.

[92] J. Liang and T. Tran. Fast multiplierless approximations of the DCT with the lifting scheme. *IEEE Transactions on Signal Processing*, 49(12):3032–3044, 2001.

[93] C.-C. Lin, J.-W. Chen, H.-C. Chang, Y.-C. Yang, Y.-H. Yang, M.-C. Tsai, J.-I. Guo, and J.-S. Wang. A 160k gates/4.5kb SRAM H.264 video decoder for HDTV applications. *IEEE Journal of Solid-State Circuits*, 42(1):170–182, Jan. 2007.

[94] H.-Y. Lin, Y.-C. Chao, C.-H. Chen, B.-D. Liu, and J.-F. Yang. Combined 2-D transform and quantization architectures for H.264 video coders. In *2005 IEEE International Symposium on Circuits and Systems*, volume 2, pages 1802–1805, 2005.

[95] Y.-L. Lin, C.-Y. Kao, J.-W. Chen, and H.-C. Kuo. *VLSI Design for Video Coding: H.264/AVC Encoding from Standard Specification to Chip*. Springer, Feb. 2010.

[96] L. Ling-Zhi, Q. Lin, R. Meng-tian, and J. Li. A 2-D forward/inverse integer transform processor of H.264 based on highly-parallel architecture. In *Proc. 4th IEEE Int. Workshop on System-on-Chip for Real-Time Applications, 2004*, pages 158–161, July 2004.

[97] G. Liu. An area-efficient IDCT architecture for multiple video standards. In *2nd Int. Conf. on Information Science and Engineering (ICISE)*, pages 3518–3522, Dec. 2010.

[98] C.-C. Lo, S.-T. Tsai, and M.-D. Shieh. Reconfigurable architecture for entropy decoding and inverse transform in H.264. *IEEE Transactions on Consumer Electronics*, 56(3):1670–1676, Aug. 2010.

[99] C. Loeffler, A. Ligtenberg, Moschytz, and S. George. Practical fast 1-D DCT algorithms with 11 multiplications. In *Acoustics, Speech, and Signal Processing, 1989. ICASSP-89., 1989 International Conference on*, volume 2, pages 988–991, 1989.

[100] Z. Ma, H. Hu, and Y. Wang. On complexity modeling of H.264/AVC video decoding and its application for energy efficient decoding. *IEEE Transactions on Multimedia*, 13(6):1240–1255, 2011.

[101] H. S. Malvar. Fast computation of the discrete cosine transform and the discrete hartley transform. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 35(10):1484–1485, 1987.

[102] H. S. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky. Low-complexity transform and quantization in H.264/AVC. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):598–603, 2003.

[103] D. Marpe, T. Wiegand, and S. Gordon. H.264/MPEG4-AVC fidelity range extensions: tools, profiles, performance, and application areas. In *Proc. of Int. Conf. on Image Processing (ICIP 2005)*, pages 593–596. IEEE, Sept. 2005.

[104] M. Martuza, C. McCrosky, and K. Wahid. A fast hybrid DCT architecture supporting H.264, VC-1, MPEG-2, AVS and JPEG codecs. In *11th International Conference on Information Science, Signal Processing and their Applications*, pages 545–549, 2012.

[105] M. Martuza and K. Wahid. A cost effective implementation of 8x8 transform of HEVC from H.264/AVC. In *25th IEEE Canadian Conference on Electrical Computer Engineering*, pages 1–4, 2012.

[106] Mentor Graphics Corporation. *ModelSim SE User's Manual*, Nov. 2010.

[107] M. N. Michael and K. W. Hsu. A low-power design of quantization for H.264 video coding standard. In *2008 IEEE International SOC Conference*, pages 201–204, 2008.

[108] Y. Moshe and N. Peleg. Implementations of H.264/AVC baseline decoder on different digital signal processors. In *47th International Symposium ELMAR*, pages 37–40, 2005.

[109] N. T. Ngo, T. T. T. Do, T. M. Le, Y. S. Kadam, and A. Bermak. ASIP-controlled inverse integer transform for H.264/AVC compression. In *The 19th IEEE/IFIP International Symposium on Rapid System Prototyping*, pages 158–164, 2008.

[110] Y. Ooi. Motion estimation system design. In K. K. Parhi and T. Nishitani, editors, *Digital Signal Processing for Multimedia Systems*, chapter 12, pages 299–327. Marcel Dekker, Inc, 1999.

[111] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi. Video coding with H.264/AVC: tools, performance and complexity. *IEEE Circuits and Systems Magazine*, 4(1):7–28, Jan. 2004.

[112] M. Owaida, M. Koziri, I. Katsavounidis, and G. Stamoulis. A high performance and low power hardware architecture for the transform & quantization stages in H.264. In *2009 IEEE International Conference on Multimedia and Expo*, pages 1102–1105, 2009.

[113] S. B. Pan and R.-H. Park. Unified systolic arrays for computation of the DCT/DST/DHT. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(2):413–419, 1997.

[114] W. Pan, A. Shams, and M. A. Bayoumi. NEDA: a new distributed arithmetic architecture and its application to one dimensional discrete cosine transform. In *1999 IEEE Workshop on Signal Processing Systems*, pages 159–168, 1999.

[115] G. Pastuszak. Transforms and quantization in the high-throughput H.264/AVC encoder based on advanced mode selection. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI '08*, pages 203–208, 2008.

[116] A. Peled and B. Liu. A new hardware realization of digital filters. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 22(6):456–462, 1974.

[117] C. Peng, D. Yu, X. Cao, and S. Sheng. A new high throughput VLSI architecture for H.264 transform and quantization. In *7th International Conference on ASIC*, pages 950–953, 2007.

[118] A. Pradini, T. M. Roffi, R. Dirza, and T. Adiono. VLSI design of a high-throughput discrete cosine transform for image compression systems. In *2011 International Conference on Electrical Engineering and Informatics*, pages 1–6, 2011.

[119] K. Rao and P. Yip. *Discrete Cosine Transform: Algorithms, Advantages, Applications*. Academic Press, 1990.

[120] I. E. Richardson. *The H.264 Advanced Video Compression Standard*. John Wiley & Sons, Ltd, 2nd edition, 2010.

[121] H. Schwarz, D. Marpe, and T. Wiegand. Overview of the scalable video coding extension of the H.264/AVC standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1103–1120, Sept. 2007.

[122] N. Sebastião, T. Dias, N. Roma, and P. Flores. Integrated accelerator architecture for DNA sequences alignment with enhanced traceback phase. In *International Conference on High Performance Computing & Simulation (HPCS 2010)*, pages 16–23, June 2010.

[123] N. Sebastião, T. Dias, N. Roma, P. Flores, and L. Sousa. Application specific programmable IP core for motion estimation: technology comparison targeting efficient embedded co-processing units. In *11th EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools*, pages 181–188, Sept. 2008.

[124] M. Shafique, L. Bauer, and J. Henkel. Optimizing the H.264/AVC video encoder application structure for reconfigurable and application-specific platforms. *Journal of Signal Processing Systems*, 60(2):183–210, 2010.

[125] A. Shams and M. Bayoumi. A 108 Gbps, 1.5 GHz 1D-DCT architecture. In *IEEE Int. Conf. on Application-Specific Systems, Architectures, and Processors*, pages 163–172, 2000.

[126] J. Shan, C. Chen, and E. Yang. High performance 2-D IDCT for image/video decoding based on FPGA. In *International Conference on Audio, Language and Image Processing*, pages 33–38, 2012.

[127] S. Shen, W. Shen, Y. Fan, and X. Zeng. A unified 4/8/16/32-point integer IDCT architecture for multiple video coding standards. In *IEEE International Conference on Multimedia and Expo*, pages 788–793, 2012.

[128] S. Srinivasan and S. L. Regunathan. An overview of VC-1. In S. Li, F. Pereira, H.-Y. Shum, and A. G. Tescher, editors, *Society of Photo-Optical Instrumentation Engineers (SPIE) Conf. Series*, volume 5960 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conf. Series*, pages 720–728, July 2005.

[129] G. J. Sullivan, J. Ohm, H. Woo-Jin, and T. Wiegand. Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1649–1668, 2012.

[130] C.-C. Sun, P. Donner, and J. Götze. VLSI implementation of a configurable IP core for quantized discrete cosine and integer transforms. *International Journal of Circuit Theory Applications*, 40(11):1107–1126, Nov. 2012.

[131] X.-T. Tran and V.-H. Tran. An efficient architecture of forward transforms and quantization for H.264/AVC codecs. *Journal on Electronics and Communications*, 1(2):122–129, Aug. 2011.

[132] A. Tumeo, M. Monchiero, G. Palermo, F. Ferrandi, and D. Sciuto. A pipelined fast 2D-DCT accelerator for FPGA-based SoCs. In *IEEE Computer Society Annual Symposium on VLSI*, pages 331–336, 2007.

[133] P. Tummeltshammer, J. C. Hoe, and M. Püschel. Time-multiplexed multiple-constant multiplication. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(9):1551–1563, Sept. 2007.

[134] K. Ugur, K. Andersson, A. Fuldseth, G. Bjontegaard, L. Endresen, J. Lainema, A. Hallapuro, J. Ridge, D. Rusanovskyy, C. Zhang, A. Norkin, C. Priddle, T. Rusert, J. Samuelsson, R. Sjoberg, and Z. Wu. High performance, low complexity video coding and the emerging HEVC standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 20(12):1688–1697, Dec. 2010.

[135] S. Venkataraman, V. R. Kanchan, K. R. Rao, and M. Mohanty. Discrete transforms via the walsh-hadamard transform. *Signal Processing*, 14(4):371–382, 1988.

[136] M. Vetterli and H. J. Nussbaumer. Simple FFT and DCT algorithms with reduced number of operations. *Signal Processing*, 6(4):267–278, 1984.

[137] Y. Voronenko and M. Püschel. Multiplierless multiple constant multiplication. *ACM Transactions on Algorithms*, 3(2), May 2007.

[138] K. Wahid, M. Martuza, M. Das, and C. McCrosky. Resource shared architecture of multiple transforms for multiple video codecs. In *24th Canadian Conf. on Electrical and Computer Engineering (CCECE)*, pages 947–950, May 2011.

[139] K. Wang, J. Chen, W. Cao, Y. Wang, L. Wang, and J. Tong. A reconfigurable multi-transform VLSI architecture supporting video codec design. *IEEE Transactions on Circuits and Systems—Part II: Analog and Digital Signal Processing*, 58(7):432–436, July 2011.

[140] C. Wei, H. Hui, L. Jinmei, T. Jiarong, and M. Hao. A high-performance reconfigurable 2-D transform architecture for H.264. In *15th IEEE Int. Conf. Electronics, Circuits and Syst.*, pages 606–609, Sept. 2008.

[141] M. Weizhen. A novel systolic array implementation of DCT, DWT and DFT. In *IEEE Region 10 Conference on Computer and Communication Systems*, volume 1, pages 211–215, 1990.

[142] T. Wiegand and H. Schwarz. *Source Coding: Part I of Fundamentals of Source and Video Coding*. Now Publishers, Jan. 2011.

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

[144] Xilinx, Inc. *Xilinx ISE 10.1 Software Manuals*, Dec. 2008.

[145] Xilinx, Inc. *Xilinx ISE 13.2 Software Manuals*, Mar. 2011.

[146] Xilinx, Inc. *7 Series FPGAs Overview (Data Sheet)*, Nov. 2012.

[147] Xilinx, Inc. *7 Series DSP48E1 Slice User Guide (UG479)*, 1.7 edition, May 2014.

[148] Xilinx, Inc. *VC707 Evaluation Board for the Virtex-7 FPGA User Guide*, Sept. 2014.

[149] J. Ying, X. Chen, Y. Fan, and X. Zeng. MUX-MCM based quantization vlsi architecture for H.264/AVC high profile encoder. In *2011 IEEE/IFIP 19th International Conference on VLSI and System-on-Chip (VLSI-SoC)*, pages 72–77, 2011.

[150] L. Yu, S. Chen, and J. Wang. Overview of AVS-video coding standards. *Image Commun.*, 24(4):247–262, Apr. 2009.

[151] C. Zhang, L. Yu, J. Lou, W.-K. Cham, and J. Dong. The technique of prescaled integer transform: Concept, design and applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(1):84–97, Jan. 2008.

[152] Q. Zhang and N. Meng. A low area pipelined 2-D DCT architecture for JPEG encoder. In *52nd IEEE International Midwest Symposium on Circuits and Systems*, pages 747–750, Aug. 2009.

[153] Y. Zhang, G. Jiang, and M. Yu. Low-complexity quantization for H.264/AVC. *Journal of Real-Time Image Processing*, 4(1):3–12, 2009.

[154] S. Zohar. New hardware realizations of nonrecursive digital filters. *IEEE Transactions on Computers*, C-22(4):328–338, 1973.

[155] H. K. Zrida, A. C. Ammari, M. Abid, and A. Jemai. Complexity/performance analysis of a H.264/AVC video encoder. In D. J. D. S. Lorente, editor, *Recent Advances on Video Coding*. InTech, 2011.

A

Transform cores for other relevant digital video standards

Contents

A.1 Overview of the VC-1, AVS and H.265/HEVC transform coding procedures . . 172
A.2 PE architectures for the VC-1, AVS and H.265/HEVC standards 175
A.3 Resource-shared architecture of the multi-standard PE 182

The MTA presented in chapter 4 was designed to fully address the data processing requirements of the modern video codecs supporting the H.264/AVC standard. Nevertheless, it can also be used to realize high performance and hardware efficient dedicated transform cores addressing other state-of-the-art video standards, as well as to implement MST cores supporting multiple video standards. Such alternative processing structures can be easily obtained by successfully exploiting the scaling capabilities and the modular nature of the proposed MTA, as explained in section 4.4.

In this appendix, the modularity of the proposed MTA is exploited, in order to extend its functionality to also support the other current state-of-the-art video standards, i.e. VC-1[128], Audio Video coding Standard (AVS)[150] and H.265/HEVC[134]. Accordingly, three new PEs targeting the VC-1, AVS and H.265/High Efficiency Video Coding (HEVC) standards are first introduced in section A.2. Then, a resource-shared multi-standard PE suitable for the processing of the four video standards in MST cores is presented in section A.3.

A.1 Overview of the VC-1, AVS and H.265/HEVC transform coding procedures

A.1.1 VC-1

One of the main goals of the VC-1 standard[128], which was initially developed by the Microsoft Corporation as WMV-9 and later standardized by the Society of Motion Picture and Television Engineers (SMTPE), is to provide efficient codec implementations for online video services. This type of video services usually requires low complexity codecs that must be capable of producing low bit rate video streams. To comply with both requirements, VC-1 also implements a hybrid motion compensated and transform coding scheme. However, the complexity of this coding procedure is much lower than the one adopted in the H.264/AVC standard.

The VC-1 transform process is similar to the one adopted by the H.264/AVC standard in the sense that it also makes use of variable-size transforms, in order to better exploit the spatial correlation of the data and to improve the coding efficiency. However, VC-1 only considers a single transform level and two distinct kernels to process all the possible block sizes, i.e. 4×4, 8×4, 4×8, and 8×8. The 8×8 transform is used to encode the Intra MBs. Conversely, the Inter MBs can be encoded using any of the four available separable transforms, provided that the optional variable-size transform mode has been enabled in the encoder[128]. Table A.1 enumerates the transform coefficient values of such 8×8 and 4×4 kernels, whose generic representations are presented in Equation 4.5 and Equation 4.6, respectively.

The VC-1 transform process also allows fast algorithm implementations to compute the inverse transforms. These transforms, which do not involve any rescaling operations, can be computed using 16-bits arithmetic and without any multiplications. In contrast, the computation of the forward

Table A.1: Coefficients of the VC-1 transform kernels.

Transform	Kernel Values						
	a	b	c	d	e	f	g
4×4 DCT	17	-	-	-	-	22	10
8×8 DCT	12	16	15	9	4	16	9

transforms is slightly more complex, due to requiring an extra normalization stage to compensate the different norms of its basis functions. Similarly to the H.264/AVC standard, such scaling factors are also absorbed by the quantization procedure[128].

A.1.2 AVS

The AVS[150] standard was developed in China and adopts a coding scheme analogous to H.264/AVC, which is why the two standards offer quite similar coding performances. Nevertheless, contrary to what happens with the H.264/AVC standard, the transform process of AVS considers the computation of a single 2-D transform for the processing of all the luma and chroma blocks composing the 16×16 pixels MBs.

In AVS Part 2 (a.k.a. AVS 1.0), which targets high definition digital video broadcasting and high density storage media, a MB is composed of four 8×8 luma blocks and of two chroma blocks with 8×8 samples each. All these blocks are processed by an 8×8 integer transform with the generic transform kernel shown in Equation 4.5. This transform was designed in conjunction with the quantization process by using the Pre-Scaled Integer Transform (PIT) technique[151], in order to reduce the rounding errors and minimize the complexity of the decoder implementation. As a result, it consists of a separable and integer precise 2-D transform that can be computed using 16-bits arithmetic, thanks to the reduced magnitude of the involved kernel values, as it can be seen in Table A.2. However, rounding and rescaling operations must be applied after the computation of the row and column transforms, not only to guarantee the accuracy of the 16-bits operation but also to harmonize the transform/quantization and entropy coding procedures.

The alternative Part 7 of AVS[47], commonly known as AVS-M, was proposed for video communication applications targeting mobile devices. Typically, these devices deal with lower resolution videos and possess limited processing capabilities and memory resources. To better adjust the complexity of the video codec to these characteristics, AVS Part 7 defines the 4×4 block size as the basic unit for the transform process. Accordingly, it specifies a 2-D separable transform for the processing of all the luma and chroma blocks, whose generic kernel is shown in Equation 4.6.

Table A.2: Coefficients of the AVS transform kernels.

Transform	Kernel Values						
	a	b	c	d	e	f	g
4×4 DCTs	2	-	-	-	-	3	1
8×8 DCTs	8	10	9	6	2	10	4

Due to the considered kernel values (enumerated in Table A.2), such 4×4 integer transform can also be computed using 16-bits arithmetic and by using only integer additions and shift operations.

A.1.3 H.265/HEVC

In H.265/HEVC[134], the basic units defined for the transform and quantization processes are called Transform Units (TUs). The TUs consist of square blocks of data that can be recursively subdivided in four equally sized blocks, starting from the 32×32 samples TU format and going all the way down to a minimum of 4×4 samples. As a result of this segmentation, several different integer transforms with multiple sizes are specified for the H.265/HEVC transform process. Nevertheless, all the transform kernels are better approximations of the DCT than those that were adopted by the H.264/AVC standard.

The H.265/HEVC transform kernels were defined by approximating scaled DCT basis functions under specific considerations, such as limiting the necessary dynamic range for transform computation or maximizing the precision and closeness to orthogonality, whenever the kernel entries are specified as integer values. Consequently, the basis vectors of these transforms have equal energy and there is no need to compensate for the different norms, as in previous video standards. Furthermore, such property also allows using the same kernels to compute both the forward and the inverse transforms. However, due to the increased dynamic range of the involved transform kernels, H.265/HEVC explicitly inserts rescaling and 16-bits clipping operations after the row-wise transform stage. This guarantees that all the transforms can be computed using 16-bits integer arithmetic.

To reduce the complexity of the encoder and to simplify the computation of the transforms, only one order-32 transform kernel is specified in the H.265/HEVC standard (see[134]). The remaining lower order kernels consist of sub-sampled versions of this kernel. The entries of such order-k kernels (with $k = 4, 8, 16$) consist of the first k values of rows $j \times \frac{32}{k}$ (with $j = 0...k-1$) of the 32×32 kernel. All the lower order kernels also present key symmetry properties, to enable fast "partially-factored" implementations using very few mathematical operations. As an example of this feature, the H.265/HEVC 8×8 and 4×4 transform kernels can be obtained from Equation 4.5 and Equation 4.6, respectively, by considering the corresponding coefficients enumerated in Table A.3.

Table A.3: Coefficients of the H.265/HEVC transform kernels.

Transform	Kernel Values						
	a	b	c	d	e	f	g
4×4 DCT	64	-	-	-	-	83	36
8×8 DCT	64	89	75	50	18	83	36

A.2 PE architectures for the VC-1, AVS and H.265/HEVC standards

The PEs that were developed to support the computation of all the transforms defined in the VC-1, AVS and H.265/HEVC standards are also based on the generic hardware structure introduced in subsection 4.2.1 and whose block diagram is depicted in Figure 4.5. Consequently, they were derived by exactly following the same methodology that was adopted to realize the H.264/AVC PE (see subsection 4.2.2), which can be summarized in the following eight design steps:

Step 1: Define the values of the `Type_T` signal, by properly encoding the set of transforms to be supported;

Step 2: Evaluate and select all the distinct positive kernel values that are required for the computation of the considered set of transforms, i.e. the basis values of all the involved transform kernels;

Step 3: Define a DAG representing an optimal addition chain for each one of the kernel values that are defined;

Step 4: Find and exploit the similarities in all the devised graphs, in order to obtain the best composite DAG jointly representing the addition chains of all the individual DAGs. If possible, such graph should consist exclusively of additions, subtractions, shifts and multiplexers;

Step 5: Develop a fast and hardware efficient hardware structure to implement the composite DAG, which should be exclusively composed of adders, subtractors and multiplexers;

Step 6: Define a control word to command the operation of all the multiplexers and addition/subtraction circuits composing the hardware structure obtained in the previous design step for each of the considered kernel values;

Step 7: Properly dispose all the multiplier control words in distinct memory segments of the ROM that is embedded in the control module of the PE, according to the encoding of the `Type_T` signal. Within each memory segment, N consecutive memory positions must be occupied with the multiplier control words corresponding to the N basis values of each of the considered order-N transform kernels. In addition, this data must be disposed in the same manner as its corresponding kernel values in the first column of the transform kernel matrix.

In the following subsections, this methodology is briefly reviewed for the design of each of the considered PEs.

A.2.1 PE for the VC-1 standard

Step 1: The 8 transforms defined in the VC-1 standard were encoded using 3 bits, as shown in Table A.4.

Table A.4: Encoding of the `Type_T` signal for the implementation of the VC-1 PE.

`Type_T`	Transform
0	Forward 4×4 DCT
1	Forward 4×8 DCT
2	Forward 8×4 DCT
3	Forward 8×8 DCT
4	Inverse 4×4 DCT
5	Inverse 4×8 DCT
6	Inverse 8×4 DCT
7	Inverse 8×8 DCT

Step 2: The values of the basis functions of all the considered transform kernels are listed in Table A.1.

Step 3/4: The composite DAG representing all the possible addition chains is shown in Figure A.1.

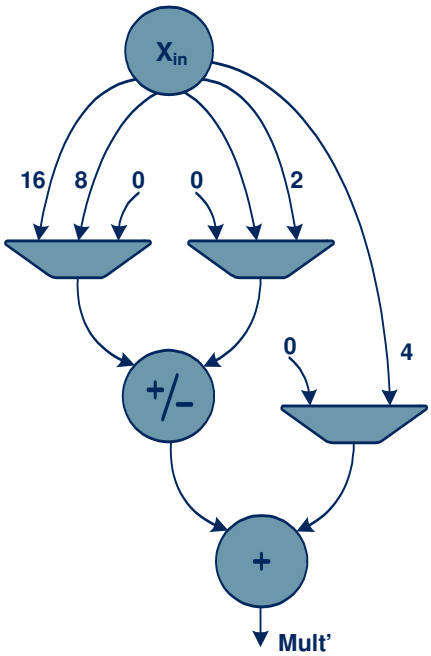


Figure A.1: DAG of the mux-MCM used in the arithmetic module of the VC-1 PE.

Step 5: Figure A.2 shows the hardware structure that was devised to implement the DAG presented in Figure A.1, which is composed of 1 addition and 1 addition/subtraction circuits, 2 multiplexers and a couple of `AND` gates. The multiplexers are used to realize the specialized barrel shifters involved in the computation of all the required multiplications, while the `AND` gates are used to set all the required zero values.

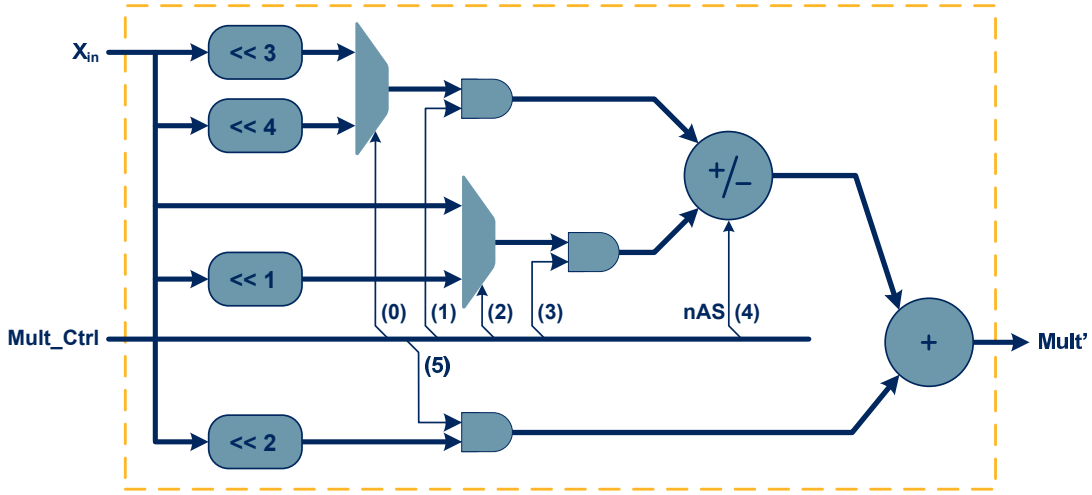


Figure A.2: Architecture of the mux-MCM used in the arithmetic module of the VC-1 PE.

Step 6: The 8 control words that are used to define the operation of the architecture presented in Figure A.2 are listed in Table A.5. The numbers shown in Figure A.2 identify the position of the bits controlling each of the architecture's circuits within the control word.

Table A.5: Multiplier control words for the VC-1 transform kernels (see Equation 4.5, Equation 4.6 and Table A.1).

Kernel Value	4	9	10	12	15	16	17	22
ROM Word	$0x20$	$0xA$	$0xE$	$0x22$	$0x1B$	$0x3$	$0xB$	$0x1F$

Step 7: The 8 distinct multiplier control words listed in Table A.5 were properly disposed throughout two separate memory segments, as shown in Figure A.3. The segment comprehending the memory positions 0-7 holds the values of the 8×8 transforms basis functions, which are also used in the computation of the 8×4 and 4×8 transforms. The other segment occupies the memory positions 8-13 and holds the values of the basis functions for the 4×4 transforms. The two most significant bits of the `Type_T` signal are used to select the desired memory segment.

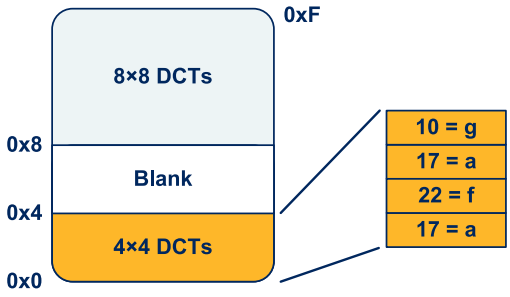


Figure A.3: Memory map of the ROM used in the VC-1 PE.

A.2.2 PE for the AVS standard

Step 1: Step 1: The 4 transforms defined in the AVS standard were encoded using 2 bits, as shown in Table A.6.

Table A.6: Encoding of the `Type_T` signal for the implementation of the AVS PE.

`Type_T`	Transform
0	Forward 4×4 DCT
1	Forward 8×8 DCT
2	Inverse 4×4 DCT
3	Inverse 8×8 DCT

Step 2: The values of the basis functions of all the considered transform kernels are listed in Table A.2.

Step 3/4: The composite DAG representing all the possible addition chains is shown in Figure A.4.

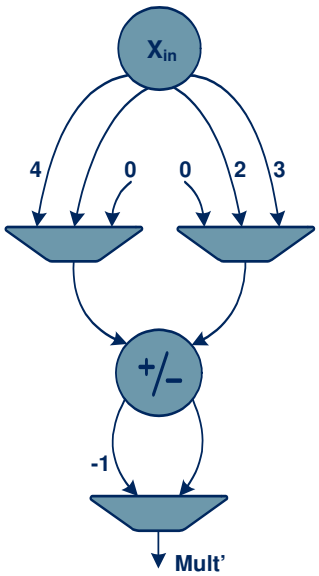


Figure A.4: DAG of the mux-MCM used in the arithmetic module of the AVS PE.

Step 5: Figure A.5 shows the hardware structure that was devised to implement the DAG presented in Figure A.4, which is composed of 1 addition/subtraction circuit, 2 multiplexers and a couple of `AND` gates. The multiplexers are used to realize the specialized barrel shifters involved in the computation of all the required multiplications, while the `AND` gates are used to set all the required zero values.

Step 6: The 8 control words that are used to define the operation of the architecture presented in Figure A.5 are listed in Table A.7. The numbers shown in Figure A.5 identify the position of the bits controlling each of the architecture's circuits within the control word.

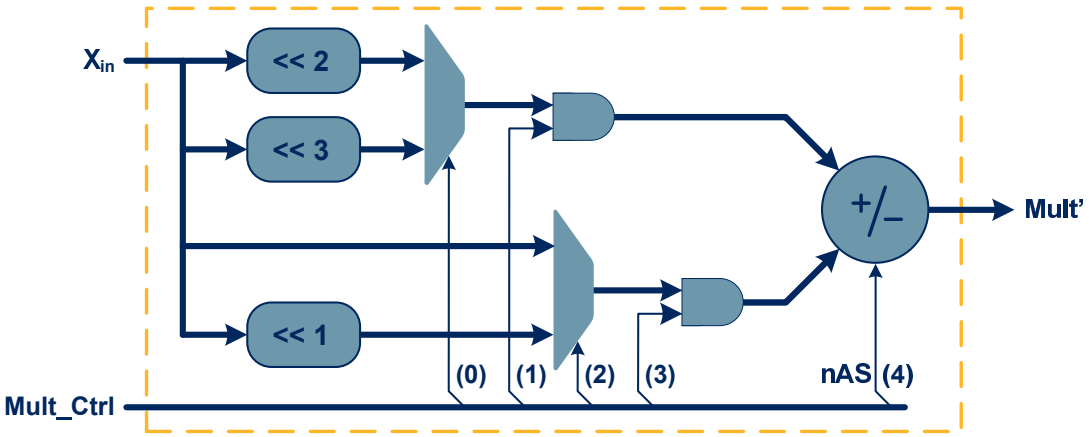


Figure A.5: Architecture of the mux-MCM used in the arithmetic module of the AVS PE.

Table A.7: Multiplier control words for the AVS transform kernels (see Equation 4.5, Equation 4.6 and Table A.2).

Kernel Value	1	2	3	4	6	8	9	10
ROM Word	$0x8$	$0xC$	$0x1A$	$0x2$	$0xE$	$0x3$	$0xB$	$0xF$

Step 7: The 8 distinct multiplier control words listed in Table A.7 were properly disposed throughout two separate memory segments, as shown in Figure A.6. The segment comprehending the memory positions 0-7 holds the values of the 8×8 transforms basis functions, while the segment occupying the memory positions 8-13 holds the values of the basis functions for the 4×4 transforms. The most significant bit of the `Type_T` signal is used to select the desired memory segment.

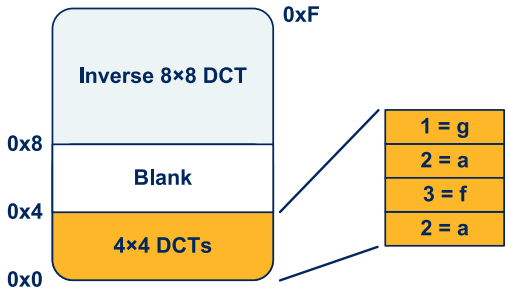


Figure A.6: Memory map of the ROM used in the AVS PE.

A.2.3 PE for the H.265/HEVC standard

Step 1: The 8 DCTs defined in the H.265/HEVC standard were encoded using 3 bits, as shown in Table A.8.

Step 2: The values of the basis functions of all the considered transform kernels are presented in[134] and partially listed in Table A.3 for the 8×8 and 4×4 kernels.

Table A.8: Encoding of the `Type_T` signal for the implementation of the H.265/HEVC PE.

`Type_T`	Transform
0	Forward 4×4 DCT
1	Forward 8×8 DCT
2	Forward 16×16 DCT
3	Forward 32×32 DCT
4	Inverse 4×4 DCT
5	Inverse 8×8 DCT
6	Inverse 16×16 DCT
7	Inverse 32×32 DCT

Step 3/4: The composite DAG representing all the possible addition chains is shown in Figure A.7.

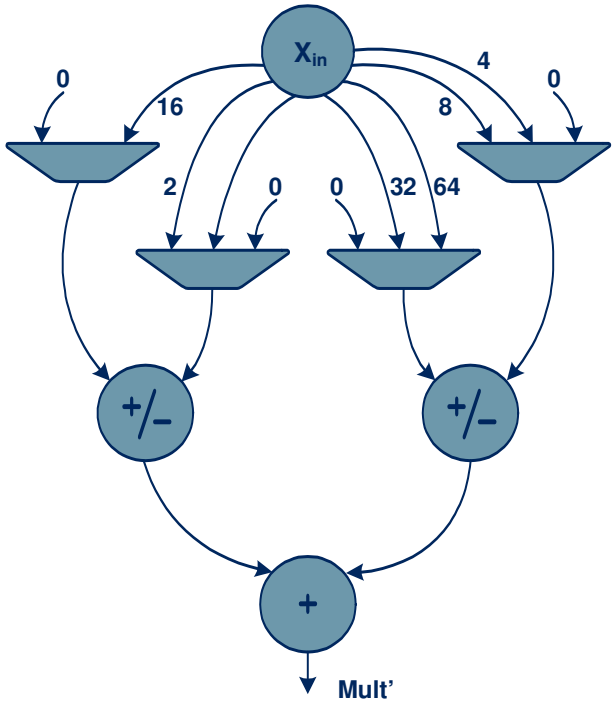


Figure A.7: DAG of the mux-MCM used in the arithmetic module of the H.265/HEVC PE.

Step 5: Figure A.8 shows the hardware structure that was devised to implement the DAG presented in Figure A.7, which is composed of 1 addition and 2 addition/subtraction circuits, 3 multiplexers and some `AND` gates. The multiplexers are used to realize the specialized barrel shifters involved in the computation of all the required multiplications, while the `AND` gates are used to set all the required zero values.

Step 6: The 8 control words that are used to define the operation of the architecture presented in Figure A.8 are listed in Table A.9. The numbers shown in Figure A.8 identify the position of the bits controlling each of the architecture's circuits within the control word.

Step 7: The 29 distinct multiplier control words listed in Table A.9 were properly disposed through-

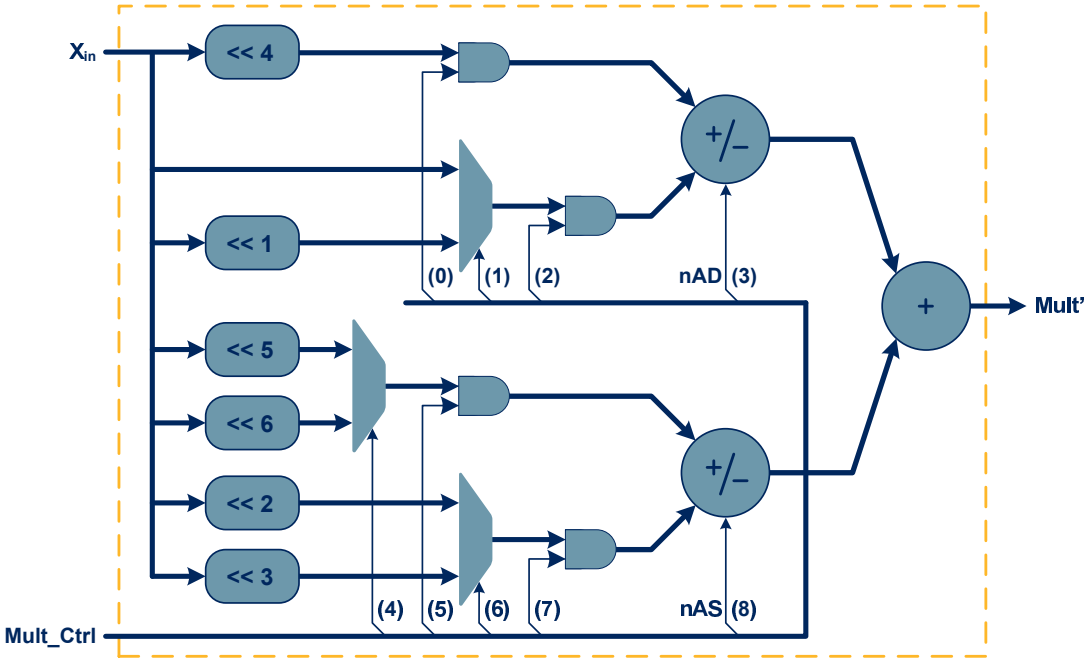


Figure A.8: Architecture of the mux-MCM used in the arithmetic module of the H.265/HEVC PE.

Table A.9: Multiplier control words for the H.265/HEVC DCT kernels (see[134], Equation 4.5, Equation 4.6 and Table A.3).

Kernel Value	4	9	13	18	22	25	31	36	38	43
ROM Word	$0x80$	$0xC4$	$0x185$	$0x7$	$0xCF$	$0xC5$	$0x2C$	$0xA0$	$0xA6$	$0x1AD$
Kernel Value	**46**	**50**	**54**	**57**	**61**	**64**	**67**	**70**	**73**	**75**
ROM Word	$0x2F$	$0x27$	$0xA7$	$0xE5$	$0x1B4$	$0x30$	$0xBC$	$0xB6$	$0xF4$	$0x1BD$
Kernel Value	**78**	**80**	**82**	**83**	**85**	**87**	**88**	**89**	**90**	
ROM Word	$0x3F$	$0x31$	$0x37$	$0xBD$	$0xB5$	$0xFD$	$0xF1$	$0xF5$	$0xF7$	

out a unique memory segment with 32 memory positions, as shown in Figure A.9. This segment holds the values of the basis functions of the 32×32 DCTs, which are also used in the computation of the 16×16, 8×8 and 4×4 DCTs.

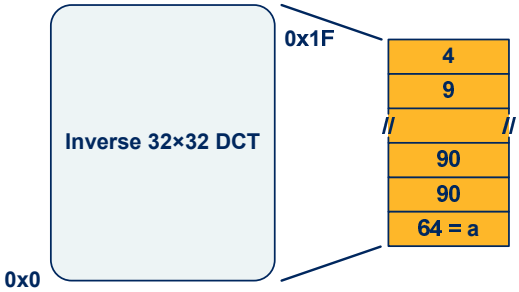


Figure A.9: Memory map of the ROM used in the H.265/HEVC PE.

A.3 Resource-shared architecture of the multi-standard PE

The methodology described in section A.2 was also used to develop a high performance and hardware efficient PE for the computation of all the DCTs defined in the H.264/AVC, VC-1, AVS and H.265/HEVC standards. The devised multi-standard architecture shares all its hardware resources for the computation of the MAC operations required by the considered 28 DCTs, which comprehend 80 distinct transform kernel values in the range from -90 to $+90$. The design steps that were considered to devise this architecture are the following:

Step 1: All the considered DCTs were encoded using 5 bits, as shown in Table A.10.

Table A.10: Encoding of the `Type_T` signal for the implementation of the multi-standard PE.

`Type_T`	Standard	Transform
0	H.264/AVC	Forward 2×2 Hadamard transform
1	H.264/AVC	Forward 4×4 Hadamard transform
2	H.264/AVC	Forward 4×4 DCT
3	H.264/AVC	Forward 8×8 DCT
4	H.264/AVC	Inverse 2×2 Hadamard transform
5	H.264/AVC	Inverse 4×4 Hadamard transform
6	H.264/AVC	Inverse 4×4 DCT
7	H.264/AVC	Inverse 8×8 DCT
8	H.265/HEVC	Forward 4×4 DCT
9	H.265/HEVC	Forward 8×8 DCT
10	H.265/HEVC	Forward 16×16 DCT
11	H.265/HEVC	Forward 32×32 DCT
12	H.265/HEVC	Inverse 4×4 DCT
13	H.265/HEVC	Inverse 8×8 DCT
14	H.265/HEVC	Inverse 16×16 DCT
15	H.265/HEVC	Inverse 32×32 DCT
16	VC-1	Forward 4×4 DCT
17	VC-1	Forward 4×8 DCT
18	VC-1	Forward 8×4 DCT
19	VC-1	Forward 8×8 DCT
20	VC-1	Inverse 4×4 DCT
21	VC-1	Inverse 4×8 DCT
22	VC-1	Inverse 8×4 DCT
23	VC-1	Inverse 8×8 DCT
24	AVS	Forward 4×4 DCT
25	AVS	Forward 8×8 DCT
26	AVS	Inverse 4×4 DCT
27	AVS	Inverse 8×8 DCT

Step 2: The values of all the involved basis functions of all the considered DCTs are presented in Equations 2.12, 2.13, 2.14, 2.15 and 2.17 for the H.264/AVC standard, Table A.1 for the VC-1 standard, Table A.2 for the AVS standard and in Table A.3 and[134] for the H.265/HEVC standard.

Step 3/4: The composite DAG representing all the possible addition chains is shown in Figure A.10.

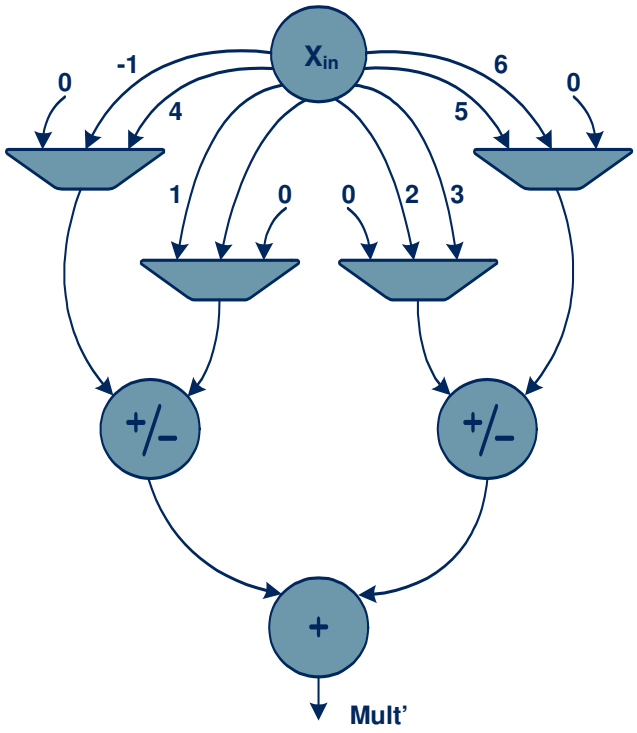


Figure A.10: DAG of the mux-MCM used in the arithmetic module of the multi-standard PE.

Step 5: Figure A.11 shows the hardware structure that was devised to implement the DAG presented in Figure A.10, which is composed of 1 addition and 2 addition/subtraction circuits,

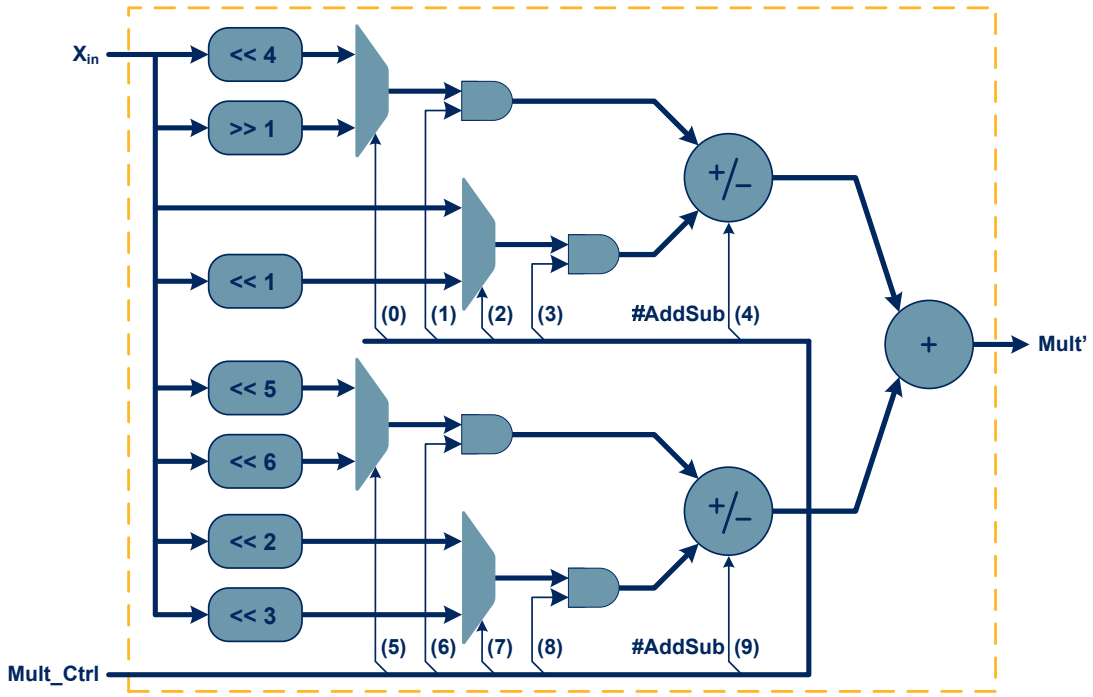


Figure A.11: Architecture of the mux-MCM used in the arithmetic module of the multi-standard PE.

4 multiplexers and some AND gates. The multiplexers are used to realize the specialized barrel shifters involved in the computation of all the required multiplications, while the AND gates are used to set all the required zero values.

Step 6: The control words that are used to define the operation of the architecture presented in Figure A.11 are listed in Table A.11. The numbers shown in Figure A.11 identify the position of the bits controlling each of the architecture's circuits within the control word.

Table A.11: Multiplier control words for the H.264/AVC, VC-1, AVS and H.265/HEVC DCT kernels (see Equations 4.5, 4.6, 2.12, 2.13, 2.14, 2.15 and 2.17, Tables A.3, A.1, A.2 and A.3 and [134]).

Kernel Value	$\frac{1}{2}$	**1**	**2**	**3**	**4**	**6**	**8**	**9**	**10**	**12**
ROM Word	$0x3$	$0x8$	$0xC$	$0x118$	$0x100$	$0x10C$	$0x180$	$0x188$	$0x18C$	$0x302$
Kernel Value	**13**	**15**	**16**	**17**	**18**	**22**	**25**	**31**	**36**	**38**
ROM Word	$0x30A$	$0x1A$	$0x2$	$0xA$	$0xE$	$0x19E$	$0x18A$	$0x58$	$0x140$	$0x15C$
Kernel Value	**43**	**46**	**50**	**54**	**57**	**61**	**64**	**67**	**70**	**73**
ROM Word	$0x35A$	$0x05E$	$0x4E$	$0x14E$	$0x1CA$	$0x368$	$0x60$	$0x178$	$0x16C$	$0x1E8$
Kernel Value	**75**	**78**	**80**	**82**	**83**	**85**	**87**	**88**	**89**	**90**
ROM Word	$0x37A$	$0x7E$	$0x62$	$0x6E$	$0x17A$	$0x16A$	$0x1FA$	$0x1E2$	$0x1EA$	$0x1EE$

Step 7: The 40 distinct multiplier control words listed in Table A.11 were properly disposed throughout eleven memory segments comprising four distinct mega segments, as it is shown in Figure A.12. The first mega segment, which is located between memory positions 0 and 31, comprehends the six memory segments holding the values of the basis functions of the all the H.264/AVC DCTs and Hadamard transforms. The mega segment occupying the memory positions 32-63 consists of the second memory segment, which holds the values of the basis functions of the 32×32 H.265/HEVC DCTs. As mentioned in subsection A.2.3, these values are also used in the computation of the 16×16, 8×8 and 4×4 H.265/HEVC DCTs. The remaining two mega segments store the data required for the computation of all the VC-1 and AVS DCTs. Although each of these mega segments comprehends 32 memory positions, only the segment local addresses in the ranges 0x0-0x3 and 0x8-0xF are used to store the values of the basis functions of the involved 4×4 and 8×8 DCTs, respectively. This approach, which was adopted to reduce the complexity of the address decoder and thus improve the overall performance, does not increases the memory capacity requirements of the resulting ROM module.

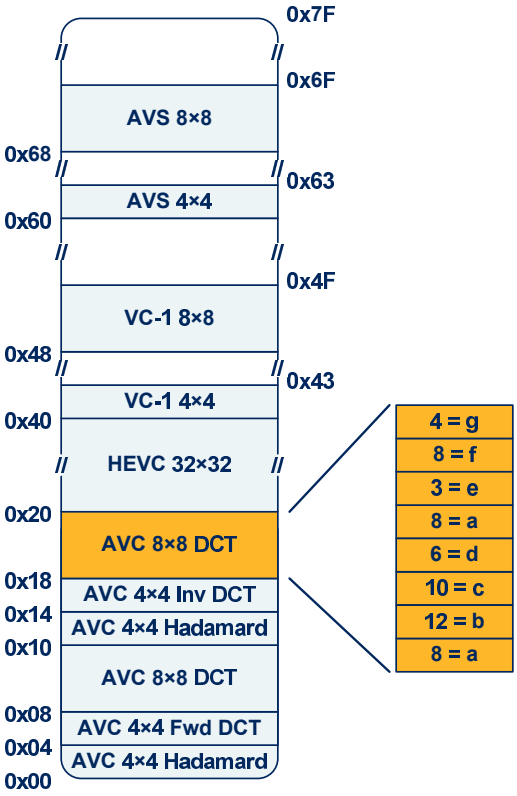


Figure A.12: Memory map of the ROM used in the multi-standard PE.

B

Test Conditions

The functionality of the proof of concept transform cores and quantizers presented in chapter 6 was assessed by using the Modelsim simulator (version 10.0b[106]) and several different test vectors. Such data was obtained with the H.264/AVC JM reference software[63], which was used to encode the first 30 frames of six benchmark standard test video sequences of classes A, B and C of the MPEG-4 Video Verification Model[64], under the following conditions:

- Adoption of the H.264/AVC HiP;
- Group of Pictures (GOP) structure using M=3 and N=14 (IBBPBBPBBPBBPBB);
- 8×8 transform mode enabled;
- Picture level quantization using fixed QP values. For the I- and P-pictures the QP was set to $\{8, 16, 24, 32, 40\}$, while for the B-pictures it was set to its double value.

The considered test video sequences, which were in the uncompressed 4:2:0 YUV Common Intermediate Format (CIF) (352×288 pixels) with a frame rate of 30 fps, present different spatial detail and amount of movement:

- Akiyo (Class A) – characterized by reduced spatial detail and reduced amount of movement. The movement, which is almost static, consists in the local displacements of the head and lips of the person in the scene;
- Carphone (Class C) – characterized by moderate spatial detail and amount of movement. The movement consists in the local displacements of the head and lips of the person in the scene, as well as in the regular translational movements of the background (car window);
- Coastguard (Class B) – characterized by medium spatial detail and medium amount of movement. The spatial detail consist in the spume over the water and the rocks on the shore, while the movement consists in the regular translational displacements of the two boats in the scene;
- Foreman (Class B) – characterized by medium spatial detail and medium amount of movement. The movement consists in the brusque displacements of the head of the person in the scene, as well as in the pan right displacement of the camera promoting the scene change;
- Mother & Daughter (Class A) – characterized by reduced spatial detail and reduced amount of movement. The small mount of movement consists in the local displacements of the head, lips and arms of one of the people in the scene (the mother);
- Table-Tennis (Class C) – characterized by moderate spatial detail and by a significant amount of movement of the translational (ball) and zoom-out types (video camera).

Figure B.1 depicts the the first frame of all the considered test video sequences.

(a) Akiyo.

(b) Carphone.

(c) Coastguard.

(d) Foreman.

(e) Mother & Daughter.

(f) Table-Tennis.

Figure B.1: Considered benchmark standard test video sequences.