UNIVERSIDADE FEDERAL DE PELOTAS
Centro de Desenvolvimento Tecnológico
Programa de Pós-Graduação em Computação

Tese

Heuristic-based Algorithms and Hardware Designs for Fast Intra-picture Prediction in AV1 Video Coding

Marcel Moscarelli Corrêa

Pelotas, 2023.

MARCEL MOSCARELLI CORRÊA

Heuristic-based Algorithms and Hardware Designs for Fast Intra-picture Prediction in AV1 Video Coding

Ph.D. Thesis submitted to the Graduate Program in Computing of the Federal University of Pelotas as partial requirement for the degree of Doctor of Philosophy.

Advisor: Luciano Volcan Agostini, Ph.D.
Co-advisors: Guilherme Ribeiro Corrêa, Ph.D.
Daniel Munari Palomino, Ph.D.

Pelotas, 2023.

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação na Publicação

C823h Corrêa, Marcel Moscarelli

Heuristic-based algorithms and hardware designs for fast intra-picture prediction in AV1 video coding / Marcel Moscarelli Corrêa ; Luciano Volcan Agostini, orientador ; Guilherme Ribeiro Corrêa, Daniel Munari Palomino, coorientadores. — Pelotas, 2023.

126 f. : il.

Tese (Doutorado) — Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2023.

1. Codificação de vídeo. 2. Predição intra. 3. Decisão de modo. 4. AV1. 5. Arquitetura em hardware. I. Agostini, Luciano Volcan, orient. II. Corrêa, Guilherme Ribeiro, coorient. III. Palomino, Daniel Munari, coorient. IV. Título.

CDD : 005

Elaborada por Maria Inez Figueiredo Figas Machado CRB: 10/1612

Marcel Moscarelli Corrêa

Heuristic-based Algorithms and Hardware Designs for Fast Intra-picture Prediction in AV1 Video Coding

Tese aprovada como requisito parcial para obtenção do grau de Doutor em Ciência da Computação no Programa de Pós-Graduação em Computação da Universidade Federal de Pelotas.

Data da Defesa: 14 de fevereiro de 2023.

Banca examinadora:

Prof. Dr. Luciano Volcan Agostini (Orientador)
Doutor em Ciência da Computação pela Universidade Federal do Rio Grande do Sul

Prof. Dr. Guilherme Ribeiro Corrêa (Coorientador)
Doutor em Engenharia Eletrotécnica e de Computadores pela Universidade de Coimbra

Prof. Dr. Daniel Munari Palomino (Coorientador)
Doutor em Ciência da Computação pela Universidade Federal do Rio Grande do Sul

Prof. Dr. Nuno Filipe Valentim Roma
Doutor em Engenharia Eletrotécnica e de Computadores pelo Instituto Superior Técnico

Prof. Dr. Gustavo Freitas Sanchez
Doutor em Ciência da Computação pela Pontifícia Universidade Católica do Rio Grande do Sul

Prof. Dr. Wagner Ishizaka Penny
Doutor em Ciência da Computação pela Universidade Federal de Pelotas

Prof. Dr. Marilton Sanchotene de Aguiar
Doutor em Ciência da Computação pela Universidade Federal do Rio Grande do Sul

Agradeço às instituições públicas de ensino
pela educação gratuita e de qualidade
que me foi fornecida.

“Sob os ventos da redemocratização, dizíamos:

ditadura nunca mais!

Hoje, depois do terrível desafio que superamos, devemos dizer:

democracia para sempre!”

(Lula)

RESUMO

CORRÊA, Marcel Moscarelli. **Heuristic-based Algorithms and Hardware Designs for Fast Intra-picture Prediction in AV1 Video Coding**. 2023. 126f.
Tese (Doutorado em Ciência da Computação) – Programa de Pós-Graduação em Computação. Universidade Federal de Pelotas, Pelotas.

A codificação de vídeo para fins de compressão é indispensável para qualquer aplicação ou serviço baseado na manipulação de vídeos digitais. Sem compressão, conteúdo de vídeo digital moderno requer uma quantidade proibitiva de dados. Um formato de codificação de vídeo define o formato de representação do conteúdo de vídeo em uma forma comprimida, para ser utilizada de maneira conveniente para armazenamento e transmissão. Formatos de vídeo são tipicamente padronizados e têm codificadores e decodificadores de vídeo desenvolvidos para eles, implementados tanto em software quanto em hardware. Esta tese apresenta algoritmos apropriados para implementação em hardware, capazes de reduzir o número de operações associadas à etapa de decisão de modo da predição intraquadros em um codificador de vídeo, que é um dos módulos do codificador que mais consome recursos de processamento. Ainda, esta tese também apresenta arquiteturas em hardware que implementam os algoritmos propostos, otimizadas para baixa potência dissipada e alta eficiência energética. Todas as soluções de software e hardware descritas nesta tese têm como alvo o formato *AOMedia Video* 1 (AV1), que é o estado da arte em formatos de vídeo abertos e livres de royalties. Todos os algoritmos propostos foram testados no software de referência do codificador AV1, utilizando-se condições comuns de teste deste campo de pesquisa, e todas as arquiteturas de hardware foram descritas em VHDL e sintetizadas para tecnologia TSMC 40nm. Os resultados de eficiência de compressão e tempo de codificação dos algoritmos propostos e, também, os resultados de custo em portas lógicas e consumo de energia das arquiteturas de hardware, confirmam que as soluções desenvolvidas durante este projeto de doutorado atendem as demandas das tecnologias atuais de vídeo, como a codificação de resoluções *Ultra-High Definition* (UHD) em alta velocidade e alta qualidade visual.

Palavras-chave: Codificação de vídeo; predição intra; decisão de modo; AV1; arquitetura em hardware.

ABSTRACT

CORRÊA, Marcel Moscarelli. **Heuristic-based Algorithms and Hardware Designs for Fast Intra-picture Prediction in AV1 Video Coding**. 2023. 126f.
Thesis (Ph.D. in Computer Science) – Graduate Program in Computing. Federal University of Pelotas, Pelotas.

Video coding for compression purposes is paramount for any application or service based on digital video. Without compression, modern digital content requires a prohibitively large amount of data. A video coding format defines the format for video content representation in a compressed form, to be used conveniently for storage or transmission. Video formats are, typically, standardized and have video encoders and decoders made for them, in both software and hardware. This thesis presents hardware-friendly algorithms capable of reducing the number of operations of the video encoder mode decision process in the intra-picture prediction module, one of its most time-consuming modules. Additionally, it also presents intra-picture prediction hardware designs, optimized for both low power and high energy efficiency, implementing the proposed algorithms. All software and hardware solutions described in this thesis target the AOMedia Video 1 (AV1) format, which is state-of-the-art in open-source and royalty-free video coding. All algorithms proposed were evaluated in the AV1 reference software using common test conditions, and all hardware designs were described in VHDL and synthesized to TSMC 40nm standard-cells technology. The encoding efficiency and encoding time results for the proposed algorithms, as well as the gate count and energy consumption results for the hardware designs, confirm that solutions developed during this Ph.D. project meet the requirements of current video technology, such as coding of Ultra-High Definition (UHD) resolutions at high speeds and high visual quality.

Keywords: Video coding; intra prediction; mode decision; AV1; hardware design.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS AND ABBREVIATIONS

1-D	One-dimensional
2-D	Two-dimensional
3-D	Three-dimensional
AC	Alternate Current (transform coefficient)
ADST	Asymmetrical Discrete Sine Transform
AOMedia	Alliance for Open Media
ASIC	Application-specific Integrated Circuit
AV1	AOMedia Video 1
AVC	Advanced Video Coding (H.264, MPEG-4 Part 10 standard)
BD-BR	Bjøntegaard Delta Bit Rate
CABAC	Context-adaptive Binary Arithmetic Coding
CAVLC	Context-adaptive Variable-length Coding
CDEF	Constrained Directional Enhancement Filter
CFL	Chroma from Luma
Codec	Coder/Decoder
CMYK	Cyan, Magenta, Yellow, Key (color model)
CTC	Common Test Conditions
dB	Decibel
DBF	Deblocking Filter
DC	Direct Current (transform coefficient)
DCT	Discrete Cosine Transform
DFS	Depth-first Search
DMPU	Directional Mode Prediction Units
DSPU	Directional Sample Prediction Units
DSGF	Dual Self-Guided Filter
FHD 1080p	Full High Definition (1920×1080 pixels)
flipADST	Inverted ADST

FSM	Finite State Machine
HD	High Definition
HD 720p	1280×720 pixels
HEA	High Error Area
HEVC	High Efficiency Video Coding (H.265, MPEG-H Part 2 standard)
HTTP	Hypertext Transfer Protocol
IDTX	Identity Transform
IEC	International Electrotechnical Commission
IP	Internet Protocol
ISO	International Organization for Standardization
ITU	International Telecommunication Union
ITU-T	ITU Telecommunication Standardization Sector
JVET	Joint Video Experts Team
KLT	Karhunen-Loeve Transform
LEA	Low Error Area
LRF	Loop Restoration Filter
MaSBM	Mode-adaptive Subsampling in Block Matching
ME	Motion Estimation
MPEG	Moving Picture Experts Group
MPEG-2	MPEG-2 Video (H.262, MPEG-2 Part 2 standard)
MSE	Mean Squared Error
PDB	Perfectly Directional Blocks
PMMU	Parallel Multiplierless Multiplication Units
PSNR	Peak Signal-to-noise Ratio
RBF	Recursive-based-filtering
RD	Rate-distortion
RDO	Rate-distortion Optimization
RGB	Red, Green, Blue (color model)

RTL	Register-transfer Level
SD	Standard Definition
SAD	Sum of Absolute Differences
SCC	Screen Content Coding
SPMU	Smooth Prediction Multiplication Unit
SSE	Sum of Squared Errors
SSNWF	Separable Symmetric Normalized Wiener Filter
SVM	Support Vector Machine
TbFMD	Texture-based Fast Mode Decision
TCP	Transfer Control Protocol
TGM	Text and Graphics with Motion
TSMC	Taiwan Semiconductor Manufacturing Company
UHD	Ultra-High Definition
UHD 4K	3840×2160 pixels
UHD 8K	7680×4320 pixels
UMPU	Unified Multifilter Prediction Unit
VCEG	Video Coding Experts Group
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
VP8	Royalty-free video codec developed by On2 Technologies
VP9	Royalty-free video codec developed by Google Inc.
VP10	Unfinished royalty-free video codec developed by Google Inc.
VVC	Versatile Video Coding (H.266, MPEG-I Part 3 standard)
YCbCr	Luminance, Blue-difference chrominance and Red-difference chrominance (color model)

CONTENTS

1 INTRODUCTION

Internet-based video traffic has been pushing telecommunication infrastructures to their limit because of the ever-increasing demand for video-based services, such as social media, streaming, video conferencing, and cloud gaming. More recently, in 2019, the world started facing the SARS-CoV-2 pandemic, which led people to heavily depend on video services for their work, educational and social routines more than ever before.

According to Cisco Systems Inc. (2020), from 2017 to 2020, this type of traffic grew 29% annually and was expected to reach 325 exabytes per month by the end of 2022, representing 82% of the global internet traffic. It was also expected that in 2022, of all video traffic, 22.3% would be in Ultra-High Definition (UHD) resolutions, 56.8% in High Definition (HD), and 20.9% in Standard Definition (SD) or lower. Furthermore, by 2023, two-thirds of the installed television sets are expected to be UHD-capable, up from 33% in 2018. This will cause an even bigger impact on the infrastructure because the increase in video definition causes a multiplicative effect on the data volume. For example, a single UHD 4K (3640×2160 pixels) uncompressed frame has four times more data than a Full HD 1080p (FHD) (1920×1080 pixels) frame, and the increase in spatial resolution often comes paired with an increase in temporal resolution (frame refresh rate), which also leads to a linear increase in data.

To address this, standardization bodies such as the Telecommunication Standardization Sector of the International Telecommunication Union (ITU-T) and the International Electrotechnical Commission (IEC) of the International Organization for Standardization (ISO), have been developing video coding standards for decades, with the Versatile Video Coding (H.266/VVC) (ITU-T, 2020; BROSS et al., 2021) being the most efficient video coding standard developed. The H.266/VVC standard was released in July 2020, coming from a long line of successful video coding standards defined by a joint effort of the ISO/IEC MPEG (Moving Picture Experts Group) and the ITU-T VCEG (Video Coding Experts Group) that includes the well-known High Efficiency Video Coding (H.265/HEVC) (ITU-T, 2013; SULLIVAN et al., 2012), Advanced Video Coding (H.264/AVC) (ITU-T, 2003; WIEGAND et al., 2003), and MPEG-2 (ITU-T, 1995).

In recent years, the licensing for commercial use of such standards started to become prohibitively expensive and exceedingly bureaucratic due to a large number

of patent holders associated to each standard, affecting even the largest companies, as explained by Rosenberg (2015) on behalf of the Cisco Systems Inc.:

> Unfortunately, the patent licensing situation for H.265 has recently taken a turn for the worse. Two distinct patent licensing pools have formed so far, and many license holders are not represented in either. There is just one license pool for H.264. The total costs to license H.265 from these two pools is up to sixteen times more expensive than H.264, per unit. H.264 had an upper bound on yearly licensing costs, whereas H.265 has no such upper limit.

Motivated by this, companies started to look for royalty-free alternatives to the standards. Indeed, the incredible growth of the internet is a consequence of its founding core technologies being open and freely implementable (MUKHERJEE *et al*., 2013), such as the HyperText Transfer Protocol (HTTP), Transfer Control Protocol (TCP), and Internet Protocol (IP). Nowadays, however, digital video technology became undeniably a central pillar of the internet experience, making free solutions for video coding a subject of great relevance.

A notable example was the start of the WebM project and the acquisition by Google Inc. of the company named On2 Technologies Inc., which originally developed the VP8 (BANKOSKI *et al*., 2011; BANKOSKI; WILKINS; XU, 2011) coder/decoder (codec), later releasing VP8 freely under the CC BY 3.0 license. The VP9 (GRANGE; RIVAZ; HUNT, 2016; MUKHERJEE *et al*., 2013) format was later developed by Google Inc., which is still used in various of Google's own video services. In 2015, as Google Inc. was working on a successor for VP9, called VP10, other companies were developing their own royalty-free and open-source video codecs: Cisco Systems Inc. and Mozilla Corporation were involved in the creation of Thor (BJØNTEGAARD *et al*., 2016) and Daala (VALIN *et al*., 2016), respectively.

In the end, these three companies, together with eleven more companies, joined efforts and founded the Alliance for Open Media (AOMedia) industry consortium, later releasing the AOMedia Video 1 (AV1) (RIVAZ; HAUGHTON, 2019; HAN *et al*., 2021) in June 2018, to be the state-of-the-art royalty-free video format. The AV1 is highly based on features from all the unreleased VP10, Thor, and Daala codecs.

Several new coding tools were developed and enhanced in the next-generation codecs (AV1 and H.266/VVC) to deal with the new requirements of video applications and to provide high coding efficiency. These improvements include larger block sizes, flexible block partitioning structures, a higher number of intra prediction modes, the support of affine modes for inter prediction, more transform sizes and types,

improved implementations of quantization and entropy coding, more in-loop filters, and many other novelties.

Even though both the AV1 and H.266/VVC can achieve a satisfactory performance for current video content, this efficiency implicates a very high computational effort. As a consequence, video encoding is an unfeasible task for software solutions when real-time processing and high resolutions are desired, even for high-end devices.

Moreover, in a world where most video-enabled devices are powered by batteries, an efficient combination of hardware-friendly algorithms and Application-specific Integrated Circuit (ASIC) designs is mandatory to produce low-power devices and to allow for faster encoding and decoding speeds in video systems. Although solutions of this type have been proposed for previous codecs, these cannot be used directly in the current codecs without being redesigned to some extent.

1.1 Research Hypothesis

The current-generation video codecs, AV1 and H.266/VVC, outperform their predecessors by a significant margin. According to experiments conducted by Nguyen and Marpe (2018), AV1 shows an improvement of 23.4% when compared to VP9, and H.266/VVC shows an improvement of 31.6% when compared to H.264/HEVC, both measured in terms of Bjøntegaard Delta Bit Rate (BD-BR) (see Section 2.4).

Such improvements are due to the increased number of block sizes supported by these codecs, and also due to the increased set of modes supported by different stages of the coder. However, this also led to the growth of the combinatorial space, making the encoder task of selecting the best modes drastically more complex. Separate studies made by Saldanha *et al.* (2020) and Bossen *et al*. (2021) suggest that the H.266/VVC reference software requires up to 27 times more computational effort than the H.265/HEVC reference software, whilst a study made by Nguyen and Marpe (2018) shows that the AV1 reference software requires up to 58 times more run time than the VP9 reference software.

Regarding the intra-picture prediction stage of the AV1, which is the focus of this project, a study made by Chuang *et al.* (2022) evaluated the impact of each of the novel tools added to the AV1 intra-prediction (see Chapter 3) individually, in terms of encoding time difference and image distortion (see Section 2.4). Table 1 shows this impact in a test setup where each tool is enabled one at a time while the others are

disabled. It can be observed that some tools alone can increase the encoding runtime significantly and, when added together, these tools are expected to increase the runtime of the baseline intra prediction stage. Regardless of the encoding time impact, these tools must not be disabled, since it can also be observed in the same table that the reduction in image distortion is also very expressive.

Table 1 Impact of individual intra prediction tools in an AV1 encoder (CHUANG *et al*., 2022).

Tool	PSNR-Y (%)	Encoding Time (%)
Intra Angle Delta	−2.42	103.74
Paeth Mode	−0.14	101.95
Smooth Modes	−0.90	104.63
Recursive-based-filtering Modes	−0.77	109.62
Filtering of Ref. Samples	−0.36	102.47
Chroma from Luma Mode	−0.36	101.04
Intra Block Copy Mode	−4.81	102.75
Palette Mode	−6.47	100.77

The AV1 was chosen as the focus of this work due to its very high commercial relevance, for being both open-source and royalty-free, and also due to its academic relevance, for adding many novel and complex tools never studied before. Furthermore, considering the high relevance of the AV1 intra prediction and the research opportunities related to it, as will be explained in Chapter 4, this stage of the encoder was chosen to be the target of this research project.

Based on these facts, the question this research project sought to answer is:

"*Considering the very high computational effort required by the AV1 intra-picture prediction, is it possible to generate novel algorithms and hardware-based solutions able to support the processing of ultra-high-definition videos in real-time?"*

Justified by the large number of novel tools introduced to the intra-picture prediction stage, associated with the small number of works in the literature seeking to improve this stage of the encoder, two main research hypotheses were explored in this thesis.

First hypothesis:

"*Information about the direction and smoothness of the input image texture, as well as the error distribution pattern of each prediction mode, can both be used to develop heuristics to reduce the number of AV1 intra-picture prediction modes processed and the number of operations executed in distortion metric calculation for block matching.*"

Second hypothesis:

"An efficiently designed hardware architecture for the original AV1 algorithms and the proposed heuristic-based algorithms is a promising solution to allow the processing of ultra-high-definition videos in real-time by the AV1 intra prediction."

Then, considering these hypotheses, the main thesis investigated in this work is:

"It is possible to reduce the computational effort of the AV1 intra-picture prediction by developing hardware-friendly heuristic-based algorithms and, then, generating efficient hardware designs able to process ultra-high-definition videos in real-time."

1.2 Main Contributions

The main contribution of this research project is the development of both software and hardware solutions for fast the intra prediction in AV1.

Many works were published with specific contributions as the project developed:

- Development of an AV1 intra-picture prediction hardware designs (CORRÊA *et al*., 2019a, 2019b, 2020a, 2020b; NETO *et al*., 2020, 2021a, 2021b, 2022);
- Development of hardware-friendly heuristic-based algorithms capable of reducing the number of operations in intra prediction (CORRÊA *et al*., 2022a, 2022b);
- Development of intra-picture prediction hardware designs optimized with the proposed algorithms (CORRÊA *et al*. 2022b).

1.3 Thesis Organization

This thesis is organized as follows:

Chapter 2 provides the basic video coding concepts needed for the understanding of the proposed solutions, and also provides information on different parts of an AV1 coder.

Chapter 3 provides an in-depth technical view of the AV1 intra-picture prediction techniques.

Chapter 4 presents a review of the related works that propose system-level solutions for complexity reduction in an AV1 encoder, as well as works that propose

dedicated hardware designs for modules of an AV1 encoder and decoder. It also discusses opportunities in the area of research.

Chapter 5 presents heuristic-based algorithms for the intra prediction stage of the AV1 encoder.

Chapter 6 presents hardware designs for the intra prediction stage of the AV1 encoder.

Chapter 7 presents the conclusions of this thesis, summarizing the major results, identifying possible extensions of the project, and pointing out future research directions in the area.

Appendix A presents the test sequences and coding parameters used for all experiments during this Ph.D. project.

Appendix B lists all the peer-reviewed published papers that were produced during the Ph.D. project.

2 VIDEO CODING BACKGROUND

This chapter presents the basic concepts behind the video coding (compression) process.

2.1 Representation of Digital Videos

A video is made of a series of still images that, when displayed at a sufficiently high refresh rate, provides the viewer with the perception of smooth movement. Video sequences vary in spatial resolution, refresh rate, color space, and other characteristics. Contemporary videos tend to be fully digital through the entire process of production, distribution, and playback.

Each still image in a digital video sequence is called a frame, organized as a matrix of pixels. In this context, a pixel is the smallest addressable element in a frame, and it is a combination of different samples of a color space.

A color space is an arbitrary model for representing colors as tuples, e.g., triples in the Red, Green, Blue (RGB) model, and quadruples in the Cyan, Magenta, Yellow, Key (CMYK) model. The RGB color model is typically used in display devices of various technologies, and it is an additive color model in which the red, green, and blue primary colors of light are added to reproduce a broader array of colors. The CMYK model, on the other hand, is a subtractive color model that mixes the subtractive primaries cyan, magenta, and yellow to block light, rather than adding it, which is particularly convenient for color printers.

For video coding, however, the Luminance, Blue-difference Chrominance, and Red-difference Chrominance (YCbCr) model is the most suitable. In this color model, the luminance information, which is closely related to the perception of brightness, is completely separated from chrominance information, allowing encoders to take advantage of the human visual system characteristic of being much more sensitive to luminance than to color information. One practical example of this is the color subsampling technique, which is used to decrease the resolution allocated to the chrominance channels, without significant loss of information being perceived by the human eye. The most common color sampling schemes are called 4:4:4, 4:2:2, 4:2:0, and 4:0:0. In the 4:4:4 sampling, the resolution of the luminance and both chrominance channels are kept intact and no color information is discarded, whereas in the 4:2:2 and 4:2:0 subsampling, the resolution of both chrominance samples is reduced by 50%

and 75%, respectively. Finally, in the 4:0:0 scheme, only luminance information is carried.

Figure 1 shows a picture in full color and the same picture with the three channels of the RGB color model separated (left), and the same for the YCbCr color model (right). It can be observed clearly that each channel of the RGB color model represents the intensity of each of the primary colors, and that there is a part of the luminance information shared among all three channels. Furthermore, it can also be observed that the luminance channel Y of the YCbCr color model has all the luminance information, seen as a grayscale image, whereas both the chrominance channels only have color information, making it impossible to perceive edges and depth information in detail due to the lack of light information.

This way, the representation of digital videos in an uncompressed format demands a prohibitive amount of storage space or network bandwidth. For example, a UHD 4K video, displayed at a refresh rate of 30 fps (frames per second), with 8 bits per channel in the 4:2:0 sampling scheme, which are qualities of videos commonly used in internet streaming and digital television broadcasting nowadays, requires a bandwidth of approximately 3 Gbps (three billion bits per second). A video with these same qualities and 30 minutes of duration requires approximately 672 GB (672 billion bytes) of storage space.

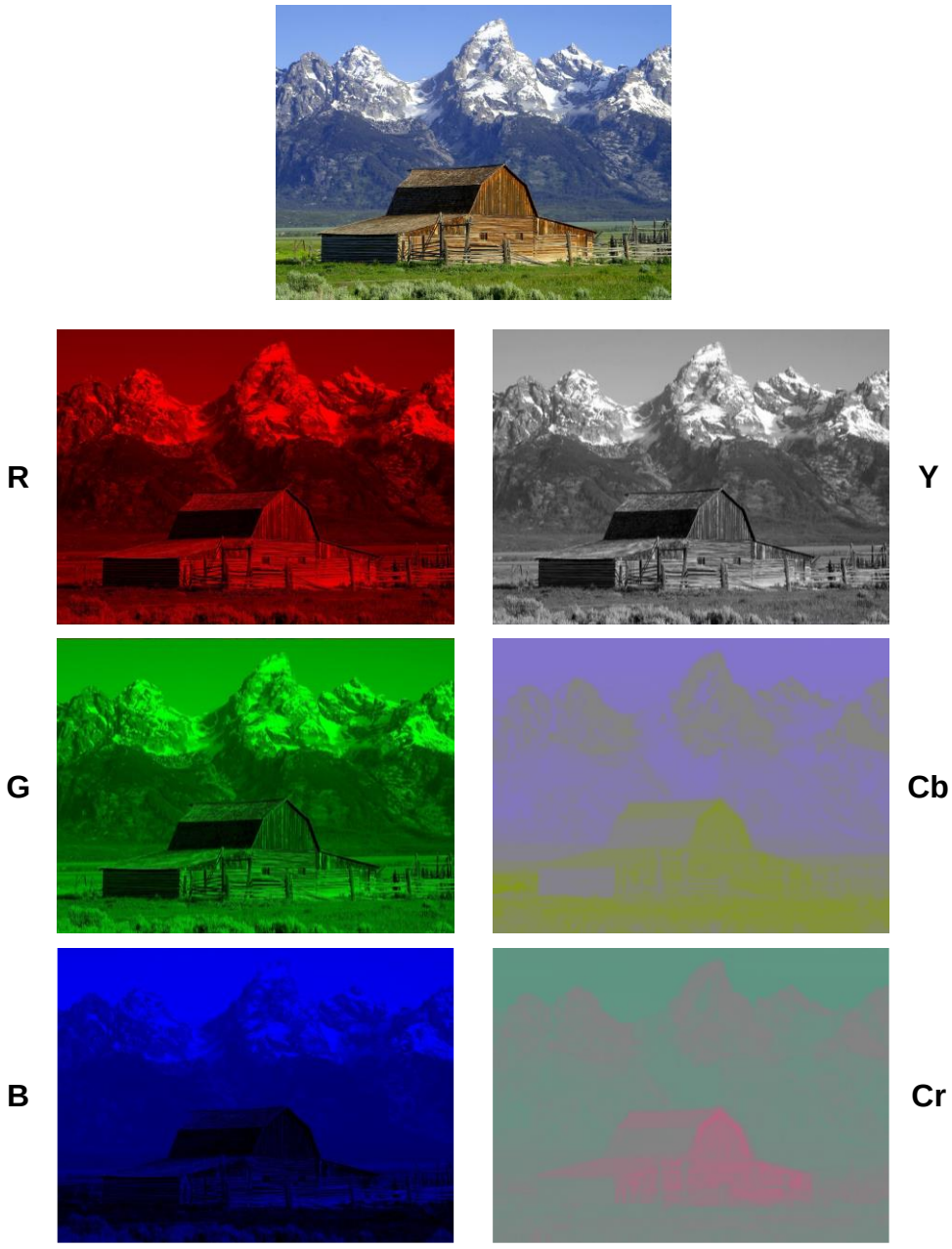

Figure 1 Example of the RGB (left) and YCbCr (right) channels separated. Picture: John Moulton Barn (Public domain).

2.2 Redundancies in Digital Videos

Although digital videos demand a very high volume of data to be represented, much of this data can be compressed due to the strong correlation both between successive frames and within the frame content itself (GHANBARI, 2011). The goal of video compression is to eliminate irrelevant and redundant data present in the video representation.

Irrelevant information in the video representation is always of a visual nature and is deemed irrelevant due to the limitations of the human visual system. For

example, the human eye is not sensible to high-frequency distortions (CORRÊA, 2014),

i.e., it can effectively perceive subtle variations of luminance over a relatively large area but struggles to perceive such variations in small areas (CORRÊA *et al*., 2016). Hence, for compression purposes, an encoder can discard information that the human eye is not sensible to.

Redundant information in a digital video can be of a visual nature (spatial and temporal) or statistical nature (entropic). These types of redundancy can be defined as follows (AGOSTINI, 2007):

- **Spatial Redundancy:** Also called intra-picture redundancy, it is the similarity between pixels spatially close to each other. This redundancy can be explored in the spatial domain by the intra-picture prediction stage of a video coder.
- **Temporal Redundancy:** Also called inter-picture redundancy, it is the similarity between frames temporally close to each other. Because of the high refresh rate of a video, pixels tend to not change from one frame to another, and changes are likely to be small variations in pixel intensity (e.g., change in lighting) and position (e.g., the motion of an object or the entire background). This redundancy can be explored by the inter-picture prediction stage of a video coder.
- **Entropic Redundancy:** In information theory, entropy is related to the occurrence frequency of symbols. In a scenario where all symbols have the same probability of occurring, the same number of bits must be used to represent these symbols. On the other hand, if some symbols have a higher probability of occurring, then this redundancy can be explored by assigning variable length codes to these symbols (i.e., codes with fewer bits assigned to more frequent symbols). The entropic redundancy is explored by the entropy coding stage of a video coder.

2.3 Compression of Digital Videos

Irrelevant and redundant information can be explored by using lossless and lossy compression. The intra-picture prediction, inter-picture prediction, and entropy coding all aim at reducing redundancy without causing any loss of information. Compression techniques that do not cause loss of information are classified as lossless compression techniques (i.e., the decompressed data is identical to the original). On the other hand, compression techniques that aim at discarding information that is not relevant or is less relevant to the human visual system are classified as lossy compression techniques (i.e., the image can be reconstructed with a visual fidelity relative to the compression parameters, but never identical to the original).

Video compression can be lossless throughout the entire process, which can be particularly useful for applications that cannot tolerate any loss of sensitive information and the addition of compression artifacts. However, for the majority of consumer applications, lossy techniques are used, as these are capable of adding substantial gains to the resulting compression rate.

2.4 Distortion Metrics

Lossy video compression causes distortion, and this distortion must be measured. Visual quality, however, is inherently subjective and, therefore, it is difficult to obtain a completely accurate measurement of it. There are standardized methodologies for the assessment of subjective image quality including, general testing methods, the grading scales used during assessments, and the viewing conditions recommended for carrying out assessments, which are well described in Recommendation ITU-R BT.500-14 (ITU-T, 2019). Quality can also be measured objectively with metrics that compare pixels of the original and the reconstructed image.

The most commonly used objective distortion metric is the Peak Signal-to-noise Ratio (PSNR). The PSNR, measured in decibels (dB), of a test image T of size $M \times N$, when compared to the original image O, is described in eq. (1), where the dividend MAX is the maximum value possible for an unsigned sample (e.g., 255 for an 8-bit sample) and the divisor is the Mean Squared Error (MSE) function, described in eq. (2), which is also a distortion metric by itself.

$$PSNR(O,T)_{dB} = 10\log_{10}\left(\frac{MAX}{MSE(O,T)}\right) \quad (1)$$

$$MSE(O,T) = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\left(T_{i,j} - O_{i,j}\right)^2 \quad (2)$$

To measure the distortion in smaller regions of an image, like when comparing a predicted block against the original in inter- and intra-picture prediction (known as block matching), simpler metrics are used, such as the MSE, the Sum of Absolute Differences (SAD) and the Sum of Squared Errors (SSE), respectively described in eqs. (2), (3), and (4).

$$SAD(O,T) = \sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\left|T_{i,j} - O_{i,j}\right| \quad (3)$$

$$SSE(O,T) = \sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\left(T_{i,j} - O_{i,j}\right)^2 \quad (4)$$

Another metric for perceptual visual quality is the Video Multi-method Assessment Fusion (VMAF) (NETFLIX INC, 2022a; LIU *et al*., 2013; LIN *et al.*, 2014), which attempts to predict subjective quality by combining multiple objective quality metrics. According to Netflix Inc. (2022b), the basic rationale is that each elementary metric may have its strengths and weaknesses, and by fusing elementary metrics into a final metric using a machine-learning algorithm, in this case, the Support Vector Machine (SVM) regressor, the final metric could preserve the strengths of the individual metrics, and deliver a more accurate final score.

However, the measurement of compression efficiency should not rely on distortion alone, but also on the resulting bit rate, that is, a Rate-distortion (RD) metric. A notable example of such a method is the Bjøntegaard Model (BJØNTEGAARD, 2011). In this model, PSNR is the metric of choice for distortion, because of its simplicity, and also because it reasonably matches subjective opinion scores.

In the scope of this thesis, the metric used to evaluate the proposed algorithms is the Bjøntegaard Delta Bit Rate (BD-BR), which reports the average bit rate difference in percent for two videos (e.g.: original versus compressed) considering the same PSNR.

2.5 Hybrid Block-based Video Encoder

Most of the contemporary encoders are based on the following signal and data processing operations: (i) inter- and intra-frame prediction, (ii) de-correlating transform (T module), (iii) quantization (Q module), and (iv) entropy coding, as shown in Figure 2. A reconstruction loop (complete decoder) with inverse quantization (IQ module) and inverse transform (IT module) is also included because the coder must use only reference frames available to the decoders, so decoders can replicate identical predictions (SALDANHA, 2021). Additionally, an optional in-loop filtering module (not shown in Figure 2) can also be included to improve the subjective image quality of reconstructed frames. Hybrid block-based encoders apply the abovementioned operations after partitioning frames into smaller blocks and use both motion- and still-picture coding techniques (SULLIVAN; WIEGAND, 1998).

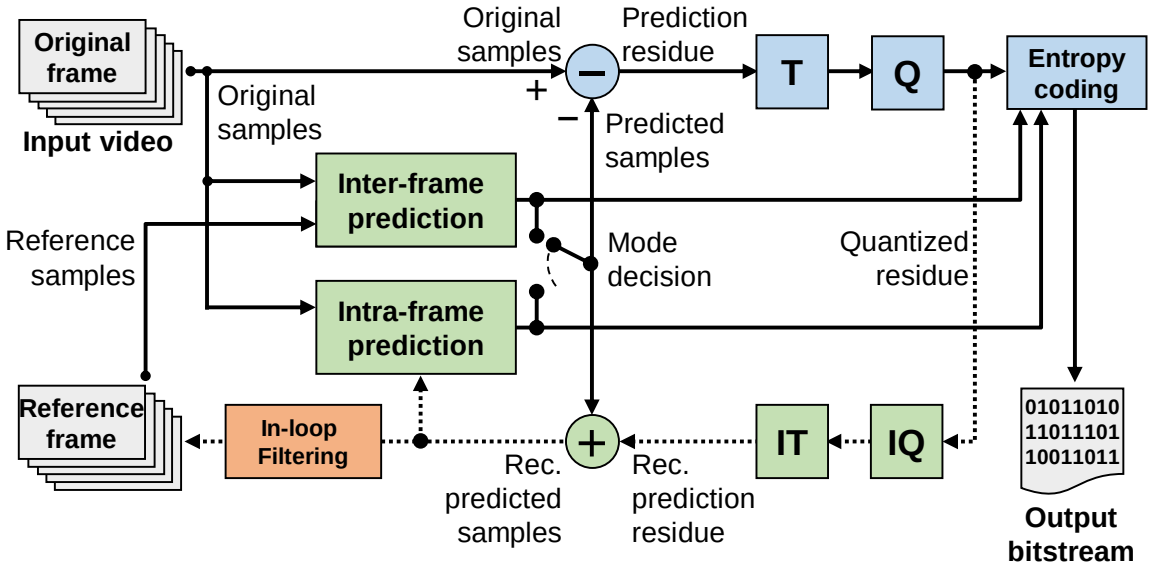

Figure 2 Diagram of a typical hybrid block-based video coder.

The following subsections describe the different stages of video compression shown in Figure 2, while briefly explaining the AV1 implementation of these stages.

2.5.1 Frame Partitioning

Before any signal processing operation can happen, in block-based video coding, a frame must be divided into several blocks of pixels of the maximum size supported by the codec. These blocks can then be further subdivided into smaller blocks during the prediction process. Each video codec defines a variable range of block sizes it can use.

In AV1, a frame is initially partitioned in Superblocks (SBs), which is the biggest block size supported (128×128 or 64×64 pixels). To deliver an optimal prediction for each SB, the encoder can further divide each SB using a 10-way partition tree structure, as illustrated in Figure 3. In the figure, partitions filled in blue are final, but all four subpartitions of the unfilled partition (SPLIT) can be recursively divided based on the same 10-way tree structure, down to 4×4 pixels, which is the smallest supported block size (RIVAZ; HAUGHTON, 2019). This way, the 24 block sizes supported in AV1, including symmetrical and rectangular sizes, are the ones contained in the following set: {4×4, 8×8, 16×16, 32×32, 64×64, 128×128, 4×8, 8×4, 8×16, 16×8, 16×32, 32×16, 32×64, 64×32, 64×128, 128×64, 4×16, 16×4, 8×32, 32×8, 16×64, 64×16, 32×128 e 128×32} (HAN *et al*., 2021).

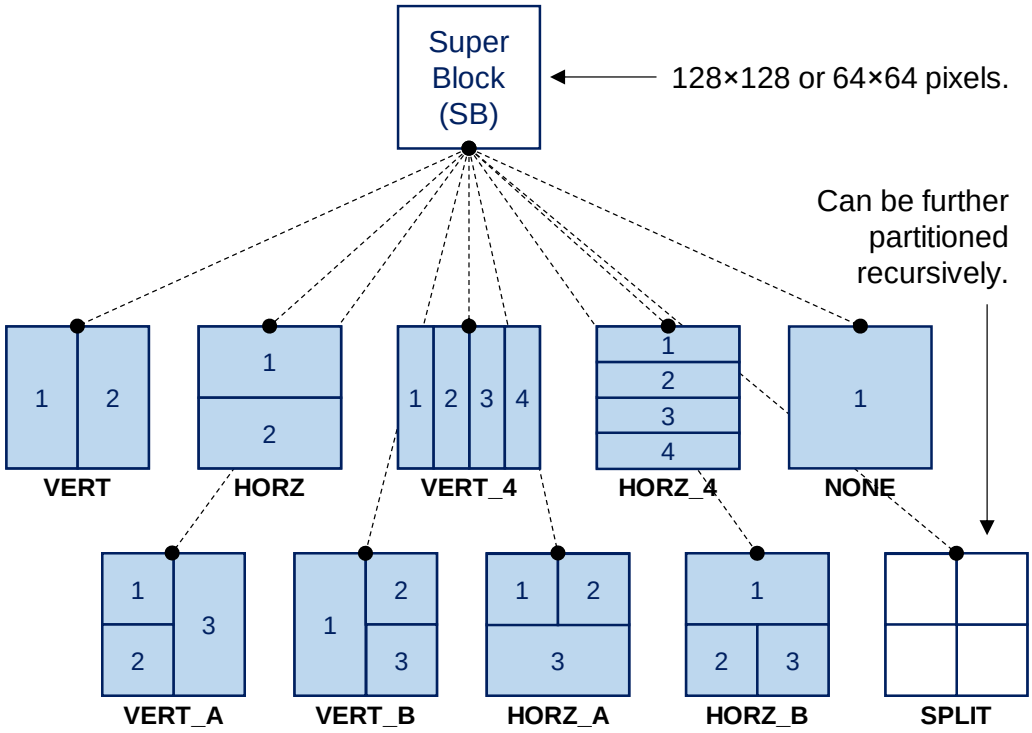

Figure 3 AV1 10-way block partitioning tree. The numbers inside each final subpartition (in blue) indicate the order these will be processed by the following stages (raster scan). The name of each partition mode is shown below blocks.

Figure 4 shows an example of frame partitioning (BEBENITA, 2017), generated using the AOM Analyzer tool (XIPH.ORG FOUNDATION, 2022). In the figure, it can be observed that the variable block size structure gives the coder freedom to explore less detailed portions of the image with large block sizes (reducing the overhead of signaling multiple smaller blocks), and more detailed portions of the image with small block sizes (prioritizing visual quality).

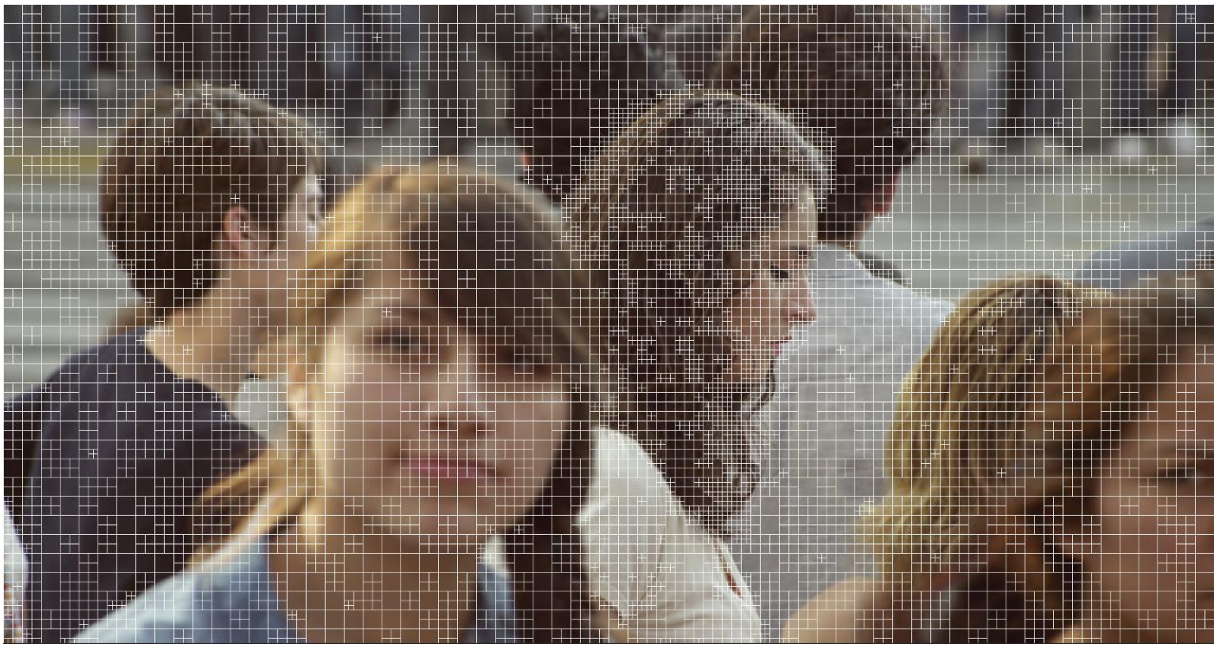

Figure 4 Example of frame partitioning of the test sequence Crosswalk (Property of Netflix Inc. and licensed under CC BY-NC-ND 4.0).

2.5.2 Prediction Stage

In inter-picture prediction, or simply inter prediction, the translational motion of areas of the frame can be estimated by searching for similar blocks in a reference frame around its original position, and by taking the difference of the best match, in what is called the Motion Estimation (ME) algorithm of a motion-compensated prediction. The ME reduces the temporal redundancy of a frame, and results in blocks of residual information (error) and Motion Vectors (MV) that are used to describe the displacement of predicted blocks. For static parts of the scene, residual differences can be zero, with no error to be coded, whereas for parts of the scene with motion, a significant error can exist, which needs to be coded.

The AV1 motion-compensated prediction supports all 24 block sizes and may use up to seven reference frames, with four being frames that precede the current frame in terms of display order, and three coming after. AV1 brings a variety of novel solutions when compared to older codecs, such as: (i) Affine Motion Compensation, which uses affine transformations to capture non-translational object movements (e.g., rotation, translation, and scaling); (ii) Compound Prediction, which linearly combines two predictions from different reference frames in a single one; and others.

In intra-picture prediction, or simply intra prediction, spatial redundancy can be explored in the spatial domain by predicting entire blocks using reconstructed samples

of previously encoded spatial neighbor blocks from the same frame. This type of prediction usually applies filters to the reference samples to generate different kinds of directional and smooth textures. Just like in the inter prediction, the objective is to obtain a predicted block that results in the lowest residual information when compared to the original block.

AV1 intra prediction supports the 19 block sizes that are equal to or smaller than 64×64 samples. For all mentioned block sizes, the codec supports 56 directional prediction algorithms (modes) to explore spatial redundancies in directional textures (e.g., samples belonging to the same edge of an object tend to be similar), and also supports various non-directional prediction modes, such as: (i) DC, similar to the mode used in many older codecs; (ii) Smooth, Smooth Vertical, and Smooth Horizontal, inspired in the H.265/HEVC Planar mode and H.264/AVC Plane mode; (iii) Paeth, evolved from the VP9 True Motion mode; (iv) five different Recursive-based-filtering (RBF) modes, which attempt to break data dependency by further dividing the intra block into smaller 4×2 patches; and (v) Chroma from Luma (CFL) mode, which predicts chrominance samples based on the information of the luminance prediction (TRUDEAU; EGGE; BARR, 2018). Two modes particularly efficient for Screen Content Coding (SCC) are also available: the Intra Block Copy (LI *et al*., 2018) and Color Palette (GUO *et al*., 2014) modes. Chapter 3 gives an in-depth view of the AV1 intra prediction module, as it is the focus of this work.

As mentioned above, the goal of both inter and intra predictions is to minimize the residual information, which can be coded much more effectively than the original visual data. Figure 5 shows an original luminance of the Akiyo test sequence (left), an example of AV1 intra prediction, restricted to the 8×8 block size to give a better visualization of block boundaries (middle); and the resulting residual information (prediction error) that must be coded (right). The figure demonstrates the low-energy nature of the residual information that can be obtained when the predictions are done effectively, where white and black areas are of high-energy (negative and positive differences), and gray areas are closer to zero energy.

Original **Predicted** **Residual Information**

Figure 5 Left: Luminance frame of the test sequence Akiyo (Property of Stadium Inc.). Middle: Same frame predicted with 8×8 AV1 intra prediction modes. Right: Residual information obtained.

Finally, it is important to mention that by using the side information that tells how a block was coded (i.e., intra mode, reference frame, motion vector, etc), a decoder can always replicate the same prediction done by the coder, and by adding it to the residual block, the reconstructed block is obtained.

2.5.3 Transform Coding

The transform coding is another stage that allows the removal of spatial redundancies in images, but in the frequency domain, as natural images tend to concentrate most of their energy in low-frequency coefficients (GHANBARI, 2011). The transform itself, however, does not result in compression, because the signal energy in the pixel domain is equal to the energy in the frequency domain. Although, in the frequency domain, coefficients with irrelevant magnitude can be quantized (resulting in zero) and high-frequency coefficients can be quantized to a higher degree for not being very important to the human visual system. For this reason, a transform kernel must also be efficient in separating components with minimal inter-dependence (decorrelation) (RICHARDSON, 2010).

The Two-dimensional (2-D) Discrete Cosine Transform II (DCT-II), as defined in eq. (5), is a widely used transform in image and video compression. When applied to an image matrix X of size $N \times M$, it results in a matrix Y of coefficients of the same size, where $Y_{0,0}$ is the zero-frequency coefficient called DC, whereas the remaining coefficients are called AC coefficients.

$$Y_{x,y} = \sum_{i}^{N-1} \sum_{j}^{M-1} X_{i,j} \cos\left[\frac{\pi}{N}\left(i + \frac{1}{2}\right)x\right] \cos\left[\frac{\pi}{M}\left(j + \frac{1}{2}\right)y\right] \quad (5)$$

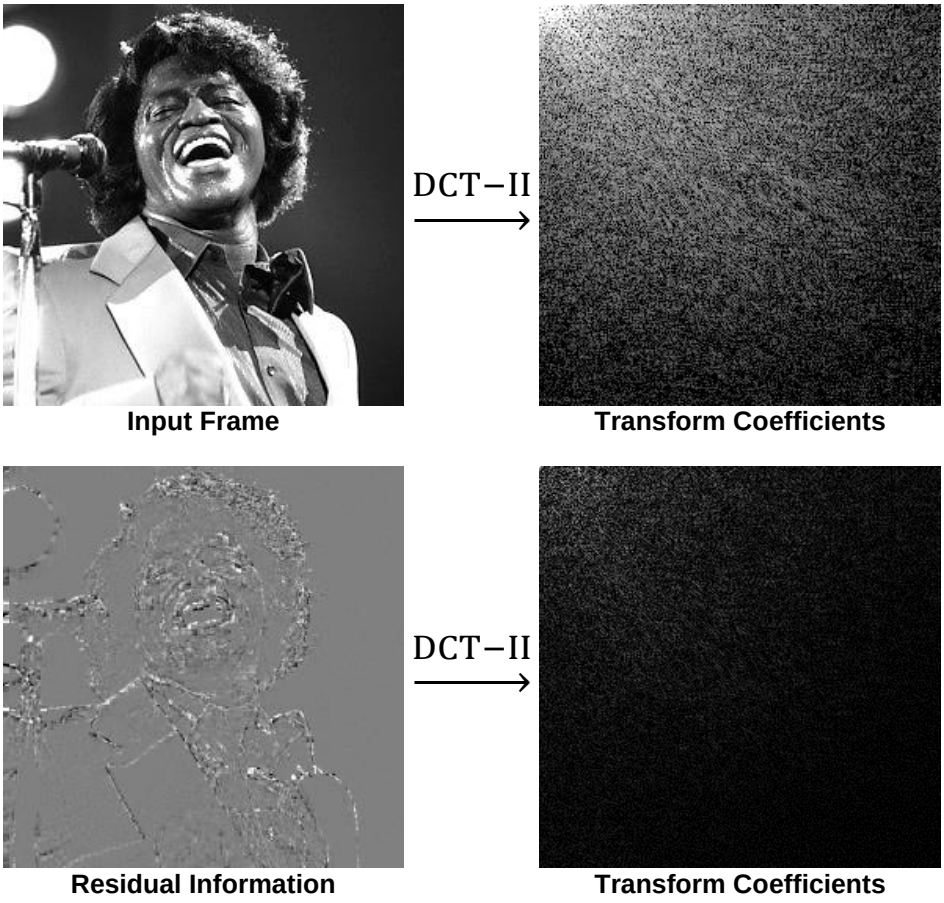

Figure 6 Top: Input image. Bottom: Residual information of the same image after AV1 intra prediction.

Figure 6 illustrates the advantage of applying the DCT-II to the low-energy residual information. At the top, an input image of artist James Brown of 256×256 pixels in size is shown, whereas, in the bottom, the residual information resulting from applying AV1 intra prediction to the same frame is shown. Two things can be observed: (i) the residual information can be represented with much smaller transform coefficients, and (ii) most big coefficients (bright area) are concentrated in the top-left (shown in logarithmic scale for better visualization).

AV1 defines transform kernels for the 19 symmetrical and rectangular sizes equal to or smaller than 64×64. A rich set of 2-D transform kernels is defined for both inter and intra predicted blocks consisting of 16 combinations of one-dimensional (1-D) vertical and horizontal DCT, Asymmetrical Discrete Sine Transform (ADST), flipped ADST, and Identity Transform (IDTX) (HAN *et al*., 2021). The DCT is used for being a good low-complexity approximation of the optimal Karhunen-Loeve Transform (KLT). For the intra predicted blocks, which tend to concentrate higher residual energy on the bottom and/or right corners, the asymmetrical transforms ADST and flipped ADST are particularly effective (PARKER et al., 2016). The IDTX, when combined with

the other 1-D transforms, provides 1-D transforms that can be useful for dominant horizontal and vertical patterns in texture (HAN *et al*., 2021). Furthermore, a 2-D IDTX is equivalent to a transform skip, and it can be effective for certain patterns found in SCC (PARKER et al., 2016).

2.5.4 Quantization

Quantization is the process of mapping a signal with a range of values to a quantized signal with a reduced range of values and, therefore, is a lossy process. Examples of the quantizer and inverse quantizer functions for image compression are described in eq. (6) and eq. (7), where *step* is directly associated with a Quantization Parameter (QP). It can be observed that the lossy aspect of the quantizer function *Q*(*Y*) is a consequence of the fractional part of the division being discarded, thus the inverse quantizer function *IQ*(*Y*) is not capable of recovering the original pre-quantized value. The QP plays an important role in the RD control of a video coder, as higher QPs will result in a smaller, and hence more compressible range of transform coefficients, whereas smaller QPs will result in a higher range of coefficients that best match the pre-quantized values.

$$Q(Y) = \left\lfloor \frac{Y}{step} \right\rfloor \tag{6}$$

$$IQ(Y) = Y \times step \tag{7}$$

The QPs in AV1 range between 0 and 255, and each QP has associated with it two different *step* values, a smaller one to be applied exclusively to the DC coefficient, and a larger one to be applied to the AC coefficients (RIVAZ; HAUGHTON, 2019). Additionally, AV1 supports 15 sets of quantization weighting matrices that can further scale the quantization step differently for each frequency coefficient (HAN *et al*., 2021), allowing for a better exploration of the human visual system.

2.5.5 Entropy Coding

The entropy coding processes matrices of quantized coefficients, which can be reordered to group the coefficients from the region with multiple zeros, and lateral data, such as motion vectors, prediction modes, and a variety of bitstream headers, to reduce their statistical redundancy. Variable-length coding (such as Huffman coding) and arithmetic coding are the common methods of entropy coding

used in video compression. Since efficient entropy coding depends on accurate symbol probability models (RICHARDSON, 2010), video standards further improved these algorithms to be context-based, using local spatial and/or temporal characteristics of the signal to estimate the probability of symbols being encoded. Notable examples of entropy coding algorithms are Context-adaptive Variable-length Coding (CAVLC) and Context-adaptive Binary Arithmetic Coding (CABAC), the latter being significantly more efficient at cost of requiring more processing power from both the encoder and decoder.

AV1 uses a context-adaptive multi-symbol arithmetic coder, with integer symbols ranging from 2 to 14, and the probability model is updated per symbol coding. Specifically, for transform coefficient coding, AV1 allows the coefficient matrix to be reordered (mapped to a 1-D array) in the following scan orders: (i) column scan for 1-D vertical transforms, (ii) row scan for 1-D horizontal transforms, and (iii) zig-zag scan starting at the DC coefficient and moving towards the opposite corner for 2-D transforms, including the 2-D IDTX (HAN *et al*., 2021).

2.5.6 In-loop Filtering

The encoding process inevitably adds artifacts in the compressed videos, mainly because of the block partitioning and the quantization. Typical coding artifacts are blocking, ringing, and blurring. These artifacts decrease the video subjective quality and compromise the quality of prediction references. Thus, all modern codecs allow the use of in-loop filtering to reduce these artifacts.

AV1 supports three optional in-loop filters: the Deblocking Filter (DBF), the Constrained Directional Enhancement Filter (CDEF) (MIDTSKOGEN; VALIN, 2018), and the Loop Restoration Filter (LRF) (MUKHERJEE *et al*., 2017). The improved filtered frames are used as reference frames for the prediction of subsequent frames.

2.5.7 Mode Decision

The RD efficiency of the compression is based on a complex interaction between various possibilities of coding parameters, like block sizes, prediction modes, transform types, quantization parameters, and others. Therefore, one of the biggest challenges in video compression is the control of an encoder.

The control of the encoder, or its mode decision, uses the Rate-distortion Optimization (RDO) technique for taking decisions of "*What part of the image should be encoded using what tool?*" by minimizing the distortion *D*, where the number of bits needed *R* is subject to a bit rate constraint R_i, as defined in eq. (8). The optimization task can be solved using Lagrangian optimization, where *D* is weighted against *R*, and the Lagrangian rate-distortion *J* is minimized for a particular value of the Lagrange multiplier λ, as read in eq. (9). Each solution to eq. (9) for a given value of the Lagrange multiplier λ corresponds to an optimal solution to eq. (8) (SULLIVAN; WIEGAND, 1998).

$$\min\{D\}, \text{where } R < R_i \tag{8}$$

$$\min\{J\}, \text{where } J = D + \lambda R \tag{9}$$

As there are no simple models for estimating the RD cost *J* that will result from a given combination of coding parameters for a block, the RDO technique in a video encoder must pass the block through all the coding loop to obtain the real RD cost for that combination of parameters (CORRÊA *et al*., 2016). The RDO is optimal for mode decision, but the number of possible coding parameter combinations is so large that testing all possibilities is an unfeasible task. In practical applications, different modules of the encoder often implement suboptimal local mode decisions, known as fast mode decision algorithms, to reduce the set of parameters sent to the RDO for the more expensive evaluation.

3 AV1 INTRA-PICTURE PREDICTION

This section gives an in-depth view of the AV1 intra prediction, which is the focus of the optimizations proposed in this thesis.

The AV1 intra prediction process is invoked for intra blocks to predict a part of the block corresponding to a transform block. When the transform size used is smaller than the intra block itself, this process is invoked multiple times within the intra block, in raster order, using the same intra mode (RIVAZ; HAUGHTON, 2018).

3.1 Reference Samples

To predict a block, the intra prediction modes use reference samples from previously reconstructed blocks located to the left and above the current block. The number of reference samples needed from each side is the sum of the width and height of the block to be predicted, plus one sample.

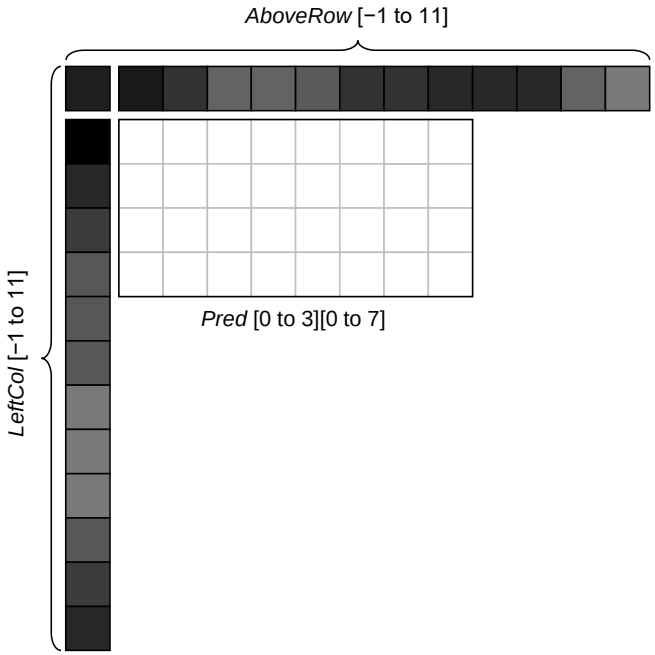

Figure 7 Block of size 8×4 to be predicted using reference arrays of size 13.

Figure 7 illustrates this concept for a block of size 8×4 (white squares). From this point forward, reconstructed samples in the current frame will be referred to as a 2-D array called *CurrFrame*, the current block to be predicted will be referred to as a 2-D array called *Pred*, and the reference samples above and to the left of *Pred* will be referred as two 1-D arrays of samples called *AboveRow* and *LeftCol*, respectively. Additionally, the width and height of *Pred* will be referred to as *W* and *H*.

Since the reference samples must come from already reconstructed transform blocks, some of these may not be available and must be somehow replaced. This happens, for example, when the coder has to predict a block located at the left edge of the frame, which does not have any reference sample to its left. At the beginning of the prediction process, for a given block, both *AboveRow*[−1] and *LeftCol*[−1] are set according to Table 2, while the remaining positions of *AboveRow* and *LeftCol* are set according to Table 3 and Table 4, respectively. In these tables, variables *X* and *Y* point to the top-left position of *Pred* within *CurrFrame*.

Table 2 Top-left reference sample derivation

Above Available?	**Left Available?**	**Procedure for *AboveRow*[−1] and *LeftCol*[−1]**
True	True	*AboveRow*[−1] and *LeftCol*[−1] are set to *CurrFrame*[*Y*−1][*X*−1].
True	False	*AboveRow*[−1] and *LeftCol*[−1] are set to *CurrFrame*[*Y*−1][*X*].
False	True	*AboveRow*[−1] and *LeftCol*[−1] are set to *CurrFrame*[*Y*][*X*−1].
False	False	*AboveRow*[−1] and *LeftCol*[−1] are set to 128, 512 or 2048, depending on the bit depth (8, 10 or 12).

Table 3 Above reference samples derivation

Above Available?	**Left Available?**	**Above Right Available?**	**Procedure for *AboveRow*[*pos*] for *pos* = 0...*w*+*h*−1**
True	True	True	If *pos*<2**W*, then *AboveRow*[*pos*] is set to *CurrFrame*[Y−1][*X*+*pos*]. Otherwise, *AboveRow*[*pos*] is set to *CurrFrame*[Y−1][*X*+2**W*−1].
True	True	False	If *pos*<*W*, then AboveRow[*pos*] is set to *CurrFrame*[*Y*−1][*X*+*pos*]. Otherwise, *AboveRow*[*pos*] is set to *CurrFrame*[*Y*−1][*X*+*W*−1].
False	True	-	*AboveRow*[*pos*] is set to *CurrFrame*[*Y*][*X*−1].
False	False	-	AboveRow[*pos*] is set to 127, 511 or 2047, depending on the bit depth (8, 10 or 12).

Table 4 Left reference samples derivation

Above Available?	**Left Available?**	**Below Left Available?**	**Procedure for *LeftCol*[*pos*] for *pos* = 0...*w*+*h*−1**
True	True	True	If *pos*<2**H*, then *LeftCol*[*pos*] is set to *CurrFrame*[Y+*pos*][X−*pos*]. Otherwise, *LeftCol*[*pos*] is set to *CurrFrame*[Y+2**H*−1][*X*−1].
True	True	False	If *pos*<*H*, then *LeftCol*[*pos*] is set to *CurrFrame*[*Y*+*pos*][*X*−1]. Otherwise, *LeftCol*[*pos*] is set to *CurrFrame*[*Y*+*H*−1][*X*−1].
True	False	-	*LeftCol*[*pos*] is set to *CurrFrame*[*Y*−1][*X*].
False	False	-	*LeftCol*[*pos*] is set to 129, 513 or 2049, depending on the bit depth (8, 10 or 12).

After both arrays of reference samples are constructed, the prediction can be applied according to any of the modes described in the next section.

3.2 Intra Prediction Modes

As mentioned briefly in Section 2.5.2, AV1 supports 56 directional prediction modes to exploit more varieties of spatial redundancy in directional textures, and 11 non-directional modes to explore the spatial correlation of samples in smooth surfaces and the coherence of luminance and chrominance planes. It also supports two modes developed particularly for SCC. The intra prediction can be used for blocks of 64×64 samples or smaller, down to the minimum size of 4×4, with a few restrictions that will be highlighted in the following sections.

3.2.1 DC Mode

The DC prediction mode appears in many other video codecs. The AV1 version, however, is based on the DC mode used in VP9 (GRANGE; RIVAZ; HUNT, 2016), which considers the availability of the reference samples. Although the construction of the reference arrays already deals with the availability issue, the DC mode uses a different method.

As described in Figure 8, when both reference arrays are available, the DC mode produces a solid surface, that is, every sample is the arithmetic mean of the reference samples used. However, if only one of the reference arrays is available, then only the available array will be used in the arithmetic mean. Lastly, if none of the array references are available, the predicted sample is set according to the bit depth.

```
dc = 0
FOR i in 0 to H – 1:
  dc = dc + LeftCol[i]
END FOR
FOR j in 0 to W – 1:
  dc = dc + AboveRow[j]
END FOR
dc = dc + (W + H) / 2
dc = dc / (W + H)
FOR i in 0 to H – 1:
  FOR j in 0 to W – 1:
    Pred[i][j] = dc
  END FOR
END FOR
```

Figure 8 DC algorithm for when all reference samples are available.

Figure 9 illustrates four examples of DC predictions for a luminance block of size 4×4, one for each DC case. In the figure, the omitted reference samples are not used by the algorithm.

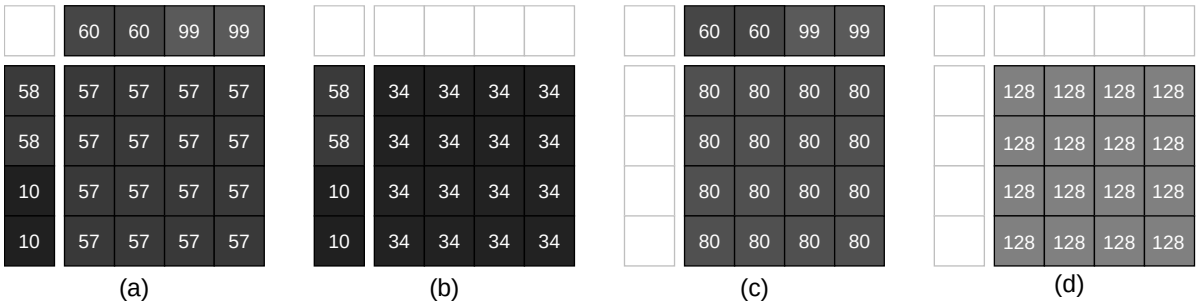

Figure 9 Example of DC predictions for a block of size 4×4 when (a) all samples are available, (b) only left samples are available, (c) only top samples are available, and (d) no samples are available.

3.2.2 Paeth Mode

The Paeth prediction mode is a novelty of the AV1, inspired by the TM mode from VP9 (MUKHERJEE *et al.*, 2013). This mode generates a fairly smooth surface by using only exact copies of reference samples.

As described in Figure 10, for each predicted sample, a comparison is done among the top-left, vertically aligned, and horizontally aligned reference samples. The algorithm then selects as predicted sample the reference sample that will result in a smoother gradient. Figure 11 illustrates an example of Paeth prediction for an 8×4 luminance block.

```
FOR i in 0 to H – 1:
  FOR j in 0 to W – 1:
    base = LeftCol[i] + AboveRow[j] – AboveRow[–1]
    pLeft    = ABS(base – LeftCol[i])
    pTop     = ABS(base – AboveRow[j])
    pTopLeft = ABS(base – AboveRow[–1])

    IF pLeft <= pTop AND pLeft <= pTopLeft:
      Pred[i][j] = LeftCol[i]
    ELSE IF pTop <= pTopLeft:
      Pred[i][j] = AboveRow[j]
    ELSE
      Pred[i][j] = AboveRow[-1]
    END IF
  END FOR
END FOR
```

Figure 10 Paeth algorithm.

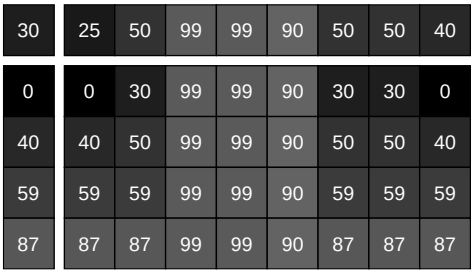

Figure 11 Example of a Paeth prediction.

3.2.3 Smooth, Smooth Vertical and Smooth Horizontal Modes

The Smooth family of prediction modes uses linear interpolation, with a precision of 1/256 of a sample, to generate very smooth surfaces. These modes were inspired by the Plane mode from H.264/AVC (WIEGAND *et al*., 2003) and the Planar mode from H.265/HEVC (SULLIVAN *et al*., 2012).

As described in Figure 12 and Figure 13, respectively, the Smooth Vertical mode interpolates samples using the *LeftCol*[*H*−1] sample and various samples from the *AboveRow* array, and the Smooth Horizontal mode interpolates samples using the *AboveRow*[*W*−1] samples and various samples from *LeftCol*. Each sample predicted by the generic mode is equivalent to the arithmetic mean of the samples predicted by the vertical and horizontal modes. In these algorithms, the *SmoothCoefficients* array stores a set of constants, which vary in size according to *W* (for the horizontal mode) or *H* (for the vertical mode), as listed in Table 5. Figure 9 illustrates examples of predictions using the Smooth family of modes.

```
1 FOR i in 0 to H – 1:
2   FOR j in 0 to W – 1:
3     a =  SmoothCoefficients[i] * AboveRow[j]
4     b = (256 – SmoothCoefficients[i]) * LeftCol[H–1]
5     Pred[i][j] = (a + b + 128) / 256
6   END FOR
7 END FOR
```

Figure 12 Smooth Vertical algorithm.

```
1 FOR i in 0 to H – 1:
2   FOR j in 0 to W – 1:
3     a =  SmoothCoefficients[j] * LeftCol[i]
4     b = (256 – SmoothCoefficients[j]) * AboveRow[W–1]
5     Pred[i][j] = (a + b + 128) / 256
6   END FOR
7 END FOR
```

Figure 13 Smooth Horizontal algorithm.

Table 5 Contents of the *SmoothCoefficients* array according to each size

Size	Coefficient Set
4	{255, 149, 85, 64}
8	{255, 197, 146, 105, 73, 50, 37, 32}
16	{255, 225, 196, 170, 145, 123, 102, 84, 68, 54, 43, 33, 26, 20, 17, 16}
32	{255, 240, 225, 210, 196, 182, 169, 157, 145, 133, 122, 111, 101, 92, 83, 74, 66, 59, 52, 45, 39, 34, 29, 25, 21, 17, 14, 12, 10, 9, 8, 8}
64	{255, 248, 240, 233, 225, 218, 210, 203, 196, 189, 182, 176, 169, 163, 156, 150, 144, 138, 133, 127, 121, 116, 111, 106, 101, 96, 91, 86, 82, 77, 73, 69, 65, 61, 57, 54, 50, 47, 44, 41, 38, 35, 32, 29, 27, 25, 22, 20, 18, 16, 15, 13, 12, 10, 9, 8, 7, 6, 6, 5, 5, 4, 4, 4}

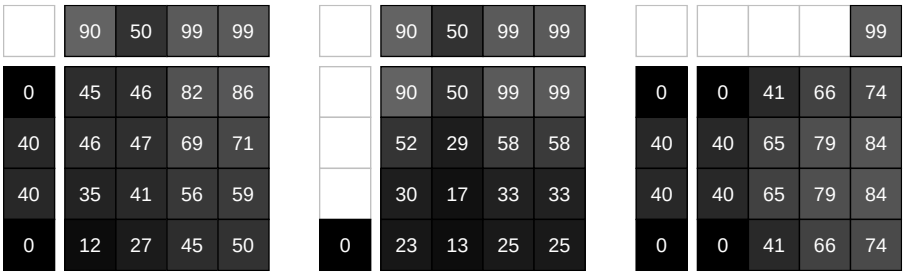

Figure 14 Example of predictions using the Smooth mode (left), Smooth Vertical mode (middle), and Smooth Horizontal mode (right).

3.2.4 Recursive-based-filtering (RBF) Modes

This group of modes is a novelty of the AV1 codec, available only for luminance blocks of size 32×32 or smaller and designed to mitigate the decaying spatial correlation as the distance between a predicted sample and the reference sample arrays increases.

The RBF algorithm divides the block *Pred*, regardless of its size, into smaller patches of size 4×2, and predicts these in raster order within *Pred*. The prediction of a single patch is done with a set of eight 7-tap filters, with different coefficients used for each one of the eight samples inside the patch. There are a total of five RBF modes, which share the same algorithm, but differ in the coefficients that the filters use.

Each patch uses as reference samples the seven adjacent samples. Therefore, some patches may use reference samples from the *AboveRow* (if it belongs to the first row of patches) and/or *LeftCol* (if it belongs to the first column of patches), or use samples from previously predicted 4×2 patches. Except for the first patch, every patch depends on the predicted samples from the ones predicted before, thus making it a recursive process with a considerable degree of data dependency.

The algorithm used in the prediction of a 4×2 patch is described in Figure 15, where:

- *B* is a 1-D array that stores predicted samples in raster order within the 4×2 patch;
- *L* is a 1-D array containing the reference samples, where *L*[0] is the reference located above and to the left of the patch, *L*[1] to *L*[4] are references located above, and *L*[5] and *L*[6] are references located at the left;
- *RbfMode* identifies the RBF mode (from 0 to 4);
- *IntraFilterTaps* is a 3-D constant array of coefficients, as listed in Table 6.

Figure 16 illustrates the four stages of an RBF prediction of an 8×4 block, highlighting in blue the reference samples used for the prediction of each 4×2 patch.

```
1  FOR i in 0 to 7:
2    aux = 0
3    FOR j in 0 to 6:
4      aux = aux + FilterCoefficients[RbfMode][i][j] * L[j]
5    END FOR
6    B[i] = (aux + 8) / 16
7  END FOR
```

Figure 15 RBF algorithm for a single 4×2 patch.

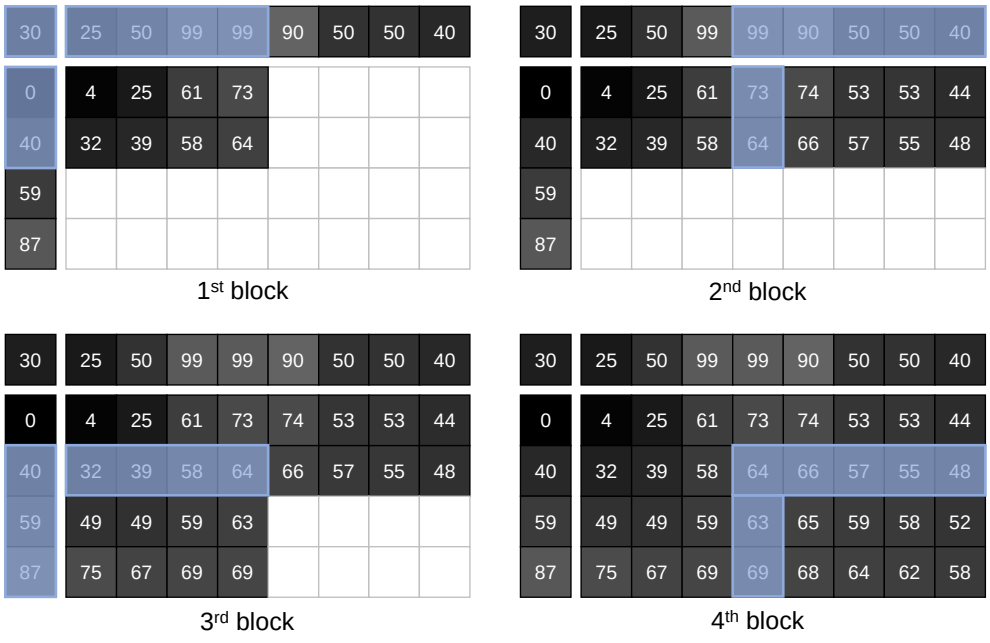

Figure 16 Example of the four stages of an RBF prediction of an 8×4 block.

Table 6 Filter coefficients stored in the *IntraFilterTaps* array

RbfMode	*B*	Coefficients
0	0	{−6, 10, 0, 0, 0, 12, 0}
	1	{−5, 2, 10, 0, 0, 9, 0}
	2	{−3, 1, 1, 10, 0, 7, 0}
	3	{−3, 1, 1, 2, 10, 5, 0}
	4	{−4, 6, 0, 0, 0, 2, 12}
	5	{−3, 2, 6, 0, 0, 2, 9}
	6	{−3, 2, 2, 6, 0, 2, 7}
	7	{−3, 1, 2, 2, 6, 3, 5}
1	0	{−10, 16, 0, 0, 0, 10, 0}
	1	{−6, 0, 16, 0, 0, 6, 0}
	2	{−4, 0, 0, 16, 0, 4, 0}
	3	{−2, 0, 0, 0, 16, 2, 0}
	4	{−10, 16, 0, 0, 0, 0, 10}
	5	{−6, 0, 16, 0, 0, 0, 6}
	6	{−4, 0, 0, 16, 0, 0, 4}
	7	{−2, 0, 0, 0, 16, 0, 2}
2	0	{−8, 8, 0, 0, 0, 16, 0}
	1	{−8, 0, 8, 0, 0, 16, 0}
	2	{−8, 0, 0, 8, 0, 16, 0}
	3	{−8, 0, 0, 0, 8, 16, 0}
	4	{−4, 4, 0, 0, 0, 0, 16}
	5	{−4, 0, 4, 0, 0, 0, 16}
	6	{−4, 0, 0, 4, 0, 0, 16}
	7	{−4, 0, 0, 0, 4, 0, 16}
3	0	{−2, 8, 0, 0, 0, 10, 0}
	1	{−1, 3, 8, 0, 0, 6, 0}
	2	{−1, 2, 3, 8, 0, 4, 0}
	3	{0, 1, 2, 3, 8, 2, 0}
	4	{−1, 4, 0, 0, 0, 3, 10}
	5	{−1, 3, 4, 0, 0, 4, 6}
	6	{−1, 2, 3, 4, 0, 4, 4}
	7	{−1, 2, 2, 3, 4, 3, 3}
4	0	{−12, 14, 0, 0, 0, 14, 0}
	1	{−10, 0, 14, 0, 0, 12, 0}
	2	{−9, 0, 0, 14, 0, 11, 0}
	3	{−8, 0, 0, 0, 14, 10, 0}
	4	{−10, 12, 0, 0, 0, 0, 14}
	5	{−9, 1, 12, 0, 0, 0, 12}
	6	{−8, 0, 0, 12, 0, 1, 11}
	7	{−7, 0, 0, 1, 12, 1, 9}

3.2.5 Chroma-from-Luma Mode

The AV1 CFL mode (TRUDEAU; EGGE; BARR, 2018), allowed only for chrominance blocks, is inspired by the proposals made by Chen *et al*. (2011) and Pu *et al*. (2013).

The stages of the CFL algorithm, as illustrated in Figure 17, are the following:

1. The reconstructed luminance block is subsampled to match the size of the chrominance block (if needed);
2. The arithmetic mean of all samples from the luminance block is calculated, and subtracted from each sample to generate a matrix called AC contributions;
3. Two matrices of scaled AC contributions are generated, one for each chrominance plane, by multiplying the AC contributions by the scaling variables *CflAlphaU* and *CflAlphaV*, to which the coder can assign values from 0 to 2 in steps of 0.125.
4. Finally, a regular DC prediction is done for each chrominance block, which are then added to the associated scaled AC contributions, to generate the final CFL prediction for both chrominance blocks.

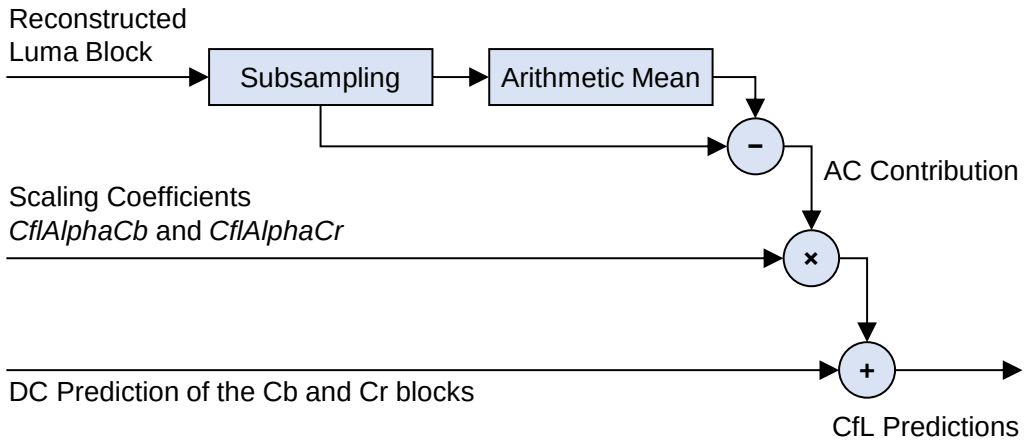

Figure 17 Stages of the CFL algorithm.

In this mode, the encoder can explore 17 possibilities for each of the two scaling variables. Considering that applying a scaling coefficient to the AC contribution requires one multiplication and one sum per sample, plus a way of testing the resulting predicted block for distortion, then the CFL mode can be considered significantly more complex than most modes if the coder is given the freedom of testing multiple possibilities.

3.2.6 Directional Prediction Modes

In AV1, there are eight directional prediction modes, called nominal modes, that were directly inherited from the VP9 (GRANGE; RIVAZ; HUNT, 2016). These modes are particularly efficient for predicting blocks that are located at heterogeneous parts of the image, e.g., where the edges of objects are located.

From each nominal angle, six new modes are defined with small variations of −9, −6, −3, +3, +6, and +9 degrees (known as Intra Angle Delta), resulting in a total of 56 directional modes. For blocks smaller than 8×8, however, only the eight nominal modes can be used.

Table 7 lists the eight nominal angles and the derived angles associated with these. Figure 18 illustrates all the angles supported.

Table 7 All the directional modes supported by AV1 and their associated angles

Nominal Mode	Nominal and Derived Angles						
D45_PRED	36	39	42	**45**	48	51	54
D67_PRED	58	61	64	**67**	70	73	76
V_PRED	81	84	87	**90**	93	96	99
D113_PRED	104	107	110	**113**	116	119	121
D135_PRED	126	129	132	**135**	138	141	144
D157_PRED	148	151	154	**157**	160	163	166
H_PRED	171	174	177	**180**	183	186	189
D203_PRED	194	197	200	**203**	206	209	211

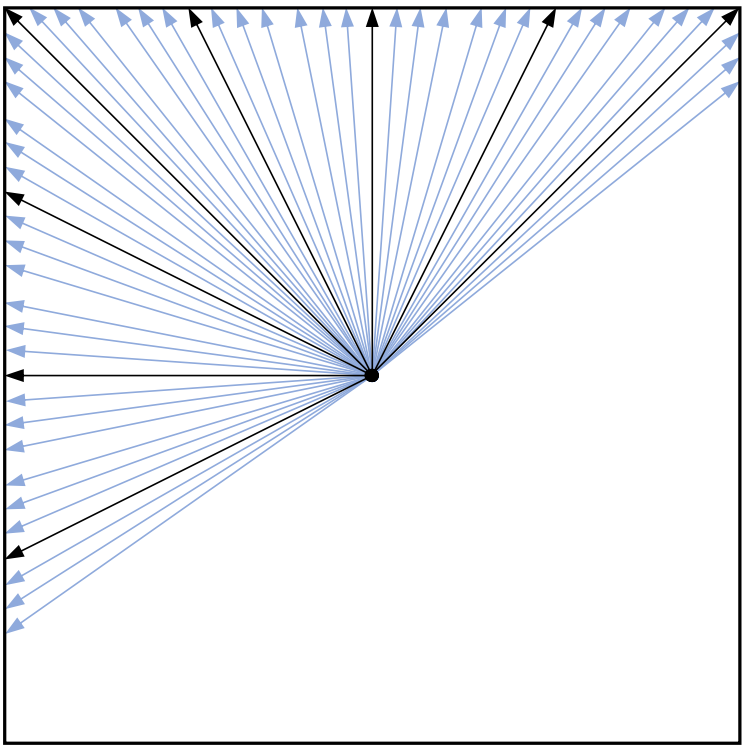

Figure 18 Nominal angles (black) and derived angles (blue) supported in AV1.

```
FOR i in 0 to H-1:
  FOR j in 0 to W-1:

    IF pAngle < 90:
      idx     = (i + 1) * dx
      base    = (idx >> (6 - upsampleAbove)) + (j << upsampleAbove)
      shift   = (idx << upsampleAbove) >> 1) & 31
      maxBase = (W+H-1) << upsampleAbove
      IF base < maxBase:
        Pred[i][j] = AboveRow[base] * (32 - shift) + AboveRow[base+1] * shift
        Pred[i][j] = (Pred[i][j] + 16) / 32
      ELSE:
        Pred[i][j] = AboveRow[maxBase]
      END IF

    ELSE IF pAngle > 90 AND pAngle < 180:
      idx  = (j << 6) - (i + 1) * dx
      base = idx >> (6 - upsampleAbove)
      IF base >= -(1 << upsampleAbove):
        shift = ((idx << upsampleAbove) >> 1) & 31
        Pred[i][j] = AboveRow[base] * (32 - shift) + AboveRow[base + 1] * shift
      ELSE:
        idx   = (i << 6) - (j + 1) * dy
        base  = idx >> (6 - upsampleLeft)
        shift = ((idx << upsampleLeft) >> 1) & 31
        Pred[i][j] = LeftCol[base] * (32 - shift) + LeftCol[base + 1] * shift
      END IF
      Pred[i][j] = (Pred[i][j] + 16) / 32

    ELSE IF pAngle > 180:
      idx     = (j + 1) * dy
      base    = (idx >> (6 - upsampleLeft)) + (i << upsampleLeft)
      shift   = (idx << upsampleLeft) >> 1) & 31
      Pred[i][j] = LeftCol[base] * (32 - shift) + LeftCol[base + 1] * shift
      Pred[i][j] = (Pred[i][j] + 16) / 32

    ELSE IF pAngle == 90:
      Pred[i][j] = AboveRow[j]

    ELSE IF pAngle == 180:
      Pred[i][j] = LeftCol[i]
    END IF

  END FOR
END FOR
```

Figure 19 Directional prediction algorithm for any prediction angle.

As described in Figure 19, AV1 uses a universal directional algorithm that links each predicted sample to a fractional position in the *AboveRow* or *LeftCol* arrays and generates the predicted sample using bilinear interpolation with a precision of 1/32 of a sample. In this algorithm: (i) the variables *upsampleAbove* and *upsampleLeft* indicate, respectively, if the *AboveRow* and *LeftCol* arrays were upscaled to twice their original size by an optional filtering process; (ii) the variable *pAngle* refers to the angle associated with the directional mode being processed; and (iii) the *dx* and *dy* variables

are partially calculated based in eq. (10) and eq. (11), respectively, and then translated as listed in Table 8.

$$refAngle_{dx} = \begin{cases} pAngle, & pAngle < 90 \\ 180 - pAngle, & 90 < pAngle < 180 \end{cases} \quad (10)$$

$$refAngle_{dy} = \begin{cases} pAngle - 90, & 90 < pAngle < 180 \\ 270 - pAngle, & 180 < pAngle \end{cases} \quad (11)$$

Table 8 Possible values for the *dx* and *dy* variables in the directional prediction

refAngle	***dx* or *dy***
3	1023
6	547
9	372
14	273
17	215
20	178
23	151
26	132
29	116
32	102
36	90
39	80
42	71
45	64
48	57
51	51
54	45
58	40
61	35
64	31
67	27
70	23
73	19
76	15
81	11
84	7
87	3

If the optional upscaling (mentioned above) and smoothing filtering tool is activated for the coding of a sequence, the reference sample arrays can be pre-processed in the scope of the directional prediction modes. A wide set of conditions must be met for determining if a specific type of filter will be used or not. These conditions are briefly explained below, and for a more detailed description, please refer to Rivaz and Haughton (2018).

The smoothing filtering process may apply a 3-tap filter to *AboveRow*[−1] and *LeftCol*[−1], depending on the values of *pAngle*, *W*, and *H*. It may also apply a 5-tap filter on the remaining samples of *AboveRow* and *LeftCol*, depending if the reference

samples stored in these arrays were originally available (see Section 3.1). There are three different sets of weights (filter strengths) for the 5-tap filter, the right one is selected based on values of *pAngle*, *W*, *H*, and if the blocks located above and to the left of *Pred* were previously predicted using one of the three Smooth modes.

The upscaling process for *AboveRow* can only happen for angles from 93 to 129 degrees inclusive, and for *LeftCol* only for angles from 183 to 211 inclusive. For both *AboveRow* and *LeftCol*, the upscaling can only happen if the sum of *W* and *H* is less than or equal to 16, or 8 if one of the blocks located above and to the left of *Pred* were previously predicted using one of the three Smooth modes. When invoked, the upscaling process doubles the size of one or both arrays of reference samples, filling each position between the original reference samples with a half-sample interpolated by a 4-tap filter.

Figure 20 illustrates the directional prediction of the eight nominal modes for a 4×4 block. In this figure, the hypothetical reference samples, common to all modes, are shown at the top.

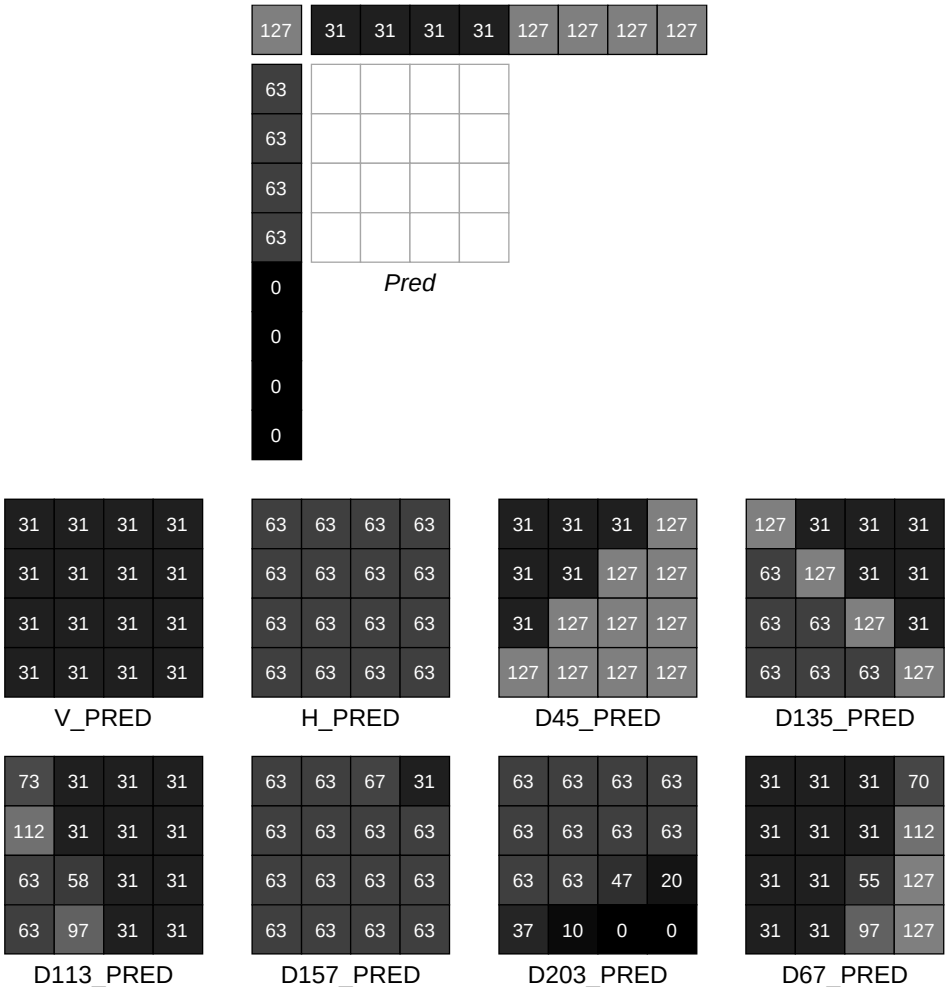

Figure 20 Example of predictions using the nominal directional modes.

3.2.7 Screen Content Prediction Modes

Xu and Liu (2022) define screen content as video content not captured by cameras, such as computer-generated text, graphics, and animation. AV1 supports two different prediction modes for SCC: The Color Palette mode and the Intra Block Copy mode.

The Color Palette prediction mode is effective when blocks can be approximated by a small number of unique colors. This mode is allowed only for blocks of size 8×8 or larger. The bitstream structure requires an array representing a color palette of two to eight colors and, also, a structure map, which is a 2-D array filled with the indexes of the colors (according to the palette) to be used in the prediction. The encoder can explore different sizes of palettes and different colors to optimize the resulting RD cost, making this mode more complex than most of the others, depending on the strategy adopted by the encoder. Figure 21 illustrates an 8×8 block predicted with a palette of five colors.

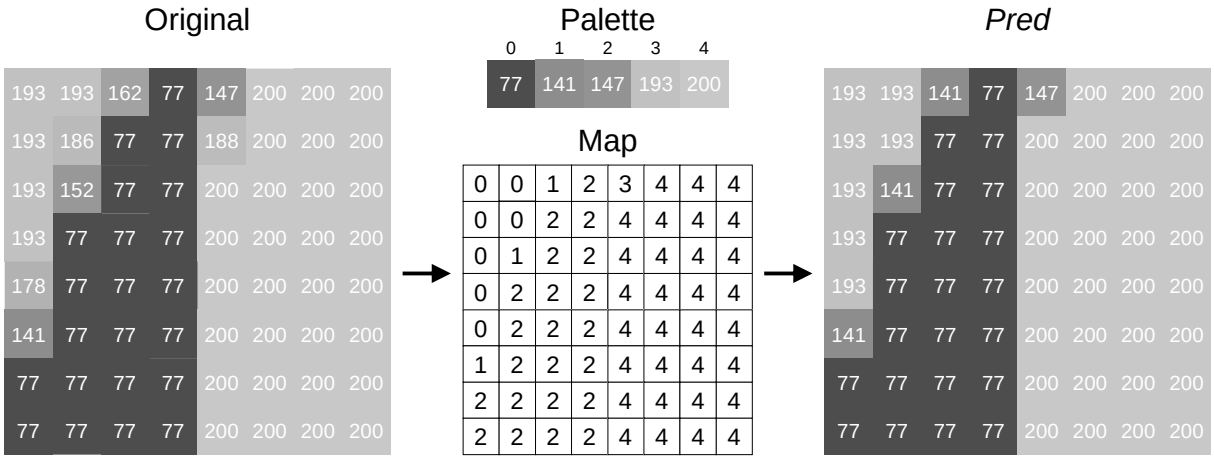

Figure 21 Example of a prediction using the Color Palette mode for an 8×8 block.

The Intra Block Copy mode allows the intra coder to refer back to previously coded samples in the same frame in the same way the inter-picture prediction refers back to previously coded samples in previously coded frames. It is very efficient in frames where many repeated textures and patterns are present. The location of the block used as reference is specified by a displacement vector in a way similar to motion vectors in motion compensation. Displacement vectors are limited to integer values for the luminance plane and may be fractional for chrominance planes, where bilinear filtering is used for interpolation. This mode is only allowed for intra-only frames, where conventional inter-picture coding cannot happen.

3.3 Compound Inter-intra Prediction

In AV1, the encoder is allowed to combine an intra predicted block with an inter predicted block to form a compound prediction. The intra prediction modes allowed are DC, Smooth, Directional Vertical, and Smooth Vertical, whereas the only inter prediction allowed is translational.

There are two forms of compound inter-intra prediction, one using wedge masks to split the block into two sections along with various oblique angles, and another using mode-specific masks. The use of masks is as defined in eq (12), where *PIntra and PInter* are respectively intra and inter predicted blocks, *M* is a mask in the form of a matrix with values in the [0, 1] range, and *C* is the resulting compound predicted block.

$$C_{x,y} = M_{x,y} \times PIntra_{x,y} + \left(1 - M_{x,y}\right) \times PInter_{x,y} \quad (12)$$

Figure 22 illustrates the 16 wedge masks supported. In each mask, most of the $M_{x,y}$ values are either 0 or 1, except near the transition edge, where there is a gradual change, with 0.5 values at the actual edge.

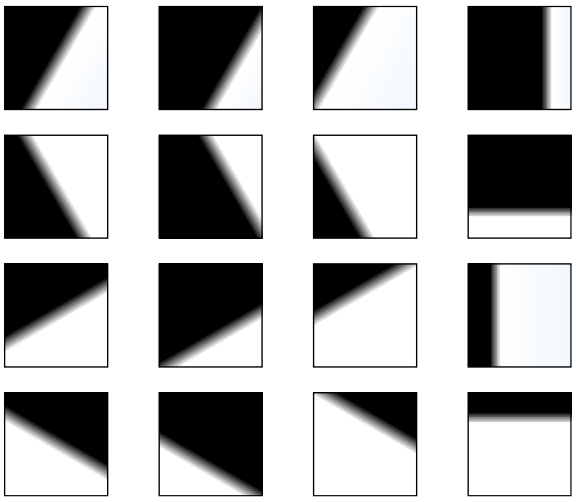

Figure 22 Compound inter-intra prediction wedge masks.

Figure 23 (HAN *et al*., 2021) illustrates the mode-specific masks. In this figure, it can be observed that for the Directional Vertical, Directional Horizontal, and Smooth modes, the mask weights are higher (i.e., prioritize the intra sample instead of the inter) for positions that are closer to the reference samples used by the mode in question. As the distance from the reference samples increase, the accuracy of the intra prediction decreases, and the mask weights prioritize the inter predicted samples instead.

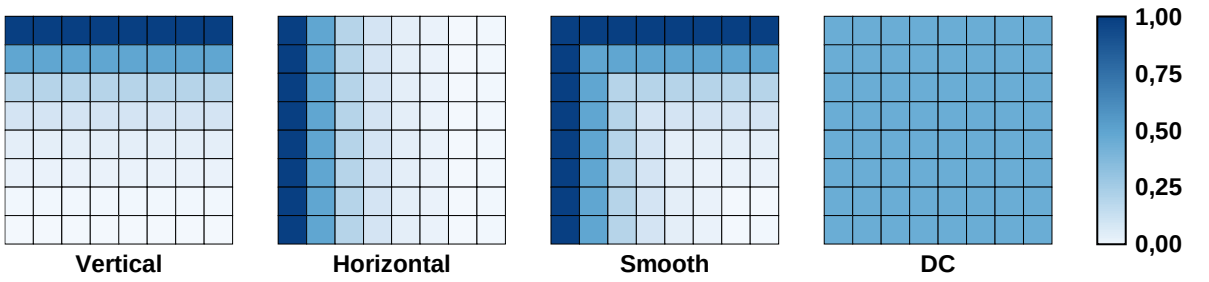

Figure 23 Compound inter-intra prediction mode-specific masks.

As described above, AV1 supports numerous possible compound inter-intra prediction. Therefore, an encoder that evaluates every possibility in the RDO loop may achieve satisfactory compression efficiency, at cost of a significant complexity added to the prediction of a single block.

4 RELATED WORKS

Firstly, this chapter presents a review of the related works that propose system-level solutions for computational effort reduction in an AV1 encoder and works that propose dedicated hardware designs for modules of an AV1 encoder and decoder. Works that were published as part of this Ph.D. project were also included in the discussion due to the small number of related works found in the literature. Finally, this chapter presents and discusses research opportunities that are aligned with the thesis of this Ph.D. project.

4.1 AV1-related Algorithmic Optimization

Most of the published papers that propose ways of reducing the complexity in the current-generation encoders, such as AV1, address the topic of block partitioning. Optimizations in this area generally consist of early termination of the block partitioning exploration, thus completely avoiding the cost of processing partitions from deeper levels of a tree branch.

A common kind of optimization is to reduce the coding search space by exploiting certain features of a codec based on previous observations. However, different methods based on neural networks also exist.

Since the release of the AV1 bitstream specification, a few works have been published covering heuristic-based and machine-learning solutions for the block partitioning stage, intra prediction stage, and inter prediction stage of the AV1 encoder. It is important to mention that system-level solutions for different stages of the encoder can and should be used together to allow for better results. Table 9 lists all these works, and the following paragraphs present them with further details.

Table 9 Summary of AV1-related algorithmic optimization

Work	Stage	Software Version	Time Saving (%)	BD-BR (%)
Chiang *et al*. (2019)	Block Partitioning (Enc.)	-	64.14	0.61
Chen *et al*. (2019)	Block Partitioning (Enc.)	1.0.0	35.7	0.61
Guo *et al*. (2018a)	Block Partitioning (Enc.)	-	33.4	0.14
Guo *et al*. (2018b)	Block Partitioning (Enc.)	-	34.7	2.12
Kim *et al*. (2019)	Inter (Enc.)	1.0.0	57.7	1.59
Jeong *et al*. (2019a)	Intra (Enc.)	-	8.67	0.04
Jeong *et al*. (2019b)	Intra (Enc.)	-	15.86	0.44
Corrêa *et al*. (2022a) *	Intra (Enc.)	2.0.0	15.36	0.60
Corrêa *et al*. (2022b) *	Intra (Enc.)	2.0.0	22.56	1.26

* Works developed as part this Ph.D. project.

Chiang, Han, and Xu (2019) propose a two-pass method for evaluating the AV1 block partitioning tree of an SB: (i) the first pass uses a binary tree that allows only the *NONE* and *SPLIT* partition modes, instead of the usual 10-way partition tree (see Figure 3), which is used only to estimate the probable depth of the tree branches, and (ii) the second pass tests all others partition modes (except the *SPLIT*), on tree nodes that were selected as *NONE* in the first pass. Each partition tested usually needs to go through the exhaustive RDO process, but the authors also propose a two-pass method for evaluating the RD cost of each partition, which consists of a simplified RD evaluation of the residual information of each prediction mode in the first pass, and the usual RDO evaluation for a subset of the best candidates selected in the first pass.

Chen *et al.* (2019) propose a conditional Bayesian inference model to perform early termination in the AV1 block partitioning tree exploration, based on how the same input was previously encoded by an HEVC encoder. Although the AV1 specifies a wider set of block sizes than HEVC, the authors also propose a prior probability estimation for the depth of AV1 partition trees that can be updated during the encoding.

Likewise, Guo *et al*. (2018a) also propose a Bayesian approach for early termination of the AV1 block partitioning tree exploration, but in the context of multi-rate video encoding, where the encoding of a video sequence is done by a reference instance of the encoder, while *N* other instances encode the same input simultaneously, each consulting the reference instance to accelerate their own RD

decisions. In this work, however, the authors wrongly affirmed that in AV1 superblocks start at the size of 64×64 and that blocks can only be divided into four or two parts.

Guo, Han, and Wen (2018b) propose an early termination scheme for the AV1 block partitioning tree exploration, in the context of multiple resolutions encoding. In this solution, high-resolution encoding is accelerated by referring to decisions made in low-resolution encoding. In this work, however, the authors once again wrongly affirmed that the AV1 block partitioning is based on a 4-way partition tree.

Kim *et al*. (2019) propose a machine learning-based solution to accelerate the AV1 inter prediction stage. The authors observed that not all video sequences benefit from compound prediction modes. Therefore, in this solution, a decision tree trained based on seven features extracted while encoding each block is used to decide whether to skip compound prediction modes or not.

Jeong, Gankhuyag, and Kim (2019a) propose an optimization to the reference software mode decision by adding an adaptive margin for early termination based on the accuracy of the mode decision when compared to the RDO.

Jeong, Gankhuyag, and Kim (2019b) propose a fast mode decision applied to blocks of chrominance samples only, based on how the same block of luminance was coded.

Corrêa *et al.* (2022a) propose a mode-adaptive distortion metric subsampling technique to reduce the cost of the SAD/SSE operations of the intra prediction stage, allowing for faster encoding times on software encoders and for lower area and lower power dissipation on hardware encoders. This work is part of this Ph.D. project and is further described in Chapter 5.

Corrêa *et al.* (2022b) propose a fast decision algorithm for the AV1 intra prediction, inspired by the direction detection algorithm used on the CDEF of the same codec. The main objective is to reduce the number of intra candidates with a low-cost heuristic, thus allowing a faster prediction time in software and also allowing a low-area and low-power intra prediction hardware design. This work is part of this Ph.D. project and is also further described in Chapter 5.

4.2 AV1-related Hardware Designs

Since the release of AV1, several hardware-related works have been published covering three areas: (i) intra-frame prediction, (ii) inter-frame prediction, (iii) in-loop filtering, and (iv) entropy coding. Most of these works were developed in the same

research group in which this project was carried out. It is important to mention that some of these works target the encoder, whereas others target the decoder, and the ones that target the former tend to be more complex due to the decisions that an encoder must take. Table 10 summarizes all these works, whereas the next sections present them in further detail.

Table 10 Summary of AV1-related hardware designs

Work	Stage	Technology	Gate Count (2-NAND)	Freq. (MHz)	Power (mW)	Throughput
Corrêa *et al*. (2019a) *	Intra (Enc.)	TSMC 40nm	247.28	315	268.36	4K@120fps
Corrêa *et al*. (2019b) *	Intra (Enc.)	TSMC 40nm	109.57	648	16.1	4K@30fps
Corrêa *et al*. (2020a) *	Intra (Enc.)	TSMC 40nm	455.8	1,296	40.92	4K@60fps
Corrêa *et al*. (2020b) *	Intra (Enc.)	TSMC 40nm	128.5	648	65.5	4K@30fps
Neto *et al*. (2020) *	Intra (Enc.)	TSMC 40nm	821.835	1,902	1,613	4K@60fps
Neto *et al*. (2021a) *	Intra (Enc.)	TSMC 40nm	584.845	1,296	4,110	4K@60fps
Neto *et al*. (2021b) *	Intra (Enc.)	TSMC 40nm	2794	1,902	82.76 631.76	4K@60fps
Neto *et al*. (2022) *	Intra (Enc.)	TSMC 40nm	2504.715	1,902	1,182 9,468	4K@60fps
Goebel *et al*. (2019)	Intra (Dec.)	TSMC 40nm	89.39	132.1	7.96	4K@60fps
Domanski *et al*. (2019)	Inter (Dec.)	TSMC 40nm	141.1	279.9	81.31	8K@30fps
Domanski *et al*. (2021)	Inter (Dec.)	TSMC 40nm	72.64	686	26.79	8K@30fps
Freitas *et al*. (2020)	Inter (Dec.)	TSMC 40nm	106.17 270.44	448.43 344.83	56.37 240.75	8K@30fps 8K@120fps
Freitas *et al*. (2021)	Inter (Dec.)	ST 65nm	104.3	441	63.14	8K@120fps
Freitas *et al*. (2022)	Inter (Dec.)	TSMC 40nm	324.79	1,000	51.15	8K@60fps
Zummach *et al*. (2020a)	Filtering (Dec.)	TSMC 40nm	369	23	65	4K@60fps
Zummach *et al*. (2020b)	Filtering (Dec.)	TSMC 40nm	185	93	43	4K@60fps
Zummach *et al*. (2020c)	Filtering (Dec.)	TSMC 40nm	39.35	16.2	3.96	4K@60fps
Palau *et al*. (2022a)	Filtering (Dec.)	TSMC 40nm	177.58	212.86	120.21	4K@60fps
Palau *et al*. (2022b)	Filtering (Dec.)	TSMC 40nm	37.78	207.03	26.36	4K@60fps
Bitencourt *et al*. (2022)	Entropy (Enc.)	ST 65nm	11.7K 11.2K	581 563	7.801 6.166	8K@120fps
Gomes *et al*. (2021)	Entropy (Dec.)	ST 65nm	34.3K	467	-	8K@60fps

* Works developed as part this Ph.D. project.

4.2.1 Designs for Intra Prediction

Corrêa *et al*. (2019a; 2019b; 2020a; 2020b) and Neto *et al*. (2020; 2021a; 2021b; 2022) presented architectures for the intra prediction module at the encoder. These works are part of this Ph.D. project, and their results lead to the final designs described in Chapter 6. On the other hand, Goebel *et al.* (2019) presented an intra prediction architecture for the decoder side. All these architectures have in common the support for every possible block partition allowed by the AV1 specification.

In the work by Corrêa *et al*. (2019a), a non-directional intra prediction module for the encoder side, limited to a single prediction mode (Paeth), and able to reach high throughput was presented. Massive parallelism is used to allow the processing of a whole 32x32 block (or any smaller block) in a single clock cycle. A throughput of 30 frames per second (fps) for UHD 8K videos was reported.

In Corrêa *et al*. (2019b), a non-directional intra prediction module for the encoder side, limited to four intra prediction modes, was presented. The authors optimized all multiplication blocks to keep the area and power within feasible limits. The parallelism strategy allowed the processing of one block row/column per clock cycle. Thus, the number of cycles depends on the width or height of the block, whichever is the largest. Similarly, in Corrêa *et al*. (2020a), a non-directional intra prediction module for the encoder side, capable of processing ten non-directional modes, was presented. A throughput of 30 fps for UHD 4K videos was reported for both architectures.

In Corrêa *et al*. (2020b) and Neto *et al*. (2020), directional intra prediction designs for the encoder side that share many similarities were described. Both designs support all 56 directional prediction modes, however, only the design proposed by Neto *et al*. (2020) gives support to the four smoothing filtering processes of reference samples and the upscaling of reference samples. All 56 prediction modes are processed in parallel in both works, one row/column per clock cycle. Although the number of prediction modes being processed in parallel is quite large, a significant amount of redundant operations is reused in Corrêa *et al*. (2020b) because all predicted blocks share the same reference samples as input due to the lack of the smoothing filtering step. This, however, could not be done with the same degree of efficiency in the architecture proposed by Neto *et al*. (2020), because for each of the 56 prediction modes, the encoder can use a different configuration of smoothing

filtering and upscaling of the reference samples. A throughput of 60 fps for UHD 4K was reported for both architectures.

In Neto *et al*. (2021a), a study regarding the redundancy of operations in the directional intra prediction filters was presented, together with an architecture with operation sharing amongst different filters. The authors reported a decrease in power dissipation and gate count with the proposed optimization when compared to a naive solution, and also reported a throughput of 60 fps for UHD 4K.

In Neto *et al*. (2021b; 2022), two different solutions for directional intra prediction were presented, both capable of operating in a high-quality or low-power setting. These architectures support the smooth filtering and upscaling process, although the low-power setting disables these features to reduce energy consumption, at cost of compression efficiency losses. A throughput of 60 fps for UHD 4K was reported for both architectures.

In Goebel *et al.* (2019), a non-directional intra prediction module for the decoder limited to the DC and CFL prediction modes was presented. The sample-level parallelism of the architecture allowed the processing of any block size as subblocks of size 4x4 (16 samples per cycle). In the decoding process, each block must be predicted only once using the prediction mode signaled in the bitstream, hence the CFL unit of this design is only used for CFL-coded blocks, but the DC unit is used for both modes because the DC algorithm is one of the steps of the CFL prediction. The authors reported a throughput of 60 fps for UHD 4K.

4.2.2 Designs for Inter Prediction

Domanski *et al.* (2019; 2021) and Freitas *et al.* (2020; 2021; 2022) presented architectures for the subpixel interpolation filter present in the inter prediction module of the decoder.

In Domanski *et al.* (2019), the sample-level parallelism of the architecture allows the processing of any block size as subblocks of size 4x4 (16 samples per cycle), but since it is a decoder design, only one of the many supported filters is used per predicted block (the one signaled in the bitstream). In Domanski *et al.* (2021), a similar architecture is presented, but making use of approximate computing to generate more hardware-friendly filter coefficients. The authors reported a throughput of 30 fps for UHD 8K for both architectures and a power reduction of 80% in the approximate solution when compared to its precise counterpart.

In Freitas *et al.* (2020), an architecture for a subset of the interpolation filters, called Regular, is presented. Similarly, in Freitas *et al.* (2021), an architecture for a subset of the interpolation filters, called Sharp, is presented. In both designs, the level of parallelism is configurable, ranging from 4 to 128 samples per cycle, and because of this, the authors reported a very high throughput of 120 fps for UHD 8K. Finally, in Freitas *et al.* (2022), an architecture for the complete set of filters is presented, for which the authors reported a throughput of 60 fps for UHD 8K.

4.2.3 Designs for In-loop Filtering

Zummach *et al.* (2020a; 2020b; 2020c) and Palau *et al*. (2022a; 2022b) presented architectures for the CDEF, DBF, and LRF in-loop filters for the AV1 decoder.

In Zummach *et al.* (2020a), a CDEF architecture for the decoder was presented. The CDEF process is applied to each area of size 8×8 within a frame, and the architecture was designed with enough parallelism to process an 8×8 area every three clock cycles. The architecture is composed of a direction search unit, which classifies the input texture with one of eight directions, and a filtering core unit, which filters the input texture using 64 filter kernels based on the detected direction. In Zummach *et al.* (2020b), another version of the architecture with lower parallelism was presented, this one capable of processing an area of 8×1 of the frame every three cycles. The authors reported a throughput of 60 fps for UHD 4K for both designs.

In Zummach *et al.* (2020c), a DBF architecture for the decoder was presented. The architecture implements a parallelism of 56 samples per cycle, which is enough to allow a very low frequency when processing high-resolution videos. The authors reported a throughput of 60 fps for UHD 4K.

In Palau *et al*. (2022a), a Dual Self-Guided Filter (DSGF) architecture is presented, whereas in In Palau *et al.* (2022b), an architecture for the Separable Symmetric Normalized Wiener Filter (SSNWF) is presented. Together, both architectures form an LRF module, which is used for denoising and/or edge enhancement. The presented hardware designs target the decoder and, according to the authors, can process at 60 fps for UHD 4K videos.

4.2.4 Designs for Entropy Coding

Bitencourt, Ramos, and Bampi (2021; 2022) presented two architectures for the arithmetic encoder of the AV1 entropy encoding stage, one being a straightforward design pipelined for high throughput, and another being an optimized version of the former with low-power techniques, such as clock gating and operand isolation. The authors reported a 20.9% power reduction in the optimized version, and that both designs can process at 120 fps for UHD 8K videos.

Gomes and Ramos (2021) presented one architecture for the arithmetic decoder of the AV1 entropy decoder. This design is divided into two main modules, one to decode symbols from a binary alphabet, and another to decode symbols from the multi-symbol alphabet. The authors report a throughput of one symbol of any alphabet per cycle, resulting in an estimated performance of 60 fps for UHD 8K videos.

4.3 Research Opportunities

In Section 4.1, it was shown that there are a few algorithmic-based solutions for computational effort reduction of an AV1 encoder. If the works of Corrêa *et al.* (2022a; 2022b) are not considered, which are both part of this Ph.D. project, then only two works focus on optimizing intra prediction, and all others target the task of simplifying the block partitioning decision, which is a global decision that affects the prediction stage of a given block, as well as all encoding stages that follow. However, there are many other decisions that an encoder must take locally in other stages, such as inter and intra prediction stages, transform coding, and entropy coding. These local decisions allow for optimization and, thus, this subject offer research opportunity.

The works of Corrêa *et al.* (2022a; 2022b) propose optimizations for the intra prediction stage of an AV1 encoder. These optimizations do not compete with most of the related works listed in this chapter, but instead, it can be paired with those to form a highly optimized encoder. The proposed solutions are fully described in Chapter 5.

In Section 4.2, several hardware architectures were described for the AV1 encoder and decoder sides. However, all of these works only propose implementations of the original algorithms defined in the AV1 specification. Although this kind of work also contributes to its field of research, there is a clear absence of works that propose hardware designs for optimized algorithms.

To fill this gap, the work of Corrêa *et al.* (2022b), which is part of this Ph.D. project, proposes a fast mode decision algorithm for the intra prediction and also proposes a hardware design for it. Designs following this strategy are described in Chapter 6.

5 HEURISTIC-BASED ALGORITHMS FOR AV1 INTRA-PICTURE PREDICTION

This chapter presents heuristic-based algorithms for the intra prediction stage of the AV1 encoder, which were published in Corrêa *et al.* (2022a; 2022b).

5.1 Algorithm 1: Texture-based Fast Mode Decision (TbFMD)

The main purpose of a fast mode decision algorithm is to mitigate the large combinatorial space of block sizes and prediction modes allowed in AV1.

The proposed heuristic-based method (CORRÊA *et al*., 2022b) operates in two steps: (i) detection of the dominant direction of the input block texture, and (ii) creation of a list with a reduced number of prediction modes (RD-list), according to the dominant direction detected.

5.1.1 Direction Detection Step

The direction detection step proposed in this work is based on a solution first described in the Daala codec (VALIN *et al*., 2016), and later added as part of the AV1 CDEF in-loop filtering stage (MIDTSKOGEN; VALIN, 2018). This algorithm has been proven efficient at detecting texture direction for the objectives of the in-loop filtering stage of the codec and, since this information can be valuable for an intra-frame prediction fast mode decision heuristic, it was adapted to define which prediction modes are more likely to be good candidates for an input block.

In the CDEF, the direction detection algorithm finds, for each 8×8 input block, the direction *d* that matches the input block by comparing it to eight Perfectly Directional Blocks (PDB), one for each of eight directions *d*. The PDBs are matrices (blocks) where all samples belonging to line *k* have the same value, which is the average of the samples from that line. The SSE distortion metric is the one used in this comparison.

At the in-loop filtering stage, the direction detection as used in the AV1 algorithm must have high accuracy, because errors at this stage can produce visible artifacts in the resulting image. On the other hand, a fast mode decision algorithm at the intra prediction stage has much more flexibility to explore a trade-off between accuracy and the use of computational resources. With the main objective of developing a fast

decision algorithm that is hardware-friendly, the direction detection algorithm from CDEF was adapted to work with PDBs of a reduced size of 4×4 samples, significantly reducing the number of operations required to compute the SSE between an input block and the eight PDB_d, although also reducing the accuracy of the algorithm. Because the PDB_d has a fixed 4×4 size, all sizes of input blocks must also be subsampled to 4×4 before undergoing the direction detection step, but their original size is used in the prediction itself. Figure 24 shows the eight PDB_d, and also how samples are distributed in lines *k*.

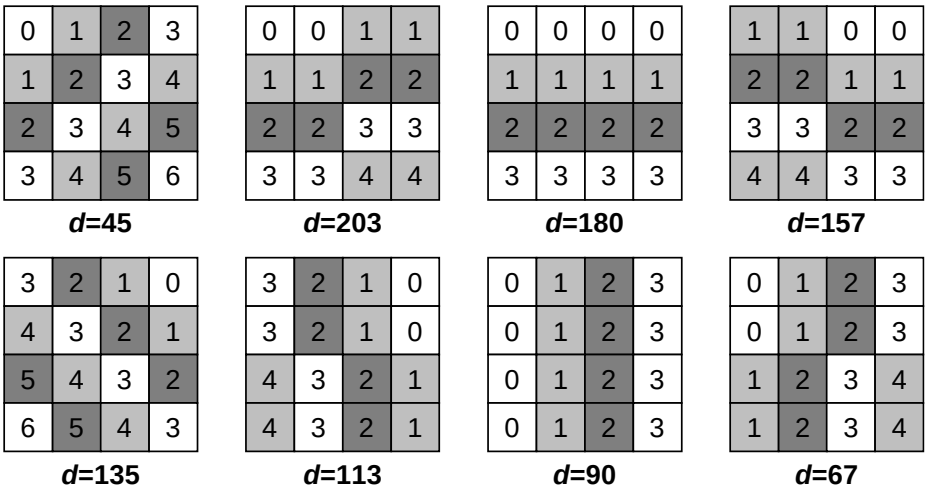

Figure 24 PDBs adapted to the reduced size of 4×4. Each direction *d* is named after an AV1 intra mode related to the same angle. The numbers inside each square identify to which line *k* a sample belongs.

As first proposed by Midtskogen and Valin (2018), for each of the eight directions *d*, the average $\mu_{d,k}$ of the samples of a line *k* is defined as in eq. (13), where x_p is the value of sample *p* in the input block, $P_{d,k}$ is the set of samples in line *k*, and $N_{d,k}$ is the number of samples in $P_{d,k}$. In the adapted algorithm proposed in this thesis, *N* ranges from 1 to 4.

$$\mu_{d,k} = \frac{1}{N_{d,k}} \sum_{p \in P_{d,k}} x_p \tag{13}$$

The error between the input block and the PDB_d, given by the SSE metric, is defined as in eq. (14).

$$E_d^2 = \sum_k \left(\sum_{p \in P_{d,k}} \left(x_p - \mu_{d,k} \right)^2 \right) \tag{14}$$

Substituting eq. (13) into eq. (14) leads to the simplified error, defined in eq. (15).

$$E_d^2 = \sum_p x_p^2 - \sum_k \frac{1}{N_{d,k}} \left(\sum_{p \in P_{d,k}} x_p \right)^2 \tag{15}$$

Finally, because the first term of eq. (15) refers only to samples from the input block, it is not influenced by the variable *d* in any way. Hence, this constant can be removed from the equation and, instead of looking for the lowest error, the algorithm must look for the highest sum S_d, expressed as in eq. (16). The dominant direction is given by the highest of the eight S_d values.

$$S_d = \sum_k \frac{1}{N_{d,k}} \left(\sum_{p \in P_{d,k}} x_p \right)^2 \tag{16}$$

Figure 25 and Figure 26 together illustrate the direction detection step. Figure 25 shows an example of an 8×8 input block being subsampled to 4×4, and Figure 26 shows the PDBs for this subsampled block and also the S_d values for these PDBs as if calculated using eq. (16). According to this example, the dominant direction of the input block texture follows a 45-degree angle, because S_{45} is the highest among all S_d values, meaning that the PDB_{45} is the most similar to the subsampled input block. In the figures, the color used for each sample, as well as their numerical label, represent their 8-bit luminance value.

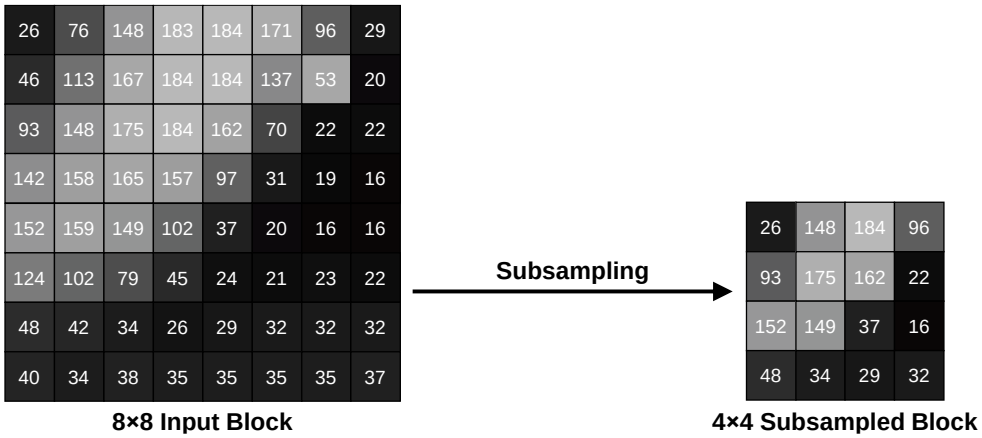

Figure 25 Example of an 8×8 to 4x4 subsampling.

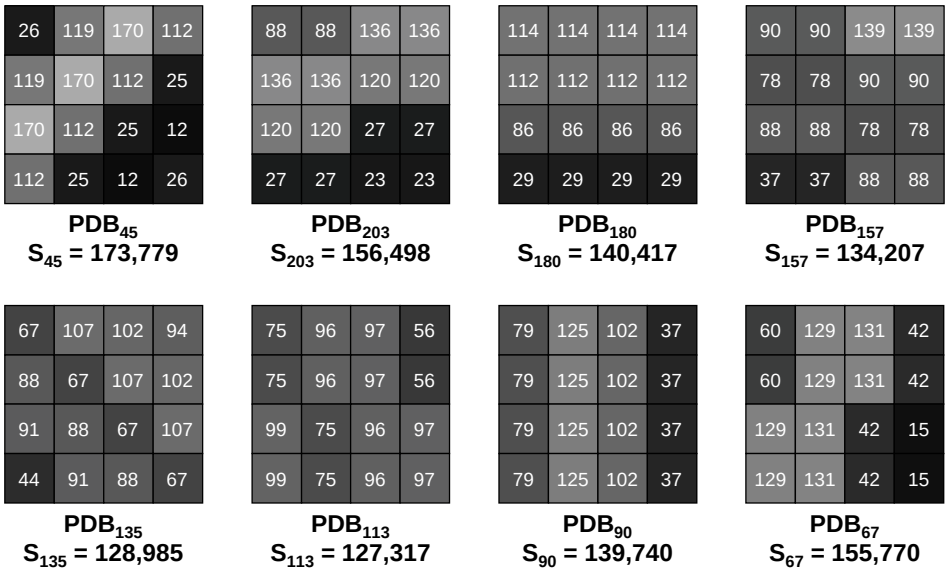

Figure 26 Example of PDBs and calculation of the dominant direction, based on the input block of the previous figure.

5.1.2 RD-list Creation Step

For a given input block, the best decision can only be achieved by passing all the possible prediction candidates through the complete encoding loop and, then, by evaluating their RD costs to select the best prediction mode. However, since the RDO is a bottleneck of the encoding process due to its prohibitive computational effort, a fast mode decision is needed to mitigate this problem.

The fast mode decision algorithm proposed creates reduced a RD-list (list of promising candidates) to be sent to the RDO by using only information from the direction detection step, i.e., based on a heuristic, prediction modes are discarded without ever being evaluated by the exhaustive RDO task. For a given input block, the decision algorithm works as follows:

1. The algorithm classifies the input block as being a smooth or a sharp texture by subtracting the S_d of the dominant direction ($S_{dominant\_d}$) from the S_d of its orthogonal direction ($S_{orthogonal\_d}$) and then performing a right shift by a threshold *Thr*, as defined in eq. (17). If the contrast *C* results in a positive value, the block is classified as a sharp texture, otherwise, smooth.

2. If the block is smooth, then a very small RD-list is created from the ND set, which is composed of non-directional modes only.

3. Otherwise, if the block is sharp, then a reduced RD-list is created from the union of the ND set and the directional mode that follows the exact detected direction. Then, it is also verified which of the two directions adjacent to the dominant direction has the highest S_d value. The three closest angle variations that follow the best adjacent direction are also appended to the final RD-list.

$$C_{dominant\_d} = \left(S_{dominant\_d} - S_{orthogonal\_d}\right) \gg Thr \quad (17)$$

Table 11 shows the reduced RD-list created for sharp blocks. It can be observed that the proposed fast mode decision reduces the set of possible directional prediction modes from 56 to just four, in the worst case, for sharp blocks. As previously discussed, no directional mode is evaluated when a block is classified as smooth.

Table 11 RD-list created for sharp blocks according to the dominant and best adjacent directions

Dominant direction (*d*)	Best adjacent direction (*d*)	Reduced RD-list
45	67	ND ∪ {45} ∪ {36, 39, 42}
	203	ND ∪ {45} ∪ {48, 51, 54}
203	45	ND ∪ {203} ∪ {206, 209, 211}
	180	ND ∪ {203} ∪ {194, 197, 200}
180	203	ND ∪ {180} ∪ {183, 186, 189}
	157	ND ∪ {180} ∪ {171, 174, 177}
157	180	ND ∪ {157} ∪ {160, 163, 166}
	135	ND ∪ {157} ∪ {148, 151, 154}
135	157	ND ∪ {135} ∪ {138, 141, 144}
	113	ND ∪ {135} ∪ {126, 129, 132}
113	135	ND ∪ {113} ∪ {116, 119, 122}
	90	ND ∪ {113} ∪ {104, 107, 110}
90	113	ND ∪ {90} ∪ {93, 96, 99}
	67	ND ∪ {90} ∪ {81, 84, 87}
67	90	ND ∪ {67} ∪ {70, 73, 76}
	45	ND ∪ {67} ∪ {58, 61, 64}

Figure 27 shows two examples of RD-lists. On the left side, an RD-list composed of the prediction modes ND ∪ {99, 96, 93, 90} is presented, created from a dominant direction 90, and the best adjacent direction 113. On the right side, an RD-list ND ∪ {90, 87, 84, 81} is presented, created from the same dominant direction, but the opposite adjacent direction. For a full picture illustrating all prediction angles, refer to Figure 18.

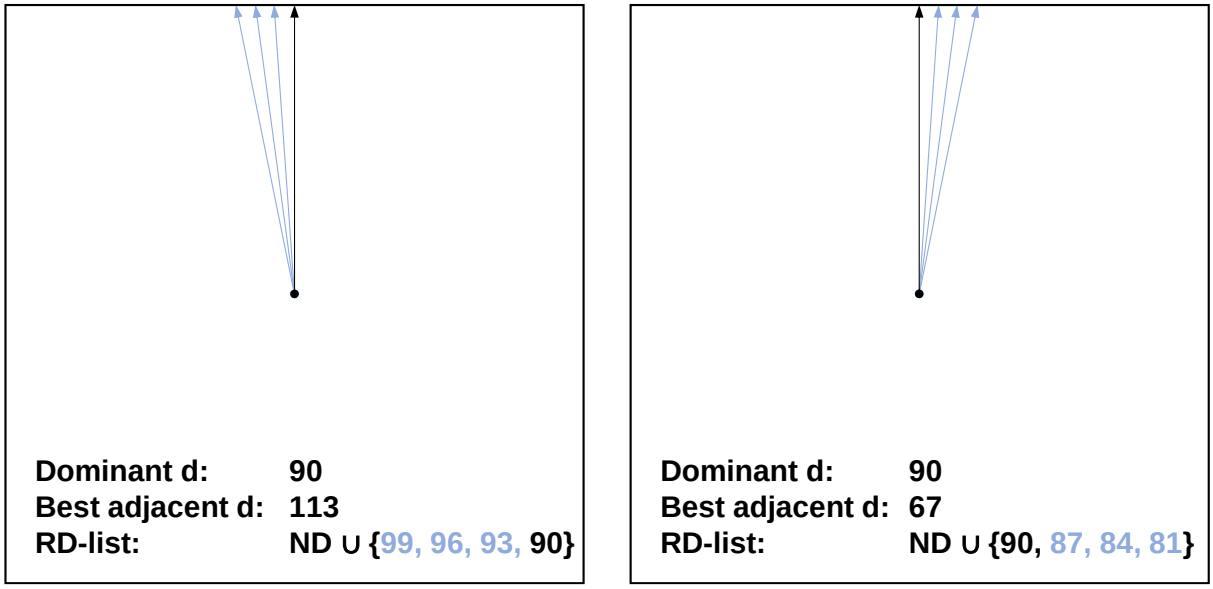

Figure 27 Examples of RD-lists created from dominant direction 90, and best adjacent direction 113 (left side) and 67 (right side).

5.2 Algorithm 2: Mode-adaptive Subsampling in Block Matching (MaSBM)

In the AV1 intra prediction, a single 64×64 superblock (SB) can be partitioned into 1,869 different intra subblocks according to a 10-way partition tree. As explained in Section 3.2, each block can be predicted by many different intra modes, resulting in a high number of predicted candidates that must be evaluated by the RDO. One way to mitigate this problem is to reduce the number of prediction modes computed, as proposed in Section 5.1, and another fairly common way is to evaluate the predicted blocks locally with a distortion metric, such as the SSE, and to create an RD-list composed of the N candidates with the lowest distortion. Selecting candidates locally using the SSE is a heuristic method, because it only considers the distortion between candidates and the input block, whereas the RDO also considers the bit rate impact of each decision.

Still, as can be verified in eq. (4), the SSE operation for blocks of size $M \times N$ can be quite expensive, requiring $M \times N$ subtractions, $M \times N$ multiplications, and $M \times N - 1$ sums of varying bit depths. Therefore, a more efficient heuristic should be able to not only select candidates locally with the SSE but also reduce the number of arithmetic operations required by each SSE.

5.2.1 Observation of SSE Error in Intra Prediction

Because the intra prediction modes are limited to using only reference samples adjacent to the left and above the input block, as explained in Section 3.1, more accurate predicted samples are obtained in positions spatially closer to the reference samples, and less accurate predicted samples are obtained as the spatial distance from the reference arrays increases.

To verify the extent of the loss of accuracy in intra predicted samples spatially located far away from the reference samples (e.g., in the bottom-right corner of a block), experiments were performed in the AV1 reference software *libaom* 2.0.0 (AOMedia, 2022). The experiments consisted of recording the residual information of every intra predicted block and computing the average residue (error) for each combination of block size and prediction mode.

Figure 28 shows, in the form of heat maps, the average error for non-directional modes, normalized to a 0 to 1 range, when applied to blocks of size 16×16. It can be noticed that prediction modes such as Smooth, Paeth, and DC, which make use of all reference samples from both the *AboveRow* and *LeftCol* arrays, show a clear pattern of lower error in the top-left region of the blocks and higher error in the bottom-right region. On the other hand, the Smooth Vertical mode, which uses all samples from *AboveRow* and only the last sample of *LeftCol*, shows a pattern of lower error in the whole top region and a small portion of the bottom-left corner.

Figure 29 shows the average error for the eight nominal directional modes when applied to blocks of the same size. Directional 180 and 203 are modes that only use references from *LeftCol* and show a pattern of lower error in the left region of the block, whereas Directional 45, 67, and 90 are modes that only use references from *AboveRow* and show a pattern of lower error in the top region. On the other hand, Directional Modes 113, 135, and 157 use reference samples from both arrays and show a pattern of lower error around both the top and left edges of the block.

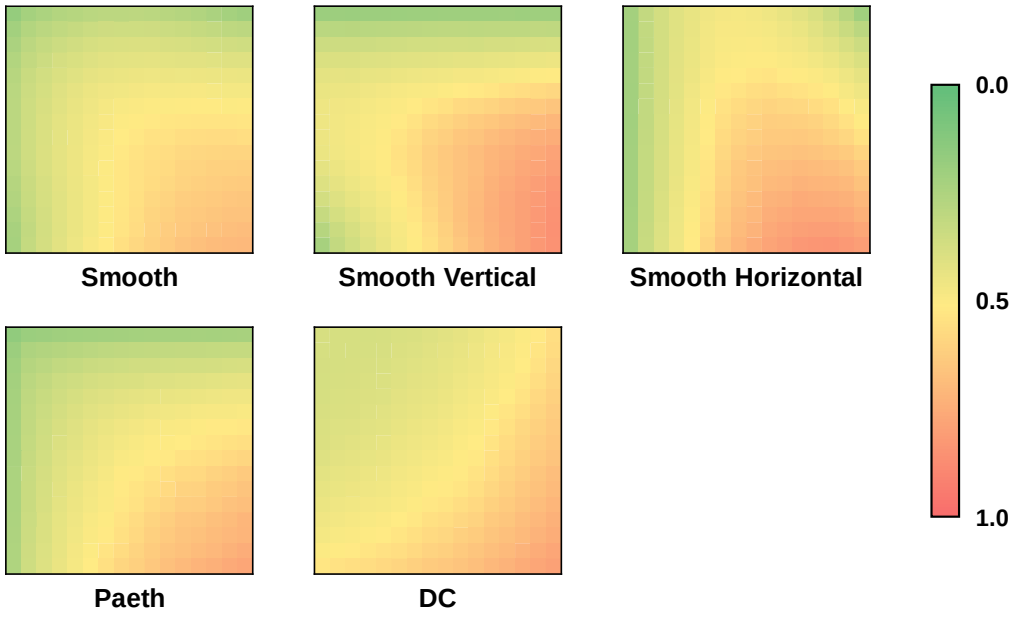

Figure 28 Heat maps showing the average error of non-directional intra modes when applied to blocks of size 16×16.

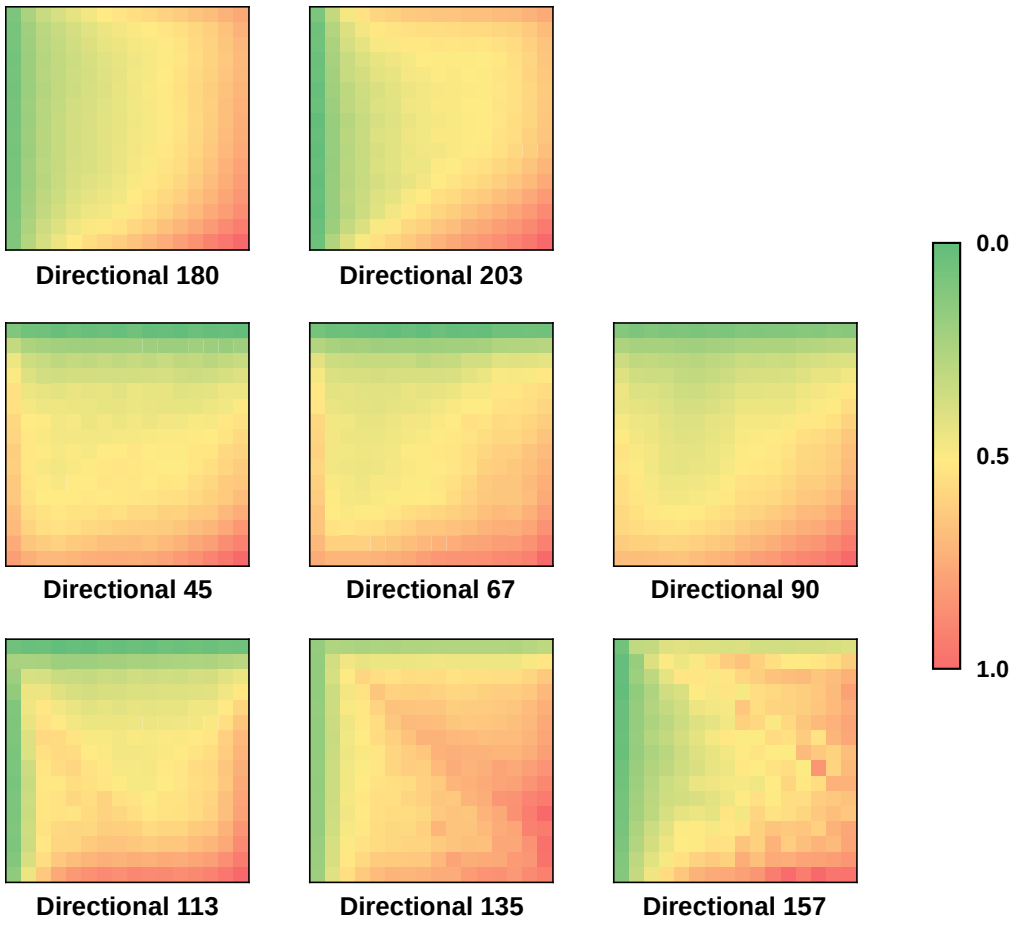

Figure 29 Heat maps showing the average error of nominal directional intra modes when applied to blocks of size 16×16.

Based on this error pattern data, a heuristic-based method of subsampling the distortion metric used for block matching can be made.

5.2.2 Mode-adaptive SSE Subsampling Masks

The proposed mode-adaptive SSE subsampling algorithm (CORRÊA *et al*., 2022a) reduces the overall cost of the SSE computation by applying a non-uniform subsampling pattern, prioritizing the warm areas of the heat map associated with each prediction mode, whilst discarding samples from the cold areas. For each possible combination of prediction mode and block size, a subsampling mask is generated offline (not during encoding time) by the algorithm.

To define an effective way of subsampling SSE operations based solely on the error patterns (heat maps) presented in Section 5.2.1, experiments were done in *libaom* 2.0.0 (AOMedia, 2022). Promising results were obtained for subsampling masks that eliminate up to three-quarters of the predicted samples during SSE computation. However, even though the warm areas are decisive in discarding bad prediction candidates, it was observed the cold areas must also be considered, but to a lesser extent.

Therefore, for a given prediction mode and block size, the proposed subsampling mask generation algorithm is done in two steps, using as input the average error (heat maps) associated with the mode and size, and delivering as output a subsampling mask, which is simply a matrix of 0's and 1's telling which positions are to be ignored or considered, respectively, during the SSE computation.

1. A parameter *HEA* (Higher Error Area) is used to determine the percentage of the highest error positions to be considered, e.g., *HEA*=25% will result in a loop that finds one-quarter of the positions in the heat map with the highest average error, and checks these positions in the mask, whilst the other three-quarters of the mask will remain unchecked.
2. A parameter *LEA* (Lower Error Area) is used to determine a percentage of the positions checked in the first step that will be unchecked uniformly, with the same number of positions checked uniformly in the empty area from the first step.

Figure 30 shows the subsampling mask obtained by the proposed algorithm using as input the average error of the Directional 90 mode in blocks of size 32×32.

The parameters used in this example are *HEA*=25% and *LEA*=25%. In the first step (left side of Figure 30), a loop checks in the mask (green positions) the 256 positions with the highest error from the heat map used as input, i.e., 25% of the 1024 positions of a block of size 32×32, according to the *HEA* parameter. In the second step, another loop unchecks uniformly 64 of the green positions (positions with an X), one at every four, and then checks the same number of positions in the originally empty area, one at every twelve positions (blue positions), i.e, 25% of the 256 checked positions, according to the *LEA* parameter.

With subsampling masks like the one from Figure 30 (right side), every time the encoder uses the Directional 90 mode in the prediction of a block of size 32×32, it will only use the green and blue positions in the distortion metric computation, while ignoring the rest of the block, thus saving a significant number of arithmetic operations.

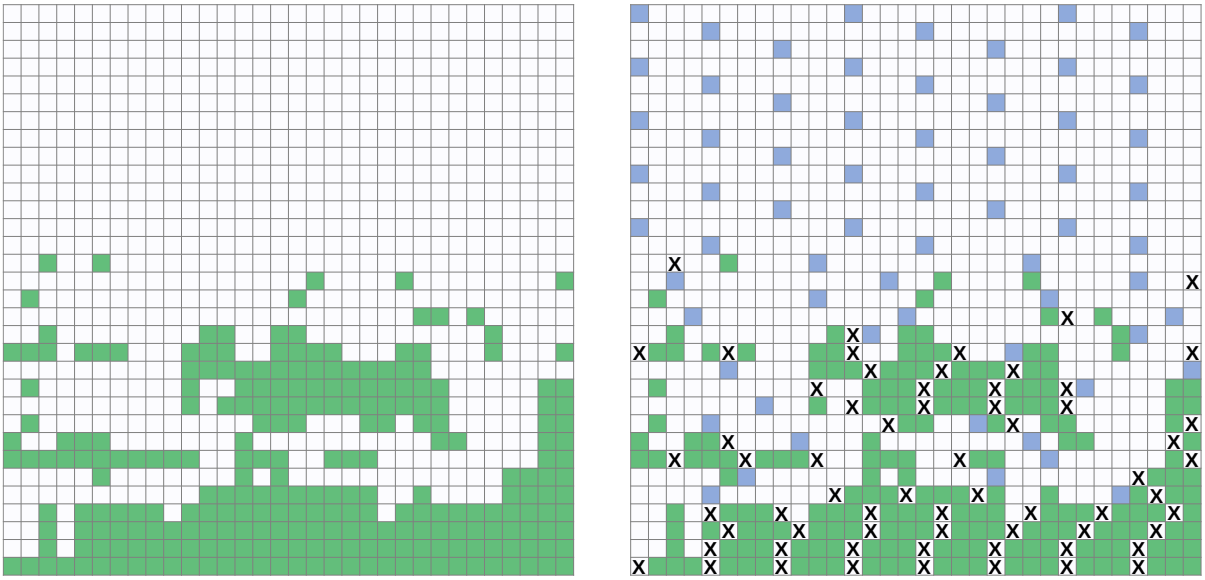

Figure 30 Left: 25% of the highest error positions are checked in the subsampling mask. Right: 25% of the previously checked positions are unchecked uniformly and redistributed uniformly in the empty area.

It is important to note that the set of subsampling masks generated by this algorithm depends on the heat maps provided, which in turn depend heavily on how the experiments were conducted. Different sets of test sequences and different sets of parameters will result in slightly different heat maps; however, the pattern of the prediction error of each mode is consistent. Furthermore, subsampling masks particularly efficient for a specific kind of content can be generated if the heat maps

were generated using the same content, e.g., heat maps showing the average error for digital screen content will lead to subsampling masks more adequate to SCC.

5.3 Results and Discussion

Sections 5.3.1 and 5.3.2 shows the results of the TbFMD and MaSBM algorithms, each one tested with different parameter combinations. Section 5.3.3 shows the results for the two algorithms combined and discusses related works.

All experiments were done with the AV1 reference software *libaom* 2.0.0 (AOMedia, 2022), following the coding parameters recommended in the document CWG-B075o (ZHAO *et al*., 2021), and using the test sequences recommended in document JVET-W2017-v1 (KARCZEWICZ; YE, 2021). The results for encoding efficiency are presented in BD-BR, and for encoding time difference are presented in ΔT (eq. 18), where $T_{proposal}$ and $T_{reference}$ are the execution time of the original software and the software modified with the proposed algorithms, respectively. More detailed information related to the software experiments can be read in Appendix A.

$$\Delta T = \frac{T_{proposal} - T_{reference}}{T_{reference}} \times 100 \quad (18)$$

5.3.1 Results for Algorithm 1: Texture-based Fast Mode Decision (TbFMD)

In the CDEF (MIDTSKOGEN; VALIN, 2018), the direction detection is used in in-loop filtering and, in that context, *Thr*=10 is considered a good threshold point. In the intra prediction context, however, a higher than ideal *Thr* value classifies more blocks as smooth, disabling all directional intra modes even if the block has edges, whereas a lower than ideal *Thr* classifies more blocks as sharp, even if they are smooth surfaces, testing directional intra modes pointlessly.

To find the ideal threshold for intra prediction, firstly the same threshold of CDEF was tested and found to be too conservative in this context, most of the time not disabling directional modes for smooth surfaces. Then, different thresholds from the interval 10≤*Thr*≤18 were evaluated. Figure 31 shows the compression efficiency and encoding time results for different threshold points. In the figure, it can be observed that the encoding time was reduced in an almost linear pattern as the threshold increased, with *Thr*=10 giving a reduction of 17.5% and *Thr*=18 giving a reduction of 28.6%. On the other hand, the encoding efficiency curve kept stable from *Thr*=10

(1.18%) to *Thr*=15 (1.26%), but plummeted beyond that point, reaching 2.00% BD-BR$_{YUV}$ for *Thr*=18, indicating that the algorithm started classifying sharp blocks as smooth incorrectly as the threshold increased beyond 15. Therefore, for the AV1 intra prediction, the ideal *Thr* for the TbFMD algorithm was defined as 15, which provided a good trade-off between encoding time reduction and encoding efficiency loss, of 22.6% and 1.26%, respectively.

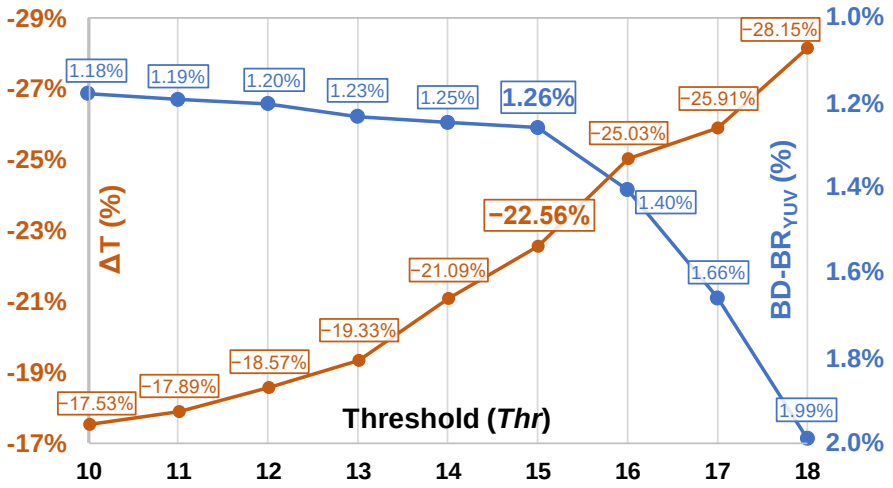

Figure 31 Encoding efficiency loss and encoding time difference for different threshold values, with *Thr*=15 showing the best trade-off.

5.3.2 Results for Algorithm 2: Mode-adaptive Subsampling in Block Matching (MaSBM)

The subsampling masks can be generated with a variety of combinations of *HEA* and *LEA* parameters. To find the most efficient setup, experiments were conducted using all combinations between *HEA*={25%, 50%, 75%} and *LEA*={0%, 25%, 50%, 75%, 100%}. It is important to note that when LEA=0%, the mask generation algorithm stops in the first stage, and when *LEA*=100%, there is no longer a mode-adaptive subsampling, and instead, a simple uniform subsampling is used for every prediction mode. In these experiments, the fast mode decision lets the intra prediction generate all possible candidates and then creates a reduced RD-list with the four candidates with the lowest error according to the subsampled SSE operations.

Table 12 shows the resulting BD-BR$_{YUV}$ and ΔT, and Figure 32 shows the behavior of the BD-BR$_{YUV}$ curves (in logarithmic scale) for all combinations of parameters. The algorithm tested used heat maps generated with experiments that

followed the parameters described in Appendix A, except that only the first 25 frames of each test sequence were used, instead of the whole sequence.

Table 12 Encoding efficiency and encoding time difference results for different *HEA* and *LEA* combinations

HEA **(%)**	***LEA*** **(%)**	**BD-BR$_{YUV}$ (%)**	**ΔT (%)**
25	**0**	1.85	−12.01
	25	1.09	
	50	0.76	
	75	0.81	
	100	1.14	
50	**0**	0.64	−3.75
	25	0.23	
	50	0.05	
	75	0.06	
	100	0.10	
75	**0**	0.45	−2.51
	25	0.18	
	50	0.04	
	75	0.04	
	100	0.08	

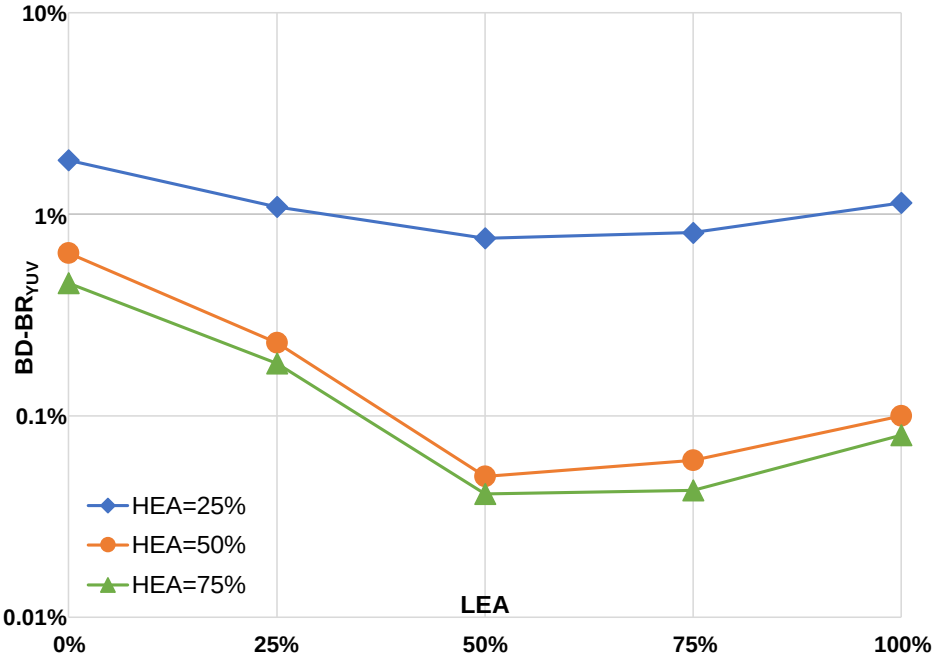

Figure 32 Encoding efficiency curves for 15 different parameter combinations.

As can be observed in Table 12, in terms of encoding time reduction, only the *HEA* parameter matters, because it is the one that defines the degree of subsampling, directly affecting the number of operations skipped. A setting of *HEA*=25% provided 12% of encoding time reduction, whereas the less aggressive settings of *HEA*=50% and 75% led to a much lower time saving of 3.8% and 2.5%, respectively.

Although *HEA*=25% provides significantly better time-saving results, it can be observed in Figure 32 (blue line) that this setting also results in a much higher impact on encoding efficiency. It can also be observed that the *HEA*=50% (orange line) and *HEA*=75% (green line) curves are very close to each other, and from a hardware-friendly perspective, *HEA*=50% was found to be an ideal choice among the two, since the software results are similar, but *HEA*=50% allows more arithmetic operations to be skipped, hence allowing more area and/or power to be saved, depending on the hardware design itself.

Regarding the *LEA* parameter, the same behavior can be observed in Figure 32 for all curves. The *LEA*=0% setting, which concentrates all the operations in the higher error area, provides the worst encoding quality results, showing that the lower error area cannot be completely ignored. Among the *LEA*=25%, *LEA*=50%, and *LEA*=75% settings, it can be observed that the uniform redistribution of 50% of the positions checked in the first stage of the mask generation, provides the best encoding quality results. It is also important to mention the specific case of the *LEA*=100% setting, which is similar to a naive non-adaptive uniform subsampling solution, that showed better results than the proposed mode-adaptive subsampling method in its *LEA*=0% and *LEA*=25% configurations.

Therefore, the *HEA*=50% and *LEA*=50% setup was found to be the ideal solution when encoding efficiency is a priority over encoding time savings, providing a negligible BD-BR_{YUV} impact of 0.05% and time-saving of 3.75%. Otherwise, if encoding time savings is the priority or a low-power design is desired, the *HEA*=25% and *LEA*=50% setup can be used, providing a BD-BR_{YUV} impact of 0.76% and time savings of 12.01%.

5.3.3 Results for Algorithms 1 and 2 Combined

The previous sections presented the results for each algorithm when used individually. This section provides the results of both the TbFMD and MaSBM algorithms interacting with each other in the same AV1 encoder.

TbFMD is executed before the prediction of intra modes, effectively reducing the number of modes computed from the maximum number of modes supported by the codec to only the set of non-directional modes (in case of a smooth input block), or the ND set plus four directional modes (in case of a sharp input block). On the other hand, MaSBM is only executed after the prediction of intra modes and, therefore,

its performance is directly affected by the output of TbFMD, because it will no longer apply subsampling to the full set of intra predicted blocks, but only to the reduced set of modes defined by TbFMD. With the integration of both algorithms, the optimized intra prediction module works with the following steps:

1. Algorithm 1 creates a reduced RD-list of 11 up to a maximum of 15, although the number of modes can be lower if certain optional modes are disabled in the encoder, such as the RBF and SCC modes.
2. The intra prediction module generates the candidates based on the RD-list.
3. All candidates are evaluated according to the mode-adaptive subsampled SSE masks and only the four best candidates are kept in the RD-list.
4. Finally, the modified RD-list with only four candidates is sent to the RDO.

Experiments were conducted with the two algorithms combined, with *Thr*=15 as the parameter for the TbFMD algorithm, and *HEA*=50 and *LEA*=50% for MaSBM. Table 13 shows detailed results of these experiments.

Table 13 Encoding efficiency and time difference results for the integration of TbFMD and MaSBM per sequence, per class and total

Class	Sequence	BD-BR$_{YUV}$ (%)	ΔT (%)
A1 (UHD 4K)	*Tango2*	1.38	−35.44
	FoodMarket4	1.17	−40.13
	Campfire	111	−32.71
	A1 Average	**1.22**	−36.09
A2 (UHD 4K)	*CatRobot*	0.85	−32.93
	DaylightRoad2	1.08	−29.66
	ParkRunning3	0.89	−26.39
	A2 Average	**0.94**	−29.66
B (1080p)	*MarketPlace*	3.15	−29.22
	RitualDance	0.63	−35.88
	Cactus	2.70	−25.30
	BasketballDrive	−0.50	−29.77
	BQTerrace	0.66	−22.68
	B Average	**1.33**	−28.57
C (480p)	*BasketballDrill*	1.82	−22.57
	BQMall	0.97	−22.25
	PartyScene	1.45	−16.03
	RaceHorses	1.52	−23.66
	C Average	**1.44**	−21.13
E (720p)	*FourPeople*	1.32	−30.53
	Johnny	1.22	−35.77
	KristenAndSara	1.86	−34.35
	E Average	**1.47**	−33.55
ABCE Average		**1.28**	**−29.80**

In this table, it can be observed that the ABCE average is lower than the sum of the impact caused by each algorithm when used individually. This is explained by the fact that MaSBM, when used individually, selects the four best candidates out of the full set of prediction modes relying only on a subsampled distortion metric. On the other hand, when paired with TbFMD, it selects the best candidates out of an already reduced set of prediction modes, and, therefore, errors related to the heuristic of subsampling are limited to a set of probable candidates. It is relevant to mention that, in this table, it can also be observed that the proposed algorithms perform better than the average for UHD 4K videos, showing that the selected parameters are suitable for very high-definition videos.

Considering the low BD-BR_{YUV} impact of 1.28%, the significant time saving of 29.8%, and their hardware-friendly characteristics, it can be said that the proposed algorithms combined achieve the goals of this Ph.D. thesis.

The algorithms in Jeong, Gankhuyag, and Kim (2019a; 2019b) can be directly compared to the one proposed in this section, as they also propose fast mode decisions for AV1 intra prediction. Jeong, Gankhuyag, and Kim (2019a) report a BD-BR impact of 0.44% and ΔT of −15.86%, which is a lower encoding efficiency impact, but also a lower time saving than the algorithm proposed in this section. Jeong, Gankhuyag, and Kim (2019b) report a BD-BR impact of 0.04% and a ΔT of −8.67%, which is a negligible impact, but also a modest time saving when compared to the algorithm proposed in this section.

The works from Guo *et al*. (2018a; 2018b), Chen *et al*. (2019), and Chiang, Han, and Xu (2019) cannot be directly compared to the one presented in this section, because these works optimize the block partitioning stage of the encoder and leave the intra prediction module untouched. However, this means that the algorithms from this Ph.D. project could be paired with the algorithms of the abovementioned works to achieve further optimization of the encoder.

6 HARDWARE DESIGNS FOR AV1 INTRA-PICTURE PREDICTION

This chapter presents hardware designs for the intra prediction stage of the AV1 encoder. These designs are based on the already published contributions of Corrêa *et al.* (2019a; 2019b; 2020a; 2020b; 2022b) and Neto *et al.* (2020; 2021a; 2021b; 2022).

The base design, before any optimization, is shown in Figure 33. This design is composed of a dedicated directional intra prediction design (Section 6.1), a dedicated non-directional intra prediction design (Section 6.2), a control unit shared by both designs (Section 6.3), and an SSE-based local decision design (Section 6.4).

Optimizations on top of the base design using the algorithms proposed in Chapter 5 are presented in Section 6.5, and results are discussed in Section 6.6.

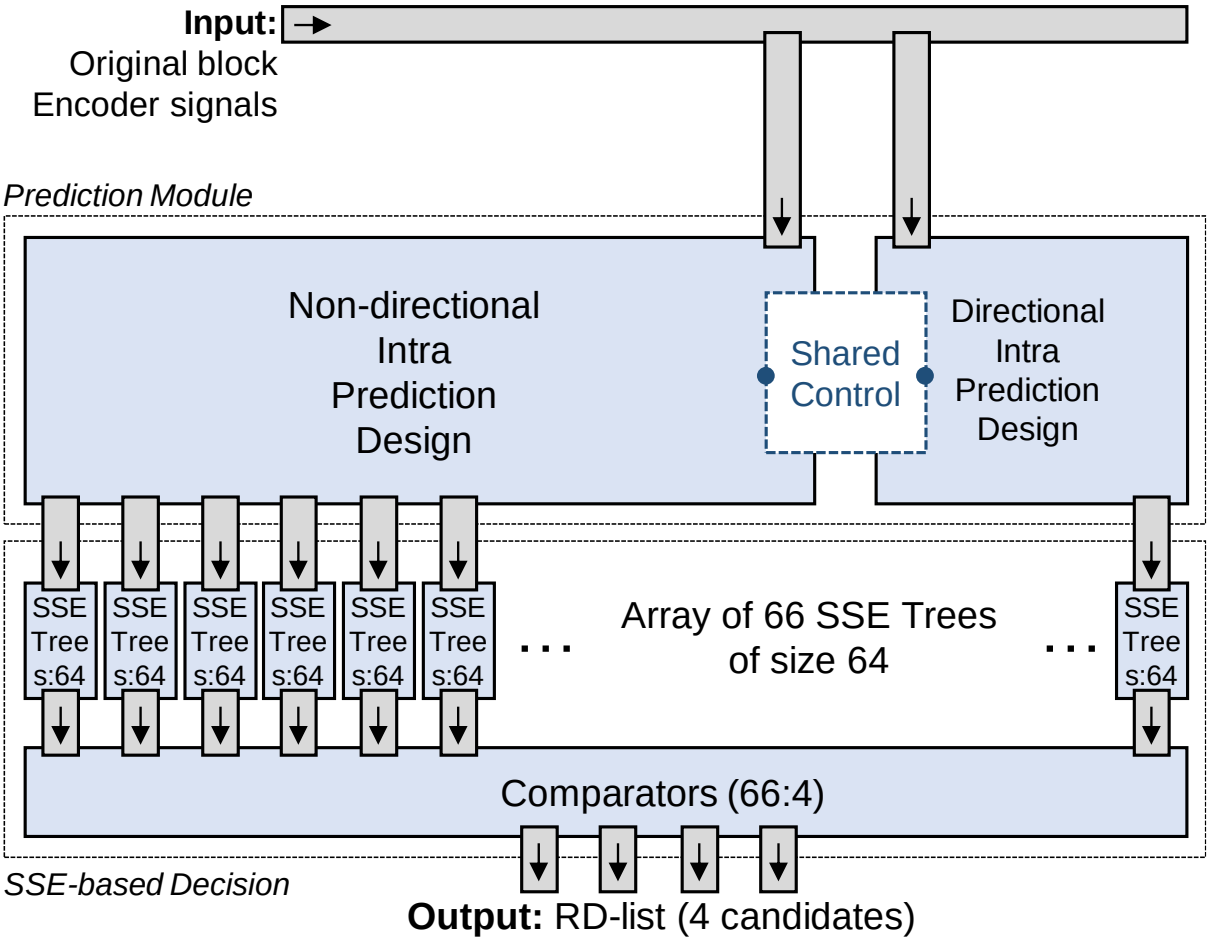

Figure 33 Intra prediction base design.

6.1 AV1 Directional Intra Prediction Design

The proposed design for the directional intra prediction (CORRÊA *et al.*, 2020b, NETO *et al.*, 2020; 2021a; 2021b; 2022) works at 64×64 SB level, which is the maximum block size allowed by the AV1 intra prediction modes.

This module is mainly composed of two Reference Sample Filtering Units (RSFUs), two Reference Sample Upscaling Units (RSUUs), 78 Directional Mode Prediction Units (DMPU) working in parallel, and several buffers for holding each possibility of filtered and upscaled reference arrays. Figure 34 shows a register-transfer level (RTL) diagram of this design.

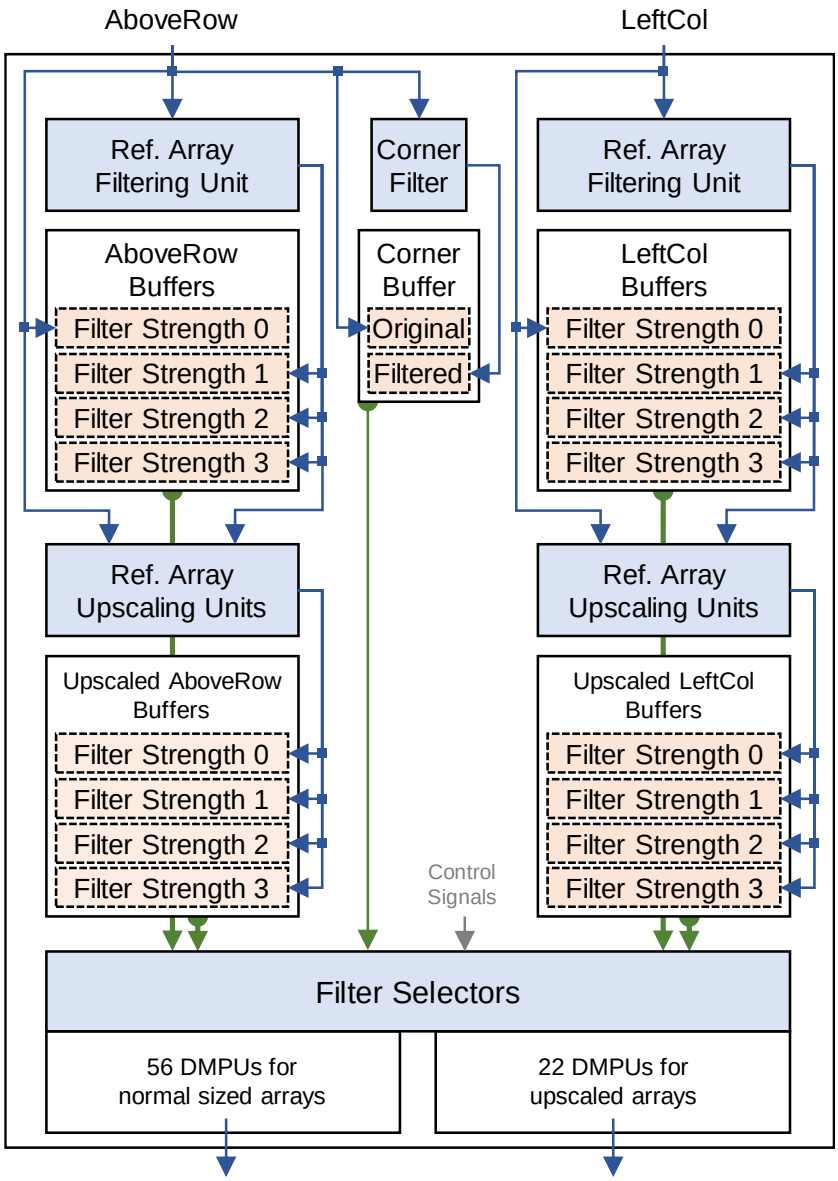

Figure 34 Directional intra prediction design.

6.1.1 Reference Sample Filtering Units

There are one corner filter and three different types of array filters in the RSFUs. The corner filter is applied to a single reference sample, which is the top-left reference sample (see Figure 7). A three-tap filter for the corner samples is defined as a combinational circuit made entirely of shift-adds operations that produces the same results as eq. (19), where *a* and *c* are *AboveRow*[0] and *LeftCol*[0], and *b* is *AboveRow*[-1]. A filter selector is used to send the filtered corner sample only to MPUs with 90<*pAngle*<180 and only if the block size follows the rule *height*+*width*>=24.

$$cf = (5a + 6b + 5c + 8) \div 16 \tag{19}$$

The array filters require up to three reference samples from each side of the sample being filtered. There are three different filter strengths, and all are needed because different DMPUs can use different filter strengths for a given reference sample, or no filter at all. The filters for strength levels one, two, and three are also combinational circuits made entirely of shift-adds operations that produce the same results as eqs. (20-22), respectively, where *x* is a sample from either *AboveRow* or *LeftCol* and *i* the index of the sample in the array.

$$fs1 = (4x_{i-1} + 8x_i + 4x_{i+1} + 8) \div 16 \tag{20}$$

$$fs2 = (5x_{i-1} + 6x_i + 5x_{i+1} + 8) \div 16 \tag{21}$$

$$fs3 = (2x_{i-2} + 4x_{i-1} + 4x_i + 4x_{i+1} + 2x_{i+2} + 8) \div 16 \tag{22}$$

There are six arrays of filters working in parallel, one of each strength for each array of reference samples. Each array of filters has 129 filters working in parallel, which is the size of the input reference sample arrays in the worst-case scenario, and is also the size of each buffer for filtered reference samples. The filter selector used to decide which filtered *LeftCol* and filtered *AboveRow* are sent to each DMPU follows rules based on the size of the input block and on how previous blocks were coded. Table 14 and Table 15 show the filter selector decisions for the nominal angles, where the first number of a pair is the filter strength if no adjacent neighbors were coded using one of the three Smooth modes, and the second number is if at least one was.

Table 14 Filter selector for *AboveRow* (only nominal modes are shown)

Block Size	DMPU					
	45	67	113	135	157	203
4×4	0, 1	0, 0	0, 0	0, 1	1, 2	1, 2
4×8, 8×4, 8×8	1, 1	0, 1	0, 1	1, 1	1, 2	1, 2
8×16, 16×8	3, 3	2, 3	2, 3	3, 3	3, 3	3, 3
16×16	3, 3	2, 3	2, 3	3, 3	3, 3	3, 3
> 16×16	3, 3	3, 3	3, 3	3, 3	3, 3	3, 3

Table 15 Filter selector for *LeftCol* (only nominal modes are shown)

Block Size	DMPU					
	45	67	113	135	157	203
4×4	1, 2	1, 2	1, 2	0, 1	0, 0	0, 0
4×8, 8×4, 8×8	1, 2	1, 2	1, 2	1, 1	0, 1	0, 1
8×16, 16×8	3, 3	3, 3	3, 3	3, 3	2, 3	2, 3
16×16	3, 3	3, 3	3, 3	3, 3	2, 3	2, 3
> 16×16	3, 3	3, 3	3, 3	3, 3	3, 3	3, 3

6.1.2 Reference Sample Upscaling Units

The content of each of the buffers for filtered reference arrays can also be upscaled according to the following rules: If no adjacent neighbors were coded using one of the three Smooth modes, upscaling happens for blocks of size {4×4, 4×8, 8×4}, otherwise, upscaling happens only for blocks of size 4×4. Therefore, more buffers were needed to store the extra samples generated in the upscaling process, each holding 33 samples needed in the worst-case scenario.

Like the filters described in Section 6.1.1, the upscaling filters were implemented as shift-adds operations in a combinational form to produce the same results as eq. (23), where *us* is the sample interpolated between the positions *i* and *i*+1 of a reference sample array. The upscaling filter also has a clip operation applied at its output to keep the resulting upscaled sampled within the bit depth limits.

$$us = (-x_{i-1} + 9x_i + 9x_{i+1} - x_{i+2} + 8) \div 16 \tag{23}$$

6.1.3 Directional Mode Prediction Units

Each DMPU is responsible for one of the 56 directional prediction modes and is composed of various Directional Sample Prediction Units (DSPU) in parallel, each capable of generating one predicted sample according to lines 10, 21, 26, and 34 of the algorithm shown in Figure 19.

The DSPU is illustrated in Figure 35. The inputs *a* and *b* are multiplied by coefficients ranging from 0 to 32 using only shift-add trees, each one designed for

minimum tree depth. This approach without the use of multipliers is relevant in an encoder solution because the high-throughput constraint can only be met with a low-latency design. Moreover, it is important to mention that one coefficient complements the other, that is, when a coefficient is equal to 15 the other will necessarily be 17 and, because of that, only 16 different DSPUs models exist and were replicated multiple times according to the need of each DMPU.

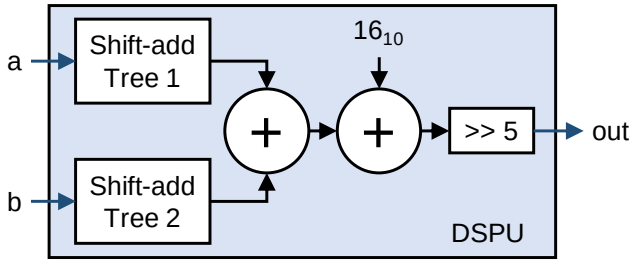

Figure 35 Directional sample prediction unit, responsible for generating one directional predicted sample from a pair of two reference samples.

The 17 DMPUs for modes associated with *pAngle*<90 (see Table 7) are each made of an array of 64 DSPUs in parallel, using reference samples from *AboveRow* as inputs. Therefore, each DMPU is capable of generating a column of 64 samples per cycle. The 10 DMPUs for modes associated with *pAngle*>180 are also made of 64 parallel DSPUs each. Their design is analog to the DMPUs for *pAngle*<90, but instead of a column, each DMPU is capable of generating a row of 64 samples per cycle using references from *LeftCol*.

In the case of the DMPUs for *pAngle*<90, the 64 positions of a column have their own *shift* values, but the *shift* values repeat among different columns (see Figure 19). This means the same array of DSPUs can be used to predict all columns of a block. Also, the coefficients required to predict columns of size 4, 8, 16, and 32 are the same required to predict the first half of a column of size 64, allowing blocks smaller than 64 samples in height to be predicted using only a subset of the array of DSPUs. The same is true for DMPUs with *pAngle*>180, but in this case, the prediction happens row by row.

The 27 DMPUs for modes associated with 90<*pAngle*<180 are made of 64 DSPUs connected to one reference array and up to 56 DSPUs connected to the other reference array. That is, these DMPUs have the combined logic of the DMPUs for *pAngle*<90 and *pAngle*>180. The exact number of DSPUs needed by each DMPU varies according to how close the mode *pAngle* is from the vertical and horizontal angles. For example, the MPU for *pAngle*=93 requires 64 DSPUs connected to

AboveRow, but only two DSPUs connected to *LeftCol*, whereas the DMPU for *pAngle*=48 requires 64 DSPUs connected to *AboveRow* and 56 DSPUs connected to *LeftCol*.

The number of reference samples used as input in each DMPU varies according to the angle associated do its prediction mode. For example, the DMPU for *pAngle*=87 (closest to the vertical angle) uses only five reference samples to predict a column of size 64, repeating the same references for multiple DSPUs, whereas the DMPU for *pAngle*=36 (farthest from the vertical angle) uses 64 different reference samples to predict a column of size 64.

Moreover, there are 22 DMPUs for the same modes already covered in the previous paragraphs, but adapted to deal with the coefficients used in the case of upscaled reference samples array. These DMPUs are much smaller, as 8×8 is the largest block size that admits upscaling.

Samples predicted using *AboveRow* as the reference array have a unique *base* index along a column, responsible for defining which reference samples are used (see Figure 19). However, as the prediction moves to the next column, all *base* indexes are simply incremented by one. This is analog for samples predicted using *LeftCol*. The DMPUs are connected to the buffers considering the indexes needed for predicting the first column/row of a block, and as the prediction moves to the next column/row every cycle, the content of each register is moved towards the leftmost register.

6.2 Non-directional Intra Prediction Module

This design for non-directional modes (CORRÊA *et al*., 2019a; 2019b; 2020a) works at the 64×64 block level, which is the maximum size allowed by the AV1 intra prediction, and shares the same control unit of the directional design, meaning that both designs work in synchrony as one complete intra prediction architecture. The RTL diagram of the design is illustrated in Figure 36.

Unlike the directional intra prediction algorithm, which is the same for all 56 modes, the non-directional intra prediction uses very distinct algorithms for each of its modes. As can be observed in Figure 36, each mode has its own module operating in parallel with the others.

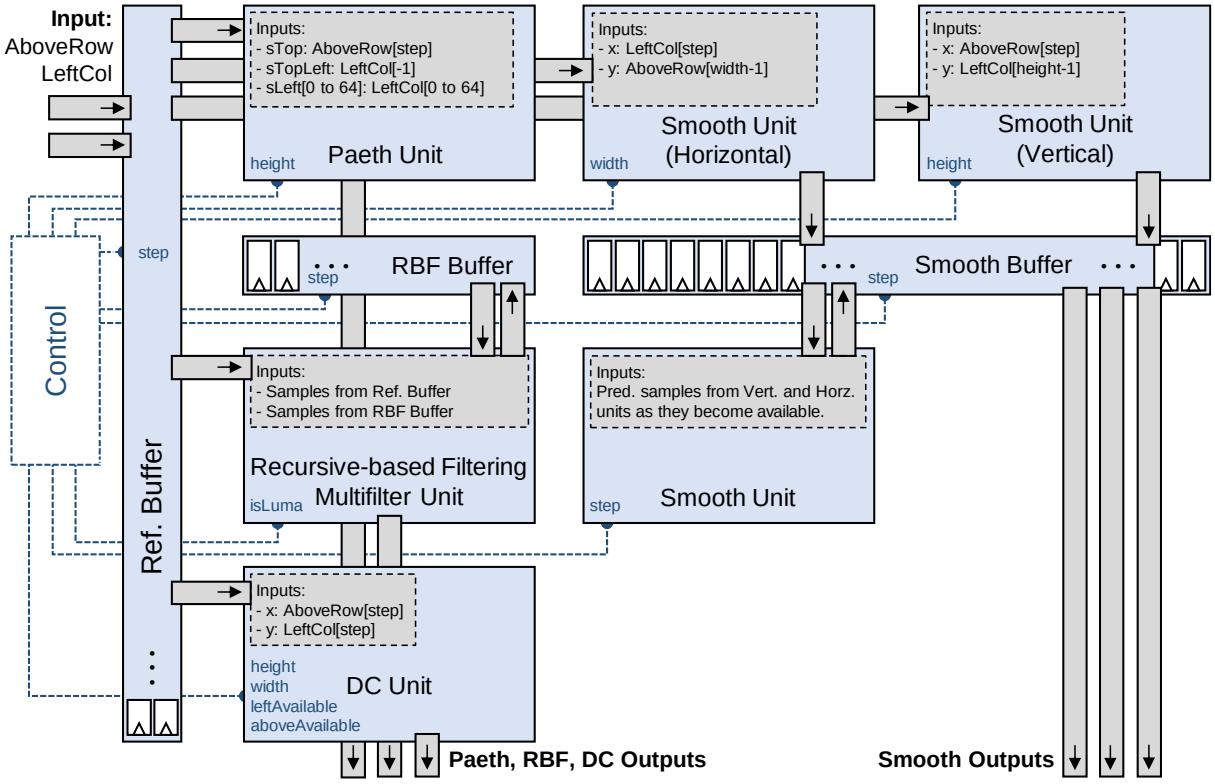

Figure 36 Non-directional intra prediction design.

6.2.1 Smooth Vertical, Smooth Horizontal and Smooth Units

As can be observed in Figure 36, there are three Smooth Units, one for each of the Smooth modes. These units share a buffer, which is needed because the Smooth Unit uses as input the predicted samples from Smooth Unit (Horizontal) and Smooth Unit (Vertical).

Since the algorithms from Figure 12 and Figure 13 are simply transposed versions of each other, and share the same constant coefficients array, a single prediction unit was designed and instantiated twice. The unit operating as a Smooth Vertical mode can predict an entire column in one clock cycle using a pair of references, thus it takes a total of *width* cycles to predict an entire block. The pair of reference samples per cycle is composed of one reference sample from the *AboveRow*, aligned with the column to be predicted (which changes every cycle) and the bottommost reference from *LeftCol* (which stays the same for the entire block). The procedure is analog for the unit operating in a Smooth Horizontal mode. It can predict an entire row in one cycle, hence it takes *height* cycles to predict an entire block. The pair of references per cycle is composed of one reference from the *LeftCol*, aligned with the row to be predicted, and the rightmost reference from *AboveRow*.

Since there are five different coefficient sets based on each possible size of the column/row to be predicted (see Table 5), a Smooth Unit for Horizontal or Vertical modes has five independent subunits to deal with each block size. These subunits, named Smooth Prediction Multiplication Unit (SPMU), generate all multiplications needed for a given pair of samples, using only shift-add operations. This strategy allows the avoidance of high latency generic multipliers, which is important on an encoder.

The shift-add trees that compose each SPMU were optimized for subexpression reuse and minimum tree depth, increasing the cost of each adder, but reducing the critical path of the tree. The algorithms used for generating such very optimized shift-add trees were highly based on methods presented by Dempster and Macleod (1995), and Vorenenko and Püschel (2007).

Figure 37 illustrates an SPMU responsible for the multiplications according to the array of coefficients of size 8. This figure also shows the next step, which is adding pairs of scaled samples, rounding, and dividing, finally resulting in eight predicted samples (each cycle). The SPMUs for sizes 4, 16, 32, and 64 are not shown in this figure. Figure 38 shows one of the two shift-add trees used in x-SPMU of size 8, which is composed of nine adders and has a depth of two.

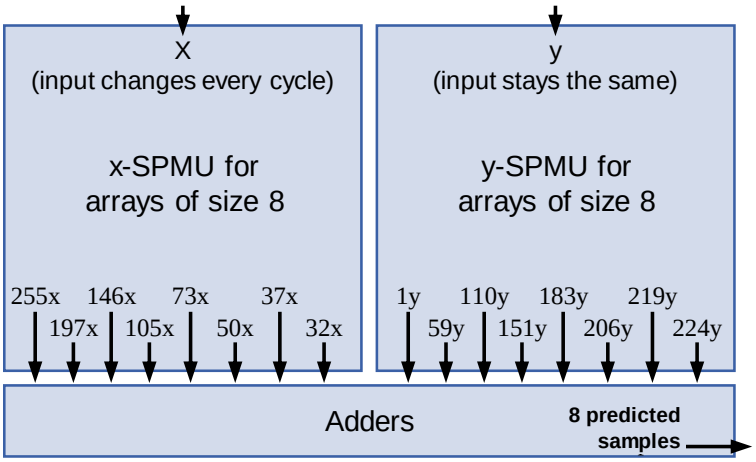

Figure 37 Smooth prediction multiplier unit or size 8.

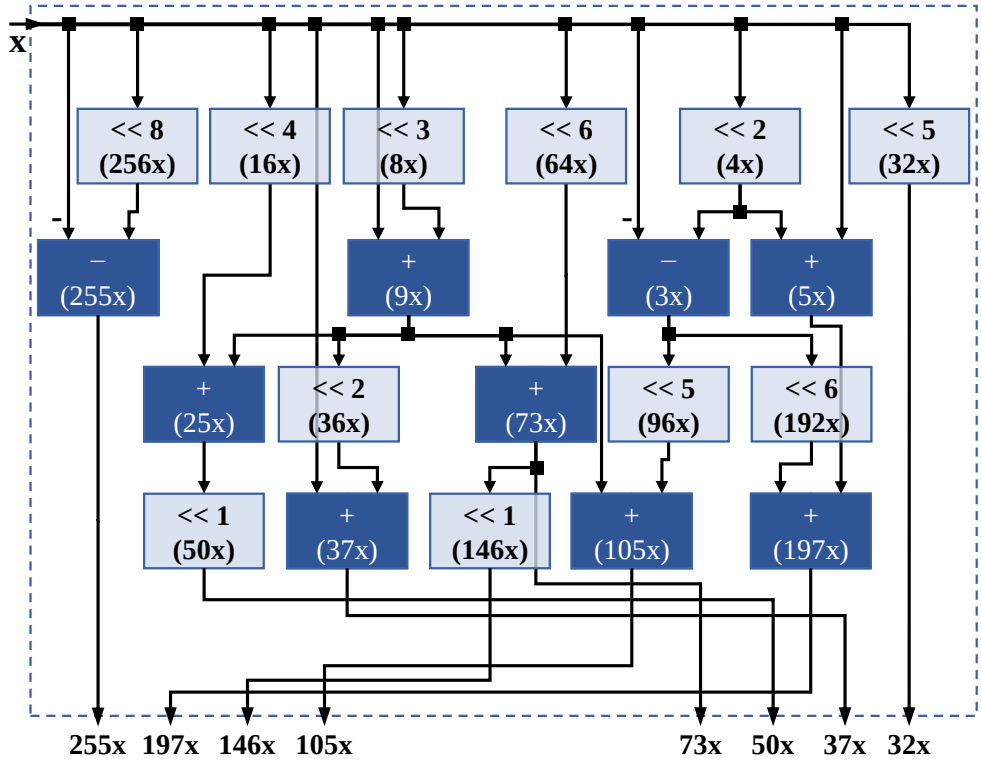

Figure 38 One of the two shift-add trees used in the SPMU for size 8, highly optimized for subexpression reuse and minimum tree depth.

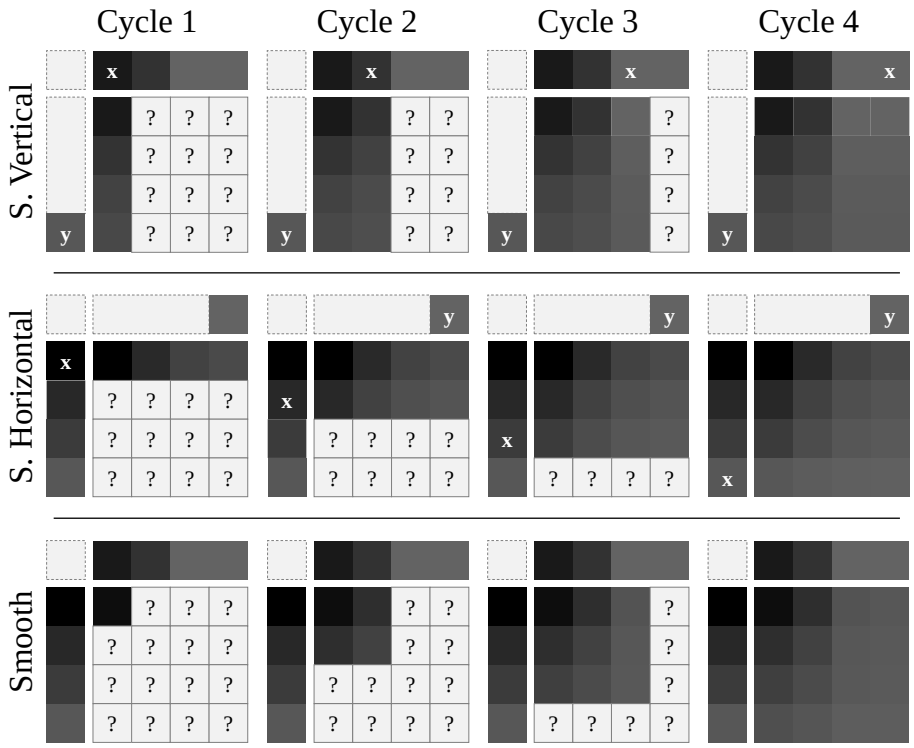

Figure 39 Prediction order of Smooth Vertical, Smooth Horizontal and Smooth modes for blocks of size 4×4. In asymmetrical blocks, one of the modes will finish before the others.

Finally, the Smooth Unit is an array of sum, rounding and division operations responsible for computing the average between the Smooth Vertical and Smooth Horizontal predicted samples, as they become available. The prediction order for all three units, for blocks of size 4×4, is illustrated in Figure 39, where filled squares are predicted samples and question mark squares are samples yet to be predicted.

6.2.2 Paeth Unit

To predict an entire column of samples per cycle, the Paeth Unit has a single instance of a circuit (top half of Figure 40) that calculates (24) and (25), where *sTop* is the reference from *AboveRow* aligned with the column and *sTopLeft* is the reference from *AboveRow*[−1]. Both (24) and (25) are calculated only once per predicted column.

$$pLeft = |sTop - sTopLeft| \tag{24}$$

$$pTopLeft_{temp} = sTop - (sTopLeft \ll 1) \tag{25}$$

It also has 64 instances of a circuit (bottom half of Figure 40) that calculates a total of *height* values of (26) and (27), all in parallel, where $sLeft_n$ is the reference horizontally aligned with the n^{th} position from the current column (read from the *LeftCol* array).

$$pTop = |sTop - sLeft_n| \tag{26}$$

$$pTopLeft_n = pTopLeft_{temp} + sLeft_n \tag{27}$$

Finally, it has 64 instances of a comparison circuit (Figure 41) for the final decision step, which compares a total of *height* 3-uples composed of *pLeft*, $pTop_n$ and $pTopLeft_n$, and selects the appropriate reference samples as output.

The Paeth Unit takes *width* cycles to predict an entire block, and it follows the same prediction order as the Smooth Unit (Vertical) (see Figure 39).

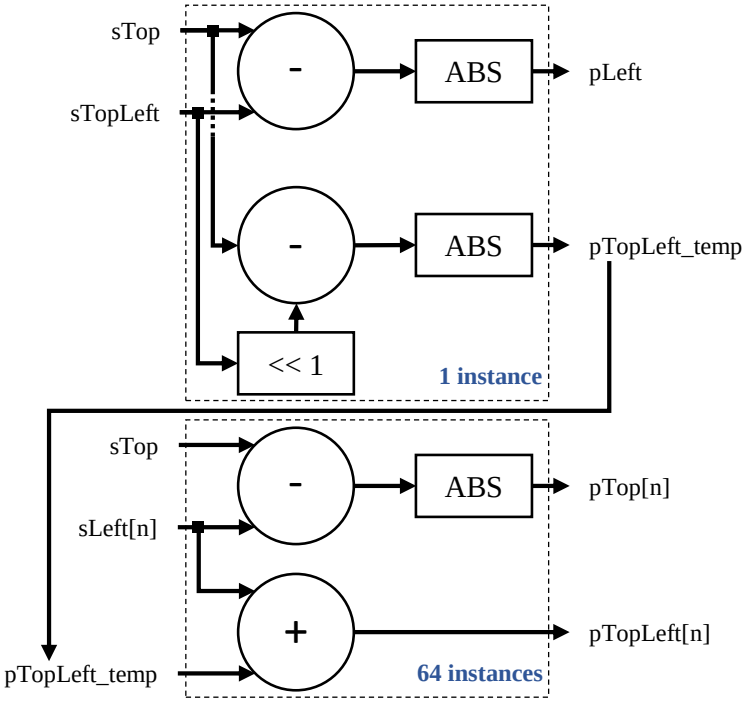

Figure 40 Paeth calculation circuit. Bottom part is replicated 64 times.

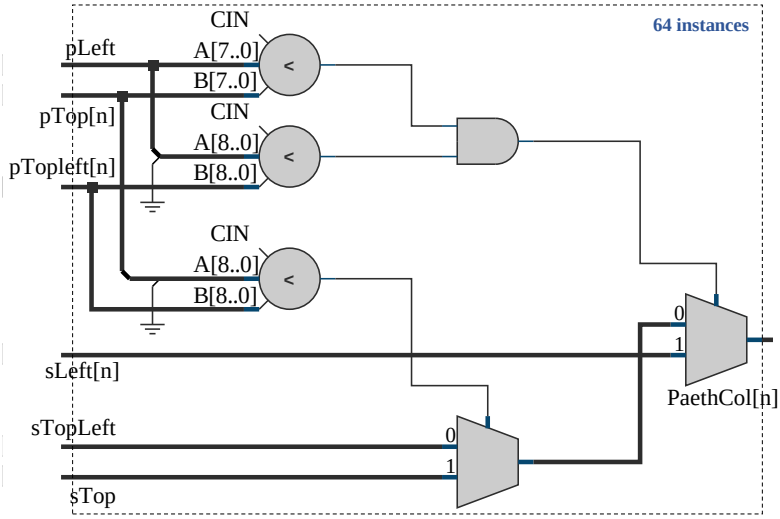

Figure 41 Paeth comparison circuit. This is replicated 64 times.

6.2.3 DC Unit

Unlike the other prediction units, the DC mode has only a single output, which is then used to represent a homogeneous predicted block. This makes the DC Unit the simplest of the prediction units.

The first part of the DC Unit (top half of Figure 42) is the sum of all the *LeftCol* references, one per cycle, using an accumulator register, and the sum of a constant value based on height for rounding purposes. In parallel to that, the same happens to

AboveRow references. This way, the DC Unit takes *height* cycles to add all *LeftCol* references and *width* cycles to add all *AboveRow* references. For asymmetrical blocks, one side finishes first and waits for the other, which means the total number of cycles needed is the largest between *height* and *width*.

The second part (bottom half of Figure 42) performs the following operations: (i) divides the left accumulator by *height* to obtain the average of *LeftCol*, (ii) divides the right accumulator by *width* to obtain the average of *AboveRow*, and (iii) adds the values stored in the accumulators and divides the result by *width+height* to obtain the overall average. The choice of which average to use as the DC Unit output (including the bit depth average) depends on the availability of adjacent reference samples, selected using the *leftAvailable* and *aboveAvailable* control flags.

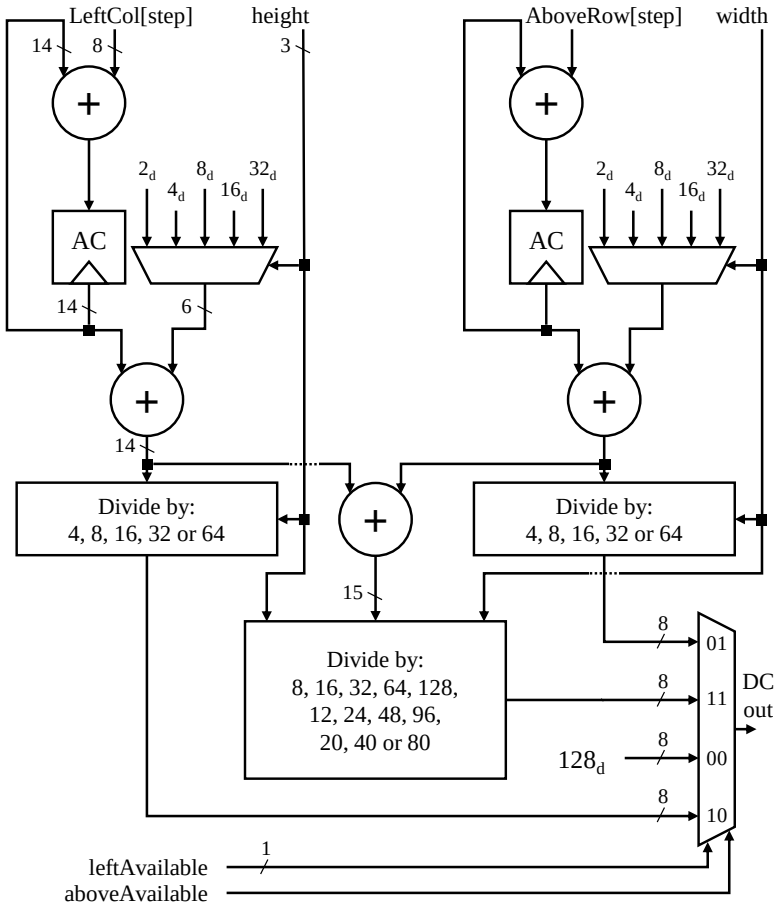

Figure 42 DC Unit.

6.2.4 Recursive-based-filtering Multifilter Unit

This prediction unit is based on a Unified Multifilter Prediction Unit (UMPU), as illustrated in Figure 43, which can apply the five RBF modes to a single 4×2 subblock.

The coefficients from Table 6 were rearranged in a way that allows maximum reuse of subexpressions for all coefficients applied to the same input. That is, the first coefficient of each set in Table 6 is applied to the first reference *L*[0], the second coefficient of each set is applied to the second reference *L*[1], and so on. The resulting sets of coefficients grouped in function of *L* are listed in Table 16.

This way, the UMPU has seven Parallel Multiplierless Multiplication Units (PMMU), one for each set of coefficients associated with each of the seven references. The PMMUs are designed as shift-add trees, in a similar way as described SPMUs from the Smooth Units, but the resulting shift-add trees are much less complex in this case because the coefficient sets from Table 16 are composed of a small number of small integers.

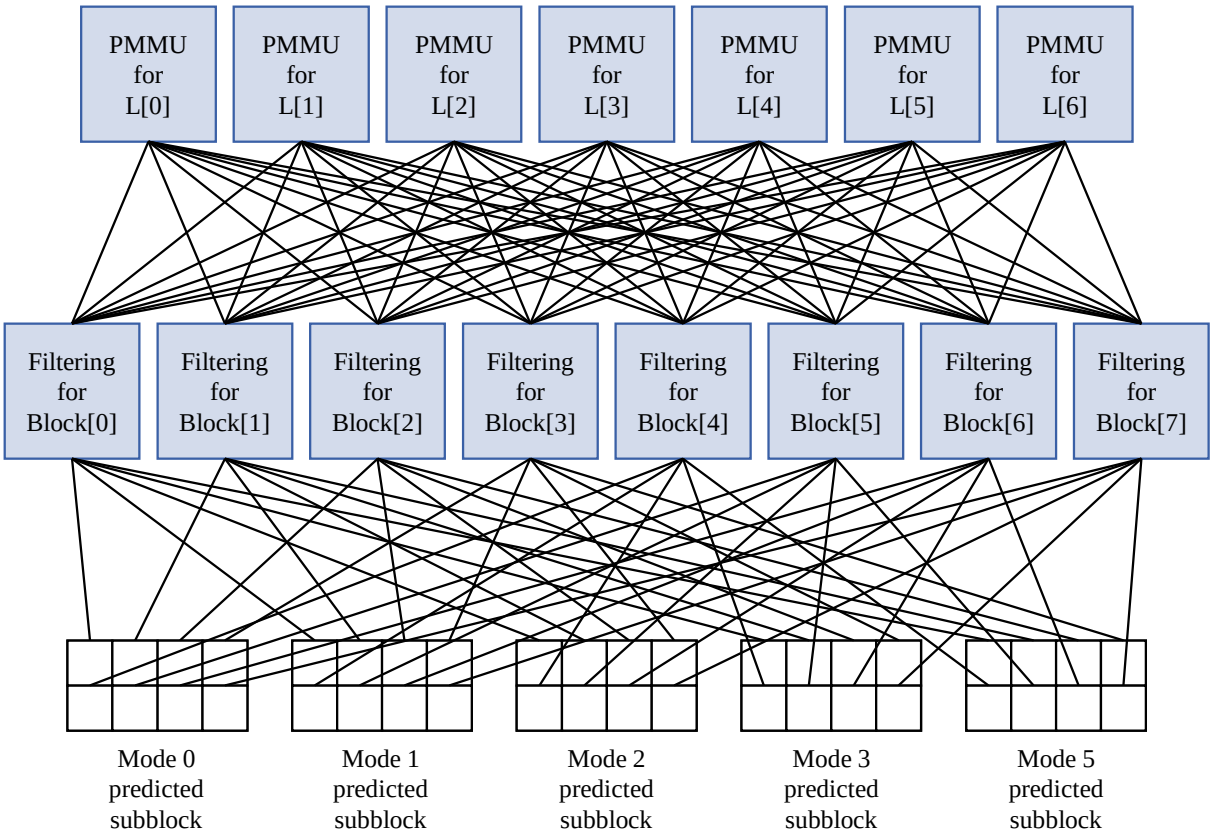

Figure 43 Unified multifilter prediction unit.

Table 16 List of coefficients used by the parallel multiplierless multiplication units

PMMU for…	Coefficients Set	Size of Set
L[0]	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12}	11
L[1]	{1, 2, 3, 4, 6, 8, 10, 12, 14, 16}	10
L[2]		
L[3]		
L[4]	{4, 6, 8, 10, 12, 14, 16}	7
L[5]	{1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 14, 16}	13
L[6]	{2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 14, 16}	12

The seven outputs of each PMMUs are the scaled references *L*, which are then used as input by eight Filtering Units, each responsible for generating one predicted sample by applying a 7-tap filter to the scaled references.

The Recursive-based-filtering Multifilter Unit has eight instances of UMPUs in a combinational setting. This long combinational path represents the critical path of the proposed architecture. This unit can process two rows of 32 samples per cycle, which means it finishes faster than the other predictions units, and then stays idle, for blocks with a width of 32 or less. The prediction order for this unit is illustrated in Figure 44.

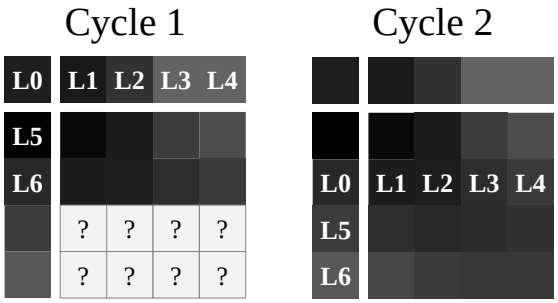

Figure 44 Prediction order of the Recursive-filtering-based multifilter unit for blocks of size 4×4. If the block width is less than 64, two entire rows will be processed per cycle, otherwise, two entire rows will be processed every two cycles.

It is important to mention that the RBF modes are only allowed for luminance blocks, hence this prediction unit stays idle during the prediction of chrominance blocks, which represents 50% of the blocks for the 4:2:0 color subsampling considered in this work.

6.3 Shared Control Unit

The control unit shared between the directional and non-directional prediction modules is a Finite State Machine (FSM), responsible for making all combinations of block partitions allowed by AV1, by exploring the partition tree (see Figure 3) using a Depth-first Search (DFS) approach, as described in Table 17.

In this table, the variable *N* indicates the partition size, which can be one of the values from the set {4, 8, 16, 32, 64}. The variable *ID*[*N*] indicates if the sub-partition of size *N*×*N* is the first, second, third, or fourth from a parent partition of size 2*N*×2*N* (see SPLIT mode in Figure 3).

In the first state, “64_START”, a new 64×64 SB is received and the prediction modules immediately execute all prediction modes on it. For clarity, all other states were named after the AV1 partition modes (see Figure 3), where the suffix N is a generic way to describe a set of states for different partition sizes.

The state “N_SPLIT” controls the DFS traversal by doing the following: (i) if the current sub-partition is not the fourth within a SPLIT mode, switch to the next sub-partition after all lower branches of the current sub-partition have been explored, (ii) if the current sub-partition is the fourth and the size is not 32×32, backtrack to the upper level, (iii) if the current sub-partition is the fourth and the size is 32×32, reset the machine by going back to the state 64_START. The state “N_SPLIT” is also executed to explore a lower level of the tree after all prediction modes have been executed for a given partition bigger than 4×4.

The remaining states are responsible for executing the intra prediction on blocks that compose a partition of size *N*×*N*. For example, the “8_HORZ_A” state operates on an 8×8 partition and executes the whole prediction process three times: twice for 4×4 blocks and once for an 8×4 block.

By following this control scheme, the directional and non-directional intra prediction modules can predict all 1,869 possible blocks inside one 64×64 SB in 7,108 clock cycles.

Table 17 Finite state machine description of the shared control unit

Current State	Prediction Information and Next State
64_START	Receives a 64×64 superblock. Next state: 64_NONE.
N_SPLIT	If *ID*[*N*] < 4: *ID*[*N*]: *ID*[*N*] + 1 Next State: *N*_NONE Otherwise, if *N* = 32 and *ID*[32] = 4: Next State: 64_START Otherwise, if N < 32 and *ID*[*N*] = 4: Next State: 2*N*_SPLIT
N_NONE	Predicts one *N*×*N* block. If *N* > 4: Next State: *N*_VERT Otherwise, if *N* = 4: Next State: 4_SPLIT
N_VERT	Predicts two *N*/2×*N* blocks. Next state: *N*_HORZ
N_HORZ	Predicts two *N*×*N*/2 blocks. Next state: *N*_VERT_A
N_VERT_A	Predicts two *N*/2×*N*/2 blocks and one *N*/2×*N* block. Next state: *N*_VERT_B
N_VERT_B	Predicts one *N*/2×*N* block and two *N*/2×*N*/2 blocks. Next state: *N*_HORZ_A
N_HORZ_A	Predicts two *N*/2×*N*/2 blocks and one *N*×*N*/2 block. Next state: *N*_HORZ_B
N_HORZ_B	Predicts one *N*×*N*/2 block and two *N*/2×*N*/2 blocks. If *N* > 8: Next State: *N*_VERT_4 Otherwise, if *N* = 8: *ID*[4]: 0 Next State: 4_SPLIT
N_VERT_4	Predicts four *N*/4×*N* blocks. Next state: *N*×*N*_HORZ_4
N_HORZ_4	Predicts four *N*×*N*/4 blocks. *ID*[*N*/2]: 0 Next state: *N/2*_SPLIT

6.4 SSE-based Decision Design

Considering that, together, the intra prediction designs produce 66 candidates, there must be a local decision to reduce the number of candidates to be sent to the RDO. In this work, an SSE-based decision, commonly found in many encoders, was implemented, which is composed by an array of SSE trees for each candidate working in parallel, followed by a comparator responsible for selecting the four best candidates.

Figure 45 illustrates an example of an SSE tree of size four, composed of four subtractors, four multipliers, and three adders of varying bit widths. In this example, four samples from a candidate can be compared with four original samples, in parallel and the accumulator register in the end allows the circuit to be used for blocks of any size. For example, the block matching of a candidate of size 4×4 comparing four

samples per cycle would take four cycles. In this work, however, to keep up with the throughput of the intra prediction modules, SSE trees of size 64 are used instead. Each SSE tree of this size requires 64 subtractors, 64 multipliers, and 63 adders.

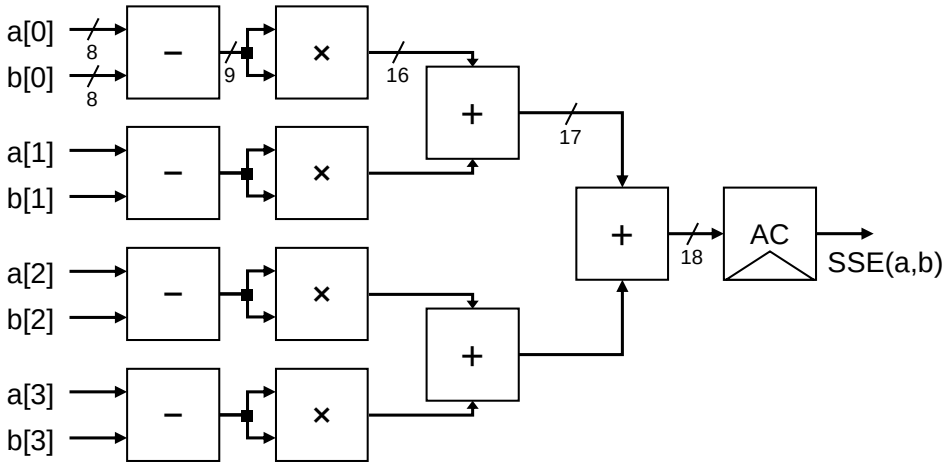

Figure 45 Example of an SSE tree of size 4.

The comparator used in the base design has to select the lowest four SSEs resulting from the SSE trees. This is done with a tree arrangement of 2:1 comparators, forming a 66:1 comparator that selects the lowest SSE among its inputs in one cycle. The inputs of the 66:1 comparator come from a buffer and, after each cycle, the selected lowest SSE is replaced in the buffer for a maximum SSE value, in a way that allows the next lowest to be selected in the following cycle, thus taking four cycles to select the four lowest SSE values. The comparator loop is called a 66:4 comparator. Figure 46 shows the structure of a 2:1 comparator (left) and a simpler notation for a 2:1 comparator (right), where black lines are signals and colored lines are comparators. Figure 47 shows the tree arrangement for a 66:1 comparator illustrated in the abovementioned notation.

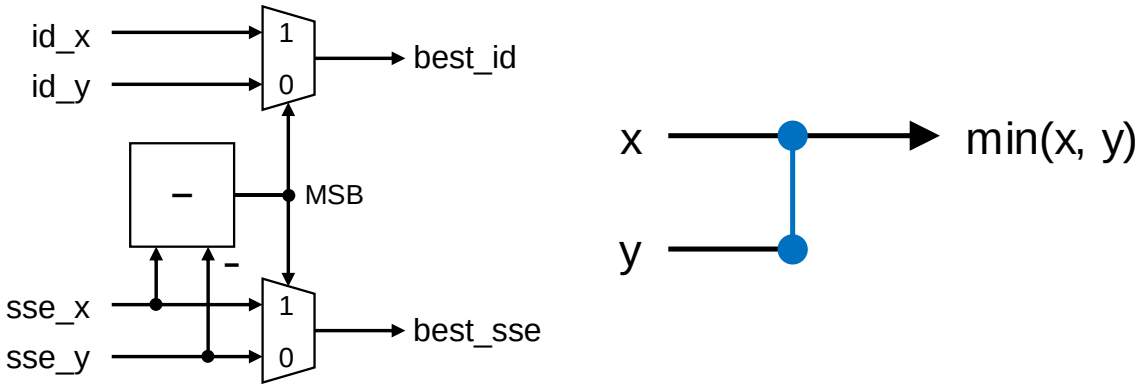

Figure 46 Left: 2:1 comparator. Right: Simplified notation for the same circuit.

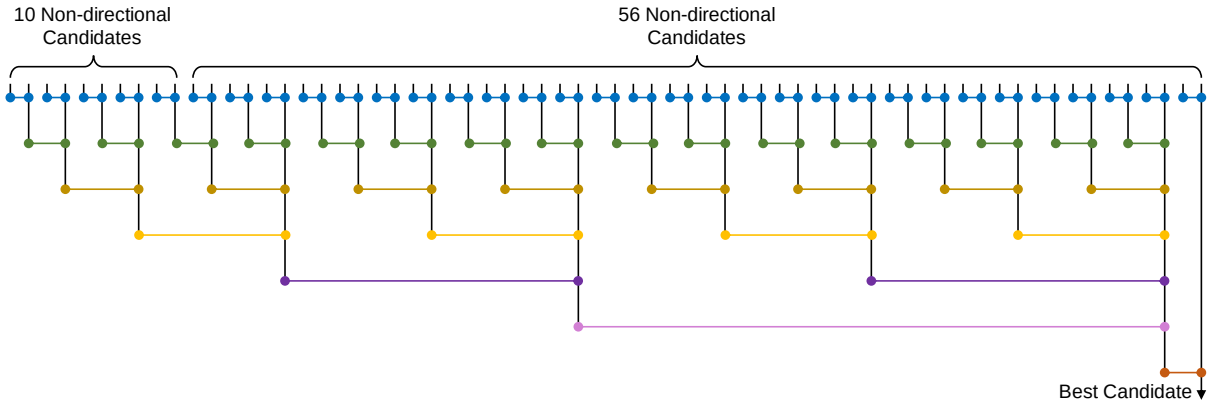

Figure 47 66:1 comparator with seven levels of depth. Each color represents a level of 2:1 comparators.

Block matching units in dedicated hardware designs can use a significant number of resources if not designed efficiently. In the HEVC intra prediction module proposed by Corrêa *et al*. (2017), the SAD trees were responsible for 37% of the gate count of the complete design. Therefore, this SSE-based design is also expected to consume an expressive number of resources, both in terms of area and power, unless optimized.

6.5 Design Optimization with TdFDM and MaSBM Algorithms

The following sections present design optimizations for the directional and non-directional intra prediction modules, based on the algorithms proposed in Chapter 5.

6.5.1 TdFDM Optimization

For this work, the proposed fast mode decision algorithm was implemented in hardware in a purely combinational way. This means that the decision of which modes will be included in the RD-list is done in only one cycle.

Figure 48 illustrates the proposed fast mode decision design (CORRÊA *et al*., 2022b). It shows that to find the dominant direction d, all eight S_d values are computed in parallel by the green-colored processing units. Figure 24 shows that eight sums can be shared between d={203, 180, 157}, and eight sums can be shared between d={113, 90, 67}. Thus, to reduce the number of required adders, these redundant sums were named, respectively, $h0$ to $h7$ (horizontal pairs) and $v0$ to $v7$ (vertical pairs).

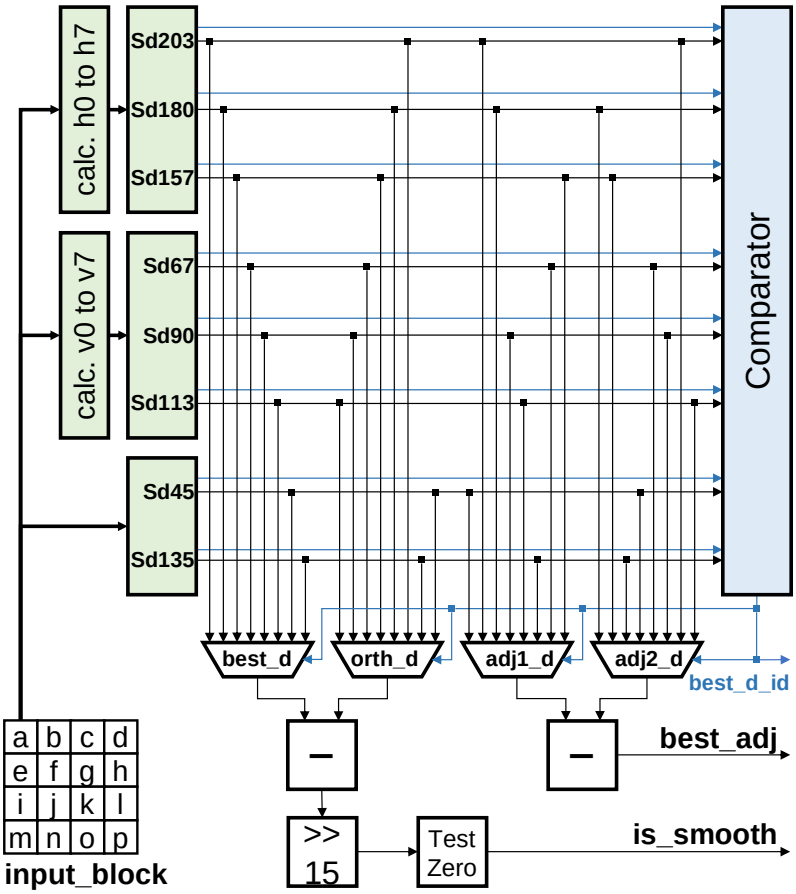

Figure 48 Texture-based fast mode decision design.

Figure 49 illustrates the circuit of the processing unit for the S_d values of d=203, d=180, and d=157, using the redundant sums $h0$ to $h7$ as input. The same circuit is replicated for the processing of d=67, d=90, and d=113, respectively, but using the $v0$ to $v7$ signals as input instead. The circuits for the processing of d=45 and d=135 (not shown) share the same design between them, but no redundant operations are shared in these circuits.

It can be observed in Figure 49 that right after the square operators, to avoid the use of dividers that would be required for the non-power of 2 divisions, multiplications in the form of shift-adds are used instead. These multiplications use the least common multiple possible for $1 \leq N_{d,k} \leq 4$, which is 12. This way, instead of dividing by $N_{d,k}$, the proposed circuit multiplies the signal by $12/N_{d,k}$ (12, 6, 4, and 3).

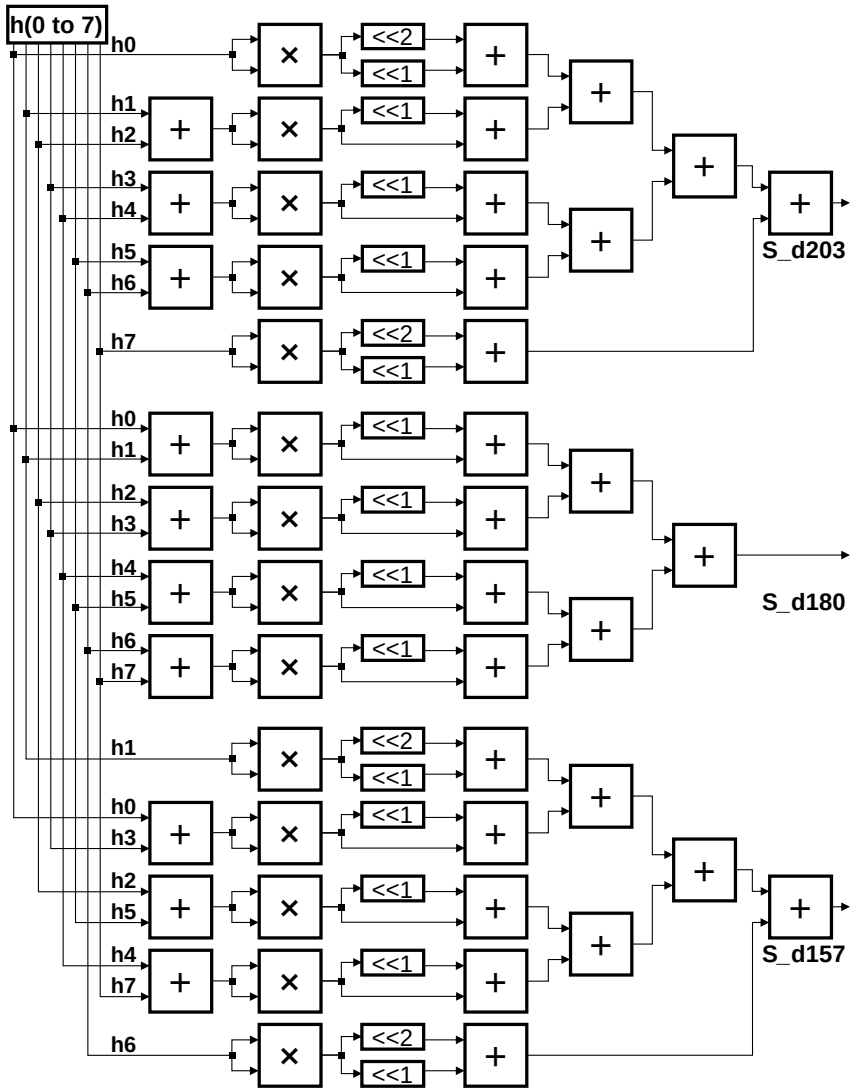

Figure 49 Processing unit for d=203, d=180 and d=157.

The computed 24-bit S_d values (*Sd203*, *Sd180*, *Sd157*, *Sd67*, *Sd90*, *Sd113*, *Sd45*, and *Sd135*) and their 3-bit *id* signals are sent to a comparator, which selects the highest value as the dominant direction. The output of the comparator is the *id* signal of the dominant direction (*best_d_id*), which is used as the selection signal for four different multiplexers.

All the multiplexers receive the eight S_d values as input, but in different orders. This way, based on the same selection signal, the multiplexers select the S_d of the dominant direction (*best_d*), the orthogonal direction (*orth_d*), and the two adjacent directions (*adj1_d* and *adj2_d*).

The *orth_d* value is subtracted from the *best_d* value, then the result is right shifted by 15 and, finally, a test is done to check if the result is zero using several 1-bit

OR gates and one NOT gate, resulting in the *is_smooth* output. The *adj1_d* is subtracted from *adj2_d* and the most significant bit is sent as the *best_adj* output.

The TbFMD design is placed before the intra prediction modules themselves, and the outputs *best_d_id* (3 bits), *best_adj* (1 bit), and *is_smooth* (1 bit) are sufficient for disabling certain directional intra prediction units using operand isolation to prevent gate switching from arithmetic operations, and clock gating to prevent changing of the state of the associated registers, according to the RD-lists listed in Table 11.

Since, in the worst-case scenario, the intra prediction module optimized with TbFMD produces 14 candidates, the SSE-based final decision can be heavily reduced. A TbFMD-optimized design does not use an array of 64 SSE trees and a 66:1 comparator, as described in Section 6.4, but instead uses only 14 SSE trees and a much smaller 14:1 comparator, which is illustrated in Figure 50.

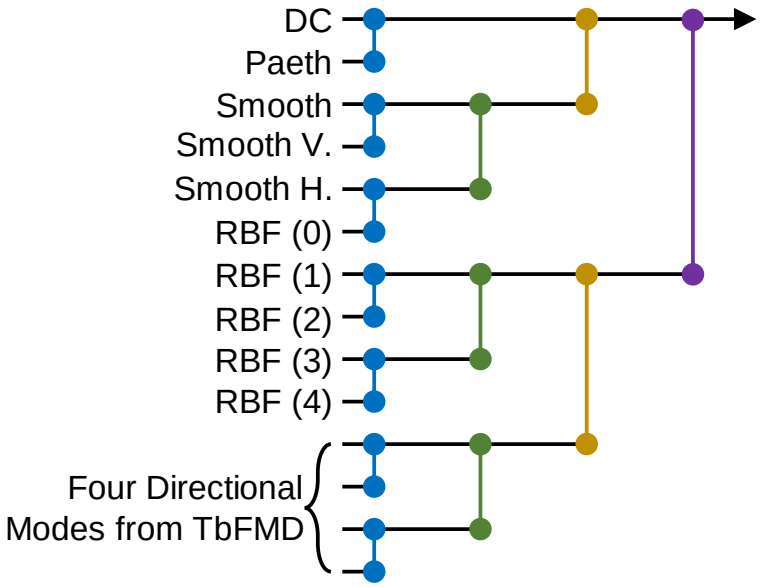

Figure 50 14:1 comparator with four levels of depth.

6.5.2 MaSBM Optimization

This optimization adds an extra control unit exclusive to the SSE trees, which is responsible for applying the operand isolation low-power technique to the inputs of each tree. Since each tree is dedicated to a single prediction mode, each can be controlled according to the subsampling mask of that specific mode.

Each position of a subsampling mask has only two states, on and off. For example, if an 8×8 block was to be compared with an SSE tree controlled by the subsampling mask shown in Figure 51: In cycle 1, only the first, sixth, and eighth

subtractor, and some operators from deeper levels of these branches would remain active, whilst the other five subtractors would receive zero as input. In cycle 2, only the fifth, seventh, and eighth subtractors would operate with real inputs. For the next six cycles, it can be observed that operations disabled in the mask are very likely to remain disabled, which makes the operand isolation effective in reducing dynamic switching, and in fact, in this example, two columns are never considered in the SSE calculation.

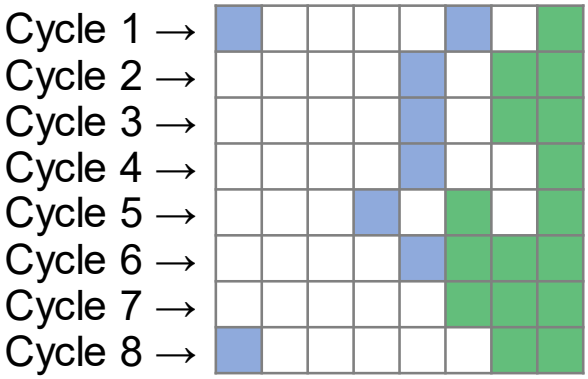

Figure 51 Example of 8×8 subsampling mask to be applied to an SSE tree.

It is also important to mention that for blocks smaller than 64 samples, the unused branches of an SSE tree also stay disabled using the same low-power technique.

For maximum optimization in terms of power reduction, the MaSBM parameters adopted are *HEA*=25% and *LEA*=50%. Therefore, with this degree of subsampling, only 25% of the predicted samples of each candidate will be considered in block matching.

An RTL diagram of the complete design, optimized with both TbFMD and MaSBM is illustrated in Figure 52. When compared to the base design shown in Figure 33, it can be observed that each of the 14 SSE trees is controlled individually by a MaSBM control unit, which also controls the comparison loop. This MaSBM unit receives the RD-list from TbFMD as input, which is important because: (i) although the first ten SSE trees are always used for the same non-directional modes, the last four SSE trees operate with varying subsets of the directional modes, and (ii) in case of smooth blocks, no directional candidates are generated and, in this case, the MaSBM control disables these trees completely.

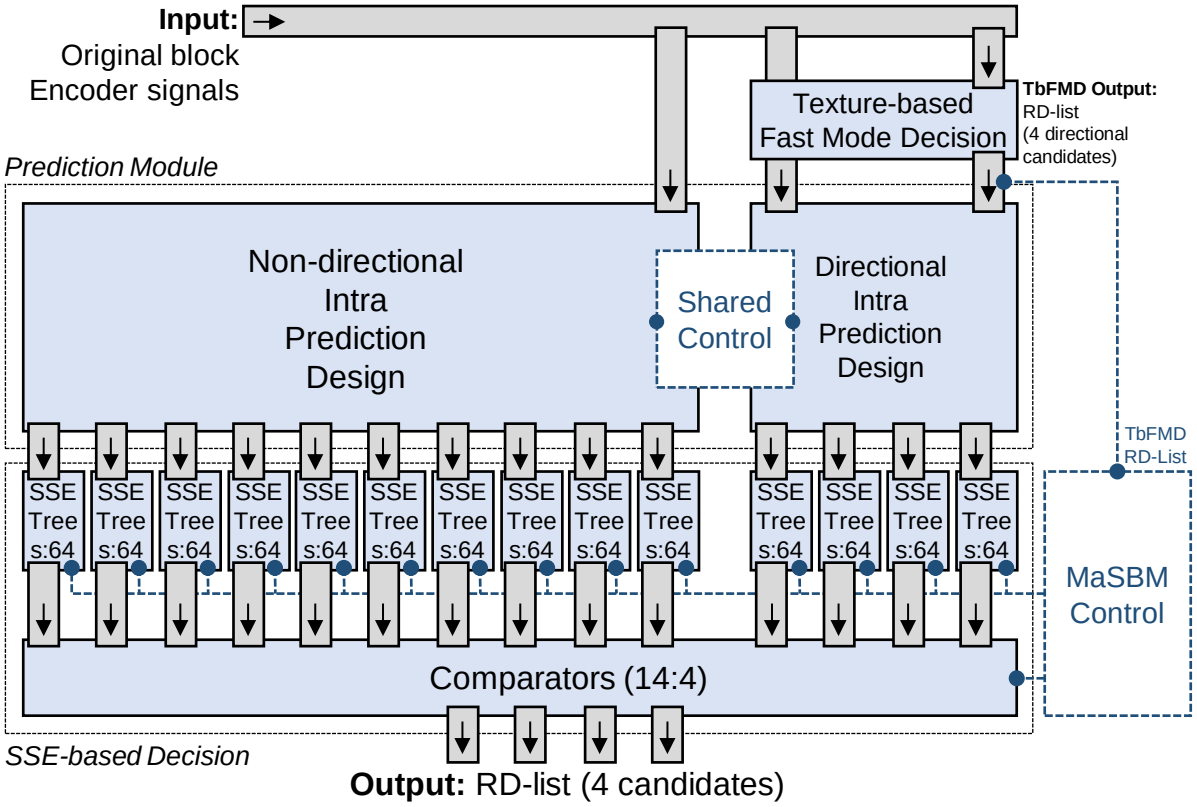

Figure 52 Intra prediction hardware design optimized with TdFMD and MaSBM.

6.6 Synthesis Results and Discussion

All hardware designs presented in the previous sections were fully described in Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL) and synthesized to the Taiwan Semiconductor Manufacturing Company (TSMC) 40nm standard-cells library, simulating the inputs using a switching rate of 20%.

The first intra prediction designs for the AV1 encoder in the literature were developed during this Ph.D. project, and to this date, there are no similar works in the literature. Thus, a fair comparison of the synthesis results with related works cannot be presented. However, intra prediction designs have been published in the past for older video coding formats. The works from Palomino *et al*. (2012), Zhou, Ding and Yu (2013), Pastuszak and Abramowski (2016), Fang, Chen and Chang (2016), Min, Xu, and Cheung (2017), and Corrêa *et al*. (2017) present intra prediction designs for the H.265 standard, whereas works from Huang *et al*. (2005), Jin, Jung and Lee (2007), Wang *et al*. (2007), Kuo and Lin (2008), Lin *et al*. (2009), Lin *et al*. (2010) present architectures for the H.264 standard.

It is important to mention that none of the architectures proposed for older formats can be used for an AV1 encoder since the algorithms and bitstream specifications are very different. Also, both the H.265 and H.264 intra prediction modules are much less complex than their AV1 counterpart. The H.265 intra prediction allows 35 modes (two of which are non-directional) and supports 4×4, 8×8, 16×16 and 32×32 block sizes, whereas the H.264 intra prediction allows nine modes (one non-directional) for the 4×4 block size and four modes (two non-directional) for the 16×16 block size.

Due to the abovementioned gap in complexity between AV1 and older codecs, and also because these designs use different technologies in the synthesis process, a fair comparison between these works is also not viable.

6.6.1 Results for the Base Intra Prediction Design

As mentioned in Section 6.3, the non-directional and directional intra prediction modules operating according to the shared control unit can predict all 1,869 possible subblocks inside a 64×64 SB in 7,108 clock cycles.

To allow real-time processing, three target throughputs were considered in this work: UHD 4K at 30 fps, FHD 1080p at 60 and 30 fps. Thus, for each case the processing of 91,124; 45,560 and 22,780 blocks of 64×64 are required per second, respectively, leading to target frequencies of 648, 324, and 162 MHz, respectively. The abovementioned target frequencies were calculated using eq. (28), where the multiplication by the constant 1.5 is done to consider the additional chrominance samples in the 4:2:0 color subsampling scheme, and the constant 4,096 represents the number of samples belonging to a 64×64 SB.

$$Freq = \frac{FrameWidth \times FrameHeight \times fps \times 7108 \times 1.5}{4096} \quad (28)$$

Since this work focuses on the intra prediction module only, the delay of the other encoder modules cannot be estimated. Therefore, the presented target frequencies consider the intra prediction modules only.

Furthermore, it is very important to mention that this means the proposed design implements a partition tree exploration that includes every single possibility, i.e., no fast decision for early termination is used, as this is not part of the scope of this work. Therefore, the following results are the worst-case scenario possible, since the use of

an early termination algorithm in block partitioning would, in average, result in a much lower number of cycles needed for each SB. There are no other known designs in the literature that implement a full exploration of the block partitioning tree.

Table 18, Table 19, Table 20, and Table 21 list the power dissipation, energy use, and gate count results for the unoptimized versions of the non-directional intra prediction module, directional intra prediction module, SSE-based decision module, and integrated base design (total), respectively.

Table 18 Synthesis results for the unoptimized non-directional intra prediction module

Target Throughput	Frequency (MHz)	Power (mW)	Energy (pJ/sample)	Area (2NAND KGates)
1080p at 30 fps	162	32.32	0.346	286.4
1080p at 60 fps	324	50.03	0.268	287.0
UHD 4K at 30 fps	648	75.33	0.202	348.5

Table 19 Synthesis results for the unoptimized directional intra prediction module

Target Throughput	Frequency (MHz)	Power (mW)	Energy (pJ/sample)	Area (2NAND KGates)
1080p at 30 fps	162	21.65	0.232	510.5
1080p at 60 fps	324	34.54	0.185	512.9
UHD 4K at 30 fps	648	79.53	0.213	601.9

Table 20 Synthesis results for the unoptimized SSE-based decision module

Target Throughput	Frequency (MHz)	Power (mW)	Energy (pJ/sample)	Area (2NAND KGates)
1080p at 30 fps	162	18.33	0.196	255.01
1080p at 60 fps	324	30.42	0.163	279.97
UHD 4K at 30 fps	648	51.08	0.137	323.14

Table 21 Synthesis results for the unoptimized base design (total)

Target Throughput	Frequency (MHz)	Power (mW)	Energy (pJ/sample)	Area (2NAND KGates)
1080p at 30 fps	162	72.30	0.774	1,051.91
1080p at 60 fps	324	114.99	0.616	1,079.87
UHD 4K at 30 fps	648	205.94	0.552	1,273.54

Table 18 and Table 19 show that, for the highest throughput target, the power dissipation and energy efficiency of the non-directional and directional prediction modes are roughly the same, although the directional intra prediction module required 72% more gates in its design.

Moreover, regarding the non-directional intra prediction module, it is relevant to observe that 69.2% of its power dissipation comes from the RBF Multifilter Unit alone. However, this result is acceptable, because this unit performs five of the ten non-

directional modes. On the other hand, this unit's gate count represents only 19% of the total, because of the massive subexpression reuse between different RBF modes. Furthermore, the RBF Multifilter Unit represents the critical path of the proposed design, due to its very high data dependency. The use of eight UMPUs connected serially results in a combinational critical path that is acceptable for the target throughput of UHD 4K at 30 fps, to increase the throughput even more, this unit would require UMPUs working in a pipelined manner, and the control unit would have to be redesigned to keep all units synchronized.

Regarding the directional intra prediction module, the power dissipation, energy efficiency, and gate count of all DMPUs for normal-sized arrays are close to the average, except of course for modes with horizontal, vertical, and diagonal angles, which do not require any interpolation. This result was expected since all the directional intra prediction modes follow one universal algorithm.

It can also be observed, in Table 21, that even though the power dissipation increases as the target frequency increases, the energy efficiency improves, since the energy use per predicted sample decreases.

Finally, regarding the SSE-based decision module, it can be observed that its cost is very high when compared to the total results. For the highest throughput target, this decision module alone requires 25.37% of the total number of gates and is responsible for 24.8% of the total power.

6.6.2 Results for the Optimized Intra Prediction Modules

The results presented in this section are divided into two distinct versions: one optimized only with TbFMD, and the other optimized with both TbFMD and MaSBM.

Table 22 lists the power dissipation, energy use, and gate count results for the directional intra prediction module optimized with TbFMD, to be compared directly against results from Table 19. It can be observed that, for the highest throughput target, the extra components of TbFMD increased the gate count by 17.1%, but reduced the power dissipated by this module by 88%, collaborating significantly with the energy efficiency of the whole design. Therefore, the TbFMD optimization offers an excellent trade-off of a small impact in gate count and an expressive gain in power reduction and energy efficiency.

Table 22 Synthesis results for the TbFMD-optimized directional intra prediction module

Target Throughput	Frequency (MHz)	Power (mW)	Energy (pJ/sample)	Area (2NAND KGates)
1080p at 30 fps	162	3.03	0.032	645.9
1080p at 60 fps	324	4.49	0.024	632.9
UHD 4K at 30 fps	648	9.54	0.026	734.9

Table 23 lists the results for the SSE-based decision optimized by the TbFMD, to be compared directly against Table 20. The use of TbFMD reduces the number of total candidates from 66 to 14, in the worst-case scenario, allowing for a much smaller number of SSE trees and a simpler SSE comparator. It can be observed that, for the highest throughput target, the downsizing of this module resulted in a reduction of 78% in power dissipated and 71% in the number of gates, which makes this decision module more viable than its unoptimized counterpart.

Table 23 Synthesis results for the TbFMD-optimized SSE-based decision module

Target Throughput	Frequency (MHz)	Power (mW)	Energy (pJ/sample)	Area (2NAND KGates)
1080p at 30 fps	162	3.84	0.041	64.75
1080p at 60 fps	324	6.08	0.033	75.59
UHD 4K at 30 fps	648	11.23	0.030	93.71

Table 24 also lists the results for the SSE-based decision, but optimized with both TbFMD and MaSBM. The use of MaSBM adds extra control components to the array of SSE trees to reduce its power dissipated with the low-power operand isolation technique, based on subsampling masks. It can be observed that when compared to the TbFMD-optimized version (Table 23), a reduction of 60% in power dissipated was achieved at the cost of an increase of 21% in the number of gates. When compared to the unoptimized version (Table 20), it achieved a very expressive power reduction of 91.2% and a reduction of 71% in the number of gates.

Table 24 Synthesis results for the TbFMD-MaSBM-optimized SSE-based decision module

Target Throughput	Frequency (MHz)	Power (mW)	Energy (pJ/sample)	Area (2NAND KGates)
1080p at 30 fps	162	1.21	0.013	77.05
1080p at 60 fps	324	2.55	0.014	89.95
UHD 4K at 30 fps	648	4.49	0.012	113.39

Table 25 lists the synthesis results for the complete and fully-optimized intra prediction design developed during this Ph.D. project, to be directly compared against the unoptimized version from Table 21. For the highest throughput target, a modest reduction of 6% in the number of gates allowed for a very significant reduction of 56.6% in total power dissipated. In this final design, it is important to note that most of the power dissipated comes from the non-directional intra prediction design, which is not optimized in any way by any of the proposed algorithms.

Table 25 Synthesis results for fully-optimized design (total)

Target Throughput	Frequency (MHz)	Power (mW)	Energy (pJ/sample)	Area (2NAND KGates)
1080p at 30 fps	162	36.56	0.392	1,009.35
1080p at 60 fps	324	57.07	0.306	1,009.85
UHD 4K at 30 fps	648	89.36	0.239	1,196.79

And, finally, Table 26 compares the unoptimized base design from Table 21 and the fully optimized design from Table 25 in terms of total power, dynamic power, and leakage power. In this comparison, the highest target throughput (UHD 4K at 30 fps) was considered. In the unoptimized design, a leakage power of 3.52 mW was observed, representing 1.71% of the total power. In the fully-optimized design, however, after the removal of an expressive number of SSE trees and after the insertion of the TbFMD and MaSBM modules, a leakage power of 3.41 mW was observed, representing a total of 3.82% of the total power. By comparing the dynamic power of both designs, it can be seen that the optimization strategies significantly reduced the switching activity of the circuit.

Table 26 Detailed power results for the unoptimized and fully optimized designs

Design	Total Power	Dynamic Power	Leakage Power
Unoptimized base design	205.94 mW	202.42 mW (98.29%)	3.52 mW (1.71%)
Fully optimized design	89.36 mW	85.95 mW (96.18%)	3.41 mW (3.82%)

7 CONCLUSIONS

In this chapter, this thesis is concluded with a summary of the main achievements, as well as some directions for future research.

The research work developed in the scope of this thesis contributed to novel solutions for reducing the number of operations in the intra prediction stage of modern video encoders. The study included extensive experiments using the AV1 video format, which was still under development when this Ph.D. project started.

At the beginning of the research work presented in this thesis, there was no work published in the technical literature focusing on the new algorithms introduced in AV1 during its development, and studies of compression efficiency and computational complexity were not available either. This way, this project started with the extensive study of the reference software during its development, of draft documents and discussion boards, which led to a major contribution of this project: the first combined overview of the AV1 and VVC features, algorithmic solutions, and hardware designs (CORRÊA *et al*., 2021).

As soon as the first release version of the AV1 specification was published, various contributions were made in the form of hardware implementations of the intra prediction algorithms (CORRÊA *et al*., 2019a; 2019b; 2020a; 2020b, NETO *et al*., 2020; 2021a; 2021b; 2022).

With enough parts of the intra prediction module of the encoder developed in hardware, the focus of this project shifted towards creating heuristic-based algorithms that could be used in either software or hardware solutions. This study led to the algorithmic contributions of this project, a fast mode decision algorithm that evaluates the characteristics of input block texture to reduce the number of intra prediction modes executed (CORRÊA *et al*., 2022b), and a mode-adaptive subsampling algorithm to reduce the number of arithmetic operations in block matching (CORRÊA *et al*., 2022a).

After achieving very positive results in the algorithm development, the focus of this project shifted back to hardware development, and two designs for the heuristic-based algorithms were made and integrated into the intra prediction modules that were developed in the early stages of this project (CORRÊA *et al*., 2022b).

The software results were very satisfactory in terms of encoding efficiency and encoding time and, on average, the algorithms showed their best performance for UHD resolutions. The hardware results were also satisfactory in terms of area, power, and

energy efficiency, the latter being paramount in a world overwhelmed by battery-powered devices.

Considering all the positive results presented in this work, it is concluded that the thesis proposed in this project, *“It is possible to reduce the computational effort of the AV1 intra-picture prediction by developing hardware-friendly heuristic-based algorithms and, then, generate efficient hardware designs able to process ultra-high-definition videos in real-time.”,* proved to be valid.

In future works, there are two clear paths to be followed:

1. The adaptation of the algorithms presented in this work to fit other state-of-the-art video formats, such as VVC, which comprises algorithms that are very different from the ones in AV1, but still follows the same block-based prediction in the spatial domain.
2. Further optimization of the MaSBM algorithm in the form of also applying the subsampling masks to the prediction process of candidates. In other words, if only a subset of the predicted samples is needed per candidate for block matching, that means that only that subset needs to be predicted, with a full prediction occurring only for the candidates selected to be passed forward to the RDO process. This should lead to massive gains in the intra prediction stage itself, as the number of arithmetic operations needed for the prediction of candidates would be reduced proportionally to the degree of subsampling used.
3. The replacement of the SSE metric in the MaSBM in favor of the less demanding SAD metric.
4. The development of a heuristic-based algorithm capable of optimizing the non-directional intra prediction stage of the encoder, which was not affected by any of the algorithms proposed in this work.

REFERENCES

AGOSTINI, L. **Desenvolvimento de Arquiteturas de Alto Desempenho Dedicadas à Compressão de Vídeo Segundo o Padrão H.264/AVC.** 2007. Thesis (Ph.D.) – Programa de Pós-Graduação em Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2007.

AOMedia. AV1 Codec Library. Available at <https://aomedia.googlesource.com/aom/>. 2022. Accessed in: January 10th, 2023.

BANKOSKI, J. *et al*. RFC 6386 VP8 Data Format and Decoding Guide. Available at <https://datatracker.ietf.org/doc/rfc6386/>. 2011. Accessed in: January 10th, 2023.

BANKOSKI, J.; WILKINS, P.; XU, Y. Technical overview of VP8, an open source video codec for the web. In: IEEE International Conference on Multimedia and Expo, Barcelona, 2011. **Proceedings:** IEEE, 2011. doi: 10.1109/ICME.2011.6012227

BEBENITA, M. AV1 Bitstream Analyzer. Available at <https://medium.com/hackernoon/av1-bitstream-analyzer-d25f1c27072b>. 2017. Accessed in: January 10th, 2023.

BITENCOURT, T.; RAMOS, F.; BAMPI, S. High-Throughput and Low-Power Architectures for the AV1 Arithmetic Encoder. In: 2021 34th SBC/SBMicro/IEEE/ACM Symposium on Integrated Circuits and Systems Design (SBCCI), 2021, Campinas. **Proceedings:** IEEE, 2021. doi: 10.1109/SBCCI53441.2021.9529994

BITENCOURT, T.; RAMOS, F.; BAMPI, S. Power-Saving 8K Real-Time AV1 Arithmetic Encoder Architecture. **IEEE Design & Test**, v.39, n.6, p.128-137, 2022. doi: 10.1109/MDAT.2022.3184625

BJØNTEGAARD, G. **VCEG-M33:** Calculation of average PSNR differences between RD-curves. Austin, 2001. Available at <https://www.itu.int/wftp3/av-arch/video-site/0104_Aus/VCEG-M33.doc>. Accessed in: January 10th, 2023.

BJØNTEGAARD, G. *et al*. The Thor Video Codec. In: 2016 Data Compression Conference (DCC)*,* 2016, pp. 476-485, doi: 10.1109/DCC.2016.74

BOSSEN, F. *et al*. **JVET-W0003:** AHG report: Test Model Software Development (AHG3). Teleconference, 2021. Available at <https://jvet-experts.org/>. Accessed in: January 10th, 2023.

BROSS, B. *et al*. Overview of the Versatile Video Coding (VVC) Standard and its Applications. **IEEE Transactions on Circuits and Systems for Video Technology**, v.31, n.10, p.3736-3764, 2021. doi: 10.1109/TCSVT.2021.3101953

CHEN, J. *et al*. **JCTVC-E266:** CE6.a.4: Chroma intra prediction by reconstructed luma samples. Geneva, 2011. Available at <https://www.itu.int/wftp3/av-arch/jctvc-site/2011_03_E_Geneva/>. Accessed in: January 10th, 2023.

CHEN, X. *et al*. A Conditional Bayesian Block Structure Inference Model for Optimized AV1 Encoding. In: 2019 IEEE International Conference on Multimedia and Expo (ICME), 2019, Shanghai. **Proceedings:** IEEE, 2019. doi: 10.1109/ICME.2019.00221

CHIANG, C.; HAN, J.; XU, Y. A Multi-Pass Coding Mode Search Framework For AV1 Encoder Optimization. In: 2019 Data Compression Conference (DCC), 2019, Snowbird. **Proceedings:** IEEE, 2019. doi: 10.1109/DCC.2019.00054

CHUANG, H.; LEI, Z.; OPALACH, A.; NORKIN, A. Analysis of AV1 coding tools. In: SPIE Optical Engineering + Applications, 2022, San Diego. **Proceedings SPIE 12226:** SPIE, 2016. doi: 10.1117/12.2635956

CISCO SYSTEMS INC. Cisco Annual Internet Report (2018–2023) White Paper. Available at: <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html>. 2020. Accessed in: January 10th, 2023.

CORRÊA, G. **Computational Complexity Reduction and Scaling for High Efficiency Video Encoders.** 2014. Thesis (Ph.D.) – Departamento de Engenharia Eletrotécnica e de Computadores, Universidade de Coimbra, Coimbra, 2014.

CORRÊA, G. *et al*. **Complexity-Aware High Efficiency Video Coding**. Cham, Springer, 2016. doi: 10.1007/978-3-319-25778-5

CORRÊA, M.; ZATT, B.; PORTO, M.; AGOSTINI, L. High-throughput HEVC intrapicture prediction hardware design targeting UHD 8K videos. In: 2017 IEEE International Symposium on Circuits and Systems (ISCAS), 2017, Baltimore. **Proceedings:** IEEE, 2017. doi: 10.1109/ISCAS.2017.8050702

CORRÊA, M. *et al*. A High Throughput Hardware Architecture Targeting the AV1 Paeth Intra Predictor. In: 2019 IEEE 10th Latin American Symposium on Circuits & Systems (LASCAS), 2019, Armenia. **Proceedings:** IEEE, 2019. doi: 10.1109/LASCAS.2019.8667544

CORRÊA, M. *et al*. High Throughput Hardware Design for AV1 Paeth and Smooth Intra Modes. In: 2019 IEEE International Symposium on Circuits and Systems (ISCAS), 2019, Sapporo. **Proceedings:** IEEE, 2019. doi: 10.1109/ISCAS.2019.8702258

CORRÊA, M. *et al*. A High-Throughput Hardware Architecture for AV1 Non-Directional Intra Modes. **IEEE Transactions on Circuits and Systems I: Regular Papers**, v.67, n.5, p.1481-1494, 2020.

CORRÊA, M. *et al*. ASIC Solution for the Directional Intra Prediction of the AV1 Encoder Targeting UHD 4K Videos. In: 2020 IEEE International Symposium on Circuits and Systems (ISCAS), 2020, Sapporo. **Proceedings:** IEEE, 2020. doi: 10.1109/ISCAS45731.2020.9180526

CORRÊA, M. *et al*. AV1 and VVC Video Codecs: Overview on Complexity Reduction and Hardware Design. **IEEE Open Journal of Circuits and Systems**, v.2, p.564-576, 2021. doi: 10.1109/OJCAS.2021.3107254

CORRÊA, M. *et al*. Mode-Adaptive Subsampling of SAD/SSE Operations for Intra Prediction Cost Reduction. In: 2022 IEEE International Symposium on Circuits and Systems (ISCAS), 2022, Austin. Proceedings: IEEE, 2022. doi: 10.1109/ISCAS48785.2022.9937507

CORRÊA, M.; PALOMINO, D.; CORRÊA, G.; AGOSTINI, L. Direction-Based Fast Mode Decision and Hardware Design for the AV1 Intra Prediction. In: Symposium on Integrated Circuits and Systems Design (SBCCI), 2022, Porto Alegre. **Proceedings:** IEEE, 2022. doi: 10.1109/SBCCI55532.2022.9893253

DEMPSTER, A.; MACLEOD, M. Use of minimum-adder multiplier blocks in FIR digital filters. **IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing**, v.42, n.9, p.569-577, 1995. doi: 10.1109/82.466647

DOMANSKI, R. *et al*. High-Throughput Multifilter Interpolation Architecture for AV1 Motion Compensation. **IEEE Transactions on Circuits and Systems II: Express Briefs**, v.66, n.5, p.883-887, 2019. doi: 10.1109/TCSII.2019.2909705

DOMANSKI, R. *et al*. Low-Power and High-Throughput Approximated Architecture for AV1 FME Interpolation. In: 2021 IEEE International Symposium on Circuits and Systems (ISCAS), 2021, Daegu. **Proceedings:** IEEE, 2021. doi: 10.1109/ISCAS51556.2021.9401224

FANG, H.; CHEN, H.; CHANG, T. Fast intra prediction algorithm and design for high efficiency video coding. In: IEEE International Symposium on Circuits and Systems (ISCAS), 2016, Montreal. **Proceedings:** IEEE, 2016. doi: 10.1109/ISCAS.2016.7538911

FREITAS, D. *et al*. Hardware Architecture for the Regular Interpolation Filter of the AV1 Video Coding Standard. In: 2020 28th European Signal Processing Conference (EUSIPCO), 2020, Amsterdam. **Proceedings:** IEEE, 2020. doi: 10.23919/Eusipco47968.2020.9287551

FREITAS, D.; DINIZ, C.; GRELLERT, M.; CORRÊA, G. High-Throughput Sharp Interpolation Filter Hardware Architecture for the AV1 Video Codec. In: 2021 34th SBC/SBMicro/IEEE/ACM Symposium on Integrated Circuits and Systems Design (SBCCI), 2021, Campinas. Proceedings: IEEE, 2021. doi: 10.1109/SBCCI53441.2021.9529993

FREITAS, D. et al. High-Throughput Multifilter VLSI Design for the AV1 Fractional Motion Estimation. In: 2022 35th SBC/SBMicro/IEEE/ACM Symposium on Integrated Circuits and Systems Design (SBCCI), 2022, Porto Alegre. Proceedings: IEEE, 2022 doi: 10.1109/SBCCI55532.2022.9893255

GHANBARI, M. **Standard Codecs:** Image compression to advanced video coding. 3rd ed. London: IET, 2011. doi: 10.1049/PBTE054E

GOEBEL, J.; ZATT, B.; AGOSTINI, L.; PORTO, M. Hardware Design of DC/CFL Intra-Prediction Decoder for the AV1 Codec. In: 32nd Symposium on Integrated Circuits and Systems Design (SBCCI), 2019, São Paulo. **Proceedings:** IEEE, 2019.

GOMES, J.; RAMOS, F. High-Performance Design for the AV1 Multi - Alphabet Arithmetic Decoder. In: 2021 34th SBC/SBMicro/IEEE/ACM Symposium on Integrated Circuits and Systems Design (SBCCI), 2021, Campinas. **Proceedings:** IEEE, 2021. doi: 10.1109/SBCCI53441.2021.9529970

GRANGE, A.; RIVAZ, P.; HUNT, J. VP9 Bitstream & Decoding Process Specification. Available at <https://www.webmproject.org/vp9/>. 2016. Accessed in: January 10th, 2023.

GUO. B. *et al*. A Bayesian Approach to Block Structure Inference in AV1-Based Multi-Rate Video Encoding. In: 2018 Data Compression Conference (DCC), 2018, Snowbird. doi: 10.1109/DCC.2018.00047

GUO. B.; Y. HAN, Y.; WEN, J. Fast Block Structure Determination in Av1-Based Multiple Resolutions Video Encoding. In: 2018 IEEE International Conference on Multimedia and Expo (ICME), 2018, San Diego. doi: 10.1109/ICME.2018.8486492

GUO, L. *et al*. Color palette for screen content coding. In: IEEE International Conference on Image Processing (ICIP), 2014, Paris. **Proceedings:** IEEE, 2014. p.5556-5560. doi: 10.1109/ICIP.2014.7026124

HAN, J. *et al*. A Technical Overview of AV1. **Proceedings of the IEEE***,* v.109, n.9, p.1435-1462, 2021. doi: 10.1109/JPROC.2021.3058584

HUANG, Y.; HSIEH, B.; CHEN, T.; CHEN, L. Analysis, fast algorithm, and VLSI architecture design for H.264/AVC intra frame coder. **IEEE Transactions on Circuits and Systems for Video Technology**, v.15, n.3, p.378-401, 2005. doi: 10.1109/TCSVT.2004.842620

ITU-T. **Recommendation H.262:** Information technology – Generic coding of moving pictures and associated audio information: Video (07/95). 1995. 211p.

ITU-T. **Recommendation H.264:** Advanced video coding for generic audiovisual services (05/03). 2003. 282p.

ITU-T. **Recommendation H.265:** High efficiency video coding (04/13). 2013. 317p.

ITU-T. **Recommendation H.266:** Versatile video coding. 2020. 516p.

ITU-T. **Recommendation ITU-R BT.500-14:** Methodologies for the subjective assessment of the quality of television images (10/2019). 2019. 102p.

JEONG, J.; GANKHUYAG, G.; KIM, Y. Fast Chroma Prediction Mode Decision based on Luma Prediction Mode for AV1 Intra Coding. In: International Conference on Information and Communication Technology Convergence (ICTC), 2019, Jeju. **Proceedings:** IEEE, 2019. doi: 10.1109/ICTC46691.2019.8939936

JEONG, J.; GANKHUYAG, G.; KIM, Y. A Fast Intra Mode Decision Based on Accuracy of Rate Distortion Model for AV1 Intra Encoding. In: 34th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), Jeju, 2019. **Proceedings:** IEEE, 2019. doi: 10.1109/ICTC46691.2019.8939936

JIN, G.; JUNG, J.; LEE, H. An Efficient Pipelined Architecture for H.264/AVC Intra Frame Processing. In: IEEE International Symposium on Circuits and Systems (ISCAS), 2007, New Orleans. **Proceedings:** IEEE, 2019. doi: 10.1109/ISCAS.2007.378825

KARCZEWICZ, M.; YE, Y. **JVET-W2017-v1:** Common test conditions and evaluation procedures for enhanced compression tool testing. Teleconference, 2021. Available at <https://jvet-experts.org/doc_end_user/documents/23_Teleconference/wg11/JVET-W2017-v1.zip>. Accessed in: January 10th, 2023.

KIM, J. *et al*. Fast Inter-prediction Based on Decision Trees for AV1 Encoding. In: 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019, Brighton. **Proceedings:** IEEE, 2019. doi: 10.1109/ICASSP.2019.8683580

KUO, H.; LIN, Y. An H.264/AVC full-mode intra-frame encoder for 1080HD video. In: IEEE International Conference on Multimedia and Expo (ICME), 2008, Hannover. **Proceedings:** IEEE, 2008. doi: 10.1109/ICME.2008.4607615

LI, J. *et al*. Intra Block Copy for Screen Content in the Emerging AV1 Video Codec. In: Data Compression Conference (DCC), 2018, Snowbird. **Proceedings:** IEEE, 2018. p.355-364. doi: 10.1109/DCC.2018.00044

LIN, Y.; KU, C.; LI, D.; CHANG, T. A 140-MHz 94 K Gates HD1080p 30-Frames/s Intra-Only Profile H.264 Encoder. **IEEE Transactions on Circuits and Systems for Video Technology**, v.19, n.3, p.432-436, 2009. doi: 10.1109/TCSVT.2009.2013511

LIN, H.; WU, K.; LIU, B; YANG, J. An Efficient VLSI Architecture for Transform-Based Intra Prediction in H.264/AVC. **IEEE Transactions on Circuits and Systems for Video Technology**, v.20, n.6, p.894-906, 2010. doi: 10.1109/TCSVT.2010.2046059

LIN, J. *et al*. A fusion-based video quality assessment (fvqa) index. In: Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2014, Siem Reap. **Proceedings:** IEEE, 2014. doi: 10.1109/APSIPA.2014.7041705

LIU, T.; LIN, Y.; LIN, W.; KUO, C. Visual quality assessment: Recent developments, coding applications and future trends. **APSIPA Transactions on Signal and Information Processing**, v.2, e.4, 2013. doi:10.1017/ATSIP.2013.5

MIDTSKOGEN, S.; VALIN, J. The Av1 Constrained Directional Enhancement Filter (Cdef). In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018, Calgary. **Proceedings:** IEEE, 2018. p.1193-1197. doi: 10.1109/ICASSP.2018.8462021

MIN, B.; XU, Z.; CHEUNG, R. A Fully Pipelined Hardware Architecture for Intra Prediction of HEVC. **IEEE Transactions on Circuits and Systems for Video Technology**, v.27, n.12, p.2702-2713, 2017. doi: 10.1109/TCSVT.2016.2593618

MUKHERJEE, D. *et al*. A Technical Overview of VP9—The Latest Open-Source Video Codec. **SMPTE Motion Imaging Journal**, v.124, n.1, p.44-54, 2015. doi: 10.5594/j18499

MUKHERJEE, D. *et al*. A switchable loop-restoration with side-information framework for the emerging AV1 video codec. In: IEEE International Conference on Image Processing (ICIP), 2017, Beijing. **Proceedings:** IEEE, 2017. p.265-269. doi: 10.1109/ICIP.2017.8296284

NETFLIX INC. VMAF - Video Multi-Method Assessment Fusion. Available at <https://github.com/Netflix/vmaf>. 2022. Accessed in: January 10th, 2023.

NETFLIX INC. Toward A Practical Perceptual Video Quality Metric. Available at <https://netflixtechblog.com/toward-a-practical-perceptual-video-quality-metric-653f208b9652>. 2022. Accessed in: January 10th, 2023.

NETO, L. *et al*. Directional Intra Frame Prediction Architecture with Edge Filter and Upsampling for AV1 Video Coding. In: Symposium on Integrated Circuits and Systems Design (SBCCI), 2020, Campinas. **Proceedings:** IEEE, 2020. doi: 10.1109/SBCCI50935.2020.9189902

NETO, L. *et al*. Exploring Operation Sharing in Directional Intra Frame Prediction of AV1 Video Coding. In: IEEE Latin America Symposium on Circuits and System (LASCAS), 2021, Arequipa. **Proceedings:** IEEE, 2021. doi: 10.1109/LASCAS51355.2021.9459136

NETO, L. *et al*. Configurable Power/Quality-Aware Hardware Design for the AV1 Directional Intra Frame Prediction. In*:* 2021 34th SBC/SBMicro/IEEE/ACM Symposium on Integrated Circuits and Systems Design (SBCCI), 2021, Campinas. **Proceedings:** IEEE, 2021. doi: 10.1109/SBCCI53441.2021.9529997

NETO, L. *et al*. Power-Quality Configurable Hardware Design for AV1 Directional Intra-Frame Prediction. **IEEE Design & Test**, v.39, n.2, p.38-45, 2022. doi: 10.1109/MDAT.2022.3146083

NGUYEN, T.; MARPE, D. Future Video Coding Technologies: A Performance Evaluation of AV1, JEM, VP9, and HM. In: Picture Coding Symposium (PCS), 2018, San Francisco. **Proceedings:** IEEE, 2018. p.31-35. doi: 10.1109/PCS.2018.8456289

PALAU, R. *et al*. An UHD 4K@60fps Dual Self-Guided Filter Targeting the AV1 Decoder. In: 2022 35th SBC/SBMicro/IEEE/ACM Symposium on Integrated Circuits and Systems Design (SBCCI), 2022, Porto Alegre. **Proceedings:** IEEE, 2022. doi: 10.1109/SBCCI55532.2022.9893236

PALAU, R. *et al*. Hardware Design for the Separable Symmetric Normalized Wiener Filter of the AV1 Decoder. In: 2022 35th SBC/SBMicro/IEEE/ACM Symposium on Integrated Circuits and Systems Design (SBCCI), 2022, Porto Alegre. **Proceedings:** IEEE, 2022. doi: 10.1109/SBCCI55532.2022.9893219

PALOMINO, D. *et al*. A memory aware and multiplierless VLSI architecture for the complete intra prediction of the HEVC emerging standard. In: in IEEE International Conference on Image Processing (ICIP), 2012, Orlando. **Proceedings:** IEEE, 2012. doi: 10.1109/ICIP.2012.6466830

PARKER, S. *et al*. On transform coding tools under development for VP10. In: SPIE Optical Engineering + Applications, 2016, San Diego. **Proceedings SPIE 9971 997119:** SPIE, 2016. doi: 10.1117/12.2239105

PASTUSZAK, G.; ABRAMOWSKI, A. Algorithm and architecture design of the H.265/HEVC intra encoder. **IEEE Transactions on Circuits and Systems for Video Technology**, v.26, n.1, p.210-222, 2016. doi: 10.1109/TCSVT.2015.2428571

PU, W. *et al*. **JCTVC-N0266:** Non RCE1: Inter Color Component Residual Prediction. Vienna, 2013. Available at <http://phenix.it-sudparis.eu/jct/doc_end_user/current_document.php?id=7982>. Accessed in: January 10th, 2023.

RICHARDSON, I. **The H.264 advanced video compression standard**. 2nd ed. Chichester: Wiley, 2010.

RIVAZ, P.; HAUGHTON, J. AV1 Bitstream & Decoding Process Specification v.1.0.0-errata1. Available at <https://aomediacodec.github.io/av1-spec/av1-spec.pdf>. 2019. Accessed in: January 10th, 2023.

SALDANHA, M. *et al*. Complexity Analysis of VVC Intra Coding. In: IEEE International Conference on Image Processing (ICIP), 2020, Abu Dhabi. **Proceedings:** IEEE, 2020. p.3119-3123. doi: 10.1109/ICIP40778.2020.9190970

SALDANHA, M. **Exploration of Encoding Time Reduction Solutions for Intra-Frame Prediction of VVC Encoders.** 2021. Thesis (Ph.D.) – Programa de Pós-Graduação em Computação, Universidade Federal de Pelotas, Pelotas, 2021.

SULLIVAN, G.; WIEGAND, T. Rate-distortion optimization for video compression. **IEEE Signal Processing Magazine**, v.15, n.6, p.74-90, 1998. doi: 10.1109/79.733497

SULLIVAN, G. *et al*. Overview of the High Efficiency Video Coding (HEVC) Standard. **IEEE Transactions on Circuits and Systems for Video Technology**, v.22, n.12, p.1649-1668, 2012. doi: 10.1109/TCSVT.2012.2221191

TRUDEAU, L.; EGGE, N.; BARR, D. Predicting Chroma from Luma in AV1. In: Data Compression Conference (DCC), 2018, Snowbird. **Proceedings:** IEEE, 2018. p.374-382. doi: 10.1109/DCC.2018.00046

VALIN, J. *et al*. Daala: A perceptually-driven still picture codec. In: IEEE International Conference on Image Processing (ICIP), 2016, Phoenix. **Proceedings:** IEEE, 2016. p.76-80. doi: 10.1109/ICIP.2016.7532322

VORONENKO, Y.; PÜSCHEL, M. Multiplierless multiple constant multiplication. **ACM Transactions on Algorithms**, v.3, n.2, 2007. doi: 10.1145/1240233.1240234

WANG, J.; WANG, J; YANG, J.; CHEN, J. A Fast Mode Decision Algorithm and Its VLSI Design for H.264/AVC Intra-Prediction. **IEEE Transactions on Circuits and Systems for Video Technology**, v.17, n.10, p.1414-1422, 2007. doi: 10.1109/TCSVT.2007.903786

WIEGAND, T. *et al*. Overview of the H.264/AVC video coding standard. **IEEE Transactions on Circuits and Systems for Video Technology**, v.13, n.7, p.560-576, 2003. doi: 10.1109/TCSVT.2003.815165

XIPH.ORG FOUNDATION. AOM Analyzer. Available at <https://github.com/xiph/aomanalyzer>. 2022. Accessed in: January 10th, 2023.

XU, X.; LIU, S. Overview of Screen Content Coding in Recently Developed Video Coding Standards. **IEEE Transactions on Circuits and Systems for Video Technology**, v.32, n.2, p.839-852, 2022, doi: 10.1109/TCSVT.2021.3064210

ZHAO, X. *et al*. **CWG-B075o:** AOM Common Test Conditions v2.0. 2021. Available at < https://aomedia.org/docs/CWG-B075o_AV2_CTC_v2.pdf>. Accessed in: January 10th, 2023.

ZHOU, N.; DING, D.; YU, L. On hardware architecture and processing order of HEVC intra prediction module. In: in Picture Coding Symposium (PCS), 2013, San Jose. **Proceedings:** IEEE, 2013. doi: 10.1109/PCS.2013.6737693

ZUMMACH, E. *et al*. High-Throughput CDEF Architecture for the AV1 Decoder Targeting 4K@60fps Videos. In: 2020 IEEE 11th Latin American Symposium on Circuits & Systems (LASCAS), 2020, San Jose. **Proceedings:** IEEE, 2020. doi: 10.1109/LASCAS45839.2020.9068979

ZUMMACH, E. *et al*. Efficient Hardware Design for the AV1 CDEF Filter Targeting 4K UHD Videos. In: 2020 IEEE International Symposium on Circuits and Systems (ISCAS), 2020, Seville. **Proceedings:** IEEE, 2020. doi: 10.1109/ISCAS45731.2020.9180525

ZUMMACH, E. *et al*. An UHD 4K@60fps Deblocking Filter Hardware Targeting the AV1 Decoder. In: 2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS), 2020, Glasgow. **Proceedings:** IEEE, 2020. doi: 10.1109/ICECS49266.2020.9294930

Appendices

Appendix A – Experimental Setup

When this research project started, the AV1 video format was still in stage of development, and there were no official guidelines on how to conduct experiments with it. On the other hand, common test conditions for experiments with the MPEG and VCEG standardized formats were available and in a very mature stage.

Because of that, the experiments conducted in this research project used a set of test sequences recommended by the Joint Video Experts Team (JVET), in the JVET-W2017-v1 document (KARCZEWICZ; YE, 2021), known as Common Test Conditions (CTC). According to the CTC, the Classes “D”, “F” and “Text and Graphics with Motion” (TGM) are not to be included by default in the test results, unless low resolutions are desired (class D with 240p videos) or digital screen content (classes F and TGM). Therefore, those three classes were left out of the experiments of this Ph.D. project. Table 27 lists the test sequences used.

Table 27 Test sequences used in the experiments conducted

Class	Resolution	Sequence	Frame Count	Frame Rate (fps)	Bit Depth
A1	3840×2160 (UHD 4K)	*Tango2*	294	60	10
		FoodMarket4	300	60	10
		Campfire	300	30	10
A2	3840×2160 (UHD 4K)	*CatRobot*	300	60	10
		DaylightRoad2	300	60	10
		ParkRunning3	300	50	10
B	1920×1080 (1080p)	*MarketPlace*	600	60	10
		RitualDance	600	60	10
		Cactus	500	50	8
		BasketballDrive	500	50	8
		BQTerrace	500	60	8
C	832×480 (480p)	*BasketballDrill*	500	50	8
		BQMall	600	60	8
		PartyScene	500	50	8
		RaceHorses	300	30	8
E	1280×720 (720p)	*FourPeople*	600	60	8
		Johnny	600	60	8
		KristenAndSara	600	60	8

When a CTC document got released by AOMedia (ZHAO, 2021), the parameters used in *libaom* 2.0.0 for the intra prediction experiments became:

- --cpu-used=0 (Slowest speed setting)
- --passes=1 (Single pass)
- --end-usage=q (Rate-control based in QP)
- --cq=x (Sets the QP as *x*)

- --kf-min-dist=0 (Disables inter prediction)
- --kf-max-dist=0 (Disables inter prediction)
- --enable-tpl-model=0 (Disables this feature)
- --enable-keyframe-filtering=0 (Disables this feature)
- --deltaq-mode=0 (Disables this feature)

Also, according to the AOMedia CTC, all the encoding quality results presented in the form of BD-BRYUV in this thesis are computed using the frame-averaged PSNRYUV, where the PSNR for each channel is computed separately and are then combined according to eq. (29).

$$PSNR_{YUV} = \frac{14 \times PSNR_Y + PSNR_U + PSNR_V}{16} \quad (29)$$

Appendix B – List of Published Papers During the Ph.D. Studies

The Ph.D. project presented in this thesis resulted in 16 peer-reviewed publications, which are listed below. Publications in small regional conferences were omitted purposely.

B.1. Journal Articles

I. PORTO, R.; **CORRÊA, M.**; GOEBEL, J.; ZATT, B.; ROMA, N.; AGOSTINI, L.; PORTO, M. UHD 8K energy-quality scalable HEVC intra-prediction SAD unit hardware using optimized and configurable imprecise adders. **Journal of Real-Time Image Processing**, v.17, p.1685-1701, 2020. doi: 10.1007/s11554-019-00934-2

II. **CORRÊA, M.**; WASKOW, B.; GOEBEL, J.; PALOMINO, D.; CORRÊA, G.; AGOSTINI, L. A High-Throughput Hardware Architecture for AV1 Non-Directional Intra Modes. **IEEE Transactions on Circuits and Systems I: Regular Papers**, v.67, n.5, p.1481-1494, 2020. doi: 10.1109/TCSI.2020.2973031

III. **CORRÊA, M.**; SALDANHA, M.; BORGES, A.; CORRÊA, G.; PALOMINO, D.; PORTO, M.; ZATT, B.; AGOSTINI, L. AV1 and VVC Video Codecs: Overview on Complexity Reduction and Hardware Design. **IEEE Open Journal of Circuits and Systems**, v.2, p.564-576, 2021. doi: 10.1109/OJCAS.2021.3107254

IV. NETO, L.; **CORRÊA, M.**; PALOMINO, D.; AGOSTINI, L.; CORRÊA, G. Power-Quality Configurable Hardware Design for AV1 Directional Intra-Frame Prediction. **IEEE Design & Test**, v.39, n.2, p.38-45, 2022. doi: 10.1109/MDAT.2022.3146083

B.2. Conference Papers

I. FERREIRA, R.; LEME, M.; **CORRÊA, M.**; AGOSTINI, L.; DINIZ, C.; ZATT, B. Approximate Subtractor Operator for Low-Power Video Coding Hardware Accelerators. In: IEEE International Conference on Electronics, Circuits and Systems (ICECS), 2019, Genoa. doi: 10.1109/ICECS46596.2019.8964783

II. **CORRÊA, M.**; ZATT, B.; PALOMINO, D.; CORRÊA, G.; AGOSTINI, L. A Fast Local Mode Decision for the HEVC Intra Prediction Based on Direction Detection. In: European Signal Processing Conference (EUSIPCO), 2019, A Coruña. doi: 10.23919/EUSIPCO.2019.8903093

III. **CORRÊA, M.**; WASKOW, B.; ZATT, B.; PALOMINO, D.; CORRÊA, G.; AGOSTINI, L. High Throughput Hardware Design for AV1 Paeth and

Smooth Intra Modes. In: 2019 IEEE International Symposium on Circuits and Systems (ISCAS), 2019, Sapporo. 2019. doi: 10.1109/ISCAS.2019.8702258

IV. **CORRÊA, M.**; WASKOW, B.; GOEBEL, J.; PALOMINO, D.; CORRÊA, G.; AGOSTINI, L. A High Throughput Hardware Architecture Targeting the AV1 Paeth Intra Predictor. In: 2019 IEEE 10th Latin American Symposium on Circuits & Systems (LASCAS), 2019, Armenia. doi: 10.1109/LASCAS.2019.8667544.

V. SALDANHA, M.; **CORRÊA, M.**; CORRÊA, G.; PALOMINO, D.; PORTO, M.; ZATT, B.; AGOSTINI, L. An Overview of Dedicated Hardware Designs for State-of-the-Art AV1 and H.266/VVC Video Codecs. In: IEEE International Conference on Electronics, Circuits and Systems (ICECS), 2020, Glasgow. 2020. doi: 10.1109/ICECS49266.2020.9294862

VI. **CORRÊA, M.**; NETO, L.; PALOMINO, D.; CORRÊA, G.; AGOSTINI, L. ASIC Solution for the Directional Intra Prediction of the AV1 Encoder Targeting UHD 4K Videos. In: 2020 IEEE International Symposium on Circuits and Systems (ISCAS), 2020, Sevilla. doi: 10.1109/ISCAS45731.2020.9180526

VII. NETO, L.; **CORRÊA, M.**; PALOMINO, D.; AGOSTINI, L.; CORRÊA, G. Directional Intra Frame Prediction Architecture with Edge Filter and Upsampling for AV1 Video Coding. In: Symposium on Integrated Circuits and Systems Design (SBCCI), 2020, Campinas. doi: 10.1109/SBCCI50935.2020.9189902

VIII. NETO, L.; **CORRÊA, M.**; PALOMINO, D.; AGOSTINI, L.; CORRÊA, G. Exploring Operation Sharing in Directional Intra Frame Prediction of AV1 Video Coding. In: IEEE Latin America Symposium on Circuits and System (LASCAS), 2021, Arequipa. doi: 10.1109/LASCAS51355.2021.9459136.

IX. NETO, L. **CORRÊA, M.**; ZATT, B.; PALOMINO, D.; AGOSTINI, L.; CORRÊA, G. Configurable Power/Quality-Aware Hardware Design for the AV1 Directional Intra Frame Prediction. In*:* 2021 34th SBC/SBMicro/IEEE/ACM Symposium on Integrated Circuits and Systems Design (SBCCI), 2021, Campinas. doi: 10.1109/SBCCI53441.2021.9529997

X. **CORRÊA, M.**; ROMA, N.; PALOMINO, D.; CORRÊA, G.; AGOSTINI, L. Mode-Adaptive Subsampling of SAD/SSE Operations for Intra Prediction Cost Reduction. In: 2022 IEEE International Symposium on Circuits and Systems (ISCAS), 2022, Austin. doi: 10.1109/ISCAS48785.2022.9937507

XI. PALAU, R.; GOEBEL, J.; ZUMMACH, E.; VIANA, R.; **CORRÊA, M.**; CORRÊA, G.; PORTO, M.; AGOSTINI, L. An UHD4K@60fps Dual Self-Guided Filter Targeting the AV1 Decoder. In: Symposium on Integrated Circuits and Systems Design (SBCCI), 2022, Porto Alegre. doi: 10.1109/SBCCI55532.2022.9893236

XII. **CORRÊA, M.**; PALOMINO, D.; CORRÊA, G.; AGOSTINI, L. Direction-Based Fast Mode Decision and Hardware Design for the AV1 Intra Prediction. In: Symposium on Integrated Circuits and Systems Design (SBCCI), 2022, Porto Alegre. doi: 10.1109/SBCCI55532.2022.9893253