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

Tese

Projeto de Hardware de Alto Desempenho para Ferramentas da Predição Inter Quadros do Codificador AV1

Robson André Domanski

Pelotas, 2023

Robson André Domanski

Projeto de Hardware de Alto Desempenho para Ferramentas da Predição Inter Quadros do Codificador AV1

Tese apresentada ao Programa de Pós-Graduação em Computação do Centro de Desenvolvimento Tecnológico da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Doutor em Ciência da Computação.

Orientador: Prof. Dr. Luciano Agostini
Coorientadores: Prof. Dr. Bruno Zatt
Prof. Dr. Marcelo Schiavon Porto

Pelotas, 2023

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação na Publicação

D666p Domanski, Robson Andre

Projeto de hardware de alto desempenho para ferramentas da predição inter quadros do codificador AV1 / Robson Andre Domanski ; Luciano Agostini, orientador ; Bruno Zatt, Marcelo Porto, coorientadores. — Pelotas, 2023.

114 f.

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

1. Codec AV1. 2. Predição inter quadros. 3. Projeto de hardware. I. Agostini, Luciano, orient. II. Zatt, Bruno, coorient. III. Porto, Marcelo, coorient. IV. Título.

CDD : 005

Elaborada por Simone Godinho Maisonave CRB: 10/1733

Robson André Domanski

Projeto de Hardware de Alto Desempenho para Ferramentas da Predição Inter Quadros do Codificador AV1

Tese aprovada, como requisito parcial, para obtenção do grau de Doutor em Ciência da Computação, Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas.

Data da Defesa: 25 de abril de 2023

Banca Examinadora:

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

Prof. Dr. Daniel Palomino
Doutor em Computação pela Universidade Federal do Rio Grande do Sul.

Profª. Dra. Roberta Carvalho Nobre Palau
Doutora em Ciência da Computação pela Universidade Federal de Pelotas.

Prof. Dr. Felipe Martin Sampaio
Doutor em Computação pela Universidade Federal do Rio Grande do Sul.

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

Este trabalho é dedicado às crianças adultas que, quando pequenas, sonharam em se tornar cientistas.

AGRADECIMENTOS

É com grande honra que expresso minha mais profunda gratidão a todos aqueles que contribuíram de forma significativa para a realização deste trabalho de pesquisa. Minha construção de conhecimento e meu trabalho durante o doutorado ocorreram de forma coletiva e colaborativa dentro do grupo de pesquisa VITECH (*Video Technology Research Group*), e desde o início me senti parte dele e fui bem recebido por todos os membros. Este trabalho representa uma contribuição importante para a área de estudo, como também uma grande importância no meu crescimento pessoal, e nada disso teria sido possível sem a colaboração de muitas pessoas.

Quero agradecer a Deus por me fazer chegar até aqui nas adversidades durante esse período. Por entrar numa área totalmente nova, pela persistência, pelo aprendizado, e por poder terminar a fase com novas experiências e conhecimentos adquiridos.

Eu gostaria de expressar minha sincera gratidão à minha família pelo apoio incondicional que me foi dado ao longo deste caminho. Sem o amor, o apoio e a paciência deles, eu não teria sido capaz de completar esta jornada. Minha família sempre me encorajou a seguir meus sonhos e a buscar o conhecimento, independentemente das dificuldades que enfrentei ao longo do caminho. A minha mãe Nair e ao meu pai Mariano (*in memoriam*), quero agradecer por me criarem com valores fortes e me motivar a buscar a excelência em tudo o que eu faço. Seus sacrifícios e dedicação incansável me inspiraram a trabalhar duro e nunca desistir, e eu sou eternamente grato por tudo o que eles fizeram por mim. Ao meu irmão Christian, quero agradecer por sua presença constante em minha vida e pelo incentivo e apoio emocional que me deu durante todo o processo de pesquisa. Por fim, quero agradecer a minha namorada Priscila, que me apoiou em todos os momentos difíceis, me dando amor e compreensão durante os anos de estudo. A todos vocês, minha família, agradeço por estarem comigo nesta jornada. Espero que este trabalho seja uma forma de retribuir o amor, a dedicação e o apoio que sempre me ofereceram.

Tenho também minha sincera gratidão ao meu orientador, Luciano Agostini, por sua orientação, paciência e encorajamento durante todo o processo de pesquisa. Sua experiência, habilidade de ensino e dedicação inabalável tornaram possível a conclusão desta tese de doutorado. Gostaria de agradecer por sua paciência e disposição em ajudar-me, principalmente ao início dessa jornada, o qual teve paciência de sentar ao meu lado e explicar e reexplicar muitas vezes algo que era totalmente novo para mim, e especialmente durante os períodos em que enfrentei dificuldades ou momentos de incerteza. E não menos importante, gostaria de agradecer a ajuda dos meus coorientadores Marcelo Porto e Bruno Zatt.

Gostaria de agradecer a todos os meus colegas de pesquisa e amigos, que estiveram ao meu lado durante todo o processo de elaboração desta tese de doutorado.

Seu apoio, conhecimento e experiência me ajudaram a crescer em minha capacidade de pesquisa e me incentivaram a persistir quando as coisas se tornaram difíceis. Primeiramente ao meu colega William, que trabalhou como bolsista desta pesquisa, que ajudou em boa parte desta jornada e o qual tive o prazer de ser coorientador do seu Trabalho de Conclusão de Curso. Aos colegas Wagner, Jones e Rafael pela troca de ideias e experiência durante toda fase do doutorado. Roberta, Ruhan, Vladimir, Alex, Murilo e Narusci gostaria de agradecer a parceira e coleguismo durante todo esse período. Por fim, gostaria de agradecer a todos os meus colegas de pesquisa, que eu tive o privilégio de conhecer e trabalhar durante o curso do meu doutorado. Sua presença e amizade tornaram esta jornada muito mais agradável e enriquecedora, e serei eternamente grato por suas contribuições neste trabalho.

Por fim, gostaria de agradecer a todos que contribuíram de alguma forma para o desenvolvimento deste trabalho.

Lembre-se sempre, o seu foco determina a sua realidade.
— QUI-GON JINN

RESUMO

DOMANSKI, Robson André. **Projeto de Hardware de Alto Desempenho para Ferramentas da Predição Inter Quadros do Codificador AV1**. Orientador: Luciano Agostini. 2023. 114 f. Tese (Doutorado em Ciência da Computação) – Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2023.

Atualmente os vídeos digitais estão cada vez mais presentes no mundo inteiro, sejam eles destinados ao lazer ou as atividades profissionais. Por consequência, pode-se observar um expressivo crescimento no número de dispositivos móveis capazes de capturar, manipular e transmitir estes vídeos digitais. Esses dispositivos móveis, por sua vez, possuem sérias restrições em termos de consumo de energia e de poder computacional. Neste contexto, a codificação de vídeos é responsável por parte importante destas restrições. Os novos codificadores são desenvolvidos com o objetivo de realizar a compressão desses vídeos de forma cada vez mais eficiente e para resoluções cada vez mais elevadas, assim tornando o suporte a vídeos digitais uma operação bastante custosa do ponto de vista de poder computacional e de consumo de energia. Nesse cenário, o codificador *AOMedia Video 1* (AV1) tem ganhado destaque. O AV1 é um codificador desenvolvido por um consórcio de grandes empresas da área, como *Google, Apple, Intel, AMD* entre outras, com o objetivo de ter elevada eficiência de compressão em uma solução livre de royalties. Para tanto, o AV1 adotou novas ferramentas de codificação de elevado custo computacional o que amplia os desafios para que dispositivos móveis suportem esse codificador. Entre as etapas do AV1 que trouxeram maiores inovações está a predição inter quadros. Por conta destas inovações, a predição inter quadros é responsável pelo maior custo computacional no codificador AV1, com cerca de 58% do tempo de execução, principalmente quando levamos em conta que a inter quadros é uma etapa dentre as oito dentro do codificador AV1. Este trabalho apresenta um conjunto de soluções em hardware dedicado para a predição inter quadros do AV1. Foram desenvolvidas arquiteturas para Estimação de Movimento Fracionária, a Compensação de Movimento Convencional e as Compensações de Movimentos Distorcidos Local e Global. Todas as arquiteturas propostas buscaram um projeto de hardware eficiente em termos de área e potência. As arquiteturas foram sintetizadas para ASIC usando a biblioteca TSMC de 40 nm e foram capazes de processar vídeos de ultra alta definição, como UHD 4K e UHD 8K a taxas de quadros variando entre 30 e 60 quadros por segundo. As arquiteturas desenvolvidas neste trabalho são as primeiras na literatura que implementaram soluções para as ferramentas da predição inter quadros do codificador AV1.

Palavras-chave: Codec AV1. Predição Inter Quadros. Projeto de Hardware.

ABSTRACT

DOMANSKI, Robson André. **High-Performance Hardware Design for AV1 Encoder Inter-Frame Prediction Tools**. Advisor: Luciano Agostini. 2023. 114 f. Thesis (Doctorate in Computer Science) – Technology Development Center, Federal University of Pelotas, Pelotas, 2023.

Currently, digital videos are increasingly present all over the world, whether they are intended for leisure or professional activities. Consequently, a significant growth in the number of mobile devices capable of capturing, manipulating and transmitting these digital videos can be observed. These mobile devices, in turn, have serious restrictions in terms of energy consumption and computational power. In this context, video encoding is responsible for an important part of these restrictions. New encoders are developed with the aim of compressing these videos more and more efficiently and for ever higher resolutions, thus making the support to digital videos a very costly operation from the point of view of computational power and energy consumption. In this scenario, the AOMedia Video 1 (AV1) encoder has gained prominence. AV1 is an encoder developed by a consortium of large companies in the area, such as Google, Apple, Intel, AMD, among others, with the objective of having high compression efficiency in a royalty-free solution. To this end, AV1 adopted new coding tools with a high computational cost, which increases the challenges for mobile devices to support this encoder. Among the stages of AV1 that brought greater innovations is the inter-frame prediction. Due to these innovations, the interframe prediction is responsible for the highest computational cost in the AV1 encoder, with about 58% of its the execution time, especially when we take into account that the interframe is one step among the eight within the AV1 encoder. This work presents a set of dedicated hardware solutions for AV1 inter-frame prediction. Efficient architectures were developed for Fractional Motion Estimation, Conventional Motion Compensation and Local and Global Warped Motion Compensations. All proposed architectures sought an efficient hardware design in terms of area and power. The architectures were synthesized to ASIC using the 40 nm TSMC library and they are able to process ultra high definition videos such as UHD 4K and UHD 8K at frame rates ranging between 30 and 60 frames per second. The architectures designed in this work are the first in the literature presenting solutions for the inter-frame prediction tools of the AV1 encoder.

Keywords: AV1 Codec. Inter-Frames Prediction. Hardware Design.

LISTA DE FIGURAS

1 Linha do tempo de desenvolvimento dos principais codificadores. Fonte: Adaptado de (GOEBEL, 2019). 20
2 Percentuais de tempo de execução dos vídeos para cada estágio de codificação para cada CQ Level do codificador AV1 (BORGES, 2020). 23
3 Diagrama de blocos genérico de codificação de vídeo. Fonte: Adaptado de (AGOSTINI; SILVA; BAMPI, 2007). 29
4 Particionamento do codificador AV1. Fonte: Adaptado de (Chen et al., 2018). 30
5 Elementos presentes na Estimativa de Movimento (AFONSO; AGOSTINI; FRANCO, 2013). 37
6 Exemplo de uso de múltiplos quadros de referência. 38
7 Posição sub-pixel FME. 42
8 Exemplos de transformações afins. Adaptado de (KOLODZIEJSKI; DOMANSKI; AGOSTINI, 2022). 45
9 Bloco atual em amarelo. Os blocos vizinhos que se referem à mesma imagem de referência que o bloco atual estão em verde. Adaptado de (Parker et al., 2017a). 46
10 Cisalhamento horizontal, vertical e combinado, respectivamente. Adaptado de (OPEN MEDIA, 2022). 48
11 Fluxograma da Compensação de Movimento Distorcido Global. . . 52
12 Arquitetura de interpolação multi-filtro (MFIA) para a Compensação de Movimento do AV1. 62
13 Arquitetura do interpolador usado na MC do AV1 denominado 1RC. 64
14 Arquitetura do interpolador usado na MC do AV1 denominado 2RC. 65
15 Arquitetura do interpolador usado na MC do AV1 denominado 4RC. 66
16 Percentuais de vezes que cada filtro é usado na FME do AV1. . . . 70
17 Arquitetura multi-filtro original (OM) usada na FME do AV1. 71
18 Exemplo de amostras usadas na interpolação. 74
19 Amostras necessárias para interpolar um bloco 4 × 4 usando filtros de 8, 6 e 4 taps. 75
20 Comparação de qualidade visual extraída de uma sequência manipulada nos testes. 77
21 Arquiteturas de interpolação multi-filtro da FME do AV1 usando a solução AM8 (MF-AM8), AM6 (MF-AM6) e AM4 (MF-AM4). 79

22 Arquiteturas dos interpoladores da FME do AV1 usando uma instâncias de MF-OM ou MF-AM8 (1-AFI-OM ou 1-AFI-AM8), MF-AM6 (1-AFI-AM6) e MF-AM4 (1-AFI-AM4). 81
23 Arquiteturas dos interpoladores da FME do AV1 usando quatro instâncias de MF-OM ou MF-AM8 (4-AFI-OM ou 4-AFI-AM8), MF-AM6 (4-AFI-AM6) e MF-AM4 (4-AFI-AM4). 82
24 Diagrama da arquitetura completa da FME usada no codificador AV1. 85
25 Arquitetura do interpolador usado no bloco UI com quatro instâncias dos multi-filtros projetados denominada 4-AF-X. 87
26 Arquitetura do interpolador usado no bloco UI com oito instâncias dos multi-filtros projetados denominada 8-AF-X. 88
27 Arquitetura da etapa de Unidade de Melhor Casamento da FME do AV1: (a) Arquitetura Unidade de Árvore de SAD ; (b) Arquitetura Acumulador de SAD (c) Arquitetura Comparador de SAD. 89
28 Diagrama temporal da FME do AV1: (a) Processamento da arquitetura 4-AF-X; (b) Processamento da arquitetura 8-AF-X. 90
29 Arquitetura multi-filtro para a Compensação de Movimento Distorcido Local (AMDL). 93
30 Arquitetura do Interpolador para Compensação do Movimento Distorcido Local com Uma Linha por Ciclo (1L-AIDL). 95
31 Arquitetura do Interpolador para Compensação de Movimento Distorcido Local com Quatro Linhas por Ciclo (4L-AIDL) 96
32 Fluxograma da arquitetura da compensação de movimento distorcido global. 98
33 Círculo de Bresenham utilizado no algoritmo FAST. Adaptado de (ROSTEN; DRUMMOND, 2006). 99
34 Arquitetura da etapa de Detecção usada na GWMC. 100
35 Arquitetura da etapa de Descrição e Casamento usada na GWMC. 101
36 Arquitetura de Hardware da etapa de Estimação usada na GWMC. 102

LISTA DE TABELAS

1 Coeficientes dos filtros: Regular 6 taps, Smooth 6 taps e Sharp 8 taps 40
2 Coeficientes dos filtros: Regular 4 taps, Smoth 4 taps e Bilinear 2 taps 41
3 Filtros FIR da Compensação de Movimento Distorcido Local 49
4 Sumário comparativo da predição inter quadros dos codificadores HEVC e AV1 58
5 Resultados de sínteses para as arquiteturas usadas na MC do AV1. 67
6 Coeficientes originais e modificados dos filtros do AV1. 73
7 Amostras Necessárias por Tamanho de Bloco 75
8 Resultado Eficiência de Codificação 76
9 Uso de hardware nas três arquiteturas multi-filtro usadas na FME do AV1 79
10 Resultados de sínteses para as arquiteturas multi-filtro da FME do AV1 80
11 Resultados de sínteses para as arquiteturas do interpolador usado na FME do AV1 84
12 Resultados de sínteses para as arquiteturas completa da FME do AV1 90
13 Resultados de síntese das arquiteturas projetadas para a LWMC do AV1. 97
14 Resultados de síntese das arquiteturas projetadas para a GWMC do AV1. 103

LISTA DE ABREVIATURAS E SIGLAS

1L-AIDL	Arquitetura do Interpolador para Compensação do Movimento Distorcido Local com Uma Linha por Ciclo
4L-AIDL	Arquitetura do Interpolador para Compensação de Movimento Distorcido Local com Quatro Linhas por Ciclo
A-SAD	Acumulador de SAD
AVC	Advanced Video Coding
AOMedia	Alliance for Open Media
AV1	AOMedia Video 1
AM4	Approximate Multiplierless with 4-taps
AM6	Approximate Multiplierless with 6-taps
AM8	Approximate Multiplierless with 8-taps
AMDL	Arquitetura Multifiltro para a Compensação de Movimento Distorcido Local
MFIA	Arquitetura de Interpolação Multifiltro
ADST	Asymmetric Discrete Sine Transform
AFI-AM4	AV1 FME Interpolation using AM4
AFI-AM6	AV1 FME Interpolation using AM6
AFI-AM8	AV1 FME Interpolation using AM8
AFI-OM	AV1 FME Interpolation using OM
BD-BR	Bjøntegaard Delta Bit Rate
BD-PSNR	Bjøntegaard Delta PSNR
HS	Cisalhamento Horizontal
VS	Cisalhamento Vertical
VCTQM	Codec de Vídeo e Medição de Qualidade
C-SAD	Comparador de SAD
CDEF	Constrained Directional Enhancement Filter
CQ	Constrained Quality
CDF	Cumulative Distribution Function

DBF	Deblocking Filter
dB	Decibéis
DCT	Discrete Cosine Transform
FAST	Features from Accelerated Segment Test
FIR	Finite Impulse Response
FIR	Finite Impulse Response
FlipADST	Flipped Asymmetric Discrete Sine Transform
FME	Fracional Motion Estimation
FPS	Frames per Second
GPP	General Purpose Processor
GWMC	Global Warped Motion Compensation
GPU	Graphic Processing Unit
HEVC	High Efficiency Vido Coding
HVS	Human Visual System
IDTX	Identity Transform
ISCAS	IEEE International Symposium on Circuits and Systems
LASCAS	IEEE Latin American Symposium on Circuits and Systems
IME	Integer Motion Estimation
ISO	International Organization for Standarlization
ITU-T	International Telecommunications Union
IDL	Interpolador para a Compensação de Movimento Distorcido Local
Intra - BC	Intra Block Copy
JVET	Joint Video Experts Team
LWMC	Local Warped Motion Compensation
LFNSST	Low Frequency Non-Separable Secondary Transform
MC	Motion Compensation
ME	Motion Estimation
MV	Motion Vector
MF-AM4	Multifilter AM4
MF-AM6	Multifilter AM6
MF-AM8	Multifilter AM8
MF-OM	Multifilter OM
MCM	Multiple Constant Multiplication
OM	Original Multifilter

OBMC	Overlapped Block Motion Compensation
PSNR	Peak Signal-to-Noise Ratio
RANSAC	Random Sample Consensus
SRC	Shift Register Chains
SRCR	Shift-Register Chain Router
SRC	Shift-Register-Chain
SAD	Sum of Absolute Differences
SB	SuperBlocks
SLRF	Switchable Loop Restoration Filter
TCAS-II	Transactions on Circuits and Systems II
UA-SAD	Unidade de Árvore de SAD
UCM	Unidade de Cálculo do Modelo
UHD	Ultra High Definition
UI	Unidade de Interpolação
UMC	Unidade de Melhor Casamento
USMM	Unidade de Seleção do Melhor Modelo
UGPA	Unidade Geradora de Pontos Aleatórios
VVC	Versatile Video Coding
VCEG	Video Coding Experts Group
VITECH	Video Technology Research Group
WMC	Warped Motion Compensation

SUMÁRIO

1 INTRODUÇÃO 19
1.1 Questão e Hipótese de Pesquisa 24
1.2 Objetivos e Contribuições 25
1.3 Apresentação do Texto 26

2 CONCEITOS DE COMPRESSÃO DE VÍDEOS E O FORMATO AV1 27
2.1 Codificador de Video AV1 29
2.1.1 Partição de Blocos 30
2.1.2 Predição Intra Quadro 31
2.1.3 Predição Inter Quadros 32
2.1.4 Transformadas e Quantização 33
2.1.5 Codificação de Entropia 33
2.1.6 Filtragem de Laço 34
2.2 Métricas de Avaliação 34

3 FERRAMENTAS DA PREDIÇÃO INTER QUADROS DO CODIFICADOR AV1 36
3.1 Estimação de Movimento 36
3.1.1 Estimação de Movimento Inteira 37
3.1.2 Estimação de Movimento Fracionária 38
3.2 Compensação de Movimento Convencional 41
3.3 Compensação de Movimento Distorcido 42
3.3.1 Transformações Afins 43
3.3.2 Compensação de Movimento Distorcido Local 44
3.3.3 Compensação de Movimento Distorcido Global 48

4 TRABALHOS RELACIONADOS 55
4.1 Trabalhos com Foco na Predição Inter Quadros do AV1 55
4.2 Trabalhos com Foco na Predição Inter Quadros do HEVC 56
4.3 Trabalhos com Computação Aproximada com Foco na Predição Inter Quadros 57
4.4 Conclusão Capítulo 57

5 ARQUITETURA PARA O INTERPOLADOR DE SUBAMOSTRAS DA COMPENSAÇÃO DE MOVIMENTO 60
5.1 Metodologia 60
5.2 Filtro Usado na Interpolação da MC do AV1 61
5.3 Interpolador Usado na MC do AV1 63
5.4 Análise e Resultados 67

6 ARQUITETURAS DESENVOLVIDAS PARA A ESTIMAÇÃO DE MOVIMENTO FRACIONÁRIA DO AV1 69
6.1 Arquitetura dos Filtros 69
6.2 Arquiteturas do Interpolador 80
6.3 Arquitetura da FME Completa 84
6.3.1 Unidade de Interpolação 86
6.3.2 Unidade de Melhor Casamento 86
6.3.3 Análise e Resultados 88

7 ARQUITETURAS PARA A COMPENSAÇÃO DE MOVIMENTO DISTORCIDO 92
7.1 Arquitetura para a Compensação de Movimento Distorcido Local . . 92
7.1.1 Arquitetura Multifiltro para a Compensação de Movimento Distorcido Local 93
7.1.2 Interpolador para a Compensação de Movimento Distorcido Local . . . 94
7.1.3 Análise e Resultados 97
7.2 Arquitetura para a Compensação de Movimento Distorcido Global . . 97
7.2.1 Etapas do Processamento da Compensação de Movimento Distorcido Global 98
7.2.2 Análise e Resultados 103

8 CONCLUSÕES 104

REFERÊNCIAS 106

APÊNDICE A LISTA DE PUBLICAÇÕES REALIZADAS DURANTE ESTA TESE 114
A.1 Artigos Publicados em Revistas 114
A.2 Artigos Publicados em Conferências 114

1 INTRODUÇÃO

Os vídeos digitais estão cada vez mais presentes no mundo inteiro, sejam eles destinados ao lazer ou às atividades profissionais. Devido a esse crescimento, o número de dispositivos móveis capazes de capturar, manipular, armazenar e transmitir esses vídeos digitais cresceu expressivamente. Em pesquisa recente, o *YouTube* indicou que mais de 70% das visualizações de seus vídeos acontecem em dispositivos móveis (YOUTUBE, 2021). Outro ponto relevante para esse debate é que 66,2% da distribuição do volume global mensal de dados móveis diz respeito ao compartilhamento de vídeo (STATISTA, 2022). Essa demanda é impulsionada principalmente pela rápida popularização de mídias sociais, como *Facebook, Instagram e TikTok*, bem como os serviços de *streaming* de vídeo, como *Youtube, Netflix, Amazon Prime* e muitos outros que mudaram severamente nossos padrões atuais de entretenimento.

Durante a pandemia do COVID-19 o consumo de vídeos cresceu de forma ainda mais expressiva, pois vários países decretaram *lockdowns*. Já no início de 2020, este episódio proporcionou um aumento de 40,8% no uso das redes sociais, e de 29,6% no consumo de serviços de *streaming* (FASTLY, 2020). Havia uma previsão que o tráfego global de internet crescesse 40% em 12 meses, mas com o COVID-19 isso aconteceu em apenas uma semana (INTERDIGITAL, 2020). Considerando este crescente tráfego de vídeos e o advento de resoluções cada vez maiores, como o *Ultra-high Definition* (UHD) 4K (3840x2160 píxeis) e 8K (7680x4320 píxeis), com crescentes taxas de quadros por segundo, passou a ser um desafio viabilizar a transmissão e o armazenamento destes conteúdos.

Atualmente existem padrões de compressão de vídeo muito eficientes, como o *High-Efficiency Vido Coding* ou HEVC (ISO/IEC-JCT1/SC29/WG11, 2013) e o *Versatile Video Coding* ou VVC (FRAUNHOFER, 2021), definidos pela *International Telecommunications Union* (ITU-T) e pela *International Organization for Standarlization* (ISO), mas seus usos são dificultados pela quantidade enorme de ferramentas presentes nesses padrões que são patenteadas por diversas empresas. Assim, além do desafio técnico de gerar soluções com elevada eficiência de codificação, surge outros desafios: os custos do uso das soluções existentes no mercado e a dificuldade jurídica

associada ao licenciamento de tantas tecnologias patenteadas (Layek et al., 2017).

Em função destas dificuldades com as tecnologias patenteadas por diversas empresas, várias empresas gigantes como *Google, Intel, Cisco, Mozilla, Microsoft, Netflix, Amazon* e muitas outras formaram a *Alliance for Open Media* (AOMedia), para o desenvolvimento de um novo codec de vídeo livre de royalties chamado AOMedia Video 1 (AV1) (AOMEDIA, 2022).

A Figura 1 apresenta a linha do tempo dos codificadores da ITU-T (linha H.26X), onde vários destes padrões são compartilhados com a ISO, como H.262 ou MPEG-2 *Part 2 – Video*, o H.264 ou MPEG-4 *Part 10 Advanced Video Coding* (AVC), o H.265 ou MPEG-H *Part 2 High Efficiency Vido Coding* e o H.266 ou MPEG-I *Part 3 - Versatile Video Coding* (VVC). A Figura 1 também apresenta a linha de codificadores que conduziram ao surgimento do AV1, todos livres de royalties.

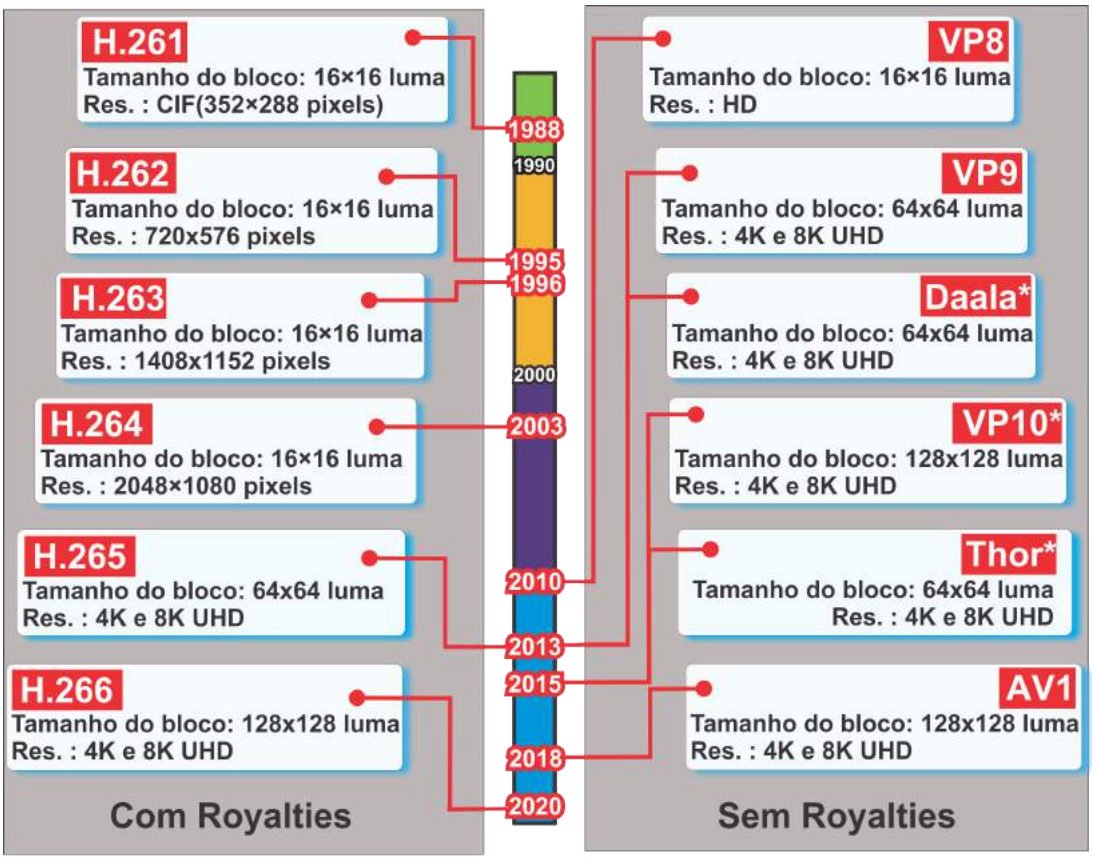

Figura 1 – Linha do tempo de desenvolvimento dos principais codificadores. Fonte: Adaptado de (GOEBEL, 2019).

O H.261 (HUITEMA; TURLETTI, 1996) foi o primeiro codificador padronizado pelo *Video Coding Experts Group* (VCEG) da ITU-T, lançado em 1988, tinha o objetivo de atender a demanda de processamento da sua época. O H.261 processava de forma eficiente as resoluções de até no máximo 352x288 píxeis, e amostras de luminância

com blocos de 16x16. Após o H.261, surgiu o H.262 (ou MPEG-2), primeiro padrão de codificação de grande sucesso comercial. Depois do H.262, surgiram o H.263 e o H.264, com o intuito de atender a demanda de novas resoluções de suas épocas. Em 2013 foi lançado o H.265 ou HEVC e, em 2020, foi lançado o H.266 ou VVC, atual estado da arte em codificação de vídeos da ITU-T e da ISO. Estes codificadores estão representados na Figura 1, em ordem cronológica.

O novo padrão de codificação de vídeo da ITU-T e ISO/IEC, o VVC, promete reduzir o tamanho dos arquivos de vídeo em 50% em comparação ao padrão anterior, o H.265, para uma mesma qualidade de vídeo objetiva. O H.266/VVC segue o padrão de compressão do H.265, e foi desenvolvido pela *Joint Video Experts Team* (JVET) com contribuições significativas do departamento de comunicação e aplicativos de vídeo do *Fraunhofer HHI*, em parcerias com empresas como *Sony, Apple, Intel, Huawei, Microsoft, Qualcomm e Ericsson*. O VVC é capaz de processar blocos de 128 x 128 com um particionamento recursivo em árvore quadrática (QT), e para cada folha da QT é permitido um novo particionamento de árvore multi-tipo recursivo (MTT), incluindo partições binárias e ternárias. O VVC é capaz de processar de forma eficiente vídeos de alta resolução, como UHD 4K e 8K, além de ser eficiente na codificação de mídias emergentes, como vídeos omnidirecionais imersivos de 360 graus e vídeos de alta faixa dinâmica (HDR) (FRAUNHOFER, 2021).

Por sua vez, o desenvolvimento do AV1 é uma confluência de três esforços distintos que estavam sendo realizados por empresas da área. De um lado, a Google, com sua linha de codificadores VPx, estava preparando a migração do seu conteúdo do VP9 (amplamente usado nos serviços da *Google* como o *YouTube*) para uma nova versão, chamada de VP10 (Topiwala; Dai; Krishnan, 2016). Por outro lado, foram aproveitados os esforços da *Mozilla* no desenvolvimento do codificador *Daala* (XIPH.ORG, 2018) e da Cisco no desenvolvimento do codificador Thor (Bjøntegaard et al., 2016). A primeira especificação de fluxo de bits e decodificação AV1 foi lançada em abril de 2018, que estabeleceu o software de referência *AOMedia Codec* versão 1.0.0, chamado de *libaom* (RIVAZ; HAUGHTON, 2019). Este codificador de vídeo de código aberto foi concebido para ser totalmente isento de royalties e escalável para qualquer dispositivo moderno em qualquer largura de banda. O AV1 é um codificador flexível para uso em conteúdo comercial e não comercial, ideal para projetos de transmissão de vídeo para resoluções UHD, alcançando uma eficiência de compressão até 30% maior que os codificadores livre de royalties que o precederam (AOMEDIA, 2020), (ZABROVSKIY; FELDMANN; TIMMERER, 2018).

Tendo em vista o amplo crescimento do uso de dispositivos embarcados capazes de manipular vídeos, tais como, *smartphones, videogames, tablets*, entre outros, questões como consumo de energia e quantidade de recursos de hardware utilizados são de grande importância. Com base nesses elementos, é possível compreender

que o desenvolvimento de hardware dedicado se mostra uma ótima alternativa, pois um software sendo executado em um Processador de Propósito Geral (GPP - *General Purpose Processor*) ou mesmo em uma Unidade de Processamento Gráfico (GPU - *Graphic Processing Unit*), possui um custo energético proibitivo, além de ser incapaz de atingir tempo real ao processar vídeos de elevada resolução. Desde modo, o projeto de arquiteturas de hardware dedicadas de elevada taxa de processamento torna-se essencial para possibilitar uma adequada relação entre eficiência de codificação e consumo energético, quando o foco são dispositivos que são dependentes de bateria para seu funcionamento.

BORGES (2020) apresenta uma análise da distribuição do custo computacional do AV1. O autor descreve que, neste caso, o custo computacional foi avaliado em termos do tempo de processamento exigido em cada módulo, o que equivale ao número aproximado de operações necessárias. A Figura 2 apresenta dois gráficos com um resumo dos resultados percentuais encontrados em (BORGES, 2020), para o tempo de codificação das diversas ferramentas do AV1 (que serão explicadas no próximo capítulo). Assim, a Figura 2 (a) apresenta os resultados para vídeos HD 1080 (1920x1080 píxeis a 60 quadros por segundo), enquanto a Figura 2 (b) apresenta os resultados para vídeos UHD 4K (3840x2160 píxeis a 60 quadros por segundo). Segundo a Figura 2, é possível observar que a predição inter quadros é responsável pelo maior custo computacional nos dois cenários, com 63.21% do tempo de execução para vídeos HD 1080 e 52.66% para vídeos UHD 4K. Desse modo, é possível perceber que a predição inter quadros, como em outros codificadores, é o módulo de maior custo computacional do codificador AV1. Por conta desse elevado custo computacional, a predição inter quadros do AV1 foi escolhida como foco desta tese de doutorado.

O projeto de hardware dedicado é essencial para lidar com o elevado custo computacional da predição inter quadros do AV1, especialmente quando vídeos de alta resolução estão sendo processados por dispositivos móveis. O projeto de hardware nesse cenário deve focar em diversos requisitos, por vezes contraditórios, entre esses, destacam-se:

- a taxa de compressão;
- a qualidade do vídeo;
- a quantidade de recursos de hardware usados;
- a largura da banda de memória;
- a energia consumida;
- a potência dissipada, entre outras (AGOSTINI; SILVA; BAMPI, 2007).

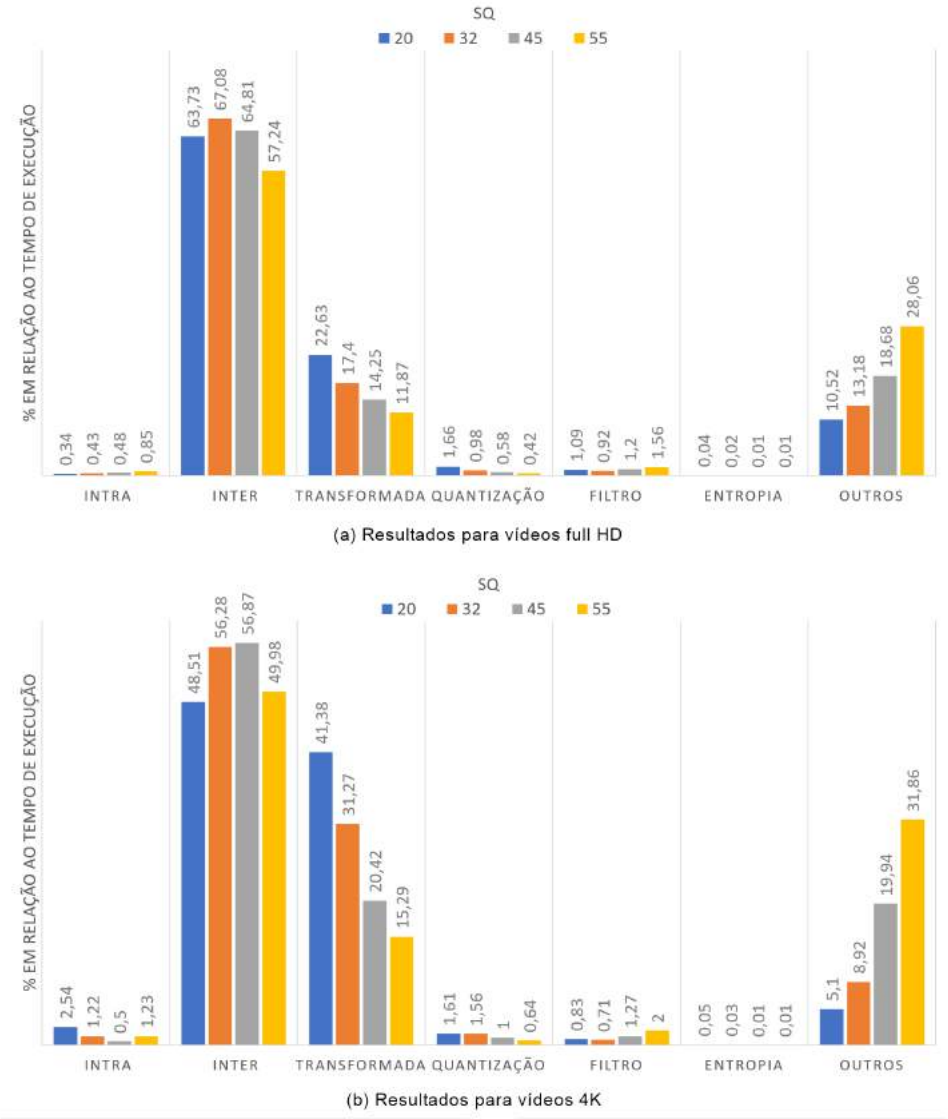

Figura 2 – Percentuais de tempo de execução dos vídeos para cada estágio de codificação para cada CQ Level do codificador AV1 (BORGES, 2020).

A relação entre taxa de compressão e qualidade do vídeo é chamada de eficiência de codificação e o ideal é que as simplificações dos algoritmos usados no projeto de hardware afetem o mínimo possível a eficiência de codificação. Por outro lado, as soluções em hardware devem ser eficientes no uso dos recursos e, principalmente, devem focar em baixo consumo de área e baixa dissipação de potência. Naturalmente, diversos desses requisitos são contraditórios e é importante buscar um balanço adequado entre ganhos e perdas nesses requisitos durante o projeto de hardware dedicado para a codificação de vídeos.

1.1 Questão e Hipótese de Pesquisa

Considerando o que foi exposto até aqui, evidencia-se que existe um vasto espaço de pesquisa e desenvolvimento sobre projeto de hardware dedicado para o novo codificador de vídeo da AOMedia, o codificador AV1. Esta presente pesquisa de doutorado começou em 2018, ano em que o formato AV1 foi introduzido, então não existiam trabalhos sobre o AV1 na literatura da época. Dada a relevância do AV1 e as muitas inovações que existem dentro dele, esta tese explora a viabilidade do projeto de arquiteturas de hardware dedicado com a busca de uma elevada eficiência em termos de área e potência para a predição inter quadros do AV1, com foco especial em vídeos de altas resoluções, como UHD 4K e UHD 8K. O foco nas ferramentas usadas na predição inter quadros se dá pelo seu alto custo computacional, conforme apresentado na Figura 2, e na experiência de trabalhos anteriores desenvolvidos no filtro de laço do padrão HEVC. A partir dessa definição inicial, foram elaboradas as questões de pesquisa que norteiam este trabalho:

Como gerar soluções eficientes em termos de área e potência para as ferramentas da predição inter quadros do formato AV1, capazes de processar vídeos de ultra altas resoluções?

Com base nessa questão, a hipótese é de que o projeto de hardware dedicado para as ferramentas da predição inter quadros do AV1 explorando paralelismo e o uso de computação aproximada é uma opção viável para atingir as taxas de processamento necessárias para processar vídeos de ultra alta resolução em tempo real e com eficiência em termos de área e potência.

Desde modo, todas as soluções desenvolvidas nesta tese investigam esta hipótese, sendo que cada arquitetura proposta apresenta diferentes desafios, os quais ficarão claros no decorrer deste texto. Os focos de investigação ficaram concentrados na Compensação de Movimento Convencional, na Estimação de Movimento Fracionária e na Compensações de Movimentos Distorcidos Local e Global, uma vez que não foi possível, por conta do tempo disponível para o desenvolvimento desta tese, também investigar as outras ferramentas da predição inter quadros do AV1. Estas ferramentas de codificação, bem como as soluções em hardware desenvolvidas serão detalhadas no decorrer deste texto.

Ao final do trabalho foi possível demonstrar que a hipótese apresentada acima é totalmente válida para as ferramentas investigadas neste trabalho.

Conforme já mencionado anteriormente, para além das publicações geradas a partir desta tese, ainda não existem outros trabalhos na literatura que possam ser comparados com as soluções desenvolvidas e apresentadas neste texto. Isso demonstra a novidade das soluções propostas e apresentadas nesse trabalho.

1.2 Objetivos e Contribuições

O objetivo geral deste trabalho é explorar a viabilidade de construção de hardware dedicado e com eficiência em termos de área e potência para a predição inter quadros do AV1, quando processando vídeos com altas resoluções (UHD 4K e 8K), explorando técnicas de computação aproximada em conjunto com técnicas de projeto de hardware de baixo consumo.

O foco desta tese está no desenvolvimento de novas arquiteturas de hardware para algumas das ferramentas da predição inter quadros do AV1, conforme descrito abaixo, onde também são resumidos os principais métodos utilizados, bem como as publicações geradas a partir de cada solução:

1. Desenvolvimento da Compensação de Movimento (*Motion Compensation* - MC) do codificador AV1 usando apenas somadores e deslocadores, ao invés de multiplicadores (método original) e usando diferentes níveis de paralelismo. Essa solução foi publicada em (Domanski et al., 2019) e está apresentada em detalhes no Capítulo 5 deste texto.

2. Desenvolvimento do interpolador da Estimação de Movimento Fracionária (*Fracional Motion Estimation* - FME) do AV1 explorando computação aproximada em dois níveis e também implementando os filtros originais (para comparação). Esse trabalho foi publicado em (DOMANSKI et al., 2021) e selecionado entre os quatro melhores da *track* de *Multimedia Systems and Applications*. Esta solução está apresentada em detalhes no Capítulo 6 deste texto.

3. Desenvolvimento de hardware visando o processo completo da FME do AV1, com base nos filtros originais da FME. Esse trabalho foi publicado em (DOMANSKI et al., 2023a) e está apresentado em detalhes no Capítulo 6 do texto.

4. Desenvolvimento de hardware para o Modo de Compensação de Movimento Distorcido Local (*Warped Local*), que está detalhada no Capítulo 7 e ainda não publicados. Essa solução, usa um interpolador com 192 filtros. A arquitetura foi desenvolvida usando apenas somadores e deslocadores, ao invés de multiplicadores (método original), e em diferentes níveis de paralelismo.

5. Desenvolvimento de hardware para o Modo de Compensação de Movimento Distorcido Global (*Warped Global*), que também está detalhada no Capítulo 7 deste texto. A *Warped Global* do AV1 também foi desenvolvida substituindo os multiplicadores por somadores e deslocadores. Esta solução ainda não foi enviada para publicação.

1.3 Apresentação do Texto

O restante deste texto está organizado da seguinte forma: O capítulo 2, apresenta uma introdução sobre codificação de vídeo e o codificador AV1. O capítulo 3, apresenta uma revisão sobre os conceitos empregados nesse trabalho, mais especificamente sobre a predição inter quadros do codificador AV1. O capítulo 4, apresenta os trabalhos encontrados na literatura sobre o tema da tese. O capítulo 5, apresenta o desenvolvimento da arquitetura e os resultados obtidos para a Compensação de Movimento do AV1. O capítulo 6, apresenta o desenvolvimento da arquitetura e os resultados obtidos para a Estimação de Movimento Fracionária do AV1. O capítulo 7, apresenta as arquiteturas desenvolvidas para as etapas da *Warped Motion* (Local e Global) e os resultados respectivos destas arquiteturas. Por fim, o capítulo 8, apresenta as conclusões dessa tese.

2 CONCEITOS DE COMPRESSÃO DE VÍDEOS E O FORMATO AV1

O vídeo digital é composto por uma série de imagens estáticas (chamadas de quadros ou *frames*), onde cada imagem individual não possui movimento próprio; porém, apresentadas ao espectador sequencialmente, em uma taxa temporal específica, fornecem uma sensação de movimento suficientemente alta para garantir uma percepção visual suave sem transições. Normalmente, a taxa de quadros para garantir a percepção suave do movimento é de cerca de 30 quadros por segundo (*Frames por Second* - fps), levando em consideração o Sistema Visual Humano (*Human Visual System* - HVS) (RICHARDSON, 2003). Entretanto, as aplicações de vídeo atuais apresentam requisitos mais elevados, aumentando a necessidade de taxas de quadros para até 120 fps, a fim de fornecer vídeos digitais com uma percepção realista de movimento, especialmente para resoluções mais elevadas, como UHD 4K e 8K.

Cada quadro é representado por uma matriz de píxeis bidimensionais, com dimensões WxH, sendo W a dimensão horizontal (largura) e H a dimensão vertical (altura), denominada resolução espacial do vídeo.

Os píxeis armazenam informações de cor e brilho para sua posição correspondente dentro de cada quadro. Para vídeos coloridos, um pixel é composto por três amostras de cor. Vários espaços de cor definem essa representação numérica de propriedades de pixel, como, por exemplo, o YCbCr (luminância, crominância azul e crominância vermelho), amplamente usado nos codificadores de vídeo (RICHARDSON, 2002).

É fácil imaginar que, em uma série de quadros sequenciais de um vídeo, existam muitas semelhanças entre os quadros. Essas semelhanças são chamadas de redundância temporal, uma vez que existe uma grande quantidade de informações repetidas no tempo na representação de vídeos. Também há o que é chamado de “redundância espacial”, pois píxeis vizinhos em um quadro tendem a ser semelhantes. Assim, os vídeos não comprimidos possuem elevada redundância em termos de textura (temporal e espacial) (SHI; SUN, 2017). Por fim, existe também a redundância entrópica, que diz respeito às probabilidades de ocorrência dos símbolos nos vídeos.

Deste modo, armazenar vídeos sem compressão, ou seja, armazenar vídeos

cheios destas redundâncias, consome muito espaço de armazenamento, e transmitir esses vídeos consome uma elevada largura de banda (CHEN; CRANTON; FIHN, 2016).

Portanto, os codificadores de vídeo procuram reduzir essas redundâncias, para que menos espaço de armazenamento seja usado e uma menor largura de banda seja consumida ao representar e transmitir esses vídeos. Essa redução, foca no descarte de dados redundantes ou irrelevantes para o HVS, visando transmitir apenas os dados necessários para descrever as informações relevantes do vídeo. Dessa maneira, os codificadores de vídeo conseguem reduzir, de maneira significativa, a quantidade de dados necessários para representar um vídeo (SHI; SUN, 2017).

Os codificadores atuais seguem um modelo chamado de codificação híbrida. As principais etapas de processamento de um codificador híbrido, incluindo o AV1, estão representadas na Figura 3. Inicialmente, cada quadro do vídeo de entrada é divido em blocos, que são processados de forma independente. Então, cada bloco passa por uma etapa de predição, que pode ser intra quadro, que busca redundâncias de informações dentro do mesmo quadro, ou inter quadros, que busca por redundâncias em quadros anteriormente processados. É importante salientar que as predições não são perfeitas, o que leva a geração de resíduos, através da subtração do resultado da predição e o bloco original. Os resíduos, por sua vez são processados pela etapa de “transformadas” (T na Figura 3) e “quantização” (Q na Figura 3). As transformadas transformam a representação dos resíduos para o domínio das frequências e, nesse domínio, a quantização elimina ou atenua as frequências para as quais o sistema visual humano é menos sensível. Esse processo de quantização gera perdas de informação e, quanto maior a perda, maior a taxa de compressão e menor a qualidade visual do vídeo; então, os codificadores atuais possuem algum tipo de controle para seleção do ponto de operação ideal na quantização. Finalmente, a informação quantizada é processada pelo codificador de entropia, que reduz a redundância entrópica na representação dos símbolos de entrada. Após a codificação de entropia, os dados estão prontos para serem transmitidos ou armazenados. Como a etapa de quantização gera perdas de informação, para garantir que o codificador e o decodificador usem as mesmas referências, o codificador possui as etapas de quantização e transformadas inversas (QI e TI na Figura 3). O resultado da quantização inversa é novamente somado ao bloco original, para gerar o bloco reconstruído. Finalmente, o bloco reconstruído passa por uma etapa de filtragem, a fim de reduzir artefatos inseridos nos vídeos, no processo de codificação. Então o bloco é armazenado para ser usado como referência para predições futuras (GOEBEL, 2019).

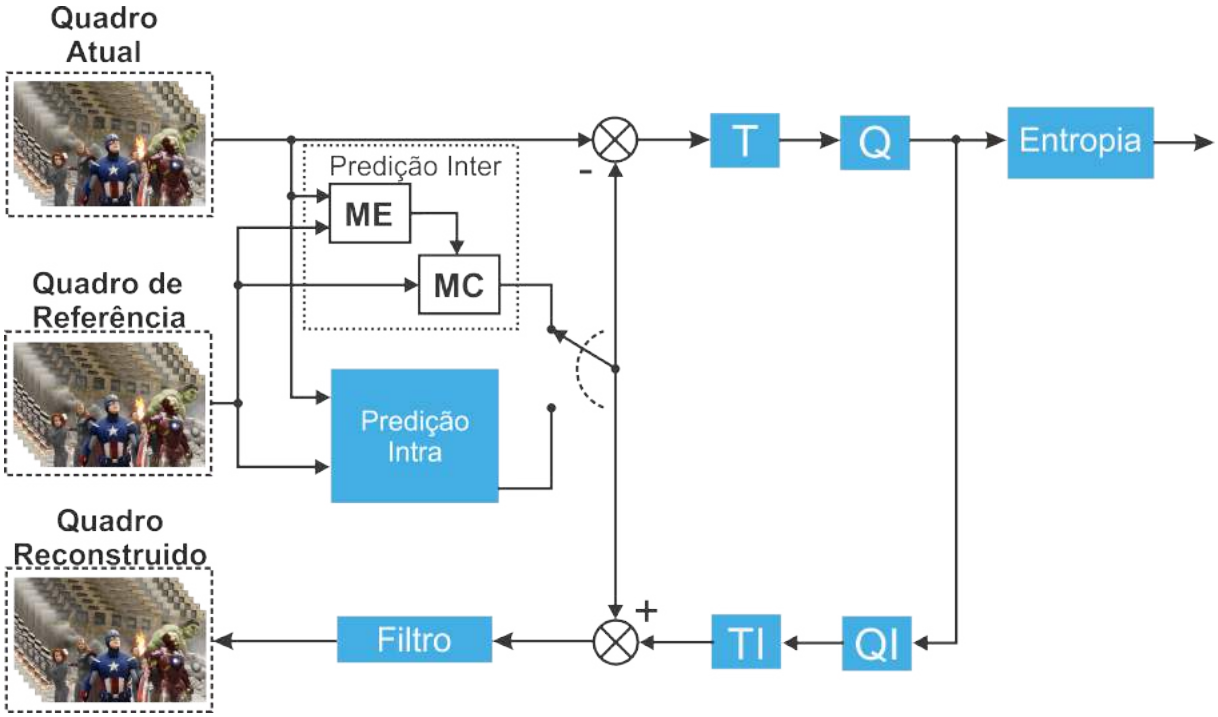

Figura 3 – Diagrama de blocos genérico de codificação de vídeo. Fonte: Adaptado de (AGOSTINI; SILVA; BAMPI, 2007).

2.1 Codificador de Video AV1

Em meados de 2015, algumas empresas como a *Google, Apple, Cisco, Samsung, IBM*, e mais um conjunto de 30 empresas líderes, de alta tecnologia, fundaram a *Alliance for Open Media* (AOMedia), para trabalhar em conjunto no objetivo de desenvolver um codificador de vídeo aberto de próxima geração que foi chamado de *AOMedia Video 1* ou AV1 (AOMEDIA, 2012). Seu desenvolvimento partiu do princípio do aprimoramento das ferramentas do VP9 (Mukherjee et al., 2013) do Google, adicionando algumas inovações propostas nos codificadores *Thor* (Bjøntegaard et al., 2016) e *Daala* (XIPH.ORG, 2018), da *Cisco* e da *Mozilla*, respectivamente, ambas empresas integrantes do *AOMedia*. Em seguida, novas ferramentas de codificação foram propostas, testadas, discutidas e incorporadas no codec da *AOMedia*. No decorrer dos anos, algumas versões de testes foram lançadas e, em meados de 2018, a primeira versão do AV1 foi lançada, incorporando uma variedade de novas ferramentas de compressão, além de recursos de alto nível, projetados para casos de uso específicos. O codificador AV1 foi criado com o intuito de: (i) proporcionar a entrega de vídeo em tempo real, em alta qualidade, (ii) ter escalabilidade para dispositivos modernos em várias larguras de banda, (iii) ter um bom desempenho computacional, (iv) ser adequado para ser otimizado para hardware e (v) buscar flexibilidade para conteúdos comerciais e não comerciais (Chen et al., 2018).

As próximas seções apresentam as principais características do AV1.

2.1.1 Partição de Blocos

O AV1 segue o mesmo modelo de codificação híbrida baseado em blocos, que já foi apresentado. Inicialmente, cada quadro é divido em blocos de tamanhos iguais, chamados de *SuperBlocks* (SB), que podem ter tamanhos de 128 x 128 ou 64 x 64 píxeis.

Os SBs podem ser divididos em tamanhos menores, visando seu processamento. Todos os tipos de particionamento que o codificador AV1 suporta, estão apresentados na Figura 4 (Chen et al., 2018).

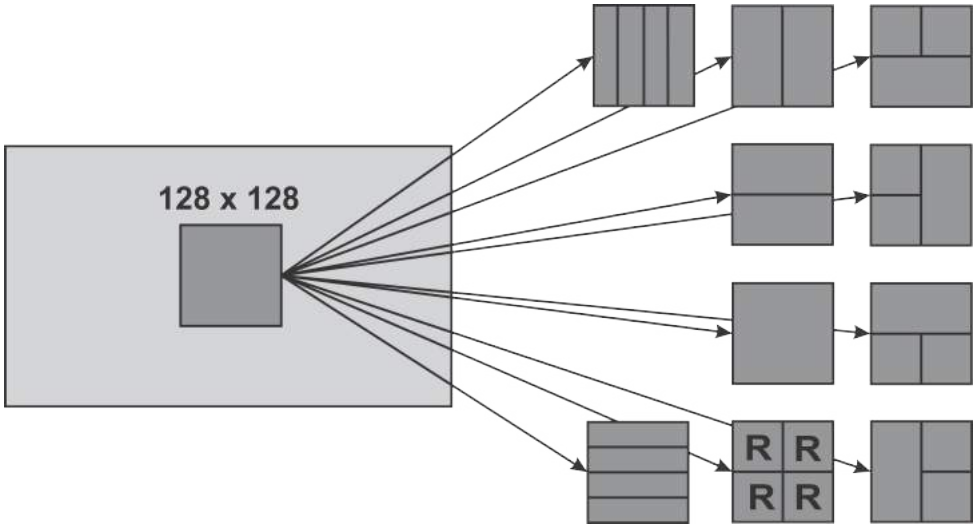

Figura 4 – Particionamento do codificador AV1. Fonte: Adaptado de (Chen et al., 2018).

O processo de partição de blocos gera uma árvore, onde o SB é a raiz desta árvore. Sempre que um bloco quadrático é considerado (blocos R na Figura 4), esse bloco pode ser subdividido, recursivamente, nos vários tipos de blocos permitidos pelo AV1 e apresentados na Figura 4.

Observando a Figura 4 é possível perceber que o bloco pode ser particionado de forma retangular em dois blocos, verticais ou horizontais. Outro modo de subdivisão do bloco é de maneira assimétrica e, neste caso, os blocos são particionados em quatro blocos com tamanhos idênticos, sendo a orientação das suas partições retangulares de 4:1 ou 1:4. Por último, existe o particionamento tipo T. Neste tipo de particionamento sempre serão gerados três blocos: dois blocos quadrados e um bloco retangular. Nenhuma destas partições não quadráticas pode ser subdivida em novos blocos.

O processo recursivo de particionamento tem sua condição de parada no tamanho de bloco de 4 x 4 amostras para luminância, que é o menor tamanho de bloco suportado no codificador AV1. Considerando a subamostra de informações de crominância suportada pelo AV1, em alguns casos bem específicos, os blocos de crominância podem ter tamanho mínimo de 2 x 2 amostras (GOEBEL, 2019) (HAN et al., 2021).

Existem ainda algumas situações especiais que restringem o particionamento em alguns casos. As partições assimétricas 1:4 e 4:1 não estão disponíveis para blocos 128 x 128 e 8 x 8 e os blocos 8x8 não podem ser particionados em T (Chen et al., 2019).

Com base nessas informações, é possível concluir que o AV1 pode suportar até 22 tamanhos de blocos, no processo de codificação para as amostras de luminância (Chen et al., 2019), partindo do pressuposto que o SB inicial possui 128 x 128 amostra de luminância. Deste modo, os tamanhos de blocos suportados são: 128 x 128, 128 x 64, 64 x 128, 64 x 64, 64 x 32, 64 x 16, 32 x 64, 16 x 64, 32 x 32, 32 x 16, 32 x 8, 16 x 32, 8 x 32, 16 x 16, 16 x 8, 16 x 4, 8 x 16, 4 x 16, 8 x 8, 8 x 4, 4 x 8 e 4 x 4 (AOMEDIA, 2020).

2.1.2 Predição Intra Quadro

A etapa de predição intra quadro do AV1 é encarregada de reduzir a redundância espacial encontrada nos quadros de um vídeo, sendo ela associada à correlação entre as amostras especialmente próximas de um mesmo quadro. Como as amostras vizinhas costumam ser muito semelhantes, ao invés de representar amostras individuais, separadamente, as amostras de cada bloco são representadas por texturas e arestas direcionais de amostras vizinhas, já processadas. Dessa forma, é possível reduzir o número de bits necessários para representar um quadro. Esta etapa acontece dentro de um quadro, e ocorre sem a necessidade de usar outros quadros como referência (BULL; ZHANG, 2021).

O codificador AV1 oferece mais modos de predição que seu antecessor, o VP9, tendo um total de 69 modos de predição intra. Esta etapa é explorada de várias formas e é um dos módulos mais inovadores no AV1. Para maiores ganhos de taxa de compressão, o AV1 define novos modos de predição, novas ferramentas e melhora alguns modos de predição, que já existiam no padrão HEVC ou no VP9 (GOEBEL, 2019). Melhorias são definidas no *Intra Orientation Prediction*, que passa a explorar mais tipos de redundância espacial em texturas direcionais. O modo *in-direction* é estendido para variar ângulos com ajuste de granularidade mais fino, e outra novidade, é aplicar um filtro passa-baixa aos valores da amostra antes de realizar a predição do bloco (Chen et al., 2018).

Já os modos não direcionais ganharam um número maior de preditores, como *Smoth, paeth, Chroma-from-Luma, Recursive-based-filtering Mode e DC*, os quais proporcionam uma melhor detecção da redundância espacial e, consequentemente, melhores taxas de compressão. Outra inovação que o AV1 trouxe na predição intra, são os modos: *Intra Block Copy* (Intra - BC) e o *Color Palette*, direcionados para vídeos de conteúdo de tela (Chen et al., 2018).

2.1.3 Predição Inter Quadros

A predição inter quadros explora a redundância temporal de um vídeo e precisa referenciar outros quadros, para realizar a predição. Portanto, a redundância temporal está diretamente relacionada entre as amostras de quadros sucessivos no vídeo, pois quadros próximos tendem a ter um nível de similaridade muito elevado, o que implica em uma alta semelhança entre as amostras ao longo ao tempo. Sendo assim, são aplicadas ferramentas para que seja possível estimar a redundância temporal, e são usados vetores de movimento para sinalizar mudanças nas posições dos blocos ao longo do tempo. Com a estimativa realizada, entra em cena um processo de compensação de movimento, que é usado para ajustar o conteúdo do quadro de referência, de acordo com o modelo de movimento e, assim, gerar o bloco atual reconstruído (SHI; SUN, 2017), que será subtraído do bloco original para gerar os resíduos.

O AV1 usa de inúmeras ferramentas que o auxiliam na estimação da redundância temporal, como as listadas abaixo:

- Estimação de Movimento Inteira: A Estimação de Movimento Inteira (*Integer Motion Estimation* - IME) é empregada exclusivamente no codificador com base no canal de luminância. Para cada segmento do SB particionado, a IME deve localizar, em um ou mais quadros de referência, o segmento que mais se assemelha ao segmento atual;

- Estimação de Movimento Fracionária: A Estimação de Movimento Fracionária (*Fracional Motion Estimation* - FME) é executada com o intuito de refinar o resultado encontrado na FME com a busca de movimentos mais suaves. Portanto, a FME é um processo que gera subamostras fracionárias em torno das amostras inteira;

- Compensação de Movimento Convencional: A Compensação de Movimento (*Motion Compensation* - MC) convencional é responsável por reconstruir os blocos previstos pela etapa da Estimação de Movimento;

- Compensação de Movimento de Bloco Sobreposto: No AV1 foi definido um algoritmo de compensação de movimento que minimiza os erros de previsão próximos às bordas do bloco, denominado Compensação de Movimento de Bloco Sobreposto (*Overlapped Block Motion Compensation* - OBMC). A OBMC é capaz de explorar as informações dos MV vizinhos espaciais mais próximos melhorando, assim, a qualidade de predição ao redor das bordas;

- Compensação de Movimento Distorcido: O AV1 implementa a Compensação de Movimento Distorcido (*Warped Motion Compensation* - WMC), com o intuito de

trazer melhorias a eficiência de codificação, que tem por base as transformações afim (*affine*);

- Modo de Predição Composta Avançada:A predição composta avançada tem como ideia principal gerar uma média ponderada entre duas predições diferentes do mesmo bloco, mais a máscara (peso) para, assim, resultar em uma predição final.

As ferramentas da predição inter quadros, que são foco deste trabalho, serão discutidas em maiores detalhes no Capítulo 3 desta tese.

2.1.4 Transformadas e Quantização

As etapas de predição intra quadros e inter quadros geram resíduos que precisam ser codificados. Esses resíduos passam pelas Transformadas e Quantização para, posteriormente, passarem pelo processo de codificação de Entropia.

As Transformadas têm como objetivo transformar os resíduos das predições intra ou inter quadros do domínio espacial para o domínio das frequências, preparando, assim, os dados para a quantização. O AV1 permite o uso de até quatro tipos diferentes de transformadas, e são elas: (i) a Transformada Discreta do Cosseno - *Discrete Cosine Transform* (DCT), (ii) a Transformada Discreta do Seno Assimétrica - *Asymmetric Discrete Sine Transform* (ADST), (iii) a Transformada Discreta do Seno Assimétrica Invertida - *Flipped Asymmetric Discrete Sine Transform* (FlipADST) e (iv) a Transformada Identidade - *Identity Transform* (IDTX) (Chen et al., 2018). Usando o princípio da separabilidade, transformadas diferentes podem ser usadas no sentido horizontal e vertical do bloco, como por exemplo DCTxADST (GOEBEL, 2019). Além disso, para blocos preditos pela predição intra, uma segunda etapa de transformadas pode ser aplicada, através da transformada denominada *Low Frequency Non-Separable Secondary Transform* (LFNSST) (ZHAO et al., 2016).

Já a etapa de Quantização ocorre após a etapa da Transformada ser concluída, e tem como principal objetivo reduzir informações imperceptíveis ao sistema visual humano. A etapa de Quantização é realizada no domínio das frequências, porém, ao contrário da Transformadas, a Quantização é um processo que gera perdas (CHEN; CRANTON; FIHN, 2016). Essas perdas são função do parâmetro de quantização, denominado *Constrained Quality* (CQ), que varia de 0 a 63 (HAN et al., 2021), cujo valor é ajustado para atender a eficiência de codificação desejada, ou seja, a relação entre o tamanho do arquivo e a qualidade do vídeo.

2.1.5 Codificação de Entropia

A etapa de codificação de entropia é uma técnica que visa a eliminação da redundância entrópica, ela é responsável por reduzir o número de bits necessários para

representar o *bitstream* produzido pelo codificador, reduzindo, assim, a redundância na representação dos símbolos. A representação dos símbolos é modificada com o uso de codificação de comprimento de palavra variável (AFONSO; AGOSTINI; SUSIN, 2019).

O AV1 utiliza o método *symbol to symbol adaptive multi-symbol* para comprimir os elementos da sintaxe, sendo que cada elemento é considerado um membro de um alfabeto específico, de N elementos e o contexto de entropia, consiste em um conjunto de N probabilidades que, por sua vez, são armazenadas em funções de distribuição cumulativa *Cumulative Distribution Function* (CDF). Um bloco ou quadro pode conter vários contextos de entropia, sendo que cada bloco escolhe o contexto de entropia que irá usar, com base nos modos selecionados pelos blocos vizinhos que, por sua vez, utilizam um codificador de entropia chamado Codec Booleano para codificar o fluxo de bits inteiro (Chen et al., 2018).

2.1.6 Filtragem de Laço

A etapa de Filtragem de Laço é responsável por reduzir os artefatos inseridos no vídeo durante todo o processo de codificação pois, como o processo de codificação é baseado em blocos, certos artefatos visuais podem surgir como descontinuidades entre blocos vizinhos (PALAU et al., 2022).

O codificador AV1 estabelece o uso de três filtros de laço, sendo eles o Filtro de Deblocagem - *Deblocking Filter* (DBF), Filtro de Aprimoramento Direcional Restrito - *Constrained Directional Enhancement Filter* (CDEF) e o Filtro de Restauração de Laço Comutável - *Switchable Loop Restoration Filter* (SLRF) (Han; Chiang; Xu, 2017).

2.2 Métricas de Avaliação

Na literatura existem diversas métricas que visam avaliar a eficiência da codificação de vídeos. Dentre todas essas métricas, existem duas principais abordagens em suas análises, que podem ser objetiva e subjetiva. Sendo que, as métricas que tem uma abordagem objetiva usam de modelos matemáticos para realizar a comparação entre o vídeo original e o vídeo modificado (codificado). Já as métricas que se baseiam em avaliações subjetivas levam em conta a percepção do usuário. Neste trabalho, são usada apenas métricas objetivas.

O *Peak Signal-to-Noise Ratio* (PSNR) foi o método objetivo usado neste trabalho, uma vez que este é o método mais usado pela comunidade da área. Este é um método que utiliza de métricas de comparações de similaridade entre dois sinais baseada em MSE (erro quadrático médio) (BULL; ZHANG, 2021). Sua medida é feita em decibéis (dB), e, quanto maior seu valor, maior a similaridade entre os sinais.

Entretanto, ponderar a eficiência de um codificador não é simples, pois existem

vários padrões de codificação, e cada um pode usar métricas e parâmetros diferentes. Para resolver este problema é necessário comparar diferentes sinais, de forma justa e ponderada e em igualdade de condições. As métricas mais comuns para definir a eficiência da codificação são *Bjøntegaard Delta PSNR* (BD-PSNR) e *Bjøntegaard Delta Bit Rate* (BD-BR), que medem a qualidade dos sinais (PSNR) e a quantidade de dados necessários para representá-los (taxa de bits).

O BD-PSNR busca calcular a diferença média da variação do PSNR (em decibéis) quando um codificador âncora e um codificador sob avaliação atingem a mesma taxa de bits (*bitrate*) (BJONTEGARD, 2001). Um valor negativo de BD-PSNR significa perda de eficiência de codificação (menor qualidade para a mesma taxa de bits), enquanto um valor positivo implica em uma melhoria na eficiência de codificação (maior qualidade para a mesma taxa de bits).

Já o BD-BR computa a variação percentual da taxa de bits entre o codificador âncora e o codificador sob avaliação, considerando vídeos com a mesma qualidade objetiva (PSNR) após a codificação (BJONTEGAARD, 2008). Neste caso, um BD-BR positivo significa perda de eficiência de codificação (maior taxa de bits para a mesma qualidade), enquanto um BD-BR negativo implica em um ganho na eficiência de codificação (menor taxa de bits para a mesma qualidade).

Para a realização do cálculo dessas métricas são fundamentais amostras diferentes de um mesmo sinal. Sendo que, a eficiência de codificação de uma solução é codificada para quatro parâmetros diferentes de *Constrained Quality* (CQ), segundo os documentos que definem as condições comuns de teste (ZEN, 2020).

3 FERRAMENTAS DA PREDIÇÃO INTER QUADROS DO CODIFICADOR AV1

Neste capítulo estão apresentadas, com mais detalhes, as ferramentas da predição inter quadros do codificador AV1 que são foco desta tese. Assim, será dado destaque para as etapas da Estimação de Movimento Fracionária, da Compensação de Movimento Convencional e da Compensação de Movimento Distorcido, que compõem o módulo da predição inter quadros do codificador AV1. Como também, um breve detalhamento sobre a etapa da Estimação de Movimento Inteira para facilitar o entendimento das demais etapas.

3.1 Estimação de Movimento

A predição inter quadros do codificador de vídeo AV1, igualmente aos demais codificadores, é composto principalmente pela etapa Estimação de Movimento (*Motion Estimation* - ME) que tenta identificar e eliminar redundâncias temporais entre os quadros sucessivos que formam uma imagem dinâmica. Essas redundâncias temporais podem ser observadas de diferentes maneiras. Um exemplo prático seria o deslocamento de um carro por uma pista. O fundo (pista) se mantém sem o movimento por vários quadros, considerando que a câmera permaneça parada, já o carro, sofre um pequeno deslocamento da sua posição em relação aos demais quadros. Sendo assim, após o primeiro deslocamento, a diferença entre os quadros sucessivos seria somente a posição do carro a cada instante. Portanto, a ME é empregada para investigar essas semelhanças e, em seguida, prevê o quadro atual, que está sendo codificado, usando quadros de referência.

Para prever o quadro atual ele é dividido em blocos, que são comparados com os blocos do quadro de referência, conforme é possível ver na Figura 5. Para essa comparação, um critério de similaridade é empregado aos blocos de uma área de busca no quadro de referência, o qual geralmente fica ao redor a posição original. Porém, nada impede que a área de busca seja definida a partir de outro ponto, o que proporciona uma maior probabilidade de aproximação da posição original, além de

reduzir a complexidade computacional quando comparado à área total de um quadro (AFONSO; AGOSTINI; FRANCO, 2013). A seguir, é gerado um Vetor de Movimento (*Motion Vector* - MV) que direcionará o bloco que gerou o melhor casamento com o bloco atual, ou seja, a maior similaridade entre o bloco do quadro atual e o bloco de referência.

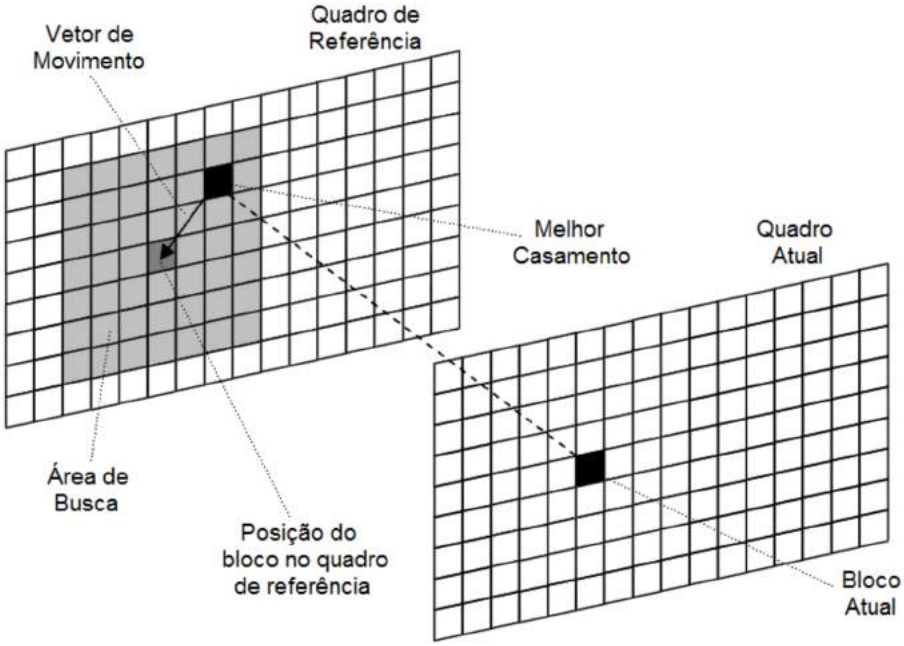

Figura 5 – Elementos presentes na Estimativa de Movimento (AFONSO; AGOSTINI; FRANCO, 2013).

Para alcançar esse resultado esperado, a ME é executada em duas etapas: Estimação de Movimento Inteira (*Integer Motion Estimation* - IME) e Estimação de Movimento Fracionaria (*Fracional Motion Estimation* - FME).

3.1.1 Estimação de Movimento Inteira

A Estimação de Movimento Inteira (*Integer Motion Estimation* - IME), é aplicada no codificador considerando apenas informações de luminância. Para cada bloco, a IME deve localizar, em um ou mais quadros de referência, qual bloco mais se assemelha ao bloco atual. Após a IME encontrar o melhor bloco no quadro de referência, ela gera um Vetor de Movimento (MV), indicando a posição deste bloco no quadro de referência. Os MV são definidos como vetores de duas dimensões (x, y) e indicam o deslocamento espacial de um bloco em um determinado quadro, em relação à posição do bloco atual no quadro atual.

O codificador AV1 permite o uso de múltiplos quadros de referência, permitindo um máximo de oito quadros em seu *buffer* de quadro decodificado. O quadro atual a ser codificado pode escolher qualquer um entre os sete possíveis quadros para usar

como seu quadro de referência. O codificador é capaz de atribuir explicitamente, a cada referência, um índice de quadro de referência exclusivo, variando de 1 a 7. Os quadros de referência com índice 1 a 4 são designados para os quadros que precedem o quadro atual, enquanto os quadros de índice 5 a 7 são para referências posteriores ao quadro atual, em relação a ordem de exibição (HAN et al., 2021). Isso é possível pois o AV1, como outros codificadores atuais, permite o processamento dos quadros fora da ordem de captura, uma vez que, antes de ser usado como referência, o quadro precisa ter sido codificado. Esse processo está apresentado na Figura 6.

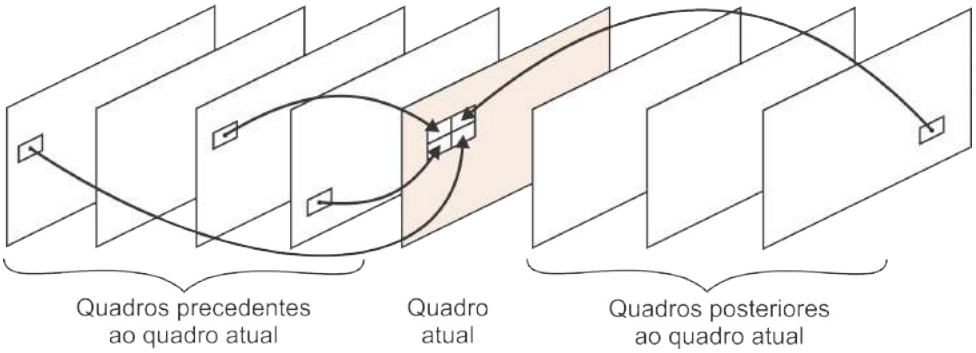

Figura 6 – Exemplo de uso de múltiplos quadros de referência.

Para encontrar o melhor bloco no quadro de referência, a IME utiliza um algoritmo de busca. Esses algoritmos variam, desde a busca exaustiva, até algoritmos sub ótimos (algoritmos rápidos), que visam reduzir a complexidade da IME. Entretanto, estes algoritmos rápidos podem introduzir impactos negativos em relação à eficiência da codificação (AGOSTINI; SILVA; BAMPI, 2007).

Além dos algoritmos de busca, a IME utiliza um critério de similaridade para decidir qual é o melhor casamento na procura sobre o quadro de referência. A soma de diferenças absolutas (*Sum of Absolute Differences* - SAD) (Vanne et al., 2006), é o critério mais usado para essa função, principalmente para projetos em hardware. O SAD consiste na soma das diferenças absolutas entre as amostras do bloco atual e as amostras do bloco candidato do quadro de referência. Deste modo, o bloco candidato que atingir o menor SAD será escolhido para representar o bloco atual. A Equação 1 define o cálculo do SAD, sendo R as amostras do bloco de referência e O as amostras do bloco do quadro de referência.

$$SAD = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} |R_{i,j} - O_{i,j}| \tag{1}$$

3.1.2 Estimação de Movimento Fracionária

Assim que o melhor vetor de movimento na etapa da IME é encontrado para o bloco atual, a Estimação de Movimento Fracionária (*Fracional Motion Estimation* - FME) é

executada, no intuito de refinar o resultado do bloco inteiro com a busca de movimentos mais suaves. Portanto, a FME é um processo que gera subamostras fracionárias em torno das amostras inteiras, de modo que, essas amostras fracionárias são obtidas através de interpolação com filtros de resposta ao impulso finito, *Finite Impulse Response* (FIR), e *Bilineares*.

O codificador AV1 estabelece um conjunto de 48 filtros para uso, sendo eles 42 filtros do tipo FIR ou Bilinear, que variam de 8 a 2 taps, com uma precisão de 1/8 de amostra para luminância, e seis filtros de identidade (RIVAZ; HAUGHTON, 2019). Os filtros de interpolação definidos no AV1 são:

i ***Regular***: um filtro de interpolação baseado em Lagrange que pode ter 6 ou 4 taps (7 filtros em cada caso);

ii ***Smooth***: um filtro de suavização projetado na janela Hamming que pode ter 6 ou 4 taps (7 filtros em cada caso);

iii ***Sharp***: um filtro de 8 taps baseado na Transformada Discreta dos Cossenos (*Discrete Cosine Transform* - DCT);

iv ***Bilinear***: um filtro de 2 taps usado em operações de interpolação rápida.

A Tabela 1 lista os filtros de interpolação: ***Regular*** (6 taps), ***Smooth*** (6 taps) e ***Sharp*** (8 taps) que são usados com tamanhos de blocos maiores que 4 x 4 (RIVAZ; HAUGHTON, 2019). A coluna Y indica a posição dos filtros para luminância, enquanto a coluna CbCr indica as posições para crominância. As demais colunas indicam, para cada filtro, quais os taps são utilizados. É importante destacar que, para cada posição da amostra a ser filtrada, existe um conjunto diferente de filtros para serem aplicados. É importante ressaltar ainda, que a FME considera apenas as amostras de luminância, por isso são usados apenas os filtros da coluna Y. Já a Compensação de Movimento, abordada na próxima seção, utiliza as amostras de luminância e de crominância.

A Tabela 2 lista os filtros de interpolação: ***Regular*** (4 taps), ***Smooth*** (4 taps) e ***Bilinear*** (2 taps) que são usados com tamanhos de blocos 4 x 4, ou no caso do ***Bilinear***, em casos de operações de interpolação rápida (RIVAZ; HAUGHTON, 2019).

Portanto, o processo de interpolação no AV1 pode selecionar seis conjuntos de filtros (considerando quatro famílias), sendo que cada família possui oito filtros em cada conjunto, de modo que, cada posição corresponde a uma posição fracionária e seus pesos variam conforme a amostra calculada, em um total de 48 filtros. É importante destacar que os coeficientes usados no cálculo das amostras de luminância $1/8$, $2/8$ e $3/8$ são semelhantes aos usados no cálculo das amostras $7/8$, $6/8$ e $5/8$, mas rotacionando os coeficientes (os filtros são simétricos). Da mesma forma, os coeficientes usados no cálculo de amostras $1/16$ a $2/16$ são os mesmos usados em $15/16$ a $14/16$,

Tabela 1 – Coeficientes dos filtros: Regular 6 taps, Smooth 6 taps e Sharp 8 taps

Y	CBCr	Regular	Smooth	Sharp
-	1/16	{2, -6, 126, 8, -2}	{2, 28, 62, 34, 2}	{-2, 2, -6, 126, 8, -2, 2, 0}
1/8	2/16	{2, -10, 122, 18, -4}	{26, 62, 36, 4}	{-2, 6, -12, 124, 16, -6, 4, -2}
-	3/16	{2, -12, 116, 28, -8, 2}	{22, 62, 40, 4}	{-2, 8, -18, 120, 26, -10, 6, -2}
2/8	4/16	{2, -14, 110, 38, -10, 2}	{20, 60, 42, 6}	{-4, 10, -22, 116, 38, -14, 6, -2}
-	5/16	{2, -14, 102, 48, -12, 2}	{18, 58, 44, 8}	{-4, 10, -22, 108, 48, -18, 8, -2}
3/8	6/16	{2, -16, 94, 58, -12, 2}	{16, 56, 46, 10}	{-4, 10, -24, 100, 60, -20, 8, -2}
-	7/16	{2, -14, 84, 66, -12, 2}	{-2, 16, 54, 48, 12}	{-4, 10, -24, 90, 70, -22, 10, -2}
4/8	8/16	{2, -14, 76, 76, -14, 2}	{-2, 14, 52, 52, 14, -2}	{-4, 12, -24, 80, 80, -24, 12, -4}
-	9/16	{2, -12, 66, 84, -14, 2}	{12, 48, 54, 16, -2}	{-2, 10, -22, 70, 90, -24, 10, -4}
5/8	10/16	{2, -12, 58, 94, -16, 2}	{10, 46, 56, 16}	{-2, 8, -20, 60, 100, -24, 10, -4}
-	11/16	{2, -12, 48, 102, -14, 2}	{8, 44, 58, 18}	{-2, 8, -18, 48, 108, -22, 10, -4}
6/8	12/16	{2, -10, 38, 110, -14, 2}	{6, 42, 60, 20}	{-2, 6, -14, 38, 116, -22, 10, -4}
-	13/16	{2, -8, 28, 116, -12, 2}	{4, 40, 62, 22}	{-2, 6, -10, 26, 120, -18, 8, -2}
7/8	14/16	{-4, 18, 122, -10, 2}	{4, 36, 62, 26,}	{-2, 4, -6, 16, 124, -12, 6, -2}
-	15/16	{-2, 8, 126, -6, 2}	{2, 34, 62, 28, 2}	{0, 2, -2, 8, 126, -6, 2, -2}

mas rotacionando os coeficientes. O codificador AV1 usa os mesmos conjuntos de filtros para as amostras de luminância e crominância, diferenciando apenas na precisão, de $1/8$ e de $1/16$, respectivamente.

A interpolação pode utilizar quaisquer filtros da Tabela 1 e da Tabela 2, coluna Y, para realizar a interpolação vertical e horizontal. A Figura 7 demonstra o processo da FME, pelo qual as amostras de subpixel são geradas, de modo que, todas as posições de subpixel estão associadas à posição da amostra inteira (amostra em azul). Sendo assim, todas as posições subpixel são geradas diretamente a partir dos valores referentes a amostra inteira, em uma ou duas etapas, que são (RIVAZ; HAUGHTON, 2019):

- As posições de subpixel na linha do pixel inteiro são geradas usando apenas filtragem horizontal;
- As posições de subpixel na coluna do pixel inteiro são geradas usando apenas filtragem vertical;
- As demais posições de subpixel (diagonais) são executadas em duas etapas, utilizando de uma combinação de filtros horizontais e verticais:
  - Primeiramente é executada a filtragem horizontal, sendo que, seu *buffer* é expandido na direção Y em ambas as extremidades do bloco, produzindo

Tabela 2 – Coeficientes dos filtros: Regular 4 taps, Smoth 4 taps e Bilinear 2 taps

Y	CBCr	Regular - 4 tap	Smoth - 4 tap	Bilinear
-	1/16	{-4, 126, 8, -2}	{30, 62, 34, 2}	{120, 8}
1/8	2/16	{-8, 122, 18, -4}	{26, 62, 36, 4}	{112, 16}
-	3/16	{-10, 116, 28, -6}	{22, 62, 40, 4}	{104, 24}
2/8	4/16	{-12, 110, 38, -80}	{20, 60, 42, 6}	{96, 32}
-	5/16	{-12, 102, 48, -10}	{18, 58, 44, 8}	{88, 40}
3/8	6/16	{-14, 94, 58, -10}	{16, 56, 46, 10}	{80, 48}
-	7/16	{-12, 84, 66, -10}	{14, 54, 48, 12}	{72, 56}
4/8	8/16	{-12, 76, 76, -12}	{12, 52, 52, 12}	{64, 64}
-	9/16	{-10, 66, 84, -12}	{12, 48, 54, 14}	{56, 72}
5/8	10/16	{-10, 58, 94, -14}	{10, 46, 56, 16}	{48, 80}
-	11/16	{-10, 48, 102, -12}	{8, 44, 58, 18}	{40, 88}
6/8	12/16	{-8, 38, 110, -12}	{6, 42, 60, 20}	{32, 96}
-	13/16	{-6, 28, 116, -10}	{4, 40, 62, 22}	{24, 104}
7/8	14/16	{-4, 18, 122, -8}	{4, 36, 62, 26}	{16, 112}
-	15/16	{-2, 8, 126, -4}	{2, 34, 62, 30}	{8, 120}

linhas filtradas que serão utilizadas pela filtragem vertical.

– Por fim, a filtragem vertical é executada.

Deste modo, a FME gera 63 amostras fracionárias para cada amostra inteira. Os blocos formados por essas amostras fracionárias são comparados utilizando um critério de similaridade, como a soma de diferenças absolutas (SAD), já apresentada. Os resultados de SAD dos blocos nas posições fracionárias são comparados com o resultado de SAD usando apenas amostras inteiras, com o intuito de buscar o bloco com menor SAD. Caso algum bloco de subpixel retorne um menor SAD em relação ao resultado da IME, então um novo vetor de movimento é gerado, considerando a posição fracionária.

3.2 Compensação de Movimento Convencional

A Compensação de Movimento (*Motion Compensation* - MC) Convencional é responsável por reconstruir os blocos previstos pela etapa da Estimação de Movimento, incluindo a IME e a FME. A MC usa os Vetores de Movimento como principal informação de entrada. Deste modo, a MC é capaz de copiar as informações do bloco com melhor resultado de eficiência no *buffer* de quadros já codificados (quadros passados e/ou futuros) e montar o quadro predito. Este quadro predito será então subtraído do

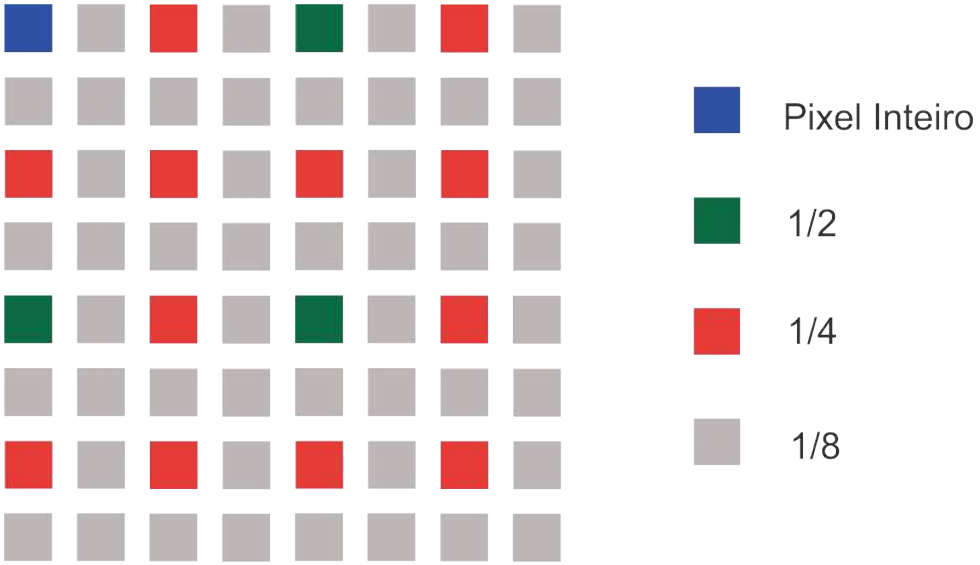

Figura 7 – Posição sub-pixel FME.

quadro atual gerando, assim, o quadro de resíduos que passará para a próxima etapa de codificação.

Os blocos reconstruídos pela MC são empregados tanto no codificador como no decodificador. No lado do codificador, os blocos reconstruídos são necessários para serem usados como referência para as predições inter dos próximos quadros e intra do quadro atual. Do lado do decodificador, além de servir de referência, a MC é também responsável por reconstituir os quadros que serão exibidos pelo usuário.

A Compensação de Movimento gera as amostras de bloco (amostras de luminância e crominância), usando os MV selecionados e os quadros de referência. No caso de amostras fracionárias, o processo de interpolação usado na MC do AV1 é baseado em filtros FIR e bilineares. Os filtros usados na MC são os mesmos apresentados na Tabela 1 e Tabela 2, lembrando que, como a MC gera as amostras de luminância e crominância, são usados todos os filtros apresentados nas tabelas (RIVAZ; HAUGHTON, 2019). Assim um total de 90 filtros podem ser usados na MC.

3.3 Compensação de Movimento Distorcido

O AV1 implementa a Compensação de Movimento Distorcido (*Warped Motion Compensation* - WMC), com o intuito de trazer melhorias a eficiência de codificação. A WMC tem como objetivo aplicar transformações não-translacionais durante a etapa de Compensação de Movimento de um determinado bloco, assim proporcionando capturar com maior precisão os movimentos presentes em cada quadro de um vídeo (Parker et al., 2017a).

A WMC tem por base as transformações afins (*affine*) e está dividida em duas partes principais: Estes, por sua vez, suportam movimentos como rotação, escala

e cisalhamento, enquanto as transformações translacionais suportam apenas deslocamento dos blocos pela imagem. A Subseção 3.3.1 aborda com mais detalhes a transformação afim.

A Compensação de Movimento Distorcido Global busca tratar os movimentos da câmera, os quais afetam o quadro todo. Já a Compensação de Movimento Distorcido Local visa capturar diversos movimentos locais, implicitamente, os quais afetam pequenas regiões no quadro. No nível local, os parâmetros são estimados a partir de blocos vizinhos, enquanto no nível global, os parâmetros são estimados usando modelos de estimação robustos. Entretanto, ambas competem com os modos de predição no nível do bloco e são selecionadas apenas se existir um ganho no custo (Parker et al., 2017b). A Subseção 3.3.2 exemplifica o funcionamento da LWMC e a Subseção 3.3.3 explica o funcionamento da GWMC do AV1.

3.3.1 Transformações Afins

Transformações afins são tipos de transformações geométricas que mantêm a colinearidade (se um conjunto de pontos estiver alinhado em uma linha antes da transformação, eles permanecerão alinhados em uma linha depois) e as proporções de distâncias entre pontos em uma linha (ZWILLINGER, 2012).

Transformações afins são classes significativas de transformações bidimensionais de geometria linear que alteram variáveis (por exemplo, valores de intensidade de pixel localizados em (x_1, y_1) em uma imagem de entrada) para novas variáveis (por exemplo, (x_2, y_2) em uma imagem de saída). Elas são realizadas por meio de combinações de translação, rotação, escala e cisalhamento, os quais são calculados por meio de uma matriz com nove parâmetros e oito graus de liberdade (SOLOMON; BRECKON, 2010).

O modelo geral de uma matriz da transformação afim é dado pela matriz apresentada em (2) (HARTLEY; ZISSERMAN, 2004):

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tag{2}$$

Na transformação afim translacional, na qual a imagem é deslocada verticalmente e horizontalmente em relação à sua posição original, a matriz é dada por (3) e os parâmetros utilizados são $h_{13} = x$ e $h_{23} = y$:

$$\begin{bmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Para a transformação afim rotacional, a qual a matriz é representada por (4) e os seus parâmetros $h_{11} = h_{22} = \cos\theta$ e $h_{21} = -h_{12} = \sin\theta$ são utilizados. Sendo que,

h_{11} e h_{22} simbolizam o cosseno do ângulo θ de rotação, por outro lado h_{21} e $-h_{12}$ simbolizam o seno.

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

A transformação afim do tipo escala é calculada utilizando os parâmetros $h_{11} = w$ e $h_{22} = h$. A transformação é dada pela matriz (5), porém é de suma importância observar que w necessita ser igual a h para que assim a escala seja proporcional.

$$\begin{bmatrix} w & 0 & 0 \\ 0 & h & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

Por fim, a transformação afim de cisalhamento usa de dois parâmetros, sendo eles, um para a direção vertical no eixo x, que é dado por $h_{12} = \tan\phi$, e outro para a direção horizontal no eixo y, que é dado por $h_{21} = \tan\psi$. A matriz de transformação é dada por (6):

$$\begin{bmatrix} 1 & \tan\phi & 0 \\ \tan\psi & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

Portanto, foi possível verificar todos os tipos que a transformada afim pode descrever, seus diversos tipos movimentos, e para melhor exemplificar cada tipo de transformada, a Figura 8 apresenta os 4 tipos citados anteriormente.

3.3.2 Compensação de Movimento Distorcido Local

O modelo de Compensação de Movimento Distorcido Local do AV1 implica na compensação de pequenos movimentos locais para um determinado bloco, em especial escala, rotação e cisalhamento, através do uso de informações dos MV dos blocos vizinhos, e é permitido para blocos de tamanho 8 x 8 ou superior.

A LWMC supõe que os movimentos locais podem ser ponderados através do padrão das atividades dos movimentos de seus vizinhos espaciais. Sendo assim, o codificador varre todos os vizinhos adjacentes mais próximos do bloco e descobre blocos cujo MV's apontem para o mesmo quadro de referência, de modo que, a LWMC permite um número máximo de oito blocos e um mínimo de dois blocos de referência (HAN et al., 2021). Os MVs e os blocos de referência encontrados são usados para inferir o movimento do bloco atual.

Como já mencionado, a LWMC usa dos MVs dos blocos vizinhos para extrair o movimento local, e como o AV1 não suporta transformações não colineares, os termos

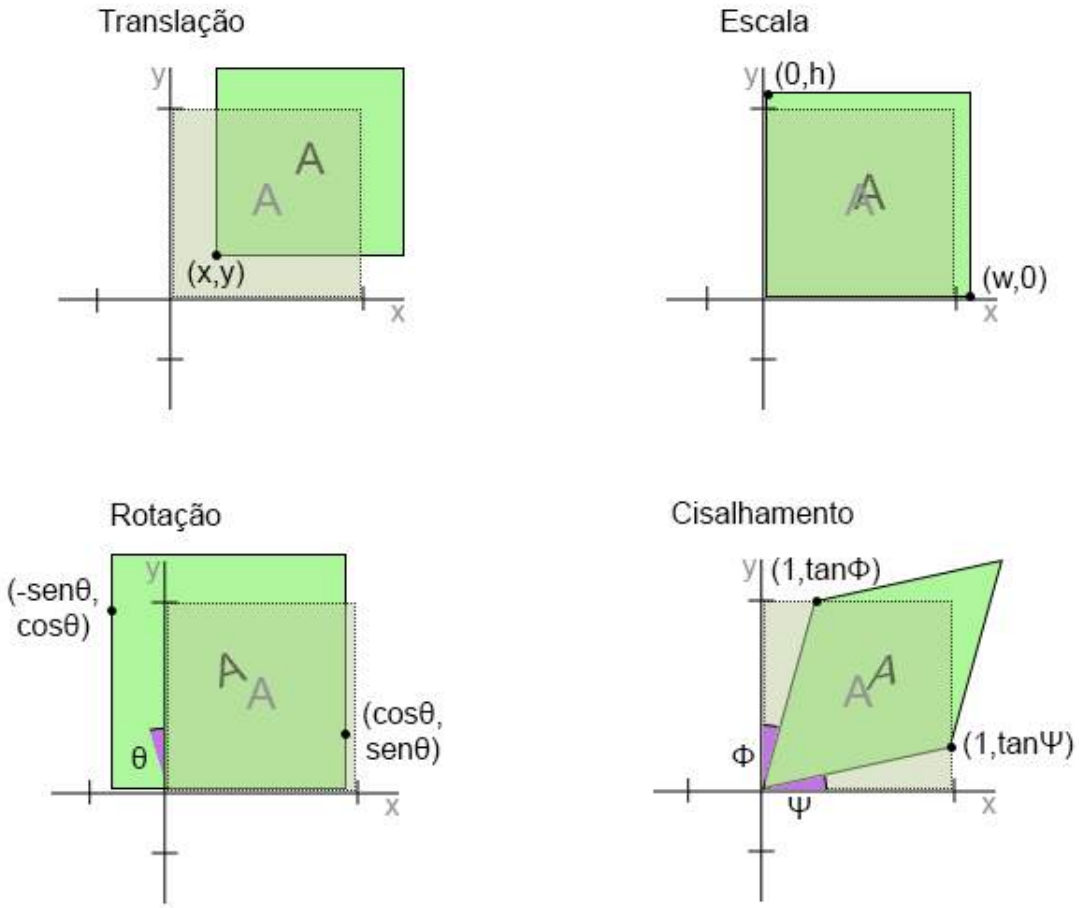

Figura 8 – Exemplos de transformações afins. Adaptado de (KOLODZIEJSKI; DOMANSKI; AGOSTINI, 2022).

h_{31}, h_{32} e h_{33} podem ser igualados a 1 e, consequentemente omitidos da matriz (2) (AOMEDIA, 2020). Deste modo, a matriz afim pode ser simplificada conforme (7):

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{7}$$

onde $\begin{bmatrix} x \\ y \end{bmatrix}$ e $\begin{bmatrix} x' \\ y' \end{bmatrix}$ representam as coordenadas de uma amostra no bloco corrente e as coordenadas distorcidas desta amostra no quadro de referência respectivamente.

A matriz (7) afim 3 x 2 tem seis graus de liberdade e permite transformar um retângulo em um quadrilátero arbitrário. Os parâmetros h_{13} e h_{23} correspondem aos parâmetros do modelo translacional. Para simplificar a estimativa do modelo, assume-se que h_{13} e h_{23} representam o vetor de movimento (MV) do bloco atual (MV_1 na Figura 9), onde $MV_1 = \binom{h_{13}}{h_{23}}$. Desta forma, a matriz pode ser reduzida a quatro graus

de liberdade, permitindo uma transformação retangular em paralelogramo e reduzindo os custos computacionais, resultando na matriz simplificada (8).

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{8}$$

A Figura 9 pode exemplificar a estimativa desses parâmetros. No bloco atual (em amarelo) $C_0 = \binom{x_0}{y_0}$ representa o centro do bloco e $C'_0 = \binom{x'_0}{y'_0}$ representa a projeção de C_0 no quadro de referência usando o vetor de movimento MV_0 para o bloco atual. Da mesma forma, no bloco de referência n (em verde), onde $C_n = \binom{x_n}{y_n}$ representa o centro do bloco e $C'_n = \binom{x'_n}{y'_n}$ representa a projeção de C_n no quadro de referência usando o vetor de movimento MV_n para o bloco n. Desde modo, o vetor de movimento referente a C_0 e C_n no bloco atual é mapeado para o vetor de movimento relacionado à C'_0 e C'_n no quadro de referência, conforme é possível observar na matriz (9) (AOMEDIA, 2022).

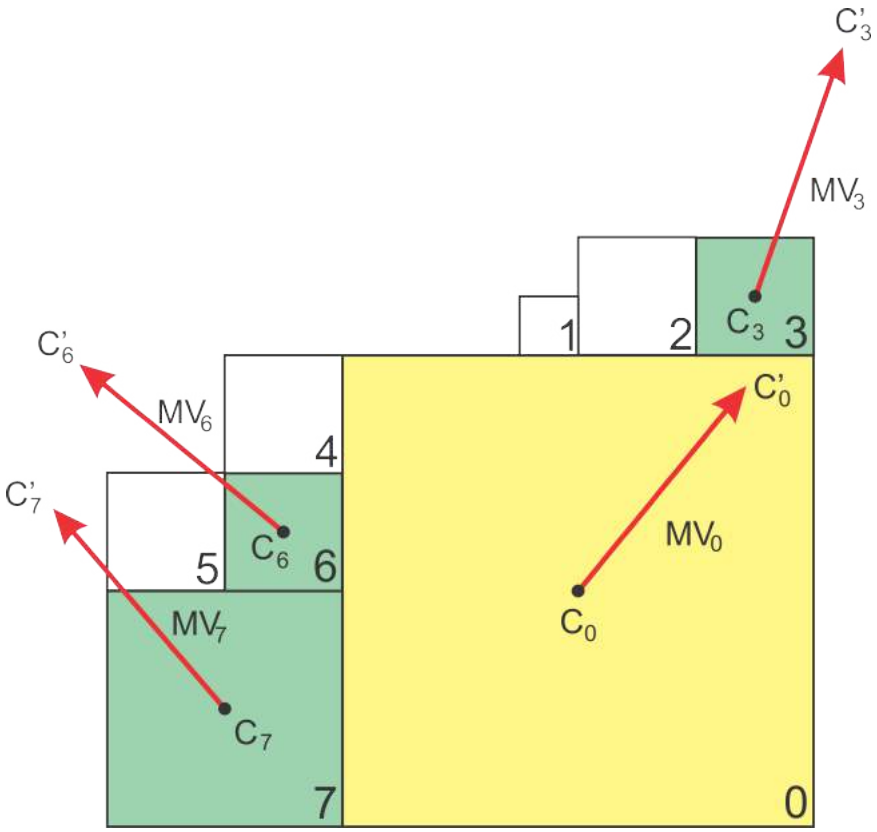

Figura 9 – Bloco atual em amarelo. Os blocos vizinhos que se referem à mesma imagem de referência que o bloco atual estão em verde. Adaptado de (Parker et al., 2017a).

$$\begin{bmatrix} x'_n & - & x'_0 \\ y'_n & - & y'_0 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_n & - & x_0 \\ y_n & - & y_0 \end{bmatrix} \tag{9}$$

Levando em conta que $n \in \{1, 2, 3\}$ é possível formular a matriz (10):

$$\begin{bmatrix} x'_1 - x'_0 & y'_1 - y'_0 \\ x'_2 - x'_0 & y'_2 - y'_0 \\ x'_3 - x'_0 & y'_3 - y'_0 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_1 - x_0 & y_1 - y_0 \\ x_2 - x_0 & y_2 - y_0 \\ x_3 - x_0 & y_3 - y_0 \end{bmatrix} \tag{10}$$

onde é possível aplicar uma regressão de mínimos quadrados com os termos:

$$X = \begin{bmatrix} x'_1 - x'_0 \\ x'_2 - x'_0 \\ x'_3 - x'_0 \end{bmatrix}, Y = \begin{bmatrix} y'_1 - y'_0 \\ y'_2 - y'_0 \\ y'_3 - y'_0 \end{bmatrix}, R = \begin{bmatrix} x_1 - x_0 & y_1 - y_0 \\ x_2 - x_0 & y_2 - y_0 \\ x_3 - x_0 & y_3 - y_0 \end{bmatrix} \tag{11}$$

O qual pode ser simplificado para:

$$\begin{bmatrix} X & Y \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} R \tag{12}$$

Finalmente resolvendo a estimativa de mínimos quadrados para $\left(\begin{smallmatrix} h_{11} & h_{21} \\ h_{12} & h_{22} \end{smallmatrix}\right)$ chega-se a matriz (13):

$$\begin{bmatrix} R^T R \end{bmatrix}^{-1} R^T \begin{bmatrix} X & Y \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} = H \tag{13}$$

Sendo assim, uma vez calculadas as amostras de projeção, os parâmetros h_{11}, h_{12}, h_{21} e h_{22} são estimados a partir dessas amostras pela minimização de mínimos quadrados da distância entre a referência e as projeções de dobra (Parker et al., 2017a).

As transformações afins podem ser interpretadas como transformações de cisalhamento (CROFT; FALCONER; GUY, 2012). Assim, a implementação LWMC do AV1 divide a transformação afim em dois cisalhamentos: um horizontal e um vertical. Isso é feito para aplicar os filtros de interpolação usados para melhorar a transformação final, que é aplicada nas direções horizontal e vertical. Esses cisalhamentos são separados da matriz de homografia original dada por (14):

$$\begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \gamma & 1 + \delta \end{bmatrix} \begin{bmatrix} 1 + \alpha & \beta \\ 0 & 1 \end{bmatrix} \tag{14}$$

A transformação final é dada por (15):

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \gamma & 1 + \delta \end{bmatrix} \begin{bmatrix} 1 + \alpha & \beta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{15}$$

Os parâmetros de cisalhamento α, β, γ, δ são determinados com base nos parâmetros h_{11}, h_{12}, h_{21} e h_{22}, respectivamente, usando filtros de interpolação de subpixel. A LWMC do AV1 define 192 filtros FIR de até 8 taps e com precisão de $1/64$ de pixel

(Chen et al., 2018), conforme apresentados na Tabela 3. O modelo final da LWMC é aplicado a blocos subdivididos com 8 x 8 píxeis, e o bloco predito final é construído reunindo todos os sub-blocos preditos.

O processo da LWMC aplica primeiro o cisalhamento horizontal e posteriormente aplica o cisalhamento vertical. Sendo que, é permitido a aplicação de um filtro separável a todo o bloco, ao invés de uma vez por pixel. Então, este processo possui as seguinte etapas (Parker et al., 2017a):

- Dividir o bloco atual em sub-blocos 8 x 8 e projetar o ponto central de cada sub-bloco nos blocos de referência;
- Filtrar horizontalmente, gerando 15 linhas de oito píxeis cada;
- Filtrar verticalmente, gerando o bloco de saída 8 x 8.

Por fim, todo o processo da Compensação de Movimento Distorcido Local pode ser observado na Figura 10.

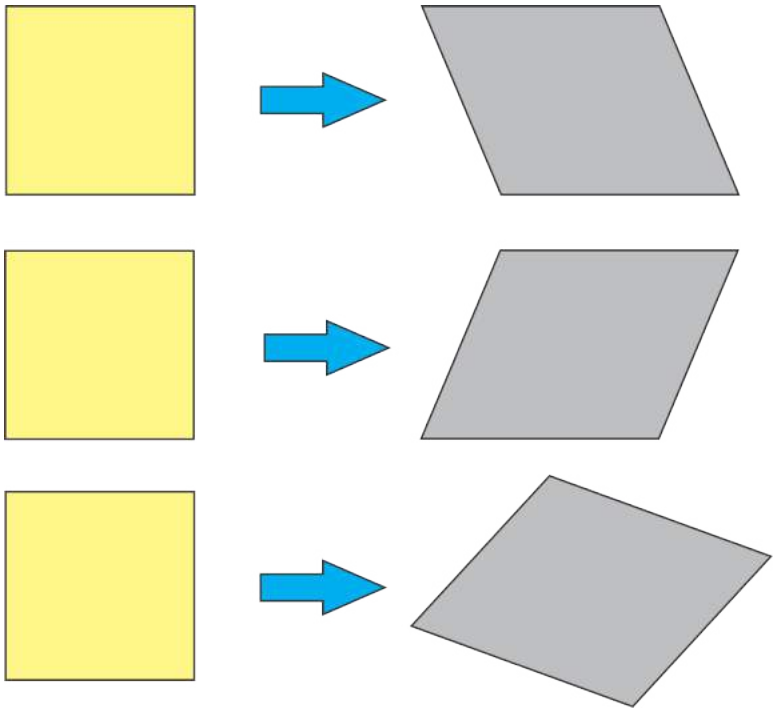

Figura 10 – Cisalhamento horizontal, vertical e combinado, respectivamente. Adaptado de (OPEN MEDIA, 2022).

3.3.3 Compensação de Movimento Distorcido Global

O modelo baseado em Compensação de Movimento Distorcido Global é responsável por identificar os movimentos que afetam o quadro como um todo, deste modo,

Tabela 3 – Filtros FIR da Compensação de Movimento Distorcido Local

Coeficientes filtros FIR		
0,0,127,1,0,0,0,0	0,-1,127,2,0,0,0,0	0,0,0,127,1,0,0,0
1,-3,127,4,-1,0,0,0	1,-4,126,6,-2,1,0,0	0,1,-3,127,4,-2,1,0
1,-5,126,8,-3,1,0,0	1,-6,125,11,-4,1,0,0	0,2,-6,126,8,-3,1,0
1,-7,124,13,-4,1,0,0	2,-8,123,15,-5,1,0,0	-1,3,-8,125,13,-5,2,-1
2,-9,122,18,-6,1,0,0	2,-10,121,20,-6,1,0,0	-1,4,-11,123,18,-7,3,-1
2,-11,120,22,-7,2,0,0	2,-12,119,25,-8,2,0,0	-1,4,-13,121,23,-8,3,-1
3,-13,117,27,-8,2,0,0	3,-13,116,29,-9,2,0,0	-1,5,-15,119,27,-10,4,-1
3,-14,114,32,-10,3,0,0	3,-15,113,35,-10,2,0,0	-2,6,-17,116,33,-12,5,-1
3,-15,111,37,-11,3,0,0	3,-16,109,40,-11,3,0,0	-2,6,-18,113,38,-13,5,-1
3,-16,108,42,-12,3,0,0	4,-17,106,45,-13,3,0,0	-2,7,-19,110,43,-15,6,-2
4,-17,104,47,-13,3,0,0	4,-17,102,50,-14,3,0,0	-2,7,-20,106,49,-16,6,-2
4,-17,100,52,-14,3,0,0	4,-18,98,55,-15,4,0,0	-2,7,-21,102,54,-17,7,-2
4,-18,96,58,-15,3,0,0	4,-18,94,60,-16,4,0,0	-2,8,-22,98,59,-18,7,-2
4,-18,91,63,-16,4,0,0	4,-18,89,65,-16,4,0,0	-2,8,-22,94,64,-19,7,-2
4,-18,87,68,-17,4,0,0	4,-18,85,70,-17,4,0,0	-2,8,-22,89,69,-20,8,-2
4,-18,82,73,-17,4,0,0	4,-18,80,75,-17,4,0,0	-2,8,-21,84,74,-21,8,-2
4,-18,78,78,-18,4,0,0	4,-17,75,80,-18,4,0,0	-2,8,-21,79,79,-21,8,-2
4,-17,73,82,-18,4,0,0	4,-17,70,85,-18,4,0,0	-2,8,-21,74,84,-21,8,-2
4,-17,68,87,-18,4,0,0	4,-16,65,89,-18,4,0,0	-2,8,-20,69,89,-22,8,-2
4,-16,63,91,-18,4,0,0	4,-16,60,94,-18,4,0,0	-2,7,-19,64,94,-22,8,-2
3,-15,58,96,-18,4,0,0	4,-15,55,98,-18,4,0,0	-2,7,-18,59,98,-22,8,-2
3,-14,52,100,-17,4,0,0	3,-14,50,102,-17,4,0,0	-2,7,-17,54,102,-21,7,-2
3,-13,47,104,-17,4,0,0	3,-13,45,106,-17,4,0,0	-2,6,-16,49,106,-20,7,-2
3,-12,42,108,-16,3,0,0	3,-11,40,109,-16,3,0,0	-2,6,-15,43,110,-19,7,-2
3,-11,37,111,-15,3,0,0	2,-10,35,113,-15,3,0,0	-1,5,-13,38,113,-18,6,-2
3,-10,32,114,-14,3,0,0	2,-9,29,116,-13,3,0,0	-1,5,-12,33,116,-17,6,-2
2,-8,27,117,-13,3,0,0	2,-8,25,119,-12,2,0,0	-1,4,-10,27,119,-15,5,-1
2,-7,22,120,-11,2,0,0	1,-6,20,121,-10,2,0,0	-1,3,-8,23,121,-13,4,-1
1,-6,18,122,-9,2,0,0	1,-5,15,123,-8,2,0,0	-1,3,-7,18,123,-11,4,-1
1,-4,13,124,-7,1,0,0	1,-4,11,125,-6,1,0,0	-1,2,-5,13,125,-8,3,-1
1,-3,8,126,-5,1,0,0	1,-2,6,126,-4,1,0,0	0,1,-3,8,126,-6,2,0
0,-1,4,127,-3,1,0,0	0,0,2,127,-1,0,0,0	0,1,-2,4,127,-3,1,0

Continua...

Coeficientes filtros FIR		
0,0,-1,127,2,0,0,0	0,0,0,1,127,0,0,0	0,0,0,-1,127,2,0,0
0,1,-5,127,6,-2,1,0	0,0,1,-3,127,4,-1,0	0,0,1,-4,126,6,-2,1
-1,2,-7,126,11,-4,2,-1	0,0,1,-5,126,8,-3,1	0,0,1,-6,125,11,-4,1
-1,3,-10,124,16,-6,3,-1	0,0,1,-7,124,13,-4,1	0,0,2,-8,123,15,-5,1
-1,4,-12,122,20,-7,3,-1	0,0,2,-9,122,18,-6,1	0,0,2,-10,121,20,-6,1
-2,5,-14,120,25,-9,4,-1	0,0,2,-11,120,22,-7,2	0,0,2,-12,119,25,-8,2
-1,5,-16,118,30,-11,4,-1	0,0,3,-13,117,27,-8,2	0,0,3,-13,116,29,-9,2
-2,6,-17,114,35,-12,5,-1	0,0,3,-14,114,32,-10,3	0,0,3,-15,113,35,-10,2
-2,7,-19,111,41,-14,6,-2	0,0,3,-15,111,37,-11,3	0,0,3,-16,109,40,-11,3
-2,7,-20,108,46,-15,6,-2	0,0,3,-16,108,42,-12,3	0,0,4,-17,106,45,-13,3
-2,7,-21,104,51,-16,7,-2	0,0,4,-17,104,47,-13,3	0,0,4,-17,102,50,-14,3
-2,8,-21,100,56,-18,7,-2	0,0,4,-17,100,52,-14,3	0,0,4,-18,98,55,-15,4
-2,8,-22,96,62,-19,7,-2	0,0,4,-18,96,58,-15,3	0,0,4,-18,94,60,-16,4
-2,8,-22,91,67,-20,8,-2	0,0,4,-18,91,63,-16,4	0,0,4,-18,89,65,-16,4
-2,8,-22,87,72,-21,8,-2	0,0,4,-18,87,68,-17,4	0,0,4,-18,85,70,-17,4
-2,8,-22,82,77,-21,8,-2	0,0,4,-18,82,73,-17,4	0,0,4,-18,80,75,-17,4
-2,8,-21,77,82,-22,8,-2	0,0,4,-18,78,78,-18,4	0,0,4,-17,75,80,-18,4
-2,8,-21,72,87,-22,8,-2	0,0,4,-17,73,82,-18,4	0,0,4,-17,70,85,-18,4
-2,8,-20,67,91,-22,8,-2	0,0,4,-17,68,87,-18,4	0,0,4,-16,65,89,-18,4
-2,7,-19,62,96,-22,8,-2	0,0,4,-16,63,91,-18,4	0,0,4,-16,60,94,-18,4
-2,7,-18,56,100,-21,8,-2	0,0,3,-15,58,96,-18,4	0,0,4,-15,55,98,-18,4
-2,7,-16,51,104,-21,7,-2	0,0,3,-14,52,100,-17,4	0,0,3,-14,50,102,-17,4
-2,6,-15,46,108,-20,7,-2	0,0,3,-13,47,104,-17,4	0,0,3,-13,45,106,-17,4
-2,6,-14,41,111,-19,7,-2	0,0,3,-12,42,108,-16,3	0,0,3,-11,40,109,-16,3
-1,5,-12,35,114,-17,6,-2	0,0,3,-11,37,111,-15,3	0,0,2,-10,35,113,-15,3
-1,4,-11,30,118,-16,5,-1	0,0,3,-10,32,114,-14,3	0,0,2,-9,29,116,-13,3
-1,4,-9,25,120,-14,5,-2	0,0,2,-8,27,117,-13,3	0,0,2,-8,25,119,-12,2
-1,3,-7,20,122,-12,4,-1	0,0,2,-7,22,120,-11,2	0,0,1,-6,20,121,-10,2
-1,3,-6,16,124,-10,3,-1	0,0,1,-6,18,122,-9,2	0,0,1,-5,15,123,-8,2
-1,2,-4,11,126,-7,2,-1	0,0,1,-4,13,124,-7,1	0,0,1,-4,11,125,-6,1
0,1,-2,6,127,-5,1,0	0,0,1,-3,8,126,-5,1	0,0,1,-2,6,126,-4,1
0,0,0,2,127,-1,0,0	0,0,0,-1,4,127,-3,1	0,0,0,0,2,127,-1,0

localiza-se em especial nos movimentos rígidos em todo o quadro, ou seja, nos movimentos de câmera (HAN et al., 2021).

A compensação de movimento da GWMC é análoga à compensação de movimento da LWMC e ambos os algoritmos utilizam o mesmo código. A principal diferença entre eles é a fonte dos parâmetros da matriz afim e o método de estimá-los. A LWMC utiliza MVs de blocos vizinhos, porém a GWMC realiza um processo de busca para localizar pontos de interesse (*features*) que possam ser utilizados para estimar esses parâmetros. Além disso, a LWMC depende apenas de informações de MVs de blocos vizinhos que compartilham o mesmo quadro de referência, enquanto a GWMC utiliza diferentes quadros de referência (Parker et al., 2017a).

A LWMC é empregada em conjunto com o modelo de blocos afins. Ao contrário da LWMC, a GWMC aceita os movimentos translacionais (que posteriormente são convertidos em MVs) e os parâmetros devem ser explicitamente transmitidos no *bitstream*. Apesar disso, a implementação padrão da GWMC não considera os cálculos dos movimentos translacionais, como também não estima movimentos afins, apenas movimentos de escala e rotação (Parker et al., 2017a).

De modo geral, as principais etapas na GWMC envolvem a identificação de ambas as imagens, a correspondência dos recursos e a estimativa dos parâmetros de movimento global com base nos recursos correspondentes. Seu modelo geral de movimento é dado por (16):

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{16}$$

onde $\begin{bmatrix} x \\ y \end{bmatrix}$ e $\begin{bmatrix} x' \\ y' \end{bmatrix}$ representam as coordenadas de pixel no quadro atual e de referência, respectivamente.

Para executar a estimativa de parâmetros da matriz afim, o GWMC executa uma série de etapas comuns em visão computacional chamadas de estimativa de homografia baseada em recursos. Essa estimativa é realizada através de algumas etapas: (i) detecção e (ii) descrição de caraterísticas; (iii) casamento em pares das características; e (iv) estimação dos parâmetros, onde as características identificadas são usadas para estimar os parâmetros do modelo de movimento (BéGAINT et al., 2018). A Figura 11 apresenta o fluxograma da para a GWMC.

As características, ou também chamadas de pontos de interesses, são referidos a pedaços de informações que representam uma caraterística em uma imagem, ou seja, são píxeis que possuem uma determinada posição e podem ser diretamente encontrados, ou que podem ser definidos também como píxeis que são repetidos, de forma consistente, entre diferentes imagens (TRUONG; NHAT; KIM, 2016). Além disso, po-

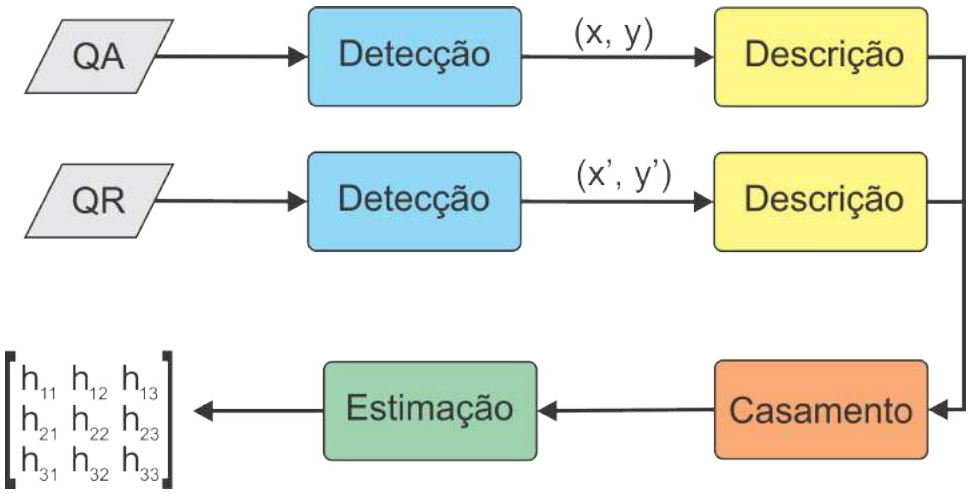

Figura 11 – Fluxograma da Compensação de Movimento Distorcido Global.

dem possuir outras características, tais como informações relevantes: rotação, escala, peso etc.

Essas características são derivadas por meio de algoritmos que escaneiam uma determinada imagem e selecionam os pontos mais significativos, com base em um critério específico. Normalmente, esses pontos têm coordenadas (x, y) e correspondem a um pixel dentro da imagem, semelhante às características que o GWMC do AV1 utiliza. Entretanto, em alguns casos, as características podem ser representadas por múltiplos píxeis, cada um representando uma aresta ou silhueta de um objeto completo, as delimitações são indicadas por essas propriedades.

A fim de identificar as características, a primeira etapa do GWMV usa o algoritmo *Features from Accelerated Segment Test* (FAST) (ROSTEN; DRUMMOND, 2006) O algoritmo FAST é um algoritmo de detecção de pontos de interesse em imagens, examinando, principalmente, pontos de canto. Estes pontos são regiões da imagem que permanecem estáveis sob possíveis alterações e, assim, podem ser rastreados de maneira eficiente entre os quadros (Parker et al., 2017a).

O algoritmo FAST realiza a detecção dos pontos de canto, examinando um círculo de 16 píxeis ao redor do pixel de interesse (p). Se 12 dos 16 píxeis forem contíguos, p é considerado um recurso (quina) na imagem (ROSTEN; DRUMMOND, 2006).

Uma vez conhecidas as características do quadro de origem e do quadro de referência, o processo de descrição de características é realizado calculando a função de correlação cruzada normalizada, entre os dois conjuntos de recursos. Um recurso (por exemplo, quina) é escolhido se for selecionado.

A correlação cruzada ou correlação entre dois sinais é o método que o GWMV usa para comparar se dois sinais são parecidos. Esta correlação pega a amostra $A = axa$ pertencente a imagem atual e a amostra $B = bxb$ da imagem de referência e seleciona $b > a$. Posteriormente, a amostra A é disposta na posição $(1, 1)$ de B, e a

normalização da correlação é aplicada obtendo um valor entre $[-1, 1]$, então a amostra A é deslocada para a posição $(1, 2)$ e novamente tem sua correlação calculada. Esse processo se repete até que a posição de (axa) esteja na posição de (bxb) (ZHAO; HUANG; GAO, 2006).

Com a detecção e descrição das características concluídas, o próximo passo é o casamento aos pares. De modo geral, o casamento entre duas imagens é representado por pares de coordenadas de uma imagem (x, y), sendo $(x, y) \leftrightarrow (x', y')$ (SZELISKI, 2010).

Para este casamento de pares a GWMC do AV1 usa do método denominado de força bruta, o qual consiste na comparação de todas as características entre par de imagens (SZELISKI, 2010). Porém, este método é custoso, por isso o GMWC do AV1 utiliza uma abordagem de força bruta, com um limite no raio de busca. Sendo assim, apenas os pontos (quina) que estejam dentro de um determinado raio são considerados na busca. Dessa forma, o algoritmo de força bruta avalia exaustivamente cada recurso (quina) em cada par de vetor descrito de uma imagem em comparação com cada recurso em outra imagem. Contudo, a busca pode ser restrita a raios de buscas que estejam a uma distância específica umas das outras, isso evita que pontos distantes sejam considerados e aumenta a precisão do algoritmo.

Com todas as demais etapas já concluídas, é possível realizar a estimação da matriz utilizando os pares de características casadas. O GWMV do AV1 implementa o método de mínimos quadrados para resolver o problema de estimativa. Deste modo, pelo menos quatro pontos devem ser identificados e a questão pode ser enunciada como: sejam $(xi, yi) \leftrightarrow (x'i, y'i)$ correspondências com $i = 1...4$ e dada a matriz (16), e com a equação (17):

$$x'_i = \frac{h_{11}x_i + h_{12}y_i + h_{13}}{h_{31}x_i + h_{32}y_i + h_{33}}, y'_i = \frac{h_{21}x_i + h_{22}y_i + h_{23}}{h_{31}x_i + h_{32}y_i + h_{33}}, \tag{17}$$

pode-se construir a matriz (18) no formato $AH = 0$:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x'_4x_4 & -x'_4y_4 & -x'_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -y'_4x_4 & -y'_4y_4 & -y'_4 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \tag{18}$$

o que define o problema de mínimos quadrados $||AH - 0||^2$. Assim, resolvendo o

problema para cada um dos quatro pontos selecionados, pode-se calcular o erro associado ao modelo e identificar quais pontos são *inliers* e quais são *outliers* usando as transformações em (17) nos pontos correspondentes (x, y) para cada par de correspondências.

Portanto, para calcular uma estimativa de mínimos quadrados para um determinado modelo, o GWMC do AV1 usa o *Random Sample Consensus* (RANSAC) (FISCHLER; BOLLES, 1981). O RANSAC foi definido como um algoritmo de estimação de parâmetros para determinar, em um conjunto de dados, todos os pontos fora da curva (*outliers*) (FISCHLER; BOLLES, 1981).

O algoritmo tem o intuito de minimizar os impactos ruidosos com *outliers* nos dados, de modo que ele encontre os parâmetros que proporcionam a melhor correspondência para o movimento dos objetos identificados, que dependem do tipo de movimento (Parker et al., 2017a).

Sendo assim, o algoritmo RANSAC é usado para estimar os parâmetros do movimento global entre os quadros que foram selecionados pela função de correlação, e, ao final do procedimento, o melhor modelo é selecionado. Seus parâmetros são então convertidos em uma matriz afim, que pode ser utilizada pela LWMC ou MC, se desejado, no nível do bloco.

4 TRABALHOS RELACIONADOS

No universo dos codificadores de vídeo, o codec AV1 é consideravelmente novo e, por este motivo, existe uma vasta área de pesquisa ainda aberta nesse tema. Entretanto, já é possível encontrar alguns poucos trabalhos (além dos trabalhos do autor desta tese) que exploram o desenvolvimento de hardware de alto desempenho para algumas das ferramentas do AV1. Para contextualizar esse trabalho de doutorado em relação ao estado da arte, foram analisadas diversas obras na literatura, os quais com foco principal no *decoder* dos codificadores, com o objetivo de mostrar os principais desafios que permanecem sem solução pelos trabalhos relacionados atuais, em relação ao projeto de hardware para as ferramentas de predição inter quadros, foco desta tese. Na seção 4.1, são apresentados os trabalhos encontrados da literatura, com foco na predição inter quadros do codificador AV1. Na seção 4.2 são apresentados alguns trabalhos desenvolvidos para a predição inter quadros do codificador HEVC, cujas ideias foram aproveitadas para os trabalhos desenvolvidos nessa tese. Na seção 4.3 são apresentadas as propostas consideradas mais interessantes para essa tese, com soluções aproximadas para predição inter quadros de codificadores de vídeo em geral. Por fim, na seção 4.4 é realizado o fechamento do capítulo, listando e discutindo os desafios ainda em aberto.

4.1 Trabalhos com Foco na Predição Inter Quadros do AV1

Como o AV1 é um codificador novo, existe uma pequena variedade de trabalhos na literatura com projeto de hardware dedicado para a predição inter quadros do codificador AV1. Existem três trabalhos do autor desta proposta (Domanski et al., 2019), (DOMANSKI et al., 2021) e (DOMANSKI et al., 2023a), que serão discutidos nas Seções subsequentes, onde são apresentados os resultados alcançados deste trabalho, portanto, não estarão listados nessa seção.

O trabalho de Freitas et al. (2020) realiza um estudo sobre quais dos filtros apresentados nas Tabelas 1 e 2 são mais utilizados dentro da FME e, posteriormente, propõe uma arquitetura de hardware para a FME definida no codificador de vídeo

AV1. A arquitetura proposta implementa apenas a família de filtros Regular, visto que foi averiguado ser o filtro mais utilizado dentro da FME. Assim, são usados apenas os 15 filtros de interpolação desta família, deixando de lado os demais.

O artigo de Chiang et al. (2017) apresenta um esquema de filtragem de interpolação adaptativa, com uma proposta que implementa um filtro de 12 derivações em conjunto com um projeto de operação complementar, que busca minimizar o impacto no desempenho de decodificação. A versão dos filtros utilizados em (Chiang et al., 2017) não é a mesma apresentada neste texto, pois o artigo em questão utilizava de versões anteriores dos filtros suportados pelo AV1.

4.2 Trabalhos com Foco na Predição Inter Quadros do HEVC

Há uma vasta variedade de trabalhos que buscam desenvolver hardware dedicado para a predição inter quadros do codificador HEVC, no entanto, foram selecionados alguns dos trabalhos, cujas ideias tiveram um maior potencial de aproveitamento no escopo desta tese. Lembrando que o HEVC possui ferramentas de codificação diferentes e mais simples do que as do AV1.

Os trabalhos de Guo; Zhou; Goto (2012) e Penny et al. (2015) propõem um filtros reconfiguráveis para reduzir a área de implementação da compensação de movimento do HEVC, através do uso de pipeline, além de um esquema de reutilização dos filtros para proporcionar uma ampla paralelização. Ambas as arquiteturas propostas usam filtros compatíveis com o codificador HEVC e reduzem a comunicação de memória e o consumo de energia.

León; Cárdenas; Castillo (2016), Filho et al. (2020) e Seidel et al. (2018) apresentaram projetos de hardware de alto desempenho visando a estimação de movimento fracionária do HEVC. Os três trabalhos buscam um projeto de hardware altamente paralelo, visando o processamento de vídeos de altas definições, como UHD. Os três trabalhos projetam toda a arquitetura da FME, suportando todos os filtros pré-definidos pelo codificador.

Já Maich et al. (2015) e Paim et al. (2016) apresentam uma solução de hardware multi padrão na interpolação de subamostras de luminância para a compensação e estimação de movimento fracionária, reaproveitando hardware de forma que ambos utilizem os mesmos filtros. Estes trabalhos desenvolveram arquiteturas altamente paralelas, capazes de suportar os padrões MPEG-2, MPEG-4, H.264/AVC, HEVC, AVS e AVS2.

4.3 Trabalhos com Computação Aproximada com Foco na Predição Inter Quadros

A computação aproximada pode ser explorada em diferentes níveis, como simplificações baseadas em hardware ou simplificações algorítmicas. Deste modo, há uma ampla gama de soluções que exploram a computação aproximada na predição inter quadros para diferentes codificadores de vídeo. Por outro lado, nenhum dos trabalhos da literatura, exceto os do próprio autor desta proposta, exploram computação aproximada para projeto de hardware para o AV1. Essa seção apresenta alguns dos trabalhos da literatura focados em outros padrões, e que foram considerados de maior interesse para essa tese.

O trabalho de El-Harouni et al. (2017) propõem uma arquitetura aproximada para a estimação de movimento, visando uma alta eficiência energética para o codificador HEVC, através da adição de diferentes aceleradores de SADs, com modos de aproximações multiníveis. PRABAKARAN et al. (2019) Apresenta algo bastante similar, visando novamente a estimação de movimento do HEVC. Nesse trabalho, somadores aproximados foram desenvolvidos a fim de construir os SADs usados na predição inter quadros. Além disso, PORTO et al. (2020) também se concentra na etapa de predição inter quadros do HEVC, propondo uma arquitetura de estimação de movimento aproximada, empregando somadores imprecisos no cálculo do SAD. Todos esses trabalhos projetaram arquiteturas visando simplificações no SAD usado tanto na IME quanto na FME.

Os trabalhos de da Silva; Siqueira; Grellert (2019) e Penny et al. (2018) focam na unidade de interpolação subpixel da etapa FME do codificador HEVC. A computação aproximada foi aplicada, alterando o número de taps dos filtros de uma forma configurável, levando a uma implementação de hardware eficiente, com uma redução considerável na comunicação de memória necessária.

Já o trabalho de Penny et al. (2020) desenvolveu uma arquitetura de hardware aproximada, de baixo consumo de energia para o interpolador da FME do HEVC, propondo dois projetos diferentes. A computação aproximada é aplicada em dois níveis: no nível do algoritmo, alterando os coeficientes do filtro, substituindo-os por valores amigáveis ao hardware, e no nível dos dados, alterando o número de taps do filtro, levando a uma implementação de hardware de baixo consumo energético e considerável redução na comunicação da memória.

4.4 Conclusão Capítulo

Dentre os trabalhos relacionados, foi possível encontrar diversos voltados ao codificador HEVC, isso por que ele foi lançado em 2013 e, assim, muitos trabalhos já

foram desenvolvidos com foco nesse padrão. Por outro lado, poucos trabalhos foram encontrados com foco no AV1 e isso em função, principalmente, da novidade desse codificador, já que sua primeira versão foi lançada no final de 2018. Mesmo com algumas ideias podendo ser aproveitadas para o AV1, quando avaliados os trabalhos para o HEVC, é necessário destacar que existem diferenças importantes entre estes codificadores. A Tabela 4 sumariza essas diferenças.

Tabela 4 – Sumário comparativo da predição inter quadros dos codificadores HEVC e AV1

Modos inter	**Codec HEVC**	**Codec AV1**
Tamanho Máximo de Bloco	64 x 64	128 x 128
Número de Tamanhos de Bloco	24	22
IME	X	X
FME	X	X
Precisão da FME	$1/4$	$1/8$
Número de Famílias de Filtros	N/A	6
Número de Filtros na FME	3	90
Compensação de Movimento	X	X
Compensação de Movimento de Bloco Sobreposto		X
Compensação de Movimento Distorcido		X
Predição Composta Avançada		X

Conforme é possível visualizar na Tabela 4, existem três novas ferramentas na predição inter quadros do codificador AV1, quando comparado com o codificador HEVC. Estas novas ferramentas já foram discutidas nos capítulos anteriores. Lembrando também que a precisão da FME do AV1 é de $1/8$ de pixel para luminância enquanto a do HEVC é de $1/4$ de pixel, além do número de famílias de filtros FIR que o AV1 estabelece, que são seis famílias enquanto no HEVC são apenas três. No total, o AV1 suporta 90 filtros na FME, enquanto o HEVC suporta apenas três filtros.

Também é muito importante destacar os diferentes métodos de partição de bloco, já que isso traz uma diferença significativa no custo computacional da predição inter quadros. No HEVC, o tamanho máximo de bloco é de 64 x 64, enquanto no AV1 esse tamanho foi quadruplicado, para 128 x 128. Entretanto, a variedade de partições suportadas em ambos os codificadores é diferente. Com isso, o HEVC suporta 24 diferentes tamanhos de bloco, enquanto o AV1 suporta até 22 tamanhos de bloco.

Após essa breve análise dos trabalhos relacionados e uma comparação com o co-

dificador HEVC, foi possível concluir que ainda existem muitos desafios a serem resolvidos na predição inter quadros do codificador AV1. Estes desafios foram norteadores para o desenvolvimento e a continuidade deste trabalho. São eles:

- Exploração do desenvolvimento de hardware dedicado para as novas ferramentas da predição inter quadros do AV1;
- Exploração do paralelismo nas novas ferramentas de predição inter quadros do AV1 com o intuito de fornecer uma elevada taxa de processamento;
- Aplicação de técnicas de computação aproximada para buscar elevada taxa de processamento e um elevado ganho em termos de área e potência.

5 ARQUITETURA PARA O INTERPOLADOR DE SUBAMOSTRAS DA COMPENSAÇÃO DE MOVIMENTO

Este capítulo apresenta o projeto de hardware focado no interpolador, usado na Compensação de Movimento (MC) do codificador AV1. Sendo assim, ele foi divido de maneira que: a primeira parte apresenta os filtros usados no interpolador da MC, e em sequência é apresentado o interpolador desenvolvido.

Um desafio na implementação da MC no AV1 é o número de diferentes tamanhos de blocos suportados. A MC suporta um total de 22 tamanhos de blocos, variando de 128 x 128 até 4 x 4, com diferentes partições retangulares, conforme explicado na Subseção2.1.1. Para resolver este problema, cada bloco pode ser dividido em sub-blocos menores, permitindo um melhor processamento e acesso à memória (PAIM et al., 2016). Considerando a necessidade de suportar tamanhos de blocos maiores compostos por pequenos sub-blocos, a utilização do menor tamanho de bloco suportado pelo AV1 (4 x 4) parece ser uma boa alternativa. Deste modo, todas as soluções arquiteturais para a MC do AV1 foram projetadas para tamanhos de bloco 4 x 4.

Sendo assim, a primeira solução desenvolvida foi o projeto de hardware para o interpolador de subamostras da Compensação de Movimento para o codificador do AV1.

Os resultados do hardware desenvolvido para o interpolador multifiltro desenvolvido foram apresentados em um artigo originalmente submetido ao *IEEE International Symposium on Circuits and Systems* (ISCAS) 2019. No entanto, este artigo foi convidado para publicação imediata no IEEE TCAS-II (Domanski et al., 2019) devido a sua classificação entre os melhores artigos do evento, o que foi considerado um reconhecimento importante para este trabalho.

5.1 Metodologia

Esta seção apresenta a metodologia empregada para o estudo e implementação das arquiteturas aqui propostas.

Conforme já citado anteriormente, ainda não foram encontrados trabalhos na li-

teratura que abordem as ferramentas do preditor inter quadros do codificador AV1 por completo, tampouco levando em consideração o uso de computação aproximada. Portanto, este trabalho apresenta o desenvolvimento de diferentes soluções capazes de prover taxas de processamento suficientes para processar vídeos de elevada resolução (UHD 4K e 8K) em tempo real e buscando eficiência em termos de área e potência.

A metodologia de desenvolvimento passa pelo estudo detalhado dos algoritmos propostos pela predição inter quadros do AV1. A versão do software de referência do AV1, chamado AOM, utilizada para o desenvolvimento deste trabalho é a *"AOMedia Project AV1 Encoder 2.0.0-684"*. A análise do impacto na eficiência de codificação das soluções aproximadas utilizará a mesma versão do software de referência citada acima, de modo que, para estas análises, serão utilizadas sequências de vídeos full HD com 60 quadros, e sequências de vídeos UHD 4K e 8K com 60 quadros, encontrados na base de dados *IETF-NETVC-Testing* (GROUP, 2020). Os parâmetros de quantização CQ com valores 20, 32, 45 e 55 serão utilizados para os testes em cada sequência de vídeo.

Já os projetos de hardware foram descritos em VHDL e verificados em nível de RTL com a ferramenta *ModelSim - INTEL FPGA STARTER EDITION 10.5b* através de *test benches*. A síntese ASIC será realizada através do uso da biblioteca TSMC 40nm com alimentação de 1.1V, fazendo o uso da ferramenta *Cadence RTL Compiler*. Os resultados de potência serão gerados a partir de entradas aleatórias da ferramenta, com atividade de chaveamento dos dados fixada em 20%. A contagem de *gates* foi calculada baseada em uma porta *Nand* de duas entradas desta tecnologia (0,9408μm^2).

Para cada arquitetura desenvolvida, foi calculado a frequência de operação mínima necessária, a qual foi determinada pela Equação 19.

$$\frac{W.H.FPS}{T.F} = 1 \tag{19}$$

Na Equação 19, W e H representam a largura e altura de um quadro do vídeo na resolução alvo, respectivamente, já FPS é a taxa de quadros por segundo, e por fim T e F representam o número de amostras entregues por ciclo e a frequência alvo.

5.2 Filtro Usado na Interpolação da MC do AV1

Uma arquitetura de interpolação multifiltro (MFIA) foi desenvolvida e está ilustrada na Figura 12. Essa arquitetura visa suportar todos os 96 filtros definidos na MC do AV1, que foram apresentados nas Tabelas 1 e 2. São 90 filtros, com taps variando entre 2 até 8 e com $1/16$ pixel de precisão, além dos seis filtros de identidade. Estes filtros são aplicados sobre blocos de luminância e de crominância. O MFIA também foi projetado pensando em diferentes metas de desempenho, uma vez que o uso de

múltiplas instâncias do MFIA permite diferentes metas de taxa de processamento, conforme será discutido na próxima seção.

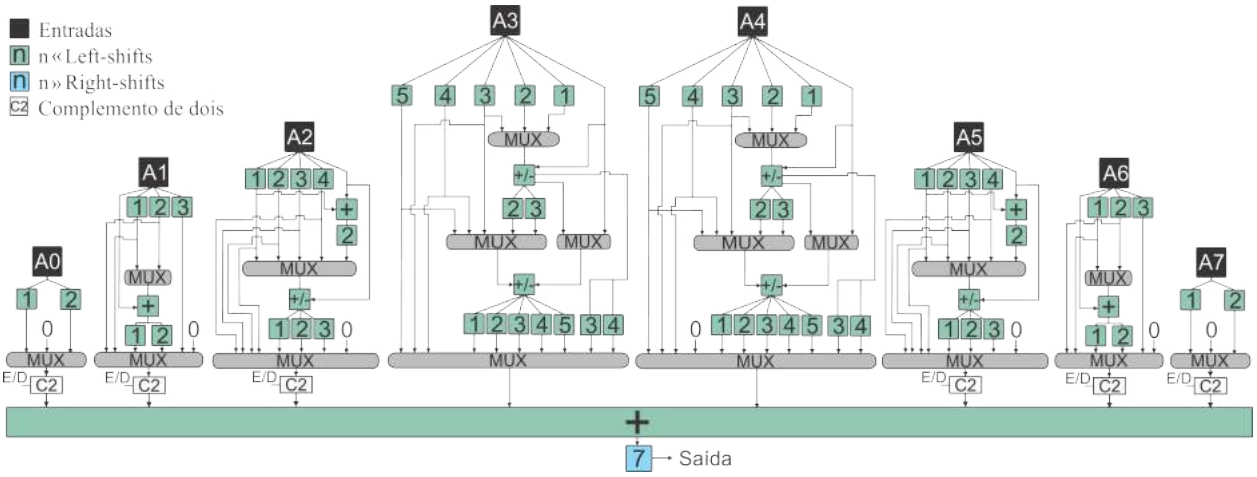

Figura 12 – Arquitetura de interpolação multi-filtro (MFIA) para a Compensação de Movimento do AV1.

O hardware implementado utiliza apenas somadores/subtratores e deslocamentos, ou seja, explora primeiramente a conversão de todos os coeficientes multiplicativos, definidos pela MC do AV1, em somas/subtrações e deslocamentos, explorando uma abordagem de *Multiple Constant Multiplication* (MCM) (VORONENKO; PüSCHEL, 2007). Deste modo, o hardware sofreu simplificações, que, consequentemente, trouxeram um ganho de desempenho. Cabe destacar que a MC não tolera imprecisão, então não foram exploradas soluções de computação aproximada nessa solução.

Os filtros usados na MC do AV1 são simétricos e tal comportamento também foi explorado no projeto MFIA. Essa simetria significa que, por exemplo, o filtro na posição $1/16$ usa os mesmos coeficientes que o filtro na posição $15/16$, apenas mudando a sequência de entrada. Por esta razão, é possível compartilhar o mesmo hardware para ambos os filtros através da seleção das entradas corretas.

Na Figura 12 é possível observar, também, que as entradas (referente a matriz de entrada) estão distribuídas de $A0, A1, ...A7$. Lembrando que, para a MC interpolar qualquer tamanho de bloco, ele requer uma matriz de entrada composta pelas amostras centrais do bloco e amostras extras convenientes, e, como estamos lidando com tamanhos de blocos 4 x 4, a matriz de entrada é definida como 11 x 11 amostras.

A Figura 12 também mostra que a saída de alguns multiplexadores está conectada a uma operação de complemento de dois, isso porque a subtração pode ser necessária em alguns filtros. O último somador realiza a soma de todas as amostras e entrega o resultado de acordo com o filtro selecionado. Finalmente, um deslocamento de sete bits para a direita é executado como uma operação de divisão. Como se pode notar

na Figura 12, o MFIA foi projetado de forma puramente combinacional, permitindo o processamento de oito amostras de entrada em paralelo.

O módulo MFIA também possui um controle de entrada adicional, que é responsável por definir qual família de filtro deve ser executada, podendo assim executar o filtro correto para cada bloco. Esta informação está disponível no fluxo de bits do vídeo codificado para cada tamanho de bloco.

5.3 Interpolador Usado na MC do AV1

Esta seção tem como objetivo descrever o projeto de três arquiteturas para o interpolador usado na MC do decodificador AV1, que utilizam a solução MFIA.

As arquiteturas suportam todos os 90 tipos de filtros, mas, no lado do decodificador, apenas um filtro deve ser aplicado para cada tamanho de bloco, de acordo com a decisão do codificador, informada no fluxo de bits codificado. Além disso, todas as arquiteturas foram projetadas para processar tamanhos de bloco 4 x 4, que é o menor tamanho de bloco suportado pelo AV1. Ao usar esse tamanho de bloco, é possível oferecer suporte a todos os tamanhos de bloco AV1 disponíveis, dividindo blocos maiores em blocos 4 x 4. A diferença entre os três designs é o nível de paralelismo suportado em cada um deles. A primeira solução é chamada de 1RC (uma linha por ciclo - *one rows per cycle*) e processa uma linha da matriz de entrada a cada ciclo de clock, usando oito instâncias de MFIA. A segunda solução processa duas linhas por ciclos e é denominada 2RC (duas linhas por ciclo - *two rows per cycle*), demandando de 16 instâncias de MFIA. Já a terceira, e última solução, processa quatro linhas por ciclo e é chamada 4RC (quatro linhas por ciclo - *four rows per cycle*), exigindo 32 instâncias de MFIA. Observe que esses projetos são totalmente independentes, atingindo objetivos diferentes.

As arquiteturas operam com tamanhos de bloco 4 x 4, conforme já mencionado, exigindo, assim, uma matriz de entrada de 11 x 11 amostras, no pior caso. Cada instância MFIA_H recebe oito amostras de entrada e gera uma amostra de saída. A amostra de saída de cada instância MFIA_H é conectada a uma cadeia de registradores de deslocamento. A cadeia de registradores de deslocamento é responsável por alimentar o valor/posição correta para cada instância de MFIA_V. Sendo assim, as saídas MFIA_V são as saídas do interpolador da MC. O que muda em relação a cada arquitetura é o número de instâncias de MFIA_H e MFIA_V, como também a organização e número de registradores necessários para conectá-los.

O 1RC lê uma linha por ciclo (onze amostras) da matriz de entrada. A Figura 13 ilustra a arquitetura do primeiro projeto do interpolador, denominado 1RC, que é composto por quatro filtros horizontais MFIA (MFIA_H na Figura 13), quatro filtros MFIA verticais (MFIA_V na Figura 13) e quatro cadeias de registradores de deslocamento

(*Shift Register Chains* - SRC) para conectar os filtros, as quais são compostas por oito registradores cada. Os quadrados pretos na Figura 13 representam as amostras de entrada inteira (luminância ou crominância). Os filtros usados na MFIA$_H$ e MFIA$_V$ são idênticos, usando a arquitetura apresentada na Figura 12. Os diferentes nomes são usados apenas para destacar quais instâncias são utilizadas nos processos de filtragem vertical e horizontal.

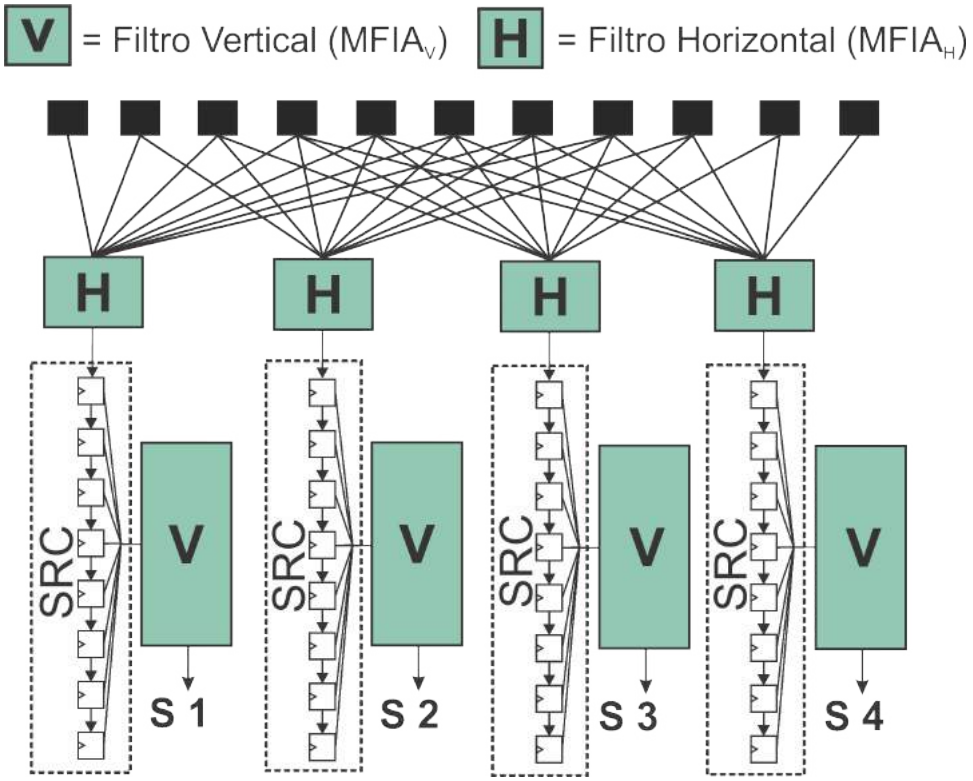

Figura 13 – Arquitetura do interpolador usado na MC do AV1 denominado 1RC.

A arquitetura 1RC tem uma latência de sete ciclos e, então, entre os ciclos de oito a onze, entrega quatro amostras válidas por ciclo. Dessa forma, onze ciclos são necessários para entregar a interpolação de um bloco de entrada 4 x 4. Esta arquitetura é capaz de processar vídeos UHD 4K a 30 fps, como ficará mais claro na próxima seção.

A Figura 14 mostra a segunda arquitetura de nível superior do interpolador desenvolvido, denominado 2RC. Essa arquitetura, por sua vez, é capaz de processar duas linhas da matriz de entrada por ciclo. O 2RC foi projetado com oito instâncias de MFIA$_H$, oito instâncias de MFIA$_V$ e dois roteadores, compostos por cadeias de registradores de deslocamento (*Shift-Register Chain Router* (SRCR) na Figura 14). Como a arquitetura anterior, a 2RC foi projetada para processar blocos com tamanho de 4 x 4, e, assim, recebe 22 amostras por ciclo da matriz de entrada, que é composta por 11 x 11 amostras. As 22 amostras de entrada estão representadas na Figura 14 como quadrados em escala de cinza.

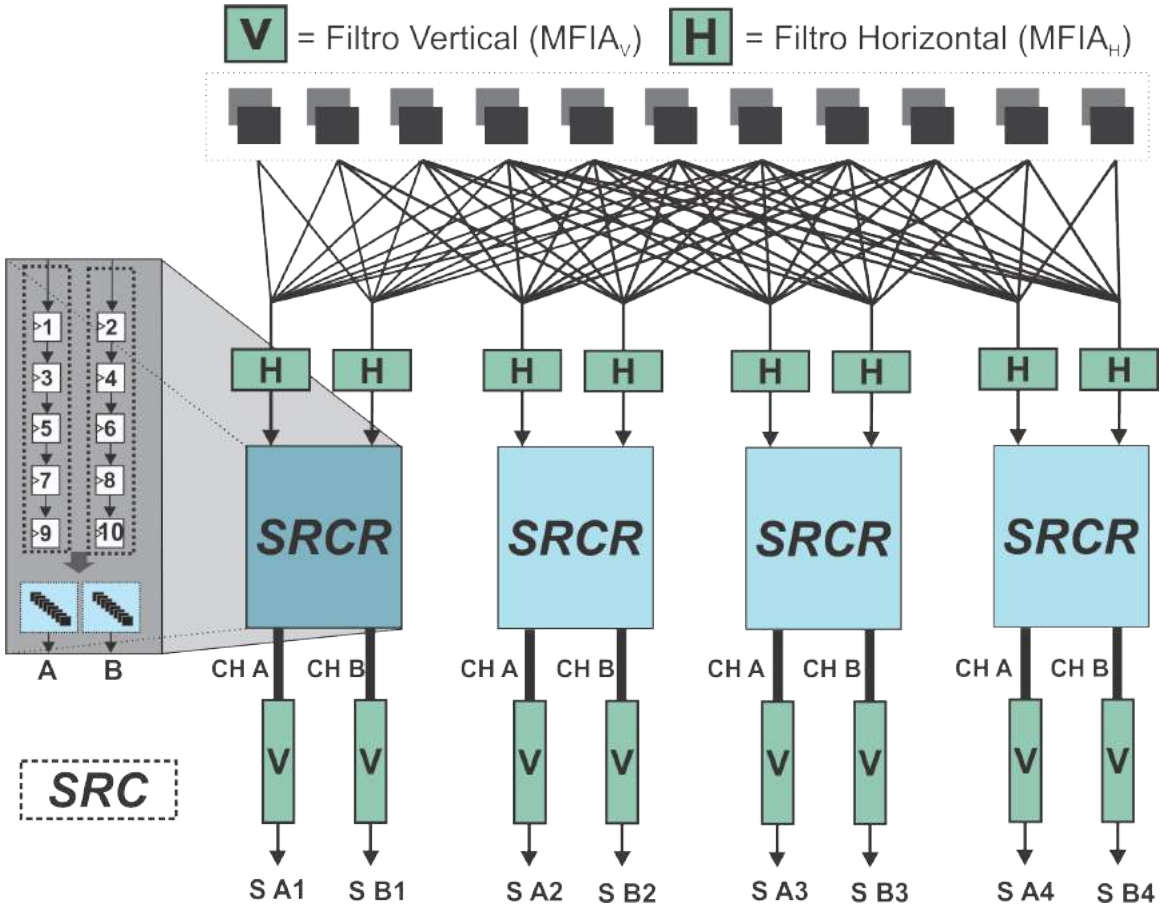

Figura 14 – Arquitetura do interpolador usado na MC do AV1 denominado 2RC.

Os SRCR's apresentados na Figura 14 são responsáveis por armazenarem todas as amostras que vêm das instâncias MFIA$_H$, assim, encaminhando essas amostras para as instâncias MFIA$_V$. Para realizar esse processamento, cada SRCR possui duas cadeias de registradores de deslocamento (SRC) com cinco registradores cada, e cada SRC é responsável por armazenar a amostra de saída correspondente da instância MFIA$_H$. Sendo assim, a primeira instância MFIA$_H$ é conectada como entrada do registrador um e a segunda instância MFIA$_H$ é conectada com a entrada do registrador dois. Da mesma forma, a primeira instância MFIA$_H$ é responsável por processar a primeira linha da matriz de entrada, e a segunda instância MFIA$_H$ processa a segunda linha da matriz de entrada. Desde modo, o SRCR é responsável por rotear essas amostras para as instâncias MFIA$_V$. A entrada da primeira instância MFIA$_V$ é conectada aos registadores 10, 9, 8, 7, 6, 5, 4 e 3. A segunda entrada da instância MFIA$_V$ é conectada aos registradores 9, 8, 7, 6, 5, 4, 3 e 2. Nota-se que o registrador um não é usado no roteamento para instâncias MFIA$_V$, porém, tal registrador é necessário para acertar o nível de paralelismo entre MFIA$_H$ e MFIA$_V$.

A arquitetura 2RC, por ter um nível de paralelismo maior que a 1RC, tem uma latência de quatro ciclos, e, então, entre os ciclos cinco e seis, entrega oito amostras

válidas por ciclo. Dessa forma, seis ciclos são necessários para entregar a interpolação de um bloco de entrada 4 x 4. Esta arquitetura é capaz de processar vídeos UHD 4K a 60 fps, conforme ficará mais claro na próxima seção.

Por fim, a Figura 15 apresenta a terceira arquitetura do interpolador desenvolvido, denominado 4RC, que é projetado para ter um nível de paralelismo maior que a 1RC e 2RC, sendo capaz de processar quatro linhas da matriz de entrada por ciclo. O 4RC foi projetado com 16 instâncias de MFIA$_H$, 16 instâncias de MFIA$_V$ e quatro cadeias de roteadores (SRCR na figura 15). Essa arquitetura, também foi projetada para processar tamanhos de blocos 4 x 4 e, sendo assim, recebe 44 amostras por ciclo da matriz de entrada, que é composta por 11 x 11 amostras. As 44 amostras de entrada são apresentadas como quadrados em escala de cinza na Figura 15.

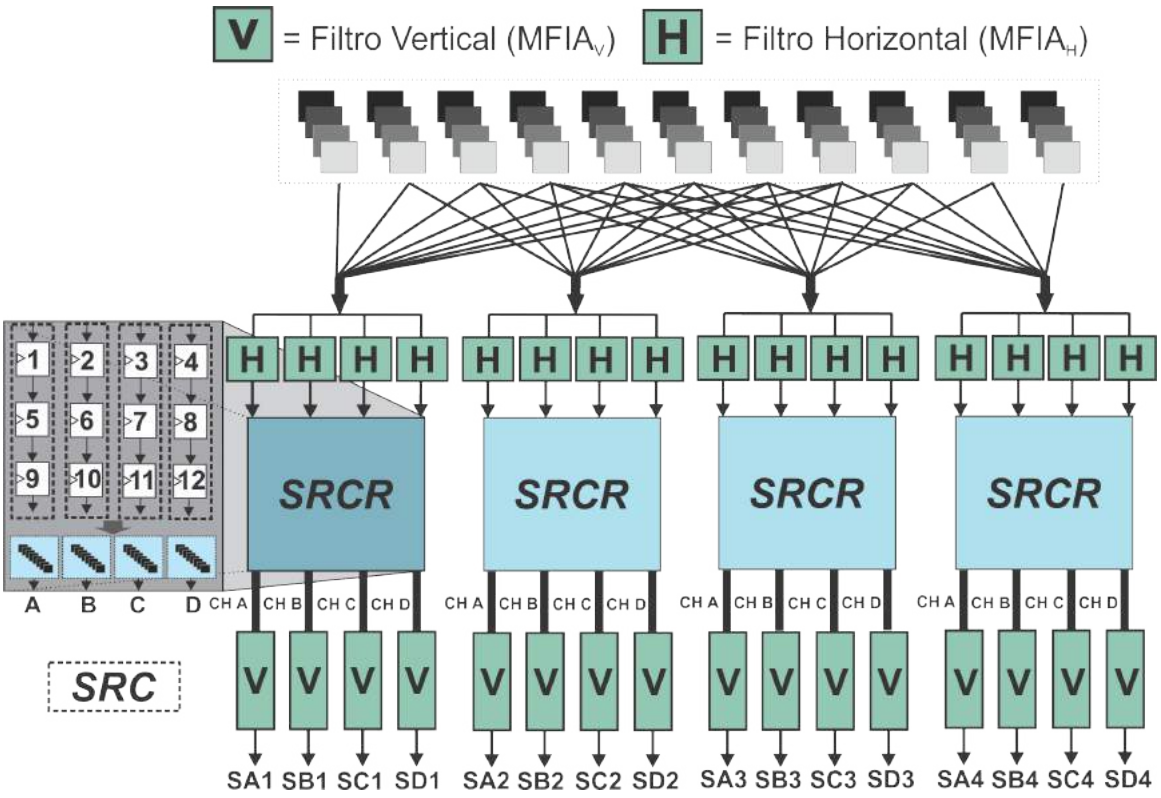

Figura 15 – Arquitetura do interpolador usado na MC do AV1 denominado 4RC.

Cada SRCR apresentado na Figura 15 é responsável por armazenar as amostras, que vêm de quatro instâncias MFIA$_H$, encaminhando-as para as instâncias MFIA$_V$. Para realizar esse processamento, o SRCR possui quatro cadeias de registradores de deslocamento (SRC) com três registradores cada, e cada SRC é responsável por armazenar a amostra de saída correspondente da instância MFIA$_H$. A primeira instância MFIA$_H$ é conectada como entrada do registrador um, a segunda instância MFIA$_H$ é conectada como entrada do registrador dois e assim por diante. Da mesma forma, a primeira instância MFIA$_H$ é responsável por processar a primeira linha de amostras de entrada, a segunda instância MFIA$_H$ processa a segunda linha de amostras

de entrada, até finalizar o processamento das quatro linhas. Com todas as amostras armazena- das no SRCR, é possível rotear essas amostras para as instâncias MFIA$_V$. A entrada da primeira instância MFIA$_V$ é conectada aos registradores 12, 11, 10, 9, 8, 7, 6 e 5. A entrada da segunda instância MFIA$_V$ é conectada aos registradores 11, 10, 9, 8, 7, 6, e 4. A terceira instância MFIA$_V$ está conectada aos registradores 10, 9, 8, 7, 6, 5, 4 e 3. Finalmente, a quarta MFIA$_V$ está conectado aos registradores 9, 8, 7, 6, 5, 4 e 2. Pode-se notar que o registrador um novamente não é usado no roteamento para instâncias MFIA$_V$ e tal registrador adicional é necessário para ajustar o nível de paralelismo entre MFIA$_H$ e MFIA$_V$.

As saídas do 4RC são compostas pelas saídas entregues por todas as instâncias MFIA$_V$. Essa arquitetura tem uma latência de dois ciclos e, no terceiro ciclo, fornece as 16 saídas válidas. Em comparação com a primeira e a segunda solução, o 4RC precisa de apenas três ciclos para processar um bloco 4 x 4. Esta arquitetura foi capaz de processar vídeos UHD 8K a 30 fps e a próxima seção apresenta os resultados de síntese que suportam essa conclusão.

5.4 Análise e Resultados

Os resultados da síntese são mostrados na Tabela 5. Os resultados do MFIA mostraram um uso de área de 4,96 Kgates e uma dissipação de energia de 2,45 mW quando operando a 279 MHz, que é a frequência alvo para as diferentes versões dos interpoladores completos.

Tabela 5 – Resultados de sínteses para as arquiteturas usadas na MC do AV1.

Arquitetura	Freq. (MHz)	Gates (K)	Power (mW)	Throughput
MFIA	279	*4,96*	*2,45*	-
1RC		40,70	16,12	2160@30fps
2RC		67,70	28,77	2160@60fps
4RC		141,10	81,31	4320@30fps

O interpolador 1RC foi projetado para processar vídeos UHD 4K (3840 x 2160 píxeis) a 30 fps. Conforme explicado anteriormente, tal arquitetura necessita de onze ciclos para gerar um bloco 4 x 4. Por outro lado, o 2RC precisa de seis ciclos para gerar um bloco 4 x 4 e foi projetado para processar vídeos 4K UHD a 60 fps, e, por fim, a arquitetura 4RC necessita de apenas três ciclos para gerar um bloco 4 x 4 e foi projetado para processar vídeos UHD 8K (7680 x 4320 píxeis) a 30 fps. Todas as

arquiteturas operaram em uma frequência de 279,9 MHz para atingir suas metas de processamento.

Ao comparar o 4RC e o 1RC, nota-se que o 4RC ocupa quase 3,5 vezes mais área e dissipa cinco vezes mais potência que o 1RC. No entanto, o 4RC pode processar vídeos 8K UHD a 30 fps, o que significa que o 4RC pode processar quatro vezes mais amostras do que o 1RC com uma taxa de transferência maior. Já ao comparar o 2RC e o 1RC, é possível perceber que o 2RC dissipa 1,78 vezes mais potência e ocupa em torno de 1,66 vezes mais área. Porém, o 2RC consegue processar duas vezes mais amostras que o 1RC, já que ele processa vídeos 4K UHD a 60 fps.

As arquiteturas apresentadas neste capítulo foram as primeiras desenvolvidas para a Compensação de Movimento do AV1, pois, até o momento, nenhum outro trabalho foi encontrado na literatura. Embora existam outros trabalhos na literatura sobre implementações em hardware sobre a MC para outros padrões, uma comparação não seria justa, visto que, os filtros empregados na MC do AV1 são bastante diferentes em termos de algoritmos e complexidade. Portanto, não existem referências que possam ser comparáveis com as arquiteturas apresentadas neste capítulo.

6 ARQUITETURAS DESENVOLVIDAS PARA A ESTIMAÇÃO DE MOVIMENTO FRACIONÁRIA DO AV1

Este capítulo aborda o projeto de hardware de alto desempenho, visando o interpolador da estimação de movimento fracionária (FME), do codificador AV1. Essa solução contou com um amplo caminho exploratório, que teve como principal foco a redução da complexidade da FME do AV1. Portanto, este capítulo será dividido em três etapas, sendo que, a primeira visa os filtros usados no interpolador da FME, a segunda o interpolador propriamente dito, e a terceira, e última, com foco na arquitetura completa da FME. As arquiteturas propostas da FME seguiram a mesma linha de raciocínio da MC em relação ao tamanho de bloco 4 x 4.

Para interpolar qualquer tamanho de bloco, a FME do AV1 requer uma matriz de entrada, composta pelas amostras centrais do bloco e amostras extras convenientes da borda. Portanto, para lidar com tamanho de bloco 4 x 4 é usada uma matriz de entrada de 11 x 11 amostras, o que totaliza 121 amostras, no pior caso. A metodologia empregada para o desenvolvimento dessas arquiteturas é a mesma apresentada na Seção 5.1.

Os resultados do hardware desenvolvido para FME do AV1 geraram dois artigos publicados, sendo ele um no *IEEE International Symposium on Circuits and Systems* (ISCAS) 2021 (DOMANSKI et al., 2021) e o outro no *Latin American Symposium on Circuits Systems* (LASCAS) 2023 (DOMANSKI et al., 2023a).

6.1 Arquitetura dos Filtros

No AV1 as amostras fracionarias são calculadas utilizando filtros *Finite Impulse Response* (FIR), os quais variam de 2 até 8 taps e possuem $1/8$ de pixel de precisão. Os 48 filtros definidos na FME do AV1 foram apresentados nas Tabelas 1 e 2, sendo seis deles filtros de identidade. A FME é aplicada apenas sobre os blocos de luminância. Ambos os filtros foram projetados explorando uma abordagem de *Multiple Constant Multiplication* (MCM) (VORONENKO; PüSCHEL, 2007) e compartilhamento de sub expressão entre todos os filtros suportados, com o intuito de reduzir o uso de

hardware e o consumo de energia, além de aumentar a taxa de processamento da arquitetura.

Inicialmente, foi realizado um experimento para descobrir quais dos filtros são mais usados na FME do AV1, e seus resultados são apresentados na Figura 16. O experimento consiste em extrair, do *bitstream*, os filtros usados para reconstruir todos os blocos, de todas as sequências (apresentados nas Tabelas 1 e 2), do lado do codificador. Em seguida, executar um *script* para contar quantas vezes as famílias de filtros foram usadas. Os resultados foram obtidos usando 11 sequências de vídeo, dos conjuntos de dados *Mozilla e Netflix*, com diferentes valores de CQ 20, 32, 43, 55, e para diferentes resoluções, conforme demonstra a Figura 16. Este experimento agrupou duas famílias de filtros *Smoth* e duas famílias de filtros *Regular*, pois ambas famílias possuem 6 e 4 taps, sendo assim, os resultados dessas famílias estão agrupados.

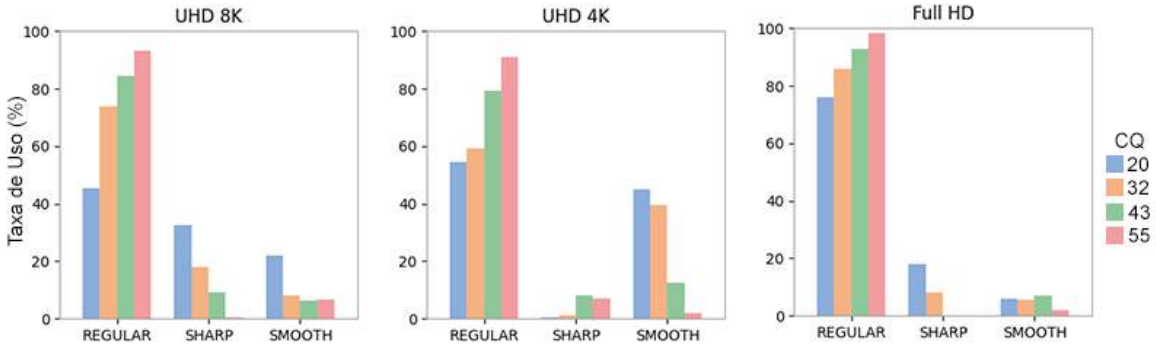

Figura 16 – Percentuais de vezes que cada filtro é usado na FME do AV1.

A utilização total de filtros comuns, considerando a média das filtragens horizontal e vertical, e todas as verificações de qualidade, mostrou que a família *Regular* foi utilizada em 75,5% das vezes. Com relação à resolução, o uso do filtro *Regular* diminui com o aumento da resolução. A segunda família de filtros mais usada é a *Smooth* e a terceira é a família *Sharp*, em termos de CQ médio e resultados de resolução. Os filtros bilineares não foram avaliados.

Esse estudo auxiliou nos próximos passos do trabalho, onde foram realizadas propostas para a simplificação dos filtros.

A primeira versão do filtro implementada foi chamada de *Multifilter OM* (MF-OM), sendo a sigla OM derivada de *Original Multiplier*, ou seja, ela representa os filtros originais usados na FME do AV1. A Figura 17 demonstra a arquitetura projetada. A Tabela 6 (Solução OM), mostra o filtro para cada família com relação à precisão $4/8$. O hardware implementado utiliza apenas somadores e deslocadores, em vez de utilizar multiplicadores, conforme a implementação padrão.

As demais versões dos filtros foram projetadas com o intuito de reduzir a complexidade da FME por dois métodos principais de computação aproximada: (i) simplificar

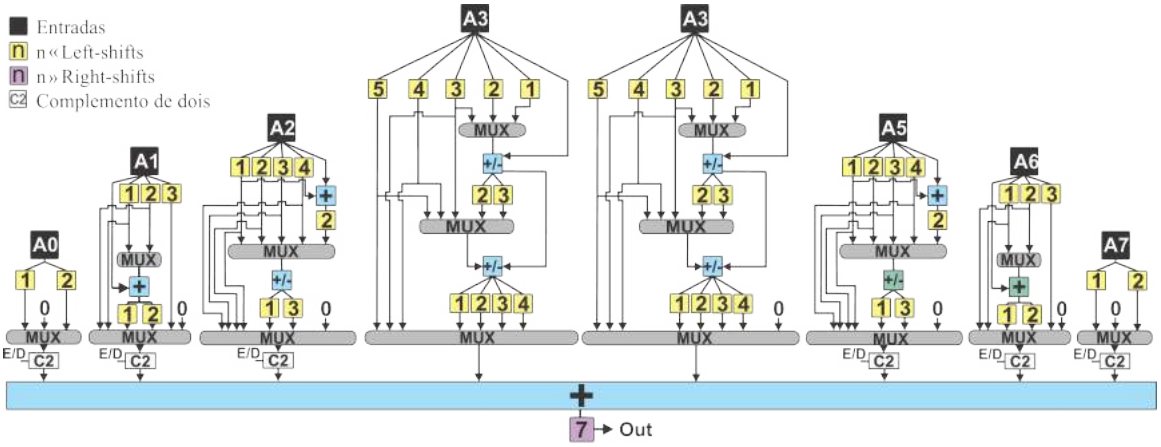

Figura 17 – Arquitetura multi-filtro original (OM) usada na FME do AV1.

os coeficientes do filtro, e (ii) reduzir o número de taps. Naturalmente, as implementações propostas consideraram os sete filtros para cada família, totalizando 42 filtros, além disso, suportaram os seis filtros de identidade. O processo de filtragem FIR é composto pela multiplicação das amostras de entrada por uma constante. A aproximação dos coeficientes pode facilitar uma simples conversão das multiplicações em somas de deslocamentos e, conforme o grau de aproximação, também será possível reduzir o número de somas necessárias. Reduzir o número de taps reduz o número de multiplicações necessárias e também, reduz a largura de banda de entrada necessária. As abordagens propostas serão discutidas a seguir.

O primeiro método buscou investigar a aproximação nos coeficientes dos filtros, e foi realizada com o intuito de reduzir ao máximo o número de cálculos necessários para gerar as amostras filtradas.

A primeira e mais agressiva aproximação foi feita transformando os coeficientes em valores de potência de dois (2^n), onde n variou de 0 a 7. Neste caso, as multiplicações foram retiradas e apenas um deslocamento é executado em cada amostra de entrada. Números negativos também são considerados, pois a subtração controlada entre taps também está disponível. No entanto, como esperado, esta solução resultou em uma perda muito elevada de eficiência de codificação, com perdas médias de até 4,8% em BD-BR (BJONTEGAARD, 2008) para resoluções Full HD e UHD 4K. Sendo assim, uma solução alternativa também foi investigada.

Esta segunda solução considerou o uso de duas adições ou subtrações, em padrões deslocados para representar os coeficientes. No entanto, esta solução não foi utilizada em todos os coeficientes, sendo que, apenas os dois coeficientes centrais são definidos por essas duas adições ou subtrações, com os demais coeficientes definidos por potência de dois, como na primeira solução. Os coeficientes centrais são os mais

importantes para os resultados, pois esses coeficientes tendem a ter a maior amplitude e multiplicam as amostras mais importantes (aquelas mais próximas da amostra filtrada).

Neste ponto, um aspecto importante deve ser destacado. A soma dos valores de todos os coeficientes em cada filtro deve ser igual a 128. Isso é necessário porque após as etapas de multiplicação e acumulação é realizada uma divisão por 128, mantendo o ganho de interpolação igual a um. Desta forma, todos os filtros aproximados respeitam esta restrição.

Um exemplo das aproximações usadas nos cálculos da amostra central é feito para o coeficiente 76, onde a seguinte operação é realizada: $((((p << 2)+p) << 2)+p) << 2$. Nesse caso, obtém-se um resultado impreciso, porque o resultado é $p \times 84$ em vez de $p \times 76$. Algumas operações são menos complexas e requerem apenas um deslocamento, como a multiplicação de p por 2, que pode ser calculada exatamente como $p << 1$.

Os valores definidos para os coeficientes do filtro (apresentados anteriormente) foram escolhidos manualmente, com a intenção de manter um equilíbrio entre as perdas causadas pela aproximação e os custos computacionais. Esta solução foi chamada de *Approximate Multiplierless with 8-taps* (AM8). A Tabela 6 (Solução AM8) apresenta os filtros para cada família com relação à precisão $4/8$.

O segundo método explorado buscou reduzir o número de taps para os filtros da FME. Esta aproximação, foi realizada usando como base a solução AM8. O número de taps no filtro define o número de multiplicações necessárias e o número de entradas necessárias. Portanto, os filtros com o maior número de taps exigirão uma grande largura de banda de E/S para a FME completa.

Em geral, para filtrar um bloco de amostras $w \times h$, um filtro com n taps exigirá amostras $(w + n - 1) \times (h + n - 1)$ para calcular a interpolação. A figura 18 mostra um exemplo das amostras necessárias para processar uma linha de um bloco 4 x 4 quando um filtro de 8 taps é aplicado. Apenas oito amostras são necessárias para gerar cada amostra interpolada, mas como há quatro amostras em cada linha para interpolar, serão necessárias 11 amostras. Uma matriz de entrada 11 x 11 (ou 121 amostras) ($n = 8$ e $w = h = 4$) é necessária para processar todo o bloco 4 x 4 nas direções horizontal e vertical. Se o o número de taps deste filtro for reduzido para 6 ($n = 6$ e $w = h = 4$), então a matriz de referência é reduzida para 9 x 9 (ou 81 amostras), o que significa uma redução na largura de banda de entrada e saída em mais de 33%. Neste caso, a multiplicação necessária também é reduzida de 128 para 96, considerando este bloco 4 x 4.

Sendo assim, neste trabalho foram avaliadas duas versões, que buscam reduzir o número de taps para os filtros da FME. A primeira denominada *Approximate Multiplierless with 6-taps* (AM6), referente ao número máximo de 6 taps e a segunda *Approxi-*

Tabela 6 – Coeficientes originais e modificados dos filtros do AV1.

Solução	Tipo do Filtro	Tap's	Coeficientes
OM	Sharp	8	-4 12 -24 80 80 -24 12 -4
	Regular	6	2 -14 76 76 -14 2
		4	-12 76 76 -12
	Smoth	6	-2 14 52 52 14 -2
		4	12 52 52 12
	Bilinear	2	64 64
AM8	Sharp	8	-2 8 -32 84 84 -16 4 -2
	Regular	6	4 -32 84 84 -16 4
		4	-32 84 84 -8
	Smoth	6	4 16 44 44 16 4
		4	32 44 44 8
	Bilinear	2	64 64
AM6	Sharp	6	4 -32 84 84 -16 4
	Regular		
		4	-32 84 84 -8
	Smoth	6	4 16 44 44 16 4
		4	32 44 44 8
	Bilinear	2	64 64
AM4	Sharp/Regular	4	-32 84 84 -8
	Smoth		32 44 44 8
	Bilinear	2	64 64

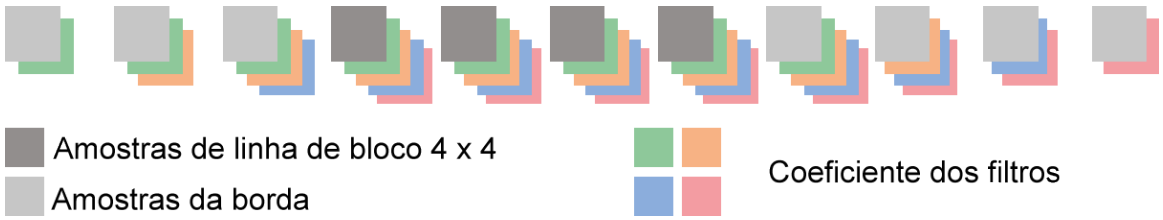

Figura 18 – Exemplo de amostras usadas na interpolação.

mate Multiplierless with 4-taps (AM4), com no máximo 4 taps, no pior caso. Lembrando que, foi utilizada a mesma estratégia da solução AM8, onde as duas amostras centrais de cada filtro foram definidas por duas adições ou subtrações e deslocamentos, e as demais foram definidas apenas por deslocamentos. Mas, para ambas as soluções (AM6 e AM4), foi necessário redefinir alguns coeficientes de filtro. Os dois coeficientes centrais permanecem os mesmos, mas, os outros coeficientes, são ajustados para que os ganhos do filtro permaneçam iguais a um.

As soluções AM6 e AM4 não sofrem alterações nas famílias de filtro *Regular* 4-tap, *Smoth* 4-tap e *Bilinear*, em relação a AM8. Além disso, a limitação de usar no máximo 6 taps (solução AM6) não afeta as famílias *Regular* 6-tap e *Smoth* 6-tap, e afeta apenas a família *Sharp* 8-tap. Por outro lado, quando o número de taps é limitado a 4 taps (solução AM4), as famílias *Smoth* 6-tap, *Regular* 6-tap e *Sharp* 8-tap sofrem alterações, pois o número de taps máximo agora é quatro. Isso pode ser observado na Tabela 6 (Solução AM8, AM6 e AM4), a qual apresenta os filtros para cada família com relação à precisão $4/8$. A Figura 19 mostra um exemplo de redução da matriz de entrada, usando apenas seis ou quatro taps em vez dos oito taps originais, exigidos pelo filtro *Sharp*.

Ainda observando a Tabela 6 nota-se que na solução AM4 a família *Smoth* 6-tap e 4-tap se tornam apenas uma, e a *Sharp* 8-tap e *Regular* 6-tap e 4-tap se unem em apenas uma família, de 4-tap. Isso ocorre, porque o processo de aproximação aplicado sobre os filtros de 6-tap e 8-tap gerará os mesmos filtros das famílias de 4-tap. Na solução AM6 as famílias *Sharp* 8-tap e *Regular* 6-tap tornam-se uma mesma família, pelo mesmo motivo.

A tabela 7 lista o número de amostras necessárias para interpolar tamanhos de blocos quadrados, usando todos os filtros disponíveis (como as versões OM ou AM8) e com filtros aproximados (AM6 e AM4), bem como a economia percentual de tais abordagens. Os tamanhos dos blocos retangulares foram omitidos, mas têm resultados proporcionais. Como pode ser visto, quanto menor o tamanho do bloco, maior o impacto percentual com a redução do número de taps, podendo chegar a uma economia de até 58.5% em relação ao número de amostras necessárias para processar um bloco.

A redução da largura de banda de entrada e saída tende a ter um impacto pro-

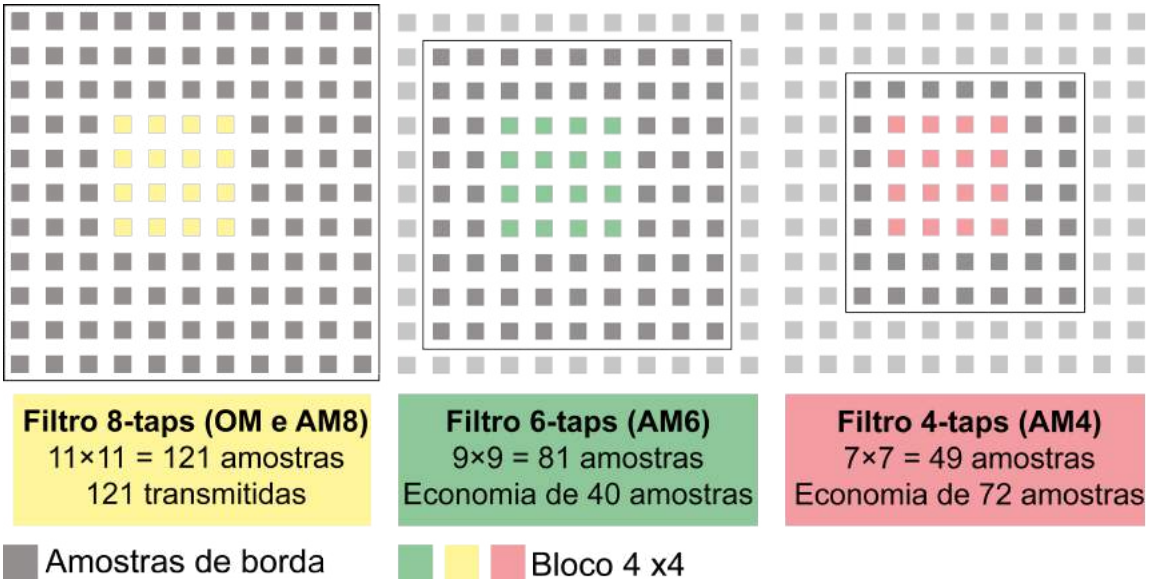

Figura 19 – Amostras necessárias para interpolar um bloco 4 × 4 usando filtros de 8, 6 e 4 taps.

porcional na largura de banda da memória do sistema, uma vez que, as amostras solicitadas são normalmente armazenadas na memória (PENNY; PALOMINO; ZATT, 2021). No entanto, não é fácil estimar os impactos finais, em termos de redução da largura de banda da memória, porque há muitas variáveis a serem consideradas, incluindo a tecnologia usada para implementar a memória, a hierarquia de memória usada e muito mais. Como este não é o foco principal deste trabalho, esta questão não será abordada com maiores detalhes.

Entretanto, é de extrema importância avaliar a perda ou ganho na eficiência de codificação nas soluções modificadas. Portanto, foram avaliados os efeitos das aproximações propostas na eficiência de codificação, em comparação com os filtros ori-

Tabela 7 – Amostras Necessárias por Tamanho de Bloco

Tamanho do	**Amostras Necessárias**			**Economia (%)**	
Bloco	**8-tap**	**6-tap**	**4-tap**	**6-tap**	**4-tap**
128×128	18,225	17,689	17,161	2,94	5,83
64×64	5,041	4,761	4,096	5,55	18,74
32×32	1,521	1,369	1,225	9,99	19,46
16×16	529	441	361	16,6	31,75
8×8	225	169	121	24,8	46,22
4×4	121	81	49	33,0	59,5

ginais. Para tanto, alguns experimentos foram realizados, utilizando o software de referência AV1 (AOMEDIA, 2020). As aproximações foram inseridas no software *libaom* e o software foi executado usando 11 sequências de vídeo, dos conjuntos de dados *Mozilla e Netflix*, seguindo o Teste de Codec de Vídeo e Medição de Qualidade (VCTQM) (GROUP, 2020). Quatro parâmetros CQ 20, 32, 43, 55 foram usados e todas as ferramentas de codificação AV1 foram ativadas. Quatro conjuntos de resultados foram então gerados, um para os filtros originais (*Original Multifilter* - OM) e um para cada versão, usando os filtros aproximados: AM8, AM6 e AM4. Um total de 176 execuções foram realizadas, considerando diferentes sequências, CQs e versões de algoritmos.

Para realizar a avaliação de eficiência e codificação foi usada a métrica Bjøntegaard Delta Bit Rate (BD-BR) (BJONTEGAARD, 2008). A Tabela 8 apresenta os resultados de eficiência de codificação para cada aproximação proposta em relação aos filtros originais, agrupando os vídeos por resolução.

Tabela 8 – Resultado Eficiência de Codificação

Sequência	**Resolução**	**BD-BR (%)**		
		AM8	**AM6**	**AM4**
Arena of Valor	Full HD	0,31	0,31	0,67
Market Place		1,41	1,41	2,46
Square and Time-lapse		1,33	1,33	2,62
Tunnel Flag		2,84	2,84	5,21
Foreman	UHD 4K	0,27	0,27	0,82
Coastguard		-0,16	-0,16	-0,14
Cactus		0,19	0,19	1,28
Bar Scene		-1,54	-1,54	-1,90
Boxing		0,88	0,88	2,06
Face	UHD 8K	0,27	0,27	0,50
Motorcycle		0,22	0,22	0,70
Média	Full HD	**1,47**	**1,47**	**2,74**
	UHD 4K	**-0,07**	**-0,07**	**0,42**
	UHD 8K	**0,24**	**0,24**	**0,60**
	Overall	**0,54**	**0,54**	**1,25**

A Tabela 8 demonstra, primeiramente, que foram obtidos resultados promissores

em relação a BD-BR, pois mesmo com a aproximação mais agressiva (AM4), os impactos médios em BD-BR são de 1,25%. Considerando as demais versões (AM6 e AM8), a perda total do BD-BR é de apenas 0,54%. Além disso, alguns resultados são negativos, indicando que, para essas sequências de vídeo específicas, as aproximações melhoraram a eficiência da codificação.

Outro ponto importante para se observar, a partir da Tabela 8, é que os resultados BD-BR para as versões AM6 e AM8 são idênticos. O motivo deste comportamento é que esta redução, de oito para seis taps, afeta apenas o filtro *Sharp*, pois todas as outras famílias de filtros possuem até seis taps, conforme Tabela 6, e como a família de filtros *Sharp* não é amplamente utilizada, como pode ser visto na Figura 16, esta pequena aproximação não afetou a eficiência da codificação.

Conforme análise da Tabela 8 observa-se que os vídeos UHD 4K tiveram um menor impacto em relação ao BD-BR, o que demostra um melhor desempenho nos resultados. Este é um ponto interessante, pois as sequências UHD 4K são as que mais utilizam filtros *Smooth*, conforme Figura 16.

Por fim, a Figura 20 demonstra um exemplo visual, extraído de um quadro de uma das sequências manipuladas, para todas as versões implementadas. Nesta imagem, o retângulo vermelho (ampliado na primeira imagem) faz uma comparação entre os resultados das três versões simplificadas (AM4, AM6 e AM8) e o original (OM). Como pode ser analisado através destas imagens, as mudanças visuais são quase imperceptíveis.

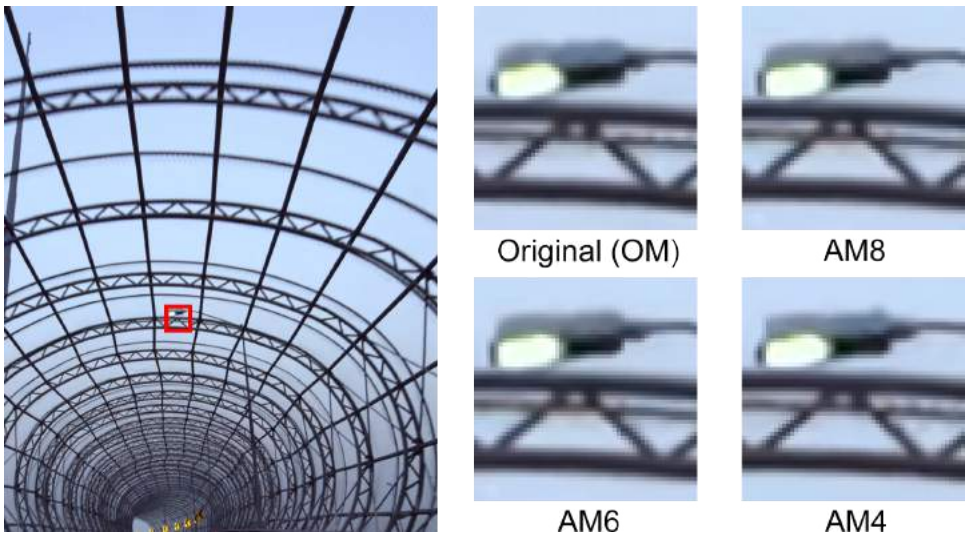

Figura 20 – Comparação de qualidade visual extraída de uma sequência manipulada nos testes.

Com a implementação das soluções aproximadas concluídas, seus projetos de hardware foram explorados. Assim, três arquiteturas foram projetadas, explorando os três níveis de aproximações discutidos anteriormente: AM4, AM6 e AM8. Conforme

discutido anteriormente, as principais ideias exploradas com essas aproximações são; a alteração dos valores dos coeficientes dos filtros para valores compatíveis com hardware sem multiplicador; e a alteração do número de taps nos filtros para reduzir as amostras de entrada necessárias.

Essas arquiteturas são projetadas para oferecer suporte a todas as 48 opções de filtragem (implementando 42 filtros e suportando os seis filtros de identidade). Elas foram denominadas *Multifilter AM8* (MF-AM8), *Multifilter AM6* (MF-AM6) e *Multifilter AM4* (MF-AM4).

A Figura 21 mostra as três arquiteturas projetadas para as soluções aproximadas. Todas as arquiteturas foram projetadas para atenderem a uma solução multifiltros. Essas três arquiteturas multifiltros são completamente combinatórias e foram implementadas sem o uso de multiplicadores. Todos os filtros são calculados usando compartilhamento de sub expressões e combinações de deslocamentos individuais, soma de deslocamentos e/ou deslocamentos parciais.

Conforme citado anteriormente, nas amostras centrais (entradas A3 e A4), das três soluções, foi incluído um cálculo adicional para aumentar a precisão dos coeficientes aplicados a essas amostras, uma vez que, são os coeficientes mais importantes no processo de interpolação. Então, nesses casos, um somador adicional e um somador/subtrator são incluídos em cada entrada e um total de quatro mudanças podem ser combinadas para gerar o valor do coeficiente final. Algumas saídas possuem um subtrator controlado para alterar o sinal de saída, de acordo com a definição do filtro, e essas operações são rotuladas como módulos C2 na Figura 21.

O acúmulo de sub-resultados é feito usando uma árvore de soma. Como o número de sub resultados se dá em função do número de taps permitidos, a árvore de soma terá diferentes profundidades. O MF-AM8 terá uma árvore de soma de três níveis, com sete somadores, o MF-AM6 também terá uma árvore de soma de três níveis, com cinco somadores e o MF-AM4 terá uma árvore de soma de dois níveis, com apenas três somadores. As árvores de adição foram omitidas da Figura 21 e são representadas por um único grande somador, para simplificar o desenho. A divisão final por 128 é realizada com um deslocamento de sete bits para a direita, conforme mostrado na Figura 21.

A Tabela 9 apresenta o hardware utilizado em cada arquitetura. Da Tabela 9 pode-se observar que o MF-AM4 usa metade dos somadores e subtratores usados pelo MF-AM8. Esta é uma redução expressiva no consumo de hardware, pois os somadores e subtratores são os elementos de maior custo de hardware dentro dessas arquiteturas.

Os resultados das sínteses das quatro arquiteturas multifiltro da FME do AV1 são mostrados na Tabela 10. A frequência usada na síntese foi a de 343 MHz, mesma frequência alvo necessária na síntese do interpolador da FME, abordado com detalhes na próxima subseção.

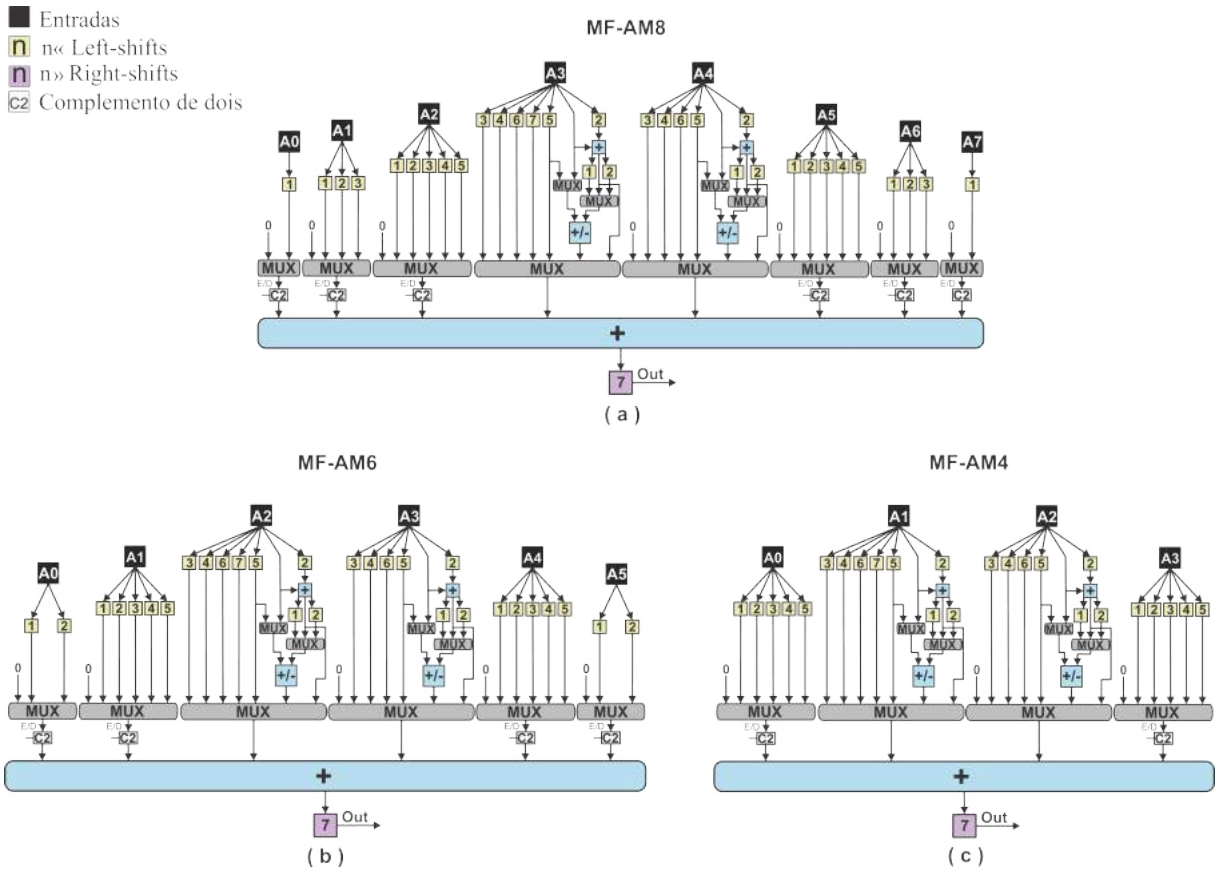

Figura 21 – Arquiteturas de interpolação multi-filtro da FME do AV1 usando a solução AM8 (MF-AM8), AM6 (MF-AM6) e AM4 (MF-AM4).

Conforme a Tabela 10, a solução MF-AM4, que é a solução mais otimizada, apresenta uma área de 1,32 Kgates, e uma potência de 0,59 mW, o que representa uma redução de 86,04% em área e 83,04% em potência, em relação a solução que usa os filtros originais (MF-OM), que apresenta uma área de 9,46 Kgates, e uma potência de 3,48 mW. Porém, a solução MF-AM4 teve a maior perda em relação a BD-BR, que foi de 1,25% conforme visto na Tabela 8. Na Tabela 10 é apresentado que a solução MF-AM8 teve uma redução de 76,95% em área e 76,72% em potência, e a solução MF-AM6 teve uma redução de 81,50% em área e 81,89% em potência em relação a solução MF-OM, e ambas as soluções apresentaram um BD-BR de 0,54%.

Tabela 9 – Uso de hardware nas três arquiteturas multi-filtro usadas na FME do AV1

Elemento	MF-AM8	MF-AM6	MF-AM4
Somadores/Subtratores	16	12	8
Mux	12	10	8
Deslocadores	36	32	28

Tabela 10 – Resultados de sínteses para as arquiteturas multi-filtro da FME do AV1

Arquitetura	Freq. (MHz)	Gates (K)	Power (mW)
MF-OM	343	9,46	3,48
MF-AM8		2,18	0,81
MF-AM6		1,75	0,63
MF-AM4		1,32	0,59

6.2 Arquiteturas do Interpolador

Esta subseção apresenta as soluções arquiteturais projetadas para o interpolador da FME do AV1. Todas as soluções usam os filtros apresentados na subseção anterior. Deste modo, cada arquitetura foi definida conforme a versão do filtro usado, e, assim, chamadas de *AV1 FME Interpolation using OM* (AFI-OM) *AV1 FME Interpolation using AM8* (AFI-AM8), *AV1 FME Interpolation using AM6* (AFI-AM6) e *AV1 FME Interpolation using AM4* (AFI-AM4). As arquiteturas foram projetadas para suportarem todos os 42 tipos de filtros, além dos seis filtros de identidade definidos pela FME do AV1. Além disso, as arquiteturas foram projetadas com diferentes níveis de paralelismo. Todas elas operam com tamanhos de bloco 4 x 4, questão já discutida anteriormente.

Quatro versões de arquiteturas de interpolação AV1 FME foram projetadas sendo capazes de processar uma linha a cada ciclo de *clock* e elas foram denominadas de 1-AFI- OM, 1-AFI-AM8, 1-AFI-AM6 E 1-AFI-AM4. Todas elas usam oito instâncias dos filtros, apresentados na subseção anterior. As arquiteturas projetadas são apresentadas na Figura 22, lembrando que a arquitetura (a) representa as arquiteturas 1-AFI-OM e 1-AFI-AM8.

A matriz de entrada, representada por caixas pretas na Figura 22 é dada por $n+3$, onde n é o número de taps usados nos filtros. As arquiteturas foram projetadas para processar 4 x 4 blocos, filtrando uma linha deste bloco por ciclo de *clock*. Como o AV1 define uma filtragem horizontal, seguida de uma filtragem vertical, serão necessárias quatro instâncias das arquiteturas multifiltros para a filtragem horizontal e quatro para a filtragem vertical. Na Figura 22 o H ou V, que precede o nome da arquitetura multifiltro, fo

i usado apenas para identificar a filtragem horizontal (H) ou vertical (V), mas as arquiteturas são idênticas.

Um *Shift-Register-Chain* (SRC) é usado entre cada arquitetura multifiltro horizontal e vertical para sincronizar as operações. Seja n o número de taps nos filtros, os

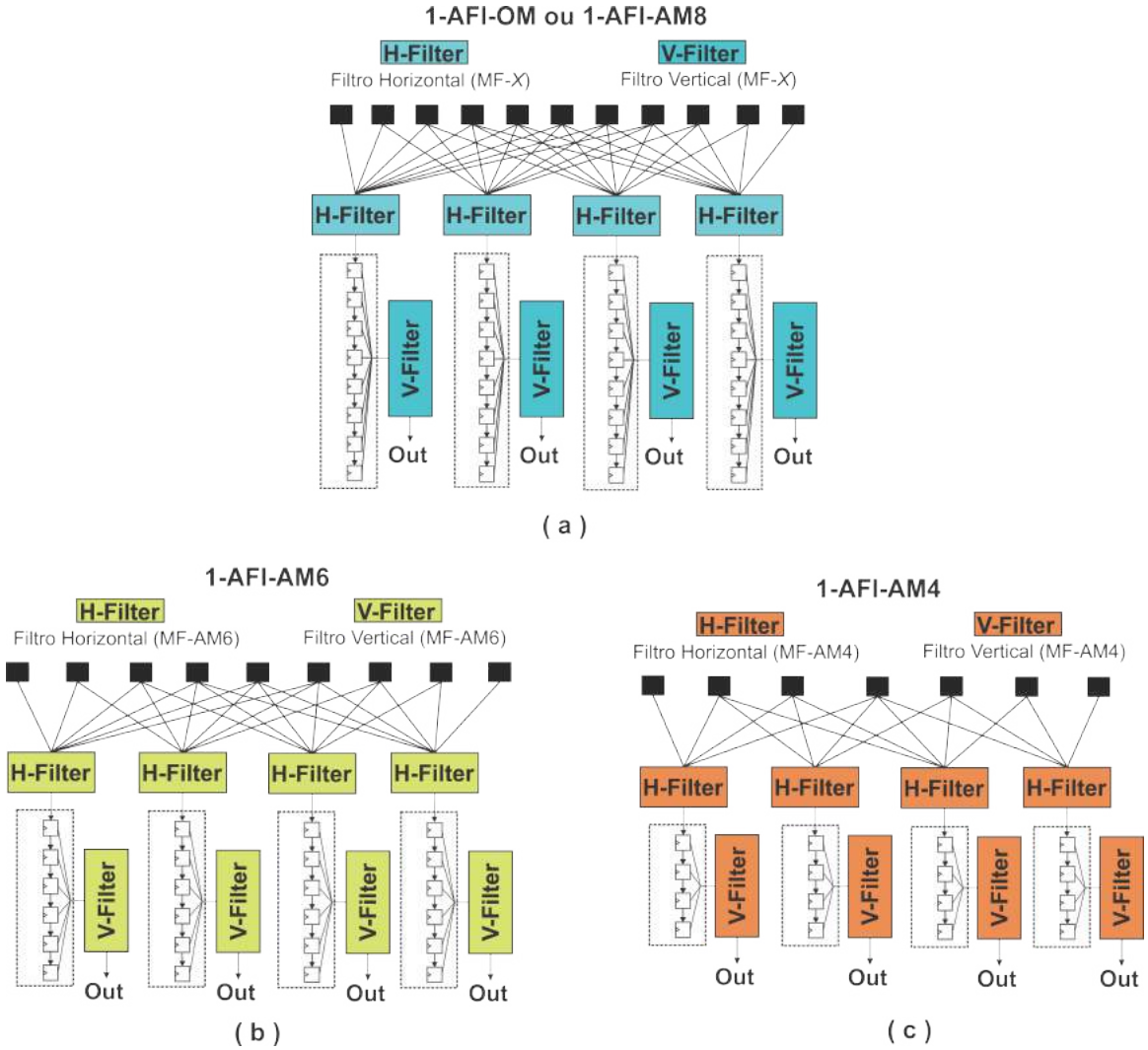

Figura 22 – Arquiteturas dos interpoladores da FME do AV1 usando uma instâncias de MF-OM ou MF-AM8 (1-AFI-OM ou 1-AFI-AM8), MF-AM6 (1-AFI-AM6) e MF-AM4 (1-AFI-AM4).

SRCs têm o mesmo tamanho de n: oito, seis ou quatro posições, dependendo de qual for a solução multifiltro utilizada. Cada SRC armazena os n resultados filtrados horizontalmente, recebendo uma amostra por ciclo de *clock* e após n ciclos, a filtragem vertical pode ser iniciada.

O número de ciclos necessários para interpolar completamente um bloco 4 x 4, na versão da arquitetura que é capaz de processar uma linha a cada ciclo de *clock*, é definido da seguinte forma. A filtragem horizontal requer um ciclo para filtrar cada linha da matriz de entrada, em todos os casos. Então, seja $n \in \{11, 9, 7\}$ a largura e a altura da matriz de entrada, as arquiteturas de filtragem horizontal levarão $n - 3$ ciclos para preencher os SRCs. Outros três ciclos serão necessários, em todos os casos, para finalizar a filtragem vertical. Então, n ciclos são necessários para terminar a interpolação de um bloco 4 x 4. Portanto, as arquitetura 1-AFI-AM8 e 1-AFI-OM

necessitarão de 11 ciclos para processar um bloco 4 x 4, a versão 1-AFI-AM6 utilizará 9 ciclos para interpolar o mesmo bloco de entrada e a versão 1-AFI-AM4 utilizará 7 ciclos. Esta ultima arquitetura é capaz de processar vídeos UHD 4K a 60 fps.

O paralelismo das arquiteturas projetadas pode ser facilmente explorado para aumentar o *throughput* alcançado. Neste caso, podem ser utilizadas quantas instâncias forem necessárias das arquiteturas MF-OM, MF-AM8, MF-AM6 e MF-AM4. Assim, as soluções arquiteturais apresentadas podem atingir diversas taxas de processamento, visando resoluções de vídeo e taxas de quadros distintas e, para atingir o *throughput* desejado, devem ser utilizadas quantas instâncias da arquitetura forem necessárias.

Sendo assim, um segundo conjunto de arquiteturas foi projetado, com a capacidade de processar quatro linhas a cada ciclo de *clock*. Essas arquiteturas foram denominadas de 4-AFI-OM, 4-AFI-AM8, 4-AFI-AM6 e 4-AFI-AM4. Cada arquitetura foi projetada com 32 instâncias dos filtros apresentados na subseção anterior. As arquiteturas projetadas são apresentadas na Figura 23.

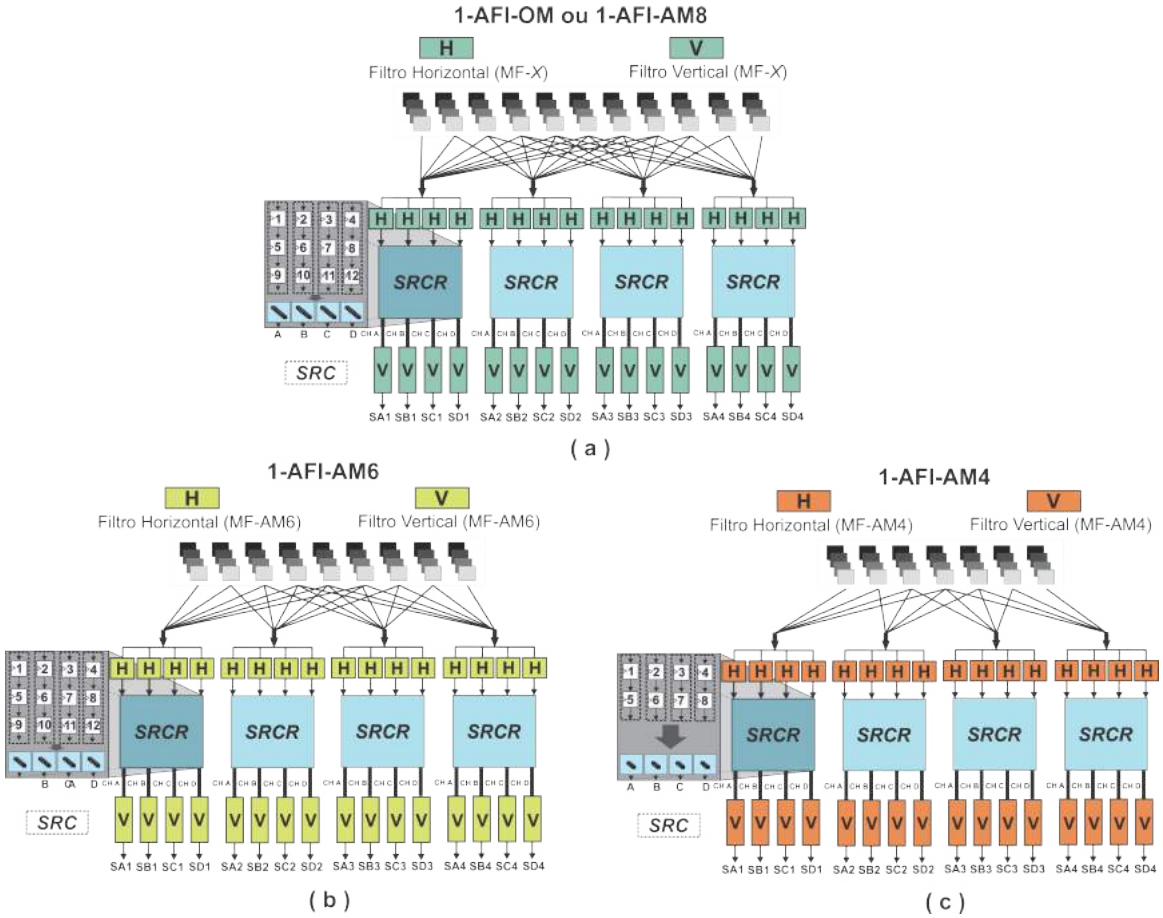

Figura 23 – Arquiteturas dos interpoladores da FME do AV1 usando quatro instâncias de MF-OM ou MF-AM8 (4-AFI-OM ou 4-AFI-AM8), MF-AM6 (4-AFI-AM6) e MF-AM4 (4-AFI-AM4).

Na Figura 23, as matrizes de entrada, representadas em tons de cinza, que são

dadas por $n+3$, onde n é o número de taps usados nos filtros. Entretanto, como estas arquiteturas processam quatro linhas por ciclo de *clock*, elas estão agrupadas em um conjunto de quatro linhas. Observando a Figura 23, é possível notar que ambas as arquiteturas foram projetadas com 16 instâncias de H$-X$, 16 instâncias de V$-X$ e quatro cadeias de roteadores (SRCR na Figura 23). Lembrando que, H ou V que precede o nome da arquitetura multifilter foi usado apenas para identificar a filtragem horizontal (H) ou vertical (V), porém, novamente, as arquiteturas são idênticas.

Os roteadores SRCR mostrados na Figura 23 são responsáveis por armazenar e conectar as amostras dos filtros H$-X$ para as instâncias dos filtros V$-X$. O SRCR é composto por quatro cadeias de SRC, que, dependendo da arquitetura, o número de registradores que compõem cada um pode diferenciar, sendo para as arquiteturas 4-AFI-OM, 4-AFI-AM8, 4-AFI-AM6 são projetadas com três registradores e a 4-AFI-AM4 com dois registradores, que são capazes de armazenar a amostra de saída correspondente da instância H$-X$.

O número de ciclos necessários para interpolar um bloco 4 x 4 por completo nas arquiteturas, capazes de processar quatro linhas por ciclo, é determinado da seguinte maneira. A filtragem horizontal requer um ciclo para processar quatro linhas da matriz de entrada, em cada instância. Logo, $n \in \{11, 9, 7\}$ seja a largura e a altura da matriz de entrada, as arquiteturas de filtragem horizontal levarão $(((n-3)/4+1)$ ciclos para preencher os SRCRs. Como resultado, as arquiteturas 4-AFI-OM, 4-AFI-AM8, 4-AFI- AM6 exigirão 3 ciclos para processar um bloco 4 x 4, enquanto a versão AFI-AM4 exigirá 2 ciclos, para processar o mesmo bloco. Esta última arquitetura é capaz de processar vídeos UHD 8K a 60 fps. Por fim, as saídas de cada arquitetura são compostas pelas saídas entregues por todas as instâncias V$-X$.

Para a versão que processa uma linha por ciclo de *clock*, é necessária uma frequência de operação de 343 MHz para processar vídeos UHD 4K a 60 quadros por segundo. Então, esta foi a primeira síntese para as quatro versões arquiteturais (1-AFI-OM, 1-AFI-AM8, 1-AFI-AM6 e 1-AFI-AM4). Agora, usando a mesma frequência de 343 MHz, mas processando quatro linhas por ciclo de clock (4-AFI-OM, 4-AFI-AM8, 4-AFI-AM6 e 4-AFI-AM4), é possível atingir uma taxa de processamento suficiente para processar vídeos UHD 8K a 60 quadros por segundo. Os resultados dessas sínteses são apresentados na Tabela 11

A primeira conclusão importante ao observar os resultados da Tabela 11 é que os diferentes níveis de imprecisão explorados neste trabalho causaram um impacto em termos de área e potência, independentemente do nível de paralelismo explorado. Os ganhos de área variam de 69,81% a 82,61% e os ganhos de potência variam de 68,09% a 81,24%, dependendo da aproximação explorada. Neste ponto, é importante ressaltar, que os resultados de potência não estão considerando a expressiva redução de comunicação de memória alcançada pelas versões arquiteturais $X-$AFI-AM6 e

Tabela 11 – Resultados de sínteses para as arquiteturas do interpolador usado na FME do AV1

Arquitetura	Freq. (MHz)	Gates (K)	Power (mW)	Throughput
1-AFI-OM	343	58,05	24,25	2160@60fps
1-AFI-AM8		17,03	6,12	
1-AFI-AM6		14,48	4,71	
1-AFI-AM4		10,09	4,79	
4-AFI-OM		203,18	98,21	4320@60fps
4-AFI-AM8		61,32	31,33	
4-AFI-AM6		49,25	22,75	
4-AFI-AM4		35,84	20,36	

$X-$AFI-AM4.

Outro ponto interessante é que mesmo com os mesmos resultados de eficiência de codificação, a arquitetura 4-AFI-AM6 pode atingir uma redução de potência de até 27,38% se comparada com a arquitetura 4-AFI-AM8 e uma redução de área de até 19,68%.

A Tabela 11 também demonstra que aumentar o número de instâncias da arquitetura leva a um aumento proporcional no *throughput*, mas com impactos importantes na potência e na área utilizada, como esperado.

6.3 Arquitetura da FME Completa

A arquitetura desenvolvida para a FME do AV1 foi projetada focando em blocos 4 x 4 como unidade básica de processamento, conforme já discutido. A arquitetura de interpolação suporta todos os 48 filtros de interpolação definidos na FME pelo codificador AV1, sendo eles os sete filtros para cada família, totalizando 42 filtros, além disso, suporta os seis filtros de identidade, e foi baseada na solução apresentada na Subseção 6.1.

A Figura 24 apresenta o diagrama de blocos de nível superior do projeto AV1 FME desenvolvido. A arquitetura FME proposta possui duas unidades principais: a Unidade de Interpolação (UI) e a Unidade de Melhor Casamento (UMC).

Esta arquitetura é alimentada por duas matrizes de amostra de entrada, exigidas pelo processo da FME do codificador AV1. A primeira matriz possui NxN amostras, onde N representam as amostras do melhor bloco (4 x 4) encontrado na IME e as

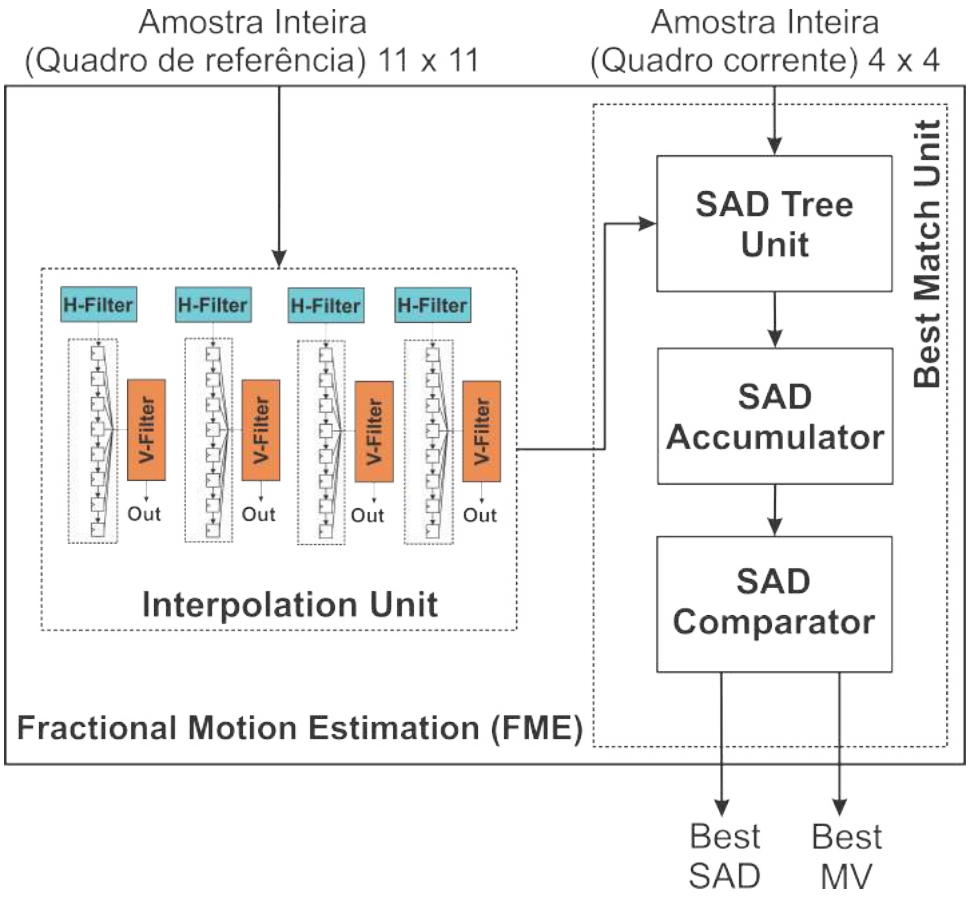

Figura 24 – Diagrama da arquitetura completa da FME usada no codificador AV1.

amostras vizinhas necessárias para o processo da interpolação. A outra matriz é uma matriz de amostras 4 x 4 que contém o bloco atual a ser codificado.

A Unidade de Melhor Casamento é responsável por calcular a Soma das Diferenças Absolutas (*Sum of Absolute Differences* - SAD) e entre as amostras do bloco atual e as amostras geradas pela Unidade de Interpolação e comparar todos os novos blocos candidatos dentro das amostras fracionárias com o bloco atual e definir a melhor correspondência. O SAD nada mais é do que um critério de distorção, e é o critério mais utilizado, principalmente para projetos de hardware, devido a sua simplicidade.

Inicialmente, o SAD de todos os blocos, compostos por amostras fracionárias é calculado no Unidade de Árvore de SAD (UA-SAD). O resultado de saída é armazenado no Acumulador de SAD (A-SAD) de acordo com cada posição de amostra fracionária, previamente calculada. Por fim, todos os valores são comparados no Comparador de SAD (C-SAD), e o melhor valor de SAD e seu respectivo VM fracionário são entregues na saída da arquitetura.

Um total de 64 (63 fracionários mais a amostra inteira) blocos 4 x 4 candidatos são gerados após o processo de interpolação, em seguida, eles são comparados com o bloco 4 x 4 atual para gerar a melhor correspondência após o processamento da FME.

Quatro versões da arquitetura da FME, tendo como base a Figura 24, foram proje-

tadas com diferentes níveis de paralelismo, e diferentes versões de filtros, que foram apresentados na Subseção 6.1. O nível de paralelismo entre uma arquitetura e outra difere no número de instâncias de filtros suportadas dentro de cada UI, e instâncias de UA-SAD na unidade UMC. As duas primeiras arquiteturas foram chamadas de *4 - AV1 FME usando OM* (4-AF-OM) e *4 - AV1 FME usando AM6* (4-AF-AM6), ambas com 32 instâncias dos filtros. Posteriormente, há as arquiteturas *8 - AV1 FME usando OM* (8-AF-OM) e *8 - AV1 FME usando AM6* (8-AF-AM6), onde cada uma contêm 64 instâncias dos filtros.

O filtro AM6 foi definido para este projeto, pois obteve uma maior economia de hardware e potência em relação ao AM8 e teve o mesmo resultado na eficiência de codificação. O número de instâncias dos filtros usados no interpolador foi definido para quatro e oito visando um maior *throughput*.

6.3.1 Unidade de Interpolação

O filtro de interpolação foi projetado utilizando a arquitetura de interpolação multi filtro MF-OM e MF-AM6 apresentados na Subseção 6.1. Já a arquitetura dos interpoladores usados foi apresentado na Subseção 6.2.

A primeira versão das arquiteturas, usada no Unidade de Interpolação, denominada 4-AF-X é apresentada na Figura 25. Estas arquiteturas são compostas por 32 instâncias do hardware multifiltro (OM ou AM6), os quais foram apresentados na Subseção 6.1, sendo 16 filtros horizontais (H-X) e 16 filtros verticais (V-X), sendo X apenas a diferença entre os filtros usados. Os demais detalhes sobre a arquitetura do interpolador já foram abordados na Subseção 6.2.

A segunda versão da arquitetura, usada na UI, denominada 8-AF-X é apresentada na Figura 26. Esta arquitetura é composta por 64 instâncias do hardware multifiltro (OM ou AM6) apresentados na Subseção 6.1: 32 filtros horizontais (H-X) e 32 filtros verticais (V-X), sendo X apenas a diferença entre os filtros usados (OM ou AM6). Os diferentes nomes são usados apenas para destacar quais instâncias são usadas nos processos de filtragem vertical e horizontal. A UI também usa 32 cadeias de registradores de deslocamento (cada uma com oito posições para a solução usando OM e seis posições para a solução que usa AM6), denominada de SRC, para conectar os filtros, representados pelas caixas pontilhadas na Figura 26. Cada caixa em tom de cinza na Figura 26 ilustra uma entrada de amostra de luminância.

6.3.2 Unidade de Melhor Casamento

Os três módulos principais da Unidade de Melhor Casamento são apresentados na Figura 27. A Figura 27 (a) apresenta o Unidade de Árvore de SAD , que é capaz de calcular o SAD de oito amostras de entrada (identificadas como "I" na Figura 27 (a)) do bloco atual sendo e oito amostras geradas pela Unidade de Interpolação

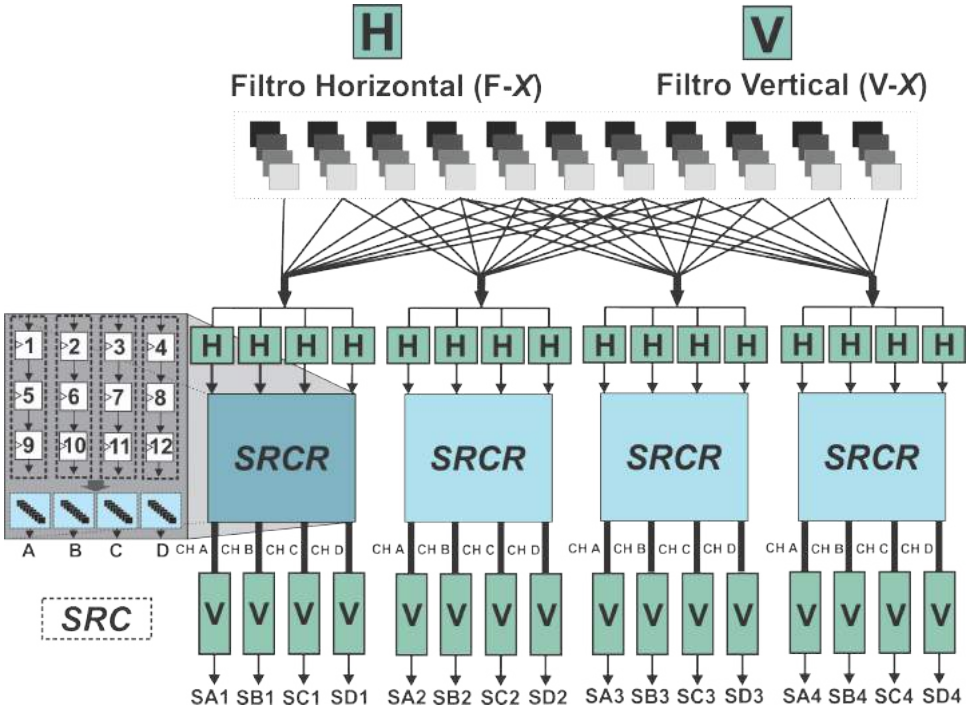

Figura 25 – Arquitetura do interpolador usado no bloco UI com quatro instâncias dos multi-filtros projetados denominada 4-AF-X.

(identificada como "F" na Figura 27 (a)). Dois grupos de oito instâncias da arquitetura UA-SAD são usados dentro da UMC, sendo assim, um novo SAD é gerado a cada ciclo de *clock*, após a latência inicial, para a solução 4-AF-X. Quatro grupos de oito instâncias da arquitetura UA-SAD são usadas para a solução 8-AF-X, assim, gerando dois novos SADs a cada ciclo de *clock*. Cada grupo recebe as 16 amostras "I" do bloco 4 x 4 que está sendo codificado e 16 ou 32 amostras "C" do bloco 4 x 4 interpolado, dependendo da solução. A cada novo ciclo as amostras "C" são atualizadas, mas as amostras "I" não são alteradas. Conforme apresentado na Figura 27 (a), a arquitetura UA-SAD divide a árvore de SAD em dois estágios de pipeline para aumentar a taxa de processamento, e o registrador de pipeline, foi posicionado entre a primeira e a segunda linha de somadores.

A Figura 27 (b) apresenta o A-SAD, que é o acumulador usado em todas as arquiteturas. Nas arquiteturas 4-AF-X e 8-AF-X, A-SAD é composto por um conjunto com 64 registradores (organizados em uma matriz com 8 × 8). Sendo assim, todos os SADs do módulo UA-SAD são armazenados, coluna por coluna, até que todos os SADs, de todos os blocos candidatos, sejam armazenados em sua respectiva posição no A-SAD.

A Figura 27 (c) apresenta o módulo Comparador de SAD. Este módulo realiza a comparação de oito SADs por ciclo de *clock*. No final de cada iteração, o SAD anterior para aquela iteração também é comparado com os valores anteriores acumulados no registrador BEST-SAD, e assim, o registrador BEST-SAD sempre terá o melhor SAD

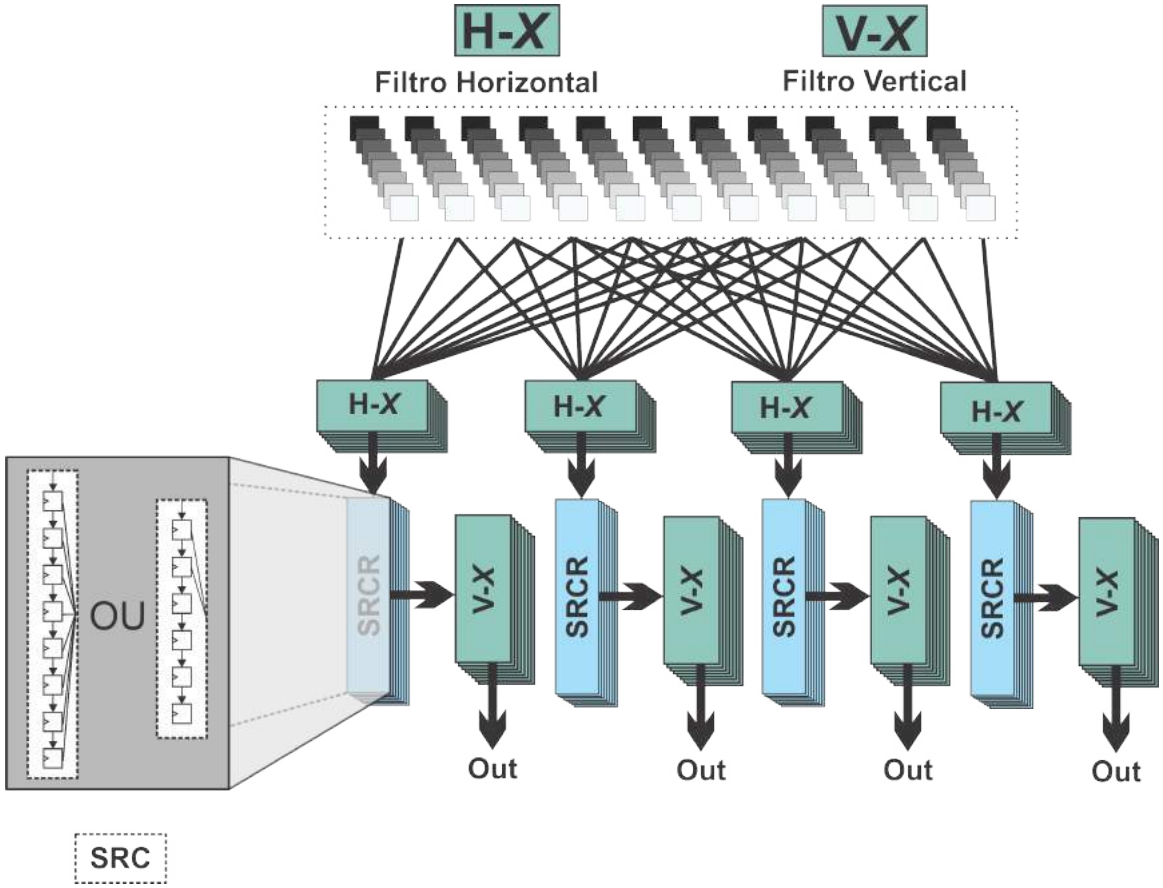

Figura 26 – Arquitetura do interpolador usado no bloco UI com oito instâncias dos multi-filtros projetados denominada 8-AF-X.

até o momento. Portanto, após oito ciclos, todos os SADs previamente calculados são processados, e o módulo C-SAD então define e entrega um par com o melhor SAD e o seu respectivo MV. As estruturas C-SAD são idênticas para ambas as soluções.

6.3.3 Análise e Resultados

A Figura 28 apresenta um diagrama temporal para as arquiteturas. Na Figura 28 (a) são mostrados os ciclos de *clock* necessários para processar um bloco 4 x 4 usando a solução 4-AF-X. Primeiro, há a latência para cumprir as cadeias de registros do interpolador (dois ciclos). Após a latência, amostras fracionárias começam a ser entregues na saída, e os primeiros SADs parciais são armazenados nos A-SAD, através do resultado obtido na UA-SAD. Desde o início, o interpolador leva 65 ciclos para calcular todos as amostras fracionárias. No entanto, as outras etapas começam a operar paralelamente, enquanto ocorre a interpolação. Após o término do interpolador, a UA-SAD gera seu último resultado e, então, com mais um ciclo, o A-SAD armazena o último SAD. Os SADs são então comparados no C-SAD, que leva oito ciclos para finalizar as comparações. Como resultado, o processamento de um bloco 4 x 4 nas arquiteturas 4-AF-X leva 74 ciclos. Já a Figura 28 (b) apresenta os ciclos de *clock* necessários

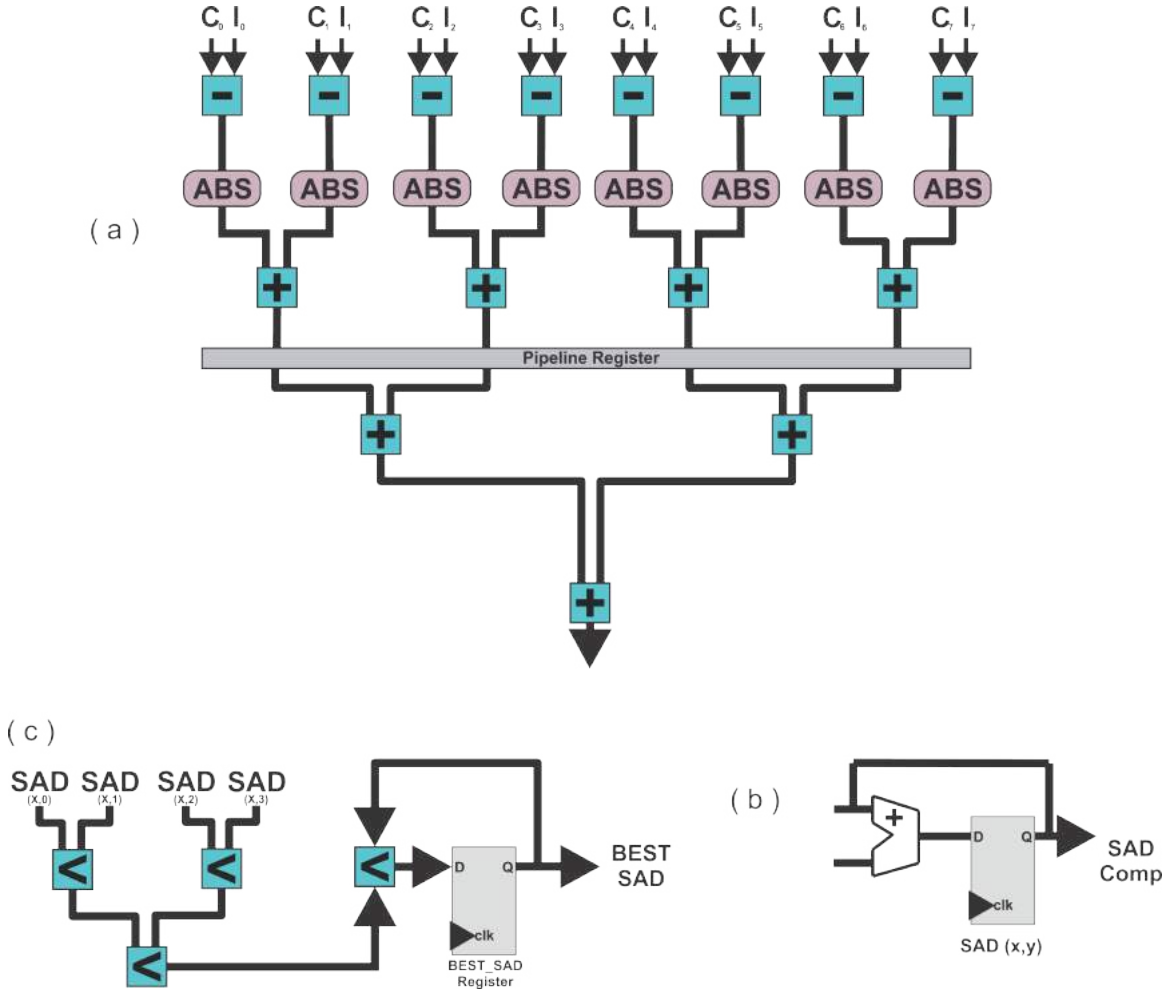

Figura 27 – Arquitetura da etapa de Unidade de Melhor Casamento da FME do AV1: (a) Arquitetura Unidade de Árvore de SAD ; (b) Arquitetura Acumulador de SAD (c) Arquitetura Comparador de SAD.

para o mesmo bloco 4 x 4, usando a solução 8-AF-X. A principal diferença está na latência inicial (um ciclo) e nos totais de ciclos que são necessários para gerar todas as amostras fracionárias, que agora são 32 ciclos. As demais etapas são semelhantes em ambas as soluções, sendo assim, o processamento do mesmo bloco 4 x 4 usando a solução 8-AF-X leva 42 ciclos.

Com os ciclos de cada arquitetura definidos, agora é possível definir a frequência alvo, conforme a Equação 19. Portanto, a frequência de operação requerida foi de 638MHz. Cabe destacar que, devido ao nível de paralelismo, cada solução alcança uma taxa de processamento diferente. A 4-AF-X é capaz de processar vídeos Full HD 60 fps, enquanto a solução 8-AF-X processa vídeos na resolução UHD 4K 30fps.

A Tabela 12 apresenta os resultados de síntese das arquiteturas da FME, propostas, considerando uma frequência de 638MHz. A versão 4-AF-AM6 teve uma economia de 69,26% de área e 77,46% de potência em relação a arquitetura 4-AF-OM. Ambas entregam a mesma taxa de processamento. A arquitetura 8-AF-AM6 teve uma

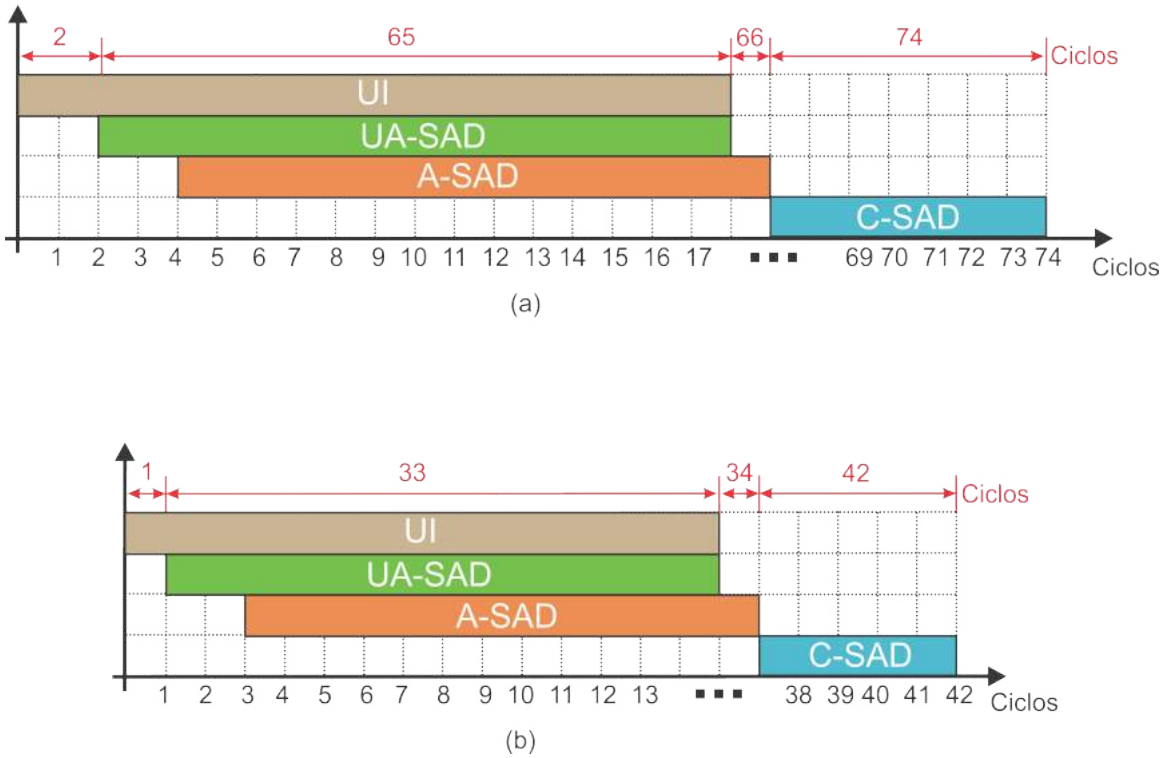

Figura 28 – Diagrama temporal da FME do AV1: (a) Processamento da arquitetura 4-AF-X; (b) Processamento da arquitetura 8-AF-X.

economia de 73,22% de área e 78,05% de potência em relação a arquitetura 8-AF-OM e, novamente, ambas entregam a mesma taxa de processamento. Entretanto, quando comparadas as arquiteturas 8-AF-OM e 4-AF-OM, que usam o mesmo conjunto (OM) de multifiltro, porém, com diferentes níveis de paralelismo, nota-se que a 8-AF- OM gasta 51,30% de área e 49,55% de potência a mais que a 4-AF-OM. Já quando comparadas as arquiteturas que usam o AM6, a 8-AF-AM6 gasta 58,88% de área e 51,02% de potência a mais que a 4-AF-AM6.

Portanto, é possível identificar que mesmo tendo uma pequena perda na eficiência de codificação, a arquitetura projetada com os multifiltros AM6 apresenta ótimos ga-

Tabela 12 – Resultados de sínteses para as arquiteturas completa da FME do AV1

Arquitetura	Freq. (MHz)	Gates (K)	Power (mW)	Throughput
4-AF-OM	638	250,82	164,51	1080@60fps
4-AF-AM6		77,08	37,07	
8-AF-OM		489,02	331,42	4320@30fps
8-AF-AM6		130,93	72,72	

nhos de área e potência, em relação aos filtros originais OM. Também é importante notar que quando analisadas apenas as arquiteturas com o mesmo conjunto de multifiltros, como esperado, é possível obter uma taxa de processamento maior com um maior nível de paralelismo, entretanto, há um custo proporcional de área e de potência.

Este capítulo apresentou a segunda arquitetura projetada nesta tese, que visa a estimação de movimento Fracionária do AV1 e, até o momento, não foi encontrado nenhum outro trabalho na literatura apresentando o desenvolvimento de hardware para a FME completa do AV1. Entretanto, o trabalho de Freitas et al. (2020) propõe uma arquitetura de hardware apenas para a família de filtros Regular da FME do codificador AV1. Para tamanhos de blocos 4 x 4, sua arquitetura obteve uma área de 71,0 Kgates e uma potência dissipada de 37,96 mW à uma frequência de 476 MHz. Como são usados apenas os filtros da família Regular, deixando de lado os demais, uma comparação não seria precisa. Existem outros trabalhos na literatura com implementações em hardware para a FME para outros padrões, onde uma comparação não seria justa, visto que, os filtros empregados na FME do AV1 são muito diferentes em termos algoritmos e de complexidade.

7 ARQUITETURAS PARA A COMPENSAÇÃO DE MOVIMENTO DISTORCIDO

Este capítulo discute o projeto de hardware de alto desempenho, visando a Compensação de Movimento Distorcido do codificador AV1. Essa solução buscou explorar essa nova ferramenta inserida no codificador AV1, tendo como principal intuito a redução da complexidade. Portanto, este capítulo será dividido em duas etapas, sendo que, a primeira visa a ferramenta de compensação de movimento distorcido local e a segunda etapa a compensação de movimento distorcido global. A metodologia utilizada para o desenvolvimento dessas arquiteturas é a mesma apresentada na Seção 5.1.

7.1 Arquitetura para a Compensação de Movimento Distorcido Local

As arquiteturas apresentadas nesta seção foram projetadas para processar tamanhos de blocos de 8 x 8, que é o menor tamanho de bloco suportado pela compensação de movimento distorcido local (*Local Warped Motion Compensation* - LWMC) do AV1. Usando este tamanho de bloco, é possível processar blocos maiores, dividindo os blocos menores em 8 x 8 e, assim, é possível processar todos os blocos suportados pelo padrão.

A arquitetura do filtro da LWMC foi projetada em dois níveis hierárquicos. No primeiro nível, um hardware de multifiltro combinacional, chamado de Arquitetura Multifiltro para a Compensação de Movimento Distorcido Local sando (AMDL), é usada para calcular cada amostra filtrada, vertical ou horizontal, em um bloco de entrada de 8 x 8. O segundo nível hierárquico define a estrutura usada para interpolar um bloco usando LWMC e essa arquitetura foi denominada Interpolador para a Compensação de Movimento Distorcido Local (IDL).

Os resultados do hardware desenvolvido para a LWMC foi aceito para apresentação no *IEEE International Conference on Image Processing* (ICIP) 2023 (DOMANSKI et al., 2023b).

7.1.1 Arquitetura Multifiltro para a Compensação de Movimento Distorcido Local

A implementação da arquitetura AMDL é dívida em dois cisalhamentos: um horizontal e um vertical. Isso é feito para aplicar os filtros de interpolação, usados para melhorar LWMC, que é aplicada nas direções horizontais e verticais. A LWMC do AV1 define 192 filtros FIR de oito taps, com precisão de $1/64$ pixel.

A arquitetura AMDL é apresentada na Figura 29. Como o nome sugere, essa arquitetura foi projetada para suportar todos os 192 filtros definidos no AV1 LWMC, usando um único hardware multifiltro. A AMDL foi projetado de forma puramente combinacional, permitindo o processamento paralelo das oito amostras de entrada necessárias para gerar cada amostra filtrada.

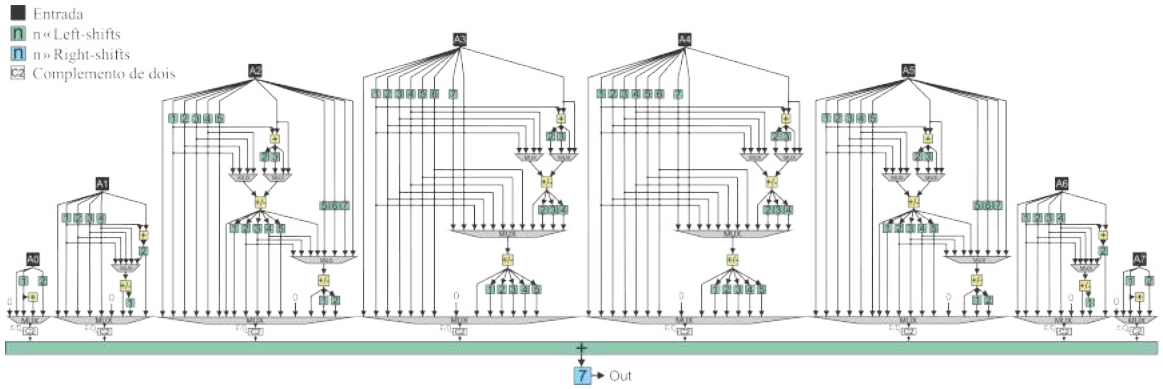

Figura 29 – Arquitetura multi-filtro para a Compensação de Movimento Distorcido Local (AMDL).

A arquitetura AMDL explora uma abordagem de *Multiple Constant Multiplication* (MCM) (VORONENKO; PüSCHEL, 2007) e compartilhamento de subexpressões entre todos os filtros suportados, reduzindo o uso de hardware e o consumo de energia, enquanto aumenta a taxa de processamento da arquitetura. Seus filtros são simétricos e esse comportamento é explorado pelo projeto AMDL. Por exemplo, o filtro na posição $1/64$ utiliza os mesmos coeficientes que o filtro na posição $63/64$, alterando apenas a ordem de entrada.

Na Figura 29, as amostras $A0 - A7$ representam as amostras de entrada e out representa a saída do filtro. Cada amostra pode ser calculada usando deslocadores (representados pelas caixas verdes na Figura 29) e somadores/subtratores (representados pelas caixas amarelas na Figura 29). Os multiplexadores são responsáveis por selecionar o tap do filtro e definir qual operaçãode somas e deslocamentos será realizada, de acordo com o filtro selecionado.

O resultado de síntese dessa arquitetura é apresentado posteriormente junto a arquitetura do interpolador completo da LWMC.

7.1.2 Interpolador para a Compensação de Movimento Distorcido Local

Esta subseção apresenta duas soluções arquiteturais, projetadas para o interpolador usado na LWMC. Ambas as soluções usam da arquitetura de multifiltro AMDL, apresentada na subseção anterior. O que diferencia uma solução da outra é o nível de paralelismo entre elas. Deste modo, as duas soluções foram denominadas de Arquitetura do Interpolador para Compensação do Movimento Distorcido Local com Uma Linha por Ciclo (1L-AIDL) e Arquitetura do Interpolador para Compensação de Movimento Distorcido Local com Quatro Linhas por Ciclo (4L-AIDL). As arquiteturas foram projetadas para suportar todos os 192 tipos de filtros suportados na LWMC do AV1. Todas as arquiteturas operam com tamanhos de bloco 8 x 8, questão já discutida anteriormente.

A Figura 30 apresenta a arquitetura 1L-AIDL. A arquitetura 1L-AIDL usa 16 instâncias da arquitetura AMDL. Oito filtros horizontais (H-Filter na Figura 30) e oito filtros verticais (V-Filter na Figura 30), além de oito Cisalhamentos Horizontais (HS na Figura 30) e oito Cisalhamentos Verticais (VS na Figura 30) usados para o cálculo da Transformada Afim. Por fim, também são necessárias três matrizes, uma matriz de entrada com 15 x 15 amostras (destacadas em vermelho na Figura 30), uma matriz intermediária com 8 x 15 amostras (destacadas em azul na Figura 30) e a matriz de saída com 8 x 8 amostras (destacadas em verde na Figura 30).

Para gerar um bloco 8 x 8, o 1L-AIDL segue o fluxo de nove passos mostrado na Figura 30. Ao longo deste fluxo, as linhas pontilhadas (passos 1, 2, 5 e 6) estão especificando quais posições da matriz devem ser avaliadas, ou seja, os passos (cálculos) usados na transformação afim; por outro lado, as linhas contínuas (passos 0, 3, 4, 7 e 8) indicam quais amostras dessas matrizes estão sendo processadas e interpoladas.

Primeiramente, durante o passo 0, a LWMC é iniciada. No passo 1, durante o primeiro ciclo do *clock*, as posições (0,0; 0,1; ... 0,7) são coletadas da matriz intermediária (representada em azul na Figura 30), e inseridas no cálculo do cisalhamento horizontal (HS).

A seguir, durante o passo 2, cada uma dessas posições da matriz intermediária são encontradas na matriz de entrada (detalhada em vermelho na Figura 30), através do cálculo HS (demostrado na Equação 15 e exemplificado na Figura 30), definindo o ponto central e encontrando, assim, o ponto correspondente de cada posição na matriz de referência. Com a posição definida pelo cálculo do cisalhamento, a arquitetura obtém três amostras à esquerda e quatro amostras à direita do pixel central definido por HS, para executar o filtro horizontal (H-Filter na Figura 30).

Na sequência, durante o passo 3, a partir desta definição, as amostras necessárias são coletadas da matriz de entrada (amostras de entrada são representadas em amarelo na Figura 30) para realizar a filtragem horizontal nos filtros horizontais (H-Filter na na Figura 30). Lembrando que, um bloco de tamanho 8 x 8 requer uma

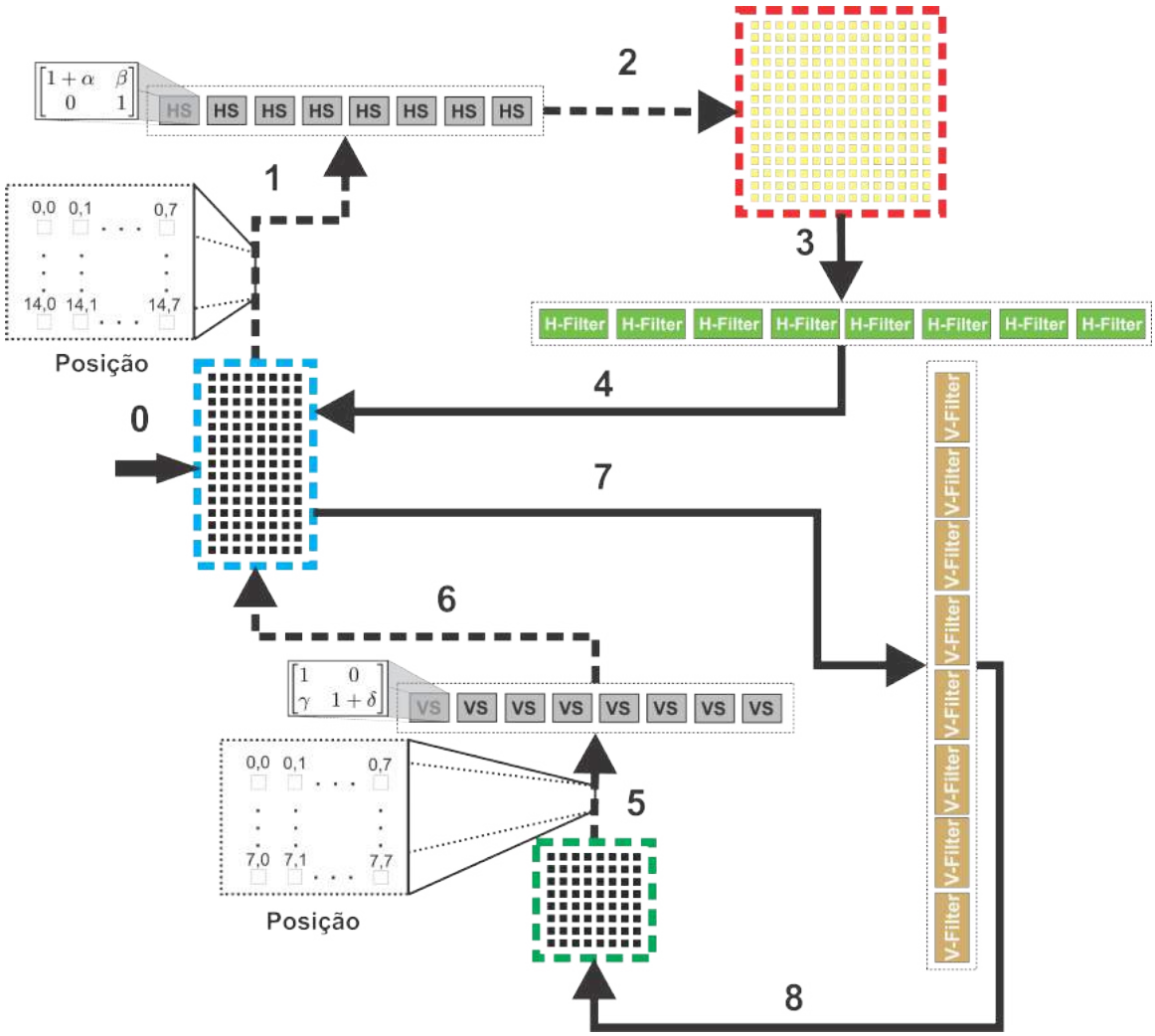

Figura 30 – Arquitetura do Interpolador para Compensação do Movimento Distorcido Local com Uma Linha por Ciclo (1L-AIDL).

matriz de entrada com tamanho 15 x 15, visto que, além das oito amostras do bloco, que está sendo avaliado, mais sete amostras (três à esquerda e quatro à direita) são necessárias para realizar a interpolação.

As saídas da filtragem horizontal preenchem as posições (0,0; 0,1; ... 0,7) da matriz intermediária durante a etapa 4. Esse processo é repetido até que a matriz intermediária 8 x 15 esteja completamente preenchida, o que leva quinze ciclos.

Na etapa 5, as posições (0,0; 0,1; ... 0,7) da matriz intermediária são coletadas para realizar o Cisalhamento Vertical (VS). Este passo é semelhante ao HS, mas, desta vez, as posições são encontradas olhando para a matriz intermediária, ao invés da matriz de entrada, como acontece durante o HS.

O cálculo do VS é realizado no passo 6 (demostrado na Equação 15 e exemplificado na Figura 30). Em seguida, com o ponto central definido, no passo 7 as amostras são organizadas para realizar a filtragem vertical, que acontece nos blocos V-Filter. Além da amostra do bloco em avaliação (amostra extraída do ponto central), são ne-

cessárias mais sete amostras (três acima e quatro abaixo) para realizar a interpolação, desta vez no sentido vertical.

Finalmente, no passo 8 (que acontece durante o 16º ciclo do *clock*), as saídas dos V-Filters preenchem as posições (0,0; 0,1; ... 0,7) da matriz de saída e assim por diante até completar todo o bloco 8 x 8, utilizando um ciclo por linha. Assim, após 23 ciclos de *clock*, todo o processo necessário para gerar um bloco de 8 x 8 está concluído.

A Figura 31 apresenta a segunda arquitetura proposta, denominada 4L-AIDL, que pode processar quatro linhas do bloco de entrada por ciclo. A 4L-AIDL foi projetada usando 64 instâncias do filtro AMDL, 32 filtros horizontais (H-Filter na Figura 31) e 32 filtros verticais (V-Filter na Figura 31), 32 instâncias de cisalhamento horizontal (HS) e cisalhamento vertical (VS), além da matriz de referência 15 x 15, da matriz intermediária 8 x 15 e da matriz resultante 8 x 8.

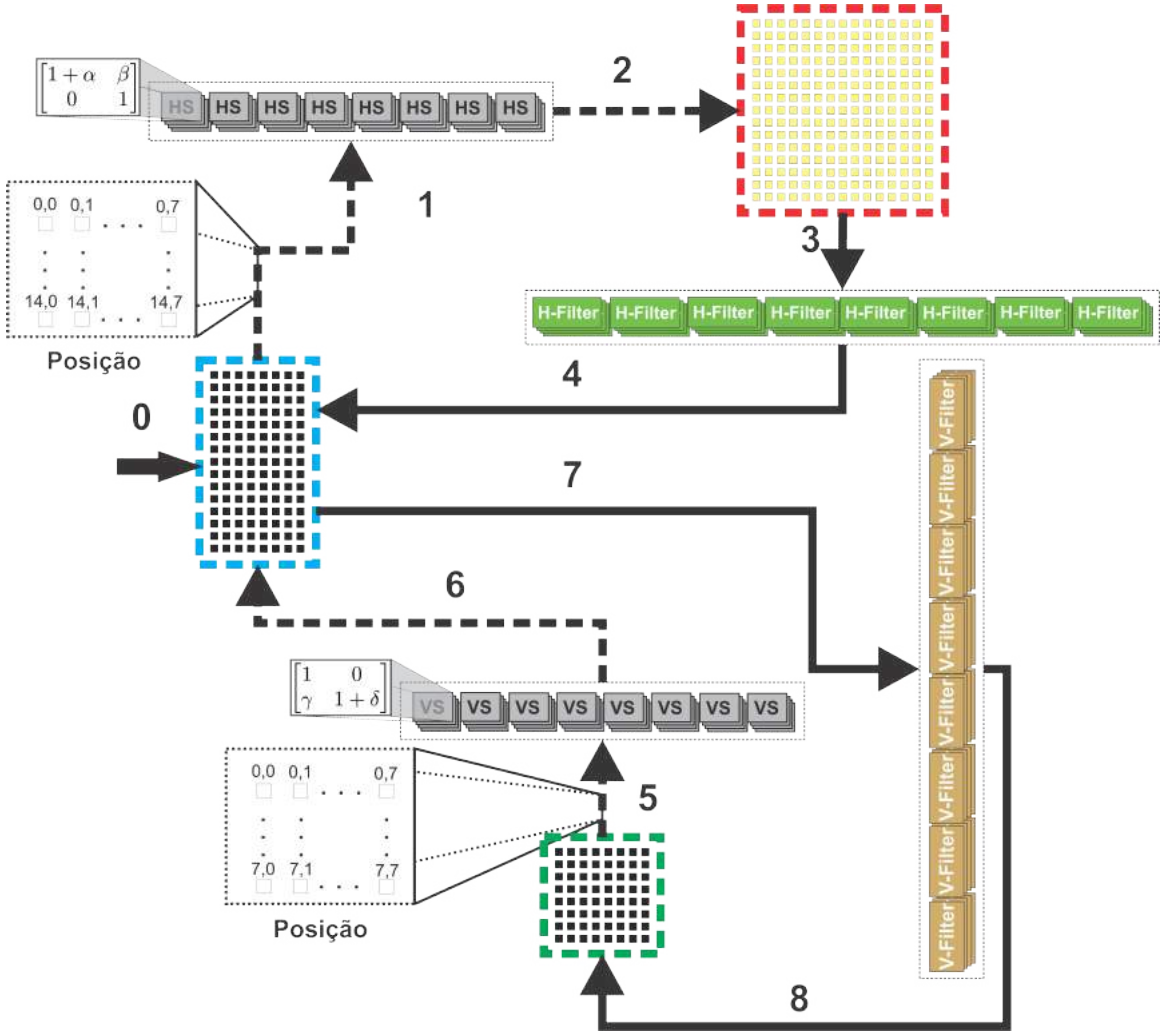

Figura 31 – Arquitetura do Interpolador para Compensação de Movimento Distorcido Local com Quatro Linhas por Ciclo (4L-AIDL)

Essa arquitetura se comporta de forma idêntica a 1L-AIDL, a principal diferença

entre elas é que a versão 4L-AIDL leva apenas quatro ciclos de *clock* para executar a primeira etapa (passos 0, 1, 2, 3, 4) e seis ciclos de *clock* para gerar o bloco 8 x 8 resultantes (passos 5, 6, 7, 8). Essa eficiência, na redução do número de ciclos, se deve ao paralelismo empregado, com as diferentes instâncias dos filtros e dos cisalhamentos, utilizados na arquitetura.

7.1.3 Análise e Resultados

Duas arquiteturas visando o interpolador usado na LWMC do AV1, com diferentes níveis de paralelismo, foram apresentados neste capítulo da tese, incluindo a arquitetura dos filtros usados.

A 1L-AIDL, que processa uma linha do bloco de entrada por ciclo, foi projetada para processar vídeos UHD 4K a 60fps. Já a 4L-AIDL, que utiliza quatro vezes mais instâncias de H-Filters, V-Filters, HSs e VSs e, assim, processa quatro linhas do bloco de entrada em paralelo, foi projetada para processar vídeos UHD 8K a 60fps.

Com a definição dos ciclos que cada arquitetura leva para processar um bloco 8 x 8, e com base na Equação 19, a frequência alvo de operação foi definida como 473MHz para todas as arquiteturas. Os resultados das arquiteturas projetadas são apresentados na Tabela 13.

Tabela 13 – Resultados de síntese das arquiteturas projetadas para a LWMC do AV1.

Arquitetura	Frequência	Gates (K)	Power (mW)	Thougpuht
AMDL	473MHz	7,47	2,74	-
1L-AIDL		115,20	47,10	4K@60fps
4L-AIDL		454,37	189,35	8K@60fps

Ao comparar as versões 4L-AIDL e 1L-AIDL nota-se que a 4L-AIDL ocupa quase 3,9 vezes mais área e dissipa quatro vezes mais potência que a 1L-AIDL, conforme esperado. Então, a relação entre potência, área e *throughput* se comporta bem quando são comparadas as duas arquiteturas.

Como não há na literatura outros trabalhos que apresentem projeto de hardware dedicado visando a LWMC, é impossível fornecer uma comparação justa da presente pesquisa com outros resultados.

7.2 Arquitetura para a Compensação de Movimento Distorcido Global

Esta seção apresenta a arquitetura projetada para a Compensação de Movimento Distorcido Global (*Global Warped Motion Compensation* - GWMC) do AV1. A Figura

32 apresenta o fluxograma da arquitetura desenvolvida para a GWMC. Conforme é possível observar na Figura 32, a arquitetura possui quatro etapas diferentes: Detecção, Descrição, Casamento e Estimação. Cada uma delas será apresentada a seguir nesta seção.

Destaca-se que arquitetura projetada para a GWMC também explora uma abordagem de *Multiple Constant Multiplication* (MCM) sempre que possível.

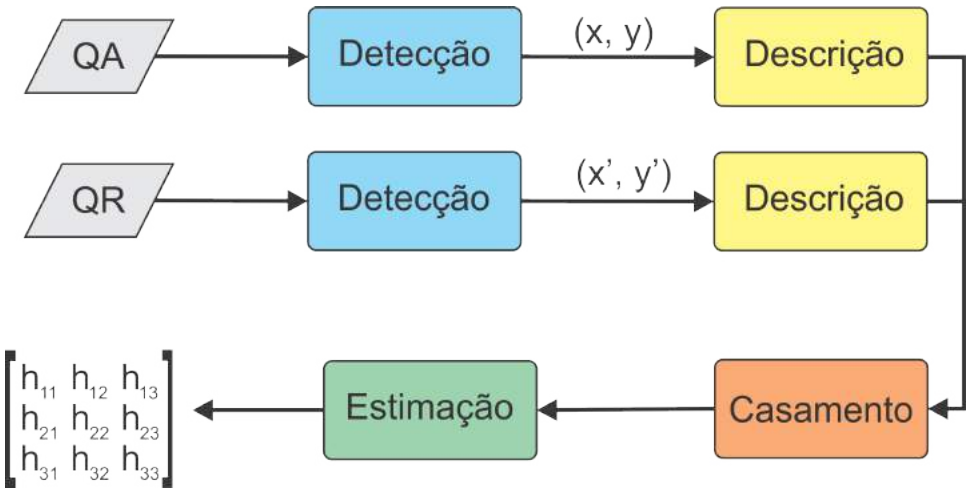

Figura 32 – Fluxograma da arquitetura da compensação de movimento distorcido global.

7.2.1 Etapas do Processamento da Compensação de Movimento Distorcido Global

A primeira etapa da arquitetura desenvolvida para a GWMC - etapa de Detecção - é aplicada tanto no quadro de referência quanto no quadro atual a ser codificado. Esta etapa consiste na varredura de todos os píxeis de ambos os quadros, com o intuito de definir se o pixel em questão é uma quina ou não e, para isso, o AV1 utiliza do algoritmo FAST. O GMWC do AV1 suporta no máximo 4096 pontos (quinas) em um quadro, independentemente do tamanho do quadro. Portanto, assim que o GWMC encontrar 4096 pontos (quinas) ele para a execução e passa para os próximos passos, mesmo ainda não percorrendo todos os píxeis de um quadro.

A implementação em hardware do algoritmo FAST requer duas etapas principais: a detecção de pontos de interesse (calcula a pontuação de canto do candidato atual (pixel)); e a validação de canto (quina). A detecção de pontos de interesse é realizada examinando cada pixel na imagem, identificando sua intensidade com a intensidade dos píxeis vizinhos. Já a validação compara se um pixel é significativamente mais brilhante ou mais escuro do que os píxeis vizinhos. Para isso, o FAST usa o círculo *Bresenham* de 16 píxeis para classificar se um pixel candidato P é uma quina ou não. Cada pixel é identificado usando um número de 1 a 16, e se um conjunto de N

píxeis contidos no círculo for classificado como mais brilhante ou mais escuro do que o candidato P, então P é classificado como um canto. É importante ressaltar que para a GWMC do AV1 $N >= 12$. A Figura 33 demostra o uso do círculo *Bresenham*.

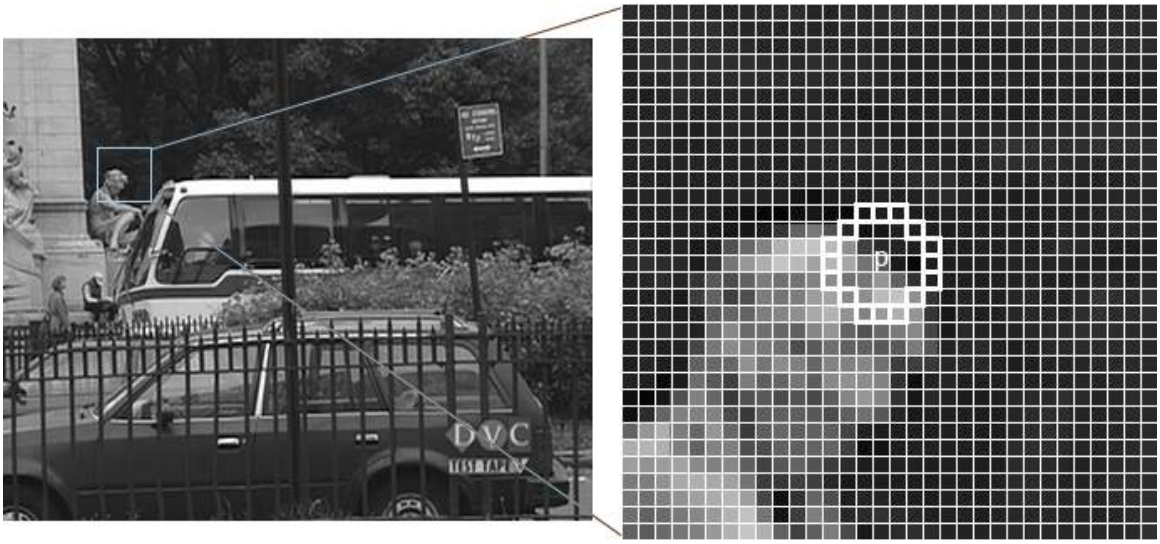

Figura 33 – Círculo de Bresenham utilizado no algoritmo FAST. Adaptado de (ROSTEN; DRUMMOND, 2006).

Na etapa de detecção de pontos de interesse é definida a Pontuação de Quina, que é calculada a partir da Equação 20 e é definida como a soma das diferenças absolutas entre a intensidade do pixel central e as intensidades dos píxeis no círculo de *Bresenham*.

$$V = \max(\sum_{x \in E} |I_{p \to x} - I_p| - t, \sum_{x \in B} |I_p - I_{p \to x}| - t) \tag{20}$$

onde t é o *threshold*, que no AV1 é definido com o tamanho fixo de 18, $I_{p \to x}$ é a intensidade do pixel do círculo e I_p é a intensidade do pixel central.

O hardware do algoritmo FAST é apresentado na Figura 34, o qual inicialmente é alimentado por um bloco de tamanho 7 x 7, tamanho necessário para processar o círculo *Bresenham*, e o *threshold*. Após obter os dados dos píxeis, é calculado o nível de limite (*thresholder*) conforme o valor do pixel central (limite escuro e limite brilhante). Então, o valor de cada pixel no círculo de *Bresenham* com o pixel central P, em relação ao nível de limite para determinar se esses píxeis são ou não mais escuros (E) ou mais brilhantes (B) do que o pixel central. Em paralelo acontece a execução da pontuação de quina. No fim, é realizado um teste de continuidade e se a soma de E ou B forem maiores ou iguais a 12, o pixel central P é considerado uma quina. Todos os pontos de quinas encontrados nessa etapa são armazenados em um banco de registradores, para que possam ser manipulados nas demais etapas.

A segunda e a terceira etapa acontecem basicamente ao mesmo tempo no AV1, e

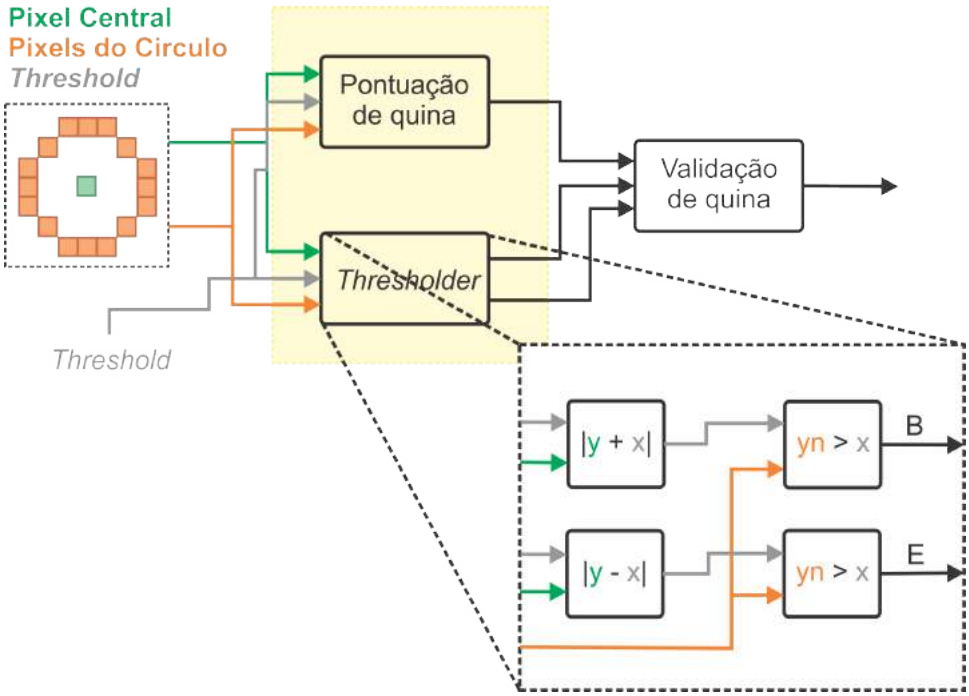

Figura 34 – Arquitetura da etapa de Detecção usada na GWMC.

elas são denominadas de: Descrição e Casamento, conforme apresentado na Figura 32. Já a Figura 35 apresenta o fluxograma detalhado dessas duas etapas. Sendo que, é considerada cada coordenada (x, y) de uma quina (no quadro atual), definida pela etapa de descrição, e é definido um raio de busca (no quadro de referência). Então, todos os pontos (quinas) do quadro de referência contidos dentro do raio de busca são percorridos e, para cada ponto, é realizado o cálculo de correlação cruzada. Assim, ao final, o par de pontos com a maior correlação até o momento é armazenado. Esse fluxo é repetido para todas as quinas do quadro atual.

O raio de busca, definido na etapa de descrição, é definido por $thresh(largura, altura)/16$, ou seja, seu tamanho vai depender do tamanho do quadro que está sendo processado. Por exemplo, um quadro com tamanho 1920 x 1080 (Full HD)) está sendo processado, o raio de busca será definido como 120 x 68, ou se o quadro for 3840 x 2160 (UHD 4K), seu raio de busca será de 240 x 135. O ponto central dessa área de busca no quadro de referência é o mesmo ponto da quina avaliada no quadro atual.

Com o raio de busca definido, o algoritmo de força bruta percorre todos os pontos definidos como quina no quadro de referência, e assim calcula a correlação cruzada entre cada ponto. A fórmula usada para a correlação cruzada na etapa de GWMC do codificador de vídeo AV1 é a definida em (21):

$$C(x, y) = \sum_{i=0}^{N} \frac{A_i * R_i}{\sqrt{A_i * A_i} * \sqrt{R_i * R_i}} \tag{21}$$

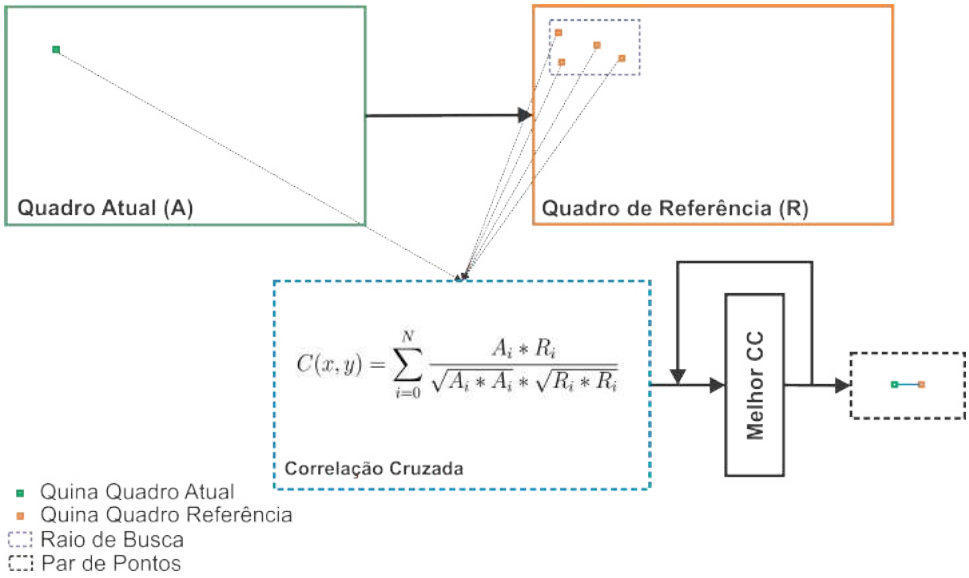

Figura 35 – Arquitetura da etapa de Descrição e Casamento usada na GWMC.

onde $C(x, y)$ é o valor da correlação cruzada para a posição (x, y), A e R é a quina no quadro atual e a quina no quadro de referência respetivamente, com suas devidas posições (x, y).

A Equação 21 calcula a correlação cruzada entre as imagens A e R em diferentes deslocamentos (x, y) (quadro de referência). Para cada deslocamento (x, y), o valor da correlação cruzada $C(x, y)$ é calculado e o deslocamento com o valor máximo de $C(x, y)$ é escolhido como o melhor casamento para a quina avaliada.

Finalmente, após as etapas anteriores, é possível realizar a estimação propriamente dita da GWMC. Para isso, o AV1 utiliza o algoritmo RANSAC. O RANSAC é um algoritmo utilizado para estimar parâmetros de um modelo matemático a partir de um conjunto de dados que contém *outliers* (pontos que não seguem o padrão dos demais). O objetivo é encontrar o melhor conjunto de dados que se ajuste ao modelo, ignorando os *outliers*.

O RANSAC seleciona aleatoriamente um número mínimo de pontos (quinas) necessários (três pontos para movimentos afim e dois pontos para rotação), para estimar os parâmetros do modelo que descreve o movimento global presente entre os quadros selecionados. Portanto, a matriz resultante, apresentada na Equação 17, aplica para cada (x_i, y_i), na lista de correspondcência, seu valor correspondente e gera o resultado (x'_i, y'_i). Dessa forma é possível calcular a distância euclidiana (erro) em relação ao par correspondente do quadro de referência. Essa etapa é repetida 20 vezes. O GWMC define como matriz resultante a matriz que obtiver o menor erro entre todos.

A Figura 36 apresenta o esquema usado nesta etapa, em que é possível observar que a arquitetura contém 3 etapas, sendo elas: Unidade Geradora de Pontos Aleatórios (UGPA); Unidade de Cálculo do Modelo (UCM) e Unidade de Seleção do Melhor

Modelo (USMM).

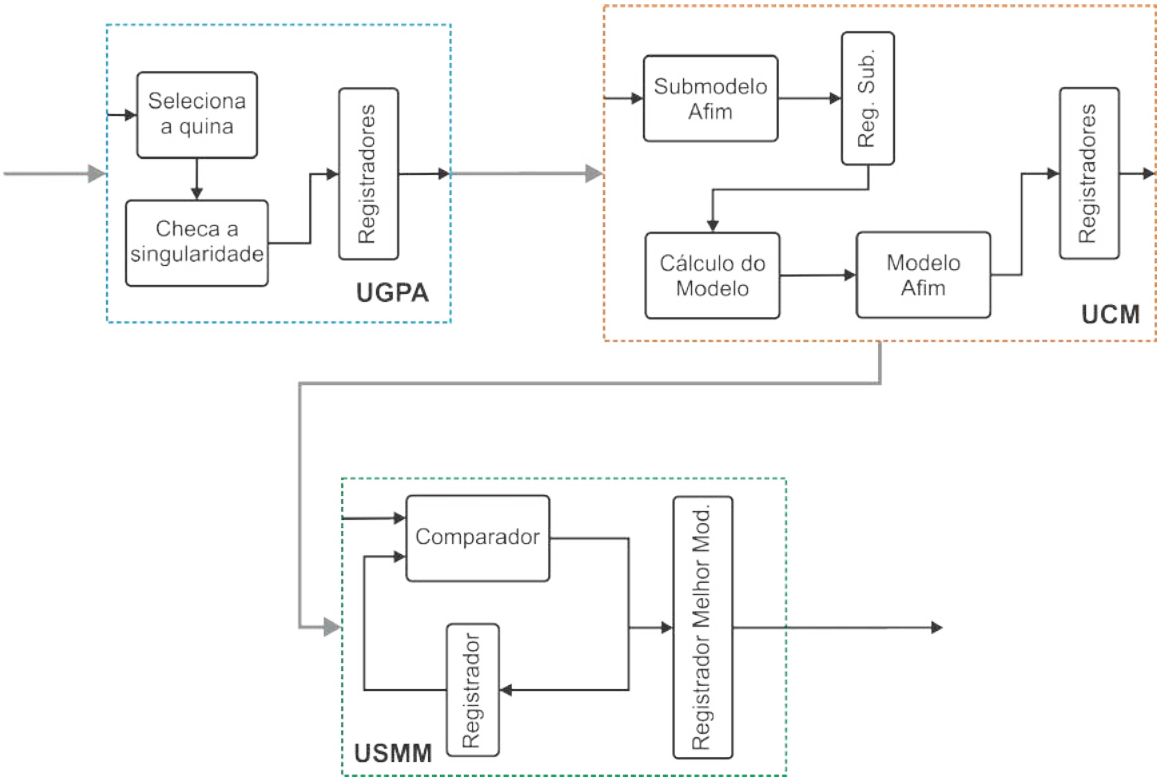

Figura 36 – Arquitetura de Hardware da etapa de Estimação usada na GWMC.

A UGPA tem a responsabilidade de selecionar as quinas no quadro atual de maneira aleatória e única. Em seguida, as quinas e suas correspondências (x, y) são lidas sequencialmente e armazenadas temporariamente em uma cadeia de registradores.

Com as quinas já definidas na etapa anterior, a UCM lê os dados que estavam nos registradores e inicia a operação de cálculo do modelo. A UCM utiliza as quinas e suas correspondências para realizar o cálculo do submodelo afim, conforme a Equação (18). Ao solucionar o problema, o modelo afim é calculado usando os recursos traduzidos e seus respetivos pontos de correspondência. Dessa forma, por fim, o erro é gerado para aquele conjunto de quinas, e esse erro é armazenado em um banco de registradores para que possa ser avaliado na próxima etapa.

Por último, a USMM tem a responsabilidade de selecionar o melhor modelo que é um modelo com um número máximo de pontos de características consistentes, ou seja, o modelo que tenha o menor erro entre todos. A unidade inclui um somador, um comparador e alguns registradores internos. Esses registradores internos armazenam os coeficientes e o número de pontos de características consistentes para o melhor modelo. A USMM faz a comparação entre o erro atual e o melhor erro encontrado até o momento e, assim, o modelo que apresentar o menor erro é atualizado com base no resultado da comparação. Depois de 20 rodadas, o melhor resultado é definido.

7.2.2 Análise e Resultados

A arquitetura para a GWMC usou a técnica de MCM, como já destacado, visando a redução no hardware na dissipação de potência.

A arquitetura projetada é capaz de operar a uma frequência de 259,2 MHZ e atingir uma taxa de processamento para vídeos UHD 4K a 30 fps. Os resultados são apresentados na Tabela 14.

Tabela 14 – Resultados de síntese das arquiteturas projetadas para a GWMC do AV1.

Arquitetura	Frequência	Gates (K)	Power (mW)	Thougpuht
GWMC	259,2 MHz	832,66	657,68	2160@30fps

Como não há outros trabalhos na literatura descrevendo projetos de hardware dedicados para a GWMC do AV1, não é possível fazer uma comparação justa de nosso trabalho com outros trabalhos.

8 CONCLUSÕES

Esta tese investigou e apresentou soluções em hardware dedicado para etapas do modo de predição inter quadros, do codificador AV1. A relevância no desenvolvimento de hardware no campo da codificação de vídeo aumenta com o tempo, pois o número de vídeos usados no dia a dia das pessoas aumentou significativamente — isso é evidente em diferentes aplicativos e dispositivos.

O codificador AV1 é um codec relativamente novo, e teve sua primeira versão lançada em 2018, assim, o desenvolvimento de hardware dedicado se tornou de suma importância. Portanto, está presente tese apresentou, ao longo de suas páginas, soluções efetivas em hardware dedicado para os algumas das ferramentas deste formato de vídeo, o qual contribui para a promoção do uso do AV1 e suas gerações futuras.

Foram desenvolvidas soluções para a Estimação de Movimento Fracionária, para a Compensação de Movimento Convencional e para a Compensação de Movimento Distorcido, da predição inter quadros do codificador AV1. Todas as soluções presentes neste texto, sempre que possível, buscaram explorar paralelismos, evitar o uso de multiplicadores e compartilhar subexpressões. Além disso, na FME, também foram exploradas técnicas de computação aproximada. Em todos os casos, o objetivo foi projetar hardwares de menor custo e menor consumo de potência.

O hardware de alto desempenho para o interpolador de subamostras da MC do codificador AV1 foi subdivido em duas etapas: a primeira, que visa os filtros usados na interpolação da MC e que utiliza apenas somadores/subtratores (aplicando MCM); e a segunda etapa, que apresenta o interpolador propriamente dito, para o qual foram projetadas três versões arquiteturais diferentes, podendo processar vídeos de diferentes resoluções e taxas de quadro, entre 4K UHD a 30 fps e 8K UHD a 30 fps.

O projeto de hardware da FME do AV1 também foi subdivido em etapas. A primeira etapa visou a implementação dos filtros usados na FME, explorando MCM e, também de computação aproximada para a simplificação dos filtros. As soluções propostas tiveram diferentes resultados de eficiência de codificação, variando de -0,07% a 2,74% de BD-BR para diferentes resoluções. Além disso, também foram gerados, diferentes resultados de área e potência. A segunda etapa visou a implementação

do interpolador usado na FME, o qual explorou diferentes níveis de paralelismo para diferentes filtros implementados na etapa anterior. Esse nível de paralelismo possibilitou que o interpolador fosse capaz de processas vídeos entre 4K UHD a 30fps e 8K UHD a 60fps. A terceira e última etapa visou a implementação do processo da FME completa, com foco em blocos 4 x 4, com todo o processo da interpolação e de busca do melhor casamento. Nessa etapa, foram explorados diferentes níveis de paralelismos para diferentes filtros, que possibilitou atingir diferentes taxas de processamento, podendo processar vídeos Full HD a 60 fps até 8K UHD a 30 fps. Consequentemente, a área e a potência também variaram entre estas implementações.

Por fim, foi apresentado o hardware projetado para a WMC, que foi dividido em duas etapas: uma arquitetura para a LWMC e outra para a GWMC. A arquitetura da LWMC, explorou MCM na implementação dos seus filtros. Foram projetadas duas arquiteturas de interpolação, visando diferentes níveis de paralelismo, o qual possibilitou atingir diferentes taxas de processamento, podendo processar vídeos entre 4K UHD a 60 fps e 8K UHD a 60 fps, com diferentes áreas e potências. A arquitetura da GWMC foi desenvolvida em uma solução única, também explorando MCM, e que é, capaz de processar vídeos UHD 4K a 30 fps.

Até o final desta tese, todas as arquiteturas apresentadas até aqui são as primeiras arquiteturas de hardware desenvolvidas com foco nas ferramentas da predição inter quadros do codificador AV1 encontrados na literatura, o que é um importante sinal da relevância desta contribuição. Ao decorrer da tese obtivemos três publicações, uma com foco na MC (Domanski et al., 2019) e duas com foco na FME (DOMANSKI et al., 2021) e (DOMANSKI et al., 2023a), além de contribuições em outros trabalhos com foco no codificador AV1. Os resultados obtidos com a GWMC serão encaminhados para publicação em breve, e um artigo contendo os resultados da LWMC foi submetido e está em avaliação. Todas as publicações resultantes de contribuições para esta tese estão listadas no Apêndice A.

Ao final do trabalho foi possível demonstrar que a hipótese inicial deste trabalho é totalmente válida para as ferramentas investigadas neste trabalho, ou seja, **que o projeto de hardware dedicado para as ferramentas da predição inter quadros do AV1 explorando paralelismo e o uso de computação aproximada é uma opção viável para atingir as taxas de processamento necessárias para processar vídeos de ultra alta resolução em tempo real e com eficiência em termos de área e potência**, pelo menos para as ferramentas de codificação investigadas nesta tese.

Trabalhos futuros estão planejadas a implementação das ferramentas restantes da predição inter quadros do AV1 (como a OBMC e a ME inteira) e, a seguir, a integração das arquiteturas desenvolvidas nesta tese com estas novas arquiteturas, para assim ser possível gerar uma arquitetura completa para a predição inter quadros do codificador AV1.

REFERÊNCIAS

AFONSO, V.; AGOSTINI, L.; FRANCO, D. **Desenvolvimento de uma arquitetura para estimação de movimento fracionária segundo o padrão emergente HEVC**. 2013. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal de Pelotas, Pelotas, Brasil.

AFONSO, V.; AGOSTINI, L.; SUSIN, A. **High-Throughput Dedicated Hardware Design Targeting the 3DHEVC-Prediction Coding Tools**. 2019. Tese (Doutorado em Ciência da Computação) — Universidade Federal do Rio Grande do Sul, Porto Alegre, Brasil.

AGOSTINI, L.; SILVA, I. da; BAMPI, S. **Desenvolvimento de Arquiteturas de Alto Desempenho Dedicadas à Compressão de Vídeo Segundo o Padrão H.264/AVC**. 2007. Tese (Doutorado em Ciência da Computação) — UFRGS.

AOMEDIA. **Alliance for Open Media - An Alliance of Global Media Innovators**. Disponível em: <http://aomedia.org/>. Acesso em: 2020-11-19.

AOMEDIA. **AV1 Bitstream e Decoding Process Specification**. https://aomedia.googlesource.com/aom/+/refs/tags/v2.0.0.

AOMEDIA. **Aliance for Open Media**. Disponível em: https://aomedia.org/.

BJONTEGAARD, G. Improvements of the BD-PSNR model. ITU-T SC16/Q6. In: VCEG MEETING, BERLIN, GERMANY, DOC. VCEG-AI11, 35., 2008. **Anais...** [S.l.: s.n.], 2008.

BJONTEGARD, G. Calculation of Average PSNR Differences between RD-curves. **ITU-T VCEG-M33**, [S.l.], 2001.

Bjøntegaard, G.; Davies, T.; Fuldseth, A.; Midtskogen, S. The Thor Video Codec. In: DATA COMPRESSION CONFERENCE (DCC), 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p.476–485.

BORGES, A. **TAnálise de Complexidade do Software de Referência AV1 Codec Library para a Codificação de Vídeos Segundo o Formato AV1**. 2020. Trabalho de TI — Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas.

BULL, D.; ZHANG, F. **Intelligent Image and Video Compression**: Communicating Pictures. [S.l.]: Elsevier Science, 2021.

BéGAINT, J.; GALPIN, F.; GUILLOTEL, P.; GUILLEMOT, C. Region-based models for motion compensation in video compression. In: PICTURE CODING SYMPOSIUM (PCS), 2018., 2018. **Anais...** [S.l.: s.n.], 2018. p.154–158.

CHEN, J.; CRANTON, W.; FIHN, M. **Handbook of Visual Display Technology**. [S.l.]: Springer International Publishing, 2016.

Chen, X. et al. A Conditional Bayesian Block Structure Inference Model for Optimized AV1 Encoding. In: IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA AND EXPO (ICME), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.1270–1275.

Chen, Y. et al. An Overview of Core Coding Tools in the AV1 Video Codec. In: PICTURE CODING SYMPOSIUM (PCS), 2018., 2018. **Anais...** [S.l.: s.n.], 2018. p.41–45.

Chiang, C. et al. Adaptive interpolation filter scheme in AV1. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2017., 2017. **Anais...** [S.l.: s.n.], 2017. p.934–937.

CROFT, H.; FALCONER, K.; GUY, R. **Unsolved Problems in Geometry**: Unsolved Problems in Intuitive Mathematics. [S.l.]: Springer New York, 2012. (Problem Books in Mathematics).

da Silva, R.; Siqueira, I.; Grellert, M. Approximate Interpolation Filters for the Fractional Motion Estimation in HEVC Encoders and their VLSI Design. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN (SBCCI), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.1–6.

Domanski, R. et al. High-Throughput Multifilter Interpolation Architecture for AV1 Motion Compensation. **IEEE Transactions on Circuits and Systems II: Express Briefs**, [S.l.], v.66, n.5, p.883–887, May 2019.

DOMANSKI, R. et al. Low-Power and High-Throughput Approximated Architecture for AV1 FME Interpolation. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS (ISCAS), 2021., 2021. **Anais...** [S.l.: s.n.], 2021. p.1–5.

DOMANSKI, R. et al. High-Throughput and Multiplierless Hardware Design for the AV1 Fractional Motion Estimation. In: LATIN AMERICAN SYMPOSIUM ON CIRCUITS SYSTEMS (LASCAS), 2023., 2023. **Anais...** [S.l.: s.n.], 2023.

DOMANSKI, R. et al. High-Throughput and Multiplierless Hardware Design for the Av1 Local Warped MC Interpolation. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2023., 2023. **Anais...** [S.l.: s.n.], 2023.

El-Harouni, W. et al. Embracing approximate computing for energy-efficient motion estimation in high efficiency video coding. In: DESIGN, AUTOMATION TEST IN EUROPE CONFERENCE EXHIBITION (DATE), 2017, 2017. **Anais...** [S.l.: s.n.], 2017. p.1384–1389.

FASTLY. **How COVID-19 is affecting internet performance**. Disponível em: <https://www.fastly.com/blog/how-covid-19-is-affecting-internet-performance>. Acesso em: 2022-03-30.

Filho, V. R. et al. Standalone Rate-Distortion FME Architecture. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN (SBCCI), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.1–6.

FISCHLER, M. A.; BOLLES, R. C. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. **Commun. ACM**, New York, NY, USA, v.24, n.6, p.381–395, June 1981.

FRAUNHOFER. **Versatile Video Coding (VVC - officially approved as ITU-T H.266 | 23090-3)**. Disponível em: <https://www.hhi.fraunhofer.de/en/departments/vca/technologies-and-solutions/h266-vvc.html>. Acesso em: 2021-03-04.

Freitas, D. et al. Hardware Architecture for the Regular Interpolation Filter of the AV1 Video Coding Standard. In: EUROPEAN SIGNAL PROCESSING CONFERENCE (EUSIPCO), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.560–564.

GOEBEL, J. W. **SOLUÇÃO ARQUITETURAL PARA A DECODIFICAÇÃO INTRA-QUADRO DO PADRÃO DE CODIFICAÇÃO AV1**. 2019. Dissertação (Mestrado em Ciência da Computação) — UFPEL.

GROUP, N. W. **Video Codec Testing and Quality Measurement**. Accessed: 2021-11-17. Disponível em: <https://tools.ietf.org/html/draft-ietf-netvc-testing-09>.

Guo, Z.; Zhou, D.; Goto, S. An optimized MC interpolation architecture for HEVC. In: IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING (ICASSP), 2012., 2012. **Anais...** [S.l.: s.n.], 2012. p.1117–1120.

Han, J.; Chiang, C.; Xu, Y. A level-map approach to transform coefficient coding. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2017., 2017. **Anais...** [S.l.: s.n.], 2017. p.3245–3249.

HAN, J. et al. **A Technical Overview of AV1**.

HARTLEY, R. I.; ZISSERMAN, A. **Multiple View Geometry in Computer Vision**. 2.ed. [S.l.]: Cambridge University Press, ISBN: 0521540518, 2004.

HUITEMA, C.; TURLETTI, T. **RTP Payload Format for H.261 Video Streams**. [S.l.]: RFC Editor, 1996. n.2032. (Request for Comments). RFC 2032. Disponível em: <https://rfc-editor.org/rfc/rfc2032.txt>.

INTERDIGITAL. **The Sustainable Future of Video Entertainment**. Disponível em: <https://www.interdigital.com/white_papers/the-sustainable-future-of-video-entertainment>. Acesso em: 2022-03-30.

ISO/IEC-JCT1/SC29/WG11. **High Efficiency Video Coding (HEVC) text specification draft 10.** [S.l.: s.n.], 2013.

KOLODZIEJSKI, W.; DOMANSKI, R.; AGOSTINI, L. **Soluções para Redução do Tempo de Execução da Ferramenta de Compensação de Movimento Distorcido Global do Codificador AV1**.

Layek, M. A. et al. Performance analysis of H.264, H.265, VP9 and AV1 video encoders. In: ASIA-PACIFIC NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM (APNOMS), 2017., 2017. **Anais...** [S.l.: s.n.], 2017. p.322–325.

León, J. S.; Cárdenas, C. S.; Castillo, E. V. A high parallel HEVC Fractional Motion Estimation architecture. In: IEEE ANDESCON, 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p.1–4.

Maich, H. et al. A multi-standard interpolation hardware solution for H.264 and HEVC. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2015., 2015. **Anais...** [S.l.: s.n.], 2015. p.2910–2914.

Mukherjee, D. et al. A Technical Overview of VP9 – The Latest Open-Source Video Codec. In: SMPTE 2013 ANNUAL TECHNICAL CONFERENCE EXHIBITION, 2013. **Anais...** [S.l.: s.n.], 2013. p.1–17.

OPEN MEDIA, A. for. **Local Warped Motion Compensation**. Disponível em: <https://github.com/AOMediaCodec/SVT-AV1/blob/master/Docs/Appendix-Local-Warped-Motion.md>. Acesso em: 2022-04-25.

Paim, G. et al. High-throughput and memory-aware hardware of a sub-pixel interpolator for multiple video coding standards. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p.2162–2166.

PAIM, G. et al. An efficient sub-sample interpolator hardware for VP9-10 standards. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p.2167–2171.

PALAU, R.; PORTO, M.; CORRêA, G.; AGOSTINI, L. **Investigação de Soluções em Hardware para os Filtros de Laço do Decodificador AV1**. 2022. Tese (Doutorado em Ciência da Computação) — Universidade Federal de Pelotas, Pelotas, Brasil.

Parker, S. et al. Global and locally adaptive warped motion compensation in video compression. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2017., 2017. **Anais...** [S.l.: s.n.], 2017. p.275–279.

Parker, S. et al. Global and locally adaptive warped motion compensation in video compression. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2017., 2017. **Anais...** [S.l.: s.n.], 2017. p.275–279.

Penny, W. et al. Real-time architecture for HEVC motion compensation sample interpolator for UHD videos. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN (SBCCI), 2015., 2015. **Anais...** [S.l.: s.n.], 2015. p.1–6.

Penny, W. et al. Power-Efficient and Memory-Aware Approximate Hardware Design for HEVC FME Interpolator. In: IEEE INTERNATIONAL CONFERENCE ON ELECTRONICS, CIRCUITS AND SYSTEMS (ICECS), 2018., 2018. **Anais...** [S.l.: s.n.], 2018. p.237–240.

Penny, W. et al. Low-Power and Memory-Aware Approximate Hardware Architecture for Fractional Motion Estimation Interpolation on HEVC. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS (ISCAS), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.1–5.

PENNY, W. I.; PALOMINO, D. M.; ZATT, M. S. P. andBruno. **Energy Efficient NoC Based Systems for Real Time Multimedia Applications using Approximate Computing**. 2021. Tese (Doutorado em Ciência da Computação) — UFPel.

PORTO, R. E. C. et al. Fast and Energy-Efficient Approximate Motion Estimation Architecture for Real-Time 4K UHD Processing. **Journal of Real-Time Image Processing**, [S.l.], Aug. 2020.

PRABAKARAN, B.; EL-HAROUNI, W.; REHMAN, S.; SHAFIQUE, M. **Approximate Multi-Accelerator Tiled Architecture for Energy-Efficient Motion Estimation**: Methodologies and CAD. [S.l.: s.n.], 2019. p.249–268.

RICHARDSON, I. E. **Video Codec Design**: Developing Image and Video Compression Systems. USA: John Wiley; Sons, Inc., 2002.

RICHARDSON, I. E. G. **H.264 and MPEG-4 Video Compression**: Video Coding for Next-Generation Multimedia. [S.l.]: Wiley, 2003. 1-281p.

RIVAZ, P.; HAUGHTON, J. **AV1 Bitstream & Decoding Process Specification. Version**: 1.0.0. Copyright 2018, The Alliance For Open Media. [S.l.: s.n.], 2019. 677 p.

ROSTEN, E.; DRUMMOND, T. Machine Learning for High-Speed Corner Detection. In: COMPUTER VISION – ECCV 2006, 2006, Berlin, Heidelberg. **Anais...** Springer Berlin Heidelberg, 2006. p.430–443.

Seidel, I.; Rodrigues Filho, V.; Agostini, L.; Güntzel, J. L. Coding- and Energy-Efficient FME Hardware Design. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS (ISCAS), 2018., 2018. **Anais...** [S.l.: s.n.], 2018. p.1–5.

SHI, Y.; SUN, H. **Image and video compression for multimedia engineering**: Fundamentals, algorithms, and standards, second edition. [S.l.]: CRC Press, 2017. Publisher Copyright: © 2008 by Taylor & Francis Group, LLC.

SOLOMON, C.; BRECKON, T. **Fundamentals of Digital Image Processing**: A Practical Approach with Examples in Matlab. [S.l.]: Wiley-Blackwell, 2010.

STATISTA. **Distribution of global monthly mobile data volume as of January 2021**. Disponível em: https://www.statista.com/statistics/383715/global-mobile-data-traffic-share/.

SZELISKI, R. **Computer Vision**: Algorithms and Applications. 1st.ed. Berlin, Heidelberg: Springer-Verlag, 2010.

Topiwala, P.; Dai, W.; Krishnan, M. HDR videocompression with VPX. In: DIGITAL MEDIA INDUSTRY ACADEMIC FORUM (DMIAF), 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p.65–70.

TRUONG; NHAT, M. T.; KIM, S. A Review on Image Feature Detection and Description. **Proceedings of the Korea Information Processing Society Conference**, [S.l.], p.667–680, 2016.

Vanne, J.; Aho, E.; Hamalainen, T. D.; Kuusilinna, K. A High-Performance Sum of Absolute Difference Implementation for Motion Estimation. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.16, n.7, p.876–883, 2006.

VORONENKO, Y.; PüSCHEL, M. Multiplierless multiple constant multiplication. **ACM Transactions on Algorithms**, [S.l.], v.3, 05 2007.

XIPH.ORG. **Daala Video Compression**. Disponível em: https://xiph.org/daala.

YOUTUBE. **Youtube Statistics.** Disponível em: https://www.youtube.com/intl/pt-BR/yt/about/press/.

ZABROVSKIY, A.; FELDMANN, C.; TIMMERER, C. Multi-codec DASH Dataset. In: ACM MULTIMEDIA SYSTEMS CONFERENCE, 9., 2018, New York, NY, USA. **Proceedings...** ACM, 2018. p.438–443. (MMSys '18).

ZEN, M. of. **AV1 Wiki**. Disponível em: <https://github.com/master-of-zen/AV1>. Acesso em: 2022-10-05.

ZHAO, F.; HUANG, Q.; GAO, W. Image Matching by Normalized Cross-Correlation. In: IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS SPEECH AND SIGNAL PROCESSING PROCEEDINGS, 2006., 2006. **Anais...** [S.l.: s.n.], 2006. v.2, p.II–II.

ZHAO, X. et al. NSST: Non-separable secondary transforms for next generation video coding. **2016 Picture Coding Symposium (PCS)**, [S.l.], p.1–5, 2016.

ZWILLINGER, D. **CRC standard mathematical tables and formulae**. 32nd ed..ed. Boca Raton, FL: CRC Press, 2012. 360p.

Apêndices

APÊNDICE A – Lista de publicações realizadas durante esta Tese

A.1 Artigos Publicados em Revistas

1. Domanski, R.; Goebel, J.; Penny, W.; Zatt, B.; Porto; M.; Agostini, L. High-Throughput Multifilter Interpolation Architecture for AV1 Motion Compensation. IEEE Transactions on Circuits and Systems II, v.66, n.5, p.883–887, May 2019.

2. Palau, R.; Silveira, B.; Domanski, R.; Loose, M.; Cerveira, A.; Sampaio, F.; Palomino, D.; Porto, M.; Corrêa, G.; Agostini, L. Modern Video Coding: Methods, Challenges and Systems. Journal of Integrated Circuits and Systems, vol. 16, n. 2, 2021.

3. Kolodziejski, W; Domanski, R.; Zatt, B.; Porto; M.; Agostini, L. Ultra-High Definition AV1 FME Interpolation Architectures Exploring Approximate Computing. JICS. Journal of Integrated Circuits and Systems, 2022.

A.2 Artigos Publicados em Conferências

1. Domanski, R.; Kolodziejski, W; Zatt, B.; Correa, G.; Porto; M.; Agostini, L. Low-Power and High-Throughput Approximated Architecture for AV1 FME Interpolation. In: 2021 IEEE International Symposium on Circuits and Systems (ISCAS), 2021.

2. Domanski, R.; Kolodziejski, W; Penny, W.; Zatt, B.; Correa, G.; Porto; M.; Agostini, L. High-Throughput and Multiplierless Hardware Design for the AV1 Fractional Motion Estimation. In: Latin American Symposium on Circuits and Systems (LASCAS), 2023.

3. Domanski, R.; Kolodziejski, W; Penny, W.; Zatt, B.; Correa, G.; Porto; M.; Agostini, L. High-Throughput and Multiplierless Hardware Design for the Av1 Local Warped MC Interpolation. In: IEEE International Conference on Image Processing (ICIP), 2023.

4. Kolodziejski, W; Domanski, R.; Zatt, B.; Correa, G.; Porto; M.; Agostini, L. Memory-Aware, Low-Power and High-Throughput AV1 FME Interpolation Architecture. In: Microelectronics Students Forum, 2021.