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

Exame de Qualificação

Projeto de Memória e Hardware Dedicados para a Predição Interquadros do Codificador VVC: Explorando Armazenamento e Computação Imprecisos

Murilo Roschildt Perleberg

Pelotas, 2022

Murilo Roschildt Perleberg

Projeto de Memória e Hardware Dedicados para a Predição Interquadros do Codificador VVC: Explorando Armazenamento e Computação Imprecisos

Exame de Qualificação apresentado 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. Marcelo Schiavon Porto

Coorientadores: Prof. Dr. Luciano Volcan Agostini

Prof. Dr. Vladimir Afonso

Pelotas, 2022

RESUMO

PERLEBERG, Murilo Roschildt. **Projeto de Memória e Hardware Dedicados para a Predição Interquadros do Codificador VVC: Explorando Armazenamento e Computação Imprecisos**. Orientador: Marcelo Schiavon Porto. 2022. 99 f. Exame de Qualificação (Doutorado em Ciência da Computação) – Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2022.

Os vídeos digitais estão se tornando cada vez mais presentes no cotidiano das pessoas, tendo papel importante no entretimento, educação, trabalho e até mesmo na saúde. Os padrões de codificação de vídeo mais recentes empregam ferramentas que permitem melhorar a eficiência de codificação dos vídeos digitais, resultando em um grande aumento na largura de banda com a memória e no custo computacional, quando comparado com os padrões de codificação anteriores. Devido a esse alto custo computacional, as ferramentas de Estimação de Movimento e Estimação de Movimento *Affine*, da predição interquadros, demandam um *hardware* dedicado para garantir o processamento de vídeos em alta resolução em tempo real, atendendo também às restrições quanto ao consumo de energia, especialmente para aplicações em dispositivos alimentados por bateria, como *smartphones*. Neste contexto, a comunicação com a memória é a responsável pela maior parte do consumo de energia. As estratégias das soluções arquiteturais da literatura para ferramentas de predição interquadros permitem obter uma redução nos acessos à memória. Porém, nenhum dos trabalhos foca o projeto arquitetural de forma a obter o menor consumo do circuito de memória em si, podendo levar a decisões sub-ótimas de projeto, ou ainda a um consumo de energia excessivo por parte da memória que não foi previamente avaliada. Assim, há espaço para o projeto de um sistema de predição interquadros priorizando a memória, de forma que seja possível minimizar o consumo de energia do sistema completo. Para tanto, o foco deste trabalho se dará em avaliar diferentes organizações de memória que possam ser integradas na predição interquadros, assim como avaliar estratégias para o *hardware* dedicado que resultem no menor consumo de energia do circuito de memória. Também serão exploradas técnicas de armazenamento e operadores imprecisos, que permitem reduzir o consumo de energia das operações da memória e também da unidade de processamento. Ao final do trabalho, espera-se obter um sistema completo para a predição interquadros do padrão VVC, com um baixo consumo de energia, e que consiga atender o desempenho necessário para processamento de vídeos de Ultra-Alta resolução (UHD) em tempo real e com uma alta eficiência de codificação.

Palavras-chave: Hardware focado em memória. Predição interquadros. Affine ME. Armazenamento Impreciso. Operadores Imprecisos.

LISTA DE ABREVIATURAS E SIGLAS

ACAA	*Accuracy Configurable Approximate Adder*
ASIC	*Application-Specific Integrated Circuit*
AV1	*AOMedia Video 1*
BL	*Bitline*
CB	*Coding Block*
CTB	*Coding Tree Block*
CTU	*Coding Tree Unit*
CU	*Coding Unit*
DA	*Data Array*
DRAM	*Dynamic Random-Access Memory*
FME	*Fractional Motion Estimation*
FPGA	*Field-Programmable Gate Array*
HEVC	*High Efficiency Video Coding*
HTM	*HEVC Test Model*
IME	*Integer Motion Estimation*
JVET	*Joint Video Experts Team*
LOA	*Lower-part Or Adder*
LUT	*LookUp Table*
MC	*Motion Compensation*
ME	*Motion Estimation*
MLC	*Multi-Layer Cells*
MPEG	*Moving Picture Experts Group*
MTJ	*Magnetic Tunnel Junction*
MTT	*Multi-Type Tree*
MVD	*Multi-view Video plus Depth*
MV	*Motion Vectors*

PC-RAM	*Phase Change Random-Access Memory*
PC	Ponto de Controle
PSNR	*Peak Signal-to-Noise Ratio*
PB	*Prediction Block*
PU	*Prediction Unit*
qps	Quadros por segundo
QT+MTT	*Quaternary Tree Plus Multi-Type Tree*
QT	*Quaternary Tree*
RAM	*Random-Access Memory*
RCA	*Ripple Carry Adder*
RD	*Rate Distortion*
ReRAM	*Resistive RAM*
SAD	*Sum of Absolute Differences*
SA	*Search Area*
SLC	*Single-Layer Cells*
SL	*Sourceline*
SRAM	*Static Random-Access Memory*
SR	*Search Range*
STT-RAM	*Spin-Transfer Torque Random-Access Memory*
TZS	*Test Zone Search*
UHD	*Ultra-High Definition*
VCEG	*Video Coding Experts Group*
VHDL	*VHSIC Hardware Description Language*
VTM	*VVC Test Model*
VVC	*Versatile Video Coding*
WL	*Wordline*

SUMÁRIO

1 INTRODUÇÃO

A evolução da internet e dos serviços de *streaming* tornou os conteúdos multimídia muito populares nas últimas décadas. Os vídeos digitais são um desses conteúdos multimídia que se tornaram cada vez mais presentes no cotidiano das pessoas, tanto pela facilidade em encontrar vídeos de entretenimento, como também pela capacidade que os dispositivos móveis atuais possuem de capturar e compartilhar momentos e recordações.

Os vídeos digitais demandam que uma grande quantidade de dados seja armazenada e transferida para o usuário. Devido a toda essa demanda, o emprego de técnicas avançadas de compressão de vídeos é fundamental para a redução na quantidade de dados a ser armazenada e transmitida para cada vídeo. Mesmo com técnicas de compressão de vídeos, desde 2017, mais de 75% do tráfego mundial da internet já é devido a transmissão de vídeos digitais (CISCO, 2018). Somado a isso, a pandemia da COVID-19 impactou fortemente o tráfego da internet mundial (BITMOVIN, 2022). Seja em busca de entretenimento ou devido ao trabalho e/ou estudos remotos, a internet mundial acabou sobrecarregada pelo uso de serviços de *streaming* e videochamadas. O impacto do tráfego de vídeos na internet mundial foi tão expressivo que serviços como *Netflix* e *Youtube* anunciaram uma redução da qualidade dos vídeos acessados para não sobrecarregar a internet e garantir o acesso aos seus serviços por todos os seus usuários (EXPRESS, 2020).

Com isso, o estudo de técnicas avançadas de compressão de vídeo é de grande importância atualmente para viabilizar o armazenamento e transmissão destes vídeos através da internet. Contudo, o emprego dessas técnicas de compressão de vídeo de alta resolução em tempo real acaba sendo um grande desafio devido ao grande custo computacional dos padrões de codificação de vídeo atuais, especialmente levando em conta o seu emprego em dispositivos portáteis movidos a bateria. Por conta disso, as aplicações e dispositivos com suporte para o processamento de vídeos digitais usualmente requerem um *hardware* dedicado para implementar os codificadores/decodificadores e obter processamento em tempo real ao mesmo tempo que atendam às restrições de consumo de energia.

Dentre os padrões de codificação de vídeo atuais, o padrão *Versatile Video Coding* (VVC) (BROSS et al., 2021a) é considerado o estado-da-arte. O VVC foi desenvolvido pelos especialistas da *Joint Video Experts Team* (JVET), uma parceria entre a *ITU-T Video Coding Experts Group* (VCEG) e a *ISO/IEC Moving Picture Experts Group* (MPEG). O VVC foi finalizado em 2020 e é um dos padrões de codificação mais importantes atualmente. Este padrão visa suceder o padrão *High Efficiency Video Coding* (HEVC), lançado em 2013 (SULLIVAN et al., 2012). De acordo com (BROSS et al., 2021a), o VVC consegue atingir uma redução de 50% na taxa de *bits* de um vídeo, mantendo a mesma qualidade objetiva, quando comparado ao padrão HEVC. Para atingir essa alta eficiência de codificação, diversas ferramentas do HEVC foram aprimoradas, enquanto outras ferramentas novas foram adotadas no fluxo de codificação, o que fez com que o tempo de codificação do *software* de referência do VVC fosse aumentado em torno de oito vezes se comparado com o tempo de codificação do *software* de referência do padrão HEVC (BROSS et al., 2021a).

Grande parte do custo computacional dos codificadores de vídeo atuais ocorre devido a avaliação das ferramentas de predição interquadros para os diversos tamanhos de bloco suportados pelo padrão de codificação. No caso do VVC, cada quadro do vídeo é dividido em blocos chamados Unidades de Codificação em Árvore (*Coding Tree Unit* - CTU), os quais possuem o tamanho máximo de 128×128 amostras. Então, cada CTU é recursivamente dividida em um ou mais Blocos de Codificação (*Coding Block* - CB), nas quais as ferramentas de predição interquadros são aplicadas. Na versão 14.0 do *software VVC Test Model* (VTM) (JVET, 2022), o *software* de referência do VVC, a predição interquadros é realizada para até 27 diferentes tamanhos de CB, o que representa um aumento no custo computacional se comparado com a predição interquadros do padrão HEVC, que suporta 24 tamanhos de CB e o tamanho máximo da CTU de 64×64 amostras (SULLIVAN et al., 2012). O processo de particionamento do VVC será detalhado na Seção 2.1.

As ferramentas de predição interquadros são utilizadas para realizar a predição da CB sendo codificada (CB atual) baseado nas informações de blocos de outros quadros de referência que já foram codificados anteriormente. Dentre as ferramentas de predição interquadros, a Estimação de Movimento (*Motion Estimation* - ME) é uma ferramenta tradicional presente em diversos codificadores de vídeo. A ME é eficiente para representar movimentos translacionais da cena, isso é, a movimentação simples de um bloco no eixo X e Y do quadro, perdendo eficiência para movimentos de rotação, *zoom* ou distorção. Assim, buscando a representação de movimentos mais complexos, os padrões de codificação mais recentes, como o padrão *AOMedia Video 1* (AV1) (HAN et al., 2021) e, no caso deste trabalho, o VVC (BROSS et al., 2021b), adotam a ferramenta de Estimação de Movimento *Affine*.

Encontrar o bloco do quadro referência a ser utilizado pelas ferramentas de predi-

ção interquadros para representar a CB atual possui um elevado custo computacional. Devido a esse elevado custo, o uso de *hardware* dedicado é adotado em diversas soluções propostas em trabalhos da literatura que visam o processamento de vídeos em alta resolução e em tempo real. Contudo, a busca dessas ferramentas também demanda uma comunicação intensiva com a memória, visto que muitos acessos à memória são necessários para obter os dados a serem processados. Essa demanda intensiva por dados leva a memória empregada em um sistema de predição interquadros a apresentar um consumo de energia dominante se considerado o consumo de energia total do sistema de predição interquadros, demandando assim uma solução para a predição interquadros focada no consumo de energia da memória do sistema.

1.1 Motivação

O padrão de codificação VVC emprega o uso de técnicas de compressão de maior custo computacional do que os seus predecessores, como o suporte para um maior número de tamanhos de bloco, maior tamanho de CTU, além de empregar uma maior gama de ferramentas de codificação. Dentre as diferentes ferramentas de codificação suportadas pelo padrão VVC, a predição interquadros é responsável por até 50,58% do tempo total de codificação, considerando o *software* VTM na versão 9.0 (GONCALVES, 2021). Dentre as ferramentas ME e *Affine* ME, a ME acaba representando o menor custo computacional, representando até 25% do tempo de execução da predição interquadros do *software* VTM (GONCALVES, 2021). Já quanto ao *Affine* ME, buscar o movimento complexo de cada bloco faz com que o *Affine* ME seja responsável por até 46% do tempo de execução da predição interquadros do *software* VTM (GONCALVES, 2021).

Além do alto custo computacional demandado pelas ferramentas de predição interquadros para representar o movimento da cena, essas ferramentas de predição demandam ainda uma grande comunicação com a memória para obter todas as amostras necessárias para calcular a similaridade entre a CB atual e cada bloco a ser avaliado do quadro de referência.

No *software* VTM na versão 8.0, a requisição dos dados necessários para realizar a ME, considerando o grande número de tamanhos de CB suportados pelo VVC além do grande número de blocos candidatos dentro da SA, faz com que a ME seja responsável por mais de 28% de todos os acessos à memória realizados durante a codificação (CERVEIRA et al., 2020). Além disso, a requisição dos dados necessários para realizar o *Affine* ME é responsável por mais de 15% de todos os acessos à memória realizados durante a codificação com o VVC (CERVEIRA et al., 2020).

Essas ferramentas de predição interquadros não requerem que o processamento seja preciso, tolerando pequenas variações em seus resultados sem detalhes percep-

tíveis na imagem, visto que pequenas variações na representação do movimento geram pouca variação na eficiência da codificação (PORTO et al., 2020) (PENNY et al., 2020) . Isso permite que os algoritmos dessas ferramentas adotem heurísticas que reduzam o seu custo computacional, além de permitir o uso de operações imprecisas durante o processamento, que também permitem reduzir o custo computacional do processamento. Assim, devido ao grande custo computacional das ferramentas ME e *Affine* ME, o uso de *hardware* dedicado para o seu processamento se torna obrigatório, e técnicas de redução do custo computacional são exploradas em diversos trabalhos da literatura.

Na literatura atual, podem ser encontrados diversos trabalhos propondo soluções arquiteturais para as ferramentas de predição interquadros, especialmente para a ME. Para o *Affine* ME, poucos trabalhos podem ser encontrados devido a sua recente implementação em padrões de codificação de vídeo. Dentre os trabalhos para a ME, as estratégias que são adotadas trazem diversos benefícios para reduzir o número de acessos à memória. Contudo, estes trabalhos ainda podem fazer a ME realizar acessos redundantes à memória, visto que os trabalhos da literatura não exploram todo o potencial do reuso de dados e de operações.

Assim, nenhum dos trabalhos encontrados propondo soluções de *hardware* dedicado para a predição interquadros foca estritamente em reduzir os acessos à memória. Os trabalhos não apresentam a organização interna das memórias e, também, não avaliam ou discutem o consumo de potência relacionado à memória que é necessária para alimentar a unidade de processamento. Contudo, a grande dependência de dados da memória dessas ferramentas de predição interquadros precisa ser considerada no projeto de um *hardware* dedicado. O processamento não eficiente dessas ferramentas pode requisitar uma expressiva taxa de comunicação com a memória, e isso pode ser crítico, pois a memória pode não conseguir atingir a taxa de transferência necessária pela unidade de processamento, especialmente ao considerar o processamento de vídeos UHD (4K e 8K) em tempo real. Além disso, o sistema de memória tende a apresentar um elevado consumo de energia, o que pode ser crítico para dispositivos móveis alimentados a bateria. Assim, as soluções de *hardware* para a predição interquadros podem ter resultados inesperados caso sejam negligenciadas as características das tecnologias de memória atuais. Em especial, o consumo de energia do sistema completo pode ultrapassar as expectativas devido ao consumo de energia da memória, ou ainda, podem ter sido tomadas decisões de projeto sub-ótimas, nos quais o consumo de energia e a largura de banda com a memória não foram consideradas de forma adequada durante o projeto do *hardware*. Logo, para tratar o elevado custo computacional das ferramentas de predição interquadros do VVC, um projeto de *hardware* eficiente precisa levar em conta a organização da memória a ser integrada com a unidade de processamento, além do uso de heurísticas com custo computacional

reduzido e de otimizações diversas para a unidade de processamento.

Um exemplo motivacional da importância do projeto de *hardware* focado em memória apresentado a seguir. Foram projetadas três diferentes árvores de Soma de Absolutas Diferenças (*Sum of Absolute Differences* - SAD), comumente empregada em diversas arquiteturas para a ME, considerando diferentes níveis de paralelismo. Para este exemplo, foram consideradas três unidades de árvores de SAD similares, específicas para o cálculo do SAD de blocos 4×4, onde cada uma das unidades possui diferentes níveis de paralelismo, processando 4, 8, ou todas as 16 amostras do bloco 4×4 a cada ciclo de *clock*. Estas três unidades foram sintetizadas para *Application-Specific Integrated Circuit* (ASIC) considerando a biblioteca de células padrão da TSMC de 40nm. A frequência utilizada para cada uma das três unidades considera o processamento de todos os blocos candidatos delimitados por uma Área de Busca (*Search Area* - SA) de 20×20 amostras e, também, o processamento de vídeos HD 1080p (1920×1080 amostras) a 30 quadros por segundo (qps). Os resultados de dissipação de potência total (potência estática + potência dinâmica) dessas três arquiteturas de árvores de SAD são representados pela linha azul na Figura 1.

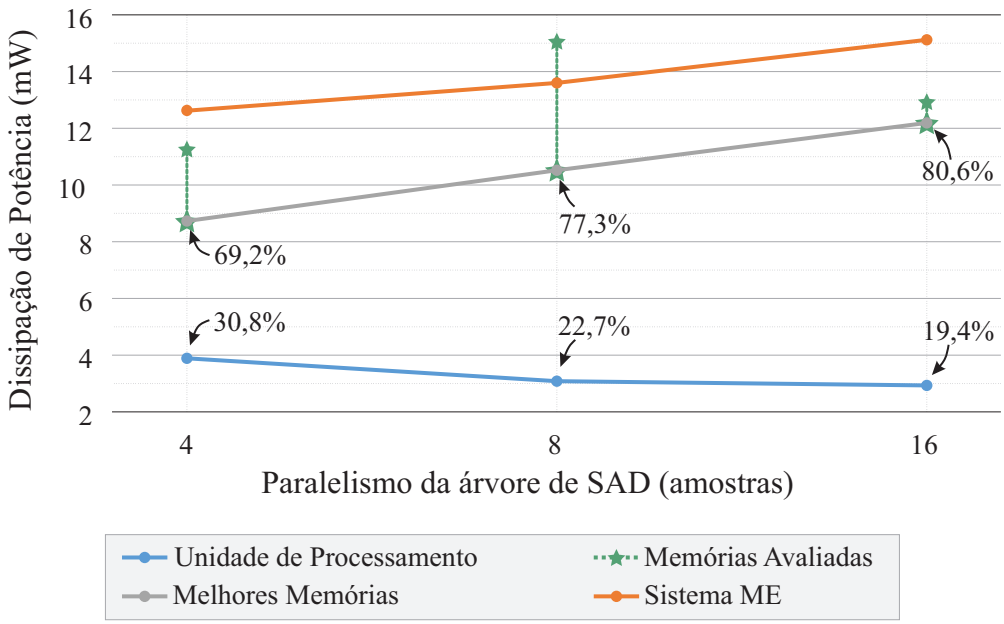


Figura 1 – Dissipação de potência de um sistema ME para o processamento de vídeos HD 1080p a 30 qps, considerando três níveis de paralelismo.

Para entregar as amostras necessárias por estas árvores de SAD, duas memórias com diferentes organizações foram modeladas para cada uma das árvores de SAD. A diferença entre essas duas memórias está no tamanho da palavra a ser utilizado dentro da memória, onde cada palavra tem tamanho igual ou metade do nível de paralelismo da unidade de processamento. Essas memórias foram modeladas para a tecnologia SRAM de 32nm, utilizando o *software* Cacti (LABS, 2022), e os resultados obtidos estão representados pelas estrelas verdes na Figura 1. Na Figura 1, a linha

cinza ainda representa a melhor organização de memória avaliada para cada um dos paralelismos, e a linha laranja representa a dissipação de potência total do sistema, considerando os resultados da unidade de processamento e a melhor memória avaliada para cada nível de paralelismo. Além disso, os valores apresentados na Figura 1 representam o percentual da dissipação de potência da unidade de processamento e da memória para a dissipação de potência total do circuito. Com base nesse exemplo, é possível destacar cinco observações quanto a importância do projeto de *hardware* focado em memória:

- Diferentes organizações internas da memória podem gerar resultados distintos, como no caso das memórias avaliadas para a árvore de SAD de paralelismo 8. Assim, memórias com diferentes organizações internas precisam ser consideradas para serem integradas com a unidade de processamento.
- O consumo de energia de todas as memórias avaliadas foi maior do que o consumo de energia da unidade de processamento, considerando qualquer um dos níveis de paralelismo. Assim, é possível constatar que o consumo de energia relacionado à memória é dominante quando comparado com a unidade de processamento, especialmente para sistemas ME.
- O consumo de energia da unidade de processamento tende a reduzir com o aumento do paralelismo da arquitetura, visto que a frequência de operação é reduzida e menos estágios de pipeline são necessários pela arquitetura.
- Considerando a melhor memória avaliada para cada nível de paralelismo, o consumo de energia da memória tende a aumentar juntamente com o aumento do paralelismo da unidade de processamento, devido a organização física da memória SRAM, que precisa entregar um número maior de dados a cada ciclo de *clock*.
- Levando em conta apenas os resultados da unidade de processamento, uma escolha ótima local seria utilizar a unidade de processamento com o maior nível de paralelismo, pois ela resultou no menor consumo de energia. Contudo, visto que os resultados da memória são dominantes, e que o consumo de energia da memória tem comportamento inverso do que o consumo de energia da unidade de processamento, a escolha correta deve ser considerar o consumo de energia de todo o sistema e, no caso dessa árvore de SAD, o melhor ponto de operação em termos de dissipação de potência do sistema completo está localizado no menor nível de paralelismo.

Como visto, o projeto de um sistema focado na memória é crucial visando obter os melhores resultados de consumo de energia para o sistema completo de um codificador de vídeo, visto que o consumo de energia da memória tende a ser dominante

sobre o consumo de energia da unidade de processamento. Porém, não apenas o projeto da unidade de processamento deve estar focado em obter o menor consumo de energia da memória (como realizando poucas requisições à memória), mas a memória também pode ser explorada buscando reduzir o seu consumo de energia. Para tanto, podem ser avaliadas tecnologias de memória que possuem diferentes características e que podem ser integradas com o sistema, ou ainda, explorando técnicas de armazenamento impreciso que resultam em uma redução no consumo de energia. Além disso, técnicas tradicionais de redução de custo computacional que viabilizem a implementação da unidade de processamento ainda devem ser empregadas em paralelo ao projeto de *hardware* focado em memória, como a exploração de heurísticas de redução de custo computacional para os algoritmos adotados nessas ferramentas, além da exploração de operadores imprecisos na unidade de processamento.

1.2 Objetivo

O objetivo deste trabalho de Doutorado é o desenvolvimento de um sistema completo para a predição interquadros do padrão VVC, com suporte às suas ferramentas de predição de maior custo computacional: ME e *Affine* ME, com foco na redução do consumo de energia do sistema. Para isso, um dos focos deste trabalho será a exploração de diferentes tecnologias de memórias que possam ser utilizadas nesse sistema, combinando com a exploração de estratégias de redução do número de acessos à memória e técnicas de armazenamento impreciso. Para a unidade de processamento, além de explorar heurísticas de redução de custo computacional de cada uma das ferramentas de predição, também será explorado o uso de operadores imprecisos em diferentes etapas dessas ferramentas de predição. Assim, no final deste trabalho é esperado obter um sistema completo para a predição interquadros do padrão VVC com alta eficiência energética para aplicações em dispositivos móveis, e que seja capaz de processar vídeos de ultra alta resolução em tempo real. Detalhes completos da proposta estão sendo apresentados no Capítulo 6.

1.3 Apresentação do Texto

O Capítulo 2 apresenta o codificador VVC com foco no particionamento de blocos e na etapa de predição interquadros. O Capítulo 3 apresenta as características e o funcionamento das operações de quatro tecnologias de memória que serão foco do trabalho. Já o Capítulo 4 apresenta os principais trabalhos na literatura propondo soluções com as quais é possível obter uma redução no consumo de energia do sistema de predição interquadros. O Capítulo 5 descreve as principais oportunidades que podem ser exploradas no projeto de *hardware* dedicado e de baixo consumo energético

para a predição interquadros do VVC, enquanto o Capítulo 6 apresenta a proposta do trabalho para o desenvolvimento de um sistema para a predição interquadros do padrão VVC priorizando a memória interna. O Capítulo 7 apresenta os resultados preliminares. Por fim, o Capítulo 8 apresenta as conclusões deste trabalho.

2 O CODIFICADOR VVC

Um vídeo digital é uma sequência de imagens (ou quadros) independentes. A taxa na qual estes quadros são capturados e/ou exibidos para o espectador é dado pela resolução temporal. Esta resolução temporal deve ser maior do que 30 quadros por segundo (qps) para passar a sensação de movimento ao espectador (MIANO, 1999), embora melhorias na qualidade subjetiva sejam observadas com o aumento desta resolução temporal (MADHUSUDANA et al., 2021). Além disso, cada quadro do vídeo é representado por um determinado número de amostras, sendo a resolução espacial dada pelo número de amostras de largura e altura de um quadro do vídeo. Uma maior resolução espacial costuma ser utilizada de forma a obter uma melhoria na experiência dos espectadores, exibindo mais detalhes e informações da cena a cada quadro. De acordo com um relatório da CISCO, em 2023, cerca de 66% dos televisores conectados à internet terão resolução UHD 4K (3840×2160 amostras) (CISCO, 2018). Por outro lado, o aumento da resolução resulta em uma grande quantidade de dados necessários para a representação dos vídeos, especialmente para as resoluções da chamada ultra alta definição (*Ultra-High Definition* - UHD), como as já comercialmente conhecidas UHD 4K (3840×2160 amostras) e UHD 8K (7680×4320 amostras).

Por conta disso, de forma a viabilizar o armazenamento e a transmissão de vídeos em alta resolução, o uso de técnicas de compressão se torna obrigatória e, neste quesito, estado-da-arte é o padrão de codificação de vídeo VVC. Para atingir uma alta eficiência de compressão, o VVC adota um conjunto de técnicas inovadoras, enquanto herda diversas técnicas do padrão HEVC. Assim, visto que este trabalho foca na predição interquadros do VVC, as próximas seções explicam os detalhes necessários para a compreensão do funcionamento da predição interquadros do VVC. A Seção 2.1 explica o particionamento de blocos adotado pelo padrão VVC. Já a Seção 2.2 apresenta, de maneira mais específica, a predição interquadros como implementada no *software* VTM, mostrando os fatores que elevam o seu custo computacional e os requisitos de memória, apresentando assim a sua relevância para o presente estudo.

2.1 Particionamento de Blocos no VVC

O VVC é um codificador baseado no processamento de blocos. Isto é, para codificar o vídeo, cada quadro é dividido em pequenos blocos a serem processados pelas ferramentas de codificação.

No VVC, cada quadro é dividido em CTUs, que possuem o formato quadrado e o tamanho máximo de 128×128 amostras. O *software* VTM utiliza originalmente o tamanho de 128×128 amostras para representar cada CTU. Na sequência, cada CTU é dividida em Unidades de Codificação (*Coding Unit* - CU), utilizando um particionamento denominado Árvore Quaternária e Árvore Multi-Tipos (*Quaternary Tree Plus Multi-Type Tree* - QT+MTT) (BROSS et al., 2021b) (HUANG et al., 2021).

O particionamento QT+MTT está representado na Figura 2. Como pode ser visto na Figura 2, inicialmente a CTU passa pelo processo de particionamento Árvore Quaternária (*Quaternary Tree* - QT), o qual pode resultar em uma sub-CU de tamanho igual ao da CTU, ou dividir a CTU em quatro sub-CUs quadradas de mesmo tamanho (HUANG et al., 2021). Esse processo QT é aplicado recursivamente sobre as sub-CUs geradas, enquanto estas sub-CU geradas possuírem um tamanho maior do que 8×8 amostras (HUANG et al., 2021). Portanto, o processo QT resulta em uma ou várias sub-CUs quadradas com tamanho entre 8×8 a 128×128 amostras.

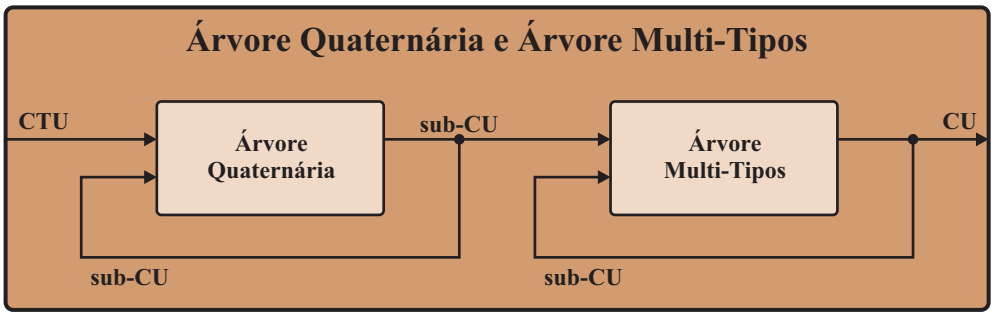


Figura 2 – Fluxo do particionamento QT+MTT do VVC a ser aplicado sobre uma CTU.

Na sequência, cada sub-CU resultante do processo QT é passada pelo processo de particionamento Árvore Multi-Tipos (*Multi-Type Tree* - MTT). O processo MTT pode resultar em uma CU de tamanho igual ao da sub-CU de entrada (não aplicar nenhuma divisão sobre a sub-CU), ou resultar na divisão da sub-CU de entrada em duas ou três CUs utilizando um particionamento binário ou ternário, respectivamente (HUANG et al., 2021). O processo MTT é aplicado recursivamente sobre as CUs geradas e, portanto, CUs resultantes do particionamento binário ou ternário podem ser novamente divididos pelo particionamento binário ou ternário (HUANG et al., 2021).

A Figura 3-(a) e Figura 3-(b) apresentam o formato das CUs resultantes do particionamento binário e ternário, respectivamente. No caso do particionamento binário, o bloco de entrada é dividido horizontalmente ou verticalmente em duas CUs de tamanho igual. No caso do particionamento ternário, o bloco de entrada é dividido

horizontalmente ou verticalmente em três CUs, no qual o bloco central possui o dobro do tamanho dos demais.

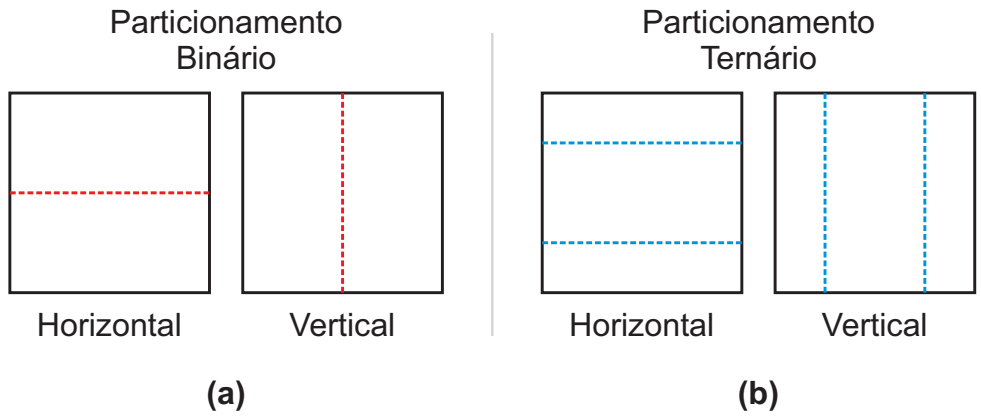


Figura 3 – Representação dos particionamentos do processo MTT. (a) Particionamento Binário. (b) Particionamento Ternário.

A Figura 4 apresenta um exemplo de uma sub-CU sendo dividida horizontalmente pelo particionamento binário e, em seguida, a sub-CU superior sendo dividida verticalmente pelo particionamento ternário.

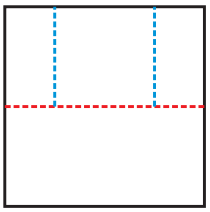


Figura 4 – Exemplo do particionamento binário seguido do particionamento ternário.

Existem duas restrições que são aplicadas no processo QT+MTT do VVC sobre sub-CUs de tamanho 128×128 buscando reduzir o custo computacional dos decodificadores do VVC (HUANG et al., 2021). A primeira restrição diz que o particionamento ternário do processo MTT não pode ser aplicado sobre sub-CUs de tamanho 128×128. Já a segunda restrição diz que o processo MTT não pode aplicar recursividade sobre sub-CUs de tamanho 128×128. Com isso, quando o processo QT resulta em uma sub-CU de tamanho 128×128, o processo MTT aplicado sobre essa sub-CU de tamanho 128×128 pode resultar em apenas três tamanhos de CU: Uma CU de tamanho 128×128, ou duas CUs de tamanho 128×64, ou duas CUs de tamanho 64×128.

A configuração original do *software* VTM ainda especifica que a recursividade do processo MTT é de nível três, o que indica que uma sub-CU não pode ser dividida mais do que três vezes consecutivas pelo processo MTT. Com isso, as amostras de cada CTU podem ser agrupadas em até 27 tamanhos de CU diferentes.

Os acrônimos CTU e CU são utilizadas para denominar o agrupamento de todos os canais de informação do quadro do vídeo (comumente, um canal de luminância e

dois canais de crominância). Já o bloco da CTU e da CU formado por um canal individual é denominado Bloco de Codificação em Árvore (*Coding Tree Block* - CTB) e CB, respectivamente. Assim, a saída do particionamento QT+MTT são as CUs e, destas CUs, os CBs são os blocos nos quais serão aplicados as ferramentas de predição interquadros (BROSS et al., 2021b). Cabe salientar que, no padrão HEVC, CUs são ainda divididas em Unidades de Predição (*Prediction Unit* - PU), com múltiplos canais de informação, e Blocos de Predição (*Prediction Block* - PB), contendo apenas um canal de informação, nos quais são aplicadas as ferramentas de predição (SULLIVAN et al., 2012).

2.2 Predição Interquadros do Padrão VVC

A predição interquadros busca identificar as redundâncias temporais entre quadros vizinhos do vídeo. Para isso, as amostras presentes na CB atual, do quadro sendo codificado, podem ser representadas sinalizando a localização de uma informação similar localizada em quadros de referência, que já foram codificados anteriormente. Existem diferentes quadros de referência que podem ser empregados para representar a informação da CB atual. Portanto, para representar a CB atual, além de sinalizar a localização da informação similar dentro do quadro de referência, deve ser sinalizado qual dos quadros de referência que a informação similar está.

Os quadros de referência são agrupados em duas listas, L_0 e L_1, de acordo com a ordem temporal de captura do vídeo, sendo que cada lista possui até 16 quadros de referência. O VVC permite que a codificação dos quadros do vídeo seja realizada fora da ordem de captura, como na configuração *Random Access* (BOSSEN et al., 2020). Nesta codificação fora de ordem, os quadros capturados antes do quadro atual estão localizados na lista L_0, enquanto que os quadros capturados após o quadro atual estão localizados na lista L_1. No caso da codificação sequencial dos quadros de acordo com a ordem de captura do vídeo, como na configuração *Low Delay* (BOSSEN et al., 2020), apenas a lista L_0 é empregada, visto que todos os quadros já codificados são temporalmente passados. Assim, para indicar qual o quadro de referência que deve ser utilizado para representar a CB atual, deve ser sinalizado qual a lista do quadro de referência, e qual a posição do quadro de referência dentro desta lista.

O fluxograma de avaliação da predição interquadros, como empregado no *software* VTM, é exibido na Figura 5. Este fluxograma é aplicado sobre cada CB a ser avaliada a predição interquadros, e é utilizado para encontrar uma informação similar em um quadro de referência a ser utilizada para representar a CB atual. Como pode ser visto na Figura 5, a predição interquadros é realizada para cada um dos quadros referência, de ambas as listas.

Inicialmente, é realizada uma predição uni-direcional em cada uma das listas. A

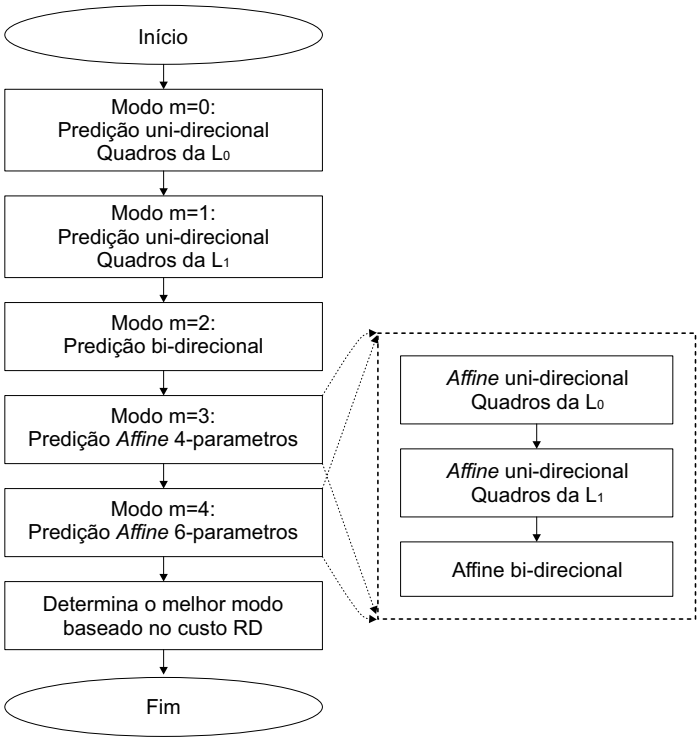


Figura 5 – Fluxograma da predição interquadros aplicada no *software* VTM. Adaptado de: (PARK; KANG, 2019)

predição uni-direcional aplica a Estimação de Movimento (*Motion Estimation* - ME) sobre cada quadro da lista avaliada. A ME realiza a busca pelo bloco mais similar à CB atual considerando apenas movimentos translacionais, como será explicado em detalhes na Seção 2.2.1. Assim, o resultado da predição uni-direcional, aplicada sobre uma lista, é um conjunto com os blocos mais similares à CB atual encontrados em cada um dos quadros de referência avaliados.

Após a avaliação da predição uni-direcional, é realizada a predição bi-direcional. Esta predição bi-direcional permite utilizar a informação de dois blocos de quadros de referência diferentes para representar a CB atual. Para avaliar a predição bi-direcional, o *software* VVC utiliza o conjunto de resultados da predição uni-direcional para cada uma das listas.

Após a avaliação da predição uni-direcional e bi-direcional, a predição *Affine* é realizada. A predição *Affine* permite sinalizar a localização de duas ou três extremidades da CB atual no quadro de referência e, com estas informações, a predição *Affine* é capaz de representar movimentos complexos contendo rotação, *zoom* ou distorção, como será melhor detalhado na Seção 2.2.2. O *software* VVC avalia a predição *Affine* em duas etapas, avaliando inicialmente a predição *Affine* com quatro parâmetros e, em seguida, a predição *Affine* com seis parâmetros. Como pode ser visto na Figura 5, assim como nas etapas da predição interquadros anteriores, a predição *Affine* para quatro e seis parâmetros também realiza a predição uni-direcional e bi-direcional, permitindo à predição *Affine* representar a CB atual utilizando a informação de um ou

dois quadros de referência. Na predição uni-direcional *Affine*, é realizado o *Affine* ME, como também será apresentada em detalhes na Seção 2.2.2, para encontrar o resultado da predição *Affine* para cada um dos quadros de referência, enquanto que a predição bi-direcional *Affine* avalia a representação da CB atual utilizando a informação de diferentes quadros de referência.

Por fim, o *software* VTM utiliza o custo taxa-distorção (*Rate Distortion* - RD) de cada modo avaliado para avaliar qual o modo a ser utilizado para representar a CB atual. O cálculo do custo RD está representada na Equação (1), e ele considera D como a distorção resultante do modo avaliado (representando assim a qualidade da imagem), λ representa o multiplicador *Legrange*, e R como a quantidade de *bits* para representar o modo avaliado (ou uma estimativa do número de *bits*). Assim, avaliar os custos RD de todos os modos avaliados permite que o resultado da predição interquadros seja o modo que resulte na menor perda de qualidade visual e a menor quantidade de *bits* para ser representado.

$$J = D + \lambda \times R \tag{1}$$

2.2.1 Estimação de Movimento

A ME busca representar a CB atual utilizando informações do quadro de referência. Um exemplo do processamento realizado pela ME está representado na Figura 6. A ME busca pelo bloco mais similar a CB atual considerando blocos candidatos dentro da SA, localizada dentro do quadro de referência. Após encontrado o bloco candidato mais similar a CB atual, um Vetor de Movimento (*Motion Vector* - MV) pode ser gerado para representar a localização do melhor bloco candidato, e esse MV utilizado como saída da ME juntamente com o resíduo, que é a diferença entre a CB atual e o bloco candidato resultante da ME. Para reconstruir a CB atual utilizando as informações do quadro de referência, a etapa de Compensação de Movimento utiliza o bloco candidato que inicia na amostra apontada pelo MV, acrescentando o resíduo neste bloco, gerando assim o bloco reconstruído.

O VVC permite que os MVs da ME tenham precisão de até 1/4 de amostra. Para encontrar esses MVs, duas etapas são aplicadas na ME: a ME Inteira (*Integer* ME - IME) e a ME Fracionária (*Fractional* ME - FME). A IME realiza a busca por blocos candidatos apenas de posições inteiras, logo resultando em um MV com precisão inteira. Já a FME realiza um refinamento utilizando amostras em posições fracionárias, e resulta na parte fracionária do MV da ME. A IME e a FME executadas pelo *software* VTM são bem similares a IME e FME do *software HEVC Test Model* (HTM), o *software* de referência do padrão HEVC. Tanto IME como FME são explicados na sequência, baseado em sua implementação no *software* VTM. No *software* VTM, a ME é executada para até 27 tamanhos de CB diferentes, com tamanhos entre 4×8 e 8×4 até

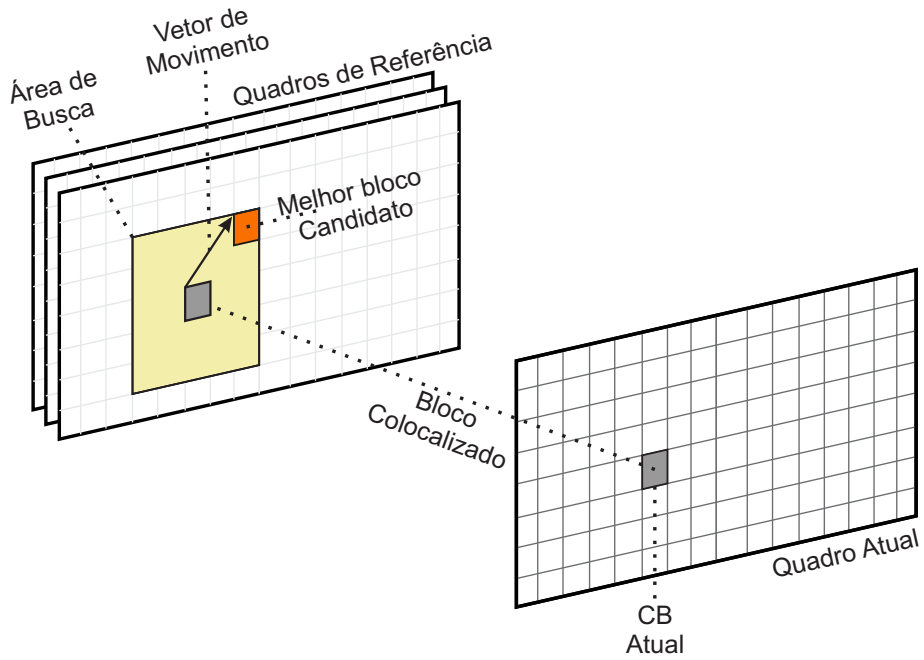


Figura 6 – Processamento realizado pela ME. Adaptado de (PORTO, 2012).

128×128.

2.2.1.1 Estimação de Movimento Inteira - IME

Dentro da SA existem diversos blocos candidatos que podem ser considerados para representar a CB atual, e diferentes algoritmos podem ser adotados para definir quais os blocos candidatos devem ser avaliados. Ao escolher cada bloco candidato, o *software* VTM calcula a similaridade entre as amostras da CB atual e as amostras do bloco candidato, utilizando a métrica da Soma das Diferenças Absolutas (*Sum of Absolute Differences* - SAD).

O cálculo do SAD está representado na Equação (2), onde as variáveis C, R, x e y representam uma amostra da CB atual, uma amostra do bloco candidato, a posição horizontal da amostra e a posição vertical da amostra, respectivamente. Portanto, o cálculo do SAD obtém a diferença entre os dois blocos, amostra por amostra, e então obtém o valor absoluto de cada diferença obtida. Por fim, o SAD é calculado pela soma de todas as diferenças absolutas computadas. Assim, um valor menor de SAD representa uma maior similaridade entre a CB atual e o bloco candidato.

$$SAD = \sum_{y=0}^{CB_{altura}} \sum_{x=0}^{CB_{largura}} \left| C_{(x,y)} - R_{(x,y)} \right| \tag{2}$$

No caso do *software* VTM, o algoritmo adotado originalmente para a escolha dos blocos candidatos é o *Test Zone Search* (TZS) (JVET, 2022). A Figura 7 apresenta o fluxograma das etapas do TZS. Como pode ser visto, o TZS é composto de quatro etapas, sendo elas Predição, Busca Inicial, *Raster* e Refinamento. Inicialmente, a

etapa de Predição e a Busca Inicial são executadas sequencialmente. Então, a etapa *Raster* apenas é aplicada caso a distância entre o resultado da Predição e o resultado da Busca Inicial seja maior que a constante *iRaster*. Caso essa distância seja menor que a constante *iRaster*, a etapa *Raster* é ignorada. A constante *iRaster* é definida com o valor cinco no *software* VTM. Por fim, a etapa de Refinamento é aplicada.

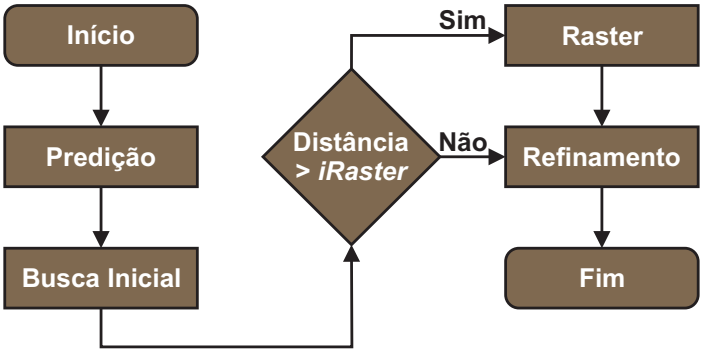


Figura 7 – Fluxograma das quatro etapas do TZS.

A etapa de Predição é a primeira a ser executada, e é utilizada para definir a posição inicial da busca a ser executada pelas próximas etapas. A Predição tenta direcionar a busca para a região onde há maior probabilidade de encontrar o bloco adequado e, para isso, a etapa de Predição avalia blocos candidatos baseado nos MVs de blocos vizinhos.

Para encontrar o melhor ponto de partida das próximas etapas, a Predição avalia os blocos candidatos referentes a até seis diferentes preditores. Cada preditor avalia o MV de diferentes blocos vizinhos (JVET, 2022), que podem ser:

- Preditor Zero: MV para o bloco colocalizado
- Preditor Esquerdo: MV do bloco vizinho esquerdo
- Preditor Superior: MV do bloco vizinho superior
- Preditor Superior-Direito: MV do bloco vizinho superior-direito
- Preditor Mediana: Média entre os MVs de blocos vizinhos
- Preditor 2Nx2N: MV da maior CB da CTB atual.

A etapa Busca Inicial é realizada tomando como ponto inicial o resultado da etapa Predição. A Busca inicial adota um esquema de busca expansiva para definir os blocos candidatos para serem avaliados (JVET, 2022). Essa busca expansiva pode ter o formato de losango ou quadrado, sendo o losango utilizado originalmente no *software* VTM. A Figura 8 apresenta o esquema em formato de losango aplicado pela Busca Inicial, estando o ponto inicial (resultado da etapa Predição) localizado no centro da

Figura. Cada um dos blocos coloridos da Figura 8 representa a primeira amostra de um bloco candidato a ser avaliado. A busca é considerada expansiva pois ela começa avaliando os quatro blocos candidatos mais próximos do ponto inicial e, então, ela expande para avaliar blocos candidatos mais distantes do ponto inicial.

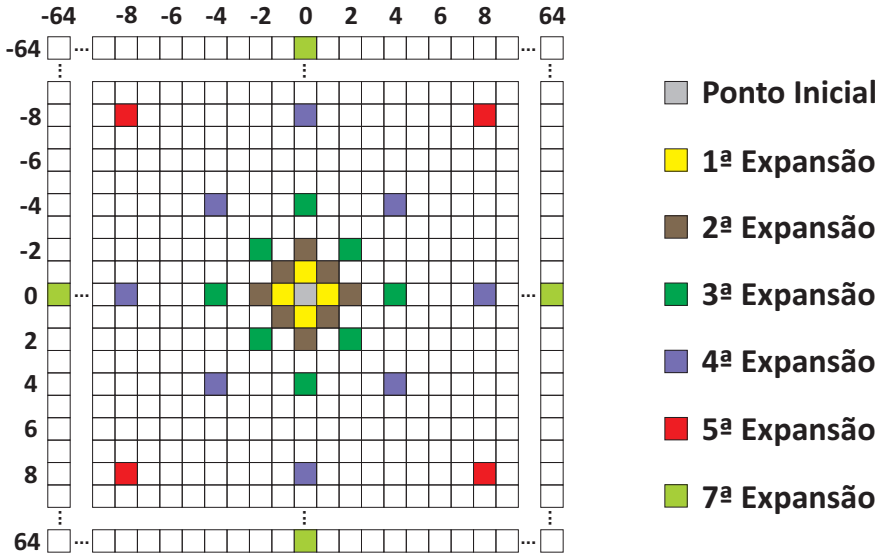


Figura 8 – Esquema de expansões e escolha dos blocos candidatos para avaliação da busca em losango.

A segunda, terceira e quarta expansões da Busca Inicial avaliam oito blocos candidatos, que estão localizados em duas, quatro, e oito amostras de distância do ponto inicial, respectivamente. Nas próximas expansões, 16 blocos candidatos são avaliados por expansão, estando esses candidatos localizados em 16, 32 e 64 amostras de distância do ponto inicial (JVET, 2022).

As últimas expansões podem não ser realizadas caso a condição de parada seja atingida, o que termina com o processamento da Busca Inicial. Essa condição de parada ocorre caso a Busca Inicial realize três expansões consecutivas sem encontrar um melhor resultado do que o encontrado nas expansões anteriores. Porém, no caso em que a Busca Inicial realiza todas as expansões, até 76 blocos candidatos podem ser selecionados para avaliação (considerando uma SA original de 192×192 amostras).

A etapa *Raster* só é executada quando o resultado da etapa Predição não for considerado satisfatório. Nesse caso, o resultado da Busca Inicial deve estar a uma distância maior do que *iRaster* amostras do resultado da etapa Predição e, assim, a etapa *Raster* é executada, a qual tenta redirecionar a busca para uma nova região da SA onde a próxima etapa do TZS poderia encontrar um melhor resultado. Quando o resultado da Busca Inicial estiver próximo do resultado da etapa Predição, a etapa *Raster* não é aplicada.

Quando a etapa *Raster* é aplicada, uma busca completa subamostrada é realizada. Essa busca avalia blocos candidatos de toda a SA, porém, apenas um a cada *iRaster*

blocos candidatos são selecionados para avaliação. Assim, considerando a Faixa de Busca (*Search Range* - SR) original de $\pm$ 64 amostras, que dita a magnitude do maior MV que pode ser obtido, a etapa *Raster* pode avaliar até 625 blocos candidatos.

Por fim, como o nome sugere, a etapa Refinamento é responsável por refinar o melhor bloco candidato obtido nas etapas anteriores. Este Refinamento emprega, recursivamente, várias iterações da busca em formato losango realizada pela Busca Inicial. Isto é, cada iteração aplica o esquema em losango exibido na Figura 8 para selecionar blocos candidatos para avaliação. Cada iteração possui a mesma condição de parada da Busca Inicial, na qual as expansões terminam caso três expansões consecutivas sejam realizadas sem obter um melhor resultado. Porém, quando todas as expansões são realizadas, até 76 blocos candidatos podem ser avaliados a cada iteração do Refinamento.

As iterações do Refinamento são recursivas pois a primeira iteração toma como ponto inicial o bloco candidato resultante das etapas anteriores (resultado da Busca Inicial ou da etapa *Raster*), enquanto que nas próximas iterações, o ponto inicial é definido pelo bloco candidato resultante da iteração imediatamente anterior. Por exemplo, o bloco candidato resultante da primeira iteração é utilizado como ponto inicial da segunda iteração. Assim, várias iterações consecutivas são realizadas enquanto uma iteração conseguir encontrar um bloco candidato melhor do que as iterações anteriores. A etapa de Refinamento só termina caso uma iteração seja executada sem encontrar um resultado melhor do que as iterações anteriores. Visto que não há limites no número de iterações consecutivas, o número total de blocos candidatos que podem ser avaliados pela etapa Refinamento é imprevisível.

Considerando o número máximo de candidatos que pode ser avaliado por cada etapa do TZS em uma SA de 192×192, que a etapa *Raster* é executada e que quatro iterações do refinamento são executadas, a busca pelo melhor bloco candidato pode requerer a avaliação de até 1011 candidatos. Então, após o término da avaliação de todos os blocos candidatos do TZS, o resultado obtido passa para a próxima etapa, a FME.

2.2.1.2 Estimação de Movimento Fracionária - FME

A etapa IME realiza a busca apenas por blocos candidatos utilizando amostras em posições inteiras do quadro referência. Assim, a etapa FME realiza um refinamento sobre o bloco candidato resultante da IME. Esse refinamento assume que o movimento não está limitado a amostras de posições inteiras, podendo ser melhor representado por blocos em posições fracionárias. A FME, como realizada no *software* VTM, é dividida em dois passos. O primeiro passo é a interpolação, enquanto o segundo passo é a busca e comparação.

A interpolação é o passo responsável pela geração de amostras em posições fraci-

onárias, ao redor das amostras do bloco candidato resultante da IME. O VVC permite que os MVs da ME tenham precisão de até $1/4$ de amostra. Portanto, são utilizados três filtros de interpolação diferentes para a geração das amostras em posições fracionárias (JVET, 2022). A Figura 9 apresenta a interpolação de um bloco 8×8. Os blocos em azul da Figura 9-(b) representam o bloco candidato resultante da etapa IME, enquanto que os blocos cinza e branco representam as amostras em posição de $1/2$ amostra e $1/4$ de amostra.

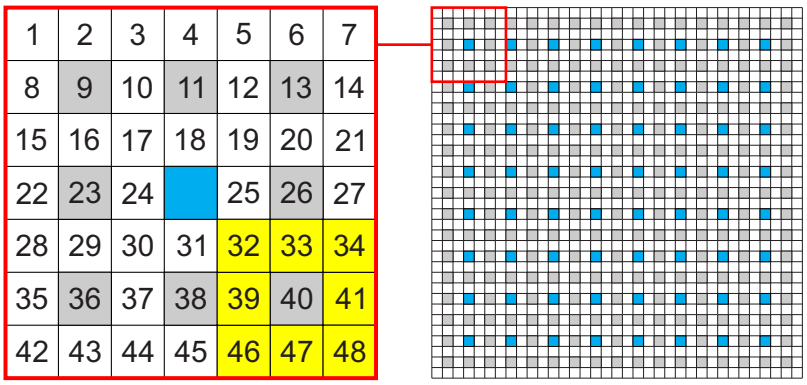
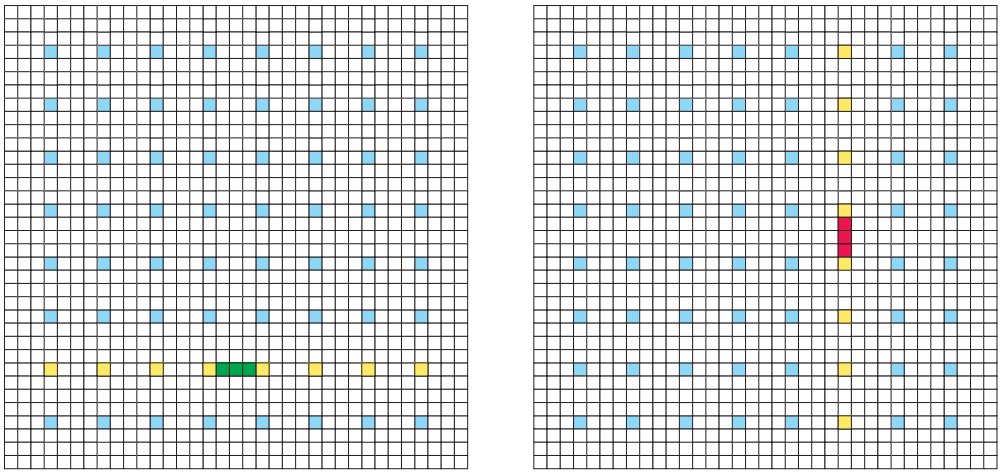


Figura 9 – FME para um bloco 8×8. (a) Primeira amostra de cada um dos 48 blocos que podem ser gerados. (b) Todas as amostras fracionarias que podem ser geradas ao redor de um bloco 8×8.

Para a geração de cada amostra da Figura 9, o padrão VVC define filtros de interpolação de 8-*taps*, que utilizam até oito amostras de posições inteiras ao redor da amostra a ser gerada. A Figura 10 apresenta as amostras em posições inteiras necessárias para a interpolação das amostras em posições fracionárias. Na Figura 10-(a), são utilizadas as oito amostras em posições inteiras representadas pelos quadrados amarelos, localizadas horizontalmente ao redor das três amostras a serem geradas, as quais estão representadas em verde. Já na Figura Figura 10-(b), são utilizadas as oito amostras em posições inteiras e representadas em amarelo, localizadas verticalmente ao redor das amostras a serem geradas, que estão representadas em vermelho.

As amostras interpoladas são então agrupadas para gerar os blocos candidatos em posições fracionárias, e esses blocos candidatos avaliados quanto a sua similaridade com a CB atual pelo passo de busca e comparação. No total, até 48 novos blocos em posições fracionárias podem ser gerados ao redor do resultado da IME (JVET, 2022). A Figura 9-(a) representa o agrupamento das amostras em posições fracionárias, na qual cada um dos quadrados de um a 48 representam a primeira amostra de cada um dos 48 blocos fracionários que podem ser gerados.

Para a avaliação destes blocos em posições fracionárias, o *software* VTM adota um pequeno algoritmo que avalia apenas 16 dos 48 blocos. Inicialmente, apenas os oito blocos candidatos em posição de $1/2$ amostra são gerados e comparados quanto a sua similaridade com a CB atual. Após, o melhor bloco avaliado (que também pode ser



(a) (b)

Figura 10 – Amostras de entrada em posições inteiras (quadrados amarelos) necessários para a interpolação de amostras em posições fracionárias (quadrados verdes e vermelhos). (a) Interpolação horizontal. (b) Interpolação vertical.

o resultado da IME) define a localização das amostras de $1/4$ de amostra que devem ser geradas e comparadas. A figura 9-(a) apresenta esse algoritmo considerando que o melhor bloco de $1/2$ de amostra foi o bloco de posição 40. Assim, os blocos de posição $1/4$ de amostra a serem gerados e avaliados são os blocos marcados em amarelo na Figura 9-(a).

2.2.1.3 Acessos à Memória

Pela Equação (2), pode ser visto que a operação do SAD requer as amostras da CB atual e também do bloco candidato. Assim, o número de amostras necessárias para calcular o valor do SAD para um determinado bloco candidato depende do tamanho da CB, sendo dado por V na Equação (3).

$$V(CB_{altura}, CB_{largura}) = 2 * CB_{altura} * CB_{largura} \tag{3}$$

Uma CTB pode ser dividida em várias CBs de mesmo tamanho, sendo que o valor de N na Equação (4) apresenta o número de CBs na qual uma CTB pode ser dividida, de acordo com o tamanho da CB. Considerando o processamento de um único bloco candidato por todas as CBs de mesmo tamanho, o número de amostras a serem requisitadas para processamento de cada candidato depende apenas do número de CBs de mesmo tamanho que cabem dentro da CTB. Este número de amostras, necessário para processar um bloco candidato por todas as CBs de um determinado tamanho, é dado por G na Equação (5).

$$N(CB_{altura}, CB_{largura}) = (128/CB_{altura}) * (128/CB_{largura}) \tag{4}$$

$$G(CB_{altura}, CB_{largura}) = N(CB_{altura}, CB_{largura}) * V(CB_{altura}, CB_{largura}) \quad (5)$$

Sendo assim, considerando que a execução do TZS na IME pode requisitar a avaliação de até 1011 blocos candidatos para cada CB, que o processamento de cada um dos 27 tamanhos de CB será realizado de maneira independente, e que a IME será realizada apenas para um quadro de referência de cada lista, o processamento da IME para cada CTB pode requisitar até 1,78 bilhões de amostras da memória. Este valor é dado por H na Equação (6) e considera os 27 diferentes tamanhos de CB, os 1011 blocos candidatos a serem avaliados pelo TZS, os dois quadros de referência e número de amostras requisitadas da memória para processar cada bloco candidato por todas as CBs de mesmo tamanho.

$$H = 27 * 1011 * 2 * G(CB_{altura}, CB_{largura}) \quad (6)$$

Assim, no pior caso, para processar um vídeo UHD 4K a 30 qps, no qual cada quadro UHD 4K possui 510 CTBs de cada canal de informação, processar a IME para todas as CTBs de um canal de informação pode demandar até 27,37TB/s de comunicação com a memória para requisitar todas as amostras necessárias para processamento, o que é inalcançável considerando as tecnologias de memória atuais (KIM et al., 2022).

2.2.2 Estimação de Movimento *Affine*

Como dito anteriormente, o *Affine* ME permite sinalizar mais informações sobre o movimento da CB atual com relação ao quadro de referência e, portanto, permite representar movimentos como rotação, *zoom* ou distorção. O padrão VVC permite o emprego do *Affine* com quatro parâmetros, que utiliza dois Pontos de Controle (PC), ou com seis parâmetros, que utiliza três PCs.

A predição com quatro parâmetros está representada na Figura 11. Como pode ser visto, são empregados dois MVs para representar a localização da CB atual no quadro de referência, um em cada PC, sendo que os PCs estão localizados nos extremos superiores da CB atual. Com quatro parâmetros, é possível representar movimentos como rotação e *zoom*.

Já a Figura 12 representa a predição *Affine* com seis parâmetros. Ela é similar com a predição quatro parâmetros. Porém, são utilizados três MVs, um a mais do que a predição com quatro parâmetros, estando este terceiro MV no PC do extremo inferior esquerdo da CB atual. Utilizando o *Affine* com seis parâmetros, é possível representar movimentos de distorção, como apresentado na Figura 12, além dos movimentos de rotação e *zoom*.

O padrão VVC permite que a predição *Affine* seja utilizada para representar CBs

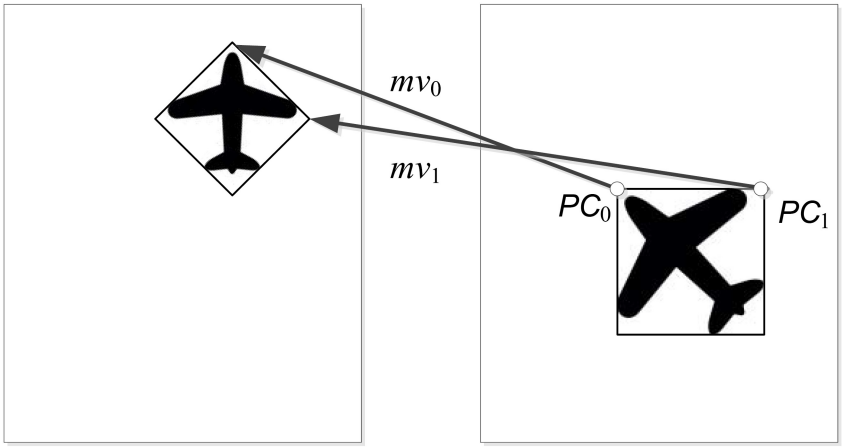


Figura 11 – Predição *Affine* com quatro parâmetros. Adaptado de (ZHANG et al., 2019)

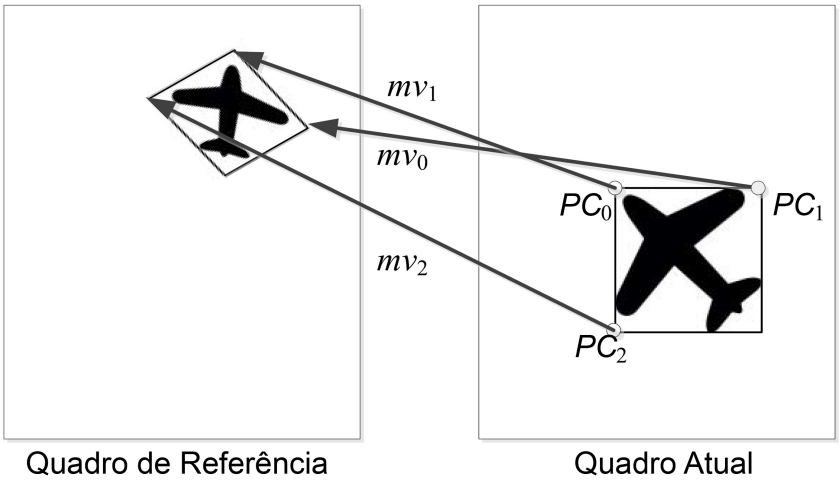


Figura 12 – Predição *Affine* com seis parâmetros. Adaptado de (ZHANG et al., 2019)

apenas maiores que 16×16 amostras. Através de avaliações realizadas no *software* VTM, predição *Affine* foi executado para 12 diferentes tamanhos de CB, com tamanhos entre 16×16 e 128×128. Na sequência, será apresentado o método de reconstrução do *Affine* e, após, a sinalização e a busca pelos MVs dos PCs.

2.2.2.1 Reconstrução Affine

A Reconstrução *Affine* realiza a reconstrução da CB atual utilizando a informação do quadro de referência apontada pelos dois ou três MVs. A Equação (7) é utilizada para definir a localização exata das amostras do quadro referência que serão utilizadas para reconstruir a CB atual quando predita com o *Affine* quatro parâmetros.

$$\begin{cases} mv^x(x,y) = \dfrac{(mv_1^x - mv_0^x)}{w}x + \dfrac{(mv_1^y - mv_0^y)}{w}y + mv_0^x \\ mv^y(x,y) = \dfrac{(mv_1^y - mv_0^y)}{w}x + \dfrac{(mv_1^y - mv_0^x)}{w}y + mv_0^y \end{cases} \tag{7}$$

Para definir a localização exata das amostras do quadro referência que serão utili-

zadas para reconstruir a CB atual, é necessário obter o MV de cada amostra a ser reconstruída. Assim, a Equação (7) é utilizada para derivar o MV de cada amostra (x, y) a ser reconstruída para a predição *Affine* com quatro parâmetros, onde (mv_0^x, mv_0^y) é o MV do PC0, (mv_1^x, mv_1^y) é o MV do PC1, e w e h representam a largura e altura da CB atual, respectivamente (CHEN; YE; KIM, 2020).

Já a Equação (8) é utilizada para derivar o MV de cada amostra (x, y) a ser reconstruída para a predição *Affine* com seis parâmetros. A diferença para a Equação (7) está no (mv_2^x, mv_2^y), que representa o MV do PC2.

$$\begin{cases} mv^x(x,y) = \dfrac{(mv_1^x - mv_0^x)}{w}x + \dfrac{(mv_2^x - mv_0^x)}{h}y + mv_0^x \\ mv^y(x,y) = \dfrac{(mv_1^y - mv_0^y)}{w}x + \dfrac{(mv_2^y - mv_0^y)}{h}y + mv_0^y \end{cases} \tag{8}$$

A reconstrução amostra por amostra, isto é, aplicar as Equações (7) e (8) para obter o MV de cada amostra a ser reconstruída, demanda um alto custo computacional. Então, o VVC adota a reconstrução baseada em sub-blocos 4×4. A Figura 13 apresenta um exemplo da divisão de uma CB 16×16 em sub-blocos 4×4. Para cada sub-bloco 4×4, a predição *Affine* herda o MV do centro deste sub-bloco, utilizando as Equações (7) ou (8). Assim, o MV do centro do sub-bloco é utilizado para herdar todas as amostras do sub-bloco 4×4. Em outras palavras, após obter o MV do centro do sub-bloco 4×4, uma compensação de movimento (*Motion Compensation* - MC) convencional é aplicada para reconstruir o sub-bloco 4×4 e, consequentemente, reconstruir a CB atual (BROSS et al., 2021b) (CHEN; YE; KIM, 2020).

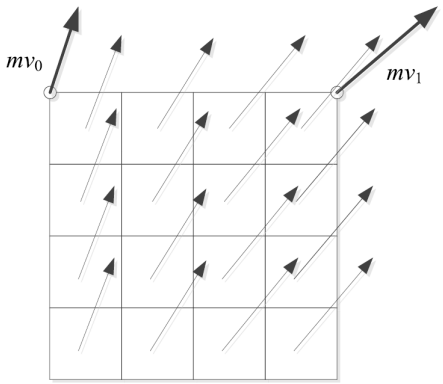


Figura 13 – MVs dos sub-blocos 4×4 dentro de uma CB 16×16. Fonte: (ZHANG et al., 2019)

No VVC, os MVs dos sub-blocos 4×4 tem precisão de $1/16$ de amostra. Assim, a partir das amostras de posições inteiras do quadro de referência, são geradas amostras em posições fracionárias utilizando um conjunto de 15 filtros de interpolação. No caso da MC para blocos 4×4, o VVC emprega o uso de filtros de interpolação de 6-*taps* (BROSS et al., 2021b).

2.2.2.2 Sinalização e Busca pelos MVs dos PCs

Existem dois modos para obter os MVs a serem utilizados pela reconstrução *Affine*. Estes dois modos são chamados de *Affine Merge-mode* e *Affine Inter-mode*.

No *Affine Merge-mode*, nenhum MV é sinalizado explicitamente para o decodificador. Neste modo, os MVs para a CB atual são herdados a partir da informação de movimento dos blocos vizinhos. Para isso, é gerada uma lista inserindo os blocos vizinhos que utilizaram predição *Affine*, os blocos vizinhos que utilizaram a ME convencional, além do MV zero (CHEN; YE; KIM, 2020) e, então, o MV a ser empregado para o *Affine Merge-mode* é sinalizado através de sua posição nesta lista.

Já no *Affine Inter-mode*, os MVs a serem utilizados na reconstrução devem ser sinalizados explicitamente. Portanto, é necessário realizar uma busca dentro da SA para obter os MVs que representem o movimento complexo da CB. Inicialmente, um conjunto de MVs é herdado a partir da informação de movimento dos blocos vizinhos para definir o ponto inicial da busca. Esse conjunto de MVs pode ser obtido dos blocos vizinhos que utilizaram predição *Affine*, dos blocos vizinhos que utilizaram a ME convencional, ou do MV zero (CHEN; YE; KIM, 2020).

Então, a partir do ponto inicial, o *software* VTM realiza uma busca iterativa (JVET, 2022) (LI et al., 2018). O Fluxograma desta busca está representado na Figura 14. Nessa busca, o conjunto de MVs herdado dos blocos vizinhos é utilizado para reconstruir a CB temporariamente, e é calculado um erro, que é a diferença entre a CB atual e a CB reconstruída temporariamente.

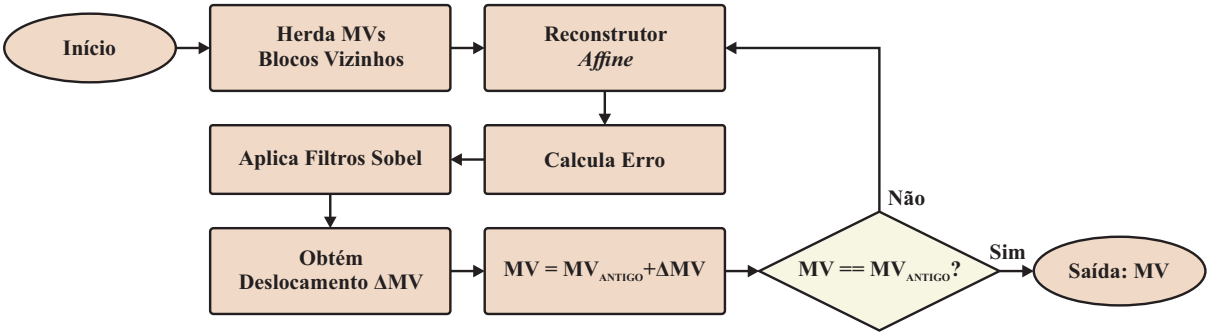


Figura 14 – Fluxo da busca pelos MVs para o *Affine* executada no *software* VTM.

Após o cálculo do erro, são calculados dois blocos de gradiente utilizando filtros *Sobel* sobre a CB reconstruída temporariamente. Filtros *Sobel* são empregados para detecção de bordas em imagens (KANOPOULOS; VASANTHAVADA; BAKER, 1988) e, portanto, o gradiente é um bloco que possui valores nulos nas regiões homogêneas, e valores relevantes nas regiões da CB reconstruída que possuem borda. No *software* VTM, os filtros *Sobel* são aplicados duas vezes sobre a CB reconstruída, gerando um bloco de gradiente horizontal, e um bloco de gradiente vertical. O cálculo do filtro *Sobel* está representado nas Equações (9) e (10), que são utilizadas para obter cada uma das amostras do gradiente horizontal e do gradiente vertical, respectivamente. Nas

Equações (9) e (10), CB_{recons} representa a CB reconstruída, *gradH* e *gradV* representam o gradiente horizontal e vertical, respectivamente, enquanto *j* e *k* representam, respectivamente, a linha e a coluna sendo filtrada e consideram as dimensões da CB atual.

$$\begin{aligned} gradH[j][k] = (&CB_{recons}[j-1][k+1] - CB_{recons}[j-1][k-1] + \\ &(2 * CB_{recons}[j][k+1]) - (2 * CB_{recons}[j][k-1]) + \\ &CB_{recons}[j+1][k+1] - CB_{recons}[j+1][k-1]) \end{aligned} \tag{9}$$

$$\begin{aligned} gradV[j][k] = (&CB_{recons}[j+1][k-1] - CB_{recons}[j-1][k-1] + \\ &(2 * CB_{recons}[j+1][k]) - (2 * CB_{recons}[j-1][k]) + \\ &CB_{recons}[j+1][k+1] - CB_{recons}[j-1][k+1]) \end{aligned} \tag{10}$$

Estes blocos de gradiente, juntamente com o erro, são utilizados pela etapa que obtém o deslocamento ΔMV. Esta etapa gera inicialmente uma matriz de coeficientes com tamanho de 7×7, embora o número de coeficientes válidos nesta matriz depende do número de parâmetros do *Affine*. Para obter os coeficientes desta matriz, a equação (11) é utilizada, na qual col e $linha$ representam cada posição desta matriz de coeficientes e consideram posições de 0 até 4 ou 6 (de acordo com o número de parâmetros do *Affine*), enquanto o valor de iC é dado pela equação (12) ou (13), de acordo com o número de parâmetros utilizado pelo *Affine*. A última linha desta matriz de coeficientes é preenchida utilizando a Equação (14), onde $ultimaLinha$ pode ser 4 ou 6, dependendo do número de parâmetros do *Affine*.

$$mtCoeff[col+1][linha] = \sum_{j=0}^{CB_{altura}} \sum_{k=0}^{CB_{largura}} int(iC[col] * iC[linha]) \tag{11}$$

$$\begin{aligned} iC_{4param} = [&gradH[j][k], \\ &(((k >> 2) << 2) + 2) * gradH[j][k] + (((j >> 2) << 2) + 2) * gradV[j][k], \\ &gradV[j][k], \\ &(((j >> 2) << 2) + 2) * gradH[j][k] - (((k >> 2) << 2) + 2) * gradV[j][k]] \end{aligned} \tag{12}$$

$$
\begin{aligned}
iC_{6param} = [&gradH[j][k], \\
&(((k >> 2) << 2) + 2) * gradH[j][k], \\
&gradV[j][k], \\
&(((k >> 2) << 2) + 2) * gradV[j][k], \\
&(((j >> 2) << 2) + 2) * gradH[j][k], \\
&(((j >> 2) << 2) + 2) * gradV[j][k]]
\end{aligned} \tag{13}
$$

$$
mtCoeff[col + 1][ultimaLinha] = \sum_{j=0}^{CB_{altura}} \sum_{k=0}^{CB_{largura}} int(iC[col] * erro[j][k]) \tag{14}
$$

Por fim, após obtida a matriz de coeficientes, algumas operações são aplicadas pelo *software* VTM sobre estes coeficientes, obtendo assim o deslocamento ΔMV. O algoritmo executado pelo *software* VTM para realizar estas operações está apresentado no Algoritmo 1.

Algoritmo 1 - Algoritmo para converter matriz de coeficientes no deslocamento $Delta$MV

Require: $mtCoeff$ ▷ Matriz de coeficientes
Require: $iOrder$ ▷ Quantidade de parâmetros do *Affine*
Require: CB_{altura} ▷ Altura da CB
Require: $CB_{largura}$ ▷ Largura da CB

```
float dAffinePara[iOrder] ← 0
int DeltaMV[3][2] ← 0
float tempMv[6] ← 0
for i ← 1, iOrder, i++ do
    temp ← abs(mtCoeff[i][i − 1])
    tempIdx ← i
    for j ← i + 1, iOrder + 1, j++ do
        if abs(mtCoeff[j][i − 1]) > temp then
            temp ← abs(mtCoeff[j][i − 1])
            tempIdx ← j
        end if
    end for
    if tempIdx ≠ i then
        for j ← 0, iOrder + 1, j++ do
```

$mtCoeff[0][j] \leftarrow mtCoeff[i][j]$
$mtCoeff[i][j] \leftarrow mtCoeff[tempIdx][j]$
$mtCoeff[tempIdx][j] \leftarrow mtCoeff[0][j]$
end for
end if
if $mtCoeff[i][i-1] = 0$ **then**
return
end if
for $j \leftarrow i+1, iOrder+1, j++$ **do**
for $k \leftarrow i, iOrder+1, k++$ **do**
$val \leftarrow ((mtCoeff[i][k] * mtCoeff[j][i-1])/mtCoeff[i][i-1])$
$mtCoeff[j][k] \leftarrow mtCoeff[j][k] - val$
end for
end for
end for
if $mtCoeff[iOrder][iOrder-1] = 0$ **then**
return
end if
$dAffinePara[iOrder-1] \leftarrow mtCoeff[iOrder][iOrder]/mtCoeff[iOrder][iOrder-1]$
for $i \leftarrow iOrder-2, -1, i--$ **do**
if $mtCoeff[i+1][i] = 0$ **then**
for $k \leftarrow 0, iOrder, k++$ **do**
$dAffinePara[k] \leftarrow 0$
end for
break
end if
$temp \leftarrow 0$
for $j \leftarrow i+1, iOrder, i++$ **do**
$temp \leftarrow temp + (mtCoeff[i+1][j] * dAffinePara[j])$
end for
$dAffinePara[i] \leftarrow (mtCoeff[i+1][iOrder] - temp)/mtCoeff[i+1][i]$
end for
$tempMv[0] \leftarrow dAffinePara[0]$
$tempMv[2] \leftarrow dAffinePara[2]$
if $iOrder = 6$ **then**
$tempMv[1] \leftarrow (dAffinePara[1] * CB_{largura}) + dAffinePara[0]$
$tempMv[3] \leftarrow (dAffinePara[3] * CB_{largura}) + dAffinePara[2]$
$tempMv[4] \leftarrow (dAffinePara[4] * CB_{altura}) + dAffinePara[0]$
$tempMv[5] \leftarrow (dAffinePara[5] * CB_{altura}) + dAffinePara[2]$

else

$tempMv[1] \leftarrow (dAffinePara[1] * CB_{largura}) + dAffinePara[0]$

$tempMv[3] \leftarrow (-dAffinePara[3] * CB_{largura}) + dAffinePara[2]$

end if

$val0 \leftarrow int(tempMv[0] * 4 + SIGN(tempMv[0]) * 0.5) << 2$

$val1 \leftarrow int(tempMv[1] * 4 + SIGN(tempMv[1]) * 0.5) << 2$

$val2 \leftarrow int(tempMv[2] * 4 + SIGN(tempMv[2]) * 0.5) << 2$

$val3 \leftarrow int(tempMv[3] * 4 + SIGN(tempMv[3]) * 0.5) << 2$

$val4 \leftarrow int(tempMv[4] * 4 + SIGN(tempMv[4]) * 0.5) << 2$

$val5 \leftarrow int(tempMv[5] * 4 + SIGN(tempMv[5]) * 0.5) << 2$

$DeltaMV[0] \leftarrow [val0, val2]$

$DeltaMV[1] \leftarrow [val1, val3]$

if $iOrder = 6$ **then**

$DeltaMV[2] \leftarrow [val4, val5]$

end if

return $DeltaMV$

O processo iterativo da busca pelos MVs finaliza caso o novo MV seja idêntico ao MV anterior, ou caso o processo atinja o limite de MVs avaliados. Esse limite de MVs avaliados varia entre cinco a oito MVs, de acordo com a predição uni-direcional ou bi-direcional e o *Affine* quatro parâmetros ou seis parâmetros (JVET, 2022) (LI et al., 2018). Porém, um refinamento ainda pode ser realizado sobre cada um dos MVs encontrados, realizando pequenas variações nas suas posições (JVET, 2022). Assim, simulações realizadas com o *software* VTM mostram que, considerando este refinamento, até 82 conjuntos de MVs podem ser avaliados para representar a CB atual utilizando a predição *Affine*, o que indicam até 82 reconstruções *Affine* a serem realizadas para cada CB.

2.2.2.3 Acessos à Memória

De maneira similar com as requisições à memória realizadas pela IME, como descrito na Seção 2.2.1.3, o processamento do *Affine* ME também demanda uma grande comunicação com a memória.

O número de amostras necessárias para realizar a reconstrução *Affine* e cálculo do erro, para um conjunto de MVs também depende do número de amostras da CB atual, sendo dado por V na Equação (3).

Considerando o processamento de cada conjunto de MVs por todas as CBs, o número de amostras que precisam ser requisitadas para a reconstrução de cada conjunto de MVs por todas as CBs de mesmo tamanho, também depende do número de CBs do respectivo tamanho que cabem dentro da CTB, e o número de amostras

necessárias para processamento de cada CB, sendo dado por G na Equação (5).

Sendo assim, ao considerar que, de acordo com as simulações realizadas, o *software* VTM realiza a avaliação de até 82 conjuntos de MVs para cada CB, que existem 12 tamanhos de CB na qual o *Affine* ME é avaliado, que o *Affine* pode possuir quatro ou seis parâmetros, além de considerar o *Affine* ME sendo realizado para um quadro de referência de cada lista, até 128,9 milhões de amostras podem ser requisitadas pelo *Affine* ME, como dado por J na Equação (15).

$$J = 82 * 12 * 2 * 2 * G(CB_{altura}, CB_{largura}) \tag{15}$$

Desta forma, considerando o pior caso do *Affine* ME durante o processamento de vídeos UHD 4K a 30 qps, processar o *Affine* ME com quatro e seis parâmetros para todas as CTBs de um canal de informação, pode demandar até 1,97TB/s de comunicação com a memória.

3 TECNOLOGIAS DE MEMÓRIA

Um dos objetivos deste trabalho é o desenvolvimento de um sistema de predição interquadros do padrão VVC em *hardware* com foco no projeto eficiente da memória interna. Por conta disso, este capítulo apresenta as características do funcionamento de diferentes tecnologias de memória. Inicialmente, a Seção 3.1 apresenta o conceito básico de um circuito de memória. Na sequencia, as próximas seções apresentam o funcionamento das operações precisas em quatro tecnologias de memória diferentes. A Seção 3.2 apresenta duas tecnologias de memória voláteis, enquanto a Seção 3.3 apresenta duas tecnologias de memória não-voláteis. Já a Seção 3.4 apresenta uma comparação das principais características destas quatro tecnologias de memória. Estas quatro tecnologias de memória abordadas neste capítulo são as tecnologias nas quais os trabalhos encontrados na literatura já exploraram algum tipo de armazenamento aproximado. Assim, na Seção 4.3, no próximo capítulo, serão apresentados alguns trabalhos da literatura que exploram o uso de operações imprecisas nestas quatro tecnologias de memória apresentadas neste capítulo.

3.1 Estrutura de Memória

Estruturas de memória são circuitos elétricos capazes de armazenar informação e de entregar essa informação para a unidade de processamento, quando requisitados. Para isso, um circuito de memória recebe o endereço da informação desejada e encontra a respectiva célula de armazenamento deste endereço, dentre os milhões ou até bilhões de células de armazenamento (YU; WANG, 2014) que compõem um circuito de memória. Então, sobre a célula encontrada serão realizadas as operações de leitura ou escrita.

Em memórias de acesso randômico (*Random-Access Memory* - RAM), as células de armazenamento são agrupadas em um *Data Array* (DA), e vários DAs são utilizados para compor o circuito de memória. Além disso, esses DA são todos organizados de forma que o tempo de acesso à qualquer DA seja similar, independentemente de sua localização dentro da memória. Isso garante que o tempo de acesso seja deter-

minístico para qualquer endereço da memória (YU; WANG, 2014).

Uma estrutura genérica de um DA é exibida na Figura 15. Como pode ser visto, cada DA possui diversas células de memória. Existem diferentes tecnologias para o armazenamento de *bits* dentro de uma célula de memória. Cada célula está localizada em uma intersecção entre uma das *bitlines* (BL) e uma das *wordlines* (WL), e cada célula é responsável pelo armazenamento de apenas um *bit* da informação. Assim, os demais componentes exibidos na Figura 15 são utilizados para encontrar as células referenciadas pelo endereço, e para ler a informação contida na célula ou escrever uma nova informação nesta célula.

Para encontrar as células apontadas pelo endereço, o endereço requisitado é dividido em duas partes. Uma das partes do endereço é utilizada pelo Decodificador de WL, que realiza a operação de leitura em todas as células da WL correspondentes ao endereço solicitado. A segunda parte do endereço é utilizada pelo Decodificador de BL, possibilitando que o Multiplexador de Coluna escolha apenas parte das células da WL que foi habilitada.

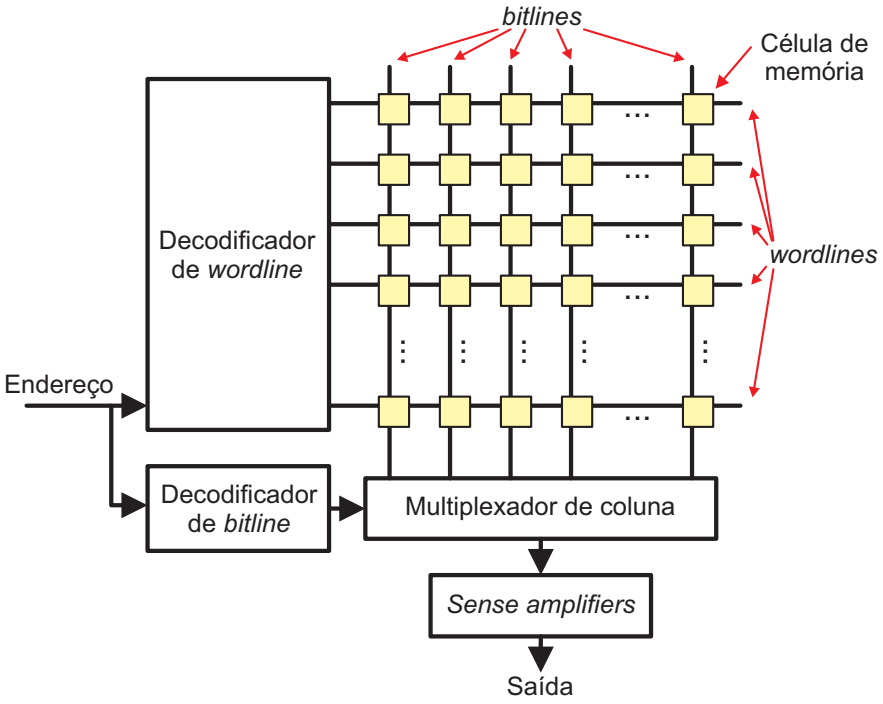


Figura 15 – Estrutura genérica de um *Data Array*. Adaptado de (YU; WANG, 2014)

Devido ao compartilhamento de carga entre uma célula de memória e a capacitância parasita de sua BL, operações de leitura resultam em variações muito pequenas na tensão de cada BL. Por conta disso, um *Sense Amplifier* precisa ser empregado na saída de cada BL para realizar as operações de leitura. Cada *Sense Amplifier* é um circuito elétrico capaz de detectar pequenas variações nos níveis de tensão de cada BL com relação a uma tensão de referência, amplificando esse valor. Assim, nas operações de leitura, o *Sense Amplifier* realiza a amplificação dessa variação na

tensão para o estado lógico do valor armazenado na célula lida, entregando a informação armazenada. Cabe notar que cada tecnologia de memória possui características diferentes de como o armazenamento da informação é realizado, o que pode exigir pequenas alterações no circuito dos *Sense Amplifiers* para a leitura correta da informação armazenada.

3.2 Memórias Voláteis

Memórias voláteis são memórias que armazenam a informação como sinal elétrico, isso é, os estados lógicos da informação são representados através do armazenamento dos níveis de tensão de cada *bit* (YU; WANG, 2014). Sendo assim, memórias voláteis dependem de energia elétrica constante para que a informação seja armazenada corretamente, acarretando na perda da informação armazenada quando o circuito elétrico é desligado (YU; WANG, 2014).

As memórias voláteis mais comuns são a memória de acesso randômico dinâmica (*Dynamic RAM* - DRAM) e a memória de acesso randômico estática (*Static RAM* - SRAM), que serão abordadas abaixo.

3.2.1 *Dynamic RAM* (DRAM)

A DRAM é a memória de construção mais básica. A Figura 16 representa uma célula de memória DRAM. Como pode ser visto, cada célula DRAM é composta apenas por um capacitor de armazenagem, o qual é responsável por armazenar o *bit* da informação, e também um transistor de acesso, o qual permite o acesso ao capacitor de armazenagem para a realização das operações de leitura/escrita das informações.

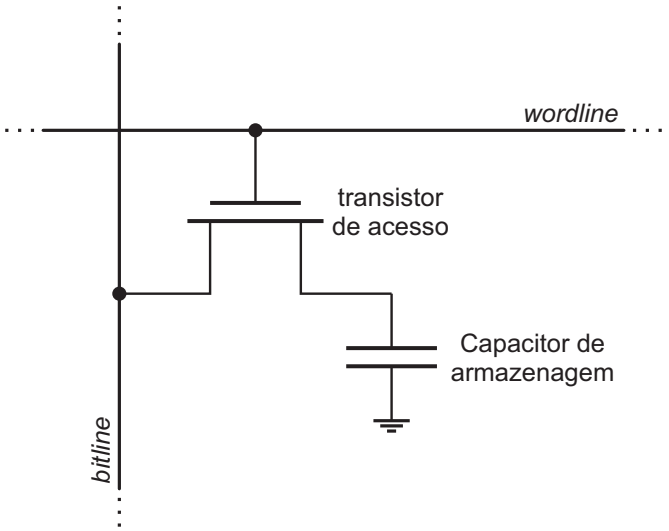


Figura 16 – Diagrama elétrico de uma célula de memória de tecnologia DRAM.

Para realizar uma operação de escrita e armazenar uma nova informação no capacitor, o nível de tensão respectivo ao estado lógico da informação a ser armazenada

deve ser carregado na trilha BL de todas as células a serem escritas, depois disso, o transistor de acesso das células deve ser ativado, a partir do carregamento da trilha WL respectiva ao endereço desejado, escrevendo, portanto, o novo nível de tensão da trilha BL no capacitor da célula.

Para realizar uma operação de leitura, basta carregar a trilha WL, ativando o transistor. Isso fará com que a carga elétrica no capacitor chegue na trilha BL e, com isso, a informação armazenada na célula possa ser transferida até o *sense amplifier*. Este realizará a leitura e a entrega dessa informação para a unidade de processamento. Contudo, em células DRAM, as leituras são destrutivas, isto é, a informação do capacitor é perdida após a realização de cada leitura (YU; WANG, 2014). Portanto, para manter a informação armazenada e permitir leituras futuras, é necessário realizar uma operação de escrita logo após uma leitura.

Células DRAM também sofrem pelo fato de que a carga elétrica do capacitor é perdida com o tempo, visto que sua implementação depende de transistores que acabam apresentando uma corrente de fuga. Com isso, podem ocorrer variações na informação armazenada no capacitor. Isso é, caso o capacitor possua carga elétrica respectiva ao estado lógico '1', essa carga elétrica é perdida com o tempo, fazendo com que a carga elétrica armazenada passe a ser considerada como o estado lógico '0'. Para evitar a perda da informação, a informação de cada célula deve ser lida e reescrita, antes da perda da informação, garantindo que a informação armazenada esteja sempre correta. Por conta disso, memórias DRAM possuem uma operação de *refresh*, que é realizada periodicamente, e que serve justamente para realizar, sequencialmente, ambas as operações de leitura e escrita em cada célula de memória. Tipicamente, a operação de *refresh* é executada periodicamente em cada célula DRAM a cada 32ms ou 64ms (YU; WANG, 2014).

Enquanto uma operação de *refresh* está sendo realizada, não é possível realizar operações de leitura ou escrita na memória (LIU et al., 2012). Com isso, podem ocorrer atrasos adicionais nas demais operações de leitura ou escrita na DRAM. Além disso, operações de *refresh* são um dos grandes vilões do consumo de energia de memórias DRAM. Estima-se que nas tecnologias atuais, o consumo das operações de *refresh* seja de mais de 50% do consumo total de energia das memórias DRAM (LIU et al., 2012) (KIM et al., 2020).

3.2.2 *Static RAM* (SRAM)

A SRAM é uma memória construída a partir de um arranjo de transistores para o armazenamento da informação. A Figura 17 apresenta uma célula de memória SRAM composta por seis transistores. Ao contrário da DRAM, a qual armazena a informação em um capacitor que perde essa informação com o tempo, células SRAM são mais estáveis e não perdem a informação com o tempo e, portanto, não necessitam de

operações de *refresh*. Contudo, a estabilidade da informação numa célula de memória SRAM depende de uma fonte de energia estável, perdendo a informação armazenada assim que a energia é desligada (YU; WANG, 2014).

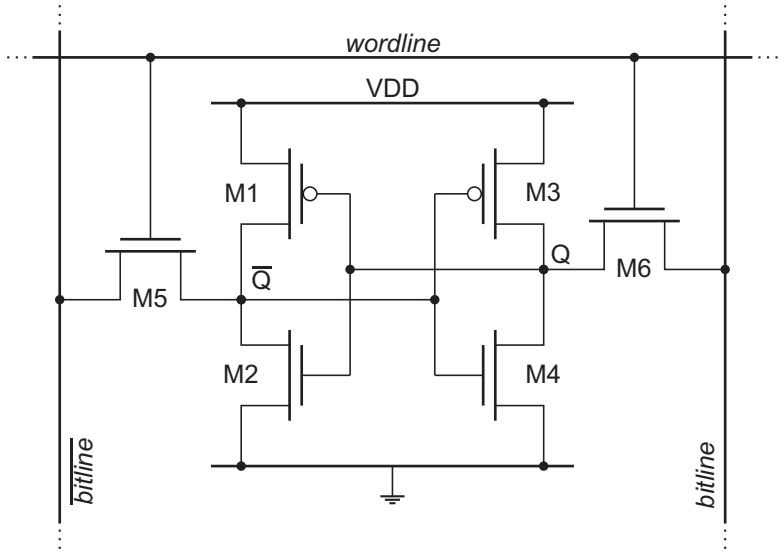


Figura 17 – Diagrama elétrico de uma célula de memória de tecnologia SRAM composta por seis transistores.

Como pode ser visto na Figura 17, os transistores *T1-T2* e *T3-T4* são dois inversores que se realimentam e assim formam o circuito responsável pelo armazenamento da informação dentro da célula SRAM. Já os transistores *T5-T6* realizam o acesso externo a essa informação, permitindo assim a leitura e escrita da informação nas células.

Para realizar uma operação de escrita, é necessário inserir a informação nas duas BL, na qual $\overline{bitline}$ recebe o complemento da informação a ser armazenada. Então, basta habilitar a respectiva WL, o que permite com que a nova informação sobrescreva a informação existente na célula.

Já para realizar uma operação de leitura, ambas as BL são pré-carregadas com o valor lógico '1'. Então, é habilitado a respectiva WL, o que faz com que a informação armazenada chegue em ambas as BL, fazendo com que uma dessas BL tenha uma pequena variação em sua tensão, visto que Q ou $\overline{Q}$ será '0'. Essa variação na tensão é percebida pelo *sense amplifier*, que reconhece qual das duas BL apresentou a variação e, portanto, consegue reconhecer o valor de Q e de $\overline{Q}$.

Quando uma célula não está executando uma operação de leitura ou de escrita, isto é, a respectiva WL está desabilitada, ela está em modo de espera (*stand-by*). Nesse modo, os dois transistores de acesso (*T5-T6*) estão desabilitados e, portanto, os dois inversores se realimentam, mantendo a informação armazenada dentro da célula.

A tecnologia SRAM permite uma alta performance para as operações de leitura e escrita. Contudo, ela demanda uma grande área quando comparada a células DRAM.

Além disso, as células SRAM possuem um alto consumo de energia quando em modo de espera. Células SRAM sofrem com o *subthreshold leakage*, que é a corrente de dreno para fonte do transistor quando a tensão entre o *gate* e a fonte do transistor é menor do que a tensão de *threshold* (V_{TH}) do transistor (YU; WANG, 2014). Em modo de espera, sempre existem três transistores da célula SRAM nesse estado e, consequentemente, resultando no consumo *subthreshold leakage*.

3.3 Memórias Não-Voláteis

Enquanto memórias voláteis armazenam a informação na forma de tensão elétrica, perdendo, portanto, a informação armazenada com o tempo ou com o desligamento da energia, memórias não-voláteis utilizam outras estratégias para armazenar a informação. Assim, memórias não-voláteis não dependem da alimentação do circuito para a manutenção da informação armazenada (YU; WANG, 2014) e, portanto, conseguem garantir o armazenamento da informação por mais de 20 anos em algumas tecnologias (BARLA; JOSHI; BHAT, 2021).

O uso das memórias não-voláteis vem sendo explorado no projeto de sistemas digitais, tanto em memórias externas quanto em memórias internas. Nesse contexto, existem duas tecnologias de memórias não-voláteis nas quais estratégias de armazenamento aproximado são exploradas por trabalhos da literatura: *Spin-Transfer Torque Magnetic RAM* (STT-RAM) e *Phase Change RAM* (PC-RAM). Assim, essas duas tecnologias são explicadas nas subseções abaixo.

3.3.1 *Spin-Transfer Torque Magnetic RAM* (STT-RAM)

As células das memórias STT-RAM armazenam a informação utilizando propriedades magnéticas. Para isso, as células STT-RAM possuem um componente que permite que a orientação magnética deste componente seja modificada através da aplicação de uma corrente elétrica. Assim, depois de definida a orientação magnética, uma célula pode ficar longos períodos de tempo, de mais de 20 anos (BARLA; JOSHI; BHAT, 2021), com a energia desligada sem perder a informação armazenada (YU; WANG, 2014) (CHEN et al., 2010).

A Figura 18 apresenta uma célula de memória de tecnologia STT-RAM. Como pode ser visto, uma célula STT-RAM é composta de um *Magnetic Tunnel Junction* (MTJ), para o armazenamento da informação, e um transistor de acesso para permitir leituras e escritas no MTJ. Além das linhas WL e BL, uma célula STT-RAM necessita de uma *sourceline* (SL) para injetar as correntes de leitura e escrita.

Como pode ser visto na Figura 18-(b), o MTJ possui uma camada fixa do lado mais próximo do transistor (onde a orientação magnética não é variável), um material isolante, e uma camada livre (que permite a alteração de sua orientação magnética

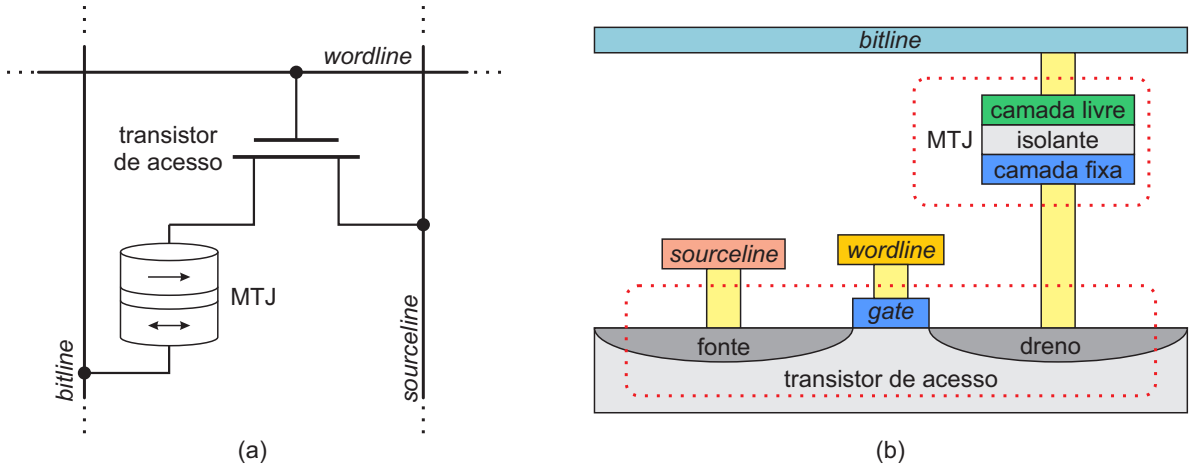


Figura 18 – Célula de memória de tecnologia STT-RAM: (a) Diagrama elétrico de uma célula; (b) Leiaute físico da célula.

através da aplicação de uma corrente elétrica). Desta forma, a resistência elétrica deste MTJ depende da orientação magnética da camada livre, a qual pode ser alterada, permitindo assim representar a informação dentro da célula STT-RAM (CHEN et al., 2010) (KÜLTÜRSAY et al., 2013).

Assim, para realizar uma escrita em uma célula STT-RAM, uma corrente deve ser aplicada sobre o MTJ. Esta corrente de escrita deve ser maior do que a corrente crítica do MTJ, que é o valor de corrente no qual a camada livre do MTJ muda sua orientação magnética (NA; KANG; JUNG, 2021). Desta forma, a corrente aplicada é capaz de alterar a orientação magnética da camada livre do MTJ para a orientação desejada (KÜLTÜRSAY et al., 2013). Neste caso, a camada fixa gera um efeito de spin torque de acordo com a direção da corrente aplicada, sendo que esse torque age sobre a magnetização da camada livre do MTJ (YU; WANG, 2014) (HOSOMI et al., 2005). Para isso, deve-se aplicar uma tensão entre a BL e a SL, e então aplicar um pulso sobre a WL para habilitar o transistor de acesso, fazendo fluir sobre o MTJ uma corrente maior do que a corrente crítica do MTJ.

Para realizar uma operação de leitura é aplicada uma pequena tensão entre a BL e a SL e, após, o transistor de acesso é habilitado. Então, o *sense amplifier* percebe a corrente resultante do processo, obtendo assim a resistência do MTJ e, consequentemente, a informação armazenada (KÜLTÜRSAY et al., 2013). Durante as leituras, a corrente resultante entre a BL e a SL deve ser pequena o suficiente para não ocasionar uma escrita, logo, essa corrente deve ser menor do que a corrente crítica do MTJ. Com isso, é possível realizar uma leitura em uma célula STT-RAM sem destruir a informação armazenada na célula, isto é, sem afetar a resistência do MTJ (KÜLTÜRSAY et al., 2013).

3.3.2 *Phase Change RAM* (PC-RAM)

De maneira similar à célula de tecnologia STT-RAM, uma célula de memória PC-RAM armazena a informação alterando a resistência elétrica do composto da célula. Para isso, uma célula de memória PC-RAM é composta de um material que permite dois estados estáveis diferentes, sendo que em cada um desses estados possui uma resistência diferente. Após definido o estado do composto, uma célula PC-RAM pode permanecer longos períodos de tempo sem energia mantendo o seu valor armazenado, podendo ultrapassar os 10 anos sem energia (WONG et al., 2010) (BARLA; JOSHI; BHAT, 2021).

A Figura 19 apresenta o diagrama elétrico e também o leiaute físico de uma célula PC-RAM (WONG et al., 2010), que possui o formato de cogumelo. Como pode ser visto, o diagrama elétrico é bem similar a uma célula de memória de tecnologia DRAM. O composto utilizado na PC-RAM permite alterar sua resistência a partir do aquecimento desse material. Para isso, uma célula PC-RAM possui um terminal aquecedor conectado ao composto da PC-RAM, como pode ser visto na Figura 19-(b).

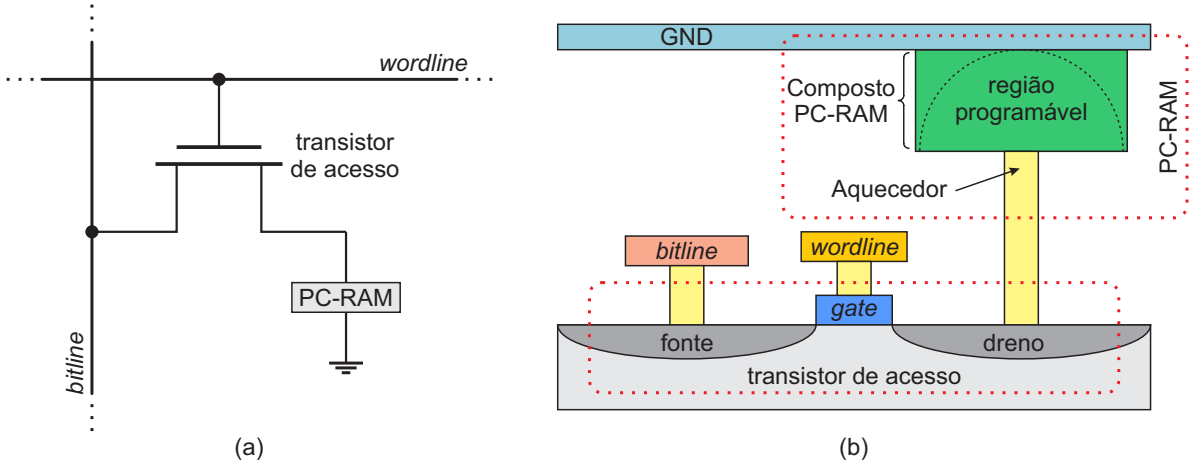


Figura 19 – Célula de memória de tecnologia PC-RAM: (a) Diagrama elétrico de uma célula; (b) Leiaute físico da célula.

Para escrever o valor lógico ’0’, é necessário elevar o composto PC-RAM para uma alta temperatura, por um curto período de tempo, a partir da aplicação de um curto pulso com uma grande corrente elétrica (WONG et al., 2010), sendo essa corrente cerca de sete vezes maior do que a corrente utilizada nas leituras da resistência da PC-RAM (LEE et al., 2010). Isso faz com que a região programável da PC-RAM resulte no estado *amorphous*, no qual o componente PC-RAM possui uma alta resistência elétrica (WONG et al., 2010).

Já para escrever o valor lógico ’1’, é necessário aquecer o composto PC-RAM para uma temperatura média (menor do que a de escrita do valor lógico ’0’), por um longo período de tempo. Para isso, é necessário um longo pulso com uma corrente elétrica de aproximadamente metade do valor de corrente necessária para escrever

o valor lógico ’0’ (WONG et al., 2010) (LEE et al., 2010). Isso faz com que a região programável da PC-RAM resulte no estado cristalino, no qual o componente PC-RAM possui uma baixa resistência elétrica (WONG et al., 2010).

Após realizada a escrita de uma informação e definida a resistência do composto PC-RAM, leituras podem ser realizadas de maneira similar a uma célula STT-RAM. Para isso, é aplicada uma pequena tensão entre BL e o terminal GND conectado no composto PC-RAM, e então o transistor de acesso é habilitado pela ativação da WL. O *sense amplifier* percebe a corrente resultante do processo, obtendo assim a resistência da PC-RAM e convertendo essa resistência para o valor lógico da informação armazenada na célula (WONG et al., 2010). A corrente aplicada em uma leitura deve ser pequena o suficiente para não ocasionar uma escrita do valor lógico ’1’, assim, sem destruir a informação armazenada na célula (WONG et al., 2010).

3.4 Comparação de Tecnologias

Essa seção apresenta uma comparação das principais características entre as quatro tecnologias de memória apresentadas anteriormente, de acordo com os dados presentes na Tabela 1.

Tabela 1 – Comparação entre as características de diferentes tecnologias de memória. Fonte: (MITTAL, 2016) (BARLA; JOSHI; BHAT, 2021) (KIM et al., 2016)

	Memória Volátil		Memória Não-Volátil	
	DRAM	SRAM	STT-RAM	PC-RAM
Área da Célula[1] (F^2)	20 $\sim$ 100	50 $\sim$ 200	6 $\sim$ 50	4 $\sim$ 12
Latência (Leitura)	Curta	Muito Curta	Curta	Curta
Latência (Escrita)	Curta	Muito Curta	Longa	Muito Longa
Energia (Leitura)	Média	Baixa	Média	Média
Energia (Escrita)	Média	Baixa	Alta	Muito Alta
Consumo Estático	Baixo	Alto	$\sim$ 0	$\sim$ 0
Tempo de Retenção	0	0	>20 Anos	>10 Anos
Número de Escritas	$> 10^{15}$	$> 10^{15}$	$> 10^{15}$	$10^8 \sim 10^{12}$
Principal Desafio	*Refresh*	*Leakage*	Escritas	Escritas

[1] Área da célula dada em F^2, o menor recurso que pode ser criado na tecnologia empregada.

Como pode ser visto na Tabela 1, células DRAM possuem área, latência e um consumo de energia com valores intermediários dentre as tecnologias apresentadas. As células DRAM precisam da operação de *refresh* constante, elevando assim o consumo estático dessas células e podendo aumentar também a latência das demais

operações sendo, portanto, o principal ponto abordado por trabalhos da literatura que buscam avanços para esta tecnologia.

As células SRAM são as que possuem a maior área, visto que tipicamente cada célula de memória para o armazenamento de um *bit* precisa de ao menos seis transistores. Contudo, por possuir apenas transistores, ela também apresenta a menor latência nas operações de leitura e escrita, além do menor consumo de energia nestas operações. A informação permanece armazenada nas células SRAM apenas enquanto a célula possuir uma fonte de alimentação. Por conta disso, células SRAM apresentam um alto consumo estático, sendo esse consumo estático o principal problema relacionado com o uso dessa tecnologia.

As células STT-RAM, conforme apresentado na Tabela 1, possuem uma pequena área quando comparadas com as demais tecnologias, enquanto para as leituras elas apresentam uma latência e um consumo de energia medianos. Já para escrita, é necessário aplicar uma pequena tensão entre a BL e a SL, para assim resultar em uma corrente sobre o MTJ que altera a orientação magnética da camada livre do MTJ, resultando assim em uma maior latência e também um alto consumo de energia nas operações de escrita. A vantagem de células STT-RAM é que depois de escrita a informação, os dados ficam armazenados na célula por mais de 20 anos (BARLA; JOSHI; BHAT, 2021), além de apresentarem um consumo estático quase nulo.

Por último, as células PC-RAM apresentam a menor área se comparado com as demais tecnologias apresentadas. Para as operações de leitura, células PC-RAM apresentam uma latência e consumo de energia similar ao das células STT-RAM. Quando comparado com as tecnologias de memórias voláteis, leituras nas células PC-RAM apresentas valores medianos de latência e consumo de energia. O grande problema das células PC-RAM são as escritas, que apresentam uma latência muito longa e um consumo de energia excessivamente alto, visto que as células PC-RAM precisam esquentar o seu composto para realizar a escrita da informação. Outra desvantagem da PC-RAM é o suporte a um menor número de escritas se comparado com as demais tecnologias apresentadas, visto que o aquecimento da célula durante a escrita acaba degradando a célula (KIM et al., 2016). De maneira similar com as células STT-RAM, depois de escrita a informação, os dados ficam armazenados na célula por um longo período de tempo, que pode ser de mais de 10 anos (BARLA; JOSHI; BHAT, 2021), também apresentando um consumo estático quase nulo.

4 TRABALHOS RELACIONADOS

Este capítulo apresenta alguns trabalhos disponíveis na literatura que propõem soluções que permitem obter uma redução no consumo de energia no projeto de *hardware* dedicado para a predição interquadros. São apresentados trabalhos propondo arquiteturas dedicadas para a ME, os quais empregam estratégias e otimizações para reduzir o custo computacional destas arquiteturas. Também serão apresentados trabalhos propondo estratégias para redução do custo computacional dos algoritmos empregados na predição interquadros, trabalhos que empregam estratégias de aproximação para as memórias, e trabalhos empregando o uso de operadores imprecisos dentro de unidades específicas. Além do tema dos trabalhos, o foco das buscas foi também direcionado para os trabalhos que apresentavam soluções para os padrões de codificação HEVC e VVC, que são os padrões mais recentes.

Este capítulo também apresenta alguns trabalhos disponíveis na literatura que, de alguma forma, conseguem privilegiar a memória do sistema de ME, seja reduzindo o número de acessos a memória, ou ainda explorando a imprecisão na unidade de memória, o que permite redução no consumo de energia do sistema de memória.

Para a descrição dos trabalhos relacionados, será considerada a eficiência de codificação através do BD-Rate (BJONTEGAARD, 2008), que mede o impacto na taxa de *bits* do vídeo para obter a mesma qualidade de imagem objetiva. Um valor negativo de BD-Rate representa um aumento na eficiência de codificação da solução proposta, enquanto que um valor positivo de BD-Rate representa uma redução na eficiência de codificação.

4.1 Arquiteturas Dedicadas para a Estimação de Movimento

Como já apresentado, a ME do VVC inclui, além da ME convencional presente nos codificadores anteriores, também o *Affine* ME. Como o Affine ME é uma novidade, são poucos os trabalhos que apresentam soluções de redução de custo computacional ou ainda soluções de *hardware* dedicado para o *Affine* ME do VVC. Contudo, existem diferentes estratégias de redução do custo computacional para a ME convencional

que podem ser encontradas em trabalhos disponíveis na literatura.

O trabalho Jou; Chang; Chang (2015) apresenta uma arquitetura para a ME do padrão HEVC. Neste trabalho, o algoritmo empregado na ME é similar ao TZS, porém, o algoritmo proposto não possui a etapa de Refinamento. Com isso, a busca termina logo após a Busca Inicial, ou após a execução da etapa *Raster*, sendo o candidato resultante sempre de uma dessas etapas. Esse trabalho adota também a estratégia de avaliar um número menor de tamanhos de bloco. Ele avalia apenas oito dos 24 diferentes tamanhos de PB que são suportados no padrão HEVC, tendo assim uma grande redução na quantidade de processamento e no volume de dados requisitados da memória. Os resultados mostraram que as estratégias propostas para reduzir o custo computacional resultam em um impacto de 5,14% no BD-Rate. O sistema proposto divide o processamento da CTU 64×64 em blocos 16×16 e, com o objetivo de maximizar o seu reuso de dados, armazena resultados parciais do processamento dos blocos 16×16 e mescla o processamento de diferentes tamanhos de PB. Isso permite o compartilhamento de recursos computacionais e a diminuição da dependência de dados. O sistema consegue processar vídeos UHD 4K a 60 qps quando sintetizado para ASIC, utilizando uma biblioteca de 90nm, e trabalhando a uma frequência de 270MHz. Os resultados demonstram que o sistema requer uma área de 778,7k *gates*. Resultados de energia do sistema proposto não foram apresentados.

O trabalho de Fan et al. (2018) apresenta um algoritmo também similar ao TZS para a etapa ME do HEVC. Esse algoritmo implementa a etapa Busca Inicial e também o Refinamento, embora permita apenas uma iteração do Refinamento. O trabalho também propõe uma técnica de compressão com perdas para comprimir os blocos candidatos, buscando reduzir o volume de comunicação com a memória. Contudo, o algoritmo proposto avalia um número maior de blocos candidatos em cada uma de suas etapas: na Busca Inicial são avaliados muito mais blocos candidatos a cada expansão, enquanto a etapa de Refinamento realiza uma busca completa com um SR de $\pm$ sete amostras ao redor do resultado da Busca Inicial. Além disso, esse trabalho suporta o processamento de todos os tamanhos de PB do padrão HEVC. Com isso, os resultados mostram um impacto de -0,5% no BD-Rate. O algoritmo permite adotar estratégias de reuso de operações apenas parcialmente, visto que cada tamanho de PB possui independência para decidir qual bloco candidato deseja aplicar o Refinamento. Esta independência pode resultar em candidatos distintos sendo refinados e, consequentemente, na impossibilidade de compartilhar operações deste processamento entre tamanhos de PB diferentes. A arquitetura apresentada requer o uso de duas memórias SRAM para utilizar a sua estratégia de reuso de dados enquanto garante o processamento de vídeos UHD 4K a 30 qps se sintetizada para ASIC utilizando uma biblioteca de células de 65nm da TSMC. Os resultados mostram que a arquitetura requer uma área de 489,4k *gates*. Para processar vídeos UHD 4K a 30

qps, o sistema precisa operar em uma frequência de 500MHz, na qual os resultados de síntese apresentam uma dissipação de potência de 128,5mW.

Em meu trabalho anterior (PERLEBERG et al., 2018), proponho uma solução para a ME do padrão HEVC, realizando algumas alterações no algoritmo TZS que visam reduzir o seu custo computacional, tornando-o determinístico quanto ao limite de candidatos que serão avaliados. Dentre essas alterações, na etapa Predição, apenas o Preditor Zero (MV para o bloco colocalizado) é avaliado. Além disso, a etapa *Raster* não é realizada e, portanto, os Refinamentos são realizados logo após a etapa Busca Inicial. Com isso, o impacto no BD-Rate foi de 14,19%. A arquitetura proposta suporta apenas o processamento de PBs quadradas, sendo que essa arquitetura possui oito unidades trabalhando em paralelo. Cada uma dessas unidades foi desenvolvida especificamente para processar a IME e a FME de apenas um tamanho de PB e, portanto, cada tamanho de PB é processado de maneira independente. Os resultados da síntese mostraram que a arquitetura consegue atingir o processamento de vídeos UHD 8K (7680×4320 amostras) com até 53 qps quando sintetizado para ASIC utilizando a biblioteca de 45nm da Nangate, e operando a uma frequência de 1,02GHz. Os resultados de síntese mostram que a arquitetura proposta requer 18,1M *gates* de recursos computacionais, e também mostram uma dissipação de potência de 426,91mW quando processando vídeos UHD 8K a 30 qps.

O trabalho de Xu et al. (2018) apresenta um sistema para a ME do padrão HEVC, no qual o algoritmo realiza uma busca completa subamostrada dividida em três estágios sequenciais. O primeiro estágio avalia um a cada quatro blocos candidatos na SA completa, o segundo estágio avalia um a cada dois blocos candidatos em uma SA de 5×5 amostras, enquanto o último estágio avalia todos os candidatos em uma SA de 3×3 amostras. O algoritmo não foi avaliado quanto a sua eficiência de compressão comparado com a implementação original do *software* HTM. O sistema desenvolvido é capaz de processar a IME e a FME sequencialmente, para apenas nove tamanhos de PB suportados pelo padrão HEVC. O trabalho propõe um algoritmo de compressão de quadros de referência para reduzir em torno de 50% a largura de banda com a memória DRAM. O trabalho também apresenta simplificações nos filtros de interpolação da FME, de forma a reduzir o custo computacional do *hardware*. O sistema proposto consegue atingir o processamento de vídeos UHD 4K a 30 qps, suportando inclusive a predição bi-direcional, quando sintetizado para ASIC utilizando a biblioteca de 28nm da TSMC, e operando em uma frequência de 350MHz. Os resultados da síntese mostram que o sistema requer uma área de 1,09M *gates*, enquanto apresenta uma dissipação de 47mW para o processamento de vídeos UHD 4K a 30 qps.

O trabalho de Liao; Shen; Tseng (2019) adota um algoritmo de busca completa para a ME do padrão HEVC, que avalia todos os blocos candidatos presentes dentro da SA. Contudo, o tamanho e a posição dessa SA são ajustados dinamicamente de

acordo com o conteúdo da cena, limitando assim o número de candidatos que serão efetivamente avaliados. O tamanho e a posição da SA são definidos de acordo com o resultado da predição, e são compartilhados para todas as PBs de cada CTU. Portanto, este algoritmo tenta cobrir a avaliação dos melhores candidatos para todas as PBs de cada CTU. As avaliações mostraram que ao variar dinamicamente o tamanho da SA, o algoritmo proposto compara, em média, 160 blocos candidatos. Contudo, nos casos em que a SA possui o SR máximo de 32 amostras, são avaliados 4225 blocos candidatos. A solução ainda desabilita a avaliação das PBs de formato assimétrico, avaliando assim apenas 12 dos 24 tamanhos de PB suportados pelo HEVC. A solução proposta apresentou em um impacto de 1,79% no BD-Rate. A adoção de um algoritmo baseado em busca completa realiza um acesso organizado aos dados da memória, o que permite uma fácil exploração do reuso de dados. O sistema proposto utilizando o algoritmo divide a CTU em pequenos sub-blocos 4×4 e realiza o cálculo do SAD desses sub-blocos para, então, compor o SAD de todos os tamanhos de PB maiores desejados. O sistema proposto consegue atingir o processamento de vídeos UHD 4K a 60 qps quando sintetizado para ASIC considerando uma biblioteca de células de 90nm, resultando em uma área de 274,5k *gates*. Para atingir essa taxa de processamento, o sistema precisa operar em uma frequência de 201MHz. Resultados quanto ao consumo de energia não foram apresentados.

O trabalho Gu et al. (2019) apresenta uma arquitetura para a IME do padrão HEVC que consegue adotar um algoritmo de busca expansiva com diferentes formatos, diferentes SR, e com um diferente número de expansões. Para isso, este trabalho avaliou a utilização de uma busca expansiva com os formatos retângulo, losango, hexágono, e um formato direcional. O trabalho também avaliou diferentes quantidades de expansões para o algoritmo, e diferentes tamanhos da SR, variando de oito a 32 amostras. O maior SR vertical suportado possui apenas metade do SR padrão do *software* de referência do padrão HEVC, isso é, o SR vertical é limitado em 32 amostras. Com base na avaliação, o trabalho apresenta quatro combinações ideais de operação. A arquitetura proposta suporta apenas o processamento de blocos quadrados, e esta arquitetura requer o armazenamento das amostras do bloco atual e do bloco candidato em uma memória interna. Para reduzir o impacto dessa memória no tamanho total do circuito, uma estratégia de *bitdropping* é empregada nos dados a serem armazenados. Esta estratégia de *bitdropping* realiza a discretização de um ou mais *bits* a serem armazenados. O ponto de operação que atinge a maior taxa de processamento emprega uma busca em formato losango, utilizando um SR de 16 amostras e realiza quatro expansões dentro dessa SR. Os resultados mostram que este ponto de operação apresenta um impacto de 0,55% no BD-Rate. Os resultados também mostram que, neste ponto de operação, a arquitetura proposta é capaz de processar vídeos UHD 4K a 139 qps quando sintetizada para ASIC utilizando uma biblioteca de 65nm, e operando a uma

frequência de 500MHz. Os resultados de síntese mostram que a arquitetura requer uma área de 225,7k *gates*. Não foram apresentados resultados quanto ao consumo de energia.

O trabalho de Afonso et al. (2019) apresenta uma solução para a predição interquadros do padrão 3D-HEVC, aplicando alterações nos algoritmos da ME e da Estimação de Disparidade (DE) (um processo similar a ME, porém, específico para vídeos 3D) para definir quais blocos candidatos devem ser avaliados por estas ferramentas. No algoritmo da ME, poucas alterações foram feitas, sendo a principal delas a remoção da etapa *Raster*. Já no algoritmo da DE, uma alteração mais profunda foi realizada, visto que foi proposto um algoritmo de busca horizontal, que utiliza para comparação apenas os candidatos localizados à direita ou à esquerda do bloco central da SA. O trabalho também propõe que a SA tenha tamanho dinâmico, definido de acordo com a informação dos mapas de profundidade, presente nos vídeos 3D (no formato *Multi-view Video plus Depth* - MVD) para representar a distância entre os objetos da cena e a câmera. Assim, objetos mais perto da câmera tem uma SA maior, enquanto objetos mais distantes da câmera tem uma SA menor, visto que estes tendem a apresentarem menores deslocamentos. Além disso, o trabalho suporta apenas o processamento de PBs quadradas, de tamanho 8×8 e 16×16. Com isso, os resultados mostraram um impacto de 24,16% no BD-Rate. O sistema proposto possui várias unidades dedicadas trabalhando em paralelo, considerando o processamento dos dois tamanhos de PB, duas ferramentas de predição interquadros (ME e DE), cada canal de informação (textura e mapas de profundidade), e também cada uma das vistas que o sistema se propõe a processar. Assim, o sistema proposto é capaz de processar 30 qps de três vistas no formato MVD com resolução HD 1080p, quando sintetizado para ASIC utilizando a biblioteca de células de 45nm da Nangate e trabalhando a uma frequência de 100MHz. Os resultados de síntese mostram que o sistema proposto requer uma área de 40,89M *gates*. Os resultados também mostram que quando processando as três vistas no formato MVD com resolução HD 1080p, o sistema apresenta um consumo de energia de 6,45W. Considerando apenas a aplicação da ME para uma vista de textura (o equivalente a um vídeo 2D convencional), o sistema proposto requer uma área de 2,27M *gates*, enquanto apresenta uma dissipação de potência de 377mW para processar vídeos de resolução HD 1080p a 30 qps.

O trabalho Gogoi; Peesapati (2020) propõe um algoritmo para a ME do padrão HEVC similar a busca expansiva do TZS, porém, com um padrão de busca levemente alterado e realizando apenas quatro expansões por iteração, avaliando assim apenas os candidatos mais próximos do resultado da etapa Predição. No trabalho, a etapa *Raster* não é utilizada, e a etapa equivalente ao Refinamento do TZS é simplificada, na qual é realizada apenas uma iteração com três expansões. Com isso, existe uma grande redução no número de blocos candidatos avaliados pelo algoritmo proposto,

quando comparado com o algoritmo TZS. Além disso, o trabalho ainda restringe os tamanhos de PBs que são suportados pelo sistema, avaliando apenas PBs de formatos quadrados e formatos simétricos, com tamanhos entre 8×8 até 32×32. Logo, apenas sete dos 24 diferentes tamanhos de PB do padrão HEVC são suportados, reduzindo ainda mais o número de blocos candidatos disponíveis para a avaliação da ME. Com todas essas alterações, os resultados mostraram um impacto médio de 2,39% no BD-Rate. O sistema proposto utilizando o algoritmo emprega o reuso de operações, calculando o SAD de diferentes tamanhos de PB em paralelo de maneira eficiente. Por não suportar o tamanho máximo da CTU do padrão HEVC, o sistema consegue atingir o processamento de vídeos UHD 8K com até 78 qps quando sintetizado para ASIC utilizando a biblioteca de 90nm da TSMC. Para atingir essa taxa de processamento, o sistema precisa trabalhar em uma frequência de 162MHz. Os resultados de síntese mostram que esse sistema requer 2,79M *gates*, enquanto a dissipação de potência é de 562mW.

4.1.1 Considerações Finais das Arquiteturas Dedicadas para a ME

A Tabela 2 apresenta as principais características e resultados dos trabalhos apresentados. Como visto, existem diversos trabalhos da literatura focando no desenvolvimento de *hardware* dedicado para as ferramentas de predição interquadros entre ME, IME e DE. Os trabalhos também empregam diferentes estratégias para redução do custo computacional da arquitetura desenvolvida e que afetam a quantidade de acessos à memória.

Tabela 2 – Características e resultados de diferentes trabalhos da literatura.

	Ferramenta Padrão	Search Range	Tamanhos de PB	BD-rate	Tecnologia ASIC	Freq. (MHz)	Area (*gates*)	Potência (mW)
Jou (2015)	ME HEVC	± 64	Quadrados	5,14%	90nm	270	778k	-
Fan (2018)	ME HEVC	± 64	Todos	-0,5%	TSMC 65nm	500	489k	128,5
Perleberg (2018)	ME HEVC	± 64	Quadrados	14,2%	Nangate 45nm	1020	18,1M	426,91
Xu (2018)	IME HEVC	$\pm 160 \times \pm 96^{\dagger}$	9 Variados	-0,97%	TSMC 28nm	350	1,1M	47
Liao (2019)	ME HEVC	Dinâmico	Quadrados Simétricos	1,79%	90nm	201	274k	-
Gu (2019)	IME HEVC	$\pm 64 \times \pm 32^{\dagger}$	Todos	0,55%	65nm	225	225k	-
Afonso (2019)	ME + DE 3D-HEVC	Dinâmico	8x8 e 16x16	24,16%	Nangate 45nm	100	2,2M	377
Gogoi (2020)	ME HEVC	± 64	Quadrados Simétricos	2,39%	TSMC 90nm	162	2,8M	562

†SR apresentado na forma $H \times V$, isto é, o SR horizontal e o SR vertical.

Uma das estratégias adotada por alguns trabalhos é de reduzir o SR da busca e, consequentemente reduzir o tamanho total da SA. Alguns trabalhos também adotam

uma restrição nos tamanhos de PB que são suportados pelas ferramentas avaliadas. Outra estratégia adotada por vários trabalhos é a proposição de algoritmos de busca simplificados (rápidos) para a ME. Estes algoritmos simplificados, se comparados com o TZS, comparam um menor número de blocos candidatos, ou possuem menor dependência de dados de outras etapas da predição interquadros ou de blocos vizinhos. Mesclando essas estratégias, os trabalhos relacionados atingiram valores distintos quanto ao impacto no BD-Rate.

Por fim, alguns trabalhos empregam o reuso de dados e de operações durante o processamento. A partir do reuso de dados, é possível solicitar parte dos dados da memória e já efetuar todo o processamento que depende dos dados solicitados, reduzindo assim a comunicação com a memória. De maneira similar, o reuso de operações permite que o resultado de operações pequenas, como o valor do SAD para pequenos blocos, possa ser utilizado para compor o resultado de operações maiores, como o valor do SAD de blocos grandes, o que resulta em uma redução no consumo de energia dinâmica da arquitetura devido ao reuso destes resultados. Contudo, o reuso empregado pelos trabalhos relacionados é apenas parcial, visto que para alguns blocos o processamento ainda é realizado de maneira independente, o que pode resultar em redundância tanto no processamento como nos acessos à memória.

Devido às diferentes estratégias de redução do custo computacional e diferentes características da implementação da arquitetura dos trabalhos propostos, os resultados relativos à unidade de processamento também são muito distintos entre esses diferentes trabalhos.

4.2 Soluções Algorítmicas para Redução do Custo Computacional da ME e do *Affine* ME

Na literatura, existem diversos trabalhos que apresentam soluções algorítmicas para reduzir o custo computacional das ferramentas de predição interquadros. Ainda são poucos os trabalhos que propõem soluções para o VVC e para o *Affine* do VVC. Contudo, existem alguns trabalhos apresentando soluções para a predição interquadros do HEVC e que, pela similaridade com a predição interquadros do VVC, podem ser aplicadas e/ou adaptadas em um codificador segundo o padrão VVC.

O trabalho Goncalves et al. (2018) avaliou que a maior parte dos resultados para melhor bloco candidato da etapa *Raster* do TZS está localizado dentro de uma região central da SA, em formato astróide. Assim, este trabalho propõe que o *Raster* seja limitado para avaliar os blocos candidatos apenas dentro desta região, em formato astróide. Os resultados mostraram que a etapa *Raster* do algoritmo resultante é capaz de cobrir 62% dos melhores candidatos encontrados pelo *Raster* original. Além disso,

também mostraram um impacto de apenas 0,04% no BD-Rate, com uma redução média de 21,6% no tempo de codificação do codificador HEVC.

O trabalho de Zhang et al. (2019) analisou o impacto de realizar a MC *Affine* sobre sub-blocos de diferentes tamanhos, utilizando como comparação a implementação original do *software* VTM na versão 3.0 que aplica a MC *Affine* para cada sub-bloco 4×4. O trabalho analisou que realizar a MC *Affine* sobre sub-blocos 2×2 acarreta em um aumento no tempo de codificação de 36%, enquanto que o impacto no BD-Rate é de -0,47%. Por outro lado, ao aplicar a MC *Affine* para blocos 8×8 o impacto no BD-Rate é de apenas 0,7%, enquanto que o tempo de codificação é reduzido em 4%. Este trabalho também avaliou que a predição das amostras centrais geralmente é mais precisa do que a predição das amostras da borda do sub-bloco 4×4, isto é, as amostras da borda do bloco predito possuem uma maior diferença com relação ao bloco atual do que as amostras centrais. Sendo assim, o trabalho propôs compor o bloco predito pelo *Affine* ME através da média de duas MC *Affine*. A primeira é a MC *Affine* convencional, feita apenas com sub-blocos 4×4. A outra aplica a MC *Affine* deslocando duas amostras para o lado, isso é, os cantos do bloco são compostos utilizando a MC *Affine* sobre blocos 2×2, enquanto que o centro do bloco é dividido em blocos 4×4 para ser aplicada a MC *Affine*. Os resultados mostraram que a solução proposta resulta em um aumento médio no tempo de codificação de apenas 6%, porém, resultam em um impacto de -0,53% no BD-Rate. Essa análise foi realizada antes da padronização do VVC e, embora seja uma ideia promissora, atualmente a solução não é compatível com o padrão VVC, pois requer modificações no decodificador para possibilitar a sua utilização.

Outro destes trabalhos relacionados é o de Goncalves (2021), que apresenta uma solução envolvendo aprendizado de máquina para reduzir o custo computacional da ferramenta *Affine* ME no codificador VVC. O aprendizado de máquina, utilizando *random forests*, define se o *Affine* ME com quatro parâmetros deve ser aplicado ou não após a ME convencional. Nos casos em que o *Affine* ME com quatro parâmetros é aplicado, o aprendizado de máquina também decide se o *Affine* ME com seis parâmetros deve ser aplicado ou não após o *Affine* ME com quatro parâmetros. A solução proposta foi capaz de reduzir o tempo médio de codificação da etapa *Affine* ME em 46,9%, e reduzir o tempo total de codificação do VVC em 8,5%, com um impacto de apenas 0,18% no BD-Rate.

4.3 Trabalhos Empregando Memória Aproximada

Existem diferentes trabalhos na literatura propondo estratégias de aproximação para as diferentes tecnologias de memória atuais. Essas estratégias permitem obter uma redução no consumo de energia estática e do consumo nos acessos de leitu-

ras/escritas de dados na memória, enquanto inserem a possibilidade de que alguns *bits* da informação armazenada sejam perdidos e, portanto, entregues incorretamente para a aplicação. Assim, os trabalhos descritos nesta seção propõem o emprego das estratégias de aproximação para diferentes tecnologias de memória, considerando aplicações tolerantes a falhas, como o armazenamento de imagens, apresentando redução no consumo de energia da memória aproximada e a taxa de erros na informação armazenada ao adotar cada estratégia.

4.3.1 *Dynamic RAM* Aproximada

Esta subseção apresenta quatro trabalhos encontrados na literatura que apresentam memórias aproximadas para a tecnologia DRAM. Todos estes trabalhos operam sobre o intervalo entre os *refreshs* dos dados da memória DRAM, aplicando diferentes intervalos de *refresh* em diferentes regiões da memória. A principal diferença entre estes trabalhos se dá pela estratégia adotada para dividir os dados em várias regiões e definir qual o intervalo de *refresh* que será utilizado em cada região.

O trabalho Advani et al. (2014) é o mais antigo encontrado que adota o uso de memória aproximada em um sistema de reconhecimento de objetos em vídeo. A análise é baseada em um sistema em *Field-Programmable Gate Array* (FPGA) que emprega um algoritmo para realizar a reconhecimento de objetos em regiões de interesse em um vídeo em tempo real. O sistema permite utilizar diferentes intervalos entre *refreshs* em diferentes regiões da memória. Assim, o sistema é capaz de identificar regiões de interesse, em um quadro do vídeo, que possuem tempo de vida menor do que o intervalo entre *refreshs* original, para desativar o *refresh* desses dados. Isto é, o sistema desativa o *refresh* de porções de um quadro que não possuem objetos para serem detectados, visto que dados fora das regiões de interesse são lidos da memória apenas uma vez. Já para as regiões de interesse que possuem os objetos a serem detectados, o sistema aplica o intervalo entre *refreshs* original, mantendo a imagem do objeto na memória. Assim, após acumular, na memória, várias regiões de interesse de diferentes quadros, o sistema realiza leituras dessas regiões de interesse para realizar o reconhecimento dos objetos. O sistema proposto possui uma tabela com a localização (coordenadas iniciais e finais) das regiões de interesse. Sendo assim, antes de aplicar o *refresh* sobre uma dada região da memória, o sistema busca nessa tabela se essa região da memória pertence a uma região de interesse, definindo assim se o *refresh* deve ou não ser aplicado sobre dada região. Os resultados do sistema mostraram que ele é capaz de atingir 94% da redução máxima da energia consumida por *refreshs*, considerando que a redução máxima é obtida com o *refresh* completamente desativado. Os resultados também mostram que o circuito necessário para adotar a solução proposta apresenta uma dissipação de potência de 2,4mW, um valor menor que 1% da dissipação de potência do sistema completo.

O trabalho de Lucas et al. (2014) também adota a ideia da existência de diferentes taxas de *refresh* para diferentes dados da memória. Porém, nesse trabalho foi empregada a ideia de que *bits* menos significativos são menos relevantes para a informação e, portanto, podem ser armazenados com menor precisão. O sistema de armazenamento proposto no trabalho possui duas ou mais regiões na memória. Ao armazenar um conjunto de dados, o sistema realiza a reorganização dos *bits* destes dados. No caso de duas regiões, os *bits* mais significativos são agrupados em uma determinada região da memória e os *bits* menos significativos são agrupados em outra. Assim, pra cada região da memória, o sistema define o intervalo de *refresh* a ser utilizado. O intervalo original é utilizado em regiões que armazenam os *bits* mais significativos, visto que erros nessas regiões podem causar falhas críticas na aplicação. Já as regiões que armazenam os *bits* menos significativos utilizam um intervalo de *refresh* maior, visto que erros nesses *bits* não causam grande falhas na aplicação. O sistema de armazenamento foi avaliado para o armazenamento de imagens não comprimidas por até 100 segundos, sendo que três simulações foram realizadas, considerando diferentes números de regiões de memória. Foi avaliado cada amostra da imagem sendo dividida em duas, quatro e oito regiões da memória. Assim, o trabalho mostrou a relação entre o intervalo de *refreshs* e a qualidade objetiva da imagem, medida em *Peak Signal-to-Noise Ratio* (PSNR), para cada uma das simulações. Para o sistema de armazenamento possuindo quatro regiões de memória, o trabalho apresentou a relação entre o intervalo entre *refreshs* e a taxa de erros em cada um dos *bits* das amostras. Este trabalho não apresenta uma avaliação do consumo de energia. Porém, comparado com um sistema de armazenamento que adota um único intervalo de *refresh* para todos os *bits* de cada amostra, a solução proposta se mostrou superior, necessitando de menos da metade da taxa de *refresh* para atingir uma mesma qualidade objetiva e, com isso, também reduzindo em mais da metade o consumo de energia dos *refreshs*.

O trabalho de Nguyen et al. (2020) utiliza uma estratégia similar a utilizada em Lucas et al. (2014), reorganizando os *bits* de cada informação a serem armazenados. No caso dos *bits* menos significativos da informação, o intervalo entre *refreshs* pode ser reduzido para reduzir o consumo de energia da memória, ou ainda, a estratégia de *bitdropping* pode ser utilizada sobre estes *bits*. O sistema de memória proposto foi avaliado utilizando aplicações de *Deep Learning*, que operam sobre ponto flutuante. Neste caso, o *bit* de sinal e os *bits* do expoente são considerados críticos e armazenados de maneira precisa, com o intervalo entre *refreshs* original. Já os *bits* da mantissa são armazenados de maneira aproximada, onde a intervalo entre *refreshs* depende da posição do *bit* na mantissa. Isto é, nos *bits* menos significativos da mantissa, existe uma maior probabilidade de ocorrerem erros do que nos *bits* mais significativos da mantissa, que também poderão conter erros. Além disso, este trabalho avaliou a utilização de *bitdropping* sobre os *bits* menos significativos da mantissa, reduzindo assim

o consumo estático da memória e também a latência para requisição dos dados, visto que a mantissa está dividida em vários endereços de memória e menos acessos seriam necessários para trazer toda a mantissa. Além disso, como cada *bit* do valor armazenado está localizado em diferentes endereços da memória, vários acessos à memória são necessários para ler a informação e, portanto, o trabalho emprega o uso de um ou mais *buffers*, que funcionam como uma pequena *cache*, armazenando os últimos valores que foram lidos da memória enquanto permitem a transposição dos valores armazenados. Os resultados mostraram que, para aplicações de *Deep Learning*, até 78% dos *bits* menos significativos da mantissa podem ser removidos sem que as redes convolucionais tenham perdas de precisão consideráveis ($<$ 1%). Também mostraram que utilizar mais de dois *buffers* resulta em uma redução no consumo de energia da memória. Com todas as técnicas propostas, a redução no consumo de energia varia de 16,7% a 41%, de acordo com a classe da aplicação *Deep Learning*, sendo esse percentual de redução no consumo de energia baseado na redução percentual de *refreshs* de cada aplicação utilizando a solução proposta. Os resultados também mostram que cada um dos *buffers* que realizam a transposição dos valores armazenados requer uma área de 243K *gates*, quando sintetizados para ASIC utilizando a biblioteca de 40nm da TSMC.

No trabalho de Yarmand et al. (2020) é proposto um esquema para determinar o nível de aproximação de cada memória em uma hierarquia de memória utilizada para o processamento de imagens por diferentes aplicações. A hierarquia de memória utilizada é composta por dois níveis de *cache* SRAM, onde a técnica de dimensionamento de tensão foi aplicada, além de uma DRAM com intervalo de *refresh* adaptativo como memória principal. Inicialmente, o esquema proposto define a taxa de erros aceitável para cada variável definida como aproximada de acordo com dados de relevância, número de operações de leitura e escrita, e também tempo de vida dessas variáveis. Então, é determinado o perfil de aproximação de todas as memórias da hierarquia, de forma a satisfazer a precisão de cada variável a ser armazenada. De resultados específicos da DRAM, o trabalho apresenta a relação entre a taxa de erros e o intervalo entre *refreshs*. O esquema completo foi avaliado para o armazenamento de imagens de diferentes *benchmarks* buscando atingir valores específicos de PSNR enquanto variava a topologia da *cache*, e os resultados mostraram que a hierarquia de memória consegue reduzir o consumo de energia de 6% até 21% de acordo com a precisão de armazenamento desejada.

4.3.2 *Static RAM* Aproximada

Esta subseção apresenta dois trabalhos propondo aproximação para a tecnologia SRAM, nas quais ambos os trabalhos empregam diferentes níveis de tensão de operação na SRAM.

O trabalho Yarmand et al. (2020), citado na subseção anterior, é um deles, propondo um esquema para determinar o nível de aproximação de cada memória em uma hierarquia de memória aproximada, composta por dois níveis de *cache* SRAM e uma DRAM como memória principal, sendo avaliado para o armazenamento de imagens de diferentes aplicações. No caso da SRAM, diferentes níveis de tensão de operação foram avaliados, reduzindo o consumo de energia estático, como também o dinâmico das leituras e escritas na memória. De resultados específicos para a SRAM, o trabalho apresenta o número de erros tanto em leituras aproximadas como em escritas aproximadas, como também o consumo estático, de acordo com diferentes níveis de tensão de operação da SRAM. Para a memória *cache* nível L2, a redução de energia atingiu até 50% em algumas topologias da *cache*, indicando que o dimensionamento da tensão da SRAM pode prover uma boa redução no consumo de energia da memória.

Já o trabalho Gong et al. (2017) explora otimizações em uma célula SRAM que permitam o emprego da técnica de redução de tensão de operação da SRAM pra redução de potência de uma memória SRAM, sem grandes impactos na taxa de falhas dos dados armazenados e também sem um grande aumento na área do circuito. Buscando reduzir as falhas dessa técnica, o trabalho explora a largura dos transistores de uma célula de memória da SRAM, avaliando quais os transistores mais propensos a inserir falhas no dado quando a tensão é reduzida, desta forma, variando a largura apenas desses transistores. O trabalho também utilizou estratégias para separar os *bits* do dado em memórias com diferentes níveis de aproximação. A solução proposta foi avaliada no passo de reconstrução do decodificador H.264/AVC, para armazenamento dos quadros de referência. Foram avaliadas estratégias de aproximar a memória SRAM para armazenamento do quadro referência que resultem na menor degradação na qualidade visual, levando em conta amostras de luminância e de crominância. Os resultados mostraram que reduzir a tensão de operação da memória para 50% da tensão original enquanto a largura de transistores específicos é aumentada, pode reduzir o consumo de energia da memória SRAM em até 70% quando utilizada para decodificar vídeos QCIF (176×144) mantendo o PSNR do vídeo acima de 30,5dB.

4.3.3 *Spin-Transfer Torque Magnetic RAM* Aproximada

Esta subseção apresenta cinco trabalhos que abordam estratégias de aproximação para a tecnologia de memória STT-RAM. Estes trabalhos exploram a magnitude e a duração das correntes de leitura e escrita dos dados na STT-RAM, o que pode acarretar em alteração nos *bits* dos dados.

O trabalho Ranjan et al. (2015) propõe um circuito de memória STT-RAM com armazenamento impreciso, onde leituras e escritas possuem diferentes níveis de aproximação. O trabalho explorou tanto a redução da corrente aplicada em algumas leituras

para buscar um menor consumo de energia, como o aumento da corrente aplicada em determinadas leituras quando um aumento na performance fosse necessário. O trabalho também explorou escritas aproximadas, onde a corrente e a duração de escrita são menores, reduzindo o consumo de energia. Para aplicar a aproximação proposta, o trabalho estende as instruções de leitura e escrita, em linguagem de máquina, para incluir também o nível de aproximação desejado, que serão utilizadas para as leituras e escritas aproximadas. O trabalho também apresenta uma solução para definir os níveis de aproximação de cada instrução de escrita e leitura de maneira automática para cada algoritmo, utilizando dados de energia consumida por cada operação na memória e também a qualidade desejada para a saída. O sistema foi testado para diferentes aplicações de classificação e reconhecimento de imagens. Os resultados mostraram uma redução de 40% no consumo de energia da memória, e 19,5% de redução do consumo de energia em toda a arquitetura, com um impacto de apenas 0,5% na qualidade da saída da arquitetura devido a aproximação de determinadas operações (o trabalho não especificou a métrica utilizada na avaliação de qualidade).

O trabalho Sampaio et al. (2015) emprega uma memória *cache* utilizando células *Multi-Layer Cells* (MLC), nas quais mais de um *bit* pode ser armazenado na mesma célula, provendo assim uma redução na área da memória se comparado com células *Single-Layer Cells* (SLC). Porém, células MLC são mais propensas a erros, tanto em leitura como em escritas, o que pode dificultar a sua utilização em aplicações não tolerantes a erros. Sendo assim, a solução deste trabalho propõe a adoção dinâmica de estratégias de correção de erros de acordo com a resiliência a erros da aplicação, o nível de satisfação do usuário (qualidade subjetiva), e o consumo de energia. A correção de erros emprega o armazenamento de códigos de correção de erros em células SLC, que podem ser utilizados quando toda a aplicação deve ser realizada de maneira precisa, ou ainda quando dados específicos não podem possuir perdas. Um controle dinâmico é utilizado para adaptar o nível de correções de erros de acordo com a qualidade de saída da aplicação. O sistema proposto foi avaliado utilizando o *software* de referência do codificador HEVC. Utilizando a estratégia proposta de corrigir os erros apenas dos dados não tolerantes a erros, os resultados mostraram uma redução média de 21% da energia dinâmica da memória se comparado com a correção de todos os dados.

O trabalho Zhao et al. (2017) também utiliza células MLC, que permitem armazenar mais de um *bit* na mesma célula, na sua solução para armazenamento de imagens não comprimidas em uma memória STT-RAM aproximada. Diferente dos demais trabalhos, Zhao et al. (2017) não realiza aproximações físicas na STT-RAM a partir de alterações nos níveis de corrente que serão aplicados em leituras e escritas. Este trabalho explora o fato de que escrever ou ler os dois *bits* de cada célula MLC requer duas leituras ou escritas. Porém, um desses *bits* (geralmente o mais significativo da

célula) pode ser escrito ou lido sem acessar o outro *bit*. Portanto, para melhorar a eficiência energética da memória, o sistema proposto permite discretizar os dados na memória, isto é, o sistema emprega uma estratégia de aproximação que permite ignorar a escrita de amostras que são muito similares com as amostras vizinhas. Para dados aproximados, é realizada apenas uma operação de escrita na memória que escreva um dos *bits* da informação nos dois endereços da célula MLC, enquanto o outro *bit* da informação é discretizado. Neste caso, em um dos endereços da célula MLC, o *bit* poderá ser lido corretamente, enquanto no outro endereço da célula MLC, o *bit* armazenado pode estar incorreto. Em caso de dados precisos, ambos os *bits* da MLC são utilizados, sem discretizar a informação. A solução proposta permite que o programador/usuário defina o nível de precisão desejada, sendo esse nível de precisão considerado pelo sistema para definir o nível de semelhança entre amostras vizinhas. O sistema foi avaliado para um simples armazenamento de imagens estáticas não comprimidas. Os resultados mostraram que o consumo de energia da memória MLC STT-RAM é reduzido em mais de 30%, mantendo o PSNR das imagens acima de 34,46dB e nível de satisfação do usuário em 98%.

O trabalho Teimoori; Ejlali (2018) explora a técnica de leitura aproximada na STT-RAM. Leituras aproximadas podem ser realizadas na STT-RAM aplicando uma corrente maior do que a indicada para a tecnologia em busca de performance, o que pode ocasionar uma escrita na célula. De acordo com Teimoori; Ejlali (2018), erros de leitura seguidos por outra instrução de leitura, em um mesmo endereço de memória, afetam mais a qualidade do que um erro de leitura seguido por uma instrução de escrita. Sendo assim, este trabalho aplica um maior nível de aproximação nas operações de leitura que precedem uma operação de escrita, enquanto aplica um menor nível de aproximação em operações de leitura que precedem outra operação de leitura, buscando não ocasionar um erro que pode ser propagado para as demais leituras. Para isso, esse trabalho propõe uma solução que agrupa as instruções de leitura de acordo com a próxima instrução realizada no mesmo endereço. Então, a solução proposta define qual o nível de aproximação que deve ser aplicado sobre cada grupo de instruções, de acordo com a performance requerida, o consumo de energia e a qualidade objetiva da aplicação. Os resultados mostraram que para 90% das operações de leitura, a solução proposta consegue encontrar qual é o tipo da próxima instrução que será executada no mesmo endereço, assim conseguindo definir um nível de aproximação ótimo, que não degrade o dado a ser lido nas próximas leituras, para 90% das instruções de leitura. A solução proposta foi avaliada para o armazenamento de imagens de diferentes *benchmarks*, e os resultados mostraram que a reorganização das instruções permite um aumento médio de 14,4 dB no PSNR da saída da aplicação, se comparado com não agrupar as operações, devido a redução no número de erros de leitura seguidos por outra instrução de leitura. Os resultados também mostraram

que devido a aproximação aplicada na memória STT-RAM, a solução proposta possui menor tempo de execução (em torno de 5%) do que executando todas as instruções de maneira precisa.

O trabalho Monazzah et al. (2020) apresenta um circuito de memória STT-RAM que permite realizar escritas aproximadas, além de um algoritmo para definir o nível de aproximação de cada operação de escrita, focado principalmente no armazenamento necessário para o processamento de imagens. A solução se baseou em uma análise de que a corrente necessária para escrever um *bit* em uma célula depende do valor já armazenado na célula. Isso é, caso a célula possua o valor lógico '0' antes da operação de escrita, é necessária uma corrente maior do que no caso em que a célula possua o valor lógico '1' antes da operação de escrita. Assim, a estratégia proposta no trabalho adapta a corrente de cada escrita baseado no valor dos *bits* a serem escritos na memória. A solução permite diferentes níveis de aproximação para cada instrução de escrita na memória, baseado nas variáveis a serem aproximadas e níveis de qualidade desejados, conforme definido pelo programador da aplicação. A solução proposta foi avaliada com *benchmarks* de processamento de imagens, e os resultados mostraram que, quando aproximando as escritas das imagens na memória, a solução proposta reduz o consumo de energia das escritas na memória em 57% se comparado com a versão completamente precisa.

4.3.4 *Phase Change RAM* Aproximada

Na busca realizada foi encontrado apenas um trabalho na literatura propondo uma solução aproximada para a tecnologia de memória PC-RAM. O trabalho Teimoori et al. (2018) apresenta uma estratégia para aplicar aproximação em uma hierarquia com dois níveis de memória, composta por uma STT-RAM como memória *cache* e uma PC-RAM como memória principal. O alvo da estratégia proposta é obter o menor tempo de resposta para a memória STT-RAM, e a maior expectativa de vida para a PC-RAM, considerando sua utilização para armazenamento de imagens e de áudio em aplicações multimídia. Para reduzir o tempo de resposta da STT-RAM, o sistema utiliza a técnica de incrementar a corrente utilizada nas leituras, enquanto reduz a duração da aplicação desta corrente de leitura, o que pode ocasionar uma escrita. Já para aumentar a expectativa de vida da PC-RAM, o sistema proposto reduz o número de pulsos que serão aplicados durante cada operação de escrita, resultando assim em operações de escrita imprecisas. O trabalho apresenta um algoritmo que define o melhor nível de aproximação para cada uma das memórias que consiga suprir a meta de erros por segundo definida pela aplicação. A estratégia proposta foi avaliada com aplicações de imagem e áudio, e os resultados mostraram uma melhora média de 22,1% no tempo de execução da aplicação, isto é, uma melhora no tempo de resposta da *cache* STT-RAM, e também uma melhora de 20% na expectativa de vida da PC-

RAM.

4.3.5 Memórias Não-Voláteis em Circuitos Dedicados para a ME

Como mencionado anteriormente, memórias não-voláteis possuem algumas vantagens com relação às memórias voláteis, como um consumo de energia reduzido ou ainda uma maior escalabilidade. Por conta destas vantagens, existem alguns trabalhos na literatura propondo a utilização de tecnologias de memória não-voláteis em circuitos dedicados para a ME, como Penny et al. (2019), Souza (2019), e Silveira (2021).

No trabalho Penny et al. (2019), uma hierarquia de memória que emprega o uso de memórias não-voláteis como memória externa para a ME do padrão HEVC é proposta. O trabalho comparou diferentes tecnologias de memória, dentre elas a tecnologia de memória volátil DRAM, e as tecnologias de memória não-volátil STT-RAM e *Resistive RAM* (ReRAM). Contudo, Penny et al. (2019) avaliou apenas o emprego de cada uma dessas tecnologias, sem considerar nenhuma técnica de armazenamento de dados aproximado. Foi avaliado o consumo de energia de diferentes organizações de memória, considerando diferentes tamanhos de memória e *caches* com diferentes associatividades, mostrando assim as melhores organizações para cada uma das tecnologias. Os resultados mostraram que, comparado com se utilizar uma memória DRAM como memória principal da ME do padrão HEVC, utilizar memórias não-voláteis resulta em uma redução do consumo de energia da memória do sistema. A redução do consumo de energia obtida foi de 33,4% e 57% ao se utilizar a tecnologia STT-RAM e ReRAM, respectivamente.

O trabalho Souza (2019) propõe a utilização de aproximação, utilizando *bitdropping* para a memória de um *hardware* dedicado para a ME de codificador de vídeo do padrão HEVC. Os resultados mostraram que ao remover apenas os dois *bits* menos significativos da informação a ser armazenada, o BD-Rate indicou degradação de apenas 0,34% em um dos vídeos, enquanto que para outros vídeos o BD-Rate indicou melhoria na eficiência da compressão de até -0,14%. Já ao remover os quatro *bits* menos significativos, a eficiência da codificação reduziu para 5,28% de BD-Rate. O trabalho avaliou que, devido à redução no número de *bits* das amostras, a ME tem uma pequena dificuldade em encontrar o melhor bloco candidato, o que resultou em um aumento de até 3% no número de acessos à memória *cache* ao remover os quatro *bits* menos significativos. Os resultados da simulação também mostraram que ao remover os quatro *bits* menos significativos a energia dinâmica das escritas na memória reduz em até 50% nas tecnologias STT-RAM, PC-RAM e ReRAM. Para a energia dinâmica das leituras, remover os quatro *bits* menos significativos resultou em maior variação do consumo de energia entre diferentes tecnologias. A energia dinâmica das leituras da tecnologia STT-RAM foi reduzida em torno de 47%, a ReRAM foi reduzida

em torno de 42%, enquanto a PC-RAM foi reduzida em apenas 10%.

No trabalho Silveira (2021) também foram avaliadas diferentes tecnologias de memória voláteis e não-voláteis em uma hierarquia de memória de um circuito dedicado para a ME de um codificador de vídeo. Dentre as tecnologias de memória não-voláteis, foram avaliadas as tecnologias STT-RAM, ReRAM e PC-RAM. Já para as tecnologias voláteis, foram avaliadas as tecnologias SRAM e DRAM. Para reduzir o impacto da memória interna no sistema, o trabalho propôs algumas estratégias, como reduzir o tamanho da SA e a exploração de *bitdropping* das amostras a serem armazenadas. Juntas, estas técnicas conseguem atingir uma redução de até 88% no tamanho da memória interna necessária para armazenar a SA. Além disso, esse trabalho também propôs uma estratégia para a compressão da SA dentro da memória interna. Essa estratégia aplica uma codificação diferencial sobre sub-blocos pequenos, e sobre os resíduos é então aplicada uma codificação de *Huffman*. Assim, essa estratégia de compressão permite o armazenamento de mais informações dentro de cada bloco da memória sem resultar em perdas de informação.

Os resultados de Silveira (2021) mostraram que 95% dos blocos 8×8 codificados com a estratégia proposta requerem no máximo oito palavras de 32 *bits* para serem armazenados, o que indica uma redução de até 50% para esses blocos. Com isso, para o acesso de cada bloco candidato, um número menor de palavras da memória é necessário, resultando em um menor número de acessos à memória. Essa redução no número de acessos também afeta positivamente o consumo dinâmico, embora exista um pequeno custo para comprimir/descomprimir o bloco. O trabalho comparou as diferentes tecnologias para a memória externa e também para a memória interna, empregando as técnicas propostas. Os resultados mostraram que a utilização de memórias não-voláteis aliada a solução proposta consegue reduzir o consumo de energia em até 77% se comparado com uma hierarquia composta por memórias voláteis e sem o emprego das estratégias propostas.

4.4 Operadores Imprecisos

Os trabalhos relacionados desta seção apresentam o uso de operadores imprecisos, explorando a resiliência a imprecisão presente no processo de codificação de vídeo, onde algumas etapas não requerem que o seu resultado seja preciso, tolerando pequenas oscilações no resultado final. Assim, estes operadores imprecisos exploram a tolerância de algumas ferramentas para realizar as operações de maneira aproximada, reduzindo o custo computacional das ferramentas de forma a obter, principalmente, uma redução no consumo de energia da arquitetura.

4.4.1 Operadores de SAD Aproximados

Como mencionado anteriormente, o SAD é um cálculo realizado diversas vezes durante o processamento da ME, sendo realizado para cada bloco candidato avaliado. Na literatura, existem alguns somadores aproximados focando em reduzir a cadeia de propagação do *carry* das operações do cálculo do SAD. Essa estratégia resulta em uma menor latência para obter o SAD de cada candidato, além de permitir uma redução na potência do cálculo do SAD. Assim, nesta seção são apresentados dois trabalhos que avaliam diversos operadores aproximados em suas arquiteturas que implementam o cálculo do SAD.

O trabalho de Paltrinieri et al. (2018) apresenta uma solução aproximada para o cálculo de SAD aplicado na ME do padrão HEVC. Esse trabalho avaliou dois modelos de somadores precisos e cinco modelos de somadores aproximados em três diferentes partes de uma arquitetura responsável pelo cálculo de SAD de um bloco 16×16. Essas três partes da arquitetura incluem a primeira subtração, o conjunto de somadores que acumula o SAD de todas as linhas (os somadores finais da arquitetura), e todos os somadores que acumulam o SAD da arquitetura. Os resultados mostram que a taxa de erros é inversamente proporcional ao consumo de energia de cada opção. Como esperado, o terceiro caso, que considera substituir todos os somadores da arquitetura, é o que apresenta a maior taxa de erros, porém, é o que apresenta o menor consumo de energia para todos os modelos de somadores aproximados. Além disso, os resultados mostram que, dentre os diferentes modelos de somador aproximado, o *Lower-part Or Adder* (LOA) apresenta a maior taxa de erros, porém, ele também apresenta o menor consumo de energia, atingindo quase 6% de redução no consumo de energia se comparado com o somador *Ripple Carry Adder* (RCA). Já o modelo *Accuracy Configurable Approximate Adder* (ACAA) apresentou o melhor equilíbrio entre a taxa de erros e o consumo de energia, quando inserido na arquitetura de SAD avaliada.

O trabalho de Porto et al. (2020) apresenta uma solução aproximada para o processamento da predição interquadros de um codificador que utiliza o padrão HEVC. Nessa solução aproximada, seis modelos de somador aproximado foram considerados e avaliados para serem utilizados no cálculo do SAD de uma arquitetura de ME completa (IME + FME). Dentre as avaliações realizadas, os seis modelos de somadores foram comparados quanto aos seus resultados individuais (área, latência e consumo de energia de cada somador) e também seu impacto na eficiência de codificação quando utilizado para o cálculo do SAD na ME completa. Assim, as análises mostraram que o somador aproximado LOA é o que apresenta os melhores resultados. Na sequência, o somador LOA foi integrado na arquitetura dedicada para a ME proposta em Perleberg et al. (2018) (descrita na Seção 4.1). Especificamente, os somadores LOA foram inseridos nas subtrações realizadas na unidade da árvore de SAD. As árvo-

res de SAD são as principais unidades operativas da arquitetura de IME de Perleberg et al. (2018), sendo essa unidade responsável pelo cálculo do SAD de uma linha do bloco atual para uma linha do bloco candidato. Nas demais somas/acumulações da árvore de SAD, somadores RCA foram utilizados. Assim, comparado com trabalhos que apresentam arquiteturas dedicadas para a IME, a ME utilizando o LOA apresentou a melhor eficiência energética. Ainda, embora não apresentado no artigo original (PORTO et al., 2020), os resultados da ME com o LOA apresentaram uma redução de aproximadamente 10% no consumo de energia comparado com a mesma arquitetura de ME utilizando operadores precisos.

4.4.2 Filtros de Interpolação Aproximados

Os filtros de interpolação são empregados na FME, e também no *Affine* ME. Os filtros possuem um custo computacional razoável visto que cada filtro realiza multiplicações por diferentes coeficientes, enquanto requerem uma grande comunicação com a memória para obter as amostras de referência. Assim, esta seção apresenta alguns trabalhos da literatura que propõe soluções aproximadas para os filtros de interpolação, com foco em reduzir o custo computacional dos filtros e também reduzir a quantidade de dados requisitada da memória pelos filtros.

O trabalho Silva; Siqueira; Grellert (2019) apresenta uma arquitetura que realiza a interpolação das amostras utilizadas pela FME do padrão HEVC. Esse trabalho avalia o uso de filtros de interpolação com três níveis de aproximação diferentes, onde a cada nível de aproximação, o número de *taps* dos filtros é reduzido. Assim, foram avaliados filtros com seis, quatro e dois *taps* com relação aos filtros originais de oito *taps* do padrão HEVC. Para tanto, o "peso" dos *taps* removidos foram adicionados nos *taps* que foram mantidos, buscando assim manter o "peso" original de 64. Além da redução no consumo de energia da redução do número de *taps*, ocorre também uma redução nos acessos à memória, visto que menos amostras em posições inteiras são necessárias para gerar as amostras em posições fracionárias. A análise realizada no trabalho mostra que para blocos de tamanho 4×4, reduzir o número de *taps* reduz os acessos à memória de 31% a 75%, de acordo com o número de *taps* utilizado. A arquitetura desenvolvida foi sintetizada para FPGA, e os resultados mostraram uma redução de 15,2%, 30,9%, e 46,7% no número de *LookUp Tables* (LUTs) ao se utilizar filtros com seis, quatro e dois *taps*, respectivamente. Os resultados mostraram um impacto médio no BD-Rate de 0,25% e 0,89% ao se utilizar quatro e dois *taps*, respectivamente. Ao utilizar seis *taps*, o impacto médio no BD-Rate foi de apenas 0,02%.

O trabalho de Penny et al. (2020) também apresenta uma arquitetura aproximada para os filtros de interpolação do padrão HEVC. Este trabalho avalia a aproximação em dois níveis. A aproximação é aplicada em nível algorítmico, no qual os coeficientes dos filtros são alterados para coeficientes que permitam uma implementação em

HW mais eficiente e com um menor consumo de energia se comparado com os coeficientes originais. E a aproximação é aplicada também em nível de dados, no qual o número de *taps* dos filtros também é reduzido de forma a reduzir a quantidade de dados trazida da memória. Além disso, devido aos coeficientes resultantes serem similares entre os filtros, a lógica dos filtros foi agrupada, obtendo assim um único filtro capaz de gerar as três amostras em posições fracionárias comumente utilizadas. Assim, a arquitetura completa utiliza um conjunto de 40 filtros para realizar em paralelo a interpolação de todas as amostras necessárias para aplicar a FME sobre um bloco 8×8. Os resultados mostram que a solução que apenas altera os coeficientes resulta em um impacto no BD-Rate de 1,02%, enquanto que a solução que altera os coeficientes e reduz o número de *taps* impacta o BD-Rate em 10,5%. Os resultados de síntese para ASIC mostraram que as duas soluções reduzem o consumo de energia de 48,4% a 57,7%, se comparado com uma implementação dos filtros com os coeficientes originais, enquanto são capazes de processar vídeos com resolução de até UHD 8K a 30 qps.

Já o trabalho de Mahdavi; Azgin; Hamzaoglu (2021) apresenta uma implementação aproximada dos filtros de interpolação do padrão VVC. De maneira similar a de Penny et al. (2020), duas soluções aproximadas foram apresentadas. Na primeira delas, apenas a redução no número de *taps* é empregada na arquitetura proposta, utilizando assim filtros com três e quatro *taps*, que resultam em um aumento no BD-Rate de 1,07%. Na segunda solução, além da redução no número de *taps*, os coeficientes são também alterados para coeficientes que permitam uma implementação em *hardware* mais eficiente se comparado com os coeficientes originais do padrão VVC e que resultam em um aumento no BD-Rate de 1,35%. Além disso, para cada uma das soluções, duas arquiteturas foram apresentadas. A primeira arquitetura implementa cada um dos 15 filtros de interpolação separadamente, enquanto que a segunda arquitetura explora o compartilhamento de operações entre diferentes filtros. As quatro arquiteturas foram implementadas em uma placa FPGA, que foi capaz de realizar o processamento de vídeos HD 1080p de 47 a 49 qps de acordo com a arquitetura. Os resultados mostram que a solução aproximada consegue ter uma taxa de processamento 12% a 22,5% superior que uma versão precisa similar, de acordo com a implementação. Também mostram que a solução aproximada tem um consumo de energia de 31,2% a 51,9% menor do que uma versão precisa similar, também de acordo com a implementação. Contudo, a estratégia de compartilhar operações entre diferentes filtros só se mostrou eficiente quanto ao consumo de energia na primeira solução, onde foi apenas reduzido o número de *taps* dos filtros. Assim, na solução que utiliza coeficientes mais eficientes de serem implementados em *hardware*, compartilhar ou não o processamento entre os 15 diferentes filtros resultou em apenas uma pequena variação no consumo de energia.

5 OPORTUNIDADES DE PESQUISA

Este capítulo descreve algumas oportunidades de pesquisa relacionadas ao projeto de *hardware* dedicado e eficiente para a predição interquadros do padrão de codificação VVC. As principais oportunidades de pesquisa estão relacionadas às novas ferramentas de predição suportadas pelo VVC, no projeto focado na memória interna e também na exploração a computação e armazenamento aproximados. Além disso, também serão discutidas as oportunidades para a redução no consumo dos acessos à memória interna.

Como foi apresentado na seção anterior, existem diversos trabalhos na literatura apresentando projetos de *hardware* para a predição interquadros, principalmente para a ME, que era a principal etapa de codificação dos padrões de codificação anteriores, como o H.264 e o HEVC. Porém, estes trabalhos focam apenas na unidade de processamento, sem maiores preocupações em reduzir o número de acessos à memória e em reduzir o consumo de energia da memória em si. Além disso, o padrão VVC introduziu a ferramenta de Affine ME dentro da predição interquadros e, até o momento da escrita deste texto, não foi encontrado na literatura nenhum trabalho propondo uma solução para o *Affine* ME compatível com o padrão VVC.

Então, a principal oportunidade de pesquisa no contexto desta tese está no desenvolvimento de um sistema (unidade de processamento e memória) para a predição interquadros do padrão VVC, incluindo a ME e o *Affine* ME, com todas as decisões de projeto focadas em obter o menor consumo de energia deste sistema. Para explorar essas oportunidades de pesquisa, existem diferentes soluções e estratégias que podem ser empregadas no desenvolvimento do sistema, as quais estão listadas nas próximas seções.

5.1 Projeto do Sistema Priorizando a Memória Interna

Vários trabalhos descritos na Seção 4.1 apresentam soluções e estratégias benéficas para reduzir o número de acessos à memória por parte da ME e, consequentemente, o consumo de energia da memória. Contudo, muitos destes trabalhos focam

as decisões do projeto da arquitetura com base apenas nos resultados da unidade de processamento, sem levar em conta o resultado destas decisões no consumo de energia da memória.

Como mostrado na Seção 1.1, o consumo de energia referente à memória é dominante em sistemas ME, além de possuir maior relevância do que os resultados da própria unidade de processamento para a tomada de decisões do projeto de *hardware* dedicado para a ME, especialmente quando o alvo é o baixo consumo de energia. Desta forma, o projeto de *hardware* dedicado para a ME deve priorizar a eficiência de sua memória interna, bem como a redução do número de acessos à memória (interna e externa). Por conta disso, a principal oportunidade é inverter a ordem do projeto de *hardware* dedicado para a ME, projetando uma memória energeticamente eficiente para entregar dados à ME e, então, projetar um *hardware* dedicado e eficiente que possa ser integrado com essa memória.

As subseções abaixo apontam características e estratégias que podem ser adotadas no projeto da memória visando obter uma redução no consumo de energia da memória utilizada em sistemas ME.

5.1.1 Exploração da Combinação de Tecnologias e Organizações de Memória

Uma das ideias do trabalho é realizar todas as definições do projeto arquitetural buscando obter o menor consumo de energia da unidade de memória e, para tanto, realizar simulações com diferentes tecnologias de memória para comprovar os ganhos obtidos, considerando memórias voláteis e também memórias não-voláteis.

Como apresentado na Seção 3.4, diferentes tecnologias de memória apresentam diferentes resultados quanto ao consumo de energia em leituras, escritas, e consumo estático. Com base nos dados estatísticos de acessos à memória da ME (percentual de operações de escrita/leitura), cabe a avaliação de diferentes tecnologias de memória, visto que algumas tecnologias apresentam um menor consumo de escrita, enquanto outras tecnologias apresentam um menor consumo de leituras. A ME tende a realizar um número maior de operações de leitura se comparado com a quantidade de operações de escrita de dados na memória. Logo, uma possível estratégia é avaliar o uso de tecnologias de memória que apresentam um baixo consumo de energia nas leituras, visando assim obter os melhores resultados quanto ao consumo de energia da memória.

Outra estratégia do uso de diferentes tecnologias é dividir os dados armazenados em diferentes regiões, e armazenar cada região utilizando uma tecnologia de memória diferente. Por exemplo, dividir os dados a serem armazenados com base na probabilidade dos dados serem lidos. Neste caso, os dados que possuem maior probabilidade de serem lidos são armazenados utilizando uma tecnologia que possua um baixo consumo de energia nas operações de leituras, provendo assim um menor consumo de

energia quando estes dados são acessados. Já os dados com menor probabilidade de serem acessados são armazenados utilizando uma tecnologia de memória que possua um menor consumo nas operações de escritas ou menor consumo estático.

Além das memórias possuírem diferentes tecnologias, elas podem ser implementadas com diferentes organizações internas, de acordo com a necessidade do sistema. Parâmetros como o tamanho de cada palavra, o número de bancos, e a quantidade de portas de leitura, impactam diretamente na quantidade de dados entregues a cada acesso à memória, mas também afetam os seus resultados quanto ao consumo de energia, latência e área. Logo, diferentes organizações internas da memória podem ser avaliadas, buscando obter a organização de memória que resulte no menor consumo de energia.

5.1.2 Exploração de Memória Aproximada em Nível Físico

Como mostrado na Seção 4.3, diversos trabalhos exploram o armazenamento aproximado, buscando, principalmente, a redução do consumo de energia das operações da memória. Embora muitos dos trabalhos encontrados tenham sido avaliados com aplicações multimídia, nenhum dos trabalhos encontrados apresentou uma solução que integrasse o uso de uma memória aproximada com uma arquitetura dedicada para alguma etapa de um codificador de vídeo.

Sendo assim, explorar aproximação dos dados da memória em nível físico é uma boa oportunidade visto que os trabalhos relacionados que exploram o uso de armazenamento aproximado foram capazes de reduzir o consumo de energia da unidade de memória sob o custo de pequenas imprecisões nos dados armazenados. A adoção de operações de leitura e escrita aproximada apresentam diferentes impactos no consumo de energia e na eficiência de codificação, de acordo com o nível de aproximação e as características dos acessos à memória da unidade de processamento. Portanto, cabe avaliar o melhor conjunto de operações aproximadas, e níveis de aproximação, para ser integrada no sistema de acordo com dados estatísticos dos acessos à memória da unidade de processamento. Por exemplo, como dito anteriormente, a ME realiza poucas operações de escrita na memória da SA se comparado com os números de operações de leitura. Assim, as operações de leitura tendem a ser as de maior consumo de energia durante o processamento da ME sendo, provavelmente, as operações que devem possuir o maior nível de aproximação para se obter os maiores ganhos de redução do consumo de energia.

5.1.3 Organização dos Acessos às Memórias Aproximadas

Os trabalhos da Seção 4.3 apresentam estratégias para o uso de memória aproximada sob o nível de projeto, no qual apenas parte da informação sofre as aproximações da memória, enquanto a outra parte da informação é armazenada de maneira

precisa. Para isso, os trabalhos encontrados e descritos na Seção 4.3 possuem mais de uma memória física ou apresentam soluções para dividir a memória em duas regiões, possuindo assim endereços de memória onde as operações são precisas, e endereços onde as operações são realizadas de maneira aproximada.

Como descrito anteriormente, embora os trabalhos relacionados tenham sido avaliados com aplicações multimídia, nenhum dos trabalhos encontrados avaliou a estratégia de possuir diferentes regiões na memória quando integrada com uma arquitetura dedicada para uma etapa de um codificador de vídeo. Sendo assim, existe a oportunidade de explorar essa estratégia de possuir diferentes regiões de memória com diferentes níveis de aproximação, quando integrada com a predição interquadros. Além disso, é possível buscar diferentes formas de explorar essa estratégia, como definindo quais informações serão armazenadas em cada região da memória com base em sua importância e estatísticas dos acessos da aplicação. Alguns exemplos de como dividir a informação em diferentes regiões incluem realizar uma sub-amostragem dos dados, onde apenas um a cada N *bits* estará na região precisa da memória, armazenar os *bits* mais significativos de todas as amostras na região precisa da memória e os *bits* menos significativos na região imprecisa. Outra possibilidade é dividir a SA de acordo com a quantidade de acessos de cada amostra, armazenando assim as amostras mais frequentemente acessadas da SA na região precisa da memória, enquanto as amostras menos acessadas são armazenadas na região aproximada da memória (SILVEIRA, 2021).

Outra oportunidade de pesquisa é o desenvolvimento de um controle dinâmico de precisão do armazenamento, de acordo com as informações dos outros blocos do quadro que já foram processados. Um exemplo seria a adoção de algoritmos para reconhecimento de objetos e regiões de interesse da cena, utilizando memória precisa para armazenar as amostras dentro destas regiões, enquanto que amostras que não pertencem a nenhuma região são armazenadas utilizando armazenamento impreciso. Com isso, é possível reduzir o consumo de energia apenas da informação menos relevante, na qual pode ser inserida uma maior aproximação, enquanto que em dados mais relevantes para o processamento da predição interquadros as operações são realizadas de maneira precisa.

5.2 Heurísticas para um *hardware* dedicado

Como visto na Seção 4.1, existem alguns trabalhos na literatura que apresentam arquiteturas de *hardware* dedicado para a ME. Nesses trabalhos, diferentes heurísticas foram aplicadas em suas arquiteturas que podem vir a serem exploradas no presente trabalho visando reduzir o custo computacional da unidade de processamento e também reduzir o número de acessos à memória, obtendo assim um menor consumo

de energia de todo o sistema.

Ao empregar o reuso de operações dentro da ME, o resultado do valor do SAD de blocos pequenos é reutilizado para compor o resultado do SAD de blocos grandes, permitindo assim uma implementação mais eficiente do processamento da ME. Já empregando o reuso de dados, é possível que alguns dados sejam requisitados da memória apenas uma vez, onde os diversos processamentos que necessitam desses dados são realizados em paralelo, evitando acessos redundantes à memória. Contudo, nenhum dos trabalhos encontrados na literatura consegue aplicar o reuso durante o processamento de todos os candidatos a serem processados, visto que sempre existe alguns candidatos no qual o processamento realizado é redundante, nos quais o processamento de diferentes tamanhos de bloco não consegue explorar o reuso de dados e de operações. Portanto, o reuso de dados e de operações ainda pode ser explorado de maneira mais eficiente se todos os tamanhos de bloco forem processados em paralelo, sem que nenhum candidato requisite acessos redundantes à memória.

Ainda, outras estratégias adotadas pelos trabalhos apresentados na Seção 4.1 podem ser exploradas no presente trabalho visando a redução do custo computacional da arquitetura e o número de acessos a memória. Adotar algoritmos de busca simplificados é benéfico pois reduz tanto o custo computacional como o número de acessos à memória para processar uma grande quantidade de blocos candidatos. Reduzir o número de tamanhos de bloco suportados pela arquitetura também contribui para reduzir o custo computacional, visto que são menos decisões que precisam ser tomadas quanto à divisão do bloco em tamanhos menores. Além disso, a alocação dinâmica da posição e do tamanho da SA, de acordo com o conteúdo da cena, contribuem para limitar o processamento apenas em regiões e cenas com pouco movimento.

Heurísticas podem ser adotadas para definir se determinadas etapas da predição interquadros devem ser aplicadas sobre a CB sendo processada. Como adotar aprendizado de máquina para definir se o *Affine* ME deve ou não ser aplicado sobre uma CB (GONCALVES, 2021), reduzindo assim o custo computacional do *Affine* ME ao custo de uma pequena redução na eficiência de codificação. O modelo de aprendizado de máquina proposto por (GONCALVES, 2021) não foi avaliado em um *hardware* dedicado para confirmar os seus ganhos em redução do custo computacional de uma arquitetura. Portanto, cabe a avaliação desse modelo de aprendizado de máquina para reduzir o custo computacional de uma arquitetura para o *Affine* ME, ou até mesmo avaliar uma heurística similar que permita reduzir o número de CBs nas quais o *Affine* ME será avaliado.

A reconstrução de diversos conjuntos de MVs parciais do *Affine* ME pode ser aplicado sobre um tamanho de sub-bloco diferente do 4×4, como definido no padrão VVC, visto que aplicar essa reconstrução sobre sub-blocos 8×8 reduz o custo compu-

tacional do *Affine* ME sob o custo de um pequeno impacto na eficiência de codificação (ZHANG et al., 2019). Embora não seja uma solução compatível com o VVC, visto que o bloco reconstruído seria diferente entre a codificação e a decodificação, é possível explorar essa ideia apenas nas avaliações de MVs parciais, sendo a reconstrução final aplicada sobre sub-blocos 4×4.

5.3 Utilização de operadores imprecisos no projeto de *hardware* dedicado

Como apresentado na Seção 4.4, existem diversos trabalhos que exploram o uso de operadores imprecisos nas etapas do *hardware* dedicado que não necessitam que o resultado da seja preciso, seja em operações mais simples como no SAD, mas também em operações mais específicas como nos filtros de interpolação.

Assim, buscando a implementação de um *hardware* dedicado para a ME com baixo consumo de energia, operadores imprecisos podem ser empregados no cálculo do SAD para reduzir a cadeia de *carry* das operações de SAD e, consequentemente, o consumo de energia relacionado ao cálculo do SAD, sob o custo de uma pequena imprecisão nos valores dos SADs. Além disso, os diversos operadores imprecisos disponíveis na literatura podem ser explorados, buscando obter o melhor equilíbrio entre o consumo de energia, a latência do circuito e também o impacto na eficiência de codificação.

Os filtros de interpolação também podem adotar o uso de operadores aproximados visando reduzir o número de *taps* dos filtros, assim buscando obter uma redução no custo computacional e no consumo de energia da unidade de processamento. Os coeficientes utilizados nos filtros também podem ser alterados para coeficientes que permitam uma implementação de menor custo computacional, resultando assim em um menor consumo de energia para o seu processamento.

Além disso, é possível aplicar o uso de operações imprecisas de maneira dinâmica, na qual a solução permita que o processamento possa ser realizado tanto de maneira precisa como de maneira aproximada, enquanto heurísticas definem qual tipo de operação deve ser aplicada durante o processamento. Assim, para determinados candidatos, ou para determinadas amostras do bloco a ser processado, o processamento possa ser realizado de maneira precisa, enquanto que para outros candidatos ou amostras do bloco o processamento possa ser realizado de maneira aproximada.

6 PROPOSTA DE TESE

A proposta de trabalho a ser desenvolvido nesta Tese de Doutorado é o desenvolvimento de um sistema de *hardware* dedicado para a predição interquadros do padrão de codificação de vídeo VVC, com as decisões do projeto focadas em memória. Esse sistema será composto pela unidade de processamento e a memória. O foco será nas etapas ME e *Affine* ME, visto que essas etapas possuem um elevado custo computacional durante a busca dos MVs, demandando, assim, um sistema dedicado para lidar com o seu alto volume de processamento enquanto codifica vídeos de alta resolução em tempo real. Além disso, essas são as etapas responsáveis pelo maior número de acessos à memória no padrão VVC (CERVEIRA et al., 2020), portanto, etapas que demandam um alto consumo de energia, tanto na unidade de processamento, quanto nos acessos à memória. Sendo assim, o objetivo principal é apresentar estratégias que propiciem um menor consumo de energia desse sistema, especialmente estratégias que propiciem uma redução no consumo de energia da memória, mas também estratégias que contribuam para o consumo de energia da unidade de processamento.

Para o projeto do sistema de memória, o objetivo será avaliar as diferentes tecnologias de memória existentes para serem integradas ao sistema de predição interquadros. Serão avaliadas as diferentes organizações internas que podem ser utilizadas na implementação da memória, de forma a obter as organizações que resultem no menor consumo de energia. Também serão exploradas soluções de memória aproximada a nível físico, como as soluções apresentadas na Seção 4.3, que permitam reduzir o consumo de energia das operações sob o custo de pequenas imprecisões na informação armazenada. Assim, a ideia é ter resultados da memória precisa e resultados da memória aproximada. Então, com base em dados estatísticos do número de acessos em cada endereço de memória, avaliar ainda a divisão dos dados em duas ou mais regiões da memória, onde cada região possui diferentes níveis de aproximação. Assim, resultando em um sistema de memória capaz de entregar os dados mais relevantes de maneira precisa, e entregar os dados menos relevantes com pequenas imprecisões.

As decisões de projeto relativas às diferentes memórias serão realizadas levando

em conta, especialmente, o consumo de energia relacionado ao sistema de memória, além de levar em conta a latência dos acessos e a área do sistema de memória. Logo, serão considerados resultados como o consumo da leitura de dados na memória interna, a escrita de dados na memória interna, a leitura de dados da memória externa e o consumo estático do sistema de memória. Desta forma, será possível obter uma solução de memória para um *hardware* dedicado para a ME que apresente o menor consumo de energia.

Para a unidade de processamento, o objetivo será implementar uma arquitetura para a predição interquadros do padrão VVC, com foco nas etapas de ME e *Affine* ME. Visando obter uma arquitetura eficiente e com um baixo consumo de energia, serão exploradas heurísticas que permitam empregar o reuso de dados e de operações, possibilitando o processamento eficiente de diferentes tamanhos de blocos suportados pelo padrão VVC. Também serão exploradas heurísticas que propiciem uma redução no custo computacional da unidade de processamento, como realizar simplificações no algoritmo de busca, e aplicando heurísticas para definir se o *Affine* ME deve ser aplicado sobre determinado bloco ou se a unidade do *Affine* ME pode ser desativada em determinados momentos. Além disso, será explorado o uso de operadores imprecisos nos filtros de interpolação e na unidade de SAD, tanto dentro da unidade de processamento da ME convencional (como já utilizado em trabalhos relacionados da literatura), como também na unidade de processamento do *Affine* ME.

Todas as heurísticas aplicadas na unidade de processamento também serão avaliadas quanto ao número de acessos à memória, sempre buscando encontrar as heurísticas que atinjam o melhor equilíbrio entre a eficiência de codificação e o menor número de acessos à memória, buscando, assim, obter o menor consumo de energia da memória considerando as diferenças no número de escritas e leituras em memória de cada heurística.

Assim, ao final do trabalho, espera-se obter um conjunto de heurísticas que permita realizar a predição interquadros do padrão VVC com um esforço computacional reduzido e com um baixo consumo de energia quando processando vídeos UHD em tempo real, com o menor impacto possível na eficiência de codificação.

6.1 Objetivos Específicos

São objetivos específicos deste trabalho:

- Estudo do padrão de codificação de vídeo VVC, como foco na etapa de predição interquadros.
- Análise estatística do número de operações de leitura e escrita necessárias pelo processamento da predição interquadros do VVC.

- Avaliação de diferentes tecnologias de memória precisas que podem ser integradas em um *hardware* dedicado para a predição interquadros do VVC.
- Avaliação do uso de memórias aproximadas na predição interquadros do VVC.
- Avaliação da divisão da informação a ser armazenada entre diferentes regiões da memória que possuem diferentes níveis de aproximação.
- Análise dos resultados obtidos para as memórias precisas e aproximadas.
- Geração de resultados do impacto na eficiência de compressão quanto à imprecisão de determinados dados na memória utilizando o *software* de referência do padrão VVC.
- Estudo da implementação do *Affine* ME no *software* de referência do padrão VVC, com foco em como os MVs são definidos.
- Exploração de heurísticas para reduzir o custo computacional da predição interquadros do VVC.
- Implementação de uma arquitetura para o *Affine* ME, explorando diferentes níveis de paralelismo e buscando diferentes estratégias para reduzir seu custo computacional.
- Implementação de uma arquitetura para a ME do padrão VVC com foco no reuso de dados e de operações.
- Síntese ASIC das arquiteturas propostas e geração de resultados quanto a eficiência de codificação com o *software* de referência do VVC, considerando as diferentes heurísticas a serem empregadas na arquitetura.
- Projeto do sistema completo de predição interquadros do VVC, com memória e unidade de processamento integrados, e com a geração dos resultados de síntese ASIC.
- Exploração de espaço de projeto quanto ao uso de operadores imprecisos nas arquiteturas da ME e do *Affine* ME.
- Escrita de artigos científicos com os resultados obtidos.
- Escrita e defesa da Tese de Doutorado.

6.2 Metodologia

O desenvolvimento do trabalho necessita de uma correta compreensão do funcionamento do codificador VVC, especialmente da etapa de predição interquadros, dado que a compreensão dos padrões de acesso à memória das ferramentas de predição interquadros é fundamental para o projeto eficiente do sistema de memória.

Para a correta compreensão do funcionamento da predição interquadros do padrão VVC, será realizado um estudo utilizando as documentações do VVC e do *software* de referência do padrão VVC. Assim, será compreendido quais os processamentos realizados por cada etapa, quais as decisões tomadas, e quais as dependências de dados de entradas de cada etapa. Para compreender o funcionamento de cada etapa da predição interquadros, será utilizada uma IDE, como o Microsoft Visual Studio (MICROSOFT, 2022), e também extraídos dados parciais do processamento realizado pelo *software* de referência do VVC. Estes dados serão extraídos utilizando sequências das condições comuns de teste (BOSSEN et al., 2020) para obter dados do funcionamento da predição interquadros em um processo mais realista. Neste ponto, já será possível extrair dados estatísticos dos acessos à memória das ferramentas de predição interquadros quando processando as condições comuns de teste (BOSSEN et al., 2020), como quais tamanhos de bloco são mais presentes durante a codificação, o número de MVs avaliados para cada CB, as amostras mais acessadas, entre outros.

Com base nos dados estatísticos das ferramentas de codificação, serão elaboradas estratégias de redução do esforço computacional para viabilizar a implementação das ferramentas de predição interquadros em *hardware* para a ME e para o *Affine* ME. As alterações necessárias no fluxo de codificação para adotar as estratégias elaboradas para redução do custo computacional serão avaliadas utilizando o *software* de referência do padrão VVC, de forma a medir o impacto dessas alterações na eficiência de codificação. Essa avaliação utilizará as condições comuns de teste (BOSSEN et al., 2020), avaliando assim as diferenças entre a implementação original do *software* de referência do VVC e a arquitetura completa desenvolvida neste trabalho para a predição interquadros do VVC.

Para gerar os resultados da memória do sistema ME, serão considerados diferentes simuladores de memória existentes na literatura, como o Cacti (LABS, 2022), o NVSim (DONG et al., 2012), o Ramulator (KIM; YANG; MUTLU, 2016) e o NVMain (POREMBA; XIE, 2012). Estes simuladores serão analisados quanto ao seu suporte para diferentes tecnologias de memória alvo do trabalho e quais os resultados que cada um dos simuladores consegue gerar (consumo de energia, latência, área etc.). Os simuladores selecionados serão, então, estudados quanto aos seus parâmetros de entrada. Então, baseado em dados estatísticos dos acessos à memória, extraídos

através de alterações no *software* de referência do padrão VVC, serão realizadas simulações para avaliar os resultados de diferentes tecnologias de memória que possam ser integradas em um *hardware* dedicado para a predição interquadros do VVC.

O passo seguinte será a avaliação da integração de memórias aproximadas, também com base nos dados estatísticos da predição interquadros. Para isso, será necessário um estudo nos trabalhos que já realizaram a avaliação de memórias aproximadas, visando compreender o processo de geração de resultados para uma memória aproximada de cada tecnologia diferente. Para obter a taxa de imprecisão dos dados armazenados em uma memória aproximada, serão realizadas simulações SPICE Monte Carlo (RAYCHAUDHURI, 2008), provavelmente utilizando o *software* LTspice (ANALOG, 2022). Já para obter os resultados de consumo de energia, latência ou outros resultados da memória aproximada, serão realizadas alterações no simulador de memória que foi utilizado para gerar os resultados da memória precisa, de acordo com o nível de imprecisão desejado para a memória.

O impacto de utilizar uma memória aproximada será avaliado utilizando o *software* de referência do padrão VVC quanto ao seu impacto na eficiência de codificação, considerando a taxa de imprecisão obtida com as simulações SPICE Monte Carlo. Também com o *software* de referência serão ainda avaliadas diferentes estratégias de dividir a informação a ser armazenada em regiões da memória que possuem diferentes níveis de aproximação, buscando, assim, as estratégias que obtenham a melhor relação entre o impacto na eficiência de codificação e o consumo de energia dos acessos à memória.

Com base nas estratégias de redução do custo computacional previamente avaliadas e o projeto arquitetural definido, a unidade de processamento para a predição interquadros será então implementada. Esta unidade de processamento será desenvolvida em *VHSIC Hardware Description Language* (VHDL), e validada utilizando o Modelsim (INTEL, 2022) com amostras reais de um vídeo e resultados parciais extraídos do *software* de referência do VVC.

A arquitetura para o *Affine* ME será desenvolvida de maneira incremental, sendo que alguns resultados preliminares estão apresentados na Seção 7.2.

Uma arquitetura para a IME já foi previamente desenvolvida durante o período do Doutorado, e os seus resultados preliminares estão apresentados na Seção 7.1. Contudo, esta arquitetura foca na IME do padrão HEVC. Assim, será necessário realizar a avaliação dessa arquitetura quando utilizada para codificar vídeos no padrão VVC e, portanto, será necessário realizar alterações no *software* de referência do padrão VVC para avaliar um processamento similar ao realizado pela arquitetura de ME já implementada. Desta forma, o desenvolvimento da arquitetura para implementar a ME do padrão VVC irá partir dessa arquitetura previamente desenvolvida, onde poucas alterações serão necessárias na descrição dessa arquitetura para obter uma imple-

mentação da IME para o VVC, devido à grande similaridade entre a IME do HEVC e do VVC.

A unidade de processamento completa será sintetizada para ASIC considerando a biblioteca de células padrão de 40nm da TSMC (TSMC, 2008), e utilizando a ferramenta RTL *Compiler*, da Cadence (CADENCE, 2020). Os resultados de potência parciais serão gerados considerando a atividade de chaveamento padrão da ferramenta RTL *Compiler*, enquanto os resultados finais serão gerados utilizando o chaveamento extraído de dados reais de uma sequência de vídeo. Com base nos resultados da síntese, unidades que apresentarem um maior consumo de energia serão otimizadas buscando reduzir o consumo de energia de toda a unidade de processamento.

Por fim, a avaliação de operadores imprecisos em determinadas operações da arquitetura será realizada apenas após o término do desenvolvimento de toda a arquitetura. Para essa avaliação de quais operadores imprecisos utilizar e em quais etapas da codificação, será realizada uma análise considerando o número de acessos à memória necessários pela arquitetura, dos resultados de performance e de potência dissipada pela arquitetura, além dos resultados de eficiência de codificação extraídos de simulações com o *software* VTM após a inserção dos operadores imprecisos.

6.3 Cronograma

O cronograma das atividades a serem realizadas durante essa Tese de Doutorado estão divididos em cinco grandes objetivos, sendo eles: 1 Revisão bibliográfica; 2 Simulação e implementação das memórias; 3 Desenvolvimento das arquiteturas; 4 Avaliação de operadores imprecisos; 5 Publicações. Para cada um destes objetivos, existem diferentes atividades associadas, as quais estão listadas abaixo. Além disso, a Tabela 3 apresenta os prazos pretendidos para o término de cada atividade.

1. Revisão Bibliográfica

    1.1 Revisão bibliográfica de trabalhos relacionados: Arquiteturas para a predição interquadros, operadores imprecisos, e utilização de memórias aproximadas.

    1.2 Estudo sobre o particionamento do padrão VVC.

    1.3 Estudo sobre a predição interquadros do padrão VVC.

2. Simulação da memória para a predição interquadros

    2.1 Geração de resultados de diferentes organizações internas de uma memória precisa.

    2.2 Geração de resultados de memória precisa para diferentes tecnologias.

2.3 Estudo sobre o processo de geração de resultados de memória imprecisa.

2.4 Geração de resultados de memória imprecisa.

2.5 Coleta de dados estatísticos dos acessos à memória e das ferramentas empregadas na codificação de vídeo utilizando o *software* de referência do padrão VVC.

2.6 Exploração do espaço de projeto quanto a dividir a informação a ser armazenada em memórias precisas e imprecisas.

3. Desenvolvimento das arquiteturas

3.1 Estudo da predição *Affine* ME como implementado no padrão VVC.

3.2 Desenvolvimento da arquitetura para o *Affine* ME.

3.3 Coleta de dados necessários para a validação da arquitetura utilizando o *software* de referência do padrão VVC.

3.4 Validação das arquiteturas.

3.5 Elaboração de estratégias de redução do custo computacional da predição interquadros.

3.6 Desenvolvimento da arquitetura para a ME do padrão VVC a partir da arquitetura de ME já desenvolvida para o padrão HEVC.

3.7 Geração dos resultados de síntese ASIC.

3.8 Otimização das memórias de acordo com os resultados da síntese ASIC.

3.9 Otimização da arquitetura de acordo com os resultados do sistema completo.

4. Avaliação de operadores imprecisos

4.1 Estudo dos operadores imprecisos utilizados nos trabalhos da literatura.

4.2 Implementação dos operadores imprecisos em diferentes operações da predição interquadros.

4.3 Análise do impacto na eficiência de codificação quanto a utilização de operadores imprecisos em diferentes operações da predição interquadros.

5. Publicações

5.1 Escrita de artigos para eventos.

5.2 Escrita de artigos para periódicos.

5.3 Escrita da tese.

5.4 Entrega da tese.

5.5 Defesa da tese.

Tabela 3 – Cronograma das atividades a serem realizadas.

	2022						2023												2024		
Atividade	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M
1.1	X	X	X	X	X	X	X	X	X												
1.2	X	X	X																		
1.3	X	X	X	X	X	X															
2.1	X	X	X	X																	
2.2			X	X	X																
2.3				X	X	X	X														
2.4					X	X	X	X	X												
2.5						X	X	X													
2.6							X	X	X	X											
3.1	X	X	X	X																	
3.2	X	X	X	X	X	X	X	X	X												
3.3					X	X	X	X	X	X	X	X									
3.4							X	X	X	X	X	X	X								
3.5								X	X	X	X										
3.6									X	X	X	X									
3.7									X	X	X	X	X	X							
3.8											X	X	X	X	X						
3.9													X	X	X	X					
4.1															X	X					
4.2															X	X	X				
4.3																X	X	X			
5.1								X	X	X	X	X	X	X	X	X					
5.2													X	X	X	X	X	X			
5.3															X	X	X	X	X	X	
5.4																				X	
5.5																					X

7 RESULTADOS PRELIMINARES

Este capítulo apresenta os resultados dos estudos focados na predição interquadros, realizados durante o período do Doutorado. Além dos estudos e análises já apresentadas ao longo deste texto, duas implementações em *hardware* já foram realizadas aplicando parte dos conhecimentos desenvolvidos ao longo desta tese. O primeiro deles foi o projeto de uma arquitetura de *hardware* dedicado para a IME do padrão HEVC. Os resultados deste trabalho foram reportados em um artigo submetido ao periódico IEEE TCSVT (PERLEBERG et al., 2022), estando atualmente em processo de revisão. Detalhes desta arquitetura de IME para o HEVC são apresentados na Seção 7.1. O segundo resultado prelimitar foi obtido com a implementação em *hardware* dos filtros de interpolação do *Affine* ME do VVC. O projeto dos 15 filtros de 6-*taps*, empregados na reconstrução *Affine*, ainda está em desenvolvimento, no entanto, os primeiros resultados são apresentados na Seção 7.2.

7.1 Arquitetura IME com reuso

No *software* HTM, o *software* de referência do HEVC (JCT-VC, 2021), cada CTU pode ser particionada em até 24 diferentes tamanhos de PB para a aplicação da IME, e, para definir os blocos candidatos a serem avaliados, o algoritmo TZS é aplicado de maneira independente para cada PB, isto é, cada PB requisita o processamento de blocos candidatos distintos.

Contudo, existe uma considerável redundância no processamento e nos acessos à memória realizados durante o processamento do TZS de maneira independente. Se mais de uma PB de uma CTB requisitar o processamento de um mesmo bloco candidato, várias operações redundantes serão realizadas, na qual as PBs grandes podem requisitar o mesmo processamento já realizado para PBs pequenas, especialmente considerando a aplicação do SAD, o qual bastaria apenas acumular o SAD de PBs pequenas para compor o valor do SAD das PBs grandes.

Da mesma forma, o processamento do TZS de maneira independente pode resultar em acessos redundantes à memória para requisitar os blocos candidatos para

avaliação. Quando PBs vizinhas estão sendo processadas e os blocos candidatos estão próximos um do outro, a IME pode acabar requisitando amostras dos blocos candidatos de maneira redundante.

Sendo assim, o foco desta arquitetura de IME para o padrão HEVC foi reduzir o custo computacional e o número de acessos à memória para obter a menor dissipação de potência do sistema IME completo (unidade de processamento e memória).

Para tanto, foi desenvolvido um algoritmo que sincroniza o TZS entre todas as PBs de uma CTB, fazendo com que todas as PBs de uma CTB avaliem os mesmos blocos candidatos. Isso permite que a arquitetura seja desenvolvida empregando o reuso de dados e de operações, o que permite o processamento de maneira eficiente de todos os 24 tamanhos de PB suportados no HEVC. Além disso, também foi avaliado o resultado da memória SRAM integrada com a arquitetura proposta, sendo que foram gerados resultados de mais de 350 organizações de memórias SRAM diferentes.

7.1.1 Algoritmo de busca baseado na CTB

A ideia do algoritmo proposto é remover a independência que existe entre diferentes PBs de uma CTB, fazendo todas as PBs de uma CTB passarem a avaliar os mesmos blocos candidatos, permitindo assim explorar todo o potencial do reuso de dados e de operações entre diferentes tamanhos de PB. Para isso, o algoritmo proposto aplica as mesmas decisões do algoritmo TZS sobre todas as PBs de uma CTB.

No caso, o algoritmo TZS é aplicado para PBs 64×64, e todas as decisões tomadas durante o processo de busca são armazenadas. Ao final da busca, estarão registradas decisões como a aplicação ou não da etapa *Raster*, quantas iterações do Refinamento foram realizadas e a localização dos Refinamentos realizados. Assim, na sequência, quando os outros 23 tamanhos de PB do HEVC forem avaliados, as mesmas decisões aplicadas sobre a PB 64×64 são aplicadas. Com isso, as PBs pequenas acabam perdendo a sua independência, visto que o TZS aplicado sobre a PB 64×64 define quais blocos candidatos devem ser avaliados.

A Figura 20 apresenta um exemplo do sincronismo entre as decisões do TZS para todos os tamanhos de PB. Neste caso, o TZS foi aplicado para a PB 64×64, e o candidato com menor valor de SAD foi encontrado no bloco candidato com deslocamento [4,4]. Logo, a etapa de Refinamento será aplicada ao redor do bloco candidato com deslocamento [4,4] (o bloco que começa no losango em amarelo). Com o algoritmo proposto, todos os tamanhos de PB devem seguir as decisões realizadas para a PB 64×64 e, sendo assim, as demais PBs também devem aplicar a etapa de Refinamento sobre o bloco candidato com deslocamento [4,4]. A Figura 20 exibe apenas as PBs 8×4 e 8×8 realizando o Refinamento sobre o bloco candidato com deslocamento [4,4], contudo, as PBs dos demais tamanhos também utilizam a decisão de realizar o Refinamento sobre este bloco candidato.

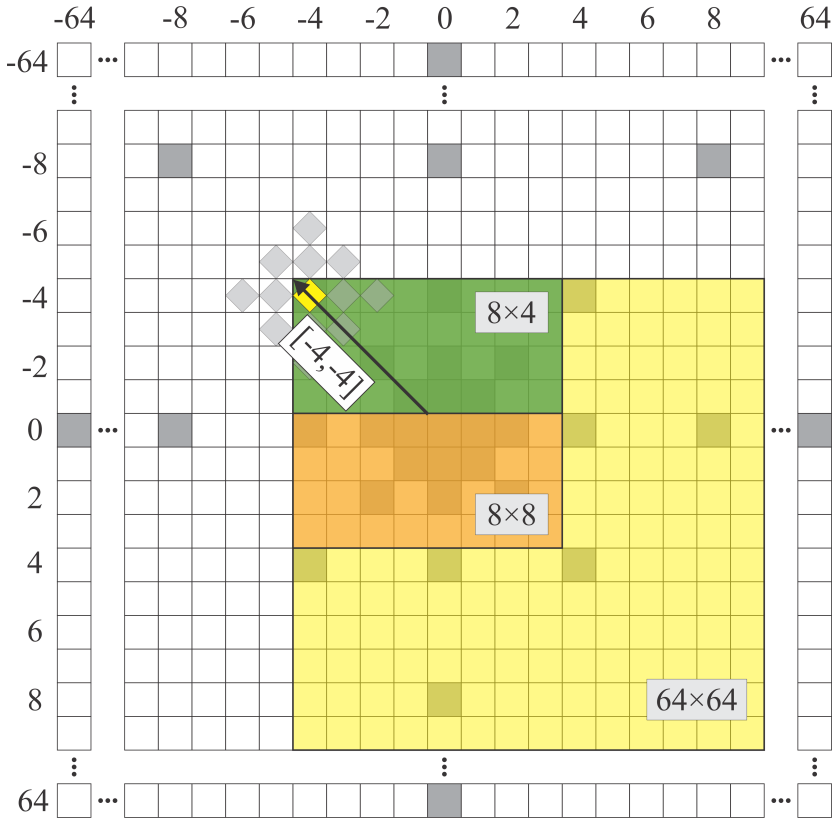


Figura 20 – Exemplo do algoritmo proposto, onde todas as PBs realizam a etapa de Refinamento sobre o bloco candidato com deslocamento [4,4].

Este algoritmo garante a possibilidade de adotar técnicas de reuso de dados na unidade de processamento, visto que, todas as PBs de uma CTB passam a avaliar blocos candidatos que possuem o mesmo deslocamento dentro da SA e, portanto, os dados referentes a cada deslocamento podem ser requisitados da memória apenas uma vez e processados por todas as PBs em paralelo. Esse algoritmo também garante a possibilidade de adotar técnicas de reuso de operações na unidade de processamento, em vista que o valor do SAD das PBs pequenas pode ser obtido inicialmente e, então, ser acumulado para obter o valor do SAD das PBs grandes. Como o algoritmo proposto permite que todos os tamanhos de PB sejam avaliados de maneira eficiente, as condições que restringiam o particionamento de algumas CTBs em PBs pequenas foi desativado e, portanto, o algoritmo considera que todos os tamanhos de PB serão sempre avaliados.

Outras restrições também foram aplicadas no algoritmo proposto visando reduzir o seu custo computacional e reduzir a dependência de decisões. Dentre elas, apenas o preditor Zero é utilizado para reduzir a dependência de dados de blocos vizinhos, o *Raster* foi limitado para avaliar apenas 25% dos seus candidatos (GONCALVES et al., 2018), as condições de parada da Busca Inicial foram removidas e o Refinamento foi limitado para apenas sete iterações consecutivas do Refinamento.

O algoritmo proposto foi avaliado utilizando o *software* de referência do HEVC (JCT-

VC, 2021). Para as simulações, foram utilizadas 22 sequências de vídeo capturadas por câmeras das condições comuns de teste (SHARMAN; SUEHRING, 2018), o que contempla seis classes de sequências, e inclui até 600 quadros em algumas sequências. Todas as simulações utilizaram a configuração *Low Delay*, e utilizam como quadro de referência apenas o quadro imediatamente anterior ao quadro atual. Os resultados mostram que sincronizar os blocos candidatos entre todos os 24 tamanhos de PB resulta em um impacto médio no BD-Rate de apenas 0,63%.

7.1.2 O Sistema de IME

A Figura 21 apresenta o sistema IME proposto. Este sistema pode ser implementado com diferentes níveis de paralelismo, chamados de *band*, que representam o número de linhas do bloco candidato recebidas em paralelo a cada ciclo de *clock*. Como a IME demanda uma grande comunicação com a memória, duas memórias SRAM foram empregadas no sistema de memória interna, empregando operações precisas. Uma delas armazena os 4kB da PB atual, enquanto a outra SRAM armazena os 36kB da SA.

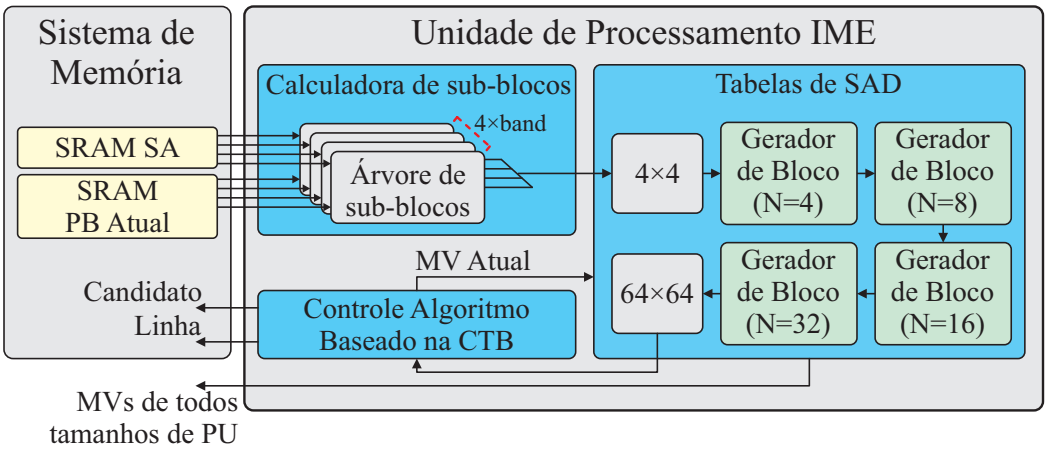


Figura 21 – Sistema IME proposto.

As memórias SRAM podem possuir diferentes organizações internas, como diferentes tamanhos de palavra, quantidade de bancos, e número de portas de leitura por banco. Diferentes organizações podem entregar a mesma quantidade de dados necessária pela unidade de processamento. Porém, estas diferentes organizações afetam o consumo de energia dos acessos à memória, a área da memória, e também a latência para entrega dos dados. Portanto, o impacto destas organizações foi medido utilizando o *software* Cacti (LABS, 2022), considerando memórias *scratchpad* e a tecnologia SRAM de 32nm.

As memórias SRAM foram avaliadas considerando dois níveis de paralelismo, quatro e oito amostras por ciclo de *clock*. Foram avaliadas memórias com a quantidade de bancos de memória variando de um a 128 bancos, com o tamanho da palavra variando de um a 1024 Bytes por palavra, enquanto mantendo o número de amostras

trazidas da memória a cada ciclo de *clock*. No total, foram avaliadas 205 organizações da memória que armazena a SA e 158 organizações da memória que armazena a PB atual. A Figura 22 apresenta os resultados de potência, em escala logarítmica, para a SRAM que armazena a SA, para os dois níveis de paralelismo avaliados. Os melhores resultados foram geralmente obtidos com um menor tamanho da palavra e uma maior quantidade de bancos. Resultados de área e latência, assim como os resultados da memória que armazena a PB atual, tiveram um padrão similar com o da Figura 22.

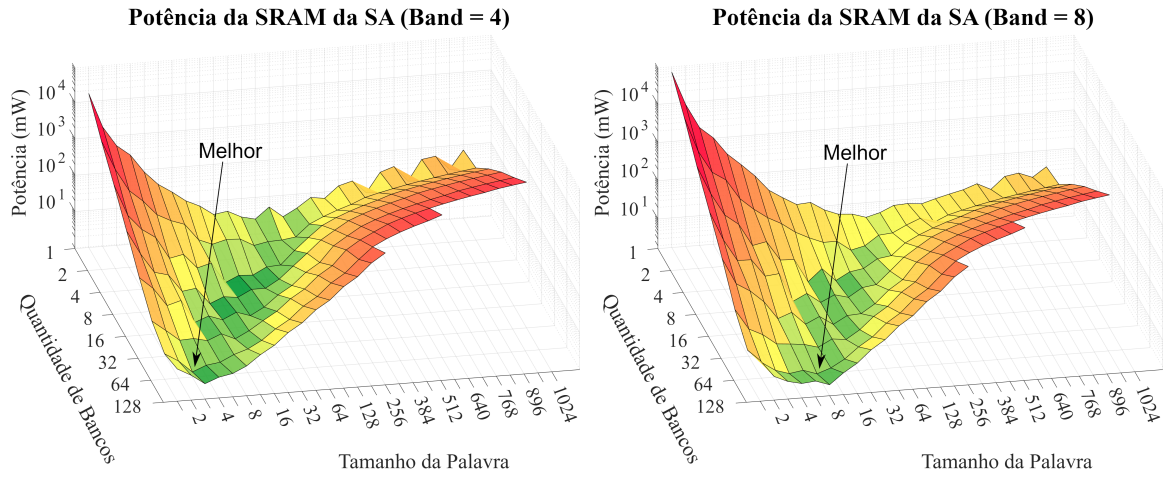


Figura 22 – Resultados das memórias SRAM avaliadas considerando paralelismo de quatro e oito amostras.

No geral, empregar um menor nível de paralelismo (*band*=4), resultou em uma menor dissipação de potência por parte da memória, visto que esta precisa entregar menos dados a cada ciclo para a unidade de processamento. A menor latência obtida para *band*=8 foi de 445ps por acesso, enquanto que a menor latência obtida para *band*=4 foi de 339ps por acesso. Essa latência indica que a SRAM utilizando o menor nível de paralelismo avaliado (*band*=4) já é capaz de entregar todas as amostras requisitadas pelo algoritmo proposto, no seu pior caso, quando processando vídeos UHD 4K a 120 qps.

Como dito, a unidade de processamento emprega o reuso de dados e de operações, de forma a processar todos os tamanhos de PB de maneira eficiente. Pela Figura 21, podemos ver que a unidade de processamento possui três unidades principais. A primeira unidade é um controle, responsável por definir os blocos candidatos a serem avaliados de acordo com o algoritmo proposto.

A segunda unidade é a Calculadora de sub-blocos, responsável pelo cálculo do SAD de sub-blocos 4×4. Para isso, essa Calculadora de sub-blocos possui várias Árvores de sub-blocos. A arquitetura de uma Árvore de Sub-blocos está representada na Figura 23, sendo que cada Árvore de Sub-blocos recebe um sub-bloco 4×4 do bloco candidato e da PB atual, e entrega como resultado o SAD entre estes sub-blocos.

Por último, a terceira unidade é uma Tabela de SAD, responsável por receber o

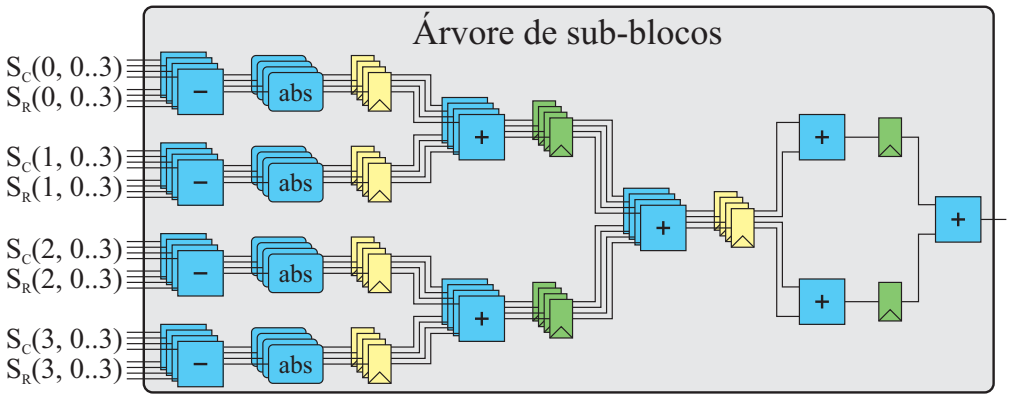


Figura 23 – Arquitetura da Árvore de Sub-blocos.

SAD dos sub-blocos 4×4, e acumular o SAD destes sub-blocos 4×4 para compor o valor do SAD de todos os demais tamanhos de PB, implementando assim o reuso de operações. Para isso, a Tabela de SAD possui unidades chamadas Gerador de Bloco, que acumulam o resultado para diferentes grupos de tamanhos de PB. O Gerador de Bloco com N=4 recebe o SAD de sub-blocos 4×4, e acumula este SAD para compor o resultado das PBs 4×8, 8×4 e 8×8. Já o Gerador de Bloco com N=8 recebe o SAD de PBs 8×8, e acumula este SAD para compor o resultado das PBs 8×16, 16×8, 16×16, além dos formatos assimétricos da CU 16×16. O mesmo se repete para os outros dois Gerador de Bloco. Por fim, o valor do SAD da PB 64×64 é utilizado pela unidade de controle para definir o próximo bloco candidato a ser avaliado de acordo com o algoritmo proposto.

O reuso de dados pode ser atingido pela unidade de processamento visto que todo o processamento que requer dados de um mesmo bloco candidato é realizado em paralelo, antes do processamento de um novo bloco candidato ser iniciado, assim reduzindo o número de acessos redundantes a um mesmo bloco candidato.

7.1.3 Resultados

A unidade de processamento desenvolvida foi descrita em VHDL e sintetizada para ASIC utilizando a ferramenta RTL Compiler, da Cadence (CADENCE, 2020), considerando a biblioteca de células de 40nm da TSMC (TSMC, 2008). Foram realizadas duas sínteses separadas, uma para cada nível de paralelismo.

A Tabela 4 apresenta os resultados de síntese obtidos para cada nível de paralelismo. Como pode ser visto, mesmo que o paralelismo seja o dobro, a área não sofre uma grande variação, dado que os recursos necessários pela Tabela de SAD são similares entre os dois níveis de paralelismo, sendo apenas o número de Árvore de Sub-blocos completamente duplicado.

Como mencionado anteriormente, foram simulados os resultados das memórias SRAM para armazenar tanto a SA como a PB atual. Os resultados de potência das melhores organizações de memória estão apresentados na Tabela 5, juntamente com

Tabela 4 – Resultados da síntese da unidade de processamento considerando diferentes paralelismos.

Band/ Paralelismo	Frequência (MHz) para 4K@120qps	Área (K *gates*)	Potência (mW) (4K@60qps)	Potência (mW) (4K@120qps)
4	1297	299,74	82,92	121,01
8	671	330,43	71,89	106,87

os resultados da unidade de processamento. Como pode ser visto nas Tabelas 4 e 5, aumentar o paralelismo da unidade de processamento tende a resultar em uma menor dissipação de potência devido a uma redução na frequência de operação da arquitetura. Por outro lado, a dissipação de potência das memórias acaba aumentando na medida em que o paralelismo aumenta, visto que as memórias precisam entregar mais dados no mesmo instante de tempo. Assim, visto que os resultados da SRAM representam mais de 89.5% da dissipação de potência do sistema IME completo, utilizar os resultados da unidade de processamento para a tomada de decisões - nesse caso, para definir o nível de paralelismo - pode resultar em uma solução sub-ótima.

Tabela 5 – Resultados de potência (em mW) do sistema IME proposto.

	Band/ Paralelismo	SRAM SA	SRAM PB Atual	Unidade de Processamento	Sistema IME Completo
4K@60qps	4	681,8	23,7	82,92	788,4
4K@60qps	8	893,4	173,7	71,89	1139,0
4K@120qps	4	1324,4	46,4	121,01	1491,8
4K@120qps	8	1680,5	326,8	106,87	2114,2

Assim, através da Tabela 5 é possível ver a importância de projetar o sistema IME focado na memória, visto que os melhores resultados de dissipação de potência estão nos sistemas com um menor nível de paralelismo. Além disso, pela Figura 22 podemos ver a importância de modelar diferentes organizações de memória para o sistema IME. Dentre as organizações de memória avaliadas, 85% delas apresentaram uma dissipação de potência pelo menos 1,2 vezes maior do que a melhor organização de memória avaliada. Portanto, adotar a organização de memória incorreta pode resultar em um grande aumento na dissipação de potência do sistema IME.

7.2 Filtros de interpolação do *Affine* ME do VVC

Como mencionado anteriormente, a reconstrução *Affine* suporta MVs com precisão de $1/16$ de amostra, além de empregar o uso de filtros de interpolação de 6-*taps*. Estes filtros utilizam seis amostras de posições inteiras de entrada, e realizam a média

ponderada destas seis amostras para gerar uma amostra em posição fracionária.

Cada filtro aplica uma média ponderada diferente, de acordo com a distância entre as amostras de entrada e a amostra fracionária gerada. Assim, os coeficientes das ponderações dependem de qual amostra fracionária deve ser gerada e, portanto, 15 filtros de interpolação diferentes são necessários. A Equação (16) apresenta a interpolação de uma amostra genérica F_N, onde A_X representa as amostras inteiras de entrada e C_X os coeficientes utilizados na filtragem (JVET, 2022).

$$F_N = (c_0 A_{-2} + c_1 A_{-1} + c_2 A_0 + c_3 A_1 + c_4 A_2 + c_5 A_3 + 32)/64 \tag{16}$$

7.2.1 Arquitetura Desenvolvida

No trabalho realizado até então, foram desenvolvidas duas implementações diferentes para cada um dos 15 filtros de interpolação de 6-*taps* utilizados na reconstrução *Affine* do VVC, totalizando 30 implementações em *hardware* dos filtros. Os 30 filtros recebem como entrada as seis amostras em posições inteiras, e entregam apenas uma amostra interpolada da sua respectiva posição fracionaria.

Como mostrado na Equação (16), o cálculo da média ponderada de cada filtro requer o cálculo de seis produtos entre as amostras de entrada e os coeficientes específicos de cada filtro. Assim, a primeira implementação emprega o uso de multiplicadores para obter estes produtos. Desta forma, cada filtro possui seis multiplicadores para obter o produto das seis amostras de entradas pelos seus respectivos coeficientes. Os resultados das entradas multiplicadas pelos coeficientes são então acumulados em um único somador. Por fim, um divisor é utilizado para realizar a divisão final da filtragem.

Já a segunda implementação não utiliza multiplicadores, sendo estes operadores substituídos por somas e deslocamentos binários. Essa implementação consiste em reformular a equação de cada filtro de forma que o cálculo de cada produto possa ser realizada por somas e deslocamentos, mas mantendo sua equivalência com os filtros do VVC. As Equações (17) a (19) apresentam um exemplo dessa reformulação para o filtro F_8, que gera amostras em posições de $1/2$ de amostra. A Equação (17) apresenta o processamento do filtro F_8 utilizando multiplicações dos seus coeficientes pelas amostras de entrada. Visto que os seis coeficientes de cada filtro são fixos e não sofrem alteração, as multiplicações por estes coeficientes podem ser substituídas por termos que permitam a implementação do filtro utilizando um conjunto de somas e deslocamentos, resultando assim na Equação (18). Os termos da Equação (18) podem ser reorganizados, de forma que as somas que requerem o mesmo deslocamento sejam agrupadas, gerando assim a Equação (19). Esta reorganização permite que resultados de algumas somas possam ser utilizados por diferentes deslocamentos, como nos termos marcados da Equação (19).

$$F_8 = (3A_{-2} - 11A_{-1} + 40A_0 + 40A_1 - 11A_2 + 3A_3 + 32)/64 \tag{17}$$

$$\begin{aligned} F_8 = [&(2+1)A_{-2} - (8+2+1)A_{-1} + (32+8)A_0 + \\ &(32+8)A_1 - (8+2+1)A_2 + (2+1)A_3 + 32]/64 \end{aligned} \tag{18}$$

$$\begin{aligned} F_8 = [&\underbrace{(A_{-2} - A_{-1} + A_3 - A_2)} + (A_0 + A_1 - A_{-1} - A_2) << 3 + \\ &\underbrace{(A_{-2} - A_{-1} + A_3 - A_2)} << 1 + (A_0 + A_1) << 5 + 32] >> 6 \end{aligned} \tag{19}$$

7.2.2 Resultados

As arquiteturas dos 30 filtros foram descritas em VHDL e sintetizados para ASIC utilizando a ferramenta RTL *Compiler*, da Cadence (CADENCE, 2020), considerando a biblioteca de células de 40nm da TSMC (TSMC, 2008). Para realizar uma comparação inicial entre as duas implementações, uma frequência de 100MHz foi utilizada na síntese. A Tabela 6 apresenta os resultados de potência e área obtidos.

Pela Tabela 6 podemos ver que, em comparação com a implementação utilizando multiplicadores, a implementação com somas e deslocamentos foi capaz de reduzir em até 21,6% a dissipação de potência total, e em até 24,6% a área do circuito. Na média, a implementação com somas e deslocamentos obteve 13,5% de redução na dissipação de potência, e 9,3% de redução na área dos filtros.

Outra característica importante para comparação é a frequência máxima de operação de cada implementação dos filtros. A Tabela 7 apresenta os resultados de síntese das duas implementações do filtro F_8 em sua frequência máxima. Os resultados mostram que a implementação utilizando somas e deslocamentos conseguiu atingir uma frequência de operação 22,9% maior do que a implementação com multiplicadores. Os resultados também mostram que, ao substituir a implementação com multiplicadores por uma implementação utilizando somas e deslocamentos, é possível obter uma redução de 2,9% e 12,8% na potência e na área do circuito, respectivamente.

Em breve, as melhores implementações destes filtros de interpolação serão integrados em uma MC 4×4 e, após, em um reconstrutor *Affine*.

Tabela 6 – Resultados de síntese considerando a frequência de operação de 100MHz.

Filtro	Multiplicadores		Somas e Deslocamentos		Δ Ganhos (%)	
	Potência Total (mW)	Área (*Gates*)	Potência Total (mW)	Área (*Gates*)	Potência Total	Área
F1	0.222	916	0.196	876	-11.71	-4.35
F2	0.238	984	0.204	893	-14.29	-9.30
F3	0.226	912	0.207	942	-8.41	3.22
F4	0.310	1206	0.255	1053	-17.74	-12.71
F5	0.287	1109	0.283	1114	-1.39	0.48
F6	0.303	1211	0.249	1028	-17.82	-15.11
F7	0.334	1300	0.277	1086	-17.07	-16.48
F8	0.320	1247	0.251	940	-21.56	-24.60
F9	0.341	1318	0.275	1086	-19.35	-17.59
F10	0.305	1216	0.249	1028	-18.36	-15.48
F11	0.286	1134	0.271	1089	-5.24	-4.01
F12	0.294	1188	0.257	1076	-12.59	-9.42
F13	0.225	911	0.209	933	-7.11	2.42
F14	0.244	1003	0.204	893	-16.39	-11.02
F15	0.213	913	0.197	876	-7.51	-4.03
Média	0.277	1134	0.239	1028	-13.54	-9.35

Tabela 7 – Resultados de síntese dos filtros em suas frequências máximas.

	Multiplicadores	Somas e Deslocamentos	Δ Ganhos (%)
Frequência (MHz)	951,47	1169,59	22,9
Potência (mW)	2,14	2,08	-2,9
Área (*gates*)	1861	1623	-12,8

8 CONCLUSÃO

Este Exame de Qualificação apresentou a motivação quanto à pesquisa nas ferramentas de predição interquadros do VVC, bem como em relação ao projeto de *hardware* dedicado e da necessidade do projeto com foco na eficiência da memória interna. Para tanto, foram apresentados os detalhes da ME convencional e também os detalhes do *Affine* ME do padrão VVC, esta última tendo sido integrada na predição interquadros do VVC para permitir a representação de movimentos de rotação, *zoom* e distorção. Foram descritos os detalhes do processamento que elevam o custo computacional destas ferramentas e que as fazem demandar uma grande quantidade de dados da memória, consequentemente, demandando um sistema de *hardware* dedicado que deve ser desenvolvido priorizando a eficiência da memória interna.

Ainda, foram apresentadas as soluções adotadas por alguns trabalhos da literatura que permitem obter uma redução no consumo de energia no projeto de *hardware* dedicado para a predição interquadros. Para tanto, foram apresentadas estratégias adotadas para reduzir o custo computacional da unidade de processamento e, consequentemente, reduzir o consumo de energia, e também técnicas de armazenamento aproximado que conseguem privilegiar o consumo de energia do sistema de memória. Também foi demonstrado que os trabalhos da literatura propondo *hardware* dedicado para a predição interquadros negligenciam o fato de que a memória do sistema de predição interquadros pode apresentar um consumo de energia mais relevante do que o consumo de energia da unidade de processamento e, com isso, o sistema resultante destes trabalhos podem ter resultados inesperados, especialmente devido ao consumo de energia da memória que não foi previamente avaliado, o que corrobora com o desenvolvimento de um sistema de *hardware* dedicado priorizando a memória.

Também foram descritas algumas oportunidades que podem ser exploradas de forma a obter um sistema de *hardware* dedicado para a predição interquadros do padrão VVC, e as vantagens dessas oportunidades para os resultados do sistema de predição interquadros. Estas oportunidades vão desde o circuito de memória até a unidade de processamento. O circuito de memória pode ser implementado com diferentes tecnologias e diferentes organizações internas, além de que a informação

pode ser armazenada em diferentes memórias com diferentes características, o que permite obter a memória com os melhores resultados de acordo com a necessidade do sistema. Este circuito de memória também pode empregar técnicas de leituras e/ou escritas aproximadas, o que permite reduzir o consumo de energia das operações realizadas em memória sob o custo de pequenas imprecisões na informação lida. Já para a unidade de processamento, este trabalho descreveu as oportunidades quanto a exploração de heurísticas que demandam um menor volume de processamento, a redução do consumo de energia através do emprego de operadores imprecisos nas unidades de ME e *Affine* ME, e também demonstrou que é possível explorar formas mais eficientes de realizar o processamento destas etapas através do reuso de dados e de operações.

Sendo assim, a proposta de tese apresentada foca no desenvolvimento de um sistema de *hardware* dedicado completo para a predição interquadros do padrão VVC, composto pela unidade de processamento e pela memória interna. O objetivo principal é obter um sistema com o menor consumo de energia e, portanto, as decisões do projeto serão focadas no projeto eficiente da memória. Para tanto, serão exploradas diferentes tecnologias e organizações internas da memória, e também o uso combinado de múltiplas memórias com diferentes características, buscando armazenar diferentes informações com base nos dados estatísticos de acessos à estas informações.

Para viabilizar a implementação das etapas ME e *Affine* ME em uma unidade de processamento com alta taxa de processamento e um baixo consumo de energia, serão exploradas diferentes estratégias de redução do custo computacional. Dentre estas estratégias, podemos citar a adoção de algoritmos de busca simplificados, a definição dinâmica da localização e do tamanho da SA, a definição dinâmica de quais tamanhos de bloco devem ser considerados, além da adoção de heurísticas para definir se determinadas etapas da predição devem ser aplicadas para determinados blocos. Além disso, o reuso de dados e de operações também será explorado, buscando realizar o processamento da predição interquadros de forma mais eficiente. Por fim, visto que o processamento da predição interquadros não requer que o processamento seja realizado de maneira precisa, também serão exploradas técnicas de armazenamento e computação imprecisos, buscando obter um menor consumo de energia do sistema.

Assim, ao final do Doutorado, é esperado conseguir evidenciar que as explorações de diferentes memórias, o conjunto de heurísticas avaliadas, e as otimizações na unidade de processamento, são capazes de reduzir o consumo de energia da memória de um sistema para a predição interquadros, garantindo a implementação das ferramentas de codificação do padrão VVC com alta eficiência de codificação e energética do sistema. É esperado que a avaliação de diferentes tecnologias, organizações e operações aproximadas na memória permitam reduzir o consumo de energia do circuito de memória. Também é esperado que a avaliação de diferentes heurísticas para as ferra-

mentas de predição interquadros reduzam a demanda por processamento, enquanto que as otimizações na unidade de processamento permitam realizar o processamento com um baixo consumo de energia enquanto mantendo a performance necessária para o processamento em tempo real de vídeos em ultra alta resolução.

REFERÊNCIAS

ADVANI, S. et al. Refresh Enabled Video Analytics (REVA): Implications on power and performance of DRAM supported embedded visual systems. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER DESIGN (ICCD), 2014. **Anais...** Institute of Electrical and Electronics Engineers (IEEE), 2014. v.32, p.501–504.

AFONSO, V. et al. Energy-Aware Motion and Disparity Estimation System for 3D-HEVC With Run-Time Adaptive Memory Hierarchy. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.29, n.6, p.1878–1892, June 2019.

ANALOG. **LTspice**. Disponível em: <https://www.analog.com/en/design-center/design-tools-and-calculators/ltspice-simulator.html>. Acesso em: 14 de jun. de 2022.

BARLA, P.; JOSHI, V. K.; BHAT, S. Spintronic devices: a promising alternative to CMOS devices. **Journal of Computational Electronics**, [S.l.], v.20, n.2, p.805–837, Jan. 2021.

BITMOVIN. **COVID-19's impact on streaming video**. Disponível em: <https://go.bitmovin.com/hubfs/Covid-19%20OTT%20streaming_Bitmovin_Analytics_Infographic.pdf>. Acesso em: 01 de fev. de 2022.

BJONTEGAARD, G. **Improvements of the BD-PSNR model**. Disponível em: <https://ci.nii.ac.jp/naid/10029505333/en/>. Acesso em: 01 de fev. de 2022.

BOSSEN, F. et al. **VTM common test conditions and software reference configurations for SDR video**. Teleconferência: JVET-T2010, 2020.

BROSS, B. et al. Overview of the Versatile Video Coding (VVC) Standard and its Applications. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.31, n.10, p.3736–3764, 2021.

BROSS, B. et al. Overview of the Versatile Video Coding (VVC) Standard and its Applications. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.31, n.10, p.3736–3764, Oct. 2021.

CADENCE. **Cadence RTL Compiler**. Disponível em: <https://www.cadence.com/content/cadence-www/global/en_US/home/training/all-courses/84441.html>. Acesso em: 01 de fev. de 2022.

CERVEIRA, A.; AGOSTINI, L.; ZATT, B.; SAMPAIO, F. Memory Profiling of H.266 Versatile Video Coding Standard. In: IEEE INTERNATIONAL CONFERENCE ON ELECTRONICS, CIRCUITS AND SYSTEMS (ICECS), 2020. **Anais...** IEEE, 2020. v.27.

CHEN, E. et al. Advances and Future Prospects of Spin-Transfer Torque Random Access Memory. **IEEE Transactions on Magnetics**, [S.l.], v.46, n.6, p.1873–1878, 2010.

CHEN, J.; YE, Y.; KIM, S. H. **Algorithm description for Versatile Video Coding and Test Model 11 (VTM 11)**. teleconference: JVET-T2002-v2, 2020.

CISCO. **Cisco Annual Internet Report (2018–2023) White Paper**. Disponível em: <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf>. Acesso em: 01 de fev. de 2022.

DONG, X.; XU, C.; XIE, Y.; JOUPPI, N. P. NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, [S.l.], v.31, n.7, p.994–1007, 2012.

EXPRESS, T. I. **COVID-19 impact**: Streaming services to dial down quality as internet speeds fall. Disponível em: <https://indianexpress.com/article/technology/tech-news-technology/coronavirus-internet-speeds-slow-netflix-hotstar-amazon-prime-youtube-reduce-streaming-quality-6331237/>. Acesso em: 21 de fev. de 2022.

FAN, Y.; HUANG, L.; HAO, B.; ZENG, X. A Hardware-Oriented IME Algorithm for HEVC and Its Hardware Implementation. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.28, n.8, p.2048–2057, Aug. 2018.

GOGOI, S.; PEESAPATI, R. A Motion Estimation Search Algorithm and its Hardware Implementation for HEVC/H.265. In: IEEE INTERNATIONAL CONFERENCE ON CONSUMER ELECTRONICS (ICCE-BERLIN), 2020. **Anais...** Institute of Electrical and Electronics Engineers (IEEE), 2020. v.10.

GONCALVES, P. H. R. **Um esquema rápido baseado em aprendizado de máquina para a predição interquadros do codificador de vídeo VVC**. 2021. Dissertação de Mestrado — Universidade Federal de Pelotas, Pelotas.

GONCALVES, P. et al. Octagonal-Axis Raster Pattern for Improved Test Zone Search Motion Estimation. In: IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING (ICASSP), 2018. **Anais...** Institute of Electrical and Electronics Engineers (IEEE), 2018. p.1763–1767.

GONG, N. et al. SPIDER: Sizing-Priority-Based Application-Driven Memory for Mobile Video Applications. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, [S.l.], v.25, n.9, p.2625–2634, Sept. 2017.

GU, C.; HUANG, L.; ZENG, X.; FAN, Y. A Micro-Code-Based Hardware Architecture of Integer Motion Estimation for HEVC. In: IFIP/IEEE INTERNATIONAL CONFERENCE ON VERY LARGE SCALE INTEGRATION (VLSI-SoC), 2019. **Anais...** Institute of Electrical and Electronics Engineers (IEEE), 2019. v.27, p.269–274.

HAN, J. et al. A Technical Overview of AV1. **Proceedings of the IEEE**, [S.l.], v.109, n.9, p.1435–1462, 2021.

HOSOMI, M. et al. A novel nonvolatile memory with spin torque transfer magnetization switching: spin-ram. In: IEEE INTERNATIONALELECTRON DEVICES MEETING, 2005. IEDM TECHNICAL DIGEST., 2005. **Anais...** [S.l.: s.n.], 2005. p.459–462.

HUANG, Y.-W. et al. Block Partitioning Structure in the VVC Standard. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.31, n.10, p.3818–3833, 2021.

INTEL. **ModelSim*-Intel® FPGA Edition Software**. Disponível em: <https://www.intel.com.br/content/www/br/pt/software/programmable/quartus-prime/model-sim.html>. Acesso em: 14 de mar. de 2022.

JCT-VC. **HEVC Test Model**. Disponível em: <https://hevc.hhi.fraunhofer.de/>. Acesso em: 01 de fev. de 2022.

JOU, S.-Y.; CHANG, S.-J.; CHANG, T.-S. Fast Motion Estimation Algorithm and Design for Real Time QFHD High Efficiency Video Coding. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.25, n.9, p.1533–1544, Sept. 2015.

JVET. **VVC Test Model**. Disponível em: <https://jvet.hhi.fraunhofer.de/>. Acesso em: 01 de fev. de 2022.

KANOPOULOS, N.; VASANTHAVADA, N.; BAKER, R. Design of an image edge detection filter using the Sobel operator. **IEEE Journal of Solid-State Circuits**, [S.l.], v.23, n.2, p.358–367, 1988.

KIM, D.-H. et al. A 16Gb 9.5Gb/S/pin LPDDR5X SDRAM With Low-Power Schemes Exploiting Dynamic Voltage-Frequency Scaling and Offset-Calibrated Readout Sense Amplifiers in a Fourth Generation 10nm DRAM Process. In: IEEE INTERNATIONAL SOLID- STATE CIRCUITS CONFERENCE (ISSCC), 2022., 2022. **Anais...** [S.l.: s.n.], 2022. v.65, p.448–450.

KIM, S. et al. Charge-Aware DRAM Refresh Reduction with Value Transformation. In: IEEE INTERNATIONAL SYMPOSIUM ON HIGH PERFORMANCE COMPUTER ARCHITECTURE (HPCA), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.663–676.

KIM, W. et al. ALD-based confined PCM with a metallic liner toward unlimited endurance. In: IEEE INTERNATIONAL ELECTRON DEVICES MEETING (IEDM), 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p.4.2.1–4.2.4.

KIM, Y.; YANG, W.; MUTLU, O. Ramulator: A Fast and Extensible DRAM Simulator. **IEEE Computer Architecture Letters**, [S.l.], v.15, n.1, p.45–49, 2016.

KüLTüRSAY, E.; KANDEMIR, M.; SIVASUBRAMANIAM, A.; MUTLU, O. Evaluating STT-RAM as an energy-efficient main memory alternative. In: IEEE INTERNATIONAL SYMPOSIUM ON PERFORMANCE ANALYSIS OF SYSTEMS AND SOFTWARE (ISPASS), 2013., 2013. **Anais...** [S.l.: s.n.], 2013. p.256–267.

LABS, H. **CACTI**. Disponível em: <https://www.hpl.hp.com/research/cacti/>. Acesso em: 01 de fev. de 2022.

LEE, B. C. et al. Phase-Change Technology and the Future of Main Memory. **IEEE Micro**, [S.l.], v.30, n.1, p.143–143, 2010.

LI, L. et al. An Efficient Four-Parameter Affine Motion Model for Video Coding. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.28, n.8, p.1934–1948, 2018.

LIAO, T.-T.; SHEN, C.-A.; TSENG, Y.-H. The algorithm and VLSI architecture of a high efficient motion estimation with adaptive search range for HEVC systems. **Journal of Real-Time Image Processing**, [S.l.], v.16, n.6, p.1943–1958, June 2019.

LIU, J.; JAIYEN, B.; VERAS, R.; MUTLU, O. RAIDR: Retention-aware intelligent DRAM refresh. In: ANNUAL INTERNATIONAL SYMPOSIUM ON COMPUTER ARCHITECTURE (ISCA), 2012., 2012. **Anais...** [S.l.: s.n.], 2012. p.1–12.

LUCAS, J.; ALVAREZ-MESA, M.; ANDERSCH, M.; JUURLINK, B. Sparkk: Quality-Scalable Approximate Storage in DRAM. In: THE MEMORY FORUM, 2014. **Anais...** [S.l.: s.n.], 2014.

MADHUSUDANA, P. C. et al. Subjective and Objective Quality Assessment of High Frame Rate Videos. **IEEE Access**, [S.l.], v.9, p.108069–108082, 2021.

MAHDAVI, H.; AZGIN, H.; HAMZAOGLU, I. Approximate Versatile Video Coding Fractional Interpolation Filters and Their Hardware Implementations. **IEEE Transactions on Consumer Electronics**, [S.l.], v.67, n.3, p.186–194, 2021.

MIANO, J. **Compressed Image File Formats**: JPEG, PNG, GIF, XBM, BMP. [S.l.]: Addison-Wesley Professional, 1999.

MICROSOFT. **Microsoft Visual Studio**. Disponível em: <https://visualstudio.microsoft.com/pt-br/>. Acesso em: 14 de mar. de 2022.

MITTAL, S. A Survey of Techniques for Architecting Processor Components Using Domain-Wall Memory. , New York, NY, USA, v.13, n.2, nov 2016.

MONAZZAH, A. M. H.; RAHMANI, A. M.; MIELE, A.; DUTT, N. CAST: Content-Aware STT-MRAM Cache Write Management for Different Levels of Approximation. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, [S.l.], v.39, n.12, p.4385–4398, Dec. 2020.

NA, T.; KANG, S. H.; JUNG, S.-O. STT-MRAM Sensing: A Review. **IEEE Transactions on Circuits and Systems II: Express Briefs**, [S.l.], v.68, n.1, p.12–18, 2021.

NGUYEN, D. T.; HUNG, N. H.; KIM, H.; LEE, H.-J. An Approximate Memory Architecture for Energy Saving in Deep Learning Applications. **IEEE Transactions on Circuits and Systems I: Regular Papers**, [S.l.], v.67, n.5, p.1588–1601, May 2020.

PALTRINIERI, A. et al. Approximate-Computing Architectures for Motion Estimation in HEVC. In: NEW GENERATION OF CAS (NGCAS), 2018., 2018. **Anais...** [S.l.: s.n.], 2018. p.190–193.

PARK, S.-H.; KANG, J.-W. Fast Affine Motion Estimation for Versatile Video Coding (VVC) Encoding. **IEEE Access**, [S.l.], v.7, p.158075–158084, 2019.

PENNY, W. et al. Energy-Efficiency Exploration of Memory Hierarchy using NVMs for HEVC Motion Estimation. In: IEEE INTERNATIONAL CONFERENCE ON ELECTRONICS, CIRCUITS AND SYSTEMS (ICECS), 2019. **Anais...** Institute of Electrical and Electronics Engineers (IEEE), 2019. v.26, p.162–165.

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.

PERLEBERG, M. et al. Memory-Centered Motion Estimation System with CTB-Based Full-Splitting Algorithm. **IEEE Transactions on Circuits and Systems for Video Technology**, Artigo submetido, 2022.

PERLEBERG, M. R. et al. Energy and Rate-Aware Design for HEVC Motion Estimation Based on Pareto Efficiency. **Journal of Integrated Circuits and Systems**, [S.l.], v.13, n.1, Aug. 2018.

POREMBA, M.; XIE, Y. NVMain: An Architectural-Level Main Memory Simulator for Emerging Non-volatile Memories. In: IEEE COMPUTER SOCIETY ANNUAL SYMPOSIUM ON VLSI, 2012., 2012. **Anais...** [S.l.: s.n.], 2012. p.392–397.

PORTO, M. S. **Desenvolvimento algorítmico e arquitetural para a estimação de movimento na compressão de vídeo de alta definição**. 2012. 166p. Tese de Doutorado — Universidade Federal do Rio Grande do Sul, Porto Alegre.

PORTO, R. et al. Fast and energy-efficient approximate motion estimation architecture for real-time 4 K UHD processing. **Journal of Real-Time Image Processing**, [S.l.], v.18, n.3, p.723–737, Sept. 2020.

RANJAN, A. et al. Approximate storage for energy efficient spintronic memories. In: ANNUAL DESIGN AUTOMATION CONFERENCE, 2015. **Anais...** ACM, 2015. v.52.

RAYCHAUDHURI, S. Introduction to Monte Carlo simulation. In: WINTER SIMULATION CONFERENCE, 2008., 2008. **Anais...** [S.l.: s.n.], 2008. p.91–100.

SAMPAIO, F. et al. Approximation-aware Multi-Level Cells STT-RAM cache architecture. In: INTERNATIONAL CONFERENCE ON COMPILERS, ARCHITECTURE AND SYNTHESIS FOR EMBEDDED SYSTEMS (CASES), 2015. **Anais...** Institute of Electrical and Electronics Engineers (IEEE), 2015. p.79–88.

SHARMAN, K.; SUEHRING, K. **Common Test Conditions**. Ljubljana: JCTVC-AF1100, 2018.

SILVA, R. da; 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.

SILVEIRA, D. S. **Caracterização energética da codificação de vídeo de alta eficiência (HEVC) em processador de propósito geral**. 2021. Tese de Doutorado — Universidade Federal do Rio Grande do Sul, Porto Alegre.

SOUZA, A. C. M. de. **Evaluation of Approximate Memories using Bit Dropping in Motion Estimation for Video Coding**. 2019. Trabalho de Conclusão de Curso — Universidade Federal do Rio Grande do Sul, Porto Alegre.

SULLIVAN, G. J.; OHM, J.-R.; HAN, W.-J.; WIEGAND, T. Overview of the High Efficiency Video Coding (HEVC) Standard. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.22, n.12, p.1649–1668, Dec. 2012.

TEIMOORI, M. T.; EJLALI, A. An Instruction-Level Quality-Aware Method for Exploiting STT-RAM Read Approximation Techniques. **IEEE Embedded Systems Letters**, [S.l.], v.10, n.2, p.41–44, June 2018.

TEIMOORI, M. T.; HANIF, M. A.; EJLALI, A.; SHAFIQUE, M. AdAM: Adaptive approximation management for the non-volatile memory hierarchies. In: DESIGN, AUTOMATION & TEST IN EUROPE CONFERENCE & EXHIBITION (DATE), 2018. **Anais...** IEEE, 2018. p.785–790.

TSMC. **40nm Technology**. Disponível em: <https://www.tsmc.com/english/dedicated/Foundry/technology/logic/l_40nm>. Acesso em: 01 de fev. de 2022.

WONG, H.-S. P. et al. Phase Change Memory. **Proceedings of the IEEE**, [S.l.], v.98, n.12, p.2201–2227, 2010.

XU, K. et al. A Low-power Pyramid Motion Estimation Engine for 4K@30fps Realtime HEVC Video Encoding. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS (ISCAS), 2018. **Anais...** Institute of Electrical and Electronics Engineers (IEEE), 2018.

YARMAND, R.; KAMAL, M.; AFZALI-KUSHA, A.; PEDRAM, M. DART: A Framework for Determining Approximation Levels in an Approximable Memory Hierarchy. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, [S.l.], v.28, n.1, p.273–286, Jan. 2020.

YU, H.; WANG, Y. **Design Exploration of Emerging Nano-scale Non-volatile Memory**. [S.l.]: Springer New York, 2014.

ZHANG, K. et al. An Improved Framework of Affine Motion Compensation in Video Coding. **IEEE Transactions on Image Processing**, [S.l.], v.28, n.3, p.1456–1469, 2019.

ZHANG, K. et al. Interweaved Prediction for Affine Motion Compensation. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (ICIP), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.3158–3161.

ZHAO, H.; XUE, L.; CHI, P.; ZHAO, J. Approximate image storage with multi-level cell STT-MRAM main memory. In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN (ICCAD), 2017. **Anais...** Institute of Electrical and Electronics Engineers (IEEE), 2017. p.268–275.