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

Tese

EXPLORAÇÃO DE COMPUTAÇÃO APROXIMADA NO PROJETO DE HARDWARE DEDICADO DE BAIXO CONSUMO PARA A CODIFICAÇÃO DE VÍDEO EM DISPOSITIVOS MÓVEIS

Roger Endrigo Carvalho Porto

Pelotas, 2020.

Roger Endrigo Carvalho Porto

EXPLORAÇÃO DE COMPUTAÇÃO APROXIMADA NO PROJETO DE HARDWARE DEDICADO DE BAIXO CONSUMO PARA A CODIFICAÇÃO DE VÍDEO EM DISPOSITIVOS MÓVEIS

Tese apresentada ao Programa de Pós-Graduação em Computação 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. Nuno Filipe Valentim Roma

Pelotas, 2020.

(PÁGINA
RESERVADA PARA
CATALOGAÇÃO)

AGRADECIMENTOS

Aos meus pais, José Sotero e Norma: por me criarem em um lar sempre cheio de amor e cuidado; pelo exemplo de perseverança e humildade; por suas visões de vida; vocês são os grandes motivadores de tudo o que faço.

À minha irmã, Aline: pelo carinho e pela amizade, mesmo estando distante; pelo tanto de vida e experiências que compartilhamos.

À Marina, minha companheira e melhor amiga: por todo amor e carinho; pelo apoio incondicional; por compartilhar meus sonhos; por não deixar que as desilusões me vençam; por ser uma mulher incrível. Tudo teria sido mais difícil sem ela.

Ao Professor Marcelo Porto, meu orientador: por ser um grande amigo desde os primórdios do GACI; pelas contribuições técnicas; por todas as estratégias elaboradas para que esse trabalho acontecesse; por ser um motivador; por tolerar minhas falhas.

Ao Professor Luciano Agostini, meu coorientador: por ser uma pessoa essencial na minha trajetória; pelas contribuições desde a minha Graduação; por ter sido sempre, e acima de tudo, um grande amigo.

Ao Professor Nuno Roma, meu coorientador: pela amizade; pela excelente acolhida no INESC-ID, em Lisboa; pelas contribuições essenciais a este trabalho.

Aos professores da UFPel: por toda a contribuição na minha formação.

Aos colegas de laboratório e parceiros de publicações: este trabalho não seria possível sem a colaboração de vocês. Obrigado!

À UFPel e ao IFSul: pelo ensino público, gratuito e de qualidade; por, mesmo sob ataques do atual governo, conseguirem ser referência em ensino, pesquisa e extensão; pelo enorme impacto positivo na vida das pessoas da região; por contribuírem não só para o progresso, mas também para termos um País mais humano, mais livre e mais igualitário; por serem, acima de tudo, espaços transformadores de pessoas (e “pessoas transformam o mundo”, citando Paulo Freire). Vida longa às instituições públicas de ensino!

O que dá o verdadeiro sentido ao encontro é a busca,
e é preciso andar muito para se alcançar o que está perto.

José Saramago

RESUMO

PORTO, Roger Endrigo Carvalho. **Exploração de Computação Aproximada no Projeto de Hardware Dedicado de Baixo Consumo para a Codificação de Vídeo em Dispositivos Móveis.** 2020. 111 f. Tese (Doutorado em Ciência da Computação) – Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2020.

A eficiência energética tornou-se uma preocupação essencial no projeto de arquiteturas em hardware dedicadas a aplicações multimídia, especialmente se o alvo de aplicação for dispositivos móveis, onde o consumo de energia é fator crítico para a vida útil das baterias. Ao mesmo tempo é um grande desafio desenvolver algoritmos e arquiteturas de hardware de baixo consumo/potência mantendo uma boa relação com a eficiência de codificação. Nesse contexto, o paradigma chamado computação aproximada é uma abordagem promissora para arquiteturas em hardware com baixo consumo. A computação aproximada toma por base a tolerância a erros de algumas aplicações para realizar simplificações. A partir dessas simplificações, é possível obter uma redução substancial no consumo energético. A codificação de vídeo é um exemplo de aplicação que pode ter seu consumo energético reduzido através da inserção de técnicas de computação aproximada, pois a aplicação de computação aproximada nos seus algoritmos geralmente resulta em perdas quase imperceptíveis nos resultados de eficiência de codificação. Levando essas questões em consideração, várias soluções de hardware com baixa dissipação de potência foram desenvolvidas e apresentadas nesta tese: uma arquitetura com foco na predição intraquadro, duas arquiteturas com foco na predição interquadros e um operador aritmético escalável. A arquitetura para cálculo de SAD na predição intraquadro é capaz de processar vídeos UHD 8K a 60 quadros por segundo com uma redução de 27% na potência dissipada e perdas de 0,28% a 1,72% na qualidade. A primeira solução aproximada para predição interquadros é uma arquitetura para estimação de movimento com baixa dissipação de potência. Os impactos na eficiência da codificação foram de 0,6% a 2,5%, contrastando com redução na dissipação de potência de 7% a 11,5%. A segunda solução é uma arquitetura completa para estimação de movimento. Como resultado, a arquitetura projetada é capaz de processar vídeos UHD 8K em tempo real a uma taxa de 120 quadros por segundo, alcançando a maior taxa de processamento e a maior eficiência energética entre todos os trabalhos relacionados. Por fim, foi desenvolvido um somador escalável em precisão e energia que pode ser configurado para suportar uma grande variedade de pontos de operação de acordo com as necessidades da aplicação alvo.

Palavras-chave: computação aproximada; eficiência energética; codificação de vídeo.

ABSTRACT

PORTO, Roger Endrigo Carvalho. **Using Approximate Computing for Energy-Efficient Video Coding Architectures Targeting Mobile Devices.** 2020. 111 f. Tese (Doutorado em Ciência da Computação) – Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2020.

Energy efficiency has become a primary concern in the design of digital systems for multimedia, especially for mobile devices. At the same time, it is a big challenge to develop energy-aware algorithms and low-power hardware architectures while keeping the trade-off with coding efficiency. In this way, a promising approach for the design of energy-efficient digital systems is a paradigm called approximate computing. This approach explores the error-resiliency or error tolerance of some applications. The error tolerance combined with the limitations of the human vision system improves the efficiency of approximate computing in video coding applications. Video coding is an example of application that can be improved in energy efficiency by inserting approximate computing techniques. The introduction of a limited amount of approximate computing in the video coding algorithms often results in almost imperceptible visual drawbacks, due to the limitations of the human visual system. With this in mind, several hardware solutions with low power dissipation were developed and presented in this Doctoral Thesis: one architecture aiming intraframe prediction, two architectures aiming interframe prediction and a power-precision scalable adder. The solution for intraframe prediction is an energy-quality scalable SAD unit. This architecture is able to process UHD 8K videos at 60 frames per second with power savings of 27% and a slight coding efficiency loss from 0.28% to 1.72%. The first approximate solution for interframe prediction is an architecture for motion estimation with low power dissipation. This architecture reached power savings from 7% to 11.5%. The impacts on the coding efficiency are negligible: between 0.6% and 2.5%.The second architecture is a complete solution for motion estimation. This solution can process UHD 8K videos in real time at 120 frames per second and has the highest throughput and the best result in terms of energy efficiency among all related works. Finally, a power-precision scalable adder has been developed. This operator is a dynamically configurable imprecise adder, supporting a wide range of applications.

Keywords: approximate computing; energy-efficiency; video coding.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACA	*Accuracy-Configurable Adder*
ASIC	*Application-Specific Integrated Circuit*
AV1	*AOMedia Video 1*
AVS2	*Audio Video Coding Standard – second generation*
BD	*Bjøntegaard-Delta*
CB	*Coding Block*
CCB	*Carry Cut-Back Adder*
CCT	Condições Comuns de Teste
CLA	*Carry Lookahead Adder*
CMOS	*Complementary Metal-Oxide-Semiconductor*
CSA	*Carry Select Adder*
CTB	*Coding Tree Block*
CTC	*Common Test Conditions*
CTU	*Coding Tree Unit*
CU	*Coding Unit*
ETA	*Error-Tolerant Adder*
FME	*Fractional Motion Estimation*
FPS	*frames per second*
FS	*Full Search Algorithm*
GeAr	*Generic Accuracy Configurable Adder*
HD	*High Definition*
HEVC	*High Efficiency Video Coding*
HM	*HEVC Test Model*
IME	*Integer Motion Estimation*
LOA	*Lower-Part-OR Adder*
LSB	*least significant bit*

ME	*motion estimation*
MSB	*most significant bit*
NP	núcleo de processamento
PB	*Prediction Block*
PPA	produto potência×atraso
PROP	*propagation*
PSNR	*Peak Signal-to-Noise Ratio*
PU	*Prediction Unit*
QP	*quantization parameter*
RCA	*Ripple Carry Adder*
SAD	*Sum of Absolute Differences*
SATD	*Sum of Absolute Transformed Differences*
SPEC	*speculation*
SSE	*Sum of Squared Errors*
TZS	*Test Zone S*earch *Algorithm*
UHD	*Ultra High Definition*
VBSME	*Variable Block-Size Motion Estimation*
VGA	*Video Graphics Array*
VHDL	*VHSIC Hardware Description Language*
VHSIC	*Very-High-Speed Integrated Circuit*
VVC	*Versatile Video Coding*
WVGA	*Wide Video Graphics Array*
WQVGA	*Wide Quarter Video Graphics Array*

SUMÁRIO

1 INTRODUÇÃO

A codificação de vídeos está presente na maioria das aplicações multimídia atuais. Junto a isso há a crescente demanda por resoluções cada vez mais altas, como Full HD, UHD 4K, e UHD 8K. Além disso, são requisitadas elevadas taxas de quadros como, por exemplo, 120, 240 e 300 quadros por segundo. Esses fatores têm aumentado drasticamente a quantidade de conteúdo de vídeo a ser processado, armazenado ou transmitido. Segundo relatório da Cisco Systems (CISCO, 2017), o tráfego de vídeo na Internet utilizou 73% do tráfego global em 2016, algo em torno de 42 exabytes por mês. Este relatório projeta que, até 2021, aplicações de vídeo representarão 82% do tráfego total da Internet, um total de 159 exabytes por mês. Em relatório mais atual (CISCO, 2020), a Cisco Systems destaca que as aplicações de vídeo possuem efeito multiplicador no tráfego da Internet e exemplifica essa afirmação através dos impactos da transição da resolução Full HD para a resolução UHD 4K. Nesse sentido, tornam-se necessários grandes esforços, acadêmicos e industriais, na tentativa de aumentar a eficiência da codificação de vídeos. No contexto da área de codificação de vídeos, a eficiência de codificação é definida pela relação entre a qualidade objetiva do vídeo codificado (também chamada de distorção) e o número de bits necessário para sua representação (também chamado de taxa) (SULLIVAN; WIEGAND, 1998). Então os codificadores de vídeo buscam otimizar essa relação, chamada taxa-distorção, em um processo chamado de Otimização Taxa-Distorção, do inglês *Rate-Distortion Optimization* (RDO) (SULLIVAN; WIEGAND, 1998). Normalmente, para diminuir o número de bits, é necessário gerar alguma degradação na qualidade do vídeo e vice e versa. Então o processo de RDO precisa estar adaptado ao tipo de aplicação alvo, priorizando mais a taxa, mais a distorção ou algum nível de balanceamento entre estes dois critérios.

A área de codificação de vídeos teve seu primeiro padrão no ano de 1988, quando foi lançado o padrão H.261 pela ITU-T. Em 1993 a ISO/IEC lançou seu padrão MPEG1.

Mas o primeiro padrão de sucesso comercial foi lançado em 1995 e foi desenvolvido em conjunto pela ITU-T e pela ISO/IEC, sendo batizado como H.262 pela ITU-T e como MPEG2 pela ISO/IEC (ITU-T; ISO/IEC JTC 1, 1994). O padrão de codificação de vídeo estado-da-arte também foi definido em conjunto pela ITU-T e a ISO/IEC, sendo lançado em 2013. Esse padrão se chama HEVC – *High Efficiency Video Coding* (HEVC, 2015). Atualmente um novo padrão da ITU-T e da ISO/IEC encontra-se em desenvolvimento, chamado de VVC – *Versatile Video Coding* (MPEG, 2019). Outras iniciativas na área também merecem destaque, como a família de codificadores chineses batizados de AVS (*Audio Video System*) (HE et al., 2014), atualmente em sua terceira geração, e o codificador da *Alliance for Open Media* (AOMedia), chamado de AV1 – AOMedia Video 1 (AOM, 2019). A AOMedia é uma aliança entre as maiores indústrias da área, como Google, Microsoft, Intel, Netflix, entre outras, que buscam padrões livres de royalties (AOM, 2019).

A codificação de vídeos é ainda mais desafiadora se considerarmos a sua crescente utilização em dispositivos móveis (CHIPPA et al., 2013). Nesse cenário, como estes dispositivos são alimentados por bateria, além da eficiência da codificação, o consumo de energia também deve ser considerado um aspecto chave. Dessa forma, o grande desafio é desenvolver algoritmos e arquiteturas de hardware para codificação de vídeos que possuam baixo consumo de energia, mantendo um compromisso com a eficiência de codificação. Em outras palavras, as aplicações devem ser eficientes não só na codificação, mas também energeticamente.

Neste sentido, uma abordagem promissora para o projeto de sistemas digitais eficientes energeticamente é o paradigma chamado computação aproximada (CHIPPA et al., 2013) (HAN; ORSHANSKY, 2013). Essa abordagem toma por base o conceito de aplicações tolerantes à imprecisão (BREUER, 2005), ou seja, aplicações onde os resultados não necessitam ser totalmente precisos, sendo aceitáveis resultados com algum nível de aproximação. Para a maioria das aplicações multimídia existe algum nível de tolerância à imprecisão, sendo possível obter uma redução substancial no consumo de energia (HAN; ORSHANSKY, 2013). Neste tipo de aplicação, a introdução de uma quantidade limitada de imprecisão nos algoritmos de processamento de vídeo muitas vezes resulta em uma quantidade insignificante de mudança visual perceptível na saída (RAHA; JAYAKUMAR; RAGHUNATHAN, 2014). Isto se torna possível porque um ser humano vai ser o observador do resultado da

codificação de vídeo e sabe-se que esse ser humano possui limitações de percepção que são naturais, típicas do sistema visual humano (GAO et al., 2010).

1.1 Hipótese a Ser Investigada

Este trabalho de investigação explora a hipótese de que é possível atingir importantes reduções na dissipação de potência com o uso de computação aproximada no contexto de codificação de vídeos.

A codificação de vídeos apresenta diversas oportunidades para exploração de computação aproximada, pois possui algoritmos que aparentam possuir diferentes graus de tolerância à imprecisão. Neste caso, a inserção de imprecisão em algumas das ferramentas de codificação pode conduzir a resultados de eficiência de codificação e de qualidade visual dos vídeos similares ou idênticos aos gerados por ferramentas precisas, mas com diminuição significativa no consumo energético.

1.2 Objetivos

Este trabalho tem como objetivo geral o desenvolvimento de arquiteturas dedicadas à codificação de vídeos com uso de computação aproximada que possuam alto desempenho e baixa dissipação de potência, mantendo a eficiência de codificação. As arquiteturas desenvolvidas devem ser compatíveis com o padrão HEVC de codificação de vídeos, estado da arte da área.

Para alcançar o objetivo geral, alguns objetivos específicos foram definidos:

- Implementação em software de operadores imprecisos e avaliação dos impactos na qualidade da codificação de vídeos;
- Implementação em hardware de operadores imprecisos e caracterização quanto ao consumo energético e ao desempenho;
- Implementação em hardware de arquiteturas imprecisas, de baixo consumo e configuráveis, para uso na codificação de vídeos;
- Implementação de novos operadores imprecisos configuráveis dedicados às etapas de predição da codificação de vídeo.

1.3 Organização do Texto

O restante deste texto está organizado como segue. No capítulo dois são apresentados os conceitos de codificação de vídeos necessários para a compreensão do trabalho

desenvolvido. No capítulo três é discutida a utilização de técnicas de computação aproximada, principalmente nas etapas de predição interquadros e intraquadro, além da apresentação dos operadores imprecisos considerados neste trabalho. Uma solução aproximada para a predição intraquadro é apresentada no capítulo 4. Da mesma forma, duas soluções aproximadas para a predição interquadro são apresentadas no capítulo 5. A seguir, o capítulo 6 apresenta um operador aritmético aproximado e configurável. Por fim, as conclusões deste trabalho são apresentadas no capítulo 7.

2 FUNDAMENTOS DE COMPRESSÃO DE VÍDEO

Este capítulo serve como uma breve introdução a alguns aspectos que serão tratados nas soluções que serão apresentadas nos próximos capítulos, com destaque para as etapas de predição do HEVC, que serão o foco das arquiteturas desenvolvidas nesse trabalho.

2.1 Modelo Genérico de um Codificador de Vídeo

A Figura 1 apresenta um modelo completo de um codificador de vídeo atual conforme detalhado em (AGOSTINI, 2007). Nela é possível observar os principais módulos e o fluxo das informações entre eles. Inicialmente, os blocos do quadro atual devem ser processados através de algum tipo de predição que pode ser intraquadro ou interquadro. O controle do codificador é o responsável por decidir qual predição deve ser utilizada para cada bloco. Na Figura 2, a predição intraquadro é o módulo que realiza a redução da redundância espacial dentro de um quadro (AGOSTINI, 2007). Dessa forma, a predição intraquadro necessita apenas das informações do quadro atual para realizar o processamento. No caso da predição interquadros, o quadro atual é comparado com um ou mais quadros de referência para que seja reduzida a redundância temporal (RICHARDSON, 2002) (AGOSTINI, 2007). Pela figura nota-se que a predição interquadros divide-se em estimação de movimento e compensação de movimento. Após a predição, são gerados os resíduos, através da subtração entre os valores do quadro original e os resultados da predição (RICHARDSON, 2002) (AGOSTINI, 2007). A seguir, o resíduo passa pelo módulo das transformadas diretas e pelo módulo da quantização direta para que seja reduzida a redundância espacial no domínio das frequências (AGOSTINI, 2007). O primeiro módulo transforma a informação do domínio espacial para o domínio das frequências e o segundo é aplicado para reduzir a redundância existente nos resíduos transformados. Ao final, os dados provenientes da quantização são processados pela codificação de entropia. Este

último módulo reduz a redundância entrópica, relacionada à forma como os dados são codificados (AGOSTINI, 2007). Na Figura 1 também é possível ver um caminho de realimentação composto pelas transformadas inversas e pela quantização inversa. Esse caminho serve para que um quadro recém codificado possa ser reconstruído e utilizado como referência para próximos quadros a serem processados.

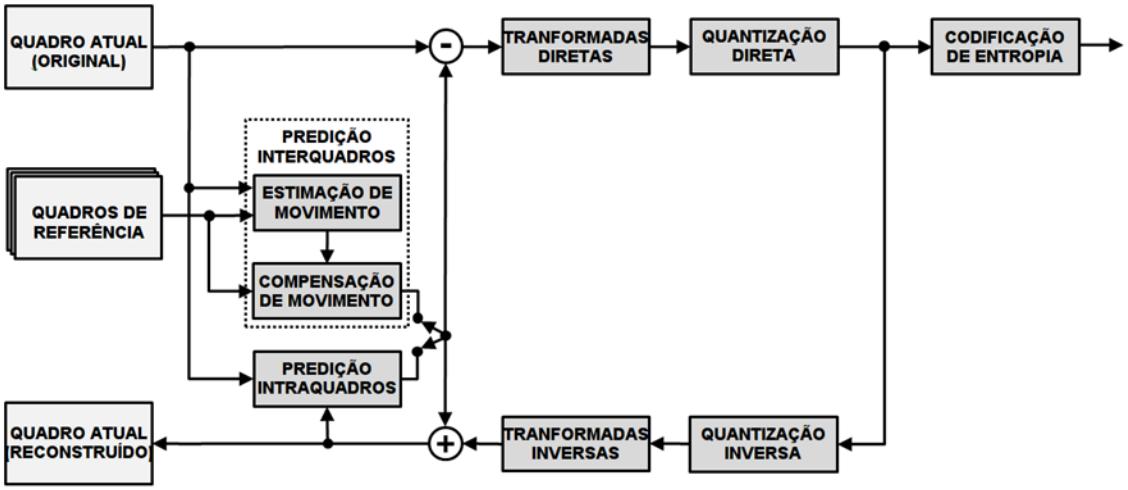


Figura 1 – Modelo completo de codificador de vídeo. Adaptado de (AGOSTINI, 2007).

Considerando que as arquiteturas desenvolvidas nesse trabalho serão compatíveis com o padrão HEVC, a seguir serão apresentados mais detalhes sobre o funcionamento das etapas de predição no padrão HEVC, foco das soluções desenvolvidas.

2.2 Particionamento de Quadros no Padrão HEVC

Desde o H.261, todos os padrões de codificação de vídeo da ITU-T e ISO / IEC seguem a abordagem da codificação de vídeo com base em blocos (SCHWARZ; SCHIERL; MARPE, 2014). Assim, cada quadro é particionado em blocos de amostras. A grande diferença entre as gerações de padrões de codificação de vídeo são os diferentes conjuntos de modos de codificação para os blocos de amostras. Os modos de codificação determinam, por exemplo, se um bloco de amostras será predito usando predição intraquadro ou interquadro (SCHWARZ; SCHIERL; MARPE, 2014).

Os padrões anteriores utilizavam o macrobloco como núcleo da codificação. Um macrobloco é formado por um bloco de 16×16 amostras de luminância e dois blocos com 8×8 amostras de crominância (considerando a amostragem de cores 4: 2: 0)

(SULLIVAN et al., 2012). Por sua vez, o padrão HEVC utiliza uma estrutura semelhante chamada *Coding Tree Unit* (CTU). O tamanho de uma CTU é selecionado pelo codificador e pode ser maior que um macrobloco tradicional (SULLIVAN et al., 2012).

Por sua vez, uma CTU é composta por um *Coding Tree Block* (CTB) com amostras de luminância e pelos CTBs com amostras de crominância relacionados ao CTB de luminância. O tamanho de um CTB de luminância é dado por *M*×*M*, com *M* podendo ser 16, 32, ou 64 amostras (SULLIVAN et al., 2012).

As CTUs podem ser divididas em unidades menores chamadas *Coding Units* (CUs). Esse particionamento pode ser representado por uma árvore quadrática (*quadtree*). Para ilustrar esta operação, a Figura 2 apresenta um exemplo. Na Figura 2, uma CTU de 64×64 amostras é dividida em CUs de 8×8, 16×16 e 32×32 amostras. O tamanho máximo de uma CU é o próprio tamanho da CTU que a contém. Por outro lado, o tamanho mínimo de uma CU é de 8×8 amostras. Sobre a composição de uma CU, esta é semelhante à composição de uma CTU. Assim, cada CU é composta por um *Coding Block* (CB) de luminância e dois CBs de crominância.

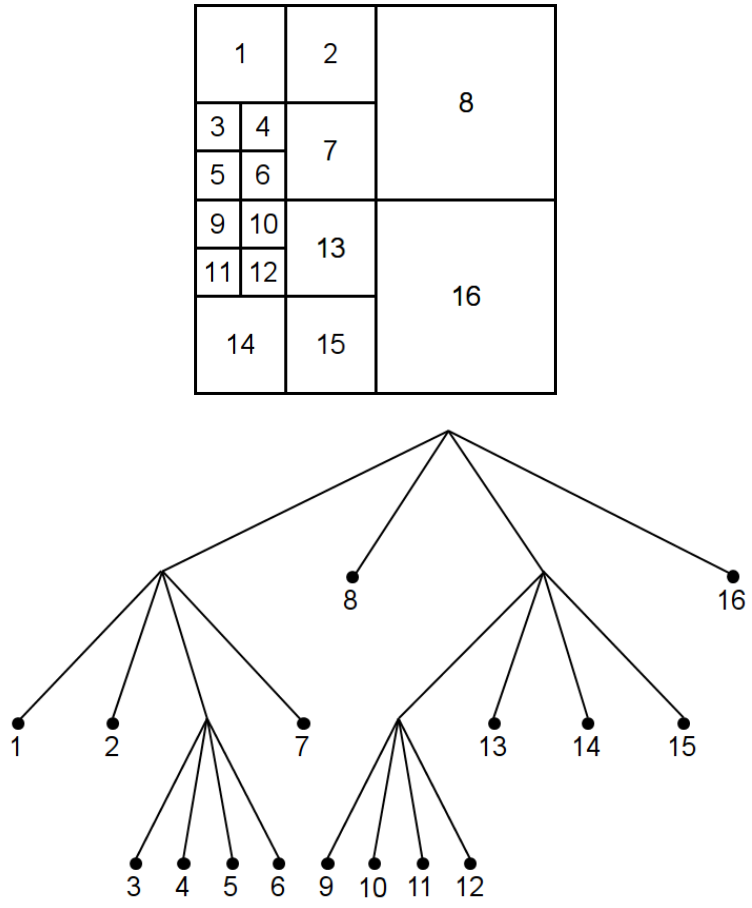


Figura 2 – Exemplo do particionamento de uma CTU em CUs. Adaptado de (SCHWARZ, 2014).

A decisão de codificar um quadro usando predição interquadro ou intraquadro é tomada no nível da CU (SULLIVAN et al., 2012), onde são definidas as *Prediction Units* (PUs). A partir dessa decisão, são formadas as PUs e os CBs são subdivididos em *Prediction Blocks* (PBs). Na predição intraquadro são permitidos apenas tamanhos quadrados de PUs. No caso da predição interquadros, além de PUs de formatos quadrados, também são permitidas PUs retangulares. A Figura 3 ilustra os formatos possível para PUs. Na Figura 3, *M* representa uma dimensão da CU.

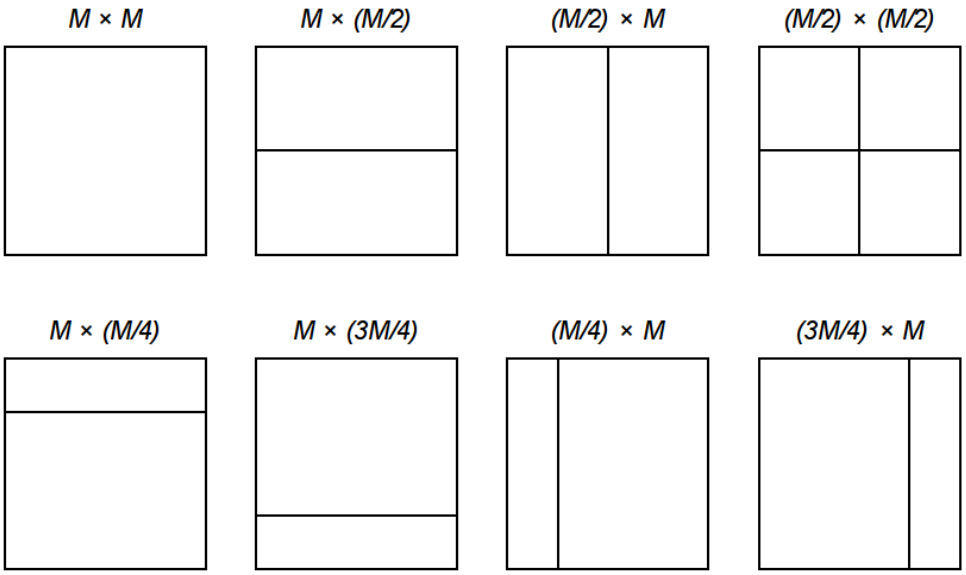


Figura 3 – Formatos possíveis de PUs. Adaptado de (SCHWARZ, 2014).

2.3 Predição Intraquadro no Padrão HEVC

O módulo de predição intraquadro do HEVC, ou predição intraquadro, suporta 35 modos, sendo 33 modos direcionais e dois não direcionais (LAINEMA et al., 2012). Os modos direcionais são adequados para áreas com estruturas direcionais e os dois modos restantes, Planar e DC, são adequados para áreas homogêneas.

A Figura 4 mostra, como exemplo, um bloco 8×8 (quadrados brancos) a ser predito usando-se 33 amostras de referência codificadas anteriormente (quadrados não brancos). De forma geral, para realizar a predição de cada bloco *N*×*N*, são necessárias *4N+1* amostras de referência. Todo bloco predito com a predição intraquadro deve passar por três etapas diferentes: pré-filtragem de amostras de referência, predição de amostras e pós-filtragem de amostras preditas (LAINEMA et al., 2012). A pré-filtragem é usada quando as amostras de referência adjacentes apresentam discrepâncias

notáveis em seus valores. Nesses casos, artefatos indesejados podem aparecer em blocos preditos. Para reduzir esse efeito, filtros de suavização são aplicados às amostras de referência antes da predição do bloco. O filtro adotado é função do tamanho do bloco e do modo de predição utilizado (LAINEMA et al., 2012).

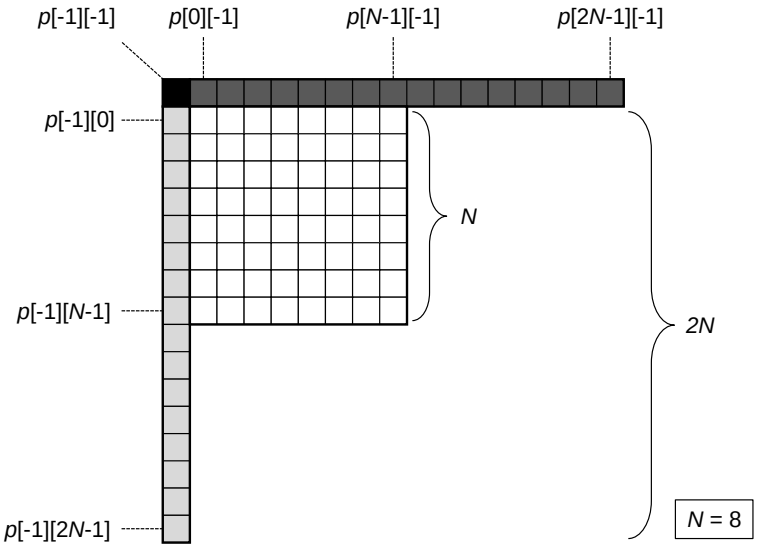


Figura 4 – Um bloco de 8×8 *pixels* a ser predito usando-se 33 amostras de referência. Fonte: CORRÊA et al., 2017.

A etapa de predição de amostra é onde a predição realmente ocorre. Nela, os blocos são calculados usando-se os 35 modos de predição disponíveis. Também é nessa etapa que os blocos são comparados com o bloco atual para selecionar quais são as melhores opções de codificação. Esta etapa é o núcleo da predição intraquadro.

Considerando um bloco de codificação em árvore (CTB) de 64×64 *pixels* (SULLIVAN et al., 2012), os preditores são aplicados em quatro tamanhos de bloco diferentes: 4×4, 8×8, 16×16 e 32×32 (LAINEMA et al., 2012). Como existem 35 modos de predição, 140 combinações são permitidas. As predições consideradas são comparadas com o bloco atual usando algum critério de distorção (WIEN, 2014). A implementação da predição intraquadro no HM, *HEVC Test Model*, software de referência do HEVC (HM, 2015), permite o uso dos critérios de distorção a Soma das Diferenças Absolutas (SAD) e a Soma das Diferenças Absolutas Transformadas (SATD) (WIEN, 2014). O critério de distorção considerado para a arquitetura que será apresentada no Capítulo 4 é o SAD. A escolha se deve ao fato de o SAD ser o critério de distorção mais simples de ser implementado em hardware e, portanto, o mais usado

em arquiteturas para codificação de vídeos (WIEN, 2014). Essa distorção deve ser calculada para todos os tamanhos de bloco disponíveis dentro de um CTB, de 4×4 a 64×64 (ZHOU et al., 2014). Como não há preditores para blocos de 64×64, os quatro blocos de 32×32 que formam o bloco de 64×64 são unidos para gerar a predição do bloco maior (HM, 2015).

A última etapa da predição intraquadro é o filtro de pós-processamento. Este é usado para reduzir as descontinuidades que alguns dos modos predição intraquadro podem gerar para as amostras previstas localizadas nas bordas superior e esquerda do bloco predito (LAINEMA et al., 2012).

2.4 Predição Interquadros no Padrão HEVC

Embora a etapa de predição interquadro seja composta também pela compensação de movimento, nesta seção será discutida apenas a etapa de estimação de movimento, foco dos trabalhos apresentados no Capítulo 5.

A estimação de movimento (ME) é uma das operações mais complexas e que demandam mais energia em um codificador de vídeo (KIM et al., 2013). A estimação de movimento consiste em uma busca pela melhor correspondência de cada bloco do quadro atual dentro de um ou mais quadros de referência processados anteriormente. O bloco do quadro atual cuja correspondência será buscada é chamado de bloco atual. Os blocos a serem avaliados, oriundos dos quadros de referência, são os blocos candidatos. Um algoritmo de busca define como a pesquisa. Para reduzir a complexidade desse processo, a busca é implementada apenas em uma região nos quadros de referência, chamada de área de busca (RICHARDSON, 2002). Ao final da busca, o bloco mais semelhante ao bloco atual, definido com melhor candidato, é apontado pelo vetor de movimento. A Figura 5 ilustra esse processo. Além disso, há a necessidade de uma métrica de distorção, ou seja, que seja utilizado um critério para definir a melhor correspondência. Dentre esses critérios, novamente o SAD é o mais utilizado, principalmente quando se considera soluções em hardware dedicadas (RICHARDSON, 2002).

Os padrões atuais de codificação de vídeo, como o padrão HEVC, por exemplo, suportam vários tamanhos de bloco. Dessa forma, a etapa de ME deve ser aplicada a cada um desses tamanhos de bloco.

O algoritmo de busca completa, *Full Search* (FS), é o algoritmo de busca que sempre encontra os melhores resultados, pois é capaz de avaliar todos os blocos candidatos possíveis dentro da área de busca. Por outro lado, o FS requer um esforço computacional proibitivo e muitos algoritmos de busca rápida foram propostos na literatura com o objetivo de superar essa limitação. Esses algoritmos atingem resultados quase ótimos. No caso do padrão HEVC, o algoritmo *Test Zone* Search (TZS) é uma alternativa implementada no software de referência.

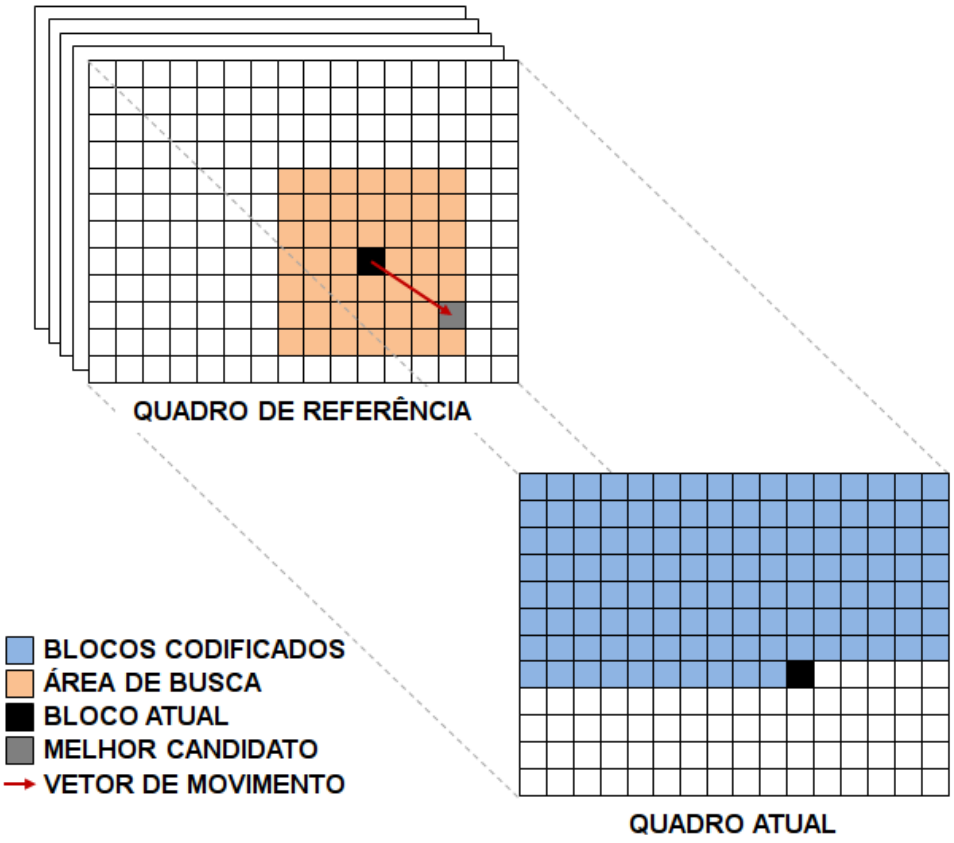


Figura 5 – Processo de estimação de movimento.

No padrão HEVC, assim como na maioria dos codificadores de vídeo atuais, também é permitido o uso de estimação de movimento fracionária (FME). Essa etapa é executada para aumentar a eficiência da ME, uma vez que a estimação de movimento inteira geralmente não é capaz de obter a melhor correspondência entre dois quadros através dessas posições inteiras (HEVC, 2015) (SULLIVAN et al., 2012). As posições fracionárias são obtidas pela interpolação de posições inteiras em uma vizinhança no bloco.

2.5 Complexidade das Operações nas Etapas de Predição do Padrão HEVC

O padrão HEVC foi idealizado para aumentar de forma significante o desempenho da compressão de vídeo em relação ao seu padrão antecessor (SULLIVAN et al., 2012). Assim, o padrão HEVC alcança uma taxa de compressão que é duas vezes maior que a taxa de compressão do padrão H.264 mantendo a mesma qualidade. Em contraponto a esse aumento no desempenho da compressão houve o aumento na complexidade das operações utilizadas na codificação.

Em relação à etapa de predição intraquadro, há 35 modos de predição para serem avaliados para bloco resultante do particionamento. Para tornar a tarefa da predição intraquadro ainda mais complexa, o núcleo desta etapa é o cálculo de SAD, uma operação executada de forma exaustiva.

Quanto à complexidade da predição interquadros, esta depende do algoritmo de busca utilizado, do número de quadros de referência suportados, do número de tamanhos de blocos suportados, do suporte ou não à FME, do tamanho da área de busca e da métrica de distorção usada.

3 COMPUTAÇÃO APROXIMADA

As aplicações multimídia estão migrando cada vez mais para dispositivos alimentados por bateria, como smartphones, câmeras digitais, entre outros. Nesse cenário, não só a eficiência da codificação de vídeo deve ser fortemente considerada. Além dela, a taxa de processamento, a dissipação de potência e o consumo de energia também devem ser considerados como aspectos fundamentais. Considerando-se o esforço computacional muito alto que já é exigido pelos codificadores de vídeo atuais, projetos de hardware dedicado têm sido amplamente utilizados para permitir o processamento em tempo real de conteúdo de vídeo com eficiência energética.

Para superar essa demanda, a computação aproximada tem sido considerada uma abordagem extremamente promissora para alcançar eficiência energética. As técnicas de computação aproximada exploram as características de aplicações tolerantes a erros, ou seja, aplicações que são resilientes a uma pequena perda de precisão ou a resultados parciais imprecisos numericamente. No escopo da codificação de vídeos, a introdução de uma quantidade limitada de imprecisão numérica na implementação de vários algoritmos geralmente resulta em perdas aceitáveis na eficiência de codificação. Isso se deve, principalmente, às características de tolerância a erros das ferramentas de codificação.

3.1 Computação Aproximada nas Etapas de Predição

Essa seção discute a utilização de técnicas de computação aproximada nas etapas de predição interquadros e intraquadro especificamente. Essa discussão serve como base para as decisões de projeto das arquiteturas que serão apresentadas nos capítulos 4, 5 e 6.

A predição interquadros é a operação que demanda mais tempo de execução dentre todas as etapas da codificação de vídeo dos padrões atuais. Em (KIM et al., 2013), o tempo de execução de cada ferramenta é detalhado utilizando-

se como exemplo o padrão HEVC (HEVC, 2015) (SULLIVAN et al., 2012). Segundo a avaliação realizada pelos autores, a predição interquadros gasta de 77% a 81% do tempo total de execução. Esse resultado é influenciado pelas repetidas vezes em que se calcula a métrica de distorção dentro da predição.

Dentre as métricas de distorção, o SAD é o mais comumente usado nos codificadores de vídeo atuais, especialmente naqueles projetados em hardware. Em muitos codificadores o SAD é usado em todas as etapas de predição da codificação de vídeo, tanto interquadros quanto intraquadro (FAN et al., 2018) (SINGH; AHAMED, 2018) (PASTUSZAK; ABRAMOWSKI, 2016) (XIONG et al., 2015). Isso torna o cálculo do SAD uma operação crucial para definir a eficiência do hardware do codificador. Por outro lado, as etapas de predição são naturalmente resilientes a pequenas imprecisões. Dessa forma, o cálculo do SAD é uma operação bastante promissora para a aplicação de técnicas de computação aproximada. Através dessas técnicas é possível reduzir a área do codificador e o consumo de energia e impactar minimamente a eficiência da codificação.

O SAD calcula a distorção entre as regiões comparadas para cada ponto do quadro original e do quadro reconstruído (KUHN, 1999) (AGOSTINI, 2007). O SAD está definido na Equação 1, onde *m* e *n* são as dimensões do quadro a ser comparado e *R* e *O* referem-se às amostras do quadro original e do quadro reconstruído, respectivamente (AGOSTINI, 2007).

$$SAD = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \left| R_{i,j} - O_{i,j} \right| \quad (1)$$

Considerando a Equação 1, é possível notar que há uma subtração acompanhada da operação de módulo e, após, há diversas somas sucessivas para acumular as diferenças calculadas. Assim, para implementar uma unidade para cálculo de SAD é necessário a utilização de uma grande quantidade de operadores aritméticos. Apesar de ser possível obter melhorias energéticas na codificação através de modificações nos algoritmos, outra alternativa promissora é realizar simplificações nos operadores, visto que eles são utilizados em grande número. As alterações mais comuns são encurtar ou truncar a cadeia de propagação do *carry*. Operadores que possuam estas alterações

são considerados operadores imprecisos. Uma definição mais ampla e alguns tipos de operadores imprecisos são apresentados na próxima seção.

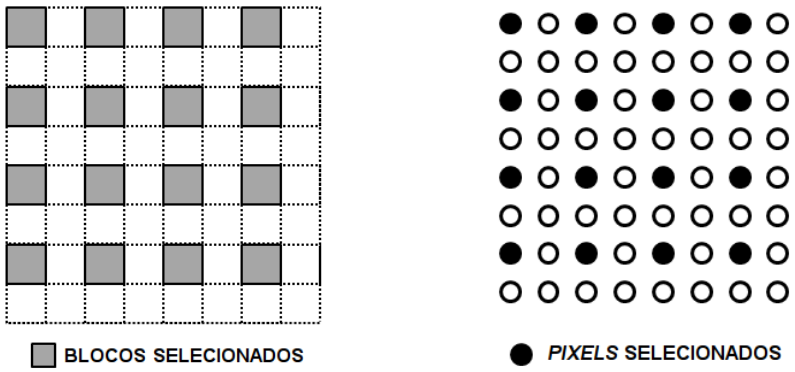
Existem muitas possibilidades de aplicação de técnicas de computação aproximada para reduzir o consumo de energia (SULLIVAN et al., 2012) (HAN; ORSHANSKY, 2013). No caso da estimação de movimento, é possível aplicar a computação aproximada tanto no nível do algoritmo quanto no nível dos operadores aritméticos.

As simplificações em nível de algoritmo têm sido as abordagens mais usadas para aliviar as demandas computacionais da estimação de movimento. Nesse sentido, existem várias possibilidades de simplificação para reduzir o consumo de energia. Uma abordagem possível é considerar algoritmos de busca rápidos (ALVAR; ABDOLLAHZADEH; SEYEDARABI, 2014) (CHIANG; KUO; SU, 2007). Esse tipo de busca realiza menos comparações que o algoritmo de busca completa. No entanto, esses algoritmos levam a bons resultados. Um exemplo disso é o algoritmo *Test Zone Search* (TZS) (TANG; DAI; CAI, 2010), que é o algoritmo padrão no software de referência do padrão HEVC.

Outra alternativa para obter redução de energia é usar alguma técnica de subamostragem, como subamostragem de *pixels* ou subamostragem de blocos (HWANG; HA; SUNWOO, 2009) (KANG et al., 2008). Na subamostragem de *pixels*, se tivermos uma taxa de 4:1, por exemplo, significa que de cada quatro *pixels* utilizaremos apenas um. Assim, o número de amostras a serem consideradas dentro de um bloco passa a ser ¼ da quantidade original. Da mesma forma, se utilizarmos subamostragem de blocos e utilizarmos uma taxa de 2:1, por exemplo, o número de blocos candidatos dentro de uma área de busca cai pela metade. Esses dois tipos de subamostragem são ilustrados na Figura 6. Qualquer algoritmo de busca pode usar a técnica subamostragem de *pixels*. Porém, a subamostragem de blocos só é possível no algoritmo de busca completa (AGOSTINI, 2007). Como no caso anterior, nas subamostragens a redução no consumo de energia é obtida reduzindo-se o número de amostras usadas nos cálculos.

Outra possibilidade nas aproximações em nível de algoritmo é realizar simplificações nos tamanhos de bloco avaliados. Os padrões de codificação de vídeo especificam uma grande variedade de tamanhos de bloco que podem ser avaliados na estimação de movimento. No padrão HEVC, por exemplo, podem ser avaliados

24 tamanhos de bloco diferentes, considerando-se tamanhos de bloco simétricos e assimétricos (SULLIVAN et al., 2012). É possível simplificar esta tarefa executando a estimação de movimento apenas para alguns tamanhos de bloco e, ainda assim, manter a conformidade com o padrão.



(a) (b)

Figura 6 – Exemplo de aplicação de subamostragem (a) de blocos e (b) de pixels.

Essas três oportunidades de aplicação de computação aproximada no nível algorítmico na predição interquadros podem reduzir significativamente o esforço computacional necessário e o consumo de energia resultante. Por outro lado, o custo dessas simplificações são eventuais degradações na eficiência da codificação. Dessa forma, essas soluções devem ser cuidadosamente aplicadas.

Ao contrário da predição interquadros, a predição intraquadro é uma operação que demanda apenas de 1% a 2% do tempo total de execução de todas as etapas da codificação do padrão HEVC (KIM et al., 2013). Porém, a predição intraquadro do HEVC é capaz de processar blocos de tamanhos 4×4, 8×8, 16×16 e 32×32 *pixels*, sendo que cada tamanho de bloco suporta 35 modos de predição diferentes, como já discutido. Isso resulta em 132 combinações possíveis. Assim, da mesma forma que para a predição interquadros, na predição intraquadro é possível realizar simplificações em nível de algoritmo e diminuir a quantidade de tamanhos de bloco avaliados ou diminuir o número de modos de predição avaliados.

Em relação ao núcleo das operações da predição intraquadro, da mesma forma que para a predição interquadros, os cálculos de SAD são as operações dominantes. Dessa forma, a mesma abordagem utilizada para a predição interquadros pode ser

utilizada também para a predição intraquadro, ou seja, é possível substituir operadores aritméticos precisos por operadores imprecisos. Com isso, reduz-se o esforço computacional necessário e o consumo de energia resultante, ao custo de eventuais degradações na eficiência da codificação. É importante destacar que os operadores imprecisos podem ser usados em combinação com outras formas de aproximação, melhorando ainda mais os resultados.

A seguir serão apresentados os operadores imprecisos que foram avaliados para utilização neste trabalho.

3.2 Operadores Imprecisos

Operadores aritméticos são circuitos essenciais em qualquer sistema digital e podem influenciar significativamente o desempenho geral do sistema (FRUSTACI et al., 2009). Grande parte do atraso e do consumo de energia de um operador aritmético convencional é devido à cadeia de propagação de *carry* (DUTT; NANDI; TRIVEDI, 2016) (ZHU et al., 2010). Isso se torna um agravante quando há a utilização de grande quantidade de operadores em uma aplicação, como acontece com as aplicações multimídia. Pela grande quantidade de operações aritméticas contidas em seus algoritmos, a codificação de vídeo é bom exemplo disso. A necessidade de redução no consumo de energia pode tornar-se ainda mais crítica se considerarmos o alvo da aplicação a ser desenvolvida. No caso de arquiteturas em hardware visando dispositivos móveis isso é uma das principais preocupações porque é necessário levar em conta a autonomia das baterias que alimentam esses dispositivos.

Para resolver esse problema de forma eficiente, algumas abordagens de computação aproximada são utilizadas nos operadores aritméticos tais como encurtar ou truncar a cadeia de propagação do *carry*. Operadores que possuam estas alterações são considerados operadores imprecisos. Dessa forma, se a aplicação alvo tolerar uma redução na precisão, podem ser obtidas melhorias no desempenho e redução no consumo de energia (HAN; ORSHANSKY, 2013) (VERMA; BRISK; IENNE, 2008).

Sendo assim, para que se obtenha a redução de consumo de energia desejada com mínimo impacto na qualidade da codificação, os operadores aritméticos convencionais devem dar lugar a soluções de baixo consumo. Entre as diversas soluções possíveis,

seis arquiteturas de operadores aritméticos imprecisos serão avaliados neste trabalho: *Lower-Part-OR*, ou LOA (MAHDIANI et al., 2010); *Error-Tolerant Adder*, ou ETA-I (ZHU et al., 2010); *Error-Tolerant Adder IV*, ou ETA-IV (ZHU et al., 2010a); *Accuracy-Configurable Adder*, ou ACA-II (KAHNG; KANG, 2012); *Generic Accuracy Configurable Adder*, ou GeAr (SHAFIQUE et al., 2015); e *Carry Cut-Back*, ou CCB (CAMUS; SCHLACHTER; ENZ, 2016). Estes operadores serão descritos a seguir.

3.2.1 *Lower-Part-OR Adder (LOA)*

Em operações aritméticas, erros em bits menos significativos causam menos impacto no resultado do que erros em bits mais significativos. Assim, pode-se dividir uma soma completa em duas partes e utilizar imprecisão em uma delas, obviamente, nos bits menos significativos (DUTT; NANDI; TRIVEDI, 2016).

No operador *Lower-Part-OR*, ou LOA (MAHDIANI et al., 2010), uma soma de *n* bits é dividida em duas partes menores, ou seja, em dois subsomadores. As duas partes resultantes não necessitam ter o mesmo tamanho em bits, o que possibilita variar o nível de imprecisão. A soma dos bits mais significativos é realizada da forma tradicional. Já para os bits menos significativos, é realizada uma simplificação eliminando-se a cadeia de propagação de *carry*. A Figura 7 apresenta a estrutura do somador LOA em detalhes.

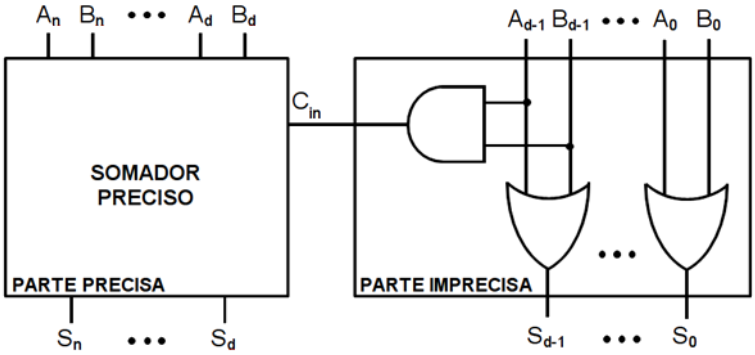


Figura 7 – Estrutura do operador LOA. Adaptado de (MAHDIANI et al., 2010).

A soma da parte precisa pode ser realizada utilizando-se qualquer arquitetura conhecida de somador completo como *Ripple Carry Adder* (RCA), *Carry Lookahead Adder* (CLA) (WESTE; HARRIS, 2010), etc. Na parte imprecisa, a soma é simplificada através de uma operação OR feita bit a bit. Dessa forma, não são usadas operações XOR (que são mais complexas) e não há uma cadeia de propagação de *carry* na parte

imprecisa da soma, o que impacta diretamente na redução de potência dissipada. Para diminuir a imprecisão, uma porta AND é utilizada na geração de um *carry-in* para a parte precisa através dos bits mais significativos das entradas da parte imprecisa. É importante destacar que, para o operador LOA, quanto maior for a parte imprecisa, maior será a redução de área, de atraso, e do consumo de energia. Em contrapartida, menor será a precisão nos resultados. Outro aspecto a ser considerado é o de que o operador LOA é uma solução bastante interessante para situações em que se necessite de somadores com pequena largura de bits. Nesses casos a magnitude da parte menos significativa representa uma porcentagem muito pequena do valor máximo que se pode representar e os erros ocasionados possuem pouca interferência no resultado final. Por fim, outro fator que torna interessante a utilização do operador LOA é sua simplicidade de implementação.

3.2.2 *Error-Tolerant Adder I (ETA-I)*

Os trabalhos (ZHU et al., 2010) e (ZHU et al., 2010a) propõem diversas versões de operadores aproximados às quais denominaram Somadores Tolerantes a Erros ou *Error-Tolerant Adders* (ETAs). A primeira versão de operador aproximado é apresentada em (ZHU et al., 2010) onde os autores propõem seu primeiro somador tolerante a erros chamado de *Error-Tolerant Adder* ou ETA-I. A ideia principal de funcionamento desse somador é apresentada na Figura 8.

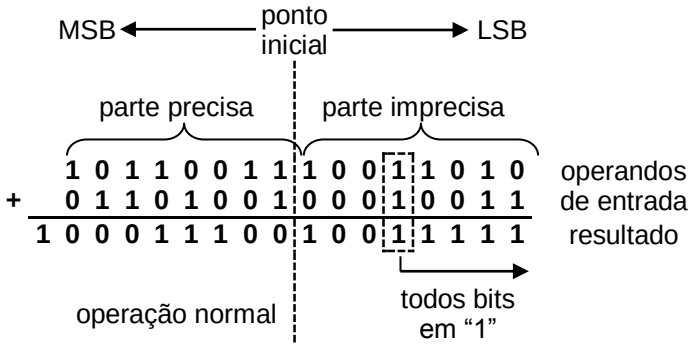


Figura 8 – Exemplo do esquema de operação do operador ETA-I. Adaptado de (ZHU et al., 2010).

Primeiramente, os operandos de entrada são divididos em duas partes: uma parte precisa que inclui os bits de ordem maior e uma parte imprecisa composta pelos bits menos significativos. Cada parte pode ter comprimentos diferentes. O processo de

soma inicia-se pelo ponto de junção das duas partes indo nas duas direções opostas simultaneamente. No exemplo da Figura 8, dois operandos de 16 bits são divididos em duas partes iguais de forma que os oito bits menos significativos componham a parte imprecisa e os oito bits mais significativos formem a parte precisa da operação.

Na parte precisa a operação de soma é realizada normalmente. Para a parte composta pelos bits menos significativos, outro método de adição é aplicado eliminando-se totalmente a cadeia de propagação de *carry*. Para minimizar o erro gerado por essa eliminação é aplicada uma estratégia especial. Essa estratégia funciona da seguinte forma: todos os bits são conferidos da esquerda para a direita; se ambos os bits de entrada forem *"0"* ou se forem diferentes, a soma convencional de 1 bit é aplicada e o processo segue para a próxima posição; se ambos os bits de entrada forem *"1"*, o processo de conferência é finalizado e, desta posição em diante, para a direita, todos os bits do resultado da soma recebem o valor *"1"*. De forma semelhante ao operador LOA, o operador ETA-I é simples de ser implementado e uma ótima solução onde sejam necessários somadores com pequeno tamanho em bits.

3.2.3 *Error-Tolerant Adder IV (ETA-IV)*

A versão IV do operador ETA é proposta em (ZHU et al., 2010a). O objetivo é resolver todas as deficiências das versões anteriores. O ETA-IV toma como base o operador *Carry Select Adder* (CSA) e dois tipos de geradores de *carry*, conforme apresentado na Figura 9.

É importante destacar que, se o valor de *X* for próximo de *N*, o consumo de energia aumenta significantemente enquanto o desempenho diminui. De acordo com (ZHU et al., 2010a), o atraso pode ser minimizado se os tamanhos dos geradores de soma e de *carry* forem escolhidos de forma adequada.

3.2.4 *Accuracy-Configurable Adder (ACA-II)*

Em (KAHNG; KANG, 2012) é discutido o fato de que, em algumas aplicações, há situações em que se necessita de resultados com maior precisão ou totalmente precisos mesmo usando-se operadores aproximados. Assim, os autores propõem um somador aproximado com precisão configurável.

Devido à sua configurabilidade, o operador *Accuracy-Configurable Adder* (ACA-II) pode operar de forma adaptativa tanto no modo aproximado como no modo preciso (KAHNG; KANG, 2012). Isso possibilita melhoria na relação entre precisão e dissipação

de potência. Um esquema de detecção e correção de erros no ACA-II também foi proposto e é realizado em um circuito complementar que, ao final de sua operação, disponibiliza nas saídas os resultados da soma correta ou os da soma aproximada.

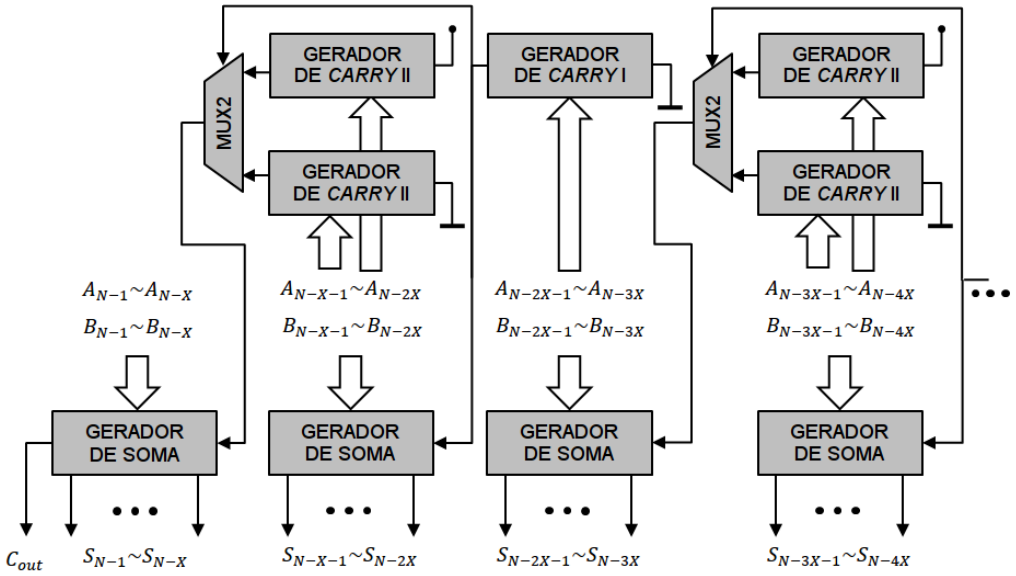


Figura 9 – Diagrama em blocos do operador ETA-IV. Adaptado de (ZHU et al., 2010a).

A Figura 10 apresenta o diagrama em blocos do operador ACA-II para uma soma de 16 bits. Para encurtar a cadeia de propagação de *carry* são utilizados subsomadores com áreas de soma sobrepostas.

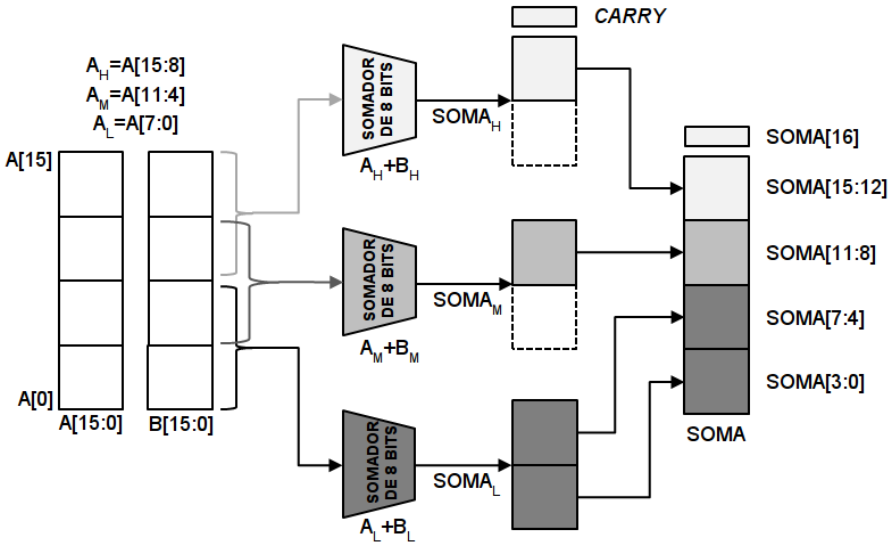


Figura 10 – Diagrama em blocos do operador ACA-II. Adaptado de (KAHNG; KANG, 2012).

No esquema apresentado na Figura 10 são utilizados três subsomadores de oito bits para realizar uma soma de 16 bits. Cada subsomador é responsável por uma faixa de soma. O subsomador da faixa intermediária (A_M+B_M) é utilizado para aumentar a precisão. Segundo experimentos realizados pelos autores, a taxa de erros é de 50,1% sem o uso do subsomador intermediário e cai para 5,5% com sua utilização. Os valores redundantes das somas, marcados em linha tracejada na Figura 10, são utilizados no circuito de detecção e correção de erros.

3.2.5 *Generic Accuracy Configurable Adder (GeAr)*

Um operador configurável de baixa latência e capaz de suportar vários modos de aproximação é apresentado em (SHAFIQUE et al., 2015) e recebeu o nome de *Generic Accuracy Configurable Adder* (GeAr). Esse operador possui um número maior de possíveis configurações se comparado com outros operadores do estado-da-arte. Isso permite um alto grau de flexibilidade no projeto.

Para definir a estrutura do operador os autores propuseram um modelo que é definido pela Equação 2

$$k = ((N - L)/R) + 1 \tag{2}$$

onde *N* é o tamanho do operador, *k* é o número de subsomadores a serem utilizados, *L* é o tamanho desses subsomadores, e *R* é o número de bits do subsomador que contribuem com o resultado final (exceto para primeiro subsomador que aplica todos os bits ao resultado).

Além disso, para definir *L* necessitamos somar *R* ao valor *P*, que representa o número de bits prévios utilizados para calcular o *carry* de cada subsomador. Portanto, para definir um operador GeAr necessitamos dos parâmetros *N*, *R*, e *P*. A Figura 11 apresenta um exemplo de operador GeAr através de um diagrama em blocos.

3.2.6 *Carry Cut-Back Adder (CCB)*

A arquitetura do operador *Carry Cut-Back* (CCB) é apresentada em (CAMUS; SCHLACHTER; ENZ, 2016). O nome desse operador deve-se à técnica de realimentação de estágios que ele utiliza. O resultado dessa realimentação é o encurtamento da cadeia de propagação do *carry*. A eficiência energética é garantida pelo fato de que o caminho crítico desse operador impreciso nunca será do tamanho do caminho crítico de um operador preciso equivalente.

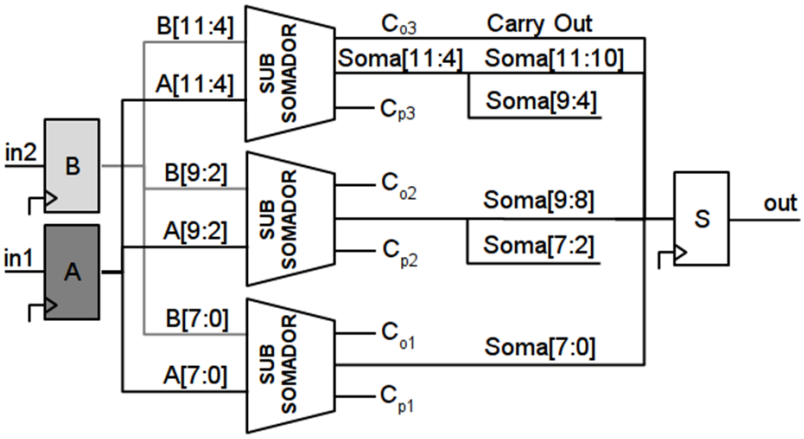


Figura 11 – Operador GeAr com N=12, R=2, P=6 e k=3. Adaptado de (SHAFIQUE et al., 2015).

A Figura 12 apresenta o diagrama em blocos do operador *Carry Cut-Back*. A soma nesse operador é feita de forma convencional pelos blocos de soma. Juntamente com esses blocos há vários multiplexadores que são utilizados para cortar a cadeia de propagação de *carry* e diminuir o caminho crítico do circuito (CAMUS; SCHLACHTER; ENZ, 2016).

Na Figura 12 é possível notar que a cadeia de propagação é cortada através de sinais gerados pelos blocos *PROP*. Esse sinal controla um multiplexador que disponibilizará em sua saída, ou o *carry* real calculado pelo bloco *SOMA*, ou o *carry* especulado a partir de uma cadeia curta dentro do bloco de especulação de *carry* (*SPEC*).

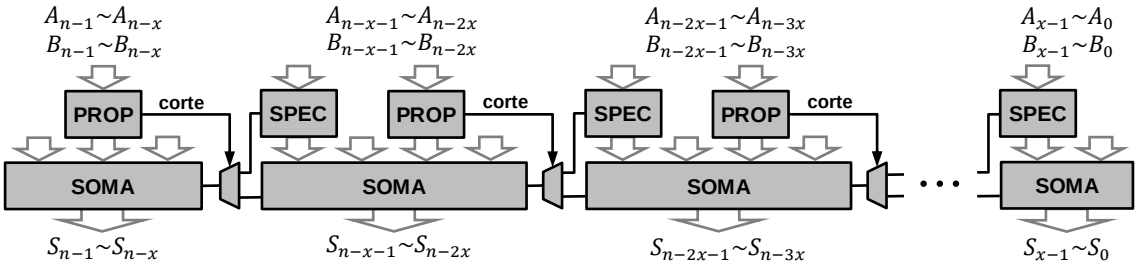


Figura 12 – Diagrama em blocos do operador CCB (com corte através de multiplexadores). Adaptado de (CAMUS; SCHLACHTER; ENZ,2016).

4 ARQUITETURA DE SAD PARA A PREDIÇÃO INTRAQUADRO COM BAIXA DISSIPAÇÃO DE POTÊNCIA E USO DE COMPUTAÇÃO APROXIMADA

Este capítulo apresenta uma arquitetura dedicada para cálculo de SAD e que possui baixa dissipação de potência através do uso de operadores imprecisos. Esta arquitetura para cálculo de SAD foi projetada especialmente para à predição intraquadro do HEVC. Além disso, ela é escalável em qualidade e potência dissipada e é capaz de codificar vídeos na resolução UHD 8K. A escalabilidade em qualidade e potência dissipada foi obtida usando-se computação aproximada através da implementação de operadores LOA em três pontos de operação imprecisos, além de uma versão precisa com somadores RCA.

A arquitetura desenvolvida neste trabalho tem como base a solução apresentada em (CORRÊA et al., 2017), que consegue processar vídeos UHD 8K em tempo real, mas não oferece configurabilidade. Além disso, a solução apresentada também pode ser usada em arquiteturas para predição interquadros ou como bloco básico para outros critérios de similaridade, como o SATD.

A ideia principal explorada neste trabalho é o uso de diferentes níveis de aritmética imprecisa para controlar a dissipação de potência de uma arquitetura de alto desempenho para predição intraquadro. A etapa de cálculo de SAD é uma parte computacionalmente intensiva nessa implementação. Assim, ela foi selecionada para receber operadores imprecisos.

4.1 Avaliação dos Operadores Imprecisos

Esta primeira avaliação considerou 26 configurações diferentes de seis operadores imprecisos. Os operadores avaliados foram descritos em C++ e estimulados através de 99.840 amostras. Essas amostras foram extraídas do primeiro quadro da sequência de

teste *BasketballPass_416x240_50.yuv*. Como as amostras de vídeo têm oito bits de largura, a largura dos somadores também foi definida como oito bits. Os resultados imprecisos dos operadores também foram comparados com os resultados de um somador convencional (preciso). Os critérios de avaliação escolhidos foram erro médio e desvio padrão.

De acordo com essa avaliação preliminar, as configurações de operadores com os melhores resultados foram:

- ACA-II usando três subsomadores de quatro bits com áreas de soma sobrepostas;
- CCB usando quatro subsomadores de dois bits e o sinal de *cut-back* definido por um bit;
- ETA-I usando três bits precisos e cinco imprecisos;
- ETA-IV usando dois subsomadores de três bits e um de dois bits, com dois bits no primeiro gerador de *carry* e três bits no segundo;
- GeAr usando dois subsomadores sobrepostos de 5 bits;
- LOA usando três bits precisos e cinco imprecisos.

A Tabela 1 mostra um resumo dos resultados de melhor configuração para os seis somadores imprecisos avaliados.

Tabela 1 – Erro médio e desvio padrão para as melhores configurações dos seis operadores imprecisos considerados nesse trabalho.

Operador	**Erro Médio**	**Desvio Padrão**
ACA-II	5,34	11,60
CCB	7,13	6,88
ETA-I	15,44	32,67
ETA-IV	1	0
GeAr	4,27	9,56
LOA	13,82	29,32

A segunda avaliação identificou, entre os seis operadores imprecisos selecionados anteriormente, as configurações que fornecem os melhores resultados no contexto específico da predição intraquadro do padrão HEVC. Para isso, o software de referência do padrão HEVC, foi usado para medir os impactos na eficiência da codificação. Dessa forma, além da versão original do HM, outras seis versões modificadas foram geradas, cada uma delas utilizando um dos somadores imprecisos apresentados anteriormente. Os operadores imprecisos foram utilizados apenas no

primeiro estágio de cálculo de SAD na predição intraquadro do HM para evitar efeitos de acumulação de erros. No cálculo de SAD, a primeira operação corresponde à subtração necessária para gerar a soma das diferenças absolutas. Além disso, foi desativado o uso da transformada Hadamard na etapa de predição para garantir que apenas fossem executadas operações de SAD e não operações de SATD (HM, 2015).

Os resultados deste experimento foram avaliados utilizando-se a métrica *BD-rate* (BJONTEGAARD, 2001), que representa o aumento ou diminuição percentual no número de bits necessários para representar o vídeo codificado, considerando a mesma qualidade de imagem objetiva (PSNR). Este experimento considerou as Condições de Comuns de Teste (*Common Test Conditions* – CTC) (BOSSEN, 2013) definidas pela comunidade do HEVC. Assim, foram utilizadas as 24 sequências de vídeo recomendadas nas CTC (com resoluções variando de 2560×1600 a 416×240 pixels) e quatro valores de QP (22, 27, 32 e 37). Dessa forma, foram geradas 576 avaliações. A configuração utilizada para o HM foi *All Intra* (BOSSEN, 2013), onde só a predição intraquadro está disponível.

Os resultados dessa avaliação são apresentados na Tabela 2 e estão organizados de acordo com as seis classes de vídeos de teste definidas nas CTC. De acordo com esses resultados, os menores impactos em *BD-rate* foram obtidos para os operadores ETA-IV e LOA, uma vez que esses somadores apresentam os melhores resultados médios.

Tabela 2 – Aumento de BD-rate (%) resultante do uso de operadores imprecisos.

	ACA-II	**CCB**	**ETA-I**	**ETA-IV**	**GeAr**	**LOA**
Classe A	1,81	1,09	1,11	0,39	1,21	0,65
Classe B	2,36	1,32	1,38	0,39	1,45	0,77
Classe C	2,92	1,39	1,39	0,33	1,66	0,76
Classe D	2,42	1,23	1,26	0,35	1,33	0,71
Classe E	3,76	2,38	2,31	0,65	2,48	1,19
Classe F	2,22	1,58	0,67	0,05	1,33	0,11
Média	**2,52**	**1,45**	**1,32**	**0,35**	**1,53**	**0,68**

Como o foco da arquitetura apresentada neste capítulo é fornecer uma solução escalável em dissipação de potência e qualidade, foi realizado um experimento complementar para avaliar dissipação de potência dos somadores imprecisos de forma isolada em uma implementação em hardware. Assim, esses operadores foram avaliados considerando sua dissipação média de potência. Essa avaliação de potência foi feita usando mais de dois bilhões de amostras extraídas da sequência de teste

FourPeople (BOSSEN, 2013). Os operadores foram descritos em VHDL estrutural e sintetizados para a biblioteca de células padrão Nangate 45nm (NANGATE, 2019), usando a ferramenta *Encounter RTL Compiler*, da Cadence. Na síntese, a frequência alvo foi definida como 269MHz e o esforço de síntese da ferramenta foi definido como "forte".

A Tabela 3 apresenta os resultados obtidos considerando a dissipação de potência, o atraso, a área e o produto potência×atraso (PPA) para os seis operadores imprecisos (nas configurações descritas anteriormente) e para dois operadores precisos: *Ripple Carry Adder* (RCA) e *Carry Lookahead Adder* (CLA).

Tabela 3 – Avaliação em hardware dos operadores imprecisos.

Operador	Potência (μW)	Área (portas)	Atraso (ps)	PPA ($x10^{-3}$)
RCA	129	535	905	116,75
CLA	137	579	774	106,04
ACA-II	130	516	786	102,18
CCB	125	506	941	117,63
ETA-I	106	512	814	86,28
ETA-IV	134	549	829	111,09
GeAr	128	496	805	103,04
LOA	105	480	693	72,77

A fim de facilitar essa comparação multivariável, a Figura 13 apresenta os resultados de implementação através de gráficos de radar. Os diferentes eixos representam a porcentagem de aumento ou diminuição em cada critério quando comparados com o operador RCA. A menor área em cinza representa o melhor resultado quando todas as variáveis comparadas são consideradas com o mesmo peso. Para permitir uma comparação completa, os resultados de BD-rate do experimento anterior também foram inseridos nesses gráficos. De acordo com os resultados de hardware, o melhor somador em todos os critérios avaliados foi o operador LOA, apresentando excelentes resultados em atraso e PPA. O somador GeAr ficou em segundo lugar em termos de uso da área, enquanto o operador ACA-II foi o segundo em atraso. O somador ETA-I foi o segundo em dissipação de potência e em PPA. Conforme apresentado na Tabela 3, alguns operadores imprecisos obtiveram resultados até piores do que os resultados dos somadores RCA e CLA em alguns critérios comparados (resultados de PPA, por exemplo). Apesar de ter a maior dissipação de energia e a maior área, o operador CLA apresentou um atraso bastante

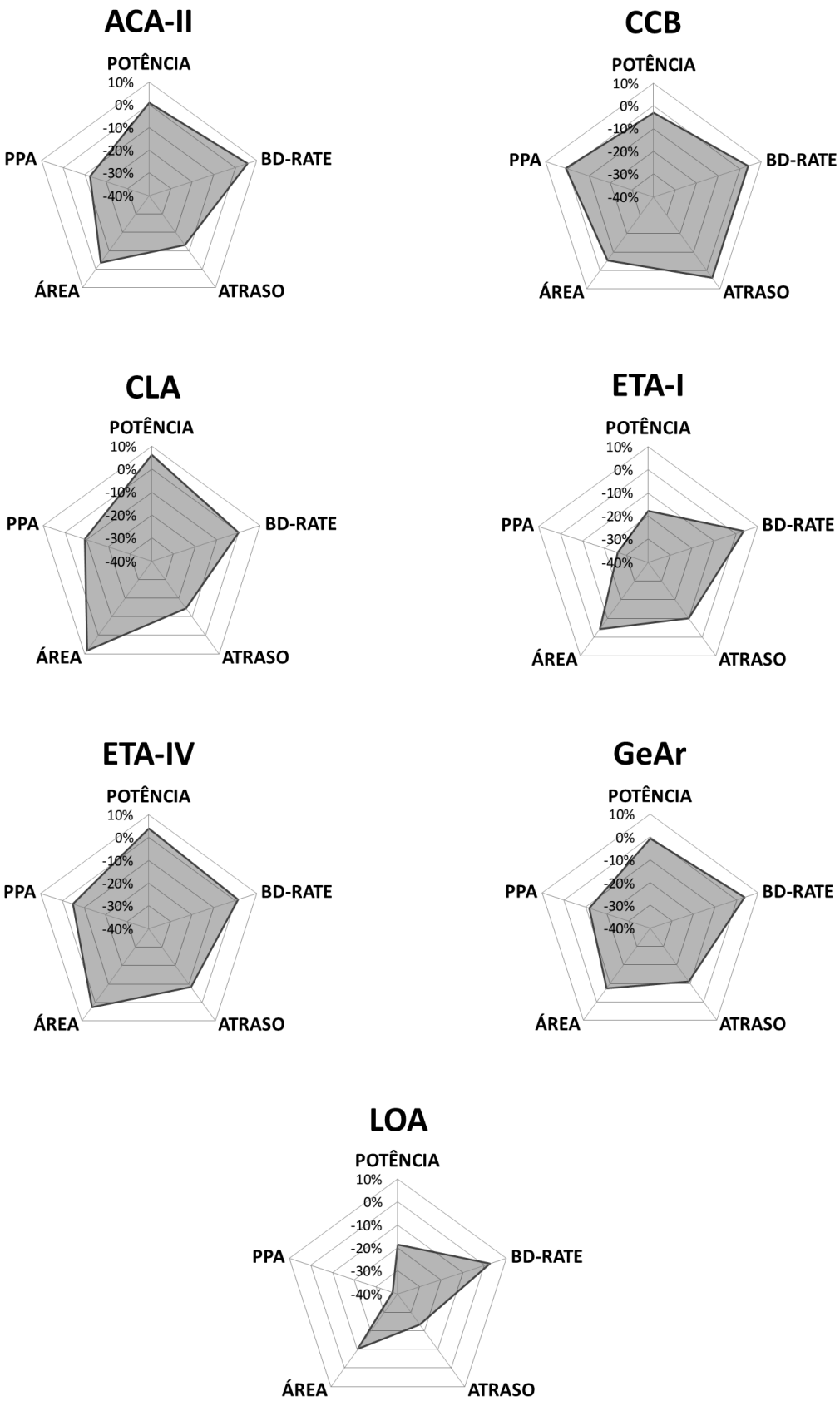


Figura 13 – Análise multivariável de operadores.

reduzido. Dessa forma, seu PPA foi menor do que o de alguns operadores imprecisos como o CCB e o ETA-IV.

Como citado anteriormente, a menor área cinza entre todos os gráficos de radar representa o melhor resultado. Nesse sentido, pode-se concluir que o operador LOA é a melhor solução. Embora o operador LOA tenha o resultado de BD-rate ligeiramente pior do que o do operador ETA-IV, os resultados de hardware do somador LOA são muito melhores do que os resultados do somador ETA-IV em todos os critérios considerados. Além disso, o operador LOA também tem uma característica interessante: o nível de imprecisão pode ser alterado. Assim, é possível projetar uma solução configurável usando vários níveis de imprecisão, com impactos distintos em área, atraso e, principalmente, em potência. Esta discussão será detalhada na próxima seção.

Por fim, considerando as características do operador LOA e os resultados da avaliação apresentada, este operador foi selecionado para ser usado na arquitetura apresentada neste capítulo. O operador LOA apresenta o melhor suporte para o projeto de uma arquitetura para cálculo de SAD que seja escalável em qualidade e energia. Além disso, ele alcançou os melhores resultados em termos de dissipação de energia e atraso, o que também é importante para processar vídeos de resolução 8K de maneira eficiente em termos de energia.

4.2 Unidade de Cálculo de SAD Escalável em Energia e Qualidade

Esta seção apresenta a arquitetura da unidade de SAD proposta para ser escalável em qualidade e energia. Esta arquitetura foi projetada para ser totalmente compatível com o módulo de predição intraquadro previamente proposto em (CORRÊA et al., 2017). Dessa forma, ela deve suportar todos os 35 modos de predição intraquadro e ser capaz de processar CTBs de 64×64 pixels. Cada CTB de 64×64 pixels contém um total de 256, 64, 16, 4 e 1 blocos de tamanhos 4×4, 8×8, 16×16, 32×32 e 64×64 pixels, respectivamente. Portanto, ao considerar um CTB de 64×64 pixels, 341 blocos devem ser processados individualmente.

4.2.1 *Arquitetura da Unidade de Cálculo de SAD*

O diagrama de blocos da unidade de SAD proposta em (CORRÊA et al., 2017) é apresentado na Figura 14. Essa arquitetura foi projetada para processar vídeos de ultra-alta definição em tempo real suportando a codificação de vídeos UHD 8K. Para

alcançar essa elevada taxa de processamento, a arquitetura utiliza 35 árvores de SAD em paralelo. Cada árvore de SAD pode processar um bloco de entrada de 16×16, 8×8 ou 4×4 pixels em um único ciclo de *clock*. No caso de blocos maiores, como de 32×32 e 64×64 pixels, estes são processados em quatro e 16 ciclos de *clock*, respectivamente.

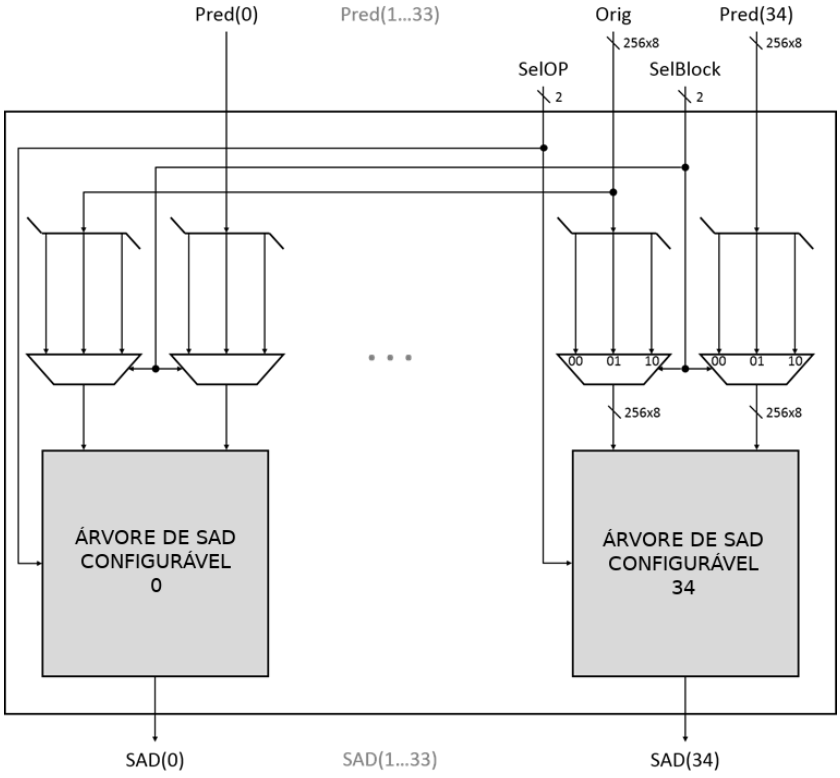


Figura 14 – Diagrama de blocos da unidade de SAD configurável. Adaptado de (CORRÊA et al., 2017)

A interface externa da unidade de cálculo de SAD recebe simultaneamente as amostras de entrada de 8 bits do bloco original e as amostras correspondentes dos 35 blocos preditos. Cada árvore de SAD recebe um bloco candidato diferente, mas todas recebem o mesmo bloco a ser predito. O controle do tamanho do bloco é feito através do sinal *SelBlock*. Esta unidade também gera os valores de SAD dos 35 blocos preditos, usando 16 bits.

A Figura 14 mostra a arquitetura interna da unidade de cálculo de SAD. Esta arquitetura é composta por uma matriz de árvores de SAD. Cada árvore é numerada de 0 a 34 de forma a corresponder a cada modo de predição intraquadro. O sinal *SelBlock* controla o multiplexador responsável por selecionar os blocos de tamanho 4×4, 8×8,

16×16 pixels e ¼ dos blocos 32×32 pixels quando os valores do sinal são *"00"*, *"01"*, *"10"* e *"11"*, respectivamente.

Considerando o elevado nível de paralelismo adotado nessa arquitetura para predição intraquadro, o processamento de um CTB completo, composto pelos 341 blocos candidatos, requer um total de 368 ciclos de *clock*. Como cada quadro de vídeo UHD 8K (7680×4320 pixels) possui um total de 12.150 CTBs de 64×64 pixels (considerando uma subamostragem de cores 4:2:0 (WIEN, 2014)), um quadro é processado em 4.471.200 ciclos de *clock* e é necessária uma frequência mínima de operação de 268,3 MHz para processar 60 quadros por segundo.

4.2.2 *Definição dos Pontos de Operação Imprecisos*

As avaliações preliminares realizadas na seção 4.1 identificaram o operador do tipo LOA como a melhor opção para implementar a arquitetura da unidade de SAD para predição intraquadro de forma que ela fosse escalável para qualidade e energia. Nesta subseção, um novo conjunto de experimentos é apresentado, desta vez, para definir os pontos de operação imprecisos dessa arquitetura. Para isso, foram consideradas oito configurações de árvores de SAD, cada uma correspondendo a um nível de imprecisão para possibilitar a escalabilidade entre dissipação de potência e qualidade. De maneira semelhante à discussão apresentada na seção 4.1, uma caracterização de potência das unidades de cálculo de SAD foi utilizada para definir uma arquitetura escalável com baixo consumo de energia. Nesta avaliação, cada uma das configurações de árvore de SAD consideradas pode processar, em paralelo, as amostras de um bloco completo de 16x16 pixels. Portanto, cada entrada da árvore de SAD é formada por 256 amostras do bloco atual e 256 amostras do bloco predito.

Como a arquitetura da árvore de SAD possui entradas de oito bits, oito pontos de operação foram considerados: um sem imprecisão e sete com imprecisões que variam de um a sete bits. Como discutido anteriormente, a imprecisão foi inserida apenas no primeiro estágio de cálculo de SAD, ou seja, nas operações de subtração e módulo, através do uso de operadores do tipo LOA. A acumulação de valores nos demais níveis da árvore SAD é realizada por operadores RCA convencionais para evitar que a propagação da imprecisão gerasse um acúmulo de erros. Cada árvore de SAD possui nove níveis, sendo que o número de operadores aritméticos por nível é 256, 128, 64, 32, 16, 8, 4, 2 e 1. Nesta avaliação, as arquiteturas das árvores de SAD foram implementadas para que fosse possível avaliar os ganhos em potência e área de cada

nível de imprecisão em relação à versão precisa. Essas arquiteturas foram descritas em linguagem VHDL. A etapa de síntese das arquiteturas considerou a mesma metodologia apresentada na seção 4.1.

Essas sete configurações imprecisas para cálculo de SAD também foram implementadas no software de referência do HEVC. Assim, o HM foi modificado para que pudessem ser avaliados os impactos na eficiência da codificação de cada nível de imprecisão. Esta avaliação também utilizou a mesma metodologia apresentada na seção 4.1.

Os resultados de síntese e da avaliação com o HM são apresentados na Tabela 4. RCA refere-se à versão precisa. Versões imprecisas são apresentadas como LOA, sendo que o número após a sigla indica o número de bits imprecisos que são usados nos operadores aritméticos. Por exemplo, LOA3 é um operador do tipo LOA com 3 bits na parte imprecisa. Em geral, quanto maior o nível de imprecisão, menor a área, menor a dissipação de energia e maior a degradação em BD-rate.

Tabela 4 – Resultados de síntese e de BD-rate para as árvores de SAD.

Operador	Potência (mW)	Área (Kgates)	Aumento em BD-rate (%)	Redução em Potência (%)	Redução em Área (%)
RCA	17,70	31,68	-	-	-
LOA1	16,73	31,33	0,27	5,48	1,10
LOA2	16,73	31,34	0,26	5,48	1,07
LOA3	15,71	29,65	0,28	11,24	6,41
LOA4	14,94	28,65	0,40	15,59	9,56
LOA5	13,83	26,71	0,68	21,86	15,69
LOA6	13,10	26,25	1,14	25,99	17,14
LOA7	11,93	24,47	1,72	32,60	22,76

Os resultados correspondentes à relação entre redução de potência e aumento de BD-rate foram utilizados para definir um conjunto de pontos de operação para a unidade SAD. A Figura 15 apresenta esta relação em um gráfico. A decisão final foi incluir LOA3, LOA5 e LOA7 como pontos de operação imprecisos da arquitetura da árvore de SAD, uma vez que as reduções em dissipação de energia são significativas e próximas a 10%, 20% e 30%, respectivamente. O aumento de BD-rate para esses três pontos de operação imprecisos foi de 0,3%, 0,7% e 1,7%, respectivamente. A versão precisa foi considerada o quarto ponto de operação. Por considerar a versão precisa, o quarto ponto de operação não possui perdas em eficiência de codificação.

Esses pontos de operação são destacados na Tabela 4 e na Figura 15. Os resultados alcançados mostraram que reduções importantes de energia podem ser

obtidas com perdas de eficiência de codificação muito baixas, considerando o objetivo da aplicação e o status do dispositivo.

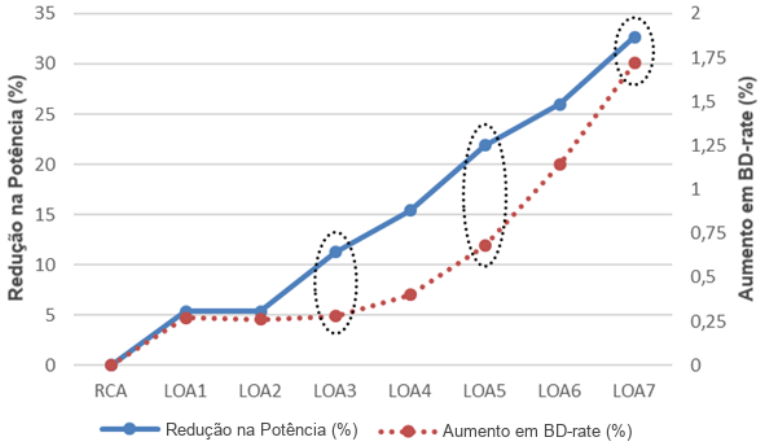


Figura 15 – Redução na dissipação de potência versus aumento em BD-rate.

4.2.3 *Arquitetura para Árvore de SAD Escalável em Energia e Qualidade*

A arquitetura proposta para a árvore de SAD usa o mesmo modelo usado na arquitetura de predição intraquadro base (CORRÊA et al., 2017). Ela foi projetada para processar 512 amostras de entrada em paralelo, sendo 256 do bloco original e 256 do bloco predito. A arquitetura é totalmente combinatória. Isso significa que o valor de SAD de um bloco predito de 16×16 pixels é calculado em um único ciclo de *clock*. A principal diferença entre a arquitetura base e a arquitetura aqui proposta está no primeiro nível das unidades de cálculo de SAD: enquanto a árvore de SAD base usa operadores RCA para calcular a etapa das diferenças, a árvore de SAD proposta usa um novo operador configurável, denominado Operador Configurável Otimizado - *Optimized Configurable Adder* (OCA), que será apresentado nos próximos parágrafos.

Uma abordagem direta e não otimizada para projetar essa arquitetura configurável para cálculo de SAD seria simplesmente instanciar as quatro arquiteturas de árvore de somadores, uma precisa (RCA) e três imprecisas (LOA3, LOA5 e LOA7). A saída dessas quatro árvores de SAD poderia ser conectada através de um multiplexador. A saída dependeria do modo de operação selecionado. No entanto, além de usar uma grande quantidade de recursos de hardware, essa solução não otimizada também tenderia a aumentar a dissipação de energia, uma vez que todos os somadores de todas as árvores de SAD iriam ser estimulados a cada nova entrada. Assim, como a

arquitetura completa para predição intraquadro utiliza 35 árvores de cálculo de SAD, a aplicação de otimizações adicionais é ainda mais importante para reduzir a dissipação de energia e o uso de área. Dessa forma, uma árvore de cálculo de SAD otimizada foi projetada, a qual explora tanto o compartilhamento de operações em comum quanto técnicas de *operand isolation*.

A Figura 16 mostra, em alto nível, o diagrama de blocos da árvore de SAD configurável e otimizada proposta neste capítulo. O elemento mais importante nessa arquitetura é o operador configurável OCA, o qual será detalhado nos próximos parágrafos. As entradas *Orig* e *Pred* (n) referem-se às amostras de oito bits do bloco original e de um dos *n* blocos candidatos. A saída *Sad (n)* disponibiliza o valor de SAD calculado para o bloco candidato *n*. Esta saída possui 16 bits. A entrada *SelBlock* tem o mesmo objetivo definido na arquitetura base da unidade SAD e a entrada *SelOp* seleciona o ponto de operação desejado.

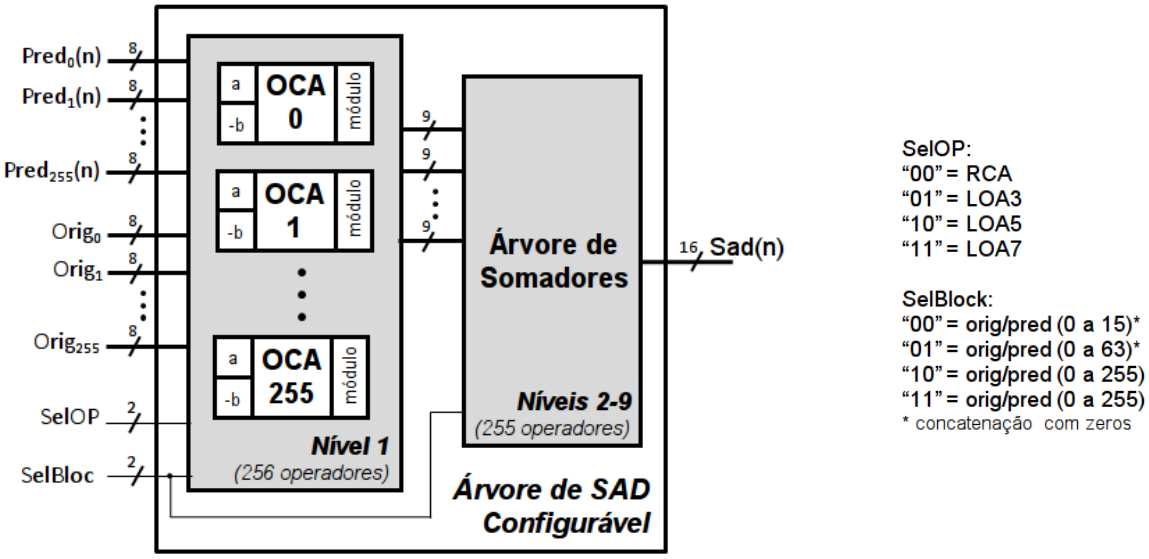


Figura 16 – Diagrama de blocos da arquitetura da árvore de SAD configurável.

O primeiro nível da árvore de SAD é implementado com 256 operadores OCA. Esses operadores implementam uma subtração seguida pelo cálculo do módulo e consideram os quatro pontos de operação definidos anteriormente. As operações de subtração e módulo foram agrupadas para reduzir o consumo de hardware. Para isso foi utilizada uma lógica combinatória com base em um operador CLA (MANO; KIME, 2001), com várias simplificações, adaptando o comportamento de um

operador LOA. O bloco Árvore de Somadores na Figura 16 é responsável pelos níveis 2 a 9 dos cálculos de SAD e é usado para acumular as diferenças absolutas. Essa árvore de somadores utiliza operadores RCA para evitar efeitos negativos da acumulação de erros.

A ideia principal que é explorada no operador OCA é a reutilização do maior número possível de bits das estruturas RCA e LOA. A Figura 17 apresenta o diagrama de blocos desse operador. Na Figura 17, as linhas pontilhadas representam a propagação de *carry*. Esta solução utiliza apenas um operador RCA de oito bits, um operador LOA de sete bits e lógica adicional para controlar a propagação de *carry* condicional que suporta os quatro pontos de operação definidos neste trabalho. Além disso, é necessária uma lógica adicional para organizar as saídas, concatenando os bits adequados das saídas dos operadores LOA e RCA para atingir cada nível de imprecisão configurável. Os pontos de operação LOA3, LOA5 e LOA7 compartilham os três bits menos significativos do operador LOA. Os pontos de operação LOA5 e LOA7 também compartilham outros dois bits do operador LOA, conforme apresentado na Figura 17. O mesmo comportamento ocorre com o operador RCA. Como exemplo, o ponto de operação LOA5 utiliza cinco bits do operador LOA e três bits do operador RCA. Os resultados parciais são concatenados para gerar os resultados do operador.

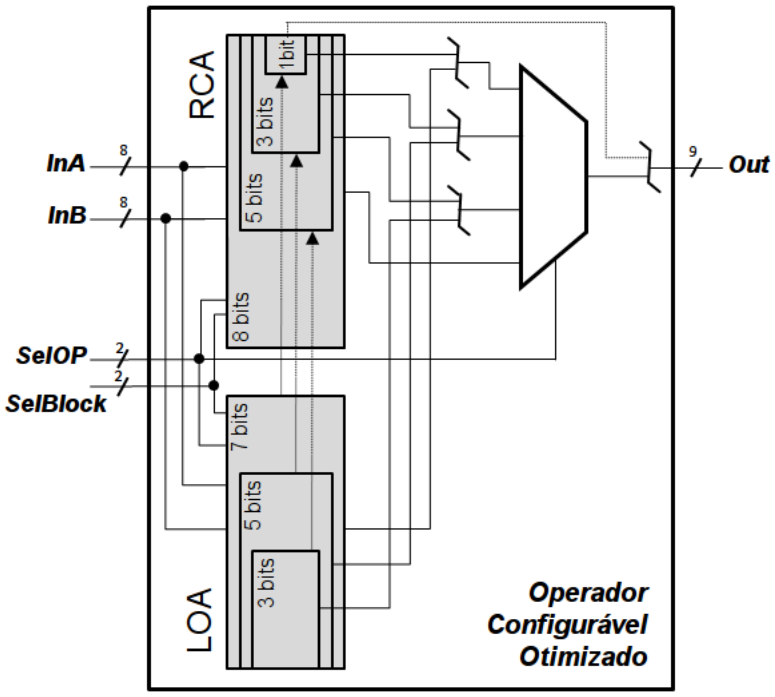


Figura 17 – Diagrama de blocos do Operador Configurável Otimizado.

Assim, uma solução não otimizada exigiria quatro somadores independentes de oito bits para cada uma das 256 operações. Isso corresponde a um RCA, um LOA3 (RCA de 5 bits mais LOA de 3 bits), um LOA5 (RCA de 3 bits mais LOA de 5 bits) e um LOA7 (RCA de 1 bit mais LOA de 7 bits). Enquanto isso, a unidade otimizada proposta economiza nove bits de soma RCA e oito bits de soma LOA. Isso corresponde a 52,9% dos bits de soma RCA e 53,3% dos bits de soma LOA. Essas economias em área e, consequentemente, em dissipação de potência são especialmente importantes quando se considera que a árvore de SAD utiliza 256 desses operadores no primeiro nível.

O uso de estruturas do tipo OCA também permite um fácil compartilhamento dos outros níveis da árvore de SAD (níveis 2 a 9 na Figura 16). Em outras palavras, a mesma estrutura de árvore de somadores é usada para todos os pontos de operação. Assim, apenas 255 operadores são usados nos níveis 2 a 9 da arquitetura da árvore de SAD otimizada, uma economia de 765 operadores. Considerando o número de bits dos somadores, esta solução usa 2.542 bits de soma ao invés de 10.168. Isso significa uma economia de 75% no número de operadores necessários à árvore de somadores.

Portanto, ao considerar toda a estrutura de cálculo de SAD, a unidade otimizada proposta requer apenas 4.590 bits RCA e 1.792 bits LOA, ao invés de 14.520 bits RCA e 3.840 bits LOA que seriam necessários sem as otimizações. Isso significa que a árvore de SAD otimizada proposta economiza 68,4% dos bits de soma RCA e 53,3% dos bits de soma LOA. Essas economias expressivas em recursos de hardware são importantes quando se considera que a unidade de SAD utiliza 35 árvores de SAD. Como consequência, essas economias de área resultam em impactos semelhantes na dissipação de potência.

Para otimizar ainda mais a dissipação de potência, também foi aplicada a técnica de isolamento de operandos – *operand isolation*. Com isso, operadores não utilizados em um determinado cálculo tiveram bloqueada a propagação dos sinais de entrada, ficando isolados. Esse isolamento é controlado pelos sinais gerados a partir de *SelBlock* e *SelOp* e é aplicado através de uma porta AND adicional inserida em cada entrada do somador.

A aplicação de isolamento de operandos considerou duas situações. A primeira é quando o ponto de operação selecionado afeta apenas o primeiro nível da árvore de SAD, a parte configurável dessa arquitetura. Nesse caso, como os operadores OCA são otimizados no nível de bit, o isolamento de operandos também é aplicado no nível

de bit. A segunda situação ocorre em todos os nove níveis da arquitetura da árvore de SAD sempre que tamanhos de blocos menores são processados (8×8 ou 4×4 *pixels*) e uma parte dos somadores não é necessária. Como exemplo, quando um bloco de 8×8 *pixels* é processado, apenas 127 saídas dos operadores da árvore de SAD são necessárias.

4.3 Resultados Síntese da Arquitetura de SAD para a Predição Intraquadro com Baixa Dissipação de Potência

A arquitetura proposta foi descrita em VHDL e sintetizada usando-se a ferramenta *Encounter RTL Compiler*, da Cadence. A síntese teve como alvo a biblioteca de células padrão Nangate 45nm (NANGATE, 2019), do mesmo modo que nos experimentos anteriores. Para garantir processamento em tempo real para a resolução de UHD 8K (7680×4320 *pixels*) a 60 quadros por segundo, a frequência de operação foi definida como 269MHz. Os recursos de hardware necessários são apresentados em número portas NAND2 equivalentes. Esse valor é obtido através da divisão da área total do circuito pela área de uma célula NAND2 (0,8 μm^2) nessa tecnologia.

As próximas seções apresentam os resultados de síntese para a árvore de SAD, base da arquitetura apresentada neste capítulo, e os resultados gerais. Além disso, é apresentada uma caracterização detalhada relacionando a redução do consumo de energia e a variação de BD-rate para diferentes resoluções de vídeo e diferentes pontos de operação imprecisos da arquitetura.

4.3.1 *Árvore de SAD Otimizada e Configurável*

A Tabela 5 apresenta os resultados de área e de dissipação de potência da arquitetura da árvore de SAD configurável quando comparada com a estrutura base original não configurável (usando apenas RCAs). Os resultados de dissipação de potência são apresentados para cada um dos pontos de operação definidos para a arquitetura otimizada.

Tabela 5 – Resultados para a arquitetura de árvore de SAD otimizada e configurável.

Critério de Avaliação	**Árvore de SAD Original**	**Árvore de SAD Otimizada e Configurável**			
		RCA	**LOA3**	**LOA5**	**LOA7**
Área (Kgates)	38,1	43,9			
Aumento em Área (%)	-	15,0			
Potência (mW)	20,3	14,9	14,1	13,4	12,8
Redução em Potência (%)	-	26,5	30,3	33,8	36,8

Os resultados destacam as significativas reduções na dissipação de potência obtidas com a arquitetura otimizada sendo comparada à árvore de SAD original. Essas reduções em dissipação de potência variam de 26,5% a 36,8% e são consequência do uso dos somadores imprecisos e também do uso da técnica de isolamento de operandos. Apesar de oferecer suporte a quatro pontos de operação e necessitar de hardware extra para implementar isolamento de operandos, a arquitetura configurável e otimizada utilizou apenas 15% a mais de área que a versão original. Mesmo com essa área extra, as reduções de dissipação de potência foram expressivas.

4.3.2 *Unidade de Cálculo de SAD Escalável em Potência e Qualidade*

Esta subseção discute os resultados experimentais obtidos após a implementação da unidade de SAD escalável em qualidade e dissipação de potência. Essa estrutura processa simultaneamente todos os modos de predição e todos os tamanhos de bloco definidos na especificação da predição intraquadro do padrão HEVC.

A Tabela 6 apresenta os resultados de síntese para a unidade de SAD escalável em qualidade e dissipação de potência. Entre os dados apresentados estão a energia consumida para processar um quadro UHD 8K e o impacto na eficiência de codificação em termos de BD-rate tanto para a arquitetura original (precisa) quanto para a arquitetura escalável proposta executando cada um de seus pontos de operação.

Tabela 6 – Resultados para a unidade de SAD escalável em qualidade e energia.

Critério de Avaliação	**Unidade de SAD Original**	**Unidade de SAD Escalável em Dissipação de Potência e Qualidade**			
		RCA	**LOA3**	**LOA5**	**LOA7**
Área (Kgates)	1.288,7	1.388,3			
Aumento de Área (%)	-	7,7			
Potência (mW)	692,3	505,3	481,2	461,3	443,1
Energia / Quadro (mJ)	11,53	8,42	8,02	7,68	7,38
Redução em Energia (%)	-	27,0	30,5	33,4	36,0
Perdas em BD-Rate (%)	-	0	0,28	0,68	1,72

Os resultados obtidos mostram que a arquitetura da unidade de SAD proposta proporciona uma redução da energia consumida entre 27% e 36% quando comparada à arquitetura original. Pode-se observar também que mesmo a solução precisa (RCA) é mais eficiente em termos de energia que a arquitetura original. Isso é resultado da aplicação da técnica de isolamento de operandos. Assim, a energia total necessária para processar um quadro UHD 8K foi reduzida de 11,53mJ para 8,42mJ, considerando-se o cálculo preciso. Ao empregar os pontos de operação aproximados,

o consumo de energia cai para 8,02mJ (LOA3), 7,68mJ (LOA5) e 7,38mJ (LOA7). Outro aspecto que pode ser observado é que a potência e a energia são proporcionais nesse caso, pois o tempo de processamento por quadro permanece constante para a mesma resolução e taxa de quadros. As economias de energia e potência são obtidas ao custo do aumento em BD-rate. Esse aumento varia de 0,28% (LOA3) a 1,72% (LOA7), como já discutido anteriormente. Em relação à área ocupada, o aumento foi de 6,8%.

4.3.3 *Escalabilidade entre Energia e Qualidade*

A Figura 18 apresenta uma caracterização detalhada relacionando a redução do consumo de energia e a variação de BD-rate para diferentes resoluções de vídeo e diferentes pontos de operação imprecisos. Os resultados médios de cada classe foram considerados para todos os vídeos recomendados nas CCT. Nos experimentos, a arquitetura proposta operou na sua frequência máxima de operação que é de 269 MHz.

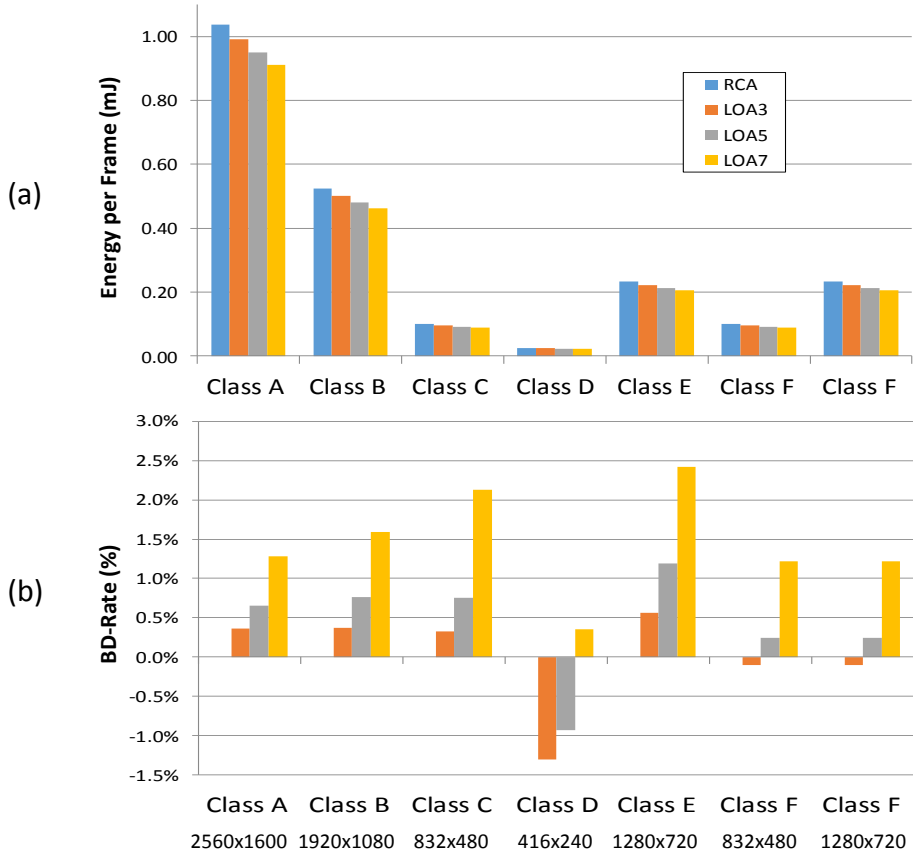


Figura 18 – Variações de energia (a) e BD-rate (b) para diferentes resoluções e diferentes pontos de operação.

Na Figura 18 (a), pode-se observar que um quadro de vídeo da Classe A (2560x1600) exige que cerca de 1,03mJ ao ser processado utilizando-se operadores RCA. Essa energia consumida pode ser reduzida para 0,91mJ (-12%) ao adotar-se cálculos imprecisos com o ponto de operação LOA7. O custo dessa redução de consumo de energia é um pequeno aumento em BD-rate (1,28%). Os pontos de operação LOA3 e LOA5 representam, respectivamente, uma redução de energia de 4% e 9% quando comparados à predição com base em operadores RCA.

Como esperado, resoluções mais baixas exigem menos energia por quadro devido à menor quantidade de dados a serem processados. No entanto, nenhuma relação clara entre a resolução de vídeo e a eficiência da codificação é observada na Figura 18 (b). Esse comportamento pode ser explicado porque o impacto da imprecisão depende do conteúdo do vídeo e não de sua resolução. Por outro lado, a relação entre os pontos de operação imprecisos é consistente em todas as resoluções de vídeo. Além disso, é possível observar que a configuração LOA3, além de não proporcionar aumento significante em BD-rate, ainda apresenta alguns ganhos em eficiência de codificação para as classes D e F. Por outro lado, a configuração LOA7 apresenta as maiores perdas, variando de 0,35% (Classe D) a 2,41% (Classe E).

Enquanto a Figura 18 demonstra a faixa de escalabilidade em termos de imprecisão e resolução de vídeo, a Figura 19 mostra o comportamento energia×qualidade ao longo do tempo, com os pontos de operação sendo modificados dinamicamente. Nesse experimento, foram codificados 50 quadros da sequência de vídeo *BQTerrace,* de acordo com as seguintes condições: quadros 0-9 usando a configuração RCA; configuração LOA3 para os quadros 10-19; LOA5 para os quadros 20-29; quadros 30-39 usando LOA7; e, finalmente, quadros 40-49 usando a configuração RCA. O valor utilizado no parâmetro de quantização foi 27. Os platôs de consumo de energia, definidos por cada ponto de operação podem ser facilmente observados. Por sua vez, a qualidade do vídeo apresenta uma variação maior devido a alterações no conteúdo do vídeo. Ainda assim, é possível observar que a degradação da qualidade de vídeo é muito pequena (PSNR reduzindo de 37,8dB para 37,74dB), observada quando o nível de imprecisão estava no máximo (especialmente para LOA 7 nos quadros 30-39). Como não há dependências entre os quadros, assim que o modo de operação muda de LOA7 para RCA é imediatamente observado um aumento na qualidade do vídeo. Isso pode ser notado nas informações para o quadro 40, na Figura 19.

Esse comportamento demonstra que a arquitetura escalável proposta pode ser facilmente ajustada por um controlador externo que poderia ser, por exemplo, um monitor de nível de bateria e é adequada para integração em um sistema completo de codificador de vídeo.

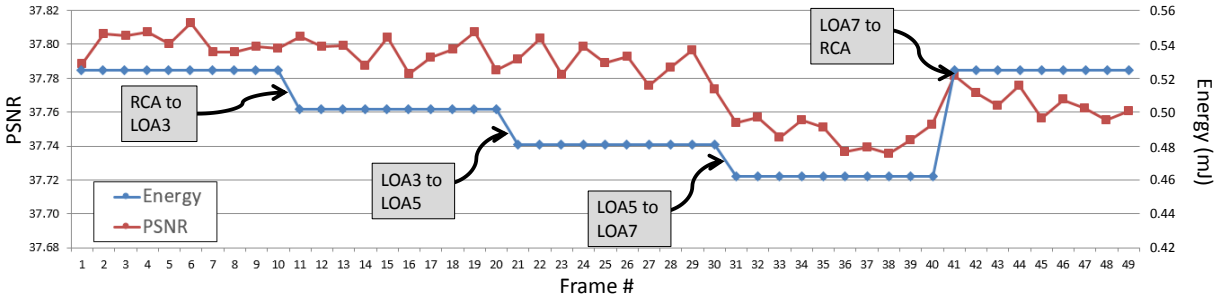


Figura 19 – Variação da energia e da qualidade de vídeo ao longo do tempo para diferentes pontos de operação.

4.4 Comparações com Outros Trabalhos com Arquiteturas para a Predição Intraquadro do HEVC

Há diversos trabalhos na literatura com soluções para a etapa de predição intraquadro do padrão HEVC, mas nenhum deles refere-se a uma arquitetura escalável em qualidade e energia. Também existem trabalhos com ênfase na decodificação do padrão HEVC, como (ZHOU et al., 2016), (CHUANG et al., 2010), (HUANG et al., 2013), (TSAI et al., 2013) e (JU et al., 2014). Porém, a decodificação não requer cálculos de SAD, o que torna impraticável realizar qualquer comparação com a arquitetura proposta. Na verdade, apenas alguns trabalhos publicados direcionados ao codificador, apresentam resultados de potência ou energia. Além disso, a diversidade de características dos projetos impede uma comparação mais justa.

Trabalhos como (FANG; CHEN; CHANG, 2016) e (LU et al., 2015) usam SATD como métrica de distorção em suas arquiteturas, o que torna inviável uma comparação justa com a arquitetura apresentada. Apesar de focar na etapa de predição intraquadro, trabalhos como (LIU et al., 2013), (KHAN et al., 2013) e (LI; SHI; WU, 2011) apresentam apenas os detalhes sobre o hardware projetado, mas não discutem a distorção introduzida.

Por sua vez, os trabalhos (LIU et al., 2013a), (ZHOU; DING; YU, 2013) e (PALOMINO et al., 2012) usam o SAD como critério de distorção e algumas comparações são possíveis, apesar de terem como alvo diferentes tecnologias de células padrão. Porém, é importante enfatizar que nenhum desses trabalhos apresenta resultados de potência ou energia, nem apresentam resultados independentes para a unidade de SAD.

O hardware apresentado em (LIU et al., 2013a) pode processar vídeos em Full HD a 30 quadros por segundo. A frequência de operação necessária para atingir essa taxa de processamento é de 600 MHz. Essa frequência é mais do que duas vezes maior que a frequência de operação necessária pela estrutura proposta nesse trabalho para processar vídeos UHD 8K. Por outro lado, o trabalho em (LIU et al., 2013a) utiliza apenas 77 Kgates, o que é muito menos do que a área requerida pela arquitetura proposta. Com relação às simplificações aplicadas, estas causam uma pequena queda de 0,13% em BD-Rate.

A arquitetura proposta em (ZHOU; DING; YU, 2013) possui um nível de paralelismo de 64 amostras por ciclo e usa uma ordem de processamento alternativa para reduzir os acessos à memória. Essa arquitetura pode processar vídeos em Full HD a 60 quadros por segundo, rodando a 400 MHz e utilizando 324 Kgates. Apesar de usar menos hardware que a estrutura apresentada nesse trabalho, a taxa de processamento alcançada por (ZHOU; DING; YU, 2013) é muito menor, exigindo uma frequência mais alta para suportar uma resolução mais baixa.

O trabalho em (PALOMINO et al., 2012) possui um nível de paralelismo de oito amostras por ciclo e é executado a 500 MHz, atingindo uma taxa de processamento capaz de processar vídeos HD 720p a 30 quadros por segundo. Este trabalho requer uma frequência de operação mais alta do que a da arquitetura proposta nesse trabalho, processando resoluções mais baixas.

Em (VANNE et al., 2006) é apresentada a implementação de uma arquitetura para cálculo de SAD com alta taxa de processamento. Os autores utilizaram árvores de somadores *Carry-Save Adder* (CSA) para obter os valores de SAD. Quando comparados a uma arquitetura com CLA, os resultados de síntese para uma tecnologia de 180nm demonstram redução de 12,5% no atraso e redução de 9% na área ocupada. Embora seja possível supor que (VANNE et al., 2006) obtém alguma redução

na potência dissipada, nenhuma análise foi disponibilizada. Por outro lado, a solução apresentada neste capítulo atinge até 36,8% de redução na dissipação de potência.

4.5 Considerações Finais do Capítulo

Este capítulo apresentou uma arquitetura para cálculo de SAD escalável em qualidade e energia. O alvo desta arquitetura é a predição intraquadro do padrão HEVC e a codificação de vídeos UHD 8K. A escalabilidade em qualidade e energia foi alcançada usando-se computação aproximada, através da implementação de operadores imprecisos em quatro pontos de operação distintos.

Seis operadores imprecisos foram avaliados. Foram considerados suas eficiências de codificação e seus resultados de hardware. A avaliação indicou o operador LOA como a melhor estrutura para ser usada na arquitetura projetada. A definição dos pontos de operação foi feita avaliando várias implementações independentes de arquiteturas de árvores de SAD. Por meio dessa avaliação foram definidos quatro pontos de operação: RCA, LOA3, LOA5 e LOA7.

A unidade de SAD projetada utilizou uma árvore de SAD configurável com base em uma estrutura de somador otimizada e configurável, e que suporta os quatro pontos de operação definidos anteriormente. Tanto o somador otimizado quanto a árvore de SAD como um todo utilizaram também a técnica de isolamento de operandos para reduzir a dissipação de potência. A unidade de SAD proposta usa 35 instâncias paralelas e configuráveis para alcançar a taxa de processamento desejada.

A unidade de cálculo de SAD escalável em qualidade e energia é capaz de realizar a predição intraquadro do HEVC para vídeos UHD 8K a 60 quadros por segundo, rodando a 269MHz e considerando quatro pontos de operação. Quando comparada com a arquitetura anterior (CORRÊA et al., 2017), a solução desenvolvida reduz a energia necessária para processar um quadro UHD 8K de 11,53mJ para 8,42mJ (cerca de 27%). Essa redução de energia é obtida ao custo de uma leve perda em eficiência de codificação de 0,28%, 0,68% e 1,72% para os três pontos de operação imprecisos. Experimentos adicionais demonstraram que a arquitetura proposta permite escalabilidade de energia e qualidade para diferentes resoluções e taxas de quadros e pode ser controlada externamente, sendo adequada para integração em um codificador de vídeo escalável.

Por fim, deve-se destacar que a metodologia aplicada e o conjunto de otimizações propostas para a unidade de SAD são aplicáveis em outras etapas de um codificador de vídeo, aumentando o potencial de escalabilidade. Além disso, a solução desenvolvida também pode ser usada em outros algoritmos de processamento de imagem, de processamento de vídeo e de visão computacional que usam o SAD como critério de similaridade. Expandindo ainda mais a utilização dos operadores de OCA, estes podem ser empregados para otimizar o cálculo de outros critérios de similaridade como SATD e SSE.

5 ARQUITETURAS PARA A PREDIÇÃO INTERQUADROS COM BAIXA DISSIPAÇÃO DE POTÊNCIA E USO DE COMPUTAÇÃO APROXIMADA

Este capítulo apresenta duas soluções arquiteturais para a etapa de predição interquadros que exploram o uso de computação aproximada. O uso de computação aproximada nessas arquiteturas teve como objetivo a redução na dissipação de potência.

A Seção 5.1 apresenta uma exploração do uso de operadores imprecisos em uma arquitetura para estimação de movimento de blocos de tamanhos variados (VBSME) desenvolvida em um trabalho anterior (PORTO; AGOSTINI; BAMPI, 2009). Esta arquitetura teve seus operadores aritméticos originais, do tipo RCA, substituídos por operadores imprecisos do tipo LOA.

A segunda solução a ser apresentada neste capítulo trata-se de uma solução completa para a estimação de movimento, por implementar tanto a parte inteira (IME – integer motion estimation) quanto a parte fracionária (FME – fractional motion estimation). Assim como no trabalho anterior, essa arquitetura tem como base o uso de operadores imprecisos do tipo LOA. Outras aproximações foram realizadas em de nível de algoritmo, reduzindo-se a quantidade de tamanhos de bloco suportados e através de modificações no algoritmo TZS. Assim, além de ser uma solução completa para a ME, essa arquitetura apresenta alta taxa de processamento e baixa dissipação de potência. Esta solução tomou por base dois trabalhos anteriores: (PORTO et al, 2019) e (PERLEBERG et al., 2018).

5.1 Exploração Arquitetural em uma VBSME para Redução na Dissipação de Potência

Esta seção apresenta uma exploração arquitetural em uma arquitetura para estimação de movimento de blocos de tamanhos variados (VBSME) desenvolvida em um trabalho anterior (PORTO; AGOSTINI; BAMPI, 2009).

Nesta exploração arquitetural, operadores imprecisos do tipo LOA substituíram os somadores *Ripple Carry Adder* (RCA) utilizados originalmente. Os operadores imprecisos foram inseridos nas unidades de cálculo da SAD a fim de obter-se uma redução na dissipação de potência em troca de um impacto mínimo na eficiência de codificação. Para isso, a arquitetura da VBSME foi replicada em três versões, cada uma utilizando um nível de imprecisão diferente nos operadores LOA. Assim, foram desenvolvidas versões da VBSME que utilizam operadores LOA de oito bits com três, quatro e cinco bits de imprecisão.

A arquitetura base para essa investigação (PORTO; AGOSTINI; BAMPI, 2009) é capaz de processar vídeos de alta definição em tempo real. Embora essa versão da VBSME projetada anteriormente visasse uma implementação para o padrão H.264/AVC, a técnica proposta também funciona para o padrão HEVC. Isso se deve à estratégia de reutilização de valores SAD de blocos menores para calcular os valores de SAD de blocos maiores. Usando essa arquitetura como referência foram projetadas três novas versões, todas utilizando operadores do tipo LOA. As três versões referem-se a operadores LOA com 3, 4 e 5 bits na parte imprecisa. Considerando-se que os operadores utilizados neste trabalho possuem largura de 8 bits, foi possível avaliar a imprecisão para metade da largura do operador (4 bits) e em ambas as direções de imprecisão, aumentando (5 bits) ou diminuindo (3 bits). Neste trabalho, essas novas versões da arquitetura da VBSME foram denominadas VBSME com uso eficiente de energia (E-VBSME).

A arquitetura da VBSME opera sobre blocos de 16×16 pixels. Para isso, a VBSME utiliza os valores de SAD de blocos menores (4×4 pixels, por exemplo) para calcular os valores de SAD de blocos maiores. Dessa forma, a arquitetura da VBSME utiliza o módulo para cálculo de ME de blocos 4×4 pixels como estrutura básica e somadores para agrupar os valores de SAD de blocos 4×4 pixels que formam os valores de SAD de blocos maiores. A estratégia de exploração de eficiência energética considera a substituição dos operadores RCA da arquitetura original pelas três opções de

operadores LOA discutidas anteriormente. É importante ressaltar que essa substituição é realizada apenas na primeira etapa de cálculo de SAD, a subtração, do mesmo modo que na arquitetura definida no capítulo anterior. O objetivo é evitar um acúmulo excessivo de erros que poderia ser causado com a utilização de operadores LOA nos estágios posteriores. Ainda sobre a arquitetura da VBSME, em uma área de busca de 16×16 pixels existem 13 blocos candidatos em uma linha e 13 blocos candidatos em uma coluna para cada bloco 4×4 pixels. Assim, 169 blocos candidatos são comparados com o bloco atual, um de cada vez, para encontrar a melhor correspondência. Dessa forma, em cada nível da arquitetura existem 169 valores de SAD, cada um correspondendo a um bloco candidato.

A Figura 20 apresenta o diagrama de blocos da arquitetura da ME para blocos de 4×4 pixels. Este é o módulo mais importante da estrutura proposta. Outros módulos da arquitetura da VBSME são explicados em detalhes em (PORTO; AGOSTINI; BAMPI, 2009). Alguns sinais foram omitidos na Figura 20 para permitir uma melhor visualização, principalmente os sinais de controle. Os principais elementos da ME 4×4 são as linhas de SAD com seus núcleos de processamento (NP), os comparadores, as memórias e o gerenciador de memória.

Na Figura 20, cada NP calcula o valor de SAD relativo a uma linha do bloco atual e uma linha do bloco candidato. Cada linha de SAD é formada por quatro NPs. Os três primeiros são responsáveis pelo processamento de quatro blocos candidatos. O último é responsável pelo processamento de apenas um bloco candidato. Dessa forma, uma linha de SAD processa os 13 blocos candidatos existentes em uma linha. A primeira linha de SAD calcula os valores de SAD para os 13 blocos candidatos que começam na primeira linha da área de busca, comparando o bloco atual a esses blocos candidatos. Por sua vez, a segunda linha de SAD faz o mesmo processo para os 13 blocos candidatos existentes na segunda linha da área de busca, e assim por diante. Esse processo é realizado até que, na décima terceira linha de cálculo de SAD, o bloco atual seja comparado com os 13 últimos blocos candidatos. Dessa forma, o bloco atual é comparado aos 169 blocos candidatos para encontrar a melhor correspondência. A arquitetura para cálculo de SAD foi projetada hierarquicamente utilizando-se 13 linhas de SAD, cada uma com 4 NPs. Isso significa que a ME 4×4 contém 52 NPs.

A Figura 21 mostra a arquitetura de um NP. Cada NP recebe quatro amostras do bloco atual (A_0 a A_3) e quatro amostras do bloco candidato (C_0 a C_3). O SAD calculado por um NP é um valor parcial e representa o valor SAD de uma linha do bloco candidato. Assim, esse valor precisa ser adicionado aos valores de SAD de outras linhas para gerar o SAD total de um bloco candidato. Cada linha de SAD agrupa quatro NPs. Dessa forma, gera o SAD total de cada bloco candidato relativo a esta linha.

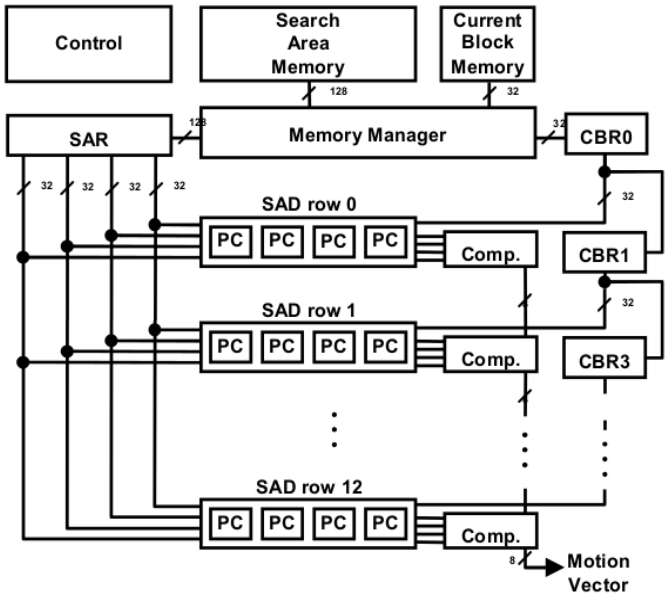


Figura 20 – Diagrama de blocos da arquitetura da ME para blocos de 4×4 pixels. Adaptado de (PORTO; AGOSTINI; BAMPI, 2009).

É importante destacar que existem 364 operadores aritméticos considerando todas as 13 linhas de SAD. Entre esses operadores, 208 são subtratores, localizados no primeiro estágio dos NPs. Esses subtratores são destacados na Figura 21 e foram o alvo da exploração arquitetural apresentada nesta seção, tendo sido substituídos por operadores imprecisos do tipo LOA. Os operadores dos próximos estágios de *pipeline* dos NPs não foram substituídos, para evitar erros acumulados nos próximos estágios.

Considerando o esquema de um operador LOA já discutido, quanto maior for a quantidade de bits imprecisos, maior será a redução de área, atraso e potência. Por outro lado, menor é a precisão, obviamente. Nesse aspecto, o operador LOA apresenta os melhores resultados de dissipação de potência e de área, enquanto possui os maiores erros de aproximação entre os operadores aproximados considerados (DUTT; NANDI; TRIVEDI, 2016). No entanto, esses erros não são um problema neste

trabalho, porque o operador LOA é aplicado a operandos de largura pequena (8 bits) e com uma largura de bit ainda menor na parte imprecisa.

Ao final desta exploração de espaço de projeto foram obtidas três novas versões da E-VBSME, levando em conta os três níveis de imprecisão mencionados anteriormente. Após, foi realizada uma avaliação completa dos impactos na eficiência da codificação causados por essa estratégia de projeto. Isso foi realizado por meio de uma extensa avaliação em software.

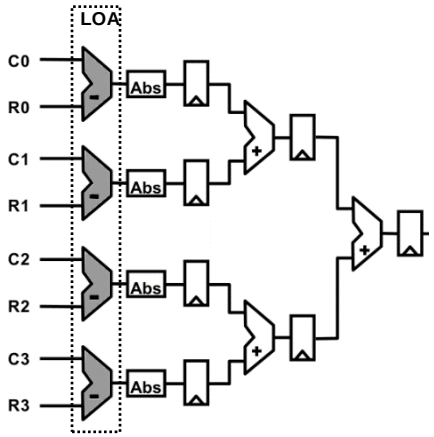


Figura 21 – Arquitetura de um Núcleo de Processamento (NP). Adaptado de (PORTO; AGOSTINI; BAMPI, 2009).

5.1.1 *Avaliação do Impacto da Computação Aproximada na Eficiência de Codificação*

De forma semelhante ao que foi apresentado na seção 4.1, neste trabalho foi realizada uma avaliação do impacto na eficiência de codificação através do uso de operadores imprecisos.

Nesse sentido, para avaliar o impacto do uso de imprecisão na eficiência de codificação, os operadores aproximados considerados foram descritos em C++ para substituir parte do código-fonte dentro do software de referência do padrão HEVC. Nessa avaliação, foram testadas e comparadas três configurações com diferentes níveis de imprecisão (de 3, 4 e 5 bits), bem como a versão não modificada, sem imprecisão (versão original do HM). Todas essas configurações foram definidas para garantir que o software executado tenha o mesmo comportamento que a arquitetura de hardware da E-VBSME apresentada anteriormente. Isso permite uma avaliação justa e precisa dos impactos da imprecisão introduzida. Para esse fim, os operadores LOA foram inseridos no HM no primeiro estágio da operação SAD, conforme definido

anteriormente. Esta primeira operação corresponde à subtração necessária para gerar as diferenças absolutas dos valores de SAD.

Os resultados dessa avaliação foram obtidos através da codificação das vinte sequências de vídeo de teste recomendadas nas CTC, as condições comuns de teste (BOSSEN, 2013). Essas sequências de teste são classificadas em cinco classes: B (1080p), C (WVGA), D (WQVGA), E (720p) e F (*screen content*). A configuração utilizada neste experimento foi *Low Delay P Main*. Foram considerados os quatro valores de QP (22, 27, 32 e 37) também recomendados nas CTC. A Tabela 7 resume os resultados dessa avaliação. Os valores de BD-rate apresentados na Tabela 7 foram calculados para os quatro diferentes QP. De acordo com os valores obtidos, o impacto varia de acordo com as características de cada classe de vídeo, mas o efeito pode ser considerado insignificante.

Tabela 7 – Degradação da qualidade, em BD-rate (%), para os três níveis de imprecisão considerados no operador LOA.

Classe	**Nível de Imprecisão**								
	3 Bits			**4 Bits**			**5 Bits**		
	Y	**U**	**V**	**Y**	**U**	**V**	**Y**	**U**	**V**
B (1080p)	0,7	0,4	0,2	1,3	0,8	0,8	2,5	1,5	1,4
C (WVGA)	0,6	0,3	0,3	1,3	0,9	0,9	2,5	1,8	1,8
D (WQVGA)	0,7	0,6	0,3	1,5	1,3	0,8	3,0	2,2	2,2
E (720p)	0,4	0,0	0,0	1,2	0,4	0,6	2,3	1,2	1,3
F (SC)	0,8	1,1	1,1	0,5	0,6	0,8	2,1	2,1	2,6
Média	**0,6**	**0,5**	**0,4**	**1,2**	**0,8**	**0,8**	**2,5**	**1,8**	**1,9**

Como uma alternativa à Tabela 7, a Figura 22 apresenta os mesmos valores de degradação da qualidade, mas, dessa vez, na forma de um gráfico. Na Figura 22 é possível ver um comportamento já esperado: quanto maior o nível de imprecisão, maior a degradação na eficiência de codificação. Considerando a configuração com imprecisão de 3 bits, os resultados mostram, em média, um aumento de apenas 0,6% em BD-rate para luminância e de 0,45% para a crominância. Ao compararem-se os resultados obtidos para imprecisão de 4 bits com os resultados para imprecisão de 3 bits, observa-se que o impacto em BD-rate é de apenas aumenta 0,6% para luminância e 0,35% para componentes de croma. No caso de imprecisão de 5 bits, as perdas foram mais representativas. Ainda assim, a magnitude dessas perdas foi de apenas 2,5% para luminância e 1,85% para crominância.

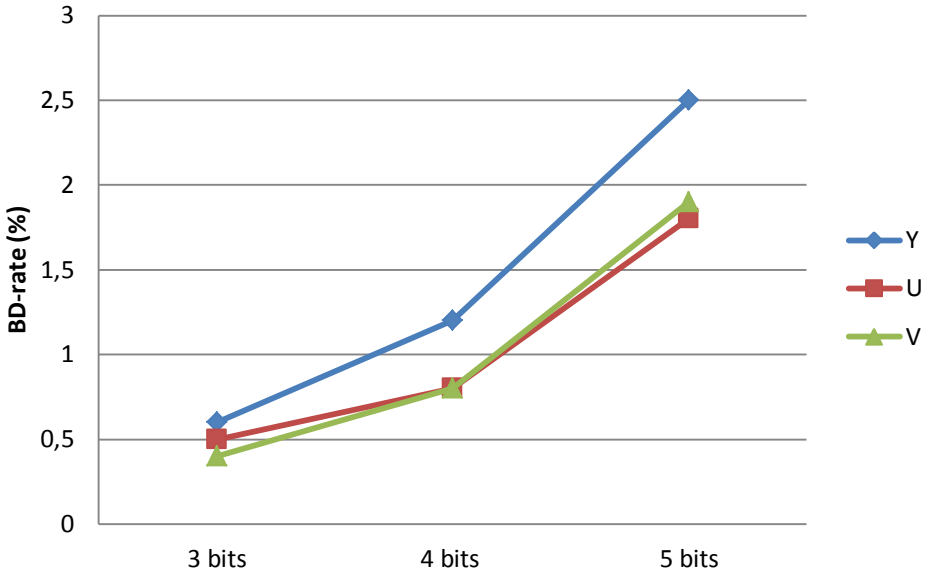


Figura 22 – Degradação da qualidade, em BD-rate (%), para os três níveis de imprecisão considerados no operador LOA.

Ao final pode-se concluir que as perdas em eficiência de codificação usando-se operadores LOA dependem do nível de imprecisão introduzido por esse tipo de operadores, mas são aceitáveis para todos os casos analisados. Os resultados de síntese, incluindo a análise de potência, são apresentados na próxima subseção.

5.1.2 *Implementação em Hardware e Caracterização*

A arquitetura da VBSME foi descrita hierarquicamente em VHDL em quatro versões diferentes: a versão original e as três versões usando operadores LOA. As arquiteturas foram sintetizadas para a biblioteca de células padrão 45nm@1.1V, da Nangate. As sínteses foram realizadas utilizando-se a ferramenta Encounter RTL Compiler, da Cadence.

As sínteses das três versões do E-VBSME tiveram como foco frequências suficientes para que a arquiteturas possam processar vídeos, em tempo real, com as resoluções HD720p (1280×720 pixels) e HD1080p (1920×1080 pixels) a 30 e 60 quadros por segundo.

Primeiramente, foi avaliado apenas o NP apresentado na Figura 21. O NP contém os cálculos de SAD, o que o torna o módulo mais importante da E-VBSME. A Tabela 8 apresenta a avaliação das quatro versões de NP. Cada versão refere-se aos operadores utilizados na primeira etapa do cálculo de SAD. A versão RCA usou

operadores do tipo *Ripple-Carry*. As versões denominadas LOA referem-se ao uso de operadores *Lower-Part-OR* e o número após a abreviação indica a largura de bits da parte imprecisa (por exemplo, LOA3 refere-se a um operador LOA de 8 bits com 3 bits de imprecisão). As frequências de operação usadas para obter os resultados de potência também são apresentadas na Tabela 8. Essas frequências foram definidas para permitir que o hardware da E-VBSME opere em tempo real com diferentes resoluções e diferentes taxas de quadros. A arquitetura da VSBME possui 53 instâncias de NP. Um desses NPs foi selecionado para avaliação e os resultados são apresentados na Tabela II.

Tabela 8 – Resultados de potência para as várias configurações de NP.

Resolução	**Frequência (MHz)**	**Potência (mW)**			
		RCA	**LOA3**	**LOA4**	**LOA5**
HD1080p@60fps	497,66	0,307	0,271	0,256	0,243
HD1080p@30fps	248,83	0,185	0,167	0,156	0,147
HD720p@60fps	221,18	0,172	0,148	0,142	0,134
HD720p@30fps	110,59	0,102	0,089	0,084	0,081

Como esperado, a dissipação de potência diminui com o aumento da imprecisão. Na Tabela 8, os maiores valores nos resultados de potência foram obtidos para o operador RCA e os menores para o operador LOA na versão com cinco bits imprecisos (LOA5). Dessa forma, os ganhos em potência alcançados com operadores LOA variam de 9,7% a 22,1% quando comparados à versão com operadores RCA. Isso configura 16,6% de ganho em potência em média. Esses são ganhos expressivos se considerarmos a pequena diminuição da eficiência do codificador, apresentada na subseção anterior, que é de 3% para o pior caso.

A Tabela 9 apresenta os resultados de área para as quatro versões de NP. Ao analisar esses resultados, pode-se concluir que o uso de operadores LOA não teve um impacto significativo em termos de uso de recursos de hardware.

Tabela 9 – Resultados de área para a arquitetura do NP.

	RCA	**LOA3**	**LOA4**	**LOA5**
Área (μm^2)	1.525	1.495	1.480	1.448
Número de Portas (Kgates)	273	288	292	289

Considerando a área total do circuito, houve uma redução entre 2% e 5% com o uso de operadores LOA. Ao considerar a contagem de portas, o uso de operadores LOA aumenta esse número de 5,5% a 7%, quando comparado a número de portas

utilizados com o uso de operadores RCA. Esses resultados podem ser explicados pelas escolhas realizadas na ferramenta de síntese. A mesma pode escolher portas complexas, especializadas em operações de adição com *ripple carry*, que estão disponíveis na biblioteca de células padrão utilizada. Então, através de células maiores a versão RCA usará uma área maior, mesmo usando menos portas do que as versões LOA.

A segunda avaliação considerou a arquitetura completa da VSBME, onde os NPs estão inseridos. Essa avaliação considerou o mesmo cenário descrito anteriormente, visando às mesmas frequências de operação e utilizando os mesmos estímulos. Os resultados de potência são apresentados na Tabela 10. Como a arquitetura da VSBME possui outros módulos que não usam operadores LOA, as reduções em dissipação de potência são um pouco menores do que os apresentados nos resultados para os NPs. Ainda assim, esses ganhos são importantes e variam de 0,5mW (LOA3 a 110,59MHz) a 2,46mW (LOA5 a 497,66MHz) quando comparados aos da versão com operadores RCA.

Tabela 10 – Resultados de potência para as arquiteturas VBSME.

Resolução	**Frequência (MHz)**	**Potência (mW)**			
		RCA	**LOA3**	**LOA4**	**LOA5**
HD1080p@60fps	497,66	21,40	19,66	18,94	18,43
HD1080p@30fps	248,83	12,84	11,81	11,40	11,11
HD720p@60fps	221,18	11,72	10,83	10,52	10,25
HD720p@30fps	110,59	7,32	6,81	6,64	6,51

Analisando os resultados da Tabela 10 é possível obter a porcentagem de economia de potência dissipada fornecida pelos operadores de LOA. A configuração com LOA5 operando a 497,66 MHz atingiu o maior percentual de economia de potência, 11,5%. Como esperado, a economia em potência aumenta à medida que o nível de imprecisão aumenta. Resumindo, as economias de potência dissipada obtidas variam entre 7% (LOA3 a 110,59 MHz) e 11,5% (LOA5 a 497,66 MHz). Esses ganhos são significativos quando comparados à diminuição da eficiência da codificação, que é de 3% no pior caso.

Os resultados de síntese da arquitetura da VSBME, usando operadores LOA, mostraram também pequenas variações em termos de uso de recursos de hardware. A Tabela 11 mostra os resultados de área para as diferentes versões da E-VBSME. Como é mostrado na Tabela 11, os resultados da implementação da VSBME seguem o

mesmo comportamento dos resultados para os NPs. A área total diminuiu conforme a imprecisão aumenta. Os ganhos estão entre 0,1% e 1,5% quando comparadas as versões com LOA e a versão RCA. O comportamento oposto foi encontrado nos resultados da contagem de portas, com perdas entre 3,6% e 4,3%. Novamente, usando portas complexas especializadas no somador, a ferramenta de síntese utiliza portas com tamanhos maiores, levando a uma área maior mesmo com uma contagem de portas menor do que nas versões com operadores LOA.

Uma maneira mais justa de comparar as versões da E-VBSME é analisar a relação entre a economia de potência obtida e o BD-Rate resultante, com o objetivo de avaliar a eficiência das soluções propostas. Dessa forma, é possível medir a quantidade de eficiência de codificação que é perdida para permitir a redução alcançada na dissipação de potência. Esses resultados são apresentados na Tabela 12. Quanto maior o valor de eficiência, melhor é o resultado. Ao considerar essa relação, os melhores resultados são os da configuração LOA3, pois apresenta os maiores valores de eficiência para todas as frequências de operação avaliadas.

Tabela 11 – Resultados de área para a arquitetura da VBSME.

	RCA	**LOA3**	**LOA4**	**LOA5**
Área (μm^2)	179.570	179.400	178.613	176.964
Número de Portas (Kgates)	29.838	30.930	31.138	30.982

Tabela 12 – Eficiência (economia de potência / BD-rate).

Resolução	**LOA3**	**LOA4**	**LOA5**
HD1080p@60fps	2,5	1,9	1,2
HD1080p@30fps	1,5	1,1	0,7
HD720p@60fps	2,2	1,0	0,6
HD720p@30fps	1,3	0,6	0,4

5.1.3 *Comparação com Trabalhos Relacionados*

Várias arquiteturas de ME foram publicadas na literatura, como (LI; TANG, 2010), (CAO et al., 2008), (SINANGIL et al., 2013), (JOU; CHANG; CHANG, 2015) e (NALLURI; ALVES; NAVARRO, 2010). Infelizmente é difícil fazer uma comparação justa da arquitetura apresentada neste trabalho com outras arquiteturas. Isso porque que os trabalhos publicados foram desenvolvidos visando diferentes padrões de codificação, sintetizados em diferentes tecnologias, focados em diferentes resoluções e implementados com diferentes configurações (área de busca, tamanhos de bloco, suporte à predição fracionária, entre outros).

A Tabela 13 apresenta um resumo de algumas dessas soluções publicadas, identificando os principais recursos desses trabalhos. Os itens comparados são: padrão de vídeo; resolução, ferramentas e tamanhos de bloco suportados; área de busca; algoritmo de busca; tecnologia CMOS utilizada na síntese; frequência de operação; número de portas utilizadas; taxa de processamento alcançada e dissipação de potência. A resolução é apresentada relacionada à taxa de quadros.

Mesmo com importantes diferenças estruturais é possível concluir que as abordagens propostas com operadores LOA apresentam os melhores resultados de potência entre todas as soluções apresentadas na Tabela 13, mesmo quando executadas com a maior frequência de operação. Infelizmente, apenas alguns dos trabalhos publicados recentemente apresentam resultados de potência. Quando comparadas com dois desses trabalhos, as versões da arquitetura com operadores LOA atingiram, no pior caso (LOA3), uma dissipação de potência 5,1 vezes menor para uma taxa de processamento 1,8 vezes menor quando comparado com (LI; TANG, 2010) e uma potência 21,5 vezes menor para uma taxa de processamento 3,3 vezes mais baixa quando comparada com (CAO et al., 2008).

Tabela 13 – Comparações com trabalhos relacionados.

	TR1*	**TR2***	**TR3***	**TR4***	**TR5***	**LOA3**	**LOA4**	**LOA5**
Padrão	H.264	H.264	HEVC	HEVC	H.264	H.264; HEVC		
Algoritmo	LBA; PDA	FS	TZS	PEPZS	FS	FS		
Área de Busca	16×16	33×33	64×64	64×64	16×16	16×16		
Tamanhos de Bloco	4×4; 4×8; 8×4; 8×8; 8×16; 16×8; 16×16	4×4; 4×8; 8×4; 8×8; 8×16; 16×8; 16×16	16×16; 32×32; 64×64	4×8; 8×4; 8×8; 8×16; 16×8; 16×16; 32×32; 64×64	4×4; 4×8; 8×4; 8×8; 8×16; 16×8; 16×16	4×4; 4×8; 8×4; 8×8; 8×16; 16×8; 16×16		
Suporte	IME	IME	IME / FME	IME / FME	IME	IME		
Resolução	1280×720 @25fps	1280×720 @45fps	3840×2160 @30fps	4096×2048 @60fps	1920×1080 @60fps	1920×1080 @60fps		
Tecnologia	0.18 um	0.18 um	65 nm	90 nm	45 nm	45 nm		
Frequência (MHz)	150	180	200	270	497	497		
Portas (Kgates)	350	160	1.830	779	30	31		
Taxa de Process. (Mpixels/s)	230,4	414,7	248,8	503,3	124,4	124,4		
Potência (mW)	101,5	423	-	-	21,40	19,66	18,94	18,43

*TR1: (LI; TANG, 2010);
TR2: (CAO et al., 2008);
TR3: (SINANGIL et al., 2013);
TR4: (JOU; CHANG; CHANG, 2015);
TR5: (PORTO; AGOSTINI; BAMPI, 2009).

5.1.4 *Considerações Finais sobre as Arquiteturas E-VBSME*

Foram apresentadas três versões de uma arquitetura para estimação de movimento de tamanhos de bloco variáveis com uso eficiente de energia (E-VBSME) através do uso de operadores aproximados. A fim de reduzir a dissipação de potência foi proposto o uso de somadores LOA para realizar os cálculos de SAD. Apenas um número reduzido de operadores foi substituído. A intenção foi restringir as perdas em eficiência de codificação. A avaliação usando o software de referência do HEVC mostrou que essa abordagem apresenta impactos pouco significantes na eficiência da codificação.

Foram avaliados três níveis de imprecisão, com o BD-Rate aumentando entre 0,6% e 2,5% na média dos vídeos analisados. As três versões da arquitetura da E-VBSME foram sintetizadas para a tecnologia de células padrão Nangate 45nm. Foram realizadas diversas comparações entre a versão original (usando apenas operadores RCA) e as três versões com operadores do tipo LOA. Os resultados de síntese indicam reduções de 9,7% a 22,1% na dissipação de potência, considerando apenas os núcleos de processamento, onde são feitos os cálculos de SAD. A arquitetura global da E-VSBME alcançou de 7% a 11,5% de redução na potência dissipada quando comparada com a versão original da arquitetura.

A comparação com trabalhos relacionados mostrou resultados bastante competitivos. Quando comparada com os trabalhos mais relevantes do estado da arte, a arquitetura de E-VBSME alcançou os melhores resultados de potência e área e a melhor relação entre potência e taxa de processamento. Esses resultados mostraram que o uso de imprecisão em aplicações para codificação de vídeo é uma alternativa quando há restrições de dissipação de potência.

5.2 Arquitetura Rápida para Estimação de Movimento de Vídeos UHD 4K em Tempo Real com Baixa Dissipação de Potência

Esta seção apresenta uma segunda arquitetura para estimação de movimento com baixa dissipação de potência. Essa arquitetura, assim como a apresentada anteriormente, tem como base o uso de operadores imprecisos do tipo LOA (MAHDIANI et al., 2010). Além disso, essa arquitetura faz uso de aproximações também em de nível de algoritmo com o propósito de reduzir a quantidade de tamanhos de bloco suportados. A arquitetura projetada é capaz de processar vídeos UHD 4K, em tempo real, a 30 quadros por segundo. Para avaliar os impactos em

termos de eficiência de codificação após o uso de técnicas de computação aproximada foram realizados experimentos com o software de referência do codificador HEVC. A arquitetura é totalmente compatível com o padrão HEVC, podendo ser facilmente adaptada para também ser compatível com outros codificadores de vídeo atuais como AV1 (AOM, 2019), AVS2 (HE et al., 2014) e VVC (MPEG, 2019). Esta solução tomou por base dois trabalhos anteriores do nosso grupo de pesquisa: (PORTO et al, 2019) e (PERLEBERG et al., 2018).

5.2.1 *Avaliação do Impacto da Computação Aproximada na Eficiência de Codificação*

Como mencionado anteriormente, várias modificações podem ser feitas nas ferramentas de codificação para reduzir sua complexidade computacional e sua dissipação de potência. Essas modificações são obrigatórias quando os alvos de um projeto são dispositivos alimentados por bateria. No entanto, qualquer modificação nas ferramentas de codificação gera impactos na eficiência da codificação. Dessa forma, todo o conjunto de modificações proposto neste trabalho deve ser cuidadosamente avaliado. Esta subseção apresenta uma avaliação do impacto da computação aproximada na eficiência de codificação da arquitetura. Primeiramente, o impacto da restrição no número de tamanhos de bloco suportados pela ME foi avaliado. Em seguida, foram realizadas avaliações para definir o operador aproximado a ser aplicado no módulo de cálculo de SAD.

As avaliações foram realizadas do software de referência do padrão HEVC. Além disso, novamente foram respeitadas as condições comuns de teste. Nelas é recomendado um conjunto de 24 sequências de vídeo para testes onde cada sequência deve ser codificada usando-se quatro parâmetros de quantização (QP): 22, 27, 32 e 37. No total, foram utilizados 60 quadros de cada sequência nesta avaliação, o que significa que mais de 2,1 bilhões de amostras de luminância foram consideradas. Apenas amostras de luminância foram consideradas porque é sobre elas que a estimação de movimento é aplicada. Além disso, todas as avaliações foram realizadas utilizando-se o perfil *Main* e considerando-se *Random Access* como configuração temporal. A razão da escolha de *Random Access* ao invés de outra configuração temporal é porque esta é uma configuração mais realista, especialmente para transmissão de vídeo em dispositivos móveis. Por fim, a eficiência de codificação foi avaliada considerando os valores de BD-rate. Os resultados da síntese ASIC foram

obtidos usando-se a ferramenta Encounter RTL Compiler, da Cadence, e tendo como alvo a tecnologia de células padrão TSMC 40nm.

5.2.1.1 Avaliação das Aproximações Algorítmicas

A avaliação realizada em termos de aproximações algorítmicas teve como objetivo o projeto futuro do hardware. Esta avaliação foi apresentada primeiramente em (PERLEBERG et al., 2018), um trabalho anterior desse grupo de pesquisa.

Os resultados apresentados consideraram apenas os quatro tamanhos de blocos quadrados suportados pelo padrão HEVC. Esses tamanhos são os mais utilizados e os mais representativos, considerando-se os valores médios apresentados para as sequências de vídeo utilizadas nos testes. As avaliações mostraram que esses quatro tamanhos juntos são usados para codificar, em média, 50,06% dos *pixels*. Em outras palavras, dos tamanhos de bloco suportados pelo padrão HEVC, esses quatro tamanhos são mais usados do que os outros 20 tamanhos restantes juntos.

O software de referência do padrão HEVC disponibiliza dois algoritmos de busca para serem usados: busca completa, através do algoritmo *Full Search* (FS); e busca rápida, utilizando-se o algoritmo *Test Zone Search* (TZS) (TANG; DAI; CAI, 2010). Considerando o esforço computacional extremamente elevado exigido pelo FS, o algoritmo TZS foi utilizado neste trabalho. Detalhes sobre o funcionamento do algoritmo TZS podem ser encontrados em (TANG; DAI; CAI, 2010). Além disso, foram introduzidas modificações na implementação do TZS para reduzir ainda mais sua complexidade. Neste trabalho, apenas o preditor zero (TANG; DAI; CAI, 2010) foi mantido na primeira etapa. Assim, a área de busca não pode ser movida para uma região distante da posição do bloco colocado dentro do quadro de referência e a área de busca fica colocalizada em relação ao bloco atual. A etapa de *Raster* também foi desativada, dada sua elevada complexidade (TANG; DAI; CAI, 2010). Para garantir um uso eficiente da memória, a predição bidirecional (TANG; DAI; CAI, 2010) também foi desativada e o número de quadros de referência foi limitado a apenas um. Para as etapas de IME e FME, o SAD foi o critério de distorção adotado. Além do SAD, o algoritmo TZS também suporta o SATD como métrica de distorção. A configuração padrão para o algoritmo TZS no software de referência do HEVC usa SAD na IME e SATD na FME. Como o foco deste trabalho é uma arquitetura rápida e com baixa dissipação de potência, o SAD foi usado tanto na IME quanto na FME. Essa opção é consistente com a maioria dos trabalhos relacionados. As modificações no algoritmo

TZS foram essenciais para permitir o projeto de uma arquitetura que fosse uma solução completa para a ME e que apresentasse alta taxa de processamento e baixa dissipação de potência.

Estas restrições foram combinadas a restrições nos tamanhos de blocos suportados. Considerando apenas os quatro tamanhos de blocos quadrados já mencionados, foram avaliadas diversas combinações entre estes tamanhos de blocos. Cada uma destas variações gerou um ponto de operação no hardware. No total, essas combinações resultaram em 11 pontos de operação distintos, uma vez que os tamanhos de blocos não foram avaliados de forma isolada.

A primeira avaliação considerou a eficiência da codificação e o consumo de energia desses 11 pontos de operação a fim de identificar quais pontos são os mais adequados para serem implementados em uma arquitetura de hardware totalmente otimizada. Para cada um desses 11 pontos de operação, uma arquitetura em hardware foi desenvolvida para permitir a avaliação de resultados em hardware. Esses resultados são detalhados em (PERLEBERG et al., 2018). Diferentes compromissos de eficiência de codificação podem ser alcançados variando os pontos de operação. Isso causa impactos também no consumo de energia e em outros resultados de hardware. Como esse é um problema de otimização multivariável, primeiramente foi aplicada uma Análise de Eficiência de Pareto (GHOSH; DEHURI, 2004) considerando dois objetivos para cada um dos 11 pontos de operação: eficiência de codificação e consumo de energia.

O consumo de energia foi estimado considerando-se o processamento de uma CTU – *Coding Tree Unit*, unidade básica de codificação no padrão HEVC (SULLIVAN et al., 2012). A avaliação foi realizada através da síntese ASIC da arquitetura e considerando a frequência necessária para processar sequências de vídeo HD 1080p a 30 quadros por segundo. Ao considerar o conjunto de 11 pontos de operação, o valor de BD-rate varia de 13,79% (ao usar todos os quatro tamanhos de bloco) a 29,68% (ao usar apenas os tamanhos de bloco 32×32 e 64×64 *pixels*). O consumo de energia varia de 11,36 µJ (ao usar todos os quatro tamanhos de bloco) a 5,32 µJ (ao usar apenas tamanhos de bloco 16×16 e 32×32 *pixels*).

A Figura 23 apresenta a análise de eficiência de Pareto para os 11 pontos de operação avaliados. Cada um dos 11 casos é representado por um ponto em preto. A legenda mostra os tamanhos de bloco adotados por cada ponto de operação.

Os quatro casos marcados pela linha verde pontilhada representam a frente de Pareto, ou seja, os quatro pontos ótimos de operação. Estes pontos são considerados ótimos porque nenhum outro ponto apresenta melhores resultados nos dois eixos. Como se pode observar, o caso que utiliza todos os tamanhos de bloco (C_{10}) é aquele que atinge o melhor valor de BD-rate, mas também é o que consome a maior quantidade de energia. Por outro lado, o caso que usa apenas os tamanhos de bloco 16×16 e 32×32 *pixels* (C_3) é o que atinge o menor consumo de energia, mas apresenta impacto considerável em BD-rate.

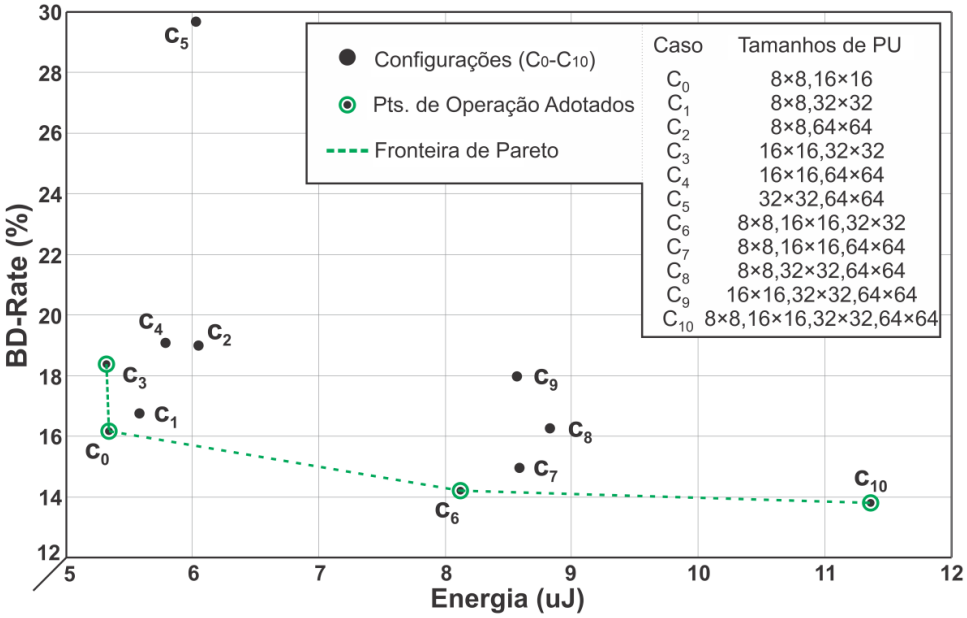


Figura 23 – Análise da Eficiência de Pareto em um espaço qualidade×energia para os pontos de operação da arquitetura.

Assim, os quatro pontos de operação da fronteira de Pareto (C_0, C_3, C_6 e C_{10}) foram utilizados como ponto de partida de uma nova análise, onde os quatro pontos de operação foram comparados considerando um cenário multivariável que inclui: (i) dissipação de potência, (ii) BD-rate, (iii) atraso, (iv) área e (v) produto potência×atraso (PPA). Essas novas análises foram desenvolvidas já no escopo deste trabalho. Análises anteriores referem-se ao trabalho de (PERLEBERG et al., 2018).

A Figura 24 mostra o resultado dessa comparação multivariável na forma de gráficos de radar. Os gráficos da Figura 24 foram gerados de forma normalizada. Dessa forma, os maiores valores de cada eixo, dentre todos os pontos de operação, são representados com o máximo valor percentual. Nessa figura, a menor área em

cinza representa a melhor solução. Como se pode observar, os melhores resultados obtidos em quase todas as variáveis avaliadas pertencem ao caso C_0 e, assim, este ponto de operação foi escolhido para a implementação detalhada que será apresentada na sequência. Esse experimento também mostrou que os modos de operação que processam menos tamanhos blocos podem levar a uma redução expressiva do esforço computacional com impacto aceitável na eficiência de codificação.

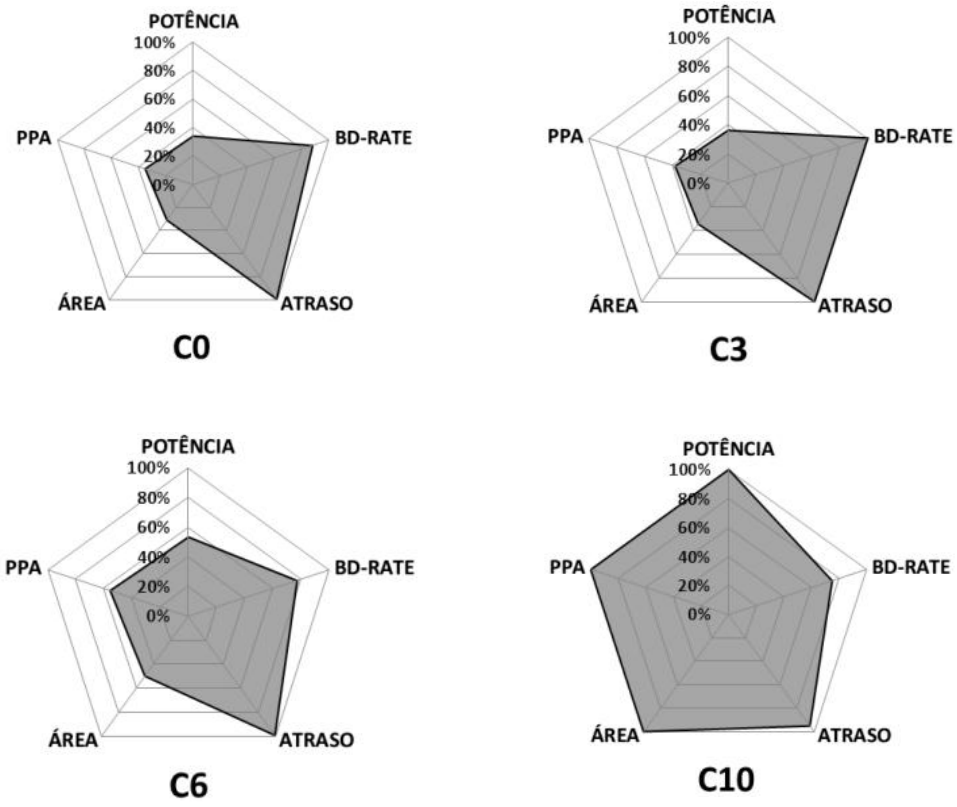


Figura 24 – Comparação multivariável dos pontos de operação da arquitetura selecionados através da Análise de Eficiência de Pareto.

Em particular, a escolha do ponto de operação C_0 representa uma dissipação de potência de 91,30 mW, 4.775 Kgates de área ocupada, um atraso de 20,97 ns, 1,91 nJ para o PPA e 16,17% de aumento de BD-rate. Quando comparado com o ponto de operação C_{10}, que possui a maior área cinza na Figura 24, o ponto de operação C_0 fornece reduções de 66% em energia, 69% em área e 65% em PPA, e aumentos de 4% no atraso e 17% em BD-rate. É importante observar que, como BD-rate é uma métrica relativa, essas porcentagens indicam a necessidade de aumentar a taxa de bits

para manter a mesma qualidade objetiva de vídeo. Se o objetivo fosse considerar a perda de qualidade para a mesma taxa de bits, deveríamos observar o efeito em BD-PSNR. Os valores de BD-PSNR foram obtidos na mesma avaliação realizada para obter os valores de BD-rate e indicam uma perda de 0,37 dB considerando-se o ponto de operação C_0.

5.2.1.2 Avaliação e Caracterização dos Operadores Imprecisos

Uma segunda avaliação foi realizada para identificar os operadores imprecisos mais adequados para serem usados na arquitetura da ME inteira e fracionária, considerando o ponto de operação C_0. Esse ponto de operação suporta tamanhos de bloco de 8×8 e 16×16 *pixels*, conforme descrito anteriormente. Essa avaliação foi feita de forma semelhante ao que foi apresentado na seção 4.1. Nesse caso, a ideia é utilizar somadores imprecisos no cálculo de SAD usados tanto na IME quanto na FME. Essa operação foi escolhida para receber aproximação por ser a operação mais computacionalmente intensiva na etapa de estimação de movimento. O objetivo foi de proporcionar a maior redução possível no consumo de energia e no uso recursos de hardware.

Para atingir esse objetivo, um conjunto de funções de somas imprecisas de 8 bits foi descrito em C++ e avaliado. Cada somador foi avaliado usando-se 99.840 amostras de luminância extraídas do primeiro quadro da sequência de vídeo de teste *BasketballPass_416x240_50.yuv*, da classe D das CTC. Foram avaliadas 26 versões dos seis operadores de 8 bits apresentados anteriormente. A partir dessas primeiras análises, foram identificadas as melhores configurações para esses operadores imprecisos: (i) ACA-II com três subsomadores sobrepostos, quatro bits cada um; (ii) CCB com dois subsomadores com quatro bits cada e um bit para definir o corte; (iii) ETA-I com três bits precisos e cinco imprecisos; (iv) ETA-IV com três subsomadores (três, três e dois bits) usando dois bits na primeira geração de *carry* e três na segunda; (v) GeAr usando dois subsomadores sobrepostos com cinco bits cada; e (vi) LOA com três bits precisos e cinco imprecisos.

Após, esses seis operadores imprecisos foram inseridos na etapa de estimação de movimento (IME e FME) do software de referência do HEVC. Dessa forma, foi possível avaliar seus impactos na eficiência de codificação em termos de BD-rate. Os operadores aproximados foram inseridos apenas no primeiro estágio dos módulos de cálculo de SAD para evitar efeitos de acumulação de erros, do mesmo modo que

nos experimentos anteriores. Os resultados médios dessa avaliação são apresentados na Tabela 14 e foram obtidos pela codificação das vinte sequências de vídeo de teste, usando os quatro valores de QP recomendados nas CTC e usando-se a configuração *Low Delay P Main*. Nessa configuração, apenas o primeiro quadro de cada sequência de teste é do tipo I, os demais são quadros do tipo P, dando ênfase à predição interquadros. As CTC não recomendam vídeos de teste da Classe A para esta configuração. Assim, restam 20 vídeos das outras classes para serem testados. De acordo com a Tabela 14, os melhores resultados foram obtidos com LOA e ETA-IV.

Tabela 14 – Resultados de BD-Rate para os operadores.

Operador	ACA-II	CCB	ETA-I	ETA-IV	GeAr	LOA
BD-rate (%)	7,5	2,8	2,5	0,2	3,9	0,8

O próximo experimento foi conduzido para obter uma caracterização em hardware dos operadores aproximados. Os resultados de síntese são os mesmos que foram apresentados na seção 4.1 e são repetidos aqui para que possam ser avaliados juntamente com os novos valores de BD-rate. Os operadores foram descritos em VHDL e sintetizados em ASIC para a biblioteca de células padrão Nangate 45nm (NANGATE, 2019), usando a ferramenta *Encounter RTL Compiler*, da Cadence.

Esses resultados experimentais foram comparados com os obtidos para um somador preciso do tipo RCA considerando o aumento ou diminuição em termos de: (i) dissipação de energia, (ii) atraso, (iii) área e (iv) produto potência × atraso. A decisão de utilizar o somador RCA como comparativo é justificada porque ele é comumente usado como referência em muitos trabalhos na literatura como, por exemplo, (MAHDIANI et al., 2010), (DUTT; NANDI; TRIVEDI, 2016), (ZHU et al., 2010) e (SHAFIQUE et al., 2015). A comparação multivariável dos operadores aritméticos aproximados é mostrada na Figura 25, novamente na forma de gráficos de radar. Os resultados de BD-rate também foram incluídos. Pode-se notar que o operador LOA alcançou o melhor desempenho pois, além de apresentar o melhor resultado em quase todos os eixos, também apresentou a menor área em cinza entre todos os operadores avaliados. Assim, o operador LOA foi escolhido para ser usado também na arquitetura de ME com eficiência energética.

A escolha do operador LOA resulta em uma potência dissipada de 105 µW, uma área com 65 células, 693ns de atraso, 72,77pJ para o PPA e BD-rate com aumento de 2,5%. Quando comparado com o somador CCB (que possui a maior área em cinza

na Figura 25), o operador LOA reduz a potência dissipada em 16%, a área em 5%, o atraso em 26%, o PDP em 38% e o valor de BD-rate em 36%.

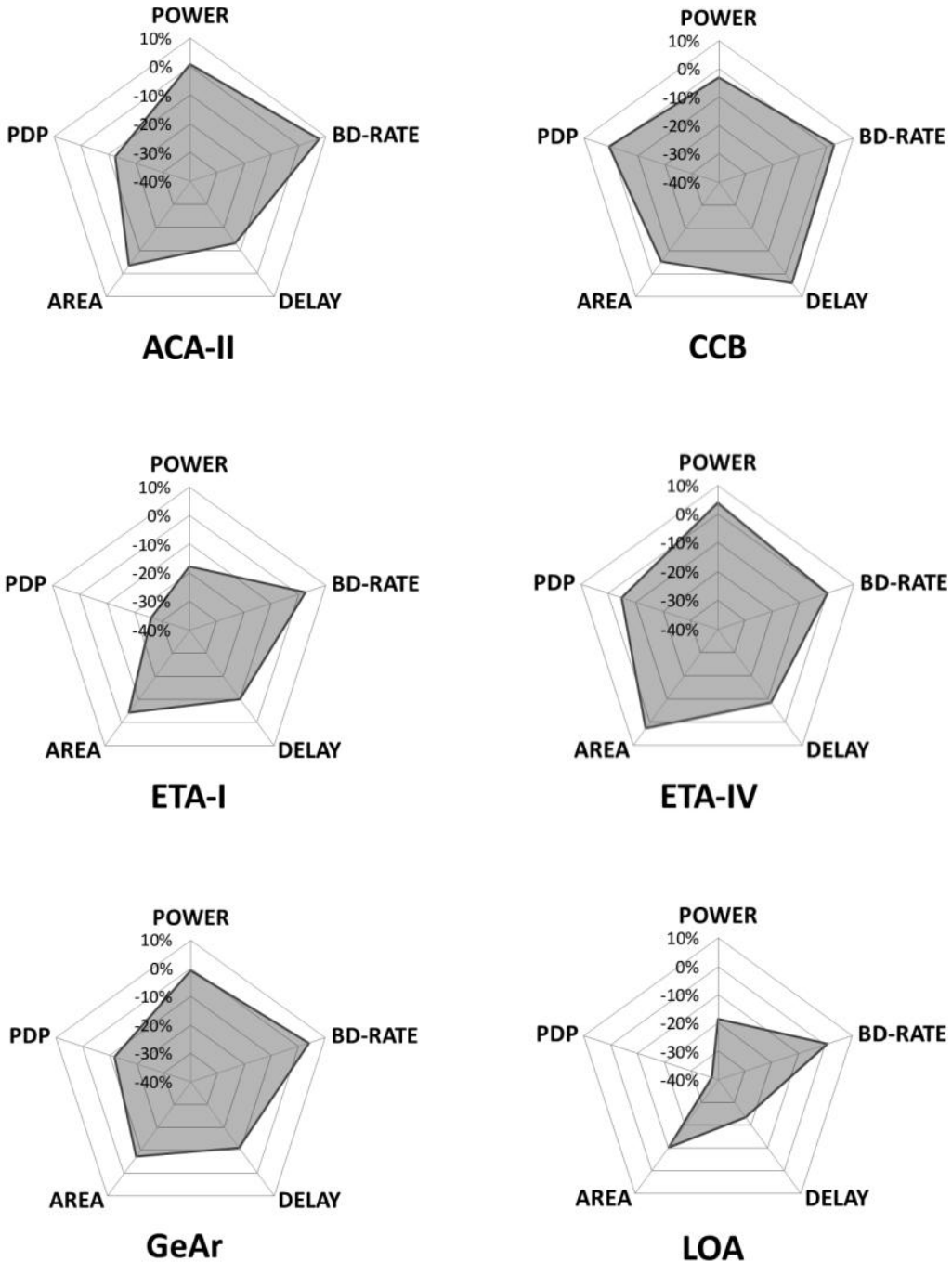


Figura 25 – Comparação multivariável dos operadores imprecisos considerados para uso na arquitetura rápida para ME.

5.2.2 *Arquitetura com Baixa Dissipação de Energia para a Estimação de Movimento do Padrão HEVC*

A arquitetura apresentada nesta subseção é uma solução completa para estimação de movimento, por implementar tanto a parte inteira (IME) quanto a parte fracionária (FME). A redução na dissipação de potência dessa arquitetura é obtida através da utilização de técnicas de computação aproximada na implementação da ME. Suas restrições e pontos de operação foram definidos na subseção 5.2.1.1. Além disso, essa arquitetura também considera aproximação no cálculo de SAD através do uso do operador aproximado definido na subseção 5.2.1.2.

A estrutura de processamento toma por base a arquitetura para ME proposta em (PERLEBERG et al., 2018). O mesmo modelo arquitetural foi utilizado, mas a nova arquitetura foi bastante simplificada. Na nova implementação são suportados apenas dois tamanhos de bloco, enquanto a versão original suportava quatro tamanhos. A seguir, as unidades de cálculo SAD foram substituídas por unidades que utilizam operadores imprecisos. A arquitetura utiliza o algoritmo TZS modificado para executar a etapa da IME. Para executar a FME foram implementados filtros de interpolação, conforme definidos pelo padrão HEVC. Essa arquitetura possui vários módulos implementados de forma dedicada para que se possa manter uma alta taxa de processamento. Assim, cada tamanho de bloco a ser processado possui unidades especificas de IME e de FME.

De acordo com as avaliações apresentadas anteriormente, o caso que resulta no melhor *tradeoff* na comparação multivariável é o caso C_0, que processa blocos com tamanhos de 8×8 e 16×16 *pixels*. Dessa forma, esses dois tamanhos de bloco foram adotados na implementação dessa arquitetura. Com o objetivo de processar vídeos de alta resolução, alguns desses módulos também foram replicados. Isso resultou em quatro módulos para processar os blocos de 8×8 *pixels* e em dois módulos para processar os blocos de tamanho 16×16 *pixels*. Todos esses módulos podem trabalhar em paralelo durante a codificação do vídeo.

A arquitetura completa para ME é apresentada na Figura 26, onde são apresentados todos os módulos que compõem a IME e a FME para os dois tamanhos de bloco adotados neste trabalho. Cada módulo processa um grupo de blocos predefinido. Quando cada módulo da FME termina de processar um bloco candidato, a unidade de controle global copia o resultado em uma tabela de SAD. Esta tabela

armazena o valor de SAD e o vetor de movimento relacionado a esse valor. Os valores armazenados somente são disponibilizados quando todos os módulos concluem o processamento da CTU.

É importante destacar que a redução dos acessos à memória externa também é essencial para a redução no consumo de energia. Porém, mesmo com o acesso à memória sendo uma grande preocupação ao projetar uma arquitetura de estimação de movimento, esse problema não será tratado aqui por não ser o foco deste trabalho. No entanto, uma alternativa para minimizar os acessos à memória externa seria armazenar localmente toda a área de busca, usando algum método de reuso de dados, como *Level C* (TUAN; CHANG; JEN, 2002), e usando uma memória adicional para as posições interpoladas da FME. Assim, tanto a IME quanto a FME poderiam operar simultaneamente sem a necessidade de acessos adicionais à memória externa.

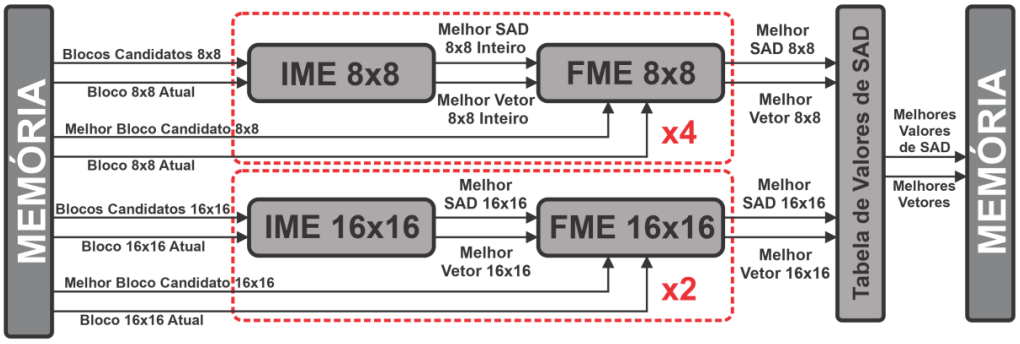


Figura 26 – Diagrama em blocos da arquitetura completa para ME. Adaptado de (PERLEBERG et al., 2018).

A seguir, a solução em hardware para as duas etapas de ME é explicada em detalhes. Após, será apresentada a arquitetura para cálculo aproximado de SAD utilizando operadores imprecisos.

5.2.2.1 Arquitetura da Estimação de Movimento Inteira

A unidade de hardware projetada para executar a etapa IME toma por base a arquitetura apresentada em (PERLEBERG et al., 2018) e foi dividida em parte de controle e parte operativa. A parte de controle seleciona os blocos candidatos a serem comparados. Esse procedimento é feito de acordo com o algoritmo TZS modificado. O algoritmo TZS modificado implementa apenas o preditor zero na etapa de busca inicial e tem a etapa *Raster* desativada, como já discutido. Assim, a área de busca fica colocalizada com o bloco atual, garantindo uma comunicação regular com a memória.

Além disso, todas as expansões do TZS nas primeiras etapas de pesquisa e refinamento sempre recebem 16 blocos candidatos da memória. O número total de expansões também diminuiu de sete para cinco, considerando-se o algoritmo TZS original em relação à versão modificada. Essas simplificações estão detalhadas em (PERLEBERG et al., 2018). A parte de controle também trata das condições de parada de cada etapa do algoritmo TZS, conforme explicado a seguir.

A parte operativa da IME realiza, em paralelo, a comparação dos 16 blocos candidatos selecionados pela parte de controle. Ao final, é produzido um vetor de movimento correspondente ao melhor bloco candidato, de acordo com o critério de similaridade SAD. O processamento é feito em uma linha por ciclo de *clock*. Dessa forma, esta unidade foi projetada para processar, em paralelo, uma linha de cada um dos 16 blocos candidatos selecionados.

Para garantir o processamento em tempo real de vídeos UHD, a comparação entre cada bloco candidato e o bloco atual é feita usando uma unidade especializada, capaz de processar cada um dos tamanhos de bloco adotados. Essa arquitetura é apresentada na Figura 27, juntamente com suas três etapas. A primeira etapa corresponde às árvores de SAD, responsáveis pelo cálculo do valor SAD de uma linha do bloco atual em relação a uma linha do bloco candidato. A arquitetura da árvore de SAD será melhor apresentada na seção 5.2.2.3.

Após as árvores de SAD realizarem o processamento, o valor de SAD obtido é acumulado pelos acumuladores de SAD. Enquanto isso, as árvores de SAD processam as linhas de blocos restantes. A arquitetura de um acumulador de SAD é apresentada na Figura 28 (a). Quando a árvore SAD conclui o processamento da última linha do bloco processado, o valor em cada acumulador de SAD corresponde ao valor de SAD do bloco candidato avaliado.

Ao final do processamento da última linha do bloco considerado, a etapa de comparação processa os valores de SAD disponibilizados pelos acumuladores juntamente com seus vetores de movimento. A comparação é feita dois a dois, com todos os valores de SAD. Quando o processamento termina, o comparador disponibiliza o menor valor de SAD e seu respectivo vetor de movimento. Devido ao uso de *pipeline* na arquitetura, o comparador precisa de quatro ciclos para processar os valores de SAD dos 16 blocos candidatos. A Figura 28 (b) apresenta o comparador de SAD.

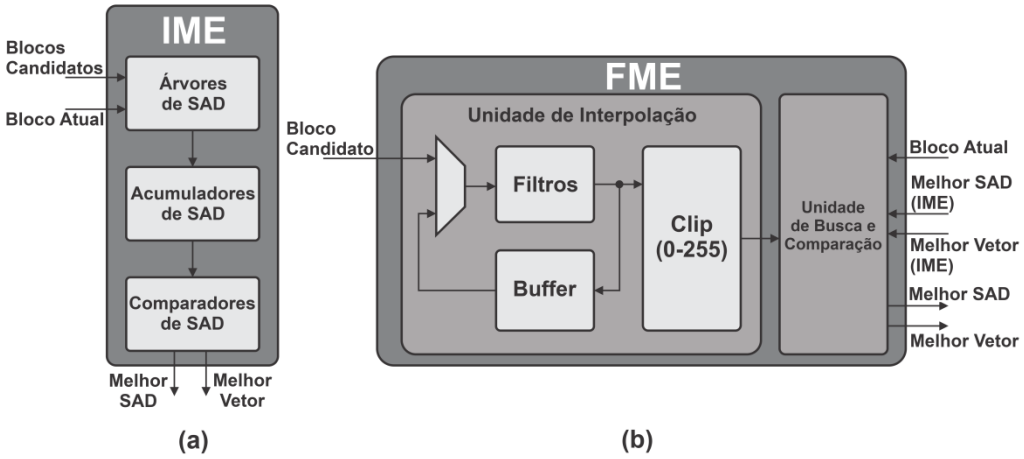


Figura 27 – Diagrama em blocos das arquiteturas dos módulos da (a) IME e da (b) FME. Adaptado de (PERLEBERG et al., 2018).

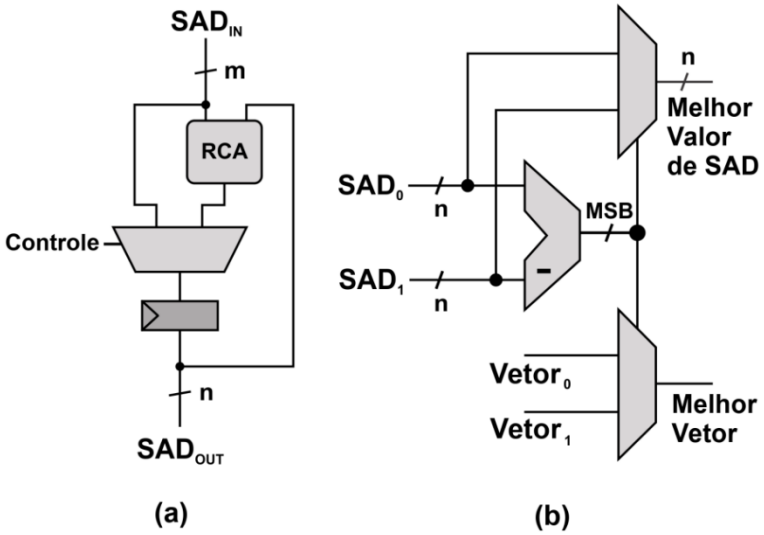


Figura 28 – Módulos do acumulador de SAD (a) e do comparador de SAD (b). Adaptado de (PERLEBERG et al., 2018).

A parte de controle do TZS modificado também é responsável pela implementação de uma condição de parada. No TZS original, é possível passar livremente por todos os blocos da área de busca enquanto a etapa de refinamento busca um valor de SAD menor em uma nova iteração. Isso é um problema quando se considera processamento em tempo real porque o número de ciclos necessários para obter o melhor bloco candidato seria indefinido e dependente dos dados do vídeo. Dessa forma, para garantir processamento em tempo real, a parte de controle da IME limita em 240 o número de blocos candidatos a serem avaliados. De acordo com

avaliações anteriores apresentadas em (PERLEBERG et al., 2018), 93,33% das execuções do TZS usam menos de 240 blocos candidatos.

O módulo IME 8×8 requer 148 ciclos de *clock* para processar os 240 blocos candidatos. Esse módulo é o gargalo da arquitetura completa da IME, definindo a taxa de processamento alcançada, como será discutido mais adiante.

5.2.2.2 Arquitetura da Estimação de Movimento Fracionária

A unidade de FME apresentada neste trabalho é totalmente compatível com o padrão HEVC e é usada para melhorar a estimação de movimento. Através de interpolação de pontos, novos blocos são criados em torno do melhor resultado gerado pela IME. Esses novos blocos são gerados em posições fracionárias de ½ e ¼ de amostra (SZE; BUDAGAVI; SULLIVAN, 2014). Após a interpolação, um bloco dentre os novos candidatos fracionários é selecionado como o melhor candidato dentro da ME. A etapa de interpolação da FME que foi definida para essa arquitetura segue o que é recomendado pelo padrão HEVC. Porém, a arquitetura dos filtros pode ser adaptada para também suportar outros codificadores.

O processo de FME foi dividido em duas unidades, conforme apresentado na Figura 27 (b). A primeira é a Unidade de Interpolação, que gera os novos blocos candidatos, em posições fracionárias. A segunda é a Unidade de Busca e Comparação. Essa unidade é utilizada para avaliar os blocos candidatos que foram gerados por interpolação e, dentre eles, selecionar o que é mais semelhante ao bloco atual.

A Unidade de Interpolação possui um multiplexador para selecionar as amostras que serão passadas para os filtros. As amostras podem ser fornecidas tanto pelo melhor bloco candidato fornecido pela IME quanto pelo *buffer* interno. Após isso, um conjunto de filtros é utilizado para interpolar as novas amostras. Dessa forma, posições fracionárias são geradas em torno das amostras inteiras do bloco de entrada. Em seguida, essa arquitetura gera e avalia todos os 48 blocos fracionários possíveis.

Existem três tipos de filtros usados para interpolar os novos blocos e seu uso depende da posição de cada amostra fracionária. Detalhes específicos desses filtros podem ser encontrados em (AFONSO et al., 2016). Após a aplicação dos filtros, o *buffer* da Unidade de Interpolação é usado para armazenar as amostras fracionárias horizontais, uma vez que as amostras horizontais podem ser utilizadas como entrada para a etapa de interpolação das amostras diagonais. Além disso, uma operação de

recorte (clipping) é aplicada às saídas dos filtros para manter todas as amostras com valores de oito bits. Dessa forma, essas amostras podem ser processadas pela Unidade de Busca e Comparação. Assim, todas as saídas da Unidade de Interpolação têm valores entre 0 e 255.

A Figura 29 apresenta uma amostra inteira e as amostras fracionárias que podem ser interpoladas em torno dela, seguindo o que é definido pelo padrão HEVC. Na Figura 29 é possível notar que essas amostras podem ser categorizadas em três grupos. De acordo com o deslocamento da amostra fracionária em relação à amostra inteira ela pode ser horizontal, vertical ou diagonal. Portanto, a geração das amostras fracionárias também foi dividida sequencialmente de acordo com o deslocamento das amostras.

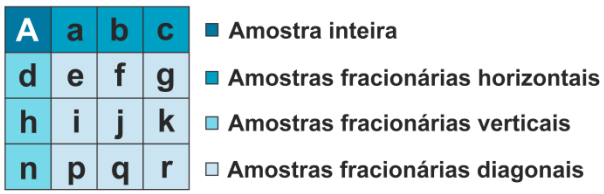


Figura 29 – Amostras fracionárias de acordo com suas posições em relação à amostra inteira. Adaptado de (PERLEBERG et al., 2018).

O processamento começa pela geração das amostras horizontais. A cada ciclo de *clock* os filtros recebem uma linha horizontal do melhor bloco da IME e geram as novas amostras que vão compor seis novos blocos fracionários horizontais. Esses blocos horizontais são armazenados em um *buffer* interno. Posteriormente, as amostras verticais são geradas. De forma semelhante à geração das amostras horizontais, a cada ciclo de *clock* a unidade de interpolação recebe uma coluna vertical do melhor bloco da IME. Esse processamento gera as amostras que são usadas para compor seis novos blocos fracionários verticais. Por fim, as amostras diagonais são geradas usando-se as amostras horizontais armazenadas no *buffer* interno. As amostras diagonais são usadas para compor 36 novos blocos fracionários em torno do melhor bloco da IME.

No final do processamento das amostras fracionárias os blocos gerados são enviados à Unidade de Busca e Comparação para avaliação. A Unidade de Busca e Comparação realiza a avaliação de maneira semelhante à utilizada na IME. Doze árvores são usadas para calcular o valor de SAD dos blocos candidatos que

estão sendo avaliados. O valor de SAD de cada candidato é armazenado em um dos 48 acumuladores de SAD enquanto a unidade de interpolação processa o restante dos blocos candidatos. Quando finaliza o processamento de todos os 48 blocos candidatos, os valores dos 48 acumuladores de SAD são repassados para o comparador. É o comparador que definirá qual bloco candidato fracionário tem o menor valor de SAD. Ao final, esse valor de SAD é enviado para a saída juntamente com seu respectivo vetor de movimento.

5.2.2.3 Arquitetura Aproximada para Cálculo de SAD

Como mencionado anteriormente, as árvores de SAD são responsáveis por calcular o valor de SAD de cada linha que está sendo processada. As árvores de SAD das arquiteturas da IME e da FME foram implementadas com operadores aproximados, conforme mostra a Figura 30.

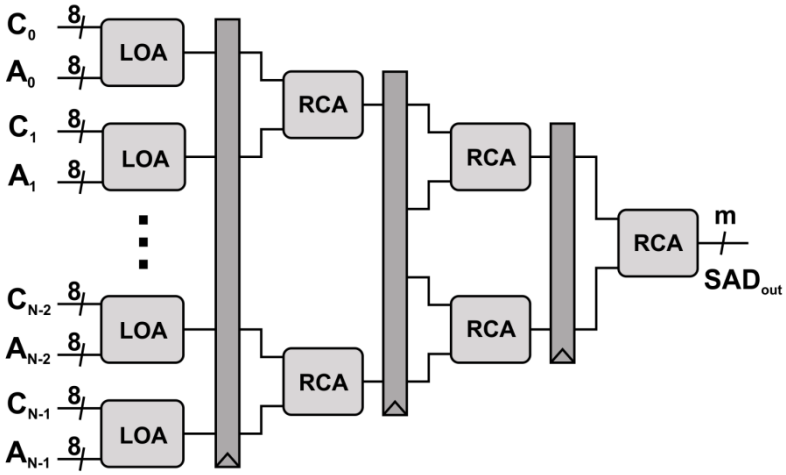


Figura 30 – Diagrama de blocos de uma árvore de SAD. Adaptado de (PERLEBERG et al., 2018).

Vários estágios de *pipeline* foram utilizados entre os operadores a fim de alcançar maiores taxas de processamento. No primeiro estágio de *pipeline*, operadores LOA foram utilizados para obter a diferença entre o bloco candidato e o bloco atual. As amostras do bloco candidato são recebidas pelas entradas C_n, enquanto que as amostras do bloco atual são fornecidas nas entradas A_n. Nos estágios de *pipeline* seguintes os valores de SAD das amostras são somados dois a dois. Para obter isso, somadores do tipo RCA foram usados para acumular os valores de SAD de toda uma linha de amostras de bloco candidato. Portanto, o resultado de cada árvore é o valor de

SAD de cada bloco candidato em relação ao bloco atual. A escolha de operadores RCA justifica-se pela ideia de avaliar os impactos na taxa de processamento a partir do uso de operadores LOA.

5.2.3 *Resultados de Síntese*

A arquitetura para ME foi sintetizada com a ferramenta Cadence Encounter RTL Compiler usando-se a biblioteca de células padrão TSMC 40nm. O esforço de síntese foi definido como "forte" e o uso de *clock gating* foi ativado. A ferramenta de estimativa de potência considerou atividade de chaveamento padrão.

A Tabela 15 resume os resultados de síntese obtidos, além de compará-los com resultados de trabalhos relacionados. Foram consideradas três frequências de operação. A primeira configuração considerou uma frequência alvo de 145MHz. Com essa frequência de operação a arquitetura pode processar vídeos UHD 4K (3840×2160 *pixels*) a 30 quadros por segundo. Essa configuração utiliza 1.541Kgates e dissipa 108,92mW. A eficiência energética dessa implementação é de 0,44nJ/amostra.

A segunda frequência alvo foi 580MHz, o que permite o processamento de vídeos UHD 8K (7680×4320 *pixels*) a 30 quadros por segundo. Nesta frequência, a dissipação de potência é de 256,5mW e a eficiência energética é de 0,26nJ/amostra.

A terceira síntese foi realizada para a frequência máxima de 2.330MHz. Na sua frequência máxima, a arquitetura pode processar vídeos UHD 4K a 480 quadros por segundo ou vídeos UHD 8K a 120 quadros por segundo. Com essa configuração a arquitetura dissipa 585,90mW e possui eficiência energética de 0,15nJ/amostra

Considerando esses três resultados de síntese, pode-se observar que quanto maior é a taxa de processamento, maior é a eficiência energética. O melhor resultado de eficiência energética é 0,15nJ/amostra, obtido com a frequência máxima de operação.

Tabela 15 – Resultados de Síntese.

	ME Aproximada			
Resolução Máxima	3840×2160	7680×4320	3840×2160	7680×4320
	30fps	30fps	480fps	120fps
Taxa de Processamento (Mpixels/s)	248,83	995,33	3.981,31	
Área (Kgates)	1.541	1.544	1.660	
Potência (mW)	108,92	256,50	585,90	
Frequência (MHz)	145	580	2.320	
Eficiência Energética (nJ/amostra)	0,44	0,26	0,15	

5.2.4 *Comparação com Trabalhos Relacionados*

Uma comparação da arquitetura desenvolvida nesse trabalho com trabalhos relacionados é apresentada na Tabela 16.

Tabela 16 – Comparação com trabalhos relacionados.

	TR1*	**TR2***	**TR3***	**TR4***	**TR5***	**TR6***	**ME Aproximada**			
Tecnologia	90 nm	65 nm	65 nm	65nm	90 nm	65 nm	40 nm			
Tipo de ME	IME + FME	FME	IME	IME	IME	IME	IME + FME			
Resolução Máxima	4096 × 2048	7680 × 4320	3840 × 2160	3840 × 2160	3840 × 2160	8192 × 4320	3840 × 2160	7680 × 4320	3840 × 2160	7680 × 4320
	60 fps	30 fps	30 fps	30 fps	30 fps	30 fps	30 fps	30 fps	480 fps	120 fps
Algoritmo de Busca	TZS modificado	-	Busca rápida	TSS	HGS modificado	Novo algoritmo rápido	TZS modificado			
Taxa de Processamento (Mpixels/s)	503,32	995,33	248,83	248,83	248,83	1.061,68	248,83	995,33	3.981,31	
Tamanhos de Bloco	64×64 a 8×8	64×64 a 16×8	64×64 a 4×4	32×32 a 8×4	32×32 a 8×8	todos do HEVC	16×16 e 8×8			
Área (Kgates)	779	1.183	434	617	1,441	2.310	1.541	1.544	1.660	
Potência (mW)	-	198,6	-	-	151,76	362	108,92	256,50	585,90	
Frequência (MHz)	270	188	720	350	250	290	145	580	2.320	
Eficiência Energética (nJ/amostra)	-	0,20	-	-	0,61	0,27	0,44	0,26	0,15	

* TR1: (JOU; CHANG; CHANG, 2015)
TR2: (HE et al., 2015)
TR3: (MEDHAT; SHALABY; SAYED, 2015)
TR4: (PARK et al., 2016)
TR5: (SINGH; AHAMED, 2018)
TR6: (JIA et al., 2020)

Novamente não é fácil fazer uma comparação completa e justa com trabalhos relacionados uma vez que esses trabalhos consideram tecnologias diferentes e tem foco em diferentes taxas de processamento. Além disso, os trabalhos relacionados consideram diferentes tipos de ferramentas de codificação, como o suporte à estimação de movimento fracionária (FME) ou estimação de movimento inteira (IME), o número de tamanhos de bloco que podem ser avaliados, os algoritmos de busca utilizados, e etc.

É importante destacar, a partir dos trabalhos apresentados na Tabela 16, que apenas o trabalho apresentado em (JOU; CHANG; CHANG, 2015) e nossa arquitetura são soluções completas para estimação de movimento, ou seja, consideram tanto IME

quanto FME. No entanto, o trabalho em (JOU; CHANG; CHANG, 2015) não apresenta seus resultados de potência dissipada e nem sua caracterização de consumo de energia. Dessa forma, também não é possível determinar a eficiência energética para realizar comparações adicionais.

A solução apresentada em (HE et al., 2015) tem uma eficiência energética ligeiramente superior à apresentada por nossa solução. No entanto, aquele trabalho trata apenas da FME, enquanto nossa arquitetura possui uma solução completa para ME. Além disso, a IME é muito mais complexa que a FME. Mesmo suportando apenas FME a solução em (HE et al., 2015) atinge uma taxa de processamento máxima que é quatro vezes menor do que a alcançada por nossa arquitetura.

Os trabalhos (MEDHAT; SHALABY; SAYED, 2015), (PARK et al., 2016), (SINGH; AHAMED, 2018) e (JIA et al., 2020) apresentam soluções apenas para IME. Entre essas três soluções, apenas (SINGH; AHAMED, 2018) e (JIA et al., 2020) apresentam resultados de potência, possibilitando determinar também suas eficiências energéticas. Quando comparada à (SINGH; AHAMED, 2018), nossa solução é 28% mais eficiente em termos de energia e trabalha com uma frequência 42% menor para o mesmo objetivo de desempenho (3840×2160 a 30fps). Além disso, nossa arquitetura pode atingir uma taxa de processamento máxima 16 vezes maior que a alcançada em (SINGH; AHAMED, 2018). Ao comparar nossa arquitetura com a apresentada em (JIA et al., 2020), a eficiência energética resultante é bastante semelhante. Porém, nossa solução pode atingir uma taxa de processamento máxima que é quase quatro vezes maior do que a alcançada em (JIA et al., 2020). Os trabalhos (MEDHAT; SHALABY; SAYED, 2015) e (PARK et al., 2016) não apresentam resultados de energia. Mesmo assim, nossa arquitetura é capaz de atingir a mesma taxa de processamento em uma frequência de operação quase cinco vezes menor que (MEDHAT; SHALABY; SAYED, 2015) e 2,4 vezes menor que (PARK et al., 2016). Isso indica que nossa solução alcançou potência mais baixa e eficiência energética mais alta do que esses dois trabalhos. Outra comparação importante está relacionada à taxa de processamento, uma vez que nossa arquitetura atingiu uma taxa de processamento máxima que é 16 vezes maior do que a alcançada em (MEDHAT; SHALABY; SAYED, 2015) e (PARK et al., 2016).

Por fim, nossa solução possui a maior taxa de processamento e a melhor eficiência energética entre os trabalhos relacionados, mesmo quando comparada aos trabalhos que implementam apenas FME ou IME.

5.2.5 *Considerações Finais da Seção*

Esta subseção apresentou uma arquitetura para estimação de movimento que utiliza computação aproximada para obter eficiência energética. Essa arquitetura suporta tanto a estimação de movimento inteira quanto a fracionária. Essa arquitetura é totalmente compatível com o padrão HEVC e pode ser facilmente adaptada para ser compatível com outros codificadores atuais como AV1 (AOM, 2019), AVS2 (HE et al., 2014) e VVC (MPEG, 2019). Para que isso seja possível, apenas são necessárias adaptações nos filtros de interpolação da FME. Nessa arquitetura, foram realizadas aproximações em nível algorítmico, limitando o número de tamanhos de bloco suportado, além de outras pequenas simplificações no algoritmo TZS. A arquitetura também explora computação aproximada no nível dos operadores aritméticos. Dessa forma, somadores imprecisos do tipo LOA foram utilizados nos cálculos de SAD para melhorar a eficiência energética. Ao executar em sua máxima frequência máxima, a arquitetura projetada é capaz de processar vídeos UHD 8K (7680×4320 pixels) em tempo real a uma taxa de 120 quadros por segundo.

Várias avaliações e análises que dão suporte à implementação da arquitetura foram apresentadas neste capítulo. Os resultados de síntese mostraram uma dissipação de potência de 108,92mW e uma área de 1.541Kgates quando o alvo é uma taxa de processamento suficiente para processar vídeos UHD 4K (3840×2160 pixels) a 30 quadros por segundo, com um impacto médio de 16,17% em BD-rate por conta das simplificações implementadas.

A arquitetura apresentada neste trabalho alcançou a maior taxa de processamento máxima entre todos os trabalhos relacionados. Além disso, oferece o melhor resultado em termos de eficiência energética, mesmo quando comparado às arquiteturas que executam apenas FME ou IME.

Considerando os resultados alcançados, pode-se concluir que o uso de computação aproximada em nível algorítmico e no nível de operadores aritméticos pode ser uma ferramenta poderosa para projetos de hardware dedicados visando codificadores de vídeo de alta resolução com eficiência energética.

6 OPERADOR ARITMÉTICO ESCALÁVEL EM POTÊNCIA E PRECISÃO

Este capítulo apresenta um operador aritmético otimizado e configurável desenvolvido no escopo desta tese. Nele, dissipação de potência e precisão podem ser ajustados dinamicamente através de vários pontos de operação.

O operador escalável em potência e precisão, *Power-Precision Scalable Adder* (2PSA), é um somador flexível que suporta uma diversidade de pontos de operação. Assim, é possível ajustar dinamicamente precisão e dissipação de potência de acordo com os requisitos da aplicação.

6.1 Arquitetura do Operador 2PSA

O somador 2PSA é um operador com precisão e potência configuráveis dinamicamente. Os pontos de operação deste somador podem ser definidos de acordo com os requisitos do sistema ou através de definições externas como o status da bateria, configurações específicas de economia de energia, entre outras. O 2PSA também pode ser configurado estaticamente para suportar um número diferente de pontos de operação (NPO). Isso pode ser feito em tempo de projeto e de acordo com a aplicação alvo. O nível máximo de pontos de operação é igual à largura de bits do operador. Por exemplo, para um 2PSA de 4 bits, os pontos de operação permitidos são P, I_1, I_2 e I_3, onde P é o ponto de operação preciso e I_n são os pontos de operação imprecisos com n bits imprecisos. O nível mínimo de pontos de operação permitido é igual a um, quando o 2PSA suporta apenas o ponto de operação preciso.

O operador 2PSA explora o conceito do operador LOA (MAHDIANI et al., 2010) em uma nova dimensão, utilizando dois subsomadores não sobrepostos com larguras de bit configuráveis dinamicamente. O operador foi otimizado para reduzir a sobrecarga de hardware causada pelo suporte a diferentes pontos de operação e utiliza isolamento de operandos (BANERJEE et al., 2005) para diminuir ainda mais a dissipação de potência.

Um operador 2PSA de *n*-bits é definido com duas entradas de *n* bits, uma entrada *PQControl* para definir dinamicamente o ponto de operação, uma saída de *n* bits e uma saída para o valor de *carry out*. Até o valor *n*, qualquer largura de bits é suportada pelo operador 2PSA. Por outro lado, a largura de bits da entrada *PQControl* depende dos pontos de operação suportados, definidos por $\lceil \log_2(NPO) \rceil$.

A Figura 31 apresenta o diagrama em blocos de um operador 2PSA de 4 bits, onde os blocos OI_n são portas utilizadas para isolamento de operandos, os blocos IA_n são operadores imprecisos de 1 bit e os blocos PA_n são operadores precisos de 1 bit. Na Figura 31, dependendo dos valores da entrada *PQControl* da Fig. 1, as saídas IA_n e PA_n são combinadas para gerar as saídas para cada nível de precisão. A mesma entrada *PQControl* controla o *carry in* de um bloco PA_n, que pode ser o *carry out* de um bloco PA_{n-1} ou de um bloco IA_{n-1}. Além disso, *PQControl* é utilizado para controlar os blocos OI_n, definindo o isolamento de operandos de cada somador de 1 bit visando reduzir a potência dissipada quando os somadores não são utilizados.

A Figura 32 apresenta as estruturas que compõem o operador 2PSA. O módulo IA_n, o somador impreciso, é inspirado na estrutura do operador LOA (MAHDIANI et al., 2010). O módulo PA_n é um somador preciso completo de 1 bit. Por fim, o módulo OI_n realiza o isolamento de operandos nas entradas dos somadores.

Conforme apresentado na Figura 31, a implementação do operador 2PSA requer hardware extra quando comparado a um operador RCA ou a um operador LOA. Esse hardware extra é necessário para suportar a seleção dinâmica dos pontos de operação. Como resultado, se o operador 2PSA for implementado para suportar um ponto de operação preciso, esse ponto de operação tende a dissipar mais potência do que um operador RCA com a mesma largura de bits. No entanto, quando selecionados, os pontos de operação imprecisos permitem reduções expressivas na dissipação de potência.

O operador 2PSA permite que tanto a aplicação quanto o usuário definam o ponto de operação. Esse ponto de operação pode ser preciso ou de baixa imprecisão, quando o dispositivo estiver conectado a uma fonte de alimentação, ou de maior imprecisão, quando o dispositivo estiver com nível baixo de carga na bateria. No primeiro caso, a maior precisão é oferecida ao custo de maior dissipação de potência. No segundo caso, a maior imprecisão possibilita aumento no tempo de uso da bateria.

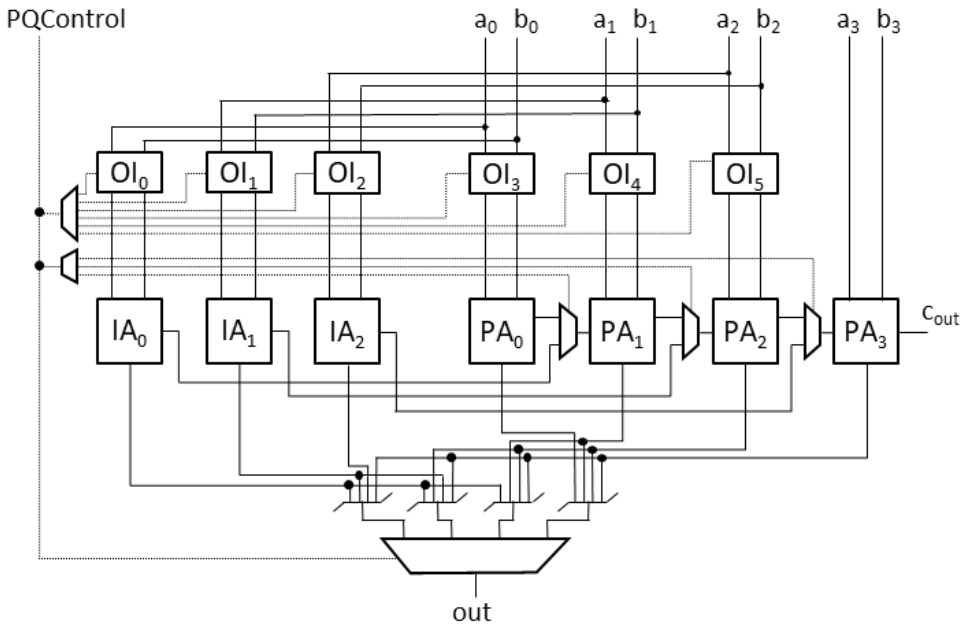


Figura 31 – Diagrama em Blocos de um Operador 2PSA com 4 bits.

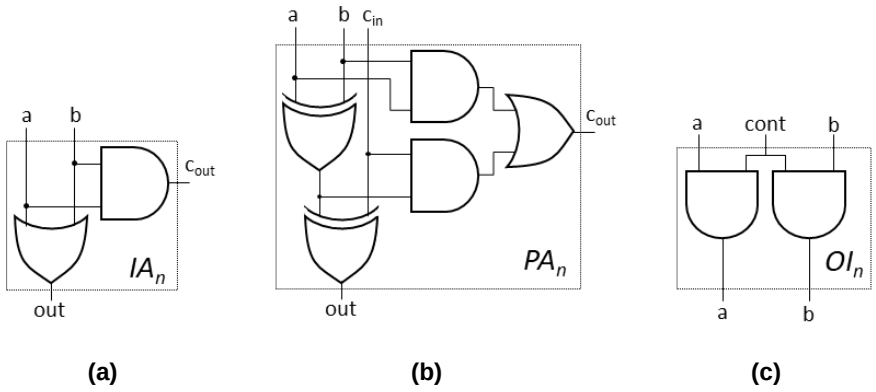


Figura 32 – Estruturas do 2PSA: (a) IA_n, (b) PA_n e (c) OI_n.

6.2 Avaliações de Qualidade e Potência para o Operador 2PSA

A avaliação de qualidade do 2PSA foi feita considerando quatro tamanhos de operador: 8, 16, 32 e 64 bits. Inicialmente, os operadores 2PSA foram descritos em linguagem C considerando todos os pontos de operação. Cada operador foi avaliado com 250.000 operações, com amostras extraídas de uma aplicação real (codificação de vídeos segundo o padrão HEVC (HEVC, 2015)). Os resultados obtidos foram comparados com os resultados de um somador preciso em termos da relação sinal-ruído (SNR), conforme definido na Equação 3.

Na Equação 3, $Média_n$ representa o valor médio da saída precisa avaliada para cada 2PSA de *n* bits e *MSE* é o erro quadrático médio para esse mesmo operador 2PSA. O MSE é calculado de acordo com o que é definido pela Equação 4. Na Equação 4, *k* é o número de operações avaliadas, P_i refere-se ao resultado preciso para a posição *i* e I_i é o resultado impreciso nessa mesma posição *i*.

$$SNR = 20 \times \log_{10}\left(\frac{Média_n}{\sqrt{MSE_n}}\right) \tag{3}$$

$$MSE = \frac{1}{k}\sum_{i=0}^{k-1}(P_i - I_i)^2 \tag{4}$$

As avaliações de potência foram realizadas considerando os quatro tamanhos de operadores 2PSA. Para isso, os operadores foram descritos em VHDL e sintetizados para a biblioteca de células padrão 45nm da Nangate (NANGATE, 2019) usando a ferramenta Encounter RTL Compiler, da Cadence. Os operadores foram configurados para suportar todos os pontos de operação disponíveis para cada largura de bit avaliada. As avaliações de potência consideraram as mesmas 250.000 operações mencionadas anteriormente. Foi realizada uma avaliação para cada ponto de operação de cada operador.

As Figuras 33, 34, 35 e 36 apresentam gráficos comparando as reduções em potência dissipada com a resposta SNR para cada operador 2PSA avaliado. As reduções em dissipação de potência representam a porcentagem de ganho de cada ponto de operação impreciso quando comparado ao ponto de operação preciso. Nas Figuras 33, 34, 35, e 36, os eixos horizontais apresentam os níveis de precisão, onde *P* indica um ponto de operação preciso e I_n indica um ponto de operação impreciso com *n* bits de imprecisão, como discutido anteriormente.

Como esperado, quanto maior o nível de imprecisão, maior a degradação da qualidade e maiores as reduções na dissipação de potência. Embora a variação para o operador de 8 bits não seja muito bem comportada, os operadores maiores (de 16, 32 e 64 bits) apresentam uma tendência linear para SNR e redução de dissipação de potência. Assim, os pontos de operação ideais dependem da tolerância à imprecisão ou dos requisitos da aplicação. Tomando o somador de 32 bits como exemplo, se a

aplicação exigir um SNR de 150dB, o ponto de operação I_8 pode ser empregado, levando a uma redução de 16% na dissipação de potência. No entanto, se 40dB é uma qualidade tolerável, o ponto de operação I_{27} pode ser usado, resultando em 57% de redução na dissipação de potência quando comparado à operação precisa. É importante destacar que esses números consideram uma implementação do 2PSA com suporte a todos os pontos de operação possíveis.

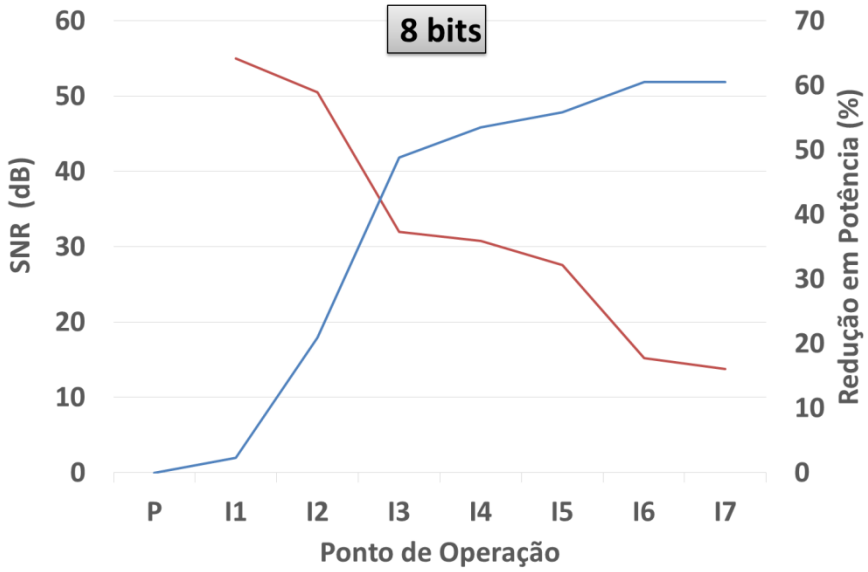


Figura 33 – Avaliações de qualidade (vermelho) e potência (azul) para um operador 2PSA de 8 bits.

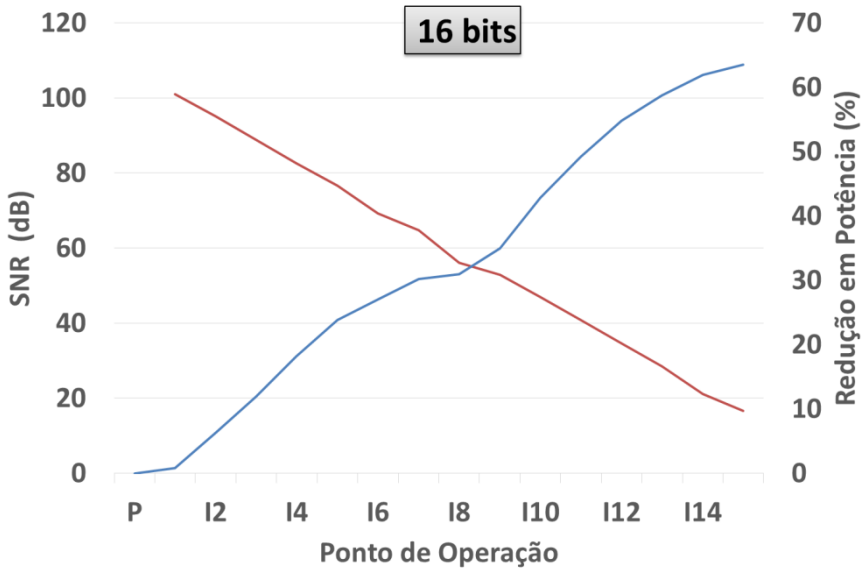


Figura 34 – Avaliações de qualidade (vermelho) e potência (azul) para um operador 2PSA de 16 bits.

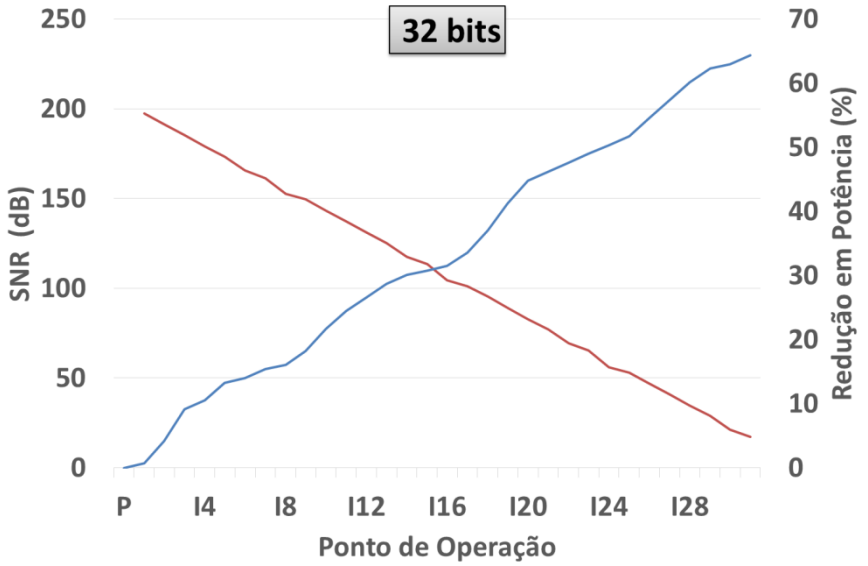


Figura 35 – Avaliações de qualidade (vermelho) e potência (azul) para um operador 2PSA de 32 bits.

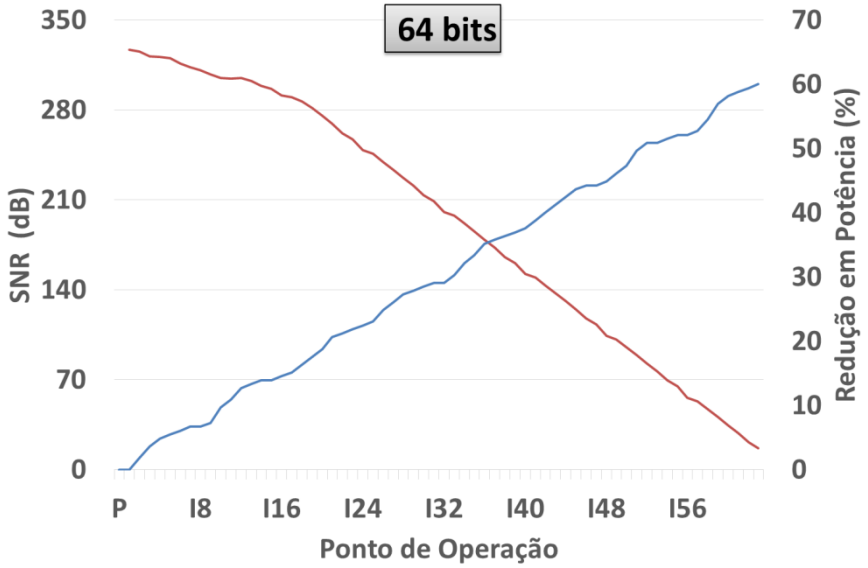


Figura 36 – Avaliações de qualidade (vermelho) e potência (azul) para um operador 2PSA de 64 bits.

6.3 Resultados de Síntese e Comparação

Como destacado anteriormente, o operador 2PSA oferece enorme flexibilidade quando se trata de número de pontos de operação. Esses pontos são definidos em tempo de projeto, enquanto que a seleção entre esses pontos de operação ocorre em tempo de execução. Quanto mais pontos de operação, maior a flexibilidade do operador. Por outro lado, o aumento no número de pontos de operação leva ao

aumento da sobrecarga de hardware, como resultado de mais e maiores estruturas de controle.

Para fornecer um exemplo prático, será apresentado um estudo de caso para quatro tamanhos de somadores (8, 16, 32 e 64 bits) com o número de pontos de operação definidos como $NPO = \log_2(n) - 1$, onde *n* representa o tamanho do somador em bits. A Tabela 17 apresenta os pontos de operação avaliados para cada largura de bit.

Tabela 17 – Pontos de operação considerados.

Largura (bits)	NPO	Pontos de Operação				
8	2	P	I4	-	-	-
16	3	P	I6	I12	-	-
32	4	P	I8	I16	I24	-
64	5	P	I12	I24	I36	I48

Além disso, os experimentos do estudo de caso também consideraram a síntese dos operadores para a biblioteca de células padrão 45nm da Nangate (NANGATE, 2019) usando a ferramenta Encounter RTL Compiler, da Cadence. Os resultados obtidos são apresentados na Tabela 18 incluindo frequência, área, potência e qualidade. Nota-se que operadores menores (8 bits, por exemplo) deixam menos espaço para melhorias, pois o ponto de operação I_4 fornece uma redução de 19% na dissipação de potência enquanto apresenta degradação expressiva (30,74dB). Assim, aumentar ainda mais a imprecisão levaria a perdas extremas em qualidade. Por sua vez, o operador 2PSA de 32 bits oferece 20% de redução na dissipação de potência com resposta de 152,46dB no ponto de operação I_8; ou 67% de redução na dissipação de potência com resposta de 56dB no ponto de operação I_{24}.

Comparando pontos de operação com qualidade semelhantes - considerando o somador de 8 bits em I_4 (30,74dB) e o somador de 16 bits em I_{12} (34,59dB) - a redução na dissipação de potência é maior no operador de 16 bits. Isso pode ser melhor entendido observando-se a relação entre *PAs* e *IAs* ativos. Enquanto o somador de 8 bits, operando em I_4, usa 4 *PAs* e 4 *IAs* (50% - 50%), o somador de 16 bits, operando em I_{12}, emprega 4 *PAs* e *12 AIs* (25% - 75%). Como os *AIs* dissipam menos potência que os *PAs*, quanto mais *AIs* ativos, maior a economia. Neste estudo de caso específico, o operador 2PSA permite escalabilidade de precisão e potência em até cinco pontos de operação.

Tabela 18 – Resultados de síntese para o operador 2PSA.

Largura (bits)	NPO	Freq. (MHz)	Área (gates)	Ponto de Operação	Potência (mW)	Redução de Potência	SNR (dB)
8	2	900	202.7	P	0.027	-	∞
				I4	0.022	19%	30.74
16	3	450	630.3	P	0.107	-	∞
				I6	0.068	36%	69.28
				I12	0.029	73%	34.59
32	4	230	1259.7	P	0.118	-	∞
				I8	0.094	20%	152.46
				I16	0.066	44%	104.30
				I24	0.039	67%	56.13
64	5	120	2513.7	P	0.136	-	∞
				I12	0.112	18%	305.04
				I24	0.093	32%	248.73
				I36	0.068	50%	179.11
				I48	0.048	65%	104.20

A ausência na literatura de outras estruturas de somadores escaláveis e que suportem vários pontos de operação dificulta a comparação direta com trabalhos publicados anteriormente. No entanto, para realizar uma comparação adequada, foram projetados somadores com os mesmos recursos do estudo de caso apresentado acima, mas de forma replicada. Por exemplo, para implementar os mesmos recursos do operador 2PSA de 16 bits (*NOP = 3*) foram utilizados um RCA de 16 bits (somador preciso), um somador LOA de 16 bits com 6 bits imprecisos (equivalente a um 2PSA de 16 bits no modo I_6) e um somador LOA de 16 bits com 12 bits imprecisos (equivalente a um 2PSA de 16 bits operando no modo I_{12}). Os modos de operação são selecionados pelo sinal *PQControl* de maneira semelhante ao funcionamento do operador 2PSA. A saída é selecionada através de multiplexadores e o isolamento de operandos é usado na entrada dos operadores para garantir a comparação.

A Figura 37 apresenta a comparação entre o operador 2PSA e uma solução utilizando os somadores replicados. Essa comparação mostra que é possível obter reduções de área entre 119,6% e 264,1% através do uso de operadores 2PSA. Em relação à redução na dissipação de potência, essa varia de 19,6% a 196,3%. Como esperado, quanto maior for o número de pontos de operação para uma mesma largura de bits, maior será a redução na dissipação de potência e na área ocupada ao comparar o operador 2PSA com implementações que usem somadores independentes suportando os mesmos pontos de operação. Por sua vez, para cada largura de bit, os pontos de operação com níveis mais altos de imprecisão apresentam maior redução de potência.

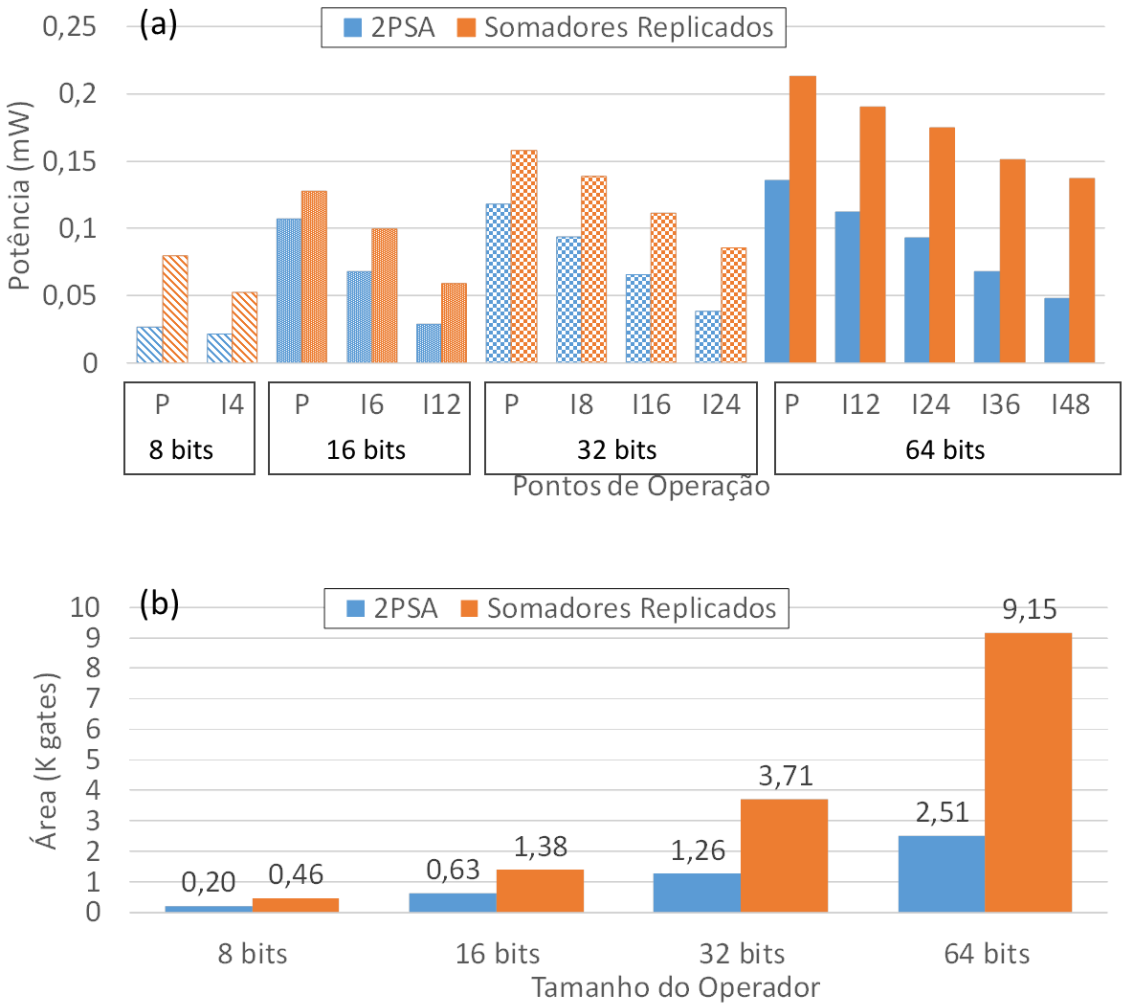


Figura 37 – Comparação entre o operador 2PSA e somadores replicados em termos de (a) potência e (b) área.

6.4 Considerações Finais sobre o Operador Aritmético Escalável em Potência e Precisão

Este capítulo apresentou um somador escalável em precisão e energia (2PSA). O 2PSA pode ser configurado para suportar uma variedade de pontos de operação de acordo com as necessidades da aplicação alvo. A seleção dinâmica dos pontos de operação permite controlar precisão e potência de acordo com o status do sistema, como tipo de fonte de alimentação ou nível de carga da bateria, ou de acordo com qualquer outra configuração do usuário. Além disso, o operador 2PSA usa isolamento de operandos para reduzir ainda mais a dissipação de potência. O operador 2PSA também foi otimizado para reduzir os recursos de hardware necessários para suportar

os vários pontos de operação. Essas otimizações também contribuem para reduzir a dissipação de potência e o atraso. Quando comparado com outras soluções capazes de suportar os mesmos níveis de imprecisão, porém sem otimizações, o operador 2PSA é capaz de alcançar ganhos de 19,6% a 196,3% em área e de 119,6% a 264,1% em potência, dependendo do tamanho do operador e do ponto de operação selecionado.

Embora o desenvolvimento do 2PSA tenha ocorrido dentro do escopo das investigações relacionadas à exploração de imprecisão na codificação de vídeos, os experimentos apresentados mostram que esse operador pode ser utilizado em qualquer tipo de aplicação que suporte computação aproximada. Ainda assim, dentro do escopo da codificação de vídeo esse operador pode ser amplamente utilizado em arquiteturas dedicadas às predições intraquadro e interquadros.

7 CONCLUSÕES

Esse trabalho buscou demonstrar a validade da tese de que é possível atingir importantes reduções na dissipação de potência com o uso de computação aproximada no contexto do projeto de hardware dedicado para a codificação de vídeo.

Os resultados apresentados demonstraram que o uso de técnicas de computação aproximada é uma abordagem extremamente importante para obter um uso eficiente de energia em projetos de hardware dedicado para a codificação de vídeos em dispositivos móveis. Como discutido, os algoritmos utilizados na codificação de vídeos são tolerantes à introdução de uma quantidade limitada de imprecisão numérica e, portanto, são aplicações promissoras para a exploração de computação aproximada.

Várias soluções de hardware com baixa dissipação de potência foram desenvolvidas e apresentadas nesta tese: uma arquitetura com foco na predição intraquadro, duas arquiteturas com foco na predição interquadros e um operador aritmético escalável em potência.

Tratando-se de soluções para a predição intraquadro, o Capítulo 4 apresentou uma arquitetura para cálculo de SAD escalável em qualidade e energia. A arquitetura apresentada é capaz de codificar vídeos na resolução UHD 8K. A escalabilidade em qualidade e energia foi obtida usando-se computação aproximada através da implementação de operadores LOA em três pontos de operação imprecisos, além de uma versão precisa com somadores RCA. A arquitetura de cálculo de SAD é capaz de processar vídeos UHD 8K a 60 quadros por segundo, rodando a 269MHz. Comparando-se essa arquitetura com uma arquitetura anterior foi possível reduzir a energia consumida em cerca de 27%, com uma degradação entre 0,28% e 1,72%, dependendo do ponto de operação impreciso escolhido. O conjunto de otimizações propostas são aplicáveis a outros algoritmos de processamento de imagens, de processamento de vídeos e de visão computacional que usam o SAD como critério de

similaridade. Outros critérios de similaridade, como SATD e SSE, também podem utilizar a solução proposta em seus cálculos.

Em relação a projetos para a predição interquadros, foram apresentadas duas soluções no Capítulo 5 desta tese.

Na primeira solução foram avaliados três níveis de imprecisão em uma arquitetura para estimação de movimento de tamanhos de bloco variáveis com baixa dissipação de potência (E-VBSME). Para obter redução na dissipação de potência, operadores LOA substituíram parte dos operadores RCA na realização dos cálculos de SAD. Impactos insignificantes na eficiência da codificação foram apresentados na avaliação feita através do software de referência do HEVC. Foram avaliados três níveis de imprecisão com o BD-Rate aumentando de 0,6% para 2,5%. Os resultados de síntese indicam economias de energia de 9,7% a 22,1%, considerando-se apenas os núcleos de processamento, e de 7% a 11,5% para a arquitetura completa da E-VSBME. Quando comparada com os trabalhos mais relevantes do estado da arte, a arquitetura de E-VBSME alcançou os melhores resultados de potência e área e a melhor relação entre potência e taxa de processamento.

A segunda solução com uso eficiente de energia para a predição interquadros é uma solução completa para estimação de movimento, suportando tanto a estimação de movimento inteira quanto a fracionária. Embora seja totalmente compatível com o padrão HEVC, essa arquitetura pode ser facilmente adaptada para codificadores atuais como AV1, AVS2 e VVC. Aproximações em nível algorítmico foram realizadas nessa arquitetura, reduzindo-se o número de tamanhos de bloco suportados. Novamente a computação aproximada foi explorada também no nível dos operadores aritméticos, com somadores imprecisos do tipo LOA sendo utilizados nos cálculos de SAD. Como resultado, a arquitetura projetada é capaz de processar vídeos UHD 8K em tempo real a uma taxa de 120 quadros por segundo. Os resultados obtidos mostraram uma dissipação de potência de 108,92 mW e área de 1.541 Kgates tendo como alvo uma taxa de processamento suficiente para processar vídeos UHD 4K a 30 quadros por segundo. A arquitetura apresentada alcançou a maior taxa de processamento máxima entre todos os trabalhos relacionados. Além disso, oferece o melhor resultado em termos de eficiência energética, mesmo quando comparado às arquiteturas que executam apenas FME ou IME.

Por fim, foi apresentado o somador 2PSA, um somador escalável em precisão e energia desenvolvido no escopo desta tese que pode ser configurado para suportar uma grande variedade de pontos de operação de acordo com as necessidades da aplicação alvo. A seleção dinâmica dos pontos de operação permite controlar precisão e potência de acordo com o status do sistema, como tipo de fonte de alimentação ou nível de carga da bateria, ou de acordo com qualquer outra configuração do usuário. Outras otimizações, como isolamento de operandos, também foram aplicadas para reduzir ainda mais a dissipação de potência. O operador 2PSA é capaz de alcançar ganhos de 19,6% a 196,3% em área e de 119,6% a 264,1% em potência, dependendo do tamanho do operador e do ponto de operação selecionado, quando comparado a soluções capazes de suportar os mesmos níveis de imprecisão usando operadores convencionais.

Considerando o conjunto de resultados apresentados nesse trabalho é possível concluir que o uso de computação aproximada é uma solução eficiente para reduzir o consumo de energia e a dissipação de potência em projetos de hardware de codificadores de vídeo. Além disso, foi possível demonstrar que é possível gerar soluções com níveis de imprecisão escalável com foco em diferentes necessidades da aplicação alvo, bem como no tipo de dispositivo que está usando a codificação e no estado de sua fonte de alimentação. Por fim, é possível concluir que a tese originalmente proposta neste trabalho se demonstrou válida a partir dos resultados apresentados.

REFERÊNCIAS

AFONSO, V.; MAICH, H.; AUDIBERT, L.; ZATT, B.; PORTO, M.; AGOSTINI, L.; SUSIN, A. Hardware Implementation for the HEVC Fractional Motion Estimation Targeting Real-Time and Low-Energy. **Journal of Integrated Circuits and Systems**, [S.l.], v. 11, n. 3, p. 106-120, 2016.

AGOSTINI, L. **Desenvolvimento de arquiteturas de alto desempenho dedicadas à compressão de vídeo segundo o padrão H.264/AVC.** 2007. 173 f. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2007.

ALVAR, S.; ABDOLLAHZADEH, M.; SEYEDARABI, H. A novel fast search motion estimation algorithm in video coding. In: IEEE INTERNATIONAL SYMPOSIUM ON INDUSTRIAL ELECTRONICS, 23., 2014, Istanbul. **Anais**... [S.l.]: IEEE, 2014, p. 934-937.

AOM – Alliance for Open Media. **Get started creating products with AV1**. Disponível em: <https://aomedia.org/av1-features/get-started/>. Acesso em: 04 dez. 2019.

CISCO SYSTEMS. **Cisco Visual Networking Index: Forecast and Methodology, 2016–2021**. Disponível em: <https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html>. Acesso em: 04 dez. 2017.

BJONTEGAARD, G. **Calculation of average PSNR differences between RD-curves**. ITU - Telecommunications Standardization Sector – Study Group 16 Question 6 - Video Coding Experts Group (VCEG), Documento VCEG-M33, 2001.

BOSSEN, F. **Common test conditions and software reference configurations.** Joint Collaborative Team on Video Coding (JCT-VC), Documento JCTVC-L1100, 2013.

BREUER, M. Multi-media applications and imprecise computation. In: IEEE EUROMICRO CONFERENCE ON DIGITAL SYSTEM DESIGN, 8., 2005, Porto. **Anais**... [S.l.]: IEEE, 2005. p. 2-7.

CAMUS, V.; SCHLACHTER, J.; ENZ, C. A low-power carry cut-back approximate adder with fixed-point implementation and floating-point precision. In: ACM/EDAC/IEEE DESIGN AUTOMATION CONFERENCE, 53., 2016, Austin. **Anais**... [S.l.]: IEEE, 2016. p. 1-6.

CAO, W.; HOU, H.; TONG, J.; LAI, J; MIN, H. A high-performance reconfigurable VLSI architecture for VBSME in H.264. **IEEE Transactions of Consumer Electronics**, [S.l.], v. 54, n. 3, p. 1338-1345, 2008.

CHIANG, J. C.; KUO, W. T.; SU, L. Y. Fast Motion Estimation using Hexagon-Based Search Pattern in Predictive Search Range. In: INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATIONS AND NETWORKS, 16., 2007, Honolulu. **Anais**… [S.l.]: IEEE, 2007, p. 1149-1153.

CHIPPA, V.; VENKATARAMANI, S.; CHAKRADHAR, S.; ROY, K.; RAGHUNATHAN, A. Approximate computing: an integrated hardware approach. In: ASILOMAR CONFERENCE ON SIGNALS, SYSTEMS AND COMPUTERS, 47., 2013, Pacific Grove. **Anais**… [S.l.]: IEEE, 2013. p. 111-117.

CHUANG, T.; TSUNG, P.; LIN, P.; CHANG, L.; MA, T.; CHEN, Y.; CHEN, L. A 59.5mW scalable/multi-view video decoder chip for Quad/3D Full HDTV and video streaming applications. In: IEEE INTERNATIONAL SOLID-STATE CIRCUITS CONFERENCE, 2010, San Francisco. **Anais**... [S.l]: IEEE, 2010. p. 330-332.

CISCO SYSTEMS. **Cisco Visual Networking Index: Forecast and Methodology, 2016–2021**. Disponível em: <https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html>. Acesso em: 04 dez. 2017.

CISCO SYSTEMS. **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.html>. Acesso em: 25 mar. 2020.

CORRÊA, M.; ZATT, B.; PORTO, M.; AGOSTINI, L. High-throughput HEVC intrapicture prediction hardware design targeting UHD 8K videos. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, 50., 2017, Baltimore. **Anais**… [S.l.]: IEEE, 2017. p. 1-4.

DUTT, S.; NANDI, S.; TRIVEDI, G. A comparative survey of approximate adders. In: IEEE INTERNATIONAL CONFERENCE RADIOELEKTRONIKA, 26., 2016, Košice. **Anais**... [S.l.]: IEEE, 2016. p. 61-65.

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, 2018.

FANG, H.; CHEN, H.; CHANG, T. Fast intra prediction algorithm and design for high efficiency video coding. In IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, 49., 2016, Montreal. **Anais**... [S.l.]: IEEE, 2016, p. 1770-1773.

FRUSTACI, F.; LANUZZA, M.; ZICARI, P.; PERRI, S.; CORSONELLO, P. Designing high-speed adders in power-constrained environments. **IEEE Transactions on Circuits and Systems II: Express Briefs**, [S.l.], v. 56, n. 2, p. 172-176, 2009.

GAO, X.; LU, W.; TAO, D.; LI, X. Image quality assessment and human visual system. **SPIE Video Communications and Imagem Processing**, [S.l], v. 7744, p 77440Z-1-77440Z-10, 2010.

GHOSH, A.; DEHURI, S. Evolutionary Algorithms for Multi-Criterion Optimization: a Survey. **International Journal of Computing and Information Sciences**, [S.l], v. 2, n. 1, p. 38-57, 2004.

GROIS, D.; MARPE, D.; MULAYOFF, A.; ITZHAKY, B.; HADAR, O. Performance comparison of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC encoders. In: IEEE PICTURE CODING SYMPOSIUM, 30., 2013. San Jose. **Anais**... [S.l.]: IEEE, 2013. Proceedings of PCS, IEEE, 2013. p. 394-397.

HAN, J.; ORSHANSKY, M. Approximate computing: an emerging paradigm for energy-efficient design. In: IEEE EUROPEAN TEST SYMPOSIUM, 18., 2013, Avignon. **Anais**… [S.l.]: IEEE, 2013. p. 1-6.

HE, Z.; YU, L.; ZHENG, X.; MA, S. AVS2-video coding standard - An application-oriented and high performance video coding standard. In: IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA AND EXPO WORKSHOPS, 2014, Chengdu. **Anais**...[S.l.]: IEEE, 2014. p. 1-5.

HE, G.; ZHOU, D.; LI, Y.; CHEN, Z.; ZHANG, T.; GOTO, S. High-Throughput Power-Efficient VLSI Architecture of Fractional Motion Estimation for Ultra-HD HEVC Video Encoding. **IEEE Transactions on Very Large Scale Integration Systems**, [S.l.], v. 23, n. 12, p. 3138-3142, 2015.

HEVC. **High Efficiency Video Coding.** Fraunhofer Heinrich Herts Institute. Disponível em:<https://hevc.hhi.fraunhofer.de/> Acesso em: 10 ago. 2015.

HM. **HEVC Reference Software**. Fraunhofer Heinrich Herts Institute. Disponível em: <https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/>. Acesso em: 10 ago. 2015.

HUANG, C.; TIKEKAR, M.; JUVEKAR, C.; SZE, V.; CHANDRAKASAN, A. A 249Mpixel/s HEVC video-decoder chip for Quad Full HD applications. In: IEEE INTERNATIONAK SOLID-STATE CIRCUITS CONFERENCE, 2013, San Francisco. **Anais**... [S.l.]: IEEE, 2013. p. 162-164.

HWANG, S.; HA, J. H.; SUNWOO, M. H. Efficient integer motion estimation algorithm using sub-sampling. In: INTERNATIONAL SOC DESIGN CONFERENCE, 6., 2009, Busan. **Anais**... [S.l.]: IEEE, 2009. p. 361-364.

ITU-T; ISO/IEC JTC 1. **Generic Coding of Moving Pictures and Associated Audio Information – Part 2: Video**, ITU-T Rec. H.262 and ISO/IEC 13818-2 (MPEG-2 Video), version 1, 1994.

JIA, L.; TSUI, C.; AU, O. C., JIA, K. A Low-Power Motion Estimation Architecture for HEVC Based on a New Sum of Absolute Difference Computation. **IEEE Transactions on Circuits and Systems for Video Technology**. [S.l.], v. 30, n. 1, p. 243-255, 2020.

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, 2015.

JU, C.; LIU, T.; CHANG, Y.; WANG, C.; LIN, H.; CHENG, C.; CHEN, C.; CHIU, M.; WANG, S.; CHAO, P.; HU, M.; YEH, F.; CHUANG, S.; LIN, H.; WU, M.; CHEN, C.; TSAI, C. A 0.2nJ/pixel 4K 60fps Main-10 HEVC decoder with multi-format capabilities for UHD-TV applications. In: IEEE EUROPEAN SOLID-STATE CIRCUITS CONFERENCE, 40., 2014, Venice. **Anais**... [S.l.]: IEEE, 2014. p. 195-198.

KAHNG, A.; KANG, S. Accuracy-configurable adder for approximate arithmetic designs. In: ACM/EDAC/IEEE DESIGN AUTOMATION CONFERENCE, 49., 2012, San Francisco. **Anais**… [S.l.]: IEEE, 2012. p. 820-825.

KANG, S.; YOO, D.; LEE, S.; KIM, Y. H. Hardware implementation of motion estimation using a sub-sampled block for frame rate up-conversion. In: INTERNATIONAL SOC DESIGN CONFERENCE, 5., 2008, Busan. **Anais**... [S.l.]: IEEE, 2008. p. II-101-II-104.

KHAN, M.; SHAFIQUE, M.; GRELLERT, M.; HENKEL, J. Hardware-software collaborative complexity reduction scheme for the emerging HEVC intra encoder. In: IEEE DESIGN, AUTOMATION AND TEST IN EUROPE, 16., 2013, Grenoble. **Anais**... [S.l.]: IEEE, 2013. p. 125-128.

KIM, S.; PARK, C.; CHUN, H.; KIM, J. A novel fast and low-complexity motion estimation for UHD HEVC. In: IEEE PICTURE CODING SYMPOSIUM, 30., 2013, San Jose. **Anais**… [S.l.]: IEEE, 2013. p. 105-108.

LAINEMA, J.; BOSSEN, F.; HAN, W.; MIN, J.; UGUR, K. Intra Coding of the HEVC Standard. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v. 22, n. 12, p. 1792-1801, 2012.

LI, P.; TANG, H. A low-power VLSI implementation for variable block size motion estimation in H.264/AVC. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, 2010, Paris. **Anais**… [S.l.]: IEEE, 2010. p. 2972-2975.

LI, F.; SHI, G.; WU, F. An efficient VLSI architecture for 4×4 intra prediction in the High Efficiency Video Coding (HEVC) standard. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING, 18., 2011, Bruxelas. **Anais**... [S.l.]: IEEE, 2011. p. 373-376.

LIU, Z.; WANG, D.; ZHU, H.; HUANG, X. 41.7BN-pixels/s reconfigurable intra prediction architecture for HEVC 2560×1600 encoder. In: IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING, 38., 2013, Vancouver. **Anais**... [S.l.]: IEEE, 2013. p. 2634-2638.

LIU, C.; SHEN, W.; MA, T.; FAN, Y.; ZENG, X. A highly pipelined VLSI architecture for all modes and block sizes intra prediction in HEVC encoder. In: IEEE INTERNATIONAL CONFERENCE ON ASIC, 2013, Shenzhen. **Anais**... [S.l.]: IEEE, 2013a. p. 1-4.

LU, W.; YU, N.; NAN, J.; WANG, D. A Hardware Structure of HEVC Intra Prediction. In: IEEE INTERNATIONAL CONFERENCE ON INFORMATION SCIENCE AND CONTROL ENGINEERING, 2., 2015, Shanghai. **Anais**...[S.l.]: IEEE, 2015, p. 555-559.

MAHDIANI, H.; AHMADI, A.; FAKHRAIE, M.; LUCAS, C. Bio-inspired imprecise computacional blocks for efficient vlsi implementation of soft-computing applications. **IEEE Transactions on Circuits and Systems – I: Regular Papers**, [S.l.], v. 57, n. 4, p. 850-862, 2010.

MANO, M.; KIME, C. **Logic and computer design fundamentals**. [S.l.]: Prentice-Hall, 2001.

MEDHAT, A.; SHALABY, A.; SAYED, M. High-throughput hardware implementation for motion estimation in HEVC encoder. In: IEEE INTERNATIONAL MIDWEST SYMPOSIUM ON CIRCUITS AND SYSTEMS, 58., 2015, Fort Collins. **Anais**… [S.l.]: IEEE, 2015. p. 1-4.

MPEG – The Moving Picture Experts Group. **Versatile Video Coding**. Disponível em: <https://mpeg.chiariglione.org/standards/mpeg-i/versatile-video-coding>. Acesso em: 4 dez. 2019.

NALLURI, P.; ALVES, L.; NAVARRO, A. High speed SAD architectures for variable block size motion estimation in HEVC video coding. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING, 21., 2014, Paris. **Anais**… [S.l.]: IEEE, 2010. p 1233-1237.

NANGATE. **NanGate FreePDK45 Open Cell Library**. Disponível em: <http://www.nangate.com>. Acesso em: 25 mar. 2019.

PALOMINO, D.; SAMPAIO, F.; AGOSTINI, L.; BAMPI, S.; SUSIN, A. A memory aware and multiplierless VLSI architecture for the complete intra prediction of the HEVC emerging standard. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING, 19., 2012, Lake Buena Vista. **Anais**... [S.l.]: IEEE, 2012. p. 201-204.

PARK, S.; CHOI, B. G., LIM, I. G., PARK, H.; KANG, S. W. An efficient motion estimation hardware architecture using Modified Reference Data Access (MRDAS) skip algorithm for high Efficiency Video Coding (HEVC) encoder. In: IEEE INTERNATIONAL CONFERENCE ON CONSUMER ELECTRONICS, 6., 2016, Berlin. **Anais**… [S.l.]: IEEE, 2016. p. 85-89.

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

PERLEBERG, M.; AFONSO, V.; CONCEIÇÃO, R.; SUSIN, A.; AGOSTINI, L.; PORTO, M.; ZATT, B. 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, p 1-12, 2018.

PIAO, Y.; MIN, J.; CHEN, J. **Encoder improvement of unified intra prediction - Document JCTVC-C207**, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11. Disponível em: <https://phenix.int-evry.fr/jct/doc_end_user/documents/3_Guangzhou/wg11/JCTVC-C207-m18245-v2-JCTVC-C207.zip>. Acesso em: 24 mar. 2017.

PORTO, R.; AGOSTINI, L.; BAMPI, S. Hardware Design of the H.264/AVC Variable Block Size Motion Estimation for Real-Time 1080HD Video Encoding. In: IEEE COMPUTER SOCIETY ANNUAL SYMPOSIUM ON VLSI, 2009, Tampa. **Anais**… [S.l]: IEEE, 2009. p. 115-120

PORTO, R.; AGOSTINI, L.; ZATT, B.; PORTO, M.; ROMA, N.; SOUSA, L. Energy-efficient motion estimation with approximate arithmetic. In: IEEE INTERNATIONAL WORKSHOP ON MULTIMEDIA SIGNAL PROCESSING, 19., 2017, Luton. **Anais**… [S.l]: IEEE, 2017. P. 1-6.

PORTO, R.; AGOSTINI, L.; ZATT, B., ROMA, N.; PORTO, M. Power-Efficient Approximate SAD Architecture with LOA Imprecise Adders. In: IEEE LATIN AMERICAN SUMPOSIUM on CIRCUITS AND SYSTEMS, 10., 2019, Armenia. **Anais**… [S.l]: IEEE, 2019. p. 65-68.

PORTO, R.; CORRÊA, M.; GOEBEL, J.; ZATT, B.; ROMA, N.; AGOSTINI, L.; PORTO, M. UHD 8K energy-quality scalable HEVC intra-prediction SAD unit hardware using optimized and configurable imprecise adders. **Journal of Real-Time Image Processing**, [Online], 2019a. DOI:10.1007/s11554-019-00934-2.

RAHA, A.; JAYAKUMAR, H.; RAGHUNATHAN, V. A power efficient video encoder using reconfigurable approximate arithmetic units. In: IEEE INTERNATIONAL CONFERENCE ON VLSI DESIGN AND INTERNATIONAL CONFERENCE ON EMBEDDED SYSTEMS, 27., 2014, Mumbai. **Anais**… [S.l.]: IEEE: 2014. p. 324-329.

RICHARDSON, I. **Video Codec Design – Developing Image and Video Compression Systems**. Chichester: John Wiley and Sons, 2002.

SCHWARZ, H.; SCHIERL, T.; MARPE, D. Block Structures and Parallelism Features in HEVC. In: SZE, V; BUDAGAVI, M.; SULLIVAN, G. (Eds.). High Efficiency Video Coding: Algorithms and Architectures. Springer, 2014. p.49-90.

SHAFIQUE, M.; AHMAD, W.; HAFIZ, R.; HENKEL, J. A low latency generic accuracy configurable adder. In: ACM/EDAC/IEEE DESIGN AUTOMATION CONFERENCE, 52., 2015, São Francisco. **Anais**… [S.l.]: IEEE, 2015. p. 1-6.

SINANGIL, M.; SZE, V.; MINHUA, Z.; CHANDRAKASAN, A. Cost and coding efficient motion estimation design considerations for High Efficiency Video Coding (HEVC) standard. **IEEE Journal of Selected Topics in Signal Processing**, [S.l.], v. 7, n. 6, p. 1017-1028, 2013.

SINGH, K.; AHAMED, S. Low power motion estimation algorithm and architecture of HEVC/H.265 for consumer applications. **IEEE Transactions on Consumer Electronics**, [S.l.], v. 64, n. 3, p. 267-275, 2018.

SULLIVAN, G.; WIEGAND, T. Rate-distortion optimization for video compression. **IEEE Signal Processing Magazine**, [S.l.], v. 15, n. 6, p. 74-90, 1998.

SULLIVAN, G.; 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, 2012.

SZE, V.; BUDAGAVI, M.; SULLIVAN, G. **High Efficiency Video Coding (HEVC) – Algorithms and Architectures**. [S.l.]: Springer, 2014.

TANG, X.; DAI, S.; CAI, C. An analysis of TZSearch algorithm in JMVC. INTERNATIONAL CONFERENCE ON GREEN CIRCUITS AND SYSTEMS, 2010, Shangai. **Anais**... [S.l.]: 2010. p. 516-520.

TSAI, C.; WANG, H.; LIU, C.; LI, Y.; LEE, C. A 446.6K-gates 0.55–1.2V H.265/HEVC decoder for next generation video applications. In: IEEE ASIAN SOLID-STATE CIRCUITS CONFERENCE, 2013, Singapore. **Anais**... [S.l.]: IEEE, 2013. p. 305-308.

TUAN, J.; CHANG, T.; JEN, C. On the Data Reuse and Memory Bandwidth Analysis for Full-Search Block-Matching VLSI Architecture. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v. 12, n. 1, p. 61-72, 2002.

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

VERMA, A.; BRISK, P.; IENNE, P. Variable latency speculative addition: a new paradigm for arithmetic circuit design. In: IEEE DESIGN, AUTOMATION AND TEST IN EUROPE CONFERENCE AND EXHIBITION, 2008, Munique. **Anais**… [S.l.]: IEEE, 2008. p. 1250-1255.

WESTE, N.; HARRIS, D. **CMOS VLSI Design: A Circuits and Systems Perspective**. [S.l.]: Addison-Wesley Publishing Company, 2010.

WIEN, M. **High Efficiency Video Coding: Coding Tools and Specification**. New York: Springer, 2014.

XIONG, J.; LI, H.; MENG, F.; WU, Q.; NGAN, K. Fast HEVC inter CU decision based on latent SAD estimation. **IEEE Transactions on Multimedia**, [S.l.], v. 17, n. 12, p. 2147-2159, 2015.

ZHOU, N.; DING, D.; YU, L. On hardware architecture and processing order of HEVC intra prediction module. In: IEEE PICTURE CODING SYMPOSIUM, 30., 2013, San Jose. **Anais**… [S.l.]: IEEE, 2013. p. 101-104.

ZHOU, J.; ZHOU, D.; SUN, H.; GOTO, S. VLSI architecture of HEVC intra prediction for 8K UHDTV applications. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING, 2014, Paris. **Anais**... [S.l.]: IEEE, 2014. p. 1273-1277.

ZHOU, D.; WANG, S.; SUN, H.; ZHOU, J.; ZHU, J.; ZHAO, Y.; ZHOU, J.; ZHANG, S.; KIMURA, S.; YOSHIMURA, T.; GOTO, S. 14.7 A 4Gpixel/s 8/10b H.265/HEVC video decoder chip for 8K Ultra HD applications. In: IEEE INTERNATIONAL SOLID-STATE CIRCUITS CONFERENCE, 2016, San Francisco. **Anais**... [S.l.]: IEEE, 2016. p. 266-268.

ZHU, N.; GOH, W.; ZHANG, W.; YEO, K.; KONG, Z. Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing. **IEEE Transactions on Very Large Scale Integration Systems**, [S.l.], v. 18, n. 8, p. 1225-1229, 2010.

ZHU, N.; GOH, W.; WANG, G.; YEO, K. Enhanced low-power high-speed adder for error-tolerant application. In: IEEE INTERNATIONAL SOC DESIGN CONFERENCE, 7., 2010, Seul. **Anais**... [S.l.]: IEEE, 2010a. p. 323-327.

APÊNDICE A: LISTA DE PUBLICAÇÕES OBTIDAS DURANTE O DOUTORADO

1. Artigos Publicados

Título: Energy-Efficient Motion Estimation with Approximate Arithmetic

Autores: Roger Porto, Luciano Agostini, Bruno Zatt, Marcelo Porto, Nuno Roma, Leonel Sousa.

Conferência: IEEE 19th International Workshop on Multimedia Signal Processing (MMSP)

Ano: 2017

DOI: 10.1109/MMSP.2017.8122248

Qualis: B1

Título: Power-Efficient Approximate SAD Architecture with LOA Imprecise Adders

Autores: Roger Porto, Luciano Agostini, Bruno Zatt, Nuno Roma, Marcelo Porto.

Conferência: IEEE 10th Latin American Symposium on Circuits and Systems (LASCAS)

Ano: 2019

DOI: 10.1109/LASCAS.2019.8667554

Qualis: B2

Título: UHD 8K Energy-Quality Scalable HEVC Intra-Prediction SAD Unit Hardware Using Optimized and Configurable Imprecise Adders

Autores: Roger Porto, Marcel Corrêa, Jones Goebel, Bruno Zatt, Nuno Roma, Luciano Agostini, Marcelo Porto.

Periódico: Journal of Real-Time Image Processing (JRTIP)

Ano: 2019

DOI: 10.1007/s11554-019-00934-2

Qualis: B1