UNIVERSIDADE CATÓLICA DE PELOTAS
CENTRO DE CIÊNCIAS SOCIAIS E TECNOLÓGICAS
MESTRADO EM ENGENHARIA ELETRÔNICA E COMPUTAÇÃO

RAFAEL DOS SANTOS FERREIRA

Arquiteturas de Hardware de Baixa Potência para Codificação de Vídeo usando Operadores Aritméticos de Codificação Híbrida

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Engenharia Eletrônica e Computação

Orientador: Prof. Dr. Cláudio Machado Diniz

Pelotas
2017

CIP — CATALOGAÇÃO NA PUBLICAÇÃO

Ferreira, Rafael dos Santos

Arquiteturas de Hardware de Baixa Potência para Codificação de Vídeo usando Operadores Aritméticos de Codificação Híbrida / Rafael dos Santos Ferreira. – Pelotas: 2017.

71 f.: il.

Dissertação (mestrado) – Universidade Católica de Pelotas. 2017. Orientador: Cláudio Machado Diniz.

1. Codificação de Vídeo. 2. HEVC. 3. Baixa Potência. 4. Arquitetura de Hardware. 5. Operadores Aritméticos. 6. Codificação Híbrida. 7. Interpolação. 8. SAD. 9. Quantização. I. Diniz, Cláudio Machado. II. Título.

“If I have seen farther than others,
it is because I stood on the shoulders of giants.”
— Sir Isaac Newton

AGRADECIMENTOS

Primeiramente, agradeço a Deus por me guiar nas estradas à noite, muitas vezes cansado, trazendo-me e levando-me com segurança aos meus destinos. Ao meu irmão Gabriel, que depositou confiança e me incentivou em fazer o mestrado. À minha amada esposa por ficar em casa cuidando de nossas filhas, enquanto eu dedicava-me aos estudos. À minha mãe, que muitas vezes me acompanhava às idas a Pelotas.Agradeço meu pai por me apoiar, apesar da minha ausência no seu momento mais importante da vida dele, obrigado pai!Agradecer ao Marcus Conde pela hospedagem e amizade. Aos professores e amigos, Cláudio e Eduardo, que mostraram serem pessoas humildes e tiveram paciência em me ensinar uma área na qual eu não tinha conhecimento. Meus colegas Anderson e Bianca, grandes parceiros de estudo; talvez sem o apoio inicial deles eu não teria a coragem de terminar o mestrado. Ao Mateus e ao Guilherme, que me ensinaram a fazer a síntese na Cadence. E, finalmente, ao Rubimar por me emprestar as chaves da sala de aula e, muitas vezes, eu esquecer de devolvê-las e trazer para Bagé. Além de todos que me ajudaram nesta empreitada, que não citei aqui, cada um com sua especial importância, tanto em minha vida quanto nessa jornada. Jornada, essa, que por muitas vezes tornou-se árdua, entretanto, graças a todos os envolvidos, hoje dou um passo importante e derradeiro em minha formação pessoal e profissional.

RESUMO

A codificação de vídeo é uma das áreas que está em grande expansão. Cada vez mais empresas estão investindo nesta área. A transmissão e o armazenamento de vídeos na forma bruta é custosa e muitas vezes impraticável, como no caso de vídeos de definição ultra alta (*Ultra High Definition* - UHD). Com este objetivo surgiram os codificadores de vídeo e os padrões de codificação de vídeo, tal como o HEVC, foco deste trabalho. Com o HEVC é possível comprimir um vídeo com aproximadamente metade do número de bits que o seu antecessor, o H.264/AVC, mantendo praticamente as mesmas características de qualidade do vídeo original. Desta forma, o desenvolvimento de circuitos integrados específicos para processamento de vídeo é uma atividade importante na área de pesquisa de sistemas digitais, uma vez que soluções em *software* geralmente não atingem desempenho e eficiência energética necessários para diversas aplicações, em especial para dispositivos móveis. Motivado pela necessidade de baixo consumo energético, este trabalho aplica o conceito de codificação híbrida, que tem por finalidade dividir os operandos em grupos de m bits, codificando cada grupo, utilizando o código *Gray* e, ainda, utilizando o comportamento do código binário para propagar o *carry* entre os grupos. Assim, o número de transições em cada grupo pode ser reduzido e uma estrutura regular pode ser obtida, onde os grupos menos significativos do resultado dependem somente dos grupos menos significativos dos operadores, reduzindo assim o número de transições entre bits. A proposta deste trabalho é a implementação de arquiteturas de hardware para módulos do padrão de codificação de vídeo HEVC utilizando operadores aritméticos de codificação híbrida, visando o baixo consumo energético. O estudo explora a viabilidade do uso da codificação híbrida na codificação de vídeo, e a quantificação do ganho em potência e energia de tais operadores. O trabalho também procura identificar quais módulos do HEVC são mais adequados para o emprego de tais operadores, visando maiores reduções no consumo de energia. Foram desenvolvidas arquiteturas de hardware para os módulos de interpolação (para estimação de movimento fracionário), para o cálculo do SAD – Soma das Diferenças Absolutas e para a Quantização. Além disso, o trabalho propõe dois novos somadores híbridos e seu uso em arquiteturas de módulos de codificação de vídeo. Resultados mostram redução de potência das arquiteturas usando os operadores aritméticos de codificação híbrida, quando comparado a mesma arquitetura usando operadores aritméticos convencionais, com codificação binária.

Palavras-chave: Codificação de Vídeo. HEVC. Baixa Potência. Arquitetura de Hardware. Operadores Aritméticos. Codificação Híbrida. Interpolação. SAD. Quantização.

Low Power Hardware Architectures for Video Coding using Hybrid Encoding Arithmetic Operators

ABSTRACT

Video coding is one of the rapidly expanding areas. More and more companies are investing in this area. The transmission and storage of videos in raw form is costly and often impractical, as in the case of ultra high definition videos (UHD). With this goal the video encoders and video coding standards arise, such as the HEVC, focus of this work. With the HEVC it is possible to compress a video with approximately half the number of bits that its predecessor, the H.264/AVC, maintaining practically the same quality characteristics of the original video. In this way, the development of specific integrated circuits for video processing is an important activity in the area of digital systems research, since software solutions generally do not achieve the performance and energy efficiency necessary for several applications, especially for mobile devices. Motivated by the need for low power consumption, this work applies the Hybrid coding concept, whose purpose is to divide the operands into groups of m bits, coding each group using the Gray code, and using the behavior of the binary code to propagate the carry between the groups. Thus, the number of transitions in each group can be reduced and a regular structure can be obtained, where the least significant groups of the result depend only on the least significant groups of operators, thus reducing the number of transitions between bits.

The goal of this work is the implementation of hardware architectures for modules of the HEVC video coding standard using arithmetic operators of hybrid coding, aiming the low energy consumption. The study explores the feasibility of using hybrid coding in video coding, and quantifying the gain in power and energy of such operators. The work seeks to identify which HEVC modules are most suitable for the employment of such operators, aiming for greater reductions in energy consumption. Hardware architectures for the Interpolation (for fractional motion estimation), the calculation of SAD - Sum of Absolute Differences, and for Quantization, were developed. In addition, the work proposes two new hybrid adders, and their use in video coding module architectures. Results show a power reduction of the architectures using hybrid encoding arithmetic operators, when compared to the same architecture using conventional arithmetic operators, with binary coding.

Keywords: Video Coding, HEVC, Low Power, Hardware Architecture, Arithmetic Operators, Hybrid Encoding, Interpolation, SAD, Quantization.

LISTA DE ABREVIATURAS E SIGLAS

ASIC *Application Specific Integrated Circuits*

AVC *Advanced Video Coding*

CB Blocos de Codificação

CfE *Call for Evidences*

CTB *Coding Tree Block*

CTU *Coding Tree Unit*

DCT *Discrete Cosine Transform*

FIR *Finite Impulse Response*

FPGA *Field Programmable Gate Array*

FPS *Frames Per Second*

HEVC *High Efficiency Video Coding*

HVS *Human Visual System*

IEC *International Electrotechnical Commission*

IEEE Instituto de Engenheiros Eletricistas e Eletronicos

ISO *International Organization for Standardization*

ITU-T *International Telecommunication Union - Telecommunication*

JCT-VC *Joint Collaborative Team on Video Coding*

MC *Motion Compensation*

ME *Motion Estimation*

MPEG *Moving Picture Experts Group*

PB *Prediction Block*

PU *Prediction Unit*

QP *Quantization Parameter*

RGB *Red, Green and Blue*

TU *Transform Unit*

YCbCr *Luminance, Chrominance blue, Chrominance Red*

WQXGA*Wide Quad eXtended Graphics Array*

WVGA *Wide Video Graphics Array*

XGA *eXtended Graphics Array*

LISTA DE FIGURAS

Figura 2.1 Espaço cores. Fonte: [Ferreira 2016] 17
Figura 2.2 Subamostragem de cor. Fonte: [Color Model Color Sub-Sampling 2012] 17
Figura 2.3 Redundância Temporal [Migliari 2015] 19
Figura 2.4 Diagrama de Blocos do Codificador de vídeo HEVC [Diniz 2015]..... 21
Figura 2.5 Blocos de Predição [Correa 2014]..... 23
Figura 2.6 Estrutura de Árvore [Correa 2014]..... 24
Figura 2.7 Particionamento de blocos usando a árvore quadtree [Correa 2014] 24
Figura 2.8 Divisão de Blocos [Correa 2014] 25
Figura 2.9 Estimação de Movimento [Afonso 2013]..... 27
Figura 2.10 Vetor de movimento fracionário [Budagavi and Sullivan 2014] 29
Figura 2.11 Representação da Configuração All Intra (AI) [Budagavi and Sullivan 2014] 34
Figura 2.12 Representação da Configuração Random Access (RA) [Budagavi and Sullivan 2014] 34
Figura 2.13 Representação do Esquema Low-Delay [Budagavi and Sullivan 2014] 35

Figura 3.1 Conversão de Binário para Hibrido m=2..... 37
Figura 3.2 Conversão de Binário para Hibrido m=2..... 38
Figura 3.3 Conversão de Binário para Hibrido m=3..... 39
Figura 3.4 Conversão de Binário para Hibrido m=4..... 39
Figura 3.5 Conversão de Binário para Hibrido m=5..... 40
Figura 3.6 Somador Híbrido Original de 2 bits [Costa 2002]..... 42
Figura 3.7 Estrutura em portas lógicas do Somador Híbrido Original de 2 bits [Costa 2002].43
Figura 3.8 Estrutura em portas lógicas do somador híbrido A de 2 bits..... 43
Figura 3.9 Estrutura em portas lógicas do somador híbrido B de 2 bits. 44
Figura 3.10 Exemplo de multiplicação numérica de 4 bits na base 4 (m=2) na codificação híbrida. 44
Figura 3.11 Estrutura Multiplicador Hibrido 45

Figura 4.1 Diagrama da arquitetura dos filtros de interpolação: (a) Usando operadores aritméticos de codificação binária; (b) Usando operadores aritméticos de codificação híbrida..... 47
Figura 4.2 Diagrama da FSM da arquitetura do filtro luma..... 48
Figura 4.3 Diagrama da FSM da arquitetura do filtro croma..... 49
Figura 4.4 Arquitetura original de SAD..... 50
Figura 4.5 Arquitetura de SAD utilizando somadores hibridos..... 51
Figura 4.6 Diagrama da arquitetura unificada de quantização direta e inversa..... 52

Figura 5.1 Metodologia para obtenção dos resultados de potência. 53

LISTA DE TABELAS

Tabela 2.1 Coeficientes dos Filtros de Luminância 30
Tabela 2.2 Coeficientes dos Filtros de Crominância 30
Tabela 2.3 Relação entre QP e Qstep 31
Tabela 2.4 Definição de sampleType - Tipo de amostra 33
Tabela 2.5 Sequências de vídeo para testes 35

Tabela 3.1 Representação Numérica 36
Tabela 3.2 Número de transições para código Binário, Híbrido (m=2) e Gray 37
Tabela 3.3 Soma de 2 bits na representação decimal,binária e híbrida. 41

Tabela 5.1 Resultados de potência e energia por operação da arquitetura do filtro de interpolação de luma. 56
Tabela 5.2 Resultados de potência e energia por operação da arquitetura do filtro de interpolação de croma 57
Tabela 5.3 Resultados de área da arquitetura do filtro luma 57
Tabela 5.4 Resultado de área da arquitetura do filtro croma 58
Tabela 5.5 Resultados de potência (em μW) da arquitetura de SAD. 59
Tabela 5.6 Resultados de área da aquitetura de SAD. 60
Tabela 5.7 Resultado de área da arquitetura de quantização. 61
Tabela 5.8 Resultado de potência (em μW) da arquitetura de quantização. 63

Tabela A.1 Representação dos coeficientes do filtro Chroma em binário e híbrido 70
Tabela A.2 Representação dos coeficientes do filtro Luma em binário e híbrido 71

SUMÁRIO

1 INTRODUÇÃO 13
1.1 Objetivos 14
1.2 Principais contribuições 14
1.3 Organização do texto 15
2 CONCEITOS BÁSICOS DE VíDEO DIGITAL E DE CODIFICAÇÃO DE VíDEO ... 16
2.1 Representação do vídeo digital 16
2.2 Espaços de cores e subamostragem 16
2.3 Qualidade de vídeo 18
2.4 Codificação de vídeo 18
2.5 *High Efficiency Video Coding* - HEVC 20
2.5.1 O Codificador de vídeo HEVC 21
2.5.2 Particionamento de blocos 23
2.6 Predição 25
2.6.1 Estimação de Movimento 26
2.6.2 Métricas de Similaridade 27
2.6.3 Estimação de Movimento Fracionário 29
2.6.4 Filtros de Interpolação 30
2.6.5 Quantização 31
2.7 Codificador de referência do HEVC (*HEVC Test Model* - HM) 33
2.8 Condições e Configurações de Teste Utilizadas nas Avaliações 33
2.9 Resumo do capítulo 35
3 OPERADORES ARITMÉTICOS DE CODIFICAÇÃO HíBRIDA 36
3.1 Codificação híbrida 36
3.2 Variação do m 38
3.3 Somadores Híbridos 39
3.3.1 Somador Original 40
3.3.2 Somador Proposto A 41
3.3.3 Somador Proposto B 41
3.4 Multiplicador Híbrido em Complemento de 2 42
3.5 Resumo do capítulo 45
4 ARQUITETURAS DESENVOLVIDAS 46
4.1 Arquitetura para o Filtro de Interpolação 46
4.2 Arquitetura para a Soma das Diferenças Absolutas 49
4.3 Quantização 51
5 IMPLEMENTAÇÃO E METODOLOGIA PARA OBTENÇÃO DOS RESULTADOS 53
5.1 Resultados e Discussões 54
5.1.1 Resultados da Arquitetura para o Filtro de Interpolação 54
5.1.1.1 Trabalhos Relacionados 54
5.1.1.2 Resultados de Síntese e Discussões 55
5.1.2 Resultados da Arquitetura para a Soma das Diferenças Absolutas 58
5.1.2.1 Trabalhos Relacionados 58
5.1.2.2 Resultados de Síntese e Discussões 58
5.1.3 Resultados da Arquitetura para a Quantização 60
5.1.3.1 Trabalhos Relacionados 60
5.1.3.2 Resultados de síntese e discussões 61
5.2 Resumo do capítulo 64
6 CONCLUSÃO 65
6.1 Trabalhos Futuros 65

6.2 Aceitação de artigo científico no tema do trabalho 66
REFERÊNCIAS 67
ANEXO A — TABELA DE COEFICIENTES 70

1 INTRODUÇÃO

A codificação de vídeo é uma área que está em grande expansão nos últimos tempos. Grandes empresas de mídia estão cada vez mais investindo em conteúdo de vídeo *on-line*. Um exemplo de sucesso a ser citado é o *Netflix*, empresa de vídeos sob demanda, que já ultrapassa o total de 65 milhões de assinantes mundiais [Paulo 2016]. A demanda por vídeos digitais tem aumentado drasticamente, chegando a bilhões de pessoas trafegando informação de vídeo ao redor do mundo simultaneamente.

A transmissão e o armazenamento de vídeos na forma bruta é custosa ou muitas vezes impraticável, como no caso de vídeos de definição ultra alta (*Ultra High Definition - UHD*). As limitações tecnológicas são o grande desafio para o compartilhamento de vídeo na internet. Pode-se citar três grandes fatores que restringem a distribuição de vídeos: limite de banda (tanto fixa quanto móvel), espaço de armazenamento e consumo energético (dispositivos móveis). Como forma de atender estas necessidades, os padrões de compressão de vídeos são constantemente aprimorados.

Um dos mais recentes e o mais eficiente padrão de codificação de vídeo é o HEVC (*High Efficiency Video Coding*) [ITU 2013]. Com o uso do HEVC é possível comprimir um vídeo com aproximadamente metade do número de bits que o seu antecessor, o H.264/AVC (*Advanced Video Coding*), mantendo praticamente as mesmas características de qualidade do vídeo original [Sullivan et al. 2012]. A codificação de vídeo faz uso de diversas ferramentas para obtenção do seu objetivo; um desses mecanismos, que pode ser destacado, é o uso de transformadas e filtros digitais. O padrão HEVC foi projetado para atender a todas as aplicações existentes atendidas pelo H.264/AVC e, principalmente, satisfazer dois requisitos específicos: a codificação de vídeos com alta resolução e o uso de arquiteturas de processamento paralelo [Budagavi and Sullivan 2014]. Para obter uma alta taxa de compressão, o HEVC emprega várias técnicas novas e outras técnicas que já existiam no padrão H.264/AVC e que foram aperfeiçoadas.

A alta compressão proporcionada pelo padrão HEVC resulta em um aumento do esforço computacional do codificador HEVC de até 3,2 vezes o esforço computacional de um codificador H.264/AVC [Vanne et al. 2012]. Além disso, é necessário um projeto de sistemas de codificação de vídeo com alta eficiência energética para lidar com o fato de a maioria dos sistemas atuais que manipulam vídeo serem operados por baterias. Diante do aumento do esforço computacional do codificador HEVC e da necessidade crescente de eficiência energética, é essencial a proposta de arquiteturas de hardware para módulos do codificador HEVC.

Neste contexto, é possível observar que os módulos de hardware do padrão HEVC são

ricos em operadores aritméticos [Diniz 2015]. Aliar o projeto de operadores aritméticos de baixa potência com o projeto de arquiteturas de hardware para módulos do codificador HEVC pode resultar em reduções de potência e consumo de energia da aplicação de codificação de vídeo. O trabalho de [Costa 2002] introduz uma nova codificação para operandos, chamada codificação híbrida, que combina a codificação *Gray* com a codificação binária para prover um compromisso entre estas duas codificações em termos de atividade de chaveamento e desempenho. Um multiplicador que opera em codificação híbrida, proposto em [Costa, Monteiro and Bampi 2007], reduz a dissipação de potência quando comparado a um multiplicador estado da arte.

O uso de operadores aritméticos de codificação híbrida em arquiteturas de codificação de vídeo é um assunto ainda não explorado. Até onde se sabe, não existem outros trabalhos que utilizam e quantificam o uso de tais operadores em arquiteturas de codificação de vídeo.

1.1 Objetivos

O objetivo geral deste trabalho é a implementação de arquiteturas de hardware para módulos do padrão de codificação de vídeo HEVC utilizando operadores aritméticos de codificação híbrida, visando o baixo consumo energético. O estudo busca explorar a viabilidade do uso da codificação híbrida na codificação de vídeo, e a quantificação do ganho em potência e energia de tais operadores. O trabalho busca identificar quais módulos do HEVC são mais adequados para o emprego de tais operadores, visando maiores reduções no consumo de energia. Foram investigados os módulos de Interpolação (para estimação de movimento fracionária) filtro de interpolação luma e croma, o módulo de Soma das Diferenças Absolutas (do inglês, *Sum of Absolute Differences*- SAD) e o módulo de Quantização. Além disso, o trabalho propôs dois novos somadores híbridos e seu uso em arquiteturas de módulos de codificação de vídeo.

1.2 Principais contribuições

As principais contribuições deste trabalho estão voltadas ao desenvolvimento de novos somadores de codificação híbrida e à exploração do uso de operadores aritméticos de codificação híbrida em diferentes arquiteturas de hardware de módulos do codificador HEVC, como listadas abaixo:

- Desenvolvimento de dois novos somadores híbridos. Esta contribuição pode ser encon-

trada na seção 3.3.

- A implementação de uma arquitetura sequencial para o filtro de interpolação de *pixel* fracionário do HEVC utilizando o multiplicador híbrido proposto em [Costa, Monteiro and Bampi 2007]. Esta arquitetura resulta em redução de potência se comparada a uma arquitetura utilizando operadores aritméticos convencionais. Esta contribuição está descrita com detalhes na seção 4.1.
- Desenvolvimento de uma arquitetura de SAD utilizando operadores híbridos. Foi aplicada na arquitetura do SAD um dos somadores híbridos propostos neste trabalho, o que resultou em uma arquitetura com redução de potência e aumento da frequência de operação quando comparado com arquitetura usando operadores aritméticos binários. Esta contribuição está detalhada na seção 4.2.
- Desenvolvimento de uma arquitetura para o módulo de Quantização aplicando os dois novos somadores híbridos propostos neste trabalho. Obteve-se com isto uma redução de potência e aumento do desempenho se comparado à arquitetura usando operadores binários. Esta contribuição está detalhada na seção 4.3.

1.3 Organização do texto

No capítulo 2 são apresentados os conceitos básicos sobre vídeo digital e sobre o novo padrão de codificação de vídeo HEVC. O capítulo 3 apresenta os conceitos básicos sobre codificação híbrida e o projeto de operadores aritméticos de codificação híbrida. Este capítulo também apresenta os dois novos somadores híbridos propostos neste trabalho. O capítulo 4 apresenta as arquiteturas de hardware de módulos do HEVC usando os operadores aritméticos de codificação hibrida. O capítulo 6 conclui este trabalho e apresenta sugestões de trabalhos futuros.

2 CONCEITOS BÁSICOS DE VÍDEO DIGITAL E DE CODIFICAÇÃO DE VÍDEO

Este capítulo faz um breve resumo sobre a codificação de vídeo e apresenta alguns recursos do novo padrão HEVC.

2.1 Representação do vídeo digital

Um vídeo é composto pela projeção de diversas imagens estáticas (quadros) em um curto intervalo de tempo, gerando assim a sensação de movimento para o observador. A taxa de quadros representa a velocidade adotada para projetar um vídeo. Precisamente, é a quantidade de quadros apresentados em um segundo. Portanto, a taxa de quadros define a frequência de exibição que os dispositivos devem alcançar, seguindo um padrão pré-definido, sendo esta medida em quadros por segundo (fps -*frames per second*) ou *Hertz* (Hz) [Gonzalez and Woods 2014]. O sistema visual humano requer uma alta taxa de quadros para perceber um movimento suave. Para transmissão em tempo real de vídeos *Full HD* é necessário um mínimo de 24 a 30 Hz. Em vídeos *UHD*, essa taxa é de 120 Hz [Mendel 2013]. Um quadro é uma matriz retangular de *pixels* que representa uma imagem estática do vídeo. A resolução de um vídeo é um valor que expressa o tamanho absoluto da matriz em número de *pixels* nas direções horizontal e vertical. Assim, com este valor é possível determinar também o número total de *pixels* contido em um quadro [Gonzalez and Woods 2014].

2.2 Espaços de cores e subamostragem

Um espaço de cores define os três canais de cor utilizados para representar qualquer cor de uma imagem digital. O formato RGB (do inglês, *Red, Green e Blue*, ou vermelho, verde e azul) é composto por três canais de cor, sendo que cada canal de cor representa uma informação de intensidade desta cor. A codificação de uma imagem digital utiliza normalmente 8 bits para representar cada canal de cor resultando em 24 bits por *pixel* a esta representação dá-se o nome de RGB24.Este formato é o mais utilizado para exibição de imagens e vídeos em dispositivos como televisores, monitores e câmeras. Outro espaço de cores bastante utilizado é o de YCbCr, onde Y representa a luminância, Cb a crominância azul e Cr a crominância vermelha. Este espaço de cores explora a característica fisiológica do olho humano, que possui receptores diferentes para detectar a intensidade luminosa (luminância) e a intensidade de cor

(crominância) [Gonzalez and Woods 2014]. Outra vantagem do espaço de cores YCbCr, que é explorado na codificação de vídeo, é o fato de que como o número de cones é bem menor que o número de bastonetes no olho humano, a informação de cor pode ser subamostrada, com baixa perda da qualidade visual. Além disso, em algumas etapas do processo de codificação, a predição é feita usando apenas o canal de luminância (Y), sendo que os resultados da aplicação das ferramentas de codificação sobre o canal de luminância podem ser extrapolados para os canais de crominância.

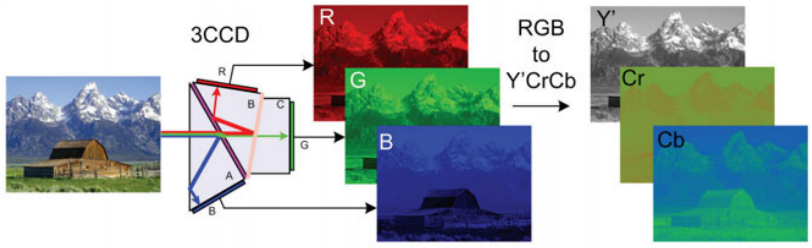

Figura 2.1: Espaço cores. Fonte: [Ferreira 2016]

A subamostragem permite a economia de 33% do número de bits comparando-se o padrão 4:2:2 e 4:4:4 e 50 % entre 4:2:0 e 4:4:4.Sendo assim este recurso é muito empregado na codificação de vídeo, elevando a taxa de compressão.

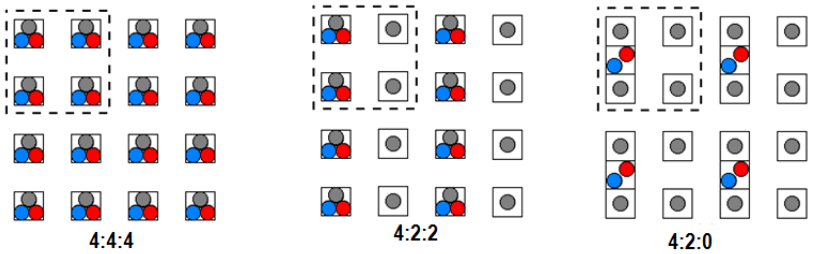

Figura 2.2: Subamostragem de cor. Fonte: [Color Model Color Sub-Sampling 2012]

2.3 Qualidade de vídeo

O PSNR (*Peak Signal Noise Ratio*) é a métrica utilizada para se medir a qualidade de uma imagem ou vídeo. Avalia a diferença entre a entrada e a saída de uma imagem ou vídeo em um processo de compressão com perdas, medindo o quanto a compressão acrescentou ruídos na imagem ou frame original. A unidade de medida adotada para esta métrica é o decibel (dB). Matematicamente, o PSNR de uma imagem de dimensão m por n é representado pela Eq. 2.1, onde o MSE é definido pela Eq. 2.2.

$$PSNR = 10\log_{10}(\frac{MAX_I^2}{MSE}) = 20\log_{10}(\frac{MAX_I}{\sqrt{MSE}}) \tag{2.1}$$

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\left\|I(_{i,j}) - k(_{i,j})\right\|^2 \tag{2.2}$$

Nas equações 2.1 e 2.2, I representa a imagem original, K a imagem resultante, M o número de linhas da imagem, N o número de colunas da imagem e Max é o valor máximo de nível de cinza de uma amostra da imagem original. Segundo [Pereira 2008] quanto maior o valor do PSNR, melhor a qualidade da imagem/vídeo após passar pelo processo de compressão.

2.4 Codificação de vídeo

Como detalhado na primeira seção deste capítulo, um vídeo digital é definido como uma sequência de imagens (ou quadros - *frames*) que são exibidos em uma determinada taxa de quadros por segundo. Cada vídeo tem uma resolução, que é a altura e largura do quadro em número de pixels. Um fluxo de vídeo é acompanhado de um fluxo de áudio e outras informações (legendas, por exemplo) sendo todas estas informações empacotadas conjuntamente para serem exibidas sincronizadamente [OliveiraJ.B 2013].

Para viabilizar o armazenamento e transmissão de vídeos digitais, é essencial a compressão de vídeo. Para entender a necessidade da compressão do vídeo podemos tomar como exemplo um vídeo não comprimido com resolução *Full HD* (1920 X 1080 *pixels*) com 8 bits para cada amostra dos 3 canais de cores, sendo exibido a 30 quadros por segundo. A taxa de bits (T), deste exemplo, é dada pela Eq. 2.3.

$$T = 1920 * 1080 * 8 * 3 * 30 = 1,49Gbps \tag{2.3}$$

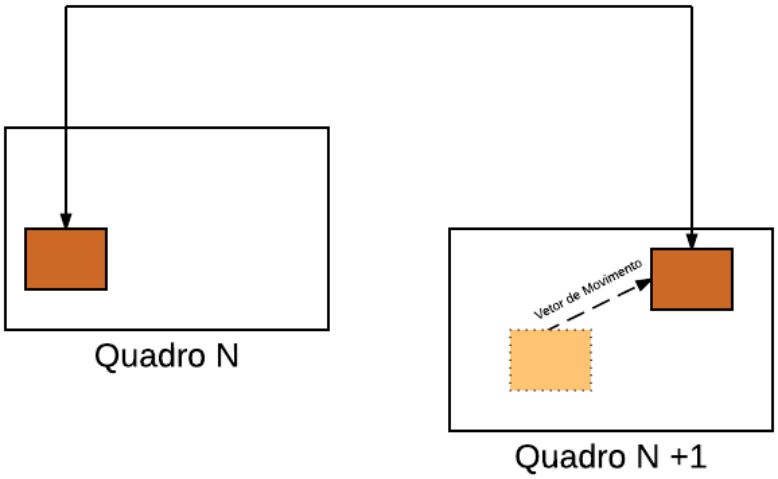

Figura 2.3: Redundância Temporal [Migliari 2015]

A compressão de vídeo visa identificar e eliminar ao máximo a redundância e o excesso de informações nos frames. Pode-se considerar como redundante qualquer informação que não contribui com novas informações relevantes para a representação da imagem/*frame*. Segundo [Richardson 2011] basicamente existem quatro tipos diferentes de redundâncias exploradas na compressão de vídeo: espacial, temporal, psico-visual e entrópica.

- **Redundância Espacial**:a redundância espacial ou frequência espacial consiste na semelhança dos pixels adjacentes de uma imagem. Um exemplo é um avião passando no céu sem nuvens, em que a informação relevante a ser transmitida é o avião, e o fundo é a parte da imagem azul cujo conteúdo de imagem é praticamente uniforme.
- **Redundância Temporal**: É baseado no aproveitamento da semelhança existente entre os quadros da sequência de vídeo. O mesmo que o exemplo anterior: a bola jogada para o céu. A diferença entre os quadros sucessivos seria somente a posição da bola a cada quadro. Dessa forma, o algoritmo enviaria a informação de um quadro completo mais o vetor de deslocamento da imagem, o qual é denominado vetor de movimento. A diferença entre o quadro N e o quadro N + 1 é o deslocamento do bloco. Assim sendo é enviado o quadro N mais a informação do vetor do movimento, evitando-se o fluxo desnecessário de informação dos quadros executados. A figura 2.3 apresenta um diagrama detalhando a redundância temporal.

- **Redundância Entrópica**: A redundância entrópica está relacionada com a forma de representação computacional dos símbolos codificados e não se relaciona diretamente ao conteúdo da imagem. A entropia é uma medida da quantidade média de informação transmitida por símbolo do vídeo. A quantidade de informação nova transmitida por um símbolo diminui na medida em que a probabilidade de ocorrência deste símbolo aumenta. Então, os codificadores que exploram a redundância entrópica têm por objetivo transmitir o máximo de informação possível por símbolo codificado e, deste modo, representar mais informações com um número menor de bits.
- **Redundância Psico-Visual**: A redundância psico-visual utiliza-se da limitação do sistema visual humano para descartar algumas informações sem afetar significativamente a percepção da qualidade visual [Carvalho 2008]. Por exemplo, o olho humano é muito mais sensível às mudanças da luminância que da crominância, assim um sistema pode eliminar algumas informações de cores sem que as pessoas percebam ou, caso isso ocorra. Existem dois tipos principais de processos utilizados para este fim. O primeiro é chamado de sub-amostragem de cor e já foi discutido neste capítulo. O segundo, conhecido por quantização, é aplicado no domínio das frequências e elimina ou atenua as frequências de menor importância para o sistema visual humano.

2.5 *High Efficiency Video Coding* - HEVC

O HEVC (*High Efficiency Video Coding*) é o mais recente padrão de codificação de vídeo, resultado de um esforço conjunto entre as organizações de padronização ITU-T VCEG (*Video Coding Experts Group*) e a ISO/IEC MPEG (*Moving Picture Experts Group*) em uma parceria conhecida como JCT-VC (*Joint Collaborative Team on Video Coding*). A primeira edição do padrão HEVC foi publicada em janeiro de 2013, em duas normas com mesmo conteúdo divulgadas pelos órgãos envolvidos [ITU 2013]. O padrão HEVC foi projetado para atender a todas as aplicações existentes atendidas pelo H.264/AVC e, principalmente, satisfazer dois objetivos:

- Suportar a codificação de vídeos de resolução mais alta, além dos já suportados pelo H.264/AVC, por exemplo, resoluções ultra altas como 3840x2160 *pixels* (4K) e 7680x4320 pixels (8K);
- Reduzir em 50% a taxa de bits do vídeo codificado, quando comparado ao padrão H.264/AVC, mantendo a mesma qualidade visual [Sullivan et al. 2012].

O objetivo do padrão HEVC era chegar as taxas de compressão duas vezes maiores que o H.264/AVC (ou seja, redução de 50% no volume de dados para representar um vídeo na mesma qualidade), entretanto, esse objetivo não é satisfeito plenamente, visto que para uma mesma qualidade visual o tamanho final do vídeo tem reduzido em torno de 40% [Oliveira 2014]. Apesar de não chegar ao objetivo original, os benefícios em qualidade subjetiva para o usuário final vão além do que as métricas objetivas indicam, especificamente o PSNR (*Peak Signal Noise Ratio*), que tem sido a métrica padrão de qualidade objetiva. Esses benefícios se tornam mais importantes para baixos *bitrates*, altas resoluções e aplicações de baixa latência.

2.5.1 O Codificador de vídeo HEVC

O codificador HEVC recebe como entrada o vídeo não comprimido e produz como saída um vídeo comprimido. Do outro lado, o decodificador recebe o vídeo comprimido e faz sua reconstrução. O par codificador/decodificador também é chamado de CODEC (do inglês, COder/DECoder) [Richardson 2010]. Um diagrama do codificador de vídeo é mostrado na Fig. 2.4.

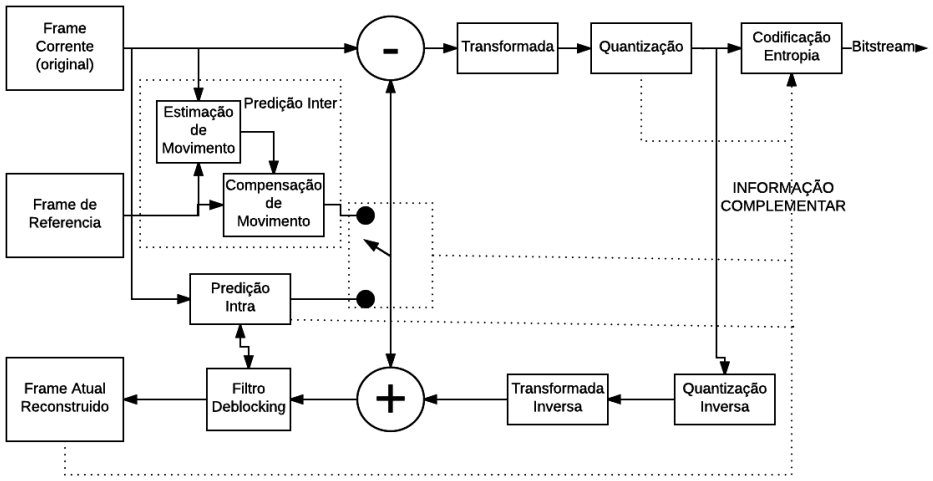

Figura 2.4: Diagrama de Blocos do Codificador de vídeo HEVC [Diniz 2015]

O codificador de vídeo é formado por 3 grandes etapas básicas: [Moreira 2015]:

1. A etapa de Predição, composta pelas etapas Predição Intra-quadro, responsável pela redução da redundância espacial, e a Predição Inter-quadros, que é encarregada pela redução da redundância temporal;

2. A etapa de Transformadas e a Quantização, que transformam os resíduos da predição (diferença entre a predição e a informação original do vídeo) para o domínio das frequências, e quantiza os coeficientes transformados, representando-os com menos *bits*. A Quantização insere perdas na informação visual. Esta perda é calculada por um parâmetro, denominado Parâmetro de Quantização (*Quantization Parameter* - QP);
3. A etapa de Codificação de Entropia, realiza uma compressão sem perdas nos coeficientes quantizados e demais informações geradas pelas etapas anteriores (modos de predição escolhidos, por exemplo).

Predição Inter-quadros é formada pelas etapas de Estimação de Movimento e Compensação de Movimento, que encontra a similaridade entre quadros vizinhos no vídeo. Já a decodificação faz o processo inverso da codificação. A partir das informações de predição e dos resíduos, a decodificação começa fazendo a decodificação de entropia e a quantização inversa que é uma operação de multiplicação sobre os resíduos para estes voltarem à amplitude do vídeo original. A transformada inversa passa estes resíduos novamente para o domínio do espaço. A partir deste ponto é possível buscar os blocos-base no quadro de referência e reconstruir o bloco a partir do bloco predito e dos resíduos, fazendo assim a compensação de movimento dos mesmos. É possível perceber que o codificador possui um caminho de decodificação interno. Isto é necessário para que o codificador use como base para a predição Inter a informação já quantizada, com perdas de informação, para que o resultado fique compatível com o processo de decodificação. Este caminho de decodificação interno ao codificador também é chamado de reconstrução. No final do processo de reconstrução, é aplicado um filtro de deblocagem (*Deblocking Filter*), que reduz efeitos de bloco inseridos por uma forte quantização. Este filtro também é incluído no final do processo de decodificação. Todo o processo de codificação é feito para cada bloco de amostras de um vídeo. A principal motivação para a utilização do particionamento baseado em blocos em codificação de vídeo ou imagens é a possibilidade de codificar cada bloco com uma configuração específica, escolhida dentro de um conjunto de parâmetros pré-definidos, considerando que, em geral, um modelo único não consegue mapear eficientemente as propriedades de uma imagem completa. A seção a seguir detalha como é feito o particionamento de blocos no padrão HEVC.

2.5.2 Particionamento de blocos

No padrão H.264/AVC, a unidade básica de codificação é o macrobloco, com 16x16 amostras de luminância e seus blocos de crominância associados (dois blocos de 8x8 pixels, quando usada a subamostragem 4:2:0). No HEVC, uma nova estrutura foi criada para o particionamento de blocos. Incialmente, o quadro é dividido em unidades de codificação chamadas *Coding Tree Units* (CTUs). Uma CTU contém em um bloco de codificação em árvore (CTB – *Coding Tree Block (CTB)* de informações de luminância e dois CTBs de crominância correspondentes. A CTB de luminância tem tamanho LxL onde L pode ser igual a 16, 32 ou 64 amostras. O tamanho do bloco é selecionado em tempo real pelo codificador dependendo do vídeo. Segundo [Correa 2014] tamanhos maiores tipicamente proporcionam melhor compressão, pois resultam em menor número de bits para representar cada bloco.

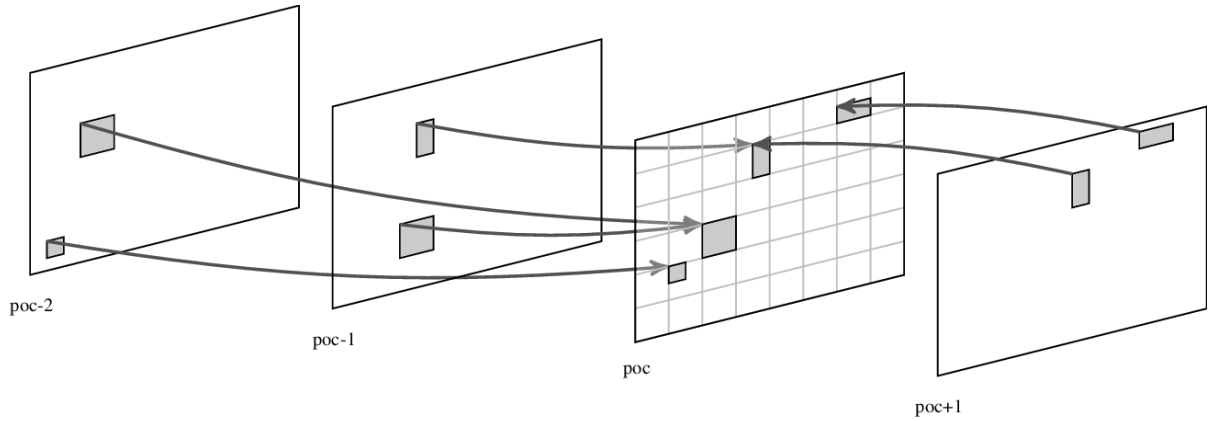

Figura 2.5: Blocos de Predição [Correa 2014]

O HEVC proporciona um particionamento dos CTBs em blocos menores utilizando uma estrutura de árvore quaternária (*quadtree*). Cada CTB é a raiz de uma árvore de codificação, que é usada para dividir a CTB em blocos de codificação (*Coding Blocks* - CBs). Por sua vez, cada CB é a raiz de uma árvore de predição e de uma árvore de transformadas. Cada árvore de predição tem apenas um nível extra de particionamento em blocos de predições (*Prediction Block* - PB). Cada CB pode ser dividido, independentemente da predição, em blocos de transformadas (*Transform Blocks* - TB) [Sullivan et al. 2012].

A sintaxe *quadtree* da CTU específica o tamanho e as posições de seus blocos de codificação de luminância e crominância. A raiz da árvore é relacionada com a CTU. Dessa forma, o tamanho da CTB de luminância é o máximo suportado por um CB de luminância. A divisão de

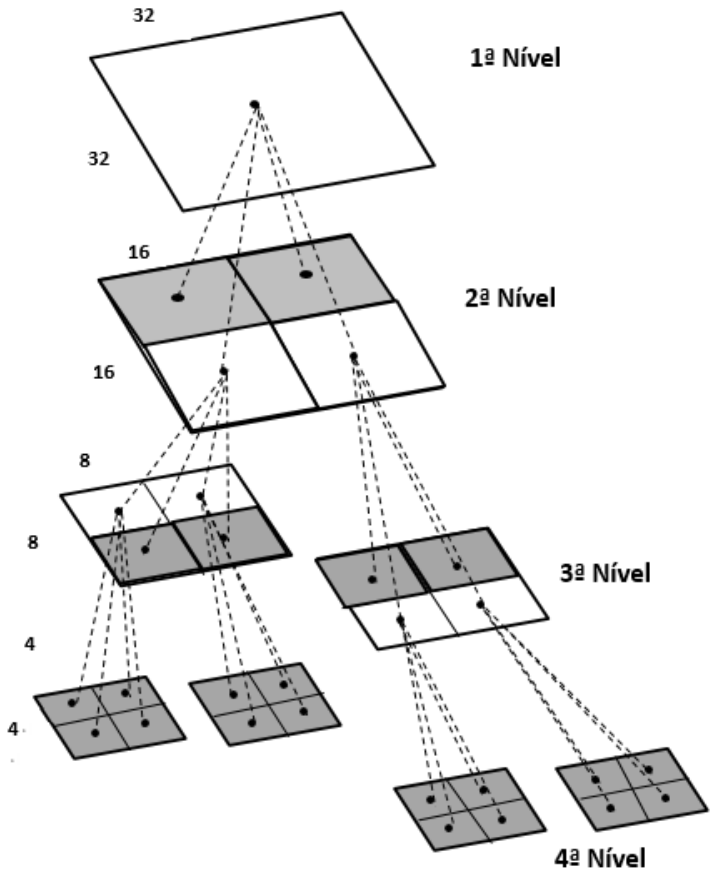

Figura 2.6: Estrutura de Árvore [Correa 2014]

uma CTU em CBs de luminância e crominância é sinalizado conjuntamente. Um CB de luminância e dois CBs de crominância em conjunto com a sintaxe associada formam uma unidade de codificação (*Coding Unit* – CU). Um CTB pode conter apenas uma CU ou ser dividido para formar CUs múltiplas, e cada CU tem um particionamento associado em unidades de predição (*Prediction Unit* – PU) e uma árvore de unidades de transformação (*Transform Unit* – TU). A Fig. 2.7 apresenta um diagrama do particionamento de unidades de codificação.

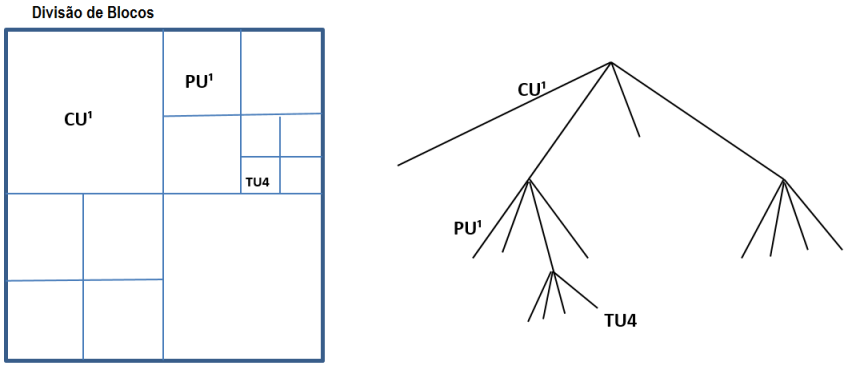

Figura 2.7: Particionamento de blocos usando a árvore quadtree [Correa 2014]

A decisão de codificar a região de uma imagem usando predição inter ou intra é feita através da unidade de predição. Dependendo da decisão de predição básica, os CBs de luminância e crominância ainda podem ser divididos em blocos de predição de luminância e crominância. O HEVC suporta tamanho de blocos de predição variando de 64x64 até 4x4.

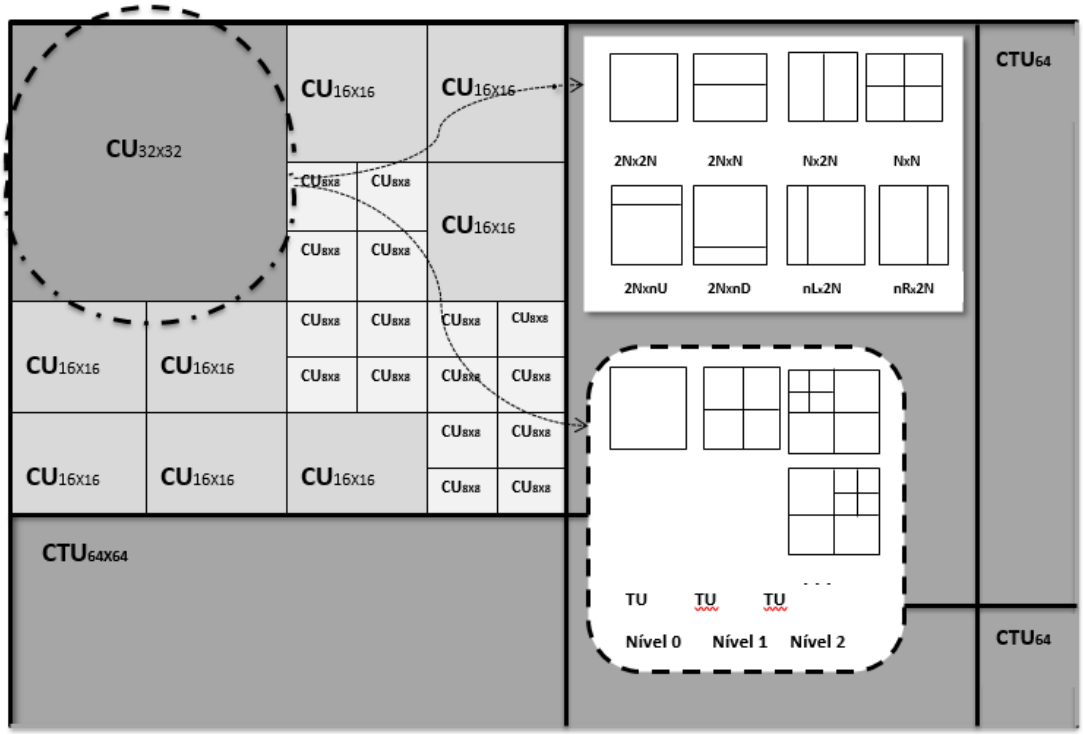

Figura 2.8: Divisão de Blocos [Correa 2014]

2.6 Predição

As etapas de predição inter-quadros e intra-quadros são aplicadas sobre blocos de *pixels* de diversos tamanhos, como mostrado na seção anterior, com o objetivo de encontrar o bloco mais semelhante possível com o original (bloco a ser codificado). O codificador, então, aplica uma série de técnicas (transformadas, quantização, etc) na diferença entre o bloco predito e o bloco original (resíduo), e envia para o decodificador o resultado. Por fim, o decodificador reconstrói o bloco predito baseado na predição escolhida que foi utilizada e somando o resíduo a este bloco, resultando no original reconstruído, servindo de referência para a predição de outros blocos.

Os tipos de redundâncias exploradas pela predição são a espacial e a temporal. A temporal existe pelo fato de haver muita semelhança entre blocos que ocupam posições próximas em dois quadros temporalmente vizinhos. Por conta disso, muitos blocos de *pixels* simplesmente

não mudam de valor de um quadro para outro em um vídeo, como por exemplo, quando um ambiente não se altera de um quadro para outro. Outros *pixels* apresentam apenas uma pequena mudança de valores, como por exemplo, quando ocorre uma variação de iluminação [Afonso 2013]. Por fim, também é possível que o bloco de *pixels* simplesmente se desloque de um quadro para outro, como por exemplo, quando um objeto se move. A exploração eficiente da redundância temporal leva ao aumento das taxas de compressão, o que é fundamental para o sucesso dos codificadores [Kalali and Hamzaoglu 2014, Lai, Chen and Huang 2010].

2.6.1 Estimação de Movimento

A estimação de movimento (ME – *Motion Estimation*) é uma parte da predição inter-quadros. Esse é o módulo do codificador de vídeo responsável por explorar e reduzir a redundância temporal.

A estimação de movimento é dividida nos seguintes passos:

1. Divisão do quadro que está sendo processado em vários blocos, considerando apenas as amostras de luminância;
2. Busca, para cada um destes blocos, pelo bloco candidato mais similar possível com o bloco atual dentro de uma área de busca de tamanho limitado em quadros vizinhos. Com a posição mais próxima do bloco original tende a se obter um melhor resultado [Kao, Kuo and Lin 2006]. A aplicação de uma área limitada permite a redução do esforço computacional da busca;
3. Geração de um vetor de movimento que indica o deslocamento do bloco atual em relação ao bloco que obteve o melhor casamento entre os blocos candidatos;
4. Subtração, *pixel* por *pixel*, do bloco original com o bloco escolhido para gerar um bloco de resíduos.

Na etapa seguinte, a Compensação de Movimento (*Motion Compensation* - MC) reconstrói o quadro, baseado nas informações dos vetores de movimento gerados pela ME. Quanto menor for o resíduo gerado na codificação, melhores serão os resultados de compressão, sendo que o resíduo ideal tem valor nulo, ou seja, representa um bloco de vídeo candidato exatamente igual ao bloco a ser codificado.

No HEVC, o processo de estimação de movimento é gerado para todos os tamanhos de blocos possíveis, mas apenas o particionamento que apresentar o melhor resultado global,

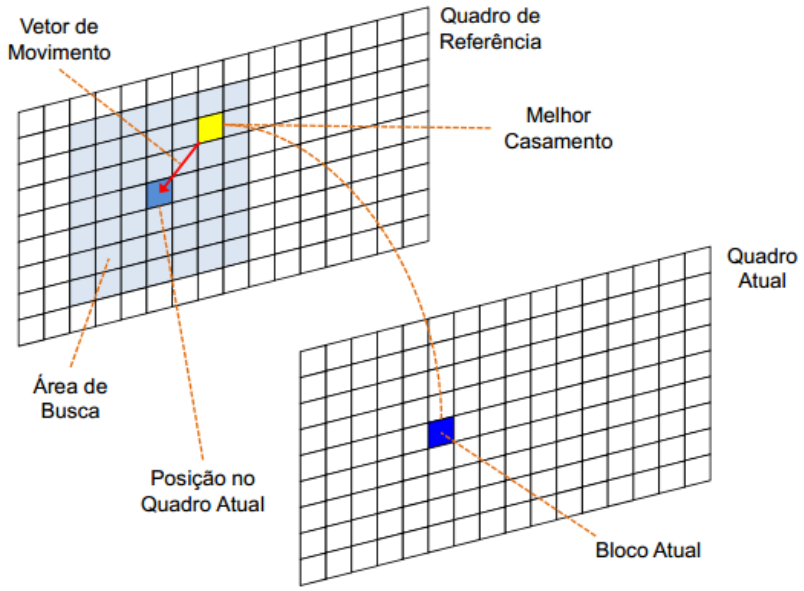

Figura 2.9: Estimação de Movimento [Afonso 2013]

avaliando a taxa de compressão e qualidade da imagem, é selecionado para a codificação. Para isto é utilizada uma técnica denominada otimização taxa-distorção (*Rate-Distortion Optimization* - RDO). Contudo, é comum ser utilizada a equação 2.4, onde J é o custo de codificação de um modo de particionamento, D é a distorção da imagem quando codificada com este modo, R é a taxa de bits por segundo do bloco codificado e *lamba* é um multiplicador de Lagrange, uma constante calculada a partir do parâmetro de quantização (*Quantization Parameter* - QP). O objetivo do RDO é escolher um modo de particionamento e uma predição que minimize o custo J.

$$J = D + \lambda \cdot R \tag{2.4}$$

A distorção é medida utilizando uma métrica de similaridade, como detalhado na próxima seção.

2.6.2 Métricas de Similaridade

As métricas de similaridade podem ser classificadas em dois tipos [Seidel et al. 2014]:

- Objetivas: são baseadas em modelos matemáticos e usadas em processamento de sinais;
- Psicovisuais: utilizam as características do HVS (Sistema Visual Humano – *Human Visual System*).

As métricas objetivas utilizam modelos matemáticos derivados da teoria de sinais e não consi-

deram as características do HVS. Enquanto que as métricas psicovisuais utilizam os modelos de HVS. As objetivas são as mais utilizadas por seu melhor desempenho e empregadas no HEVC, seguem abaixo [Seidel et al. 2014]:

- SAD – Somas das Diferenças Absolutas: esta métrica de similaridade é constituída pela soma do valor absoluto das diferenças entre cada amostra original (Ori) e a amostra de um bloco candidato a ser usado como referência (Ref), após o processo de reconstrução. Por ter uma baixa complexidade computacional é adequada para implementação direta em *hardware* e muito utilizada para determinar a distorção de blocos na estimação de movimento.

$$SAD = \sum_{i=0}^{M} \sum_{j=0}^{N} |Ori_i, j - Ref_i, j| \tag{2.5}$$

- SSD – Soma das Diferenças Quadráticas: é a soma do quadrado das diferenças entre cada amostra original e a amostra, após o processo de reconstrução. Tem como característica principal evidenciar os erros significativos. Envolve as variáveis Ori (amostra original de um bloco candidato) e Ref (amostra do quadro de referência).

$$SSD = \sum_{i=0}^{M} \sum_{j=0}^{N} (Ori_i, j - Ref_i, j)^2 \tag{2.6}$$

- SATD – Somas das Diferenças Transformadas Absolutas: é baseada na SAD, porém utiliza a transformada Hadamard (H) em uma partição de 4X4 de diferenças antes de efetuar a soma absoluta dos coeficientes. Tem maior custo que a SAD, e melhor qualidade por operar no domínio das frequências (obtendo uma aproximação da Transformada Discreta dos Cossenos – *Discrete Cosine Transform* (DCT), através da transformada Hadamand) A equação 2.7 representa a matriz de transformada de Hadamard 4X4.

$$SATD = \frac{1}{2} \sum_{i=0}^{3} \sum_{j=0}^{3} |H(Ori_i, j - Dec_i, j)H^T| \tag{2.7}$$

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \tag{2.8}$$

2.6.3 Estimação de Movimento Fracionário

Esta técnica é importante e pode ser empregada na etapa ME, tem como objetivo gerar resíduos com menos amplitude e, consequentemente, ampliar a eficiência na codificação. Funciona da seguinte forma, é realizada uma interpolação entre posições inteiras de amostras no quadro de referência, permitindo a busca em posições interpoladas de sub-pixel, além das posições inteiras de *pixel*, observa-se na figura 2.10. A FME é usada por padrões atuais, como o HEVC e o H.264/AVC. O H.264/AVC prevê a utilização de vetores de movimento com precisão de 1/2 *pixel* e 1/4 de *pixel*. No padrão HEVC, o processo de FME sofreu alterações que permitem melhorias na qualidade de codificação, porém manteve a utilização de vetores de movimento com precisão de 1/2 pixel e 1/4 de pixel.

Figura 2.10: Vetor de movimento fracionário [Budagavi and Sullivan 2014]

A técnica de FME é uma ferramenta que possibilita uma maior eficiência na codificação, pois com ela há a possibilidade de bons casamentos entre os blocos dos quadros, atual e de referência, respectivamente. Além disso, permite reduzir os resíduos e consequentemente a taxa de bits do vídeo codificado. Isto somente é possível porque os movimentos entre os quadros temporalmente vizinhos normalmente não se limitam a posições inteiras de *pixel*. A medida que a precisão aumenta na técnica de FME, os ganhos obtidos em termos de compressão acabam se tornando cada vez menos significativos. Isso pode ser explicado por decorrência da necessidade de mais vetores de movimento, além de mais bits na representação da FME [Carvalho 2008]. Nos primeiros documentos HEVC foi cogitada a utilização de precisão de 1/8 de *pixel* para as amostras de luminância, porém não foi adotado no padrão. O processo de FME é composto de duas etapas: Uma etapa de interpolação, na qual são gerados os valores das amostras em posições de sub-pixel e outra etapa de busca. Essa última onde os valores gerados são comparados com o melhor casamento em posições inteiras [Corrêa, Schoenknecht and Agostini 2010].

2.6.4 Filtros de Interpolação

A interpolação de *pixel* utiliza dois filtros principais: o de luminância (luma) e o de crominância (croma). Os dois filtros são originados a partir do filtro de Resposta ao Impulso Finita (do inglês, *Finite Impulse Response*). As amostras (*samples*) de luminância de precisão de meio e um quarto de *pixel* no HEVC são obtidas através da aplicação de filtros de 8 e 7 *taps*, respectivamente. Estes filtros são muito utilizados na codificação de vídeo, consumindo aproximadamente 25% do tempo de processamento do codificador HEVC em *software* [Diniz 2015].

No padrão HEVC é proposta a utilização de um filtro de interpolação separável de 8 *taps*, baseado em DCT (*Discrete Cosine Transform*), que interpola diretamente as amostras de luminância das posições inteiras de *pixel*, antes da busca fracionária por um melhor casamento em 1/4 de *pixel*. As amostras de crominância no HEVC são obtidas através da aplicação de filtros de 4 *taps*, e devido ao uso da subamostragem de cores 4:2:0, a precisão destas amostras é de um oitavo de *pixel* [ITU 2013]. As Tabelas 2.1 e 2.2 mostram os coeficientes dos filtros de luminância e crominância, respectivamente.

Tabela 2.1: Coeficientes dos Filtros de Luminância

Tipos de filtro	**Coeficientes dos filtros**							
1/4	-1	4	-10	58	17	5	1	
2/4	-1	4	-11	40	40	-11	4	-1
3/4	1	-5	17	58	-10	4	-1	
Índice	-3	-2	-1	0	1	2	3	4

Tabela 2.2: Coeficientes dos Filtros de Crominância

Tipos de filtro	**Coeficientes dos filtros**							
1/8			-2	58	10	-2		
2/8			-4	54	16	-2		
3/8			-6	46	28	-4		
4/8			-4	36	36	-4		
5/8			-4	28	46	-6		
6/8			-2	16	54	-4		
7/8			-2	10	58	-2		
Índice	-3	-2	-1	0	1	2	3	4

2.6.5 Quantização

A quantização direta processa o sinal de entrada restringindo o valor final em um intervalo de valores possíveis. O sinal resultante é representado com menos bits que o valor original. Na quantização inversa, o processo é inverso, reconstrói o sinal próximo ao original (valor inicial antes da quantização) no qual é feita uma aproximação não exata do valor original. Todo o processo de quantização é gerenciado pela variável QP (*Quantization Parameter*) que controla o impacto gerado pela quantização. O valor de QP é inversamente proporcional a qualidade do vídeo.

Não obstante, a quantização utilizada no HEVC é semelhante à do padrão anterior, o H.264/AVC. Nesse sentido, os cálculos realizados podem variar tanto de acordo com o tipo de amostra, luminância ou crominância, o tipo de predição quanto ao tamanho do bloco a ser quantizado. Contudo, as operações realizadas na quantização é uma multiplicação por uma constante, já a soma do resultado por outra e o deslocamento do resultado da soma e o resultado da soma é deslocado por outra constante. Essas constantes são diretamente controladas pelo QP, que possui 52 valores possíveis, podendo variar de 0 até 51. Resta salientar que para cada valor de QP existe um valor Qstep. Dentre os valores de Qstep, os primeiros seis são relativos aos seis primeiros valores de QP. E, ainda, são definidos pelo padrão conforme a tabela 2.3. Já os demais valores de Qstep são derivados dos seis primeiros, isto porque o Qstep reduz de valor pela metade a cada variação de 6 valores de QP, de modo que o mapeamento dos valores de QP para Qstep são aproximações logarítmicas [Gonçalves 2014].

Tabela 2.3: Relação entre QP e Qstep

QP	0	1	2	3	4	5	6	...	12
QStep	26214	23302	20560	18396	16384	14564	13107	...	6553

Na equação 2.9 está descrita a quantização utilizada no padrão HEVC 2.9.

$$Z = (|W| * QCoeff + offset) >> qbits \tag{2.9}$$

A equação acima representa a amostra resultante do módulo de transformadas. De todo modo, o *QCoeff* é uma constante multiplicativa que está relacionada com a predição com o QP e o tamanho do bloco quantizado. Já a constante *offset* faz a correção do arredondamento da quantização.O *qBits* define o deslocamento no final do cálculo. É importante destacar que o sinal da amostra quantizada deve ser o mesmo sinal da amostra de entrada, por isto é realizada uma operação sign que copia o sinal da entrada para saída. Desta forma, o cálculo da

quantização direta é realizado apenas considerando o módulo de W. Por fim, a constante de deslocamento *qBits* é influenciada por QP e pelo tamanho de bloco, esta constante está definida nas equações 2.10 e 2.11, em que N é o tamanho de bloco de entrada.

$$qbits = 21 + (\frac{QP}{6}) - TrSize \tag{2.10}$$

$$TrSize = log_2 N \tag{2.11}$$

Importante se faz saber que o cálculo do *offset* pode ser definido através do tipo de *slice* utilizado na codificação e está representado pela equação 2.12. *Slice* e *Tiles* são recursos utilizados para particionar os blocos para posteriormente serem processador de forma independente possibilitando um processo paralelo.

$$offset = \begin{cases} 171 & se \quad slice \quad for \quad I \\ 85 & Caso \quad contrario \end{cases}$$

$$offset = offset << (iBits - 9) \tag{2.12}$$

A constante QCoeff é definida por parâmetros acessados pela matriz de coeficientes multiplicativos *mQuantCoef*. A indexação da matriz de coeficientes é mostrada em 2.13.

$$QCoeff = mQuantCoef[TrSize2][ScalingListType][QP\%6] \tag{2.13}$$

Conforme [Gonçalves 2014], os parâmetros utilizados na Eq. 2.13 são: *TrSize* é definido na Eq. 2.11 acima. E podendo variar o bloco de entrada entre os seguintes tamanhos: 4x4, 8x8, 16x16 e 32x32. QP%6 é o resto da divisão do QP com o valor 6. *ScalingListType* (*slt*): é estabelecido pela relação do tipo de predição (inter ou intra-quadros) e a de amostra da entrada (que pode ser basicamente luminância e crominância) representado pela equação 2.14 e também pela Tabela 2.4.

$$slt = \begin{cases} 3 & se \quad INTRA \\ 0 & Caso \quad contrario \end{cases}$$

$$slt = slt + sampleType \tag{2.14}$$

Ao final é executado um clip, que limita o resultado em 16 bits (-32768 a 32767).

Tabela 2.4: Definição de sampleType - Tipo de amostra

Tipo	Luminância	Crominância(U+V)	Crominância(U)	Crominancia(V)
Valor	0	3	1	3

2.7 Codificador de referência do HEVC (*HEVC Test Model* - HM)

O codificador de referência HM tem como principal propósito prover uma implementação de referência comum de um codificador HEVC. Entretanto, deve ser útil como uma plataforma de testes, ou seja, deve-se valer para avaliação de tecnologias de codificação, além de servir como referência de desenvolvimentos independentes de codificadores e decodificadores. Para isso, deve estar em conformidade com o padrão HEVC [Sullivan et al. 2012]. O codificador HM (escrito em C++) não foi projetado visando ser um código pronto para ser utilizado em produtos de mercado. Além disso, há bastante a ser feito em termos de otimização de desempenho, pois o *software* é significativamente lento. Mesmo que a velocidade do codificador HM seja baixa para a maioria das situações, alguns aprimoramentos de tempo foram feitos durante o seu desenvolvimento. Como, por exemplo, enquanto mais de 100 horas foram necessárias para codificar um único vídeo HD de 10 segundos na primeira versão do codificador HM, após várias melhorias, chegou-se a menos de 5 horas para a mesma tarefa [Fonseca De Oliveira and Sampaio De Alencar]. Existem diversos fatores que contribuem para o *software* ser lento, entre eles destaca-se o processo de otimização taxa-distorção.

2.8 Condições e Configurações de Teste Utilizadas nas Avaliações

JCT-VC normatiza as condições de testes e configurações do HM [Bross et al. 2013], então um documento criado pelo JCT-VC requer algumas condições de testes especificadas pelo comitê. Dentre essas condições pode-se citar um conjunto de configurações para o codificador de forma a permitir a comparação justa entre codificadores. Como, por exemplo:

- **All Intra (AI):** nessa categoria todos os quadros são codificados usando predição intra, exemplificado na figura 2.11. Esta configuração é apropriada para baixo atraso e taxa de *bits* superior à aplicação.

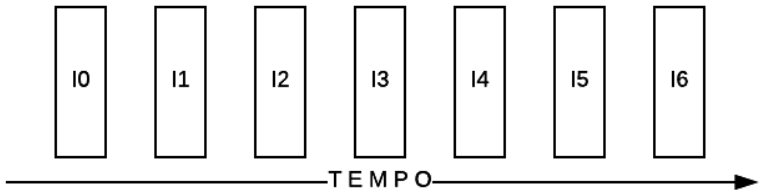

Figura 2.11: Representação da Configuração All Intra (AI) [Budagavi and Sullivan 2014]

- **Random Access (RA)**: essa configuração é usada como uma reordenação dos quadros, onde os quadros intra são inseridos periodicamente para permitir acesso aleatório a cada segundo, visto na figura 2.12. Esta configuração, por sua vez, simula o que deve acontecer em um ambiente de transmissão da televisão comum, por exemplo.

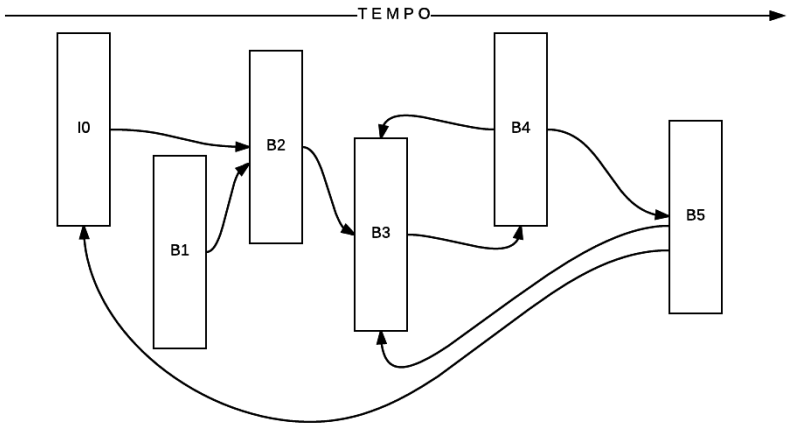

Figura 2.12: Representação da Configuração Random Access (RA) [Budagavi and Sullivan 2014]

- **Low Delay (LD)**: a partir dessa configuração nenhuma reordenação é usada. Contudo, apenas o primeiro quadro é codificado usando somente predição Intra, como mostrado na figura 2.13. Além disso, esta configuração simula o que deve acontecer em transmissões de videoconferências.

Então, um documento criado pela JCT-VC também fornece uma lista de 24 sequências de vídeo para teste, separadas em classes de acordo com suas resoluções e características. Na tabela 2.5 são mostrados 4 vídeos de resolução WQXGA (2560x1600 pixels), 5 vídeos de resolução *Full* HD (1920x1080 pixels), 5 vídeos de resolução HD (1280x720 pixels), 1 vídeo de resolução XGA (1024x768 pixels), 5 vídeos de resolução WVGA (832x480 pixels) e 4 vídeos de resolução WQVGA (416x240pixels).

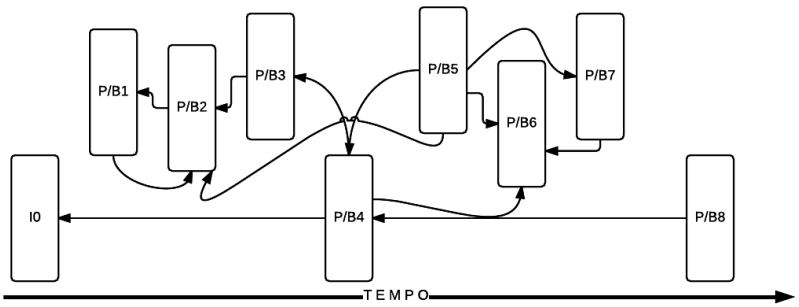

Figura 2.13: Representação do Esquema Low-Delay [Budagavi and Sullivan 2014]

Tabela 2.5: Sequências de vídeo para testes

Classificação das Sequências	Nome	Número de quadros	Quadros por segundo
WQXGA 2560 x 1600	Traffic	150	30
	PeopleOnStreet	150	30
	Nebuta	300	60
	SteamLocomotive	300	60
Full HD 1920x1080	Kimono	240	24
	ParkScene	240	24
	Cactus	500	50
	BQTerrace	600	60
	BasketballDrive	500	50
WVGA 832x480	RaceHorses	300	30
	BQMall	600	60
	PartyScene	500	50
	BasketballDrill	500	50
WQVGA 416x240	RaceHorses	300	30
	BQSquare	600	60
	BlowingBubbles	500	50
	BasketballPass	500	50
HD 1280x720	FourPeople	600	60
	Johnny	600	60
	KristenAndSara	600	60

2.9 Resumo do capítulo

Em suma, este capítulo apresentou uma revisão geral sobre os conceitos da codificação de vídeo, bem como sobre os módulos do codificador de vídeo abordados neste trabalho, sobre os quais são desenvolvidas arquiteturas aplicando os operadores de codificação híbrida. Este capítulo serve de base para o entendimento das contribuições deste trabalho, descritas nos capítulos 3 e 4.

3 OPERADORES ARITMÉTICOS DE CODIFICAÇÃO HÍBRIDA

Este capítulo apresenta os operadores aritméticos de codificação híbrida utilizados neste trabalho. Antes de detalhar os operadores aritméticos (multiplicador e somador híbridos), é apresentado o conceito de codificação híbrida, sua vantagem em relação a outras codificações e seus métodos de conversão para a codificação binária. São apresentados ainda neste capítulo os dois novos somadores propostos no contexto deste trabalho.

3.1 Codificação híbrida

A codificação híbrida foi proposta em [Costa 2002], tendo por finalidade a ideia de manipular os operandos em grupos de *m-bits* e codificar cada grupo usando o código *Gray*, podendo ser usada para operandos que operem em representação de complemento de 2. A Tabela 3.1 mostra o código híbrido em complemento de 2 para um número de 4 *bits* e m=2.

Tabela 3.1: Representação Numérica

Decimal	Binário	Híbrido	Gray
-5	1011	1110	1110
-4	1100	1000	1010
-3	1101	1001	1011
-2	1110	1011	1001
-1	1111	1010	1000
0	0000	0000	0000
1	0001	0001	0001
2	0010	0011	0011
3	0011	0010	0010
4	0100	0100	0110
5	0101	0101	0101

O código híbrido apresenta a mínima dependência das entradas de dados apresentada pelo código binário e a característica de baixa atividade de chaveamento apresentada pelo código *Gray*. A Tabela 3.2 apresenta o número de transições dos códigos binário, *Gray* e híbrido para uma determinada sequência de contagem.

Pode ser observado na Tabela 3.2, que o código híbrido apresenta um valor percentual intermediário entre os códigos binário e *Gray* em termos de número de transições. Desta forma, para sistemas onde a capacitância chaveada no barramento de dados é significativa e onde os dados apresentam um alto grau de correlação, a utilização do código híbrido pode reduzir o

Tabela 3.2: Número de transições para código Binário, Híbrido (m=2) e Gray

Num. Bits	Número de Transições			Diferença %	
	Binário	Gray	Hibrido (m=2)	Hib ->Bin	Hib ->Gray
n = 4 bits	30	16	20	-33,3%	+25%
n = 8 bits	510	256	340	-33,3%	+32,8%
n = 16 bits	131070	65536	87380	-33,3%	+33,3%

consumo de potência em até cerca de um terço em relação ao código binário [Costa 2002]. O número de transições para sequências de contagem, como apresentadas na Tabela 3.2, é calculado de acordo com as equações 3.1, 3.2, e 3.3, para os códigos binário, *Gray* e híbrido, respectivamente [Costa 2002].

$$Bin = 2^{(n+1-2)} \tag{3.1}$$

$$Gray = 2^n \tag{3.2}$$

$$Hib = 2^m \frac{2^n-1}{2^m-1} \tag{3.3}$$

Outra característica apresentada pelo código híbrido m=2 é a facilidade de mudança de representação para o código binário, como mostra o exemplo da Fig. 3.1. Desta forma, o processo de codificação/decodificação dos dados utiliza um *hardware* de reduzida complexidade com uma porta EXOR ligada a cada grupo de m=2 *bits*. Neste caso, o código híbrido também pode ser utilizado como método de codificação para os barramentos de endereços.

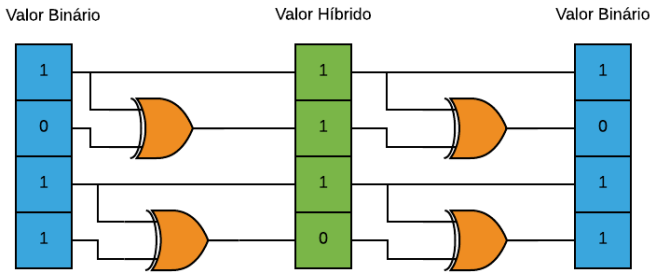

Figura 3.1: Conversão de Binário para Hibrido m=2

3.2 Variação do m

Para melhor explicação da variação do valor de m, vamos adotar a seguinte metodologia: a palavra em código binário será chamada de S, já a palavra em código híbrido será chamada de H. Toda conversão inicia-se do *bit* mais significativo para o menos significativo. O primeiro passo é agrupar a palavra em grupos de m bits. Para m=2, agrupa-se a palavra de dois em dois bits, para m=3 agrupa-se a palavra de três em três bits e assim sucessivamente. Após o agrupamento, aplica-se a operação XOR no grupo selecionado. Para uma palavra S de 3 bits em código binário (S2, S1 e S0) é feita a seguinte operação de conversão para uma palavra em código híbrido H (H2, H1 e H0) como mostrado em (3.4), (3.4), (3.6).

$$H2 = S2 \tag{3.4}$$

$$H1 = (S1)xor(S2) \tag{3.5}$$

$$H0 = (S0)xor(S1) \tag{3.6}$$

Como pode ser notado sempre existe o destaque do *bit* mais significativo na equação 3.4, passando direto para o *bit* híbrido mais significativo de destino sem aplicação da operação lógica XOR.

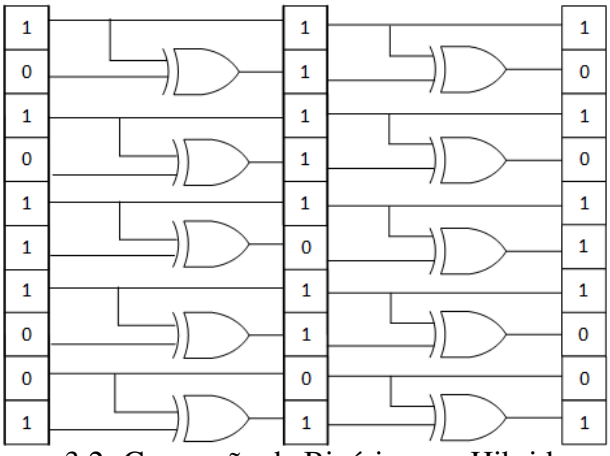

Figura 3.2: Conversão de Binário para Hibrido m=2

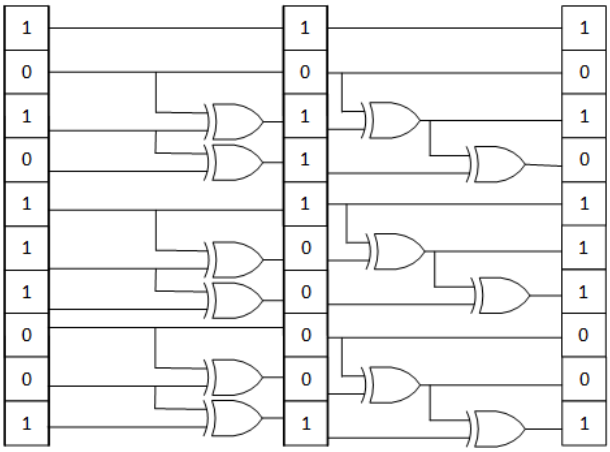

Figura 3.3: Conversão de Binário para Hibrido m=3

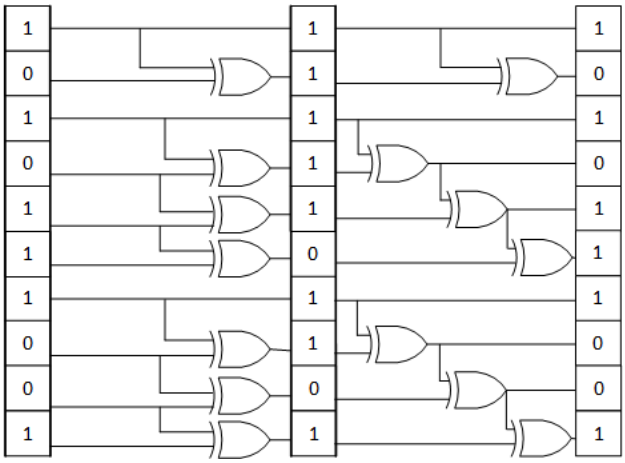

Figura 3.4: Conversão de Binário para Hibrido m=4

O agrupamento de 2 *bits* (m=2) é o único que se diferencia dos demais. Neste caso, é aplicada diretamente a operação no agrupamento de dois *bits*, não sendo necessário o destaque o *bits* mais significativo como os demais (H3 = S3).

3.3 Somadores Híbridos

Uma das principais contribuições deste trabalho é apresentada nesta seção. Dois novos somadores híbridos, neste texto referidos como Somador A e Somador B, foram propostos. Estes somadores foram aplicados posteriormente nas arquiteturas de hardware de módulos do codificador de vídeo HEVC. Para melhor entendimento dos somadores, são apresentadas suas versões para somar duas palavras A e B de dois bits cada uma (A1...A0 e B1...B0). Também é apresentado um somador híbrido apresentado na literatura, referido no texto como "somador original".

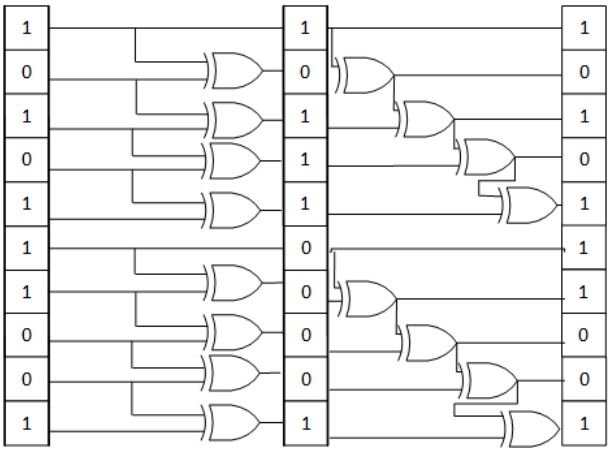

Figura 3.5: Conversão de Binário para Hibrido m=5

A tabela 3.3 mostra a soma de dois bits , ou seja, a soma de valores de 0 até 2 para as entradas a e b e 0 ou 1 para entrada Cin. Os resultados são expressos nas representações binária e híbrida. Através de algumas simplificações chegamos as duas versões A (3.3.2) e B (3.3.3). Alterando as estruturas de portas lógicas de xor para mux, já que o mux pode ter comportamento igual ao de uma porta xor, finalmente chegamos a dois somadores com comportamento diferentes dependentes dos dados de entrada e arquitetura empregada. Como poderá ser visto nas próximas sessões os somadores novos desenvolvidos tem menor área e número de células que o criado por [Costa 2002] e comportamentos distintos conforme circuito aplicado.

3.3.1 Somador Original

A implementação do circuito somador elaborado por [Costa 2002] tem em seu núcleo um somador binário convencional e, nas suas entradas e saídas, circuitos de conversão híbrido-binário e binário-híbrido, respectivamente. O diagrama deste somador é mostrado na figura 3.1. Neste momento, não havia sido proposto um circuito somador que operasse diretamente em código híbrido e que apresentasse menor dissipação de potência que este somador. Entretanto, como apontado por [Costa 2002], dependendo da correlação dos dados e dos valores de capacitância de carga pode-se atingir uma redução de potência total com este somador, se comparado ao somador binário.

A figura 3.7 apresenta o diagrama em portas lógicas do somador proposto por [Costa 2002]. Este somador tem o seguinte estrutura lógica: sete portas XOR, seis portas AND e quatro portas OR, totalizando dezessete portas lógicas.

.

Tabela 3.3: Soma de 2 bits na representação decimal,binária e híbrida.

Decimal				**Binário**				**Híbrida**			
a	**b**	**Cin**	**s**	**a**	**b**	**Cin**	**s**	**a**	**b**	**Cin**	**s**
0	0	0	0	00	00	0	000	00	00	0	000
0	1	0	1	00	01	0	001	00	01	0	001
0	2	0	2	00	10	0	010	00	11	0	011
1	0	0	1	01	00	0	001	01	00	0	001
1	1	0	2	01	01	0	010	01	01	0	011
1	2	0	3	01	10	0	011	01	11	0	010
2	0	0	2	10	00	0	010	11	00	0	011
2	1	0	3	10	01	0	011	11	01	0	010
2	2	0	4	10	10	0	100	11	11	0	100
0	0	1	1	00	00	1	001	00	00	1	001
0	1	1	2	00	01	1	010	00	01	1	011
0	2	1	3	00	10	1	011	00	11	1	010
1	0	1	2	01	00	1	010	01	00	1	011
1	1	1	3	01	01	1	011	01	01	1	010
1	2	1	4	01	10	1	100	01	11	1	100
2	0	1	3	10	00	1	011	11	00	1	010
2	1	1	4	10	01	1	100	11	01	1	100
2	2	1	5	10	10	1	101	11	11	1	101

3.3.2 Somador Proposto A

A figura 3.6 mostra o diagrama do novo somador híbrido A, proposto neste trabalho. Este somador tem uma estrutura diferente do somador original proposto por [Costa 2002], uma vez que possui número maior de multiplexadores. O somador A é composto por cinco portas XOR, cinco multiplexadores, uma porta AND, uma porta OU e dois inversores. O somador A tem um número menor de portas lógicas e caminho crítico reduzido em relação ao somador original [Costa 2002], resultando assim em baixa dissipação de potência.

3.3.3 Somador Proposto B

A figura 3.9 se refere ao segundo somador híbrido proposto, chamado Somador B. Este somador tem treze portas lógicas: cinco portas XOR, quatro multiplexadores, uma porta AND, uma porta OR e um inversor. O somador B tem um número maior de portas lógicas quando comparado ao somador binário *full-adder* de 2 bits, porém o resultado de dissipação de potência é inferior ao somador binário em várias situações, devido à codificação híbrida dos operandos.

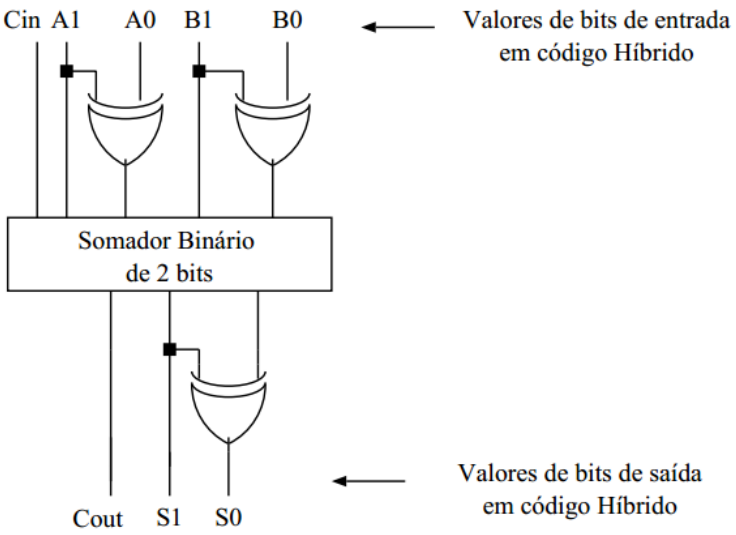

Figura 3.6: Somador Híbrido Original de 2 bits [Costa 2002].

Os resultados me termos de atraso, área, potência e energia destes somadores serão avaliados no capítulo 4 quando são empregados estes somadores em arquiteturas de hardware de módulos do codificador de vídeo HEVC.

3.4 Multiplicador Híbrido em Complemento de 2

De acordo com o trabalho de [Costa 2002] é possível gerar uma simples estrutura de multiplicação de 2 *bits* utilizando 8 portas lógicas. Esta estrutura apresenta menores valores de área e potência em relação à estrutura binária. Uma vantagem que pode ser observada é que, para o uso do multiplicador híbrido, não é necessária nenhuma conversão de tipos entre as codificações híbrida e binária, visto que o multiplicador trabalha diretamente com a codificação híbrida. No multiplicador híbrido tipo *array*, a operação é realizada em cada grupo de m *bits*, onde m representa o número de bits que são multiplicados diretamente. Por exemplo, para m=2, como dois bits são multiplicados simultaneamente, logo a operação é representada na base 4. A linha final para operação de multiplicação em código híbrido é obtida a partir da soma de cada grupo de 2 *bits* dos termos dos produtos parciais. Os valores dos resultados em decimal são obtidos pela conversão de cada grupo de 2 *bits*, assumindo-se uma representação em base 4. A figura 3.10 mostra um exemplo de multiplicação de 4 bits na codificação híbrida na base 4 (m=2).

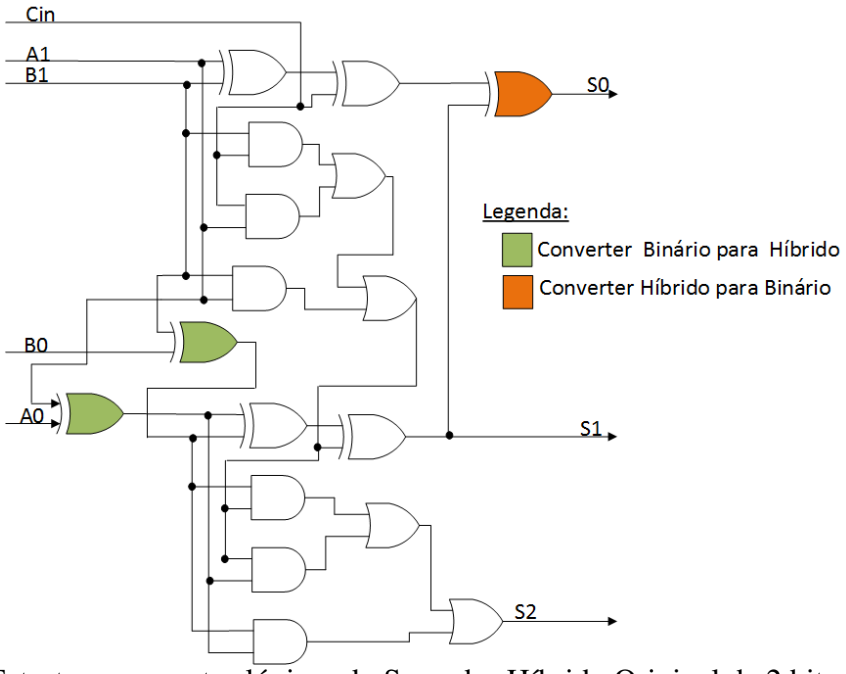

Figura 3.7: Estrutura em portas lógicas do Somador Híbrido Original de 2 bits [Costa 2002].

Pode-se observar, no exemplo da figura 3.10, que três tipos de multiplicação são realizados, sendo uma multiplicação do Tipo I, que realiza uma multiplicação sem sinal (11 x 11 = 2 x 2), duas multiplicações do Tipo II, que realizam multiplicações de um termo sem sinal por um termo com sinal (11 x 11 = -2 x 2 ou 2 x -2) e uma multiplicação do Tipo III, que realiza a multiplicação de dois termos com sinal (11 x 11 = -2 x -2). Deve-se também observar, na figura, que a extensão do sinal é realizada pelo termo (10), que representa -1 na codificação híbrida.

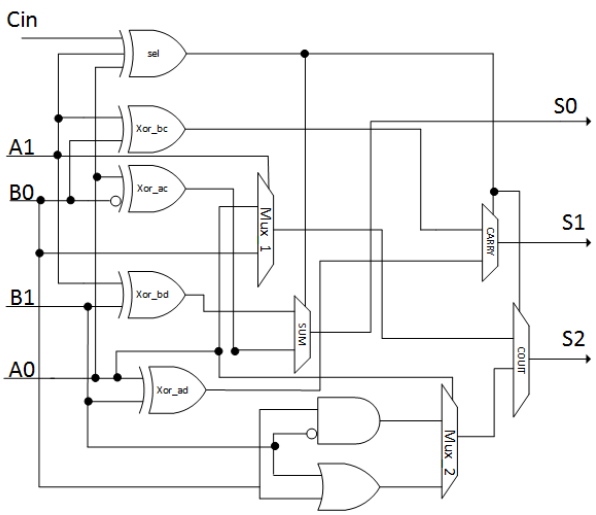

Figura 3.8: Estrutura em portas lógicas do somador híbrido A de 2 bits.

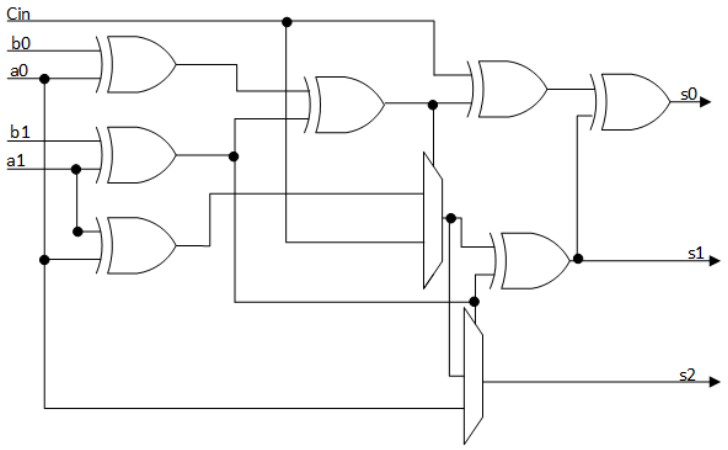

Figura 3.9: Estrutura em portas lógicas do somador híbrido B de 2 bits.

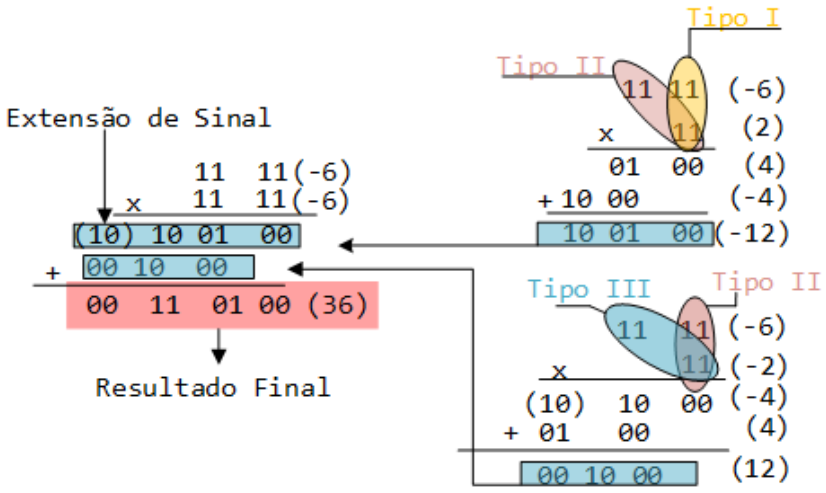

Figura 3.10: Exemplo de multiplicação numérica de 4 bits na base 4 (m=2) na codificação híbrida.

A arquitetura do multiplicador híbrido proposto em [Costa, Monteiro and Bampi 2007] pode ser vista na figura 3.11, para operação com largura de bits W=8, base-4 (m=2) em complemento de 2. Como pode ser visto, o multiplicador híbrido *array* apresenta uma estrutura regular, com os elementos básicos sendo utilizados para realizar os produtos parciais. Os dois bits menos significativos são obtidos imediatamente após a primeira multiplicação. Os demais bits são obtidos pela soma dos termos dos produtos parciais. Pode ser observado também que se utiliza uma linha de circuitos somadores responsáveis pelas somas dos termos dos produtos parciais. Para uma arquitetura em código híbrido de W bits serão utilizados ((w / 2) -1) linhas de somadores.

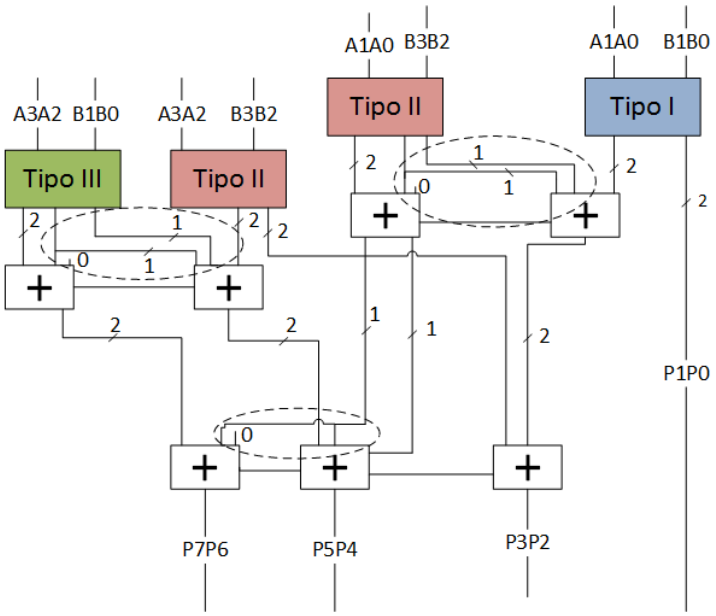

Figura 3.11: Estrutura Multiplicador Hibrido

3.5 Resumo do capítulo

Este capítulo apresentou os conceitos básicos sobre codificação híbrida (combinando a codificação binária e gray), sua conversão para o código binário e seus benefícios em termos de número de transições comparado ao código binário. Foram ainda apresentados os operadores aritméticos híbridos da literatura utilizados neste trabalho, bem como os dois novos somadores híbridos propostos neste trabalho. Como mostra os resultados os somadores são extremamente dependentes da natureza dos dados de entrada. O Capítulo 4 avalia o uso destes operadores em arquiteturas de hardware para módulos do codificador HEVC desenvolvidas neste trabalho, quantificando a redução de potência que pode ser obtida com estes operadores se comparado a operadores binários convencionais.

4 ARQUITETURAS DESENVOLVIDAS

Este capítulo apresenta a proposta de arquiteturas de hardware para três módulos do codificador de vídeo HEVC: filtro de interpolação, cálculo de SAD e Quantização. As arquiteturas utilizam a codificação híbrida para os operandos e os operadores aritméticos descritos no Capítulo 3. Este capítulo descreve as arquiteturas, a metodologia empregada para extrair resultados de síntese e uma discussão dos resultados e dos trabalhos relacionados.

4.1 Arquitetura para o Filtro de Interpolação

O Filtro de Interpolação do HEVC é um dos módulos mais intensivos em computação do codificador HEVC, por ser muito utilizado no processo de ME [Diniz 2015]. Consequentemente, é também um dos módulos que mais consome energia no codificador HEVC. O trabalho de [Costa 2002] demonstra que os operadores aritméticos de codificação híbrida, revisados no Capítulo 3, tem o potencial de redução de potência e energia comparado a operadores aritméticos convencionais, que trabalham na codificação binária.

Neste trabalho, foi desenvolvida uma arquitetura sequencial para o filtro de interpolação, utilizando operadores aritméticos de codificação híbrida, de forma a diminuir potência e energia quando comparado à solução binária. O mesmo *template* arquitetural é utilizado para construir os filtros de interpolação de luma, de 7 e 8 *taps*, e o filtro de interpolação de croma, 4 *taps*. A diferença entre as arquiteturas dos filtros de luma e croma encontra-se unicamente na máquina de estados (*Finite State Machine* - FSM) que armazena os valores dos coeficientes apropriados utilizados e gerencia o número de entradas (4 ou 8 entradas), como será detalhado a seguir.

Antes de passar aos detalhes da arquitetura, é preciso ressaltar que nesta arquitetura todos os coeficientes dos filtros e os pixels de entrada dos filtros são convertidos para a codificação híbrida. Para um melhor entendimento desta questão, as Tabelas mostradas no Anexo A mostram os valores dos coeficientes dos filtros de crominância e luminância, respectivamente, nas representações decimal, binária e híbrida, de forma a facilitar a discussão sobre o consumo energético das arquiteturas propostas. Como apresentado na sessão 2.6.4, existem 3 tipos de filtros de interpolação de luminância (1/4, 2/4 e 3/4), para gerar as amostras fracionárias de luminância de diferentes posições. Além disto, existem 7 diferentes tipos de filtro de interpolação de crominância (1/8, 2/8, 3/8, 4/8, 5/8, 6/8, 7/8) para gerar as amostras fracionárias de crominância de diferentes posições.

A arquitetura sequencial proposta para os filtros de interpolação do HEVC é mostrada na

Fig. 4.1. As figuras 4.1-a e 4.1-b apresentam a arquitetura utilizando operadores aritméticos de codificação binária e híbrida, respectivamente. Ambas arquiteturas incluem um multiplicador que faz a multiplicação dos pixels de entrada pelos coeficientes do filtro a ser calculado. O somador acumula os resultados parciais das multiplicações, que são armazenados no registrador. Os multiplexadores de entrada selecionam o *pixel* e o coeficiente apropriado em cada instante de tempo para cada multiplicação parcial, dependendo do tipo de filtro. Estes multiplexadores são controlados por máquinas de estado, como detalhado nesta seção. O multiplexador final seleciona se a soma parcial deve ser deslocada de 6 para direita ou não, sendo isto aplicado para alguns tipos de filtro somente após computadas todas os resultados parciais. A saída de ambas arquiteturas é um *pixel* fracionário.

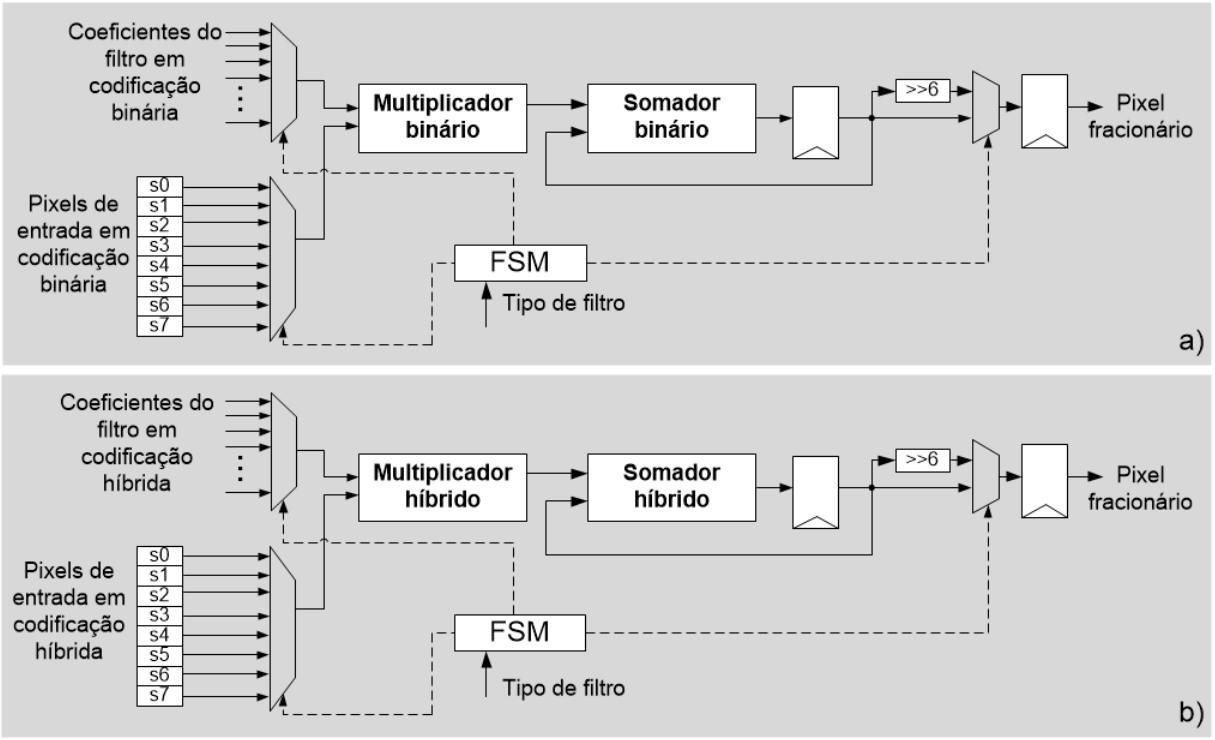

Figura 4.1: Diagrama da arquitetura dos filtros de interpolação: (a) Usando operadores aritméticos de codificação binária; (b) Usando operadores aritméticos de codificação híbrida.

No HEVC, os pixels são geralmente representados com 8 bits em uma representação sem sinal (de 0 a 255). Já os coeficientes dos filtros são representados com uma representação com sinal. Já que é necessária a multiplicação com sinal do *pixel* pelo coeficiente, neste trabalho foi estendida a representação dos pixels e coeficientes para 10 bits em ambas arquiteturas (com operadores binários e híbridos). Foi utilizado 10 bits e não 9 bits pois no código híbrido com m=2 o código *gray* é formado em grupos de 2 bits, logo, o número de bits de entrada deve ser par. Deste modo, os multiplicadores binários e híbridos neste trabalho possuem 10 *bits* de

entrada e 20 *bits* de saída. Já que a saída do multiplicador é de 20 bits, o somador (acumulador) também é de 20 bits. Não há necessidade de um somador com número maior de bits pois a acumulação de 8 pixels não causa *overflow* na representação de 20 bits, no pior caso.

As máquinas de estados controlam a seleção dos multiplexadores de entrada para envio dos *pixels* e coeficientes para o restante da arquitetura. As máquinas de estado possuem algumas variações dependendo do filtro e da representação (binária ou híbrida) utilizada. Portanto, quatro FSMs foram desenvolvidas, para cada filtro (luma e croma) e uma para cada versão, binária e híbrida. As Figuras 4.2 e 4.3 apresentam o diagrama das FSMs para os filtros de luma e croma, respectivamente. Em ambas FSMs, o tipo e filtro é selecionado primeiro. Com base nisto, a FSM vai para o estado correspondente, onde cada estado seleciona um *pixel* de entrada (S0 a S7, para o filtro luma, e S0 a S3, para o filtro croma) e o coeficiente correspondente com base no tipo de filtro. Observando as máquinas de estado, são necessários 10 ciclos de *clock* para calcular um *pixel* fracionário na arquitetura do filtro de luma e 5 ciclos de *clock* na arquitetura do filtro de croma

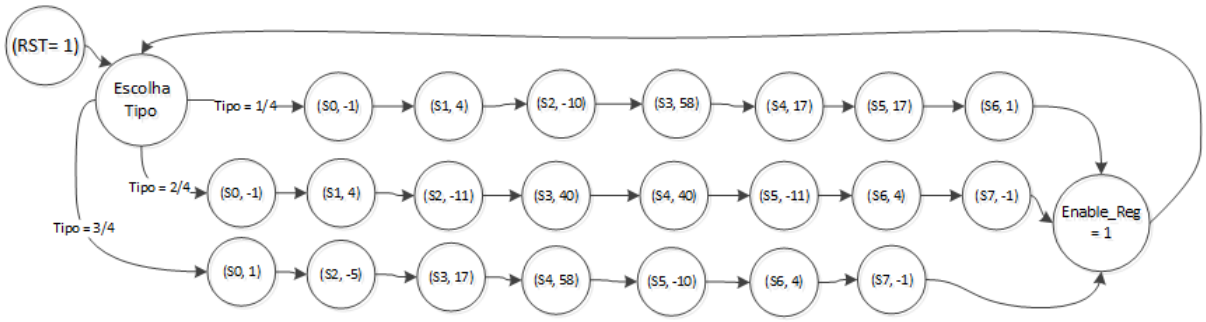

Figura 4.2: Diagrama da FSM da arquitetura do filtro luma

A frequência alvo de operação é de 103.608 MHz (9.64 ps) para a arquitetura do filtro de luma e 31.1 MHz (3.2150 ps) para a arquitetura do filtro de croma ambas focada na compensação de movimento. Com esta frequência, a arquitetura atinge o desempenho para processar vídeos de resolução 720x480 pixels a 30 quadros por segundo. As equações 4.2 e 4.3 apresentam o cálculo da frequência alvo de operação para as arquiteturas dos filtros de luma e croma, respectivamente, considerando a resolução mencionada e uma subamostragem 4:2:0.

$$f = Hpixels * Vpixels * (quadros/segundo) * (ciclos/pixel) \tag{4.1}$$

$$f(luma) = 720 * 480 * 30 * 10 = 103.608MHz \tag{4.2}$$

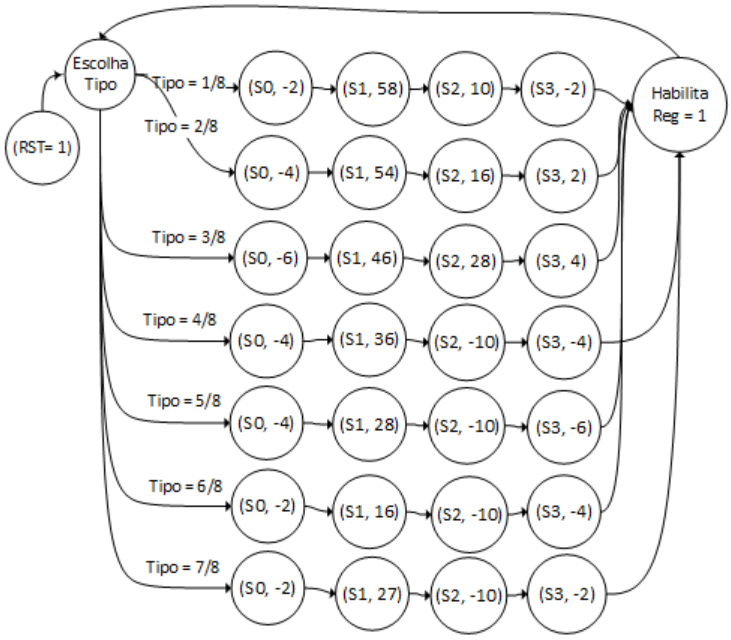

Figura 4.3: Diagrama da FSM da arquitetura do filtro croma

$$f(croma) = 360 * 240 * 2 * 6 = 31.104MHz \tag{4.3}$$

Isto foi feito alterando a função *filter* da classe TComInterpolationFilter. A função *filter* é uma função generalista que descreve todos filtros de interpolação definidos no HEVC.

4.2 Arquitetura para a Soma das Diferenças Absolutas

A Soma das Diferenças Absolutas (SAD) é a métrica mais utilizada para determinar a maior semelhança entre dois blocos de vídeo no processo de Estimação de Movimento. Esta métrica também pode ser aplicada para determinar o melhor *offset* para Estimação de Movimento Fracionário e para escolher o melhor modo de predição Intra. O SAD é calculado pela soma de diferenças absolutas *pixel* a *pixel* entre dois blocos: o bloco a ser codificado e um bloco de referência, como já apresentado no Capítulo 2. Quando aplicado somente na Estimação de Movimento Inteira, o cálculo do SAD consome aproximadamente até 12% do tempo de execução do codificador HEVC em *software* [Vanne et al. 2012]. Devido a sua simplicidade, a métrica SAD é adotada quando o foco do projeto do sistema é a eficiência de energia.

A arquitetura de hardware de SAD é composta por subtratores, operadores absolutos, uma árvore de adição e um acumulador para calcular o valor final do SAD. Para obter alto desempenho, muitas unidades de hardware de SAD são usadas em paralelo dentro de arquiteturas de estimação de movimento. O uso de muitas arquiteturas de SAD em paralelo dissipa muita

potência.

O diagrama da arquitetura de hardware SAD desenvolvida neste trabalho é mostrado na figura 4.4. Ela é composta por oito subtratores, para subtrair oito amostras do bloco original (*Orig*) com oito amostras do bloco de referência (*Ref*), gerando as amostras residuais que são armazenadas em oito registradores. Oito operadores para cálculo do absoluto (módulo) recebem como entrada o resíduo e executam esta operação. Uma árvore de somadores acumula os oito valores absolutos e gera um valor SAD parcial. O SAD parcial é armazenado em um registrador em cada ciclo de *clock*. Logo, após oito ciclos de *clock* o valor de SAD final é calculado para um bloco de 8x8 amostras.

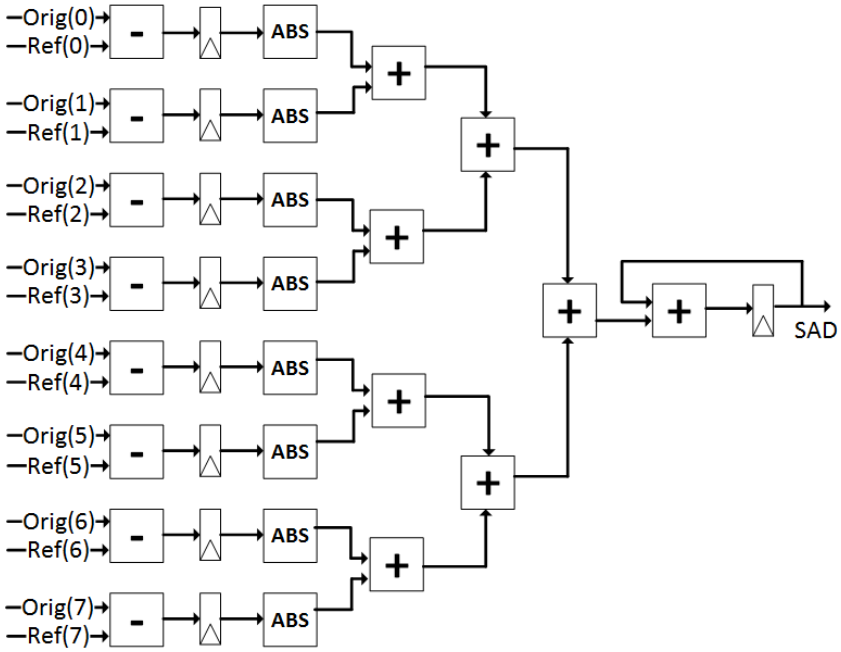

Figura 4.4: Arquitetura original de SAD

Com base na arquitetura mostrada na figura 4.4, foi desenvolvida uma arquitetura de SAD empregando somadores híbridos, como mostrado na figura 4.5. Nesta arquitetura, as amostras de de entrada (original e de referência) são codificadas em binário. As operações de subtração e cálculo do absoluto são feitas em binário. Antes da árvore de somadores, é realizada a conversão de código binário para código híbrido (módulo Bin para Hib na figura 4.5). No capítulo 3, a figura 3.1 mostra o módulo de conversão de binário para híbrido. Nos somadores da árvore de somadores e no acumulador é empregado o somador híbrido A, proposto neste trabalho. Também foi avaliado o somador híbrido original proposto em [Costa 2002]. O resultado final da árvore de somadores precisa ser novamente convertido de híbrido para binário (conforme conversor mostrado na figura 3.1).

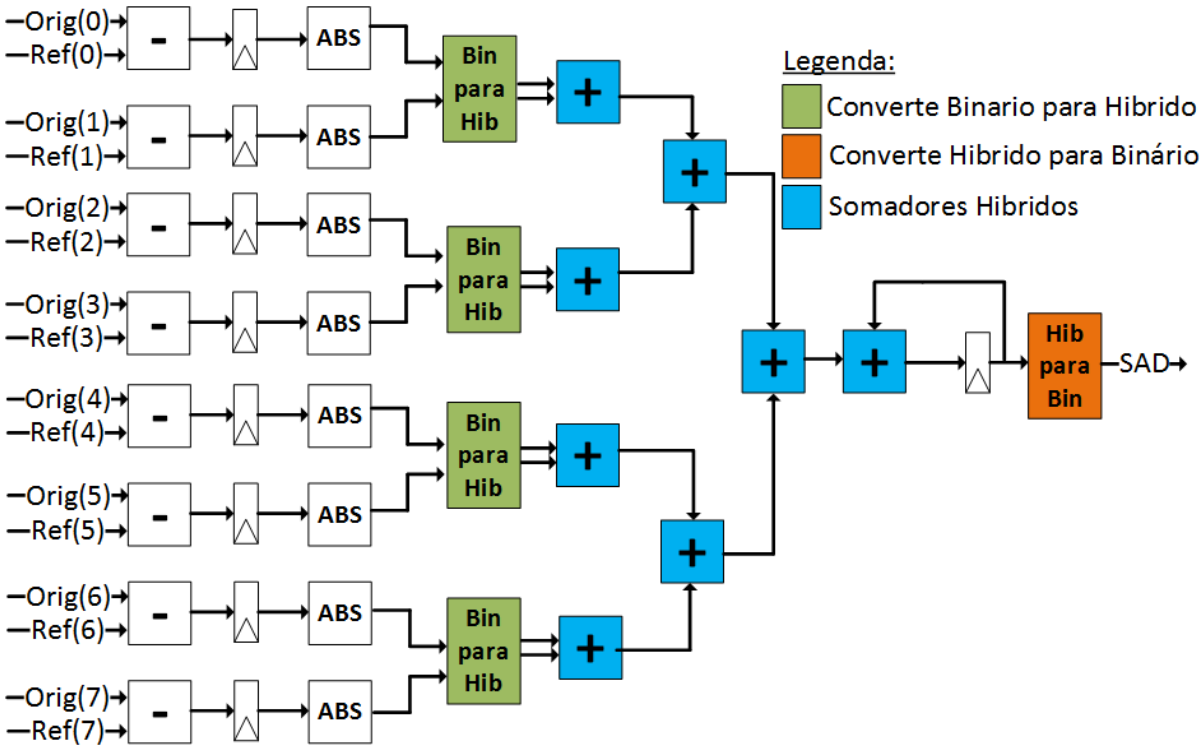

Figura 4.5: Arquitetura de SAD utilizando somadores hibridos

4.3 Quantização

Esta seção apresenta uma nova arquitetura de hardware de quantização unificada (quantização e quantização inversa) para o codec HEVC usando operadores aritméticos híbridos (capítulo 3. A arquitetura desenvolvida utiliza um multiplicador híbrido de 16 bits proposto em [Costa, Monteiro and Bampi 2007] e os dois somadores híbridos propostos neste trabalho (seção 3.3). A arquitetura de quantização proposta usa operadores aritméticos híbridos para reduzir o consumo energético em comparação com a mesma arquitetura usando operadores aritméticos binários. O referencial teórico da quantização pode ser visto na sessão 2.6.5.

O diagrama da arquitetura unificada de quantização e quantização inversa proposta é mostrado na Fig. 4.6. A configuração para realizar quantização ou quantização inversa é feita selecionando o sinal de OPT, que ignora o cálculo do absoluto se selecionado a quantização inversa. O *datapath* da arquitetura binária (a versão que usa operadores aritméticos binários) é composto pelos seguintes operadores: absoluto, multiplicador, somador, deslocador, retorno do sinal original (antes da operação absoluta) e operação de clip para manter o valor final no intervalo de 16 bits. A arquitetura híbrida usa o multiplicador híbrido proposto por [Costa 2002] e a inclusão do dois somadores híbridos (Somador A e Somador B) propostos neste trabalho, além do módulo de valor absoluto híbrido.

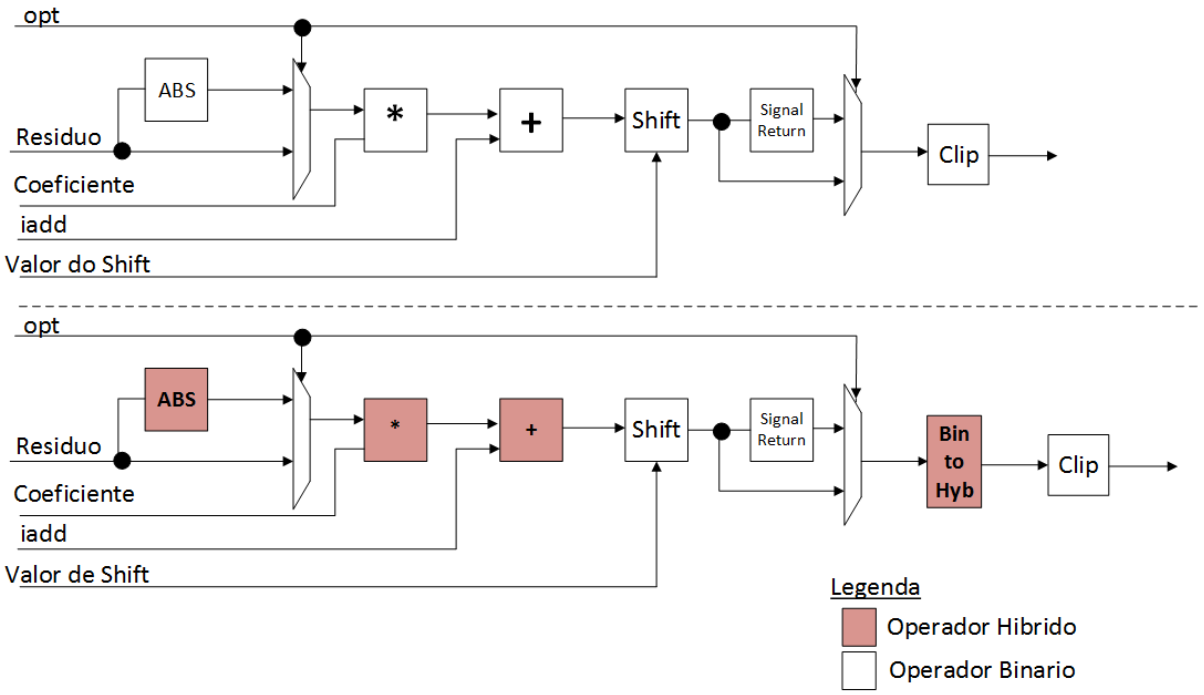

Figura 4.6: Diagrama da arquitetura unificada de quantização direta e inversa.

Ambas as arquiteturas (binária e híbrida) processam uma amostra por ciclo. Uma vez que não existem dependências nas operações de quantização entre os coeficientes, pode-se obter um maior desempenho colocando-se mais de uma arquitetura em paralelo, dependendo da exigência de desempenho. O foco deste trabalho não está em atingir o maior desempenho (em termos de amostras quantizadas por ciclo), mas em reduzir o consumo energético da arquitetura de quantização.

5 IMPLEMENTAÇÃO E METODOLOGIA PARA OBTENÇÃO DOS RESULTADOS

As arquiteturas apresentadas foram descritas na linguagem de descrição de *hardware* VHDL (*Very High Speed Integrated Circuits Hardware Description Language*) com foco na síntese para células padrão (*standard cells*). Foi utilizada a tecnologia de 45 nm e a biblioteca de células *Nangate Open Cell Library*. A síntese foi realizada usando a ferramenta *Cadence RTL Compiler*, usando as voltagens de 0,95 V, 1,10 V e 1,25 V. A ferramenta de síntese foi otimizada para redução de potência e não para redução de área.

Os resultados de potência são gerados através da extração de informações do HM utilizando sequência de vídeos reais. Desta forma, obtém-se um resultado de potência muito mais realista do que as estimativas de potência fornecidas pela ferramenta de síntese quando não são utilizados estes dados reais. A metodologia para obtenção dos resultados de potência é apresentada na Fig. 5.1.

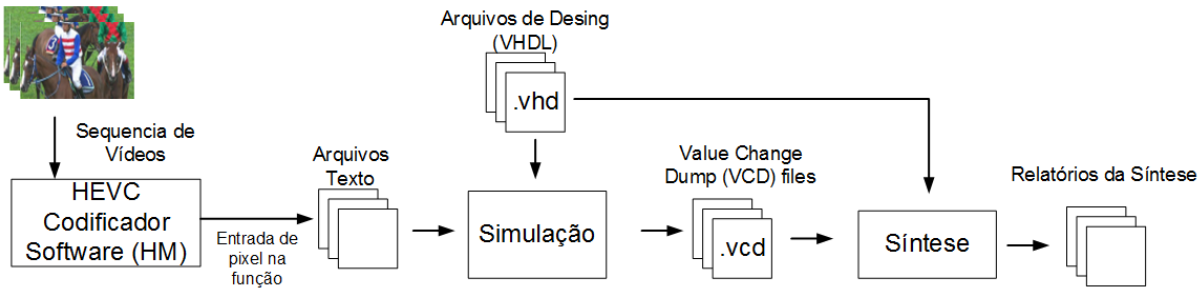

Figura 5.1: Metodologia para obtenção dos resultados de potência.

As sequências de vídeo utilizadas são aquelas sugeridas no documento *Common Test Conditions* (CTC) do HEVC [Bossen 2013] (veja seção 2.8. Estas sequências de vídeo são fornecidas como entrada do codificador de vídeo do *software* HM versão 16.7. O código do *software* HM foi modificado para imprimir as amostras de entrada de cada função, dependendo da arquitetura avaliada (Filtro de Interpolação, SAD ou Quantização). As informações extraídas do HM foram exportadas para arquivos texto. Abaixo é mostrado um exemplo do arquivo gerado. Cada arquitetura estudada utiliza um arquivo específico de estímulo.

```
1 /* Parte do Arquivo enviado para a sintese */
2 /*Video:BlowingBubbles  */
3 /*funcao de interpolacao */
4
5 /* Amostras em Decimal*/
6 102;102;102;102;
7 102;102;103;104;
```

```
8
9  /* Amostras em Binario */
10 0001100110;0001100110;0001100110;0001100110;
11 0001100110;0001100110;0001100111;0001101000;
12
13 /* Amostras em Hibrido */
14 0001110111;0001110111;0001110111;0001110111;
15 0001110111;0001110111;0001110110;0001111100;
```

Os arquivos texto e as descrições VHDL das arquiteturas são inseridas na ferramenta de simulação da Cadence (*irun*). Para cada vídeo utilizado foi criado um *testbench* específico contemplando a estrutura necessária para estimular a arquitetura analisada. A arquitetura foi estimulada com 10.000 vetores de entrada, dependendo do circuito estudado representa um total de 2 *frames* por sequência. Para cada simulação é gerado um arquivo chamado *Value Change Dump* (VCD) que representa a atividade de chaveamento do circuito. Os arquivos VCD e as descrições VHDL das arquiteturas são finalmente inseridas na ferramenta RTL *Compiler* para síntese e geração dos relatórios de potência, área, atraso (*delay*). Os resultados de síntese das arquiteturas são detalhados nas próximas seções.

5.1 Resultados e Discussões

5.1.1 Resultados da Arquitetura para o Filtro de Interpolação

5.1.1.1 Trabalhos Relacionados

Em relação ao projeto de arquiteturas para filtros de interpolação do padrão HEVC, são encontrados alguns trabalhos na literatura. A maior parte dos trabalhos são focados no aumento de desempenho (*throughput*) e redução de área. Poucos trabalhos são focados na redução de potência. O trabalho de Kalali et al. [Kalali and Hamzaoglu 2014] propõe uma arquitetura paralela para o filtro de interpolação do HEVC usando operadores de soma e deslocamento. Este trabalho apresenta resultados de síntese para *standard cells* com tecnologia 90 nm e também resultados de mapeamento da arquitetura para FPGA, usando a frequência alvo de 200 MHZ. Esta arquitetura pode chegar a um *throughput* para resolução de 3840x2160 pixels a 30 fps. Neste trabalho são utilizados dois vídeos para testes, o *Tennis* e *Kimono*, ambos com resolução 1920x1080. O consumo energético é de aproximadamente 1.500 μJ. O trabalho de Zhou et al. [Zhou, Zhou and Lian 2015] também propõe uma arquitetura paralela usando operadores de

soma e deslocamento. A síntese é feita para *standard cells* na tecnologia de 90 nm para uma frequência alvo de 193 Mhz alcançando um *throughput* para resolução de 3840x2160 pixels a 47 fps. Este trabalho não apresenta resultados de potência ou energia, somente apresentando o cálculo do*throughput*. O trabalho de Huang et al. [Huang et al. 2013] propõe a unificação dos dois filtros de interpolação, luma e croma, em uma única estrutura de hardware. A implementação da arquitetura é realizada na tecnologia de 90 nm, atingindo a frequência de 200 MHz e alcançando um *throughput* para resolução de 3840x2160 a 30 fps. Este trabalho utilizou 3 vídeos para comparação, porém o autor não mencionou o uso do HM. Além disso, o trabalho compara sua estrutura com arquiteturas especializadas para filtros de luma e de croma. O foco do projeto, neste trabalho, se dá na otimização para desempenho e não realiza estimação de potência nem de energia. No trabalho de Pastuszak et al. [Pastuszak and Trochimiuk 2013] é proposta uma arquitetura sequencial para os filtros de interpolação. A arquitetura é implementada em tecnologia de 130 nm e utiliza frequência de 200 MHz a 400 MHz dependendo da resolução a ser atingida. O *throughput* chega a uma resolução HD 1280x720 pixels a 60 fps. O trabalho não apresenta estimativas de potência e energia. O trabalho de Afonso et al. [Afonso et al. 2015] propõe uma arquitetura completa para estimação de movimento possuindo as seguintes características: frequência de 49,6 MHz para resolução de 1920x1080 a 30 fps e 396,8 Mhz para resolução de 4096x2160 a 60 fps. A arquitetura dissipa potência de 6,3 mW e 48,67 mW para as resoluções 1920x1080 a 30 fps e 4096x2160 a 60 fps, respectivamente.

5.1.1.2 Resultados de Síntese e Discussões

As tabelas 5.1 e 5.2 apresentam os resultados de potência e energia por operação das arquiteturas para o filtros de interpolação de luma e croma, respectivamente. São apresentados os resultados para as arquiteturas usando operadores aritméticos de codificação binária e de codificação híbrida, para as 6 sequências de vídeo avaliadas. A energia por operação (em Joules) é calculada conforme a equação 5.1, onde C é o número de ciclos da arquitetura para calcular 1 *pixel* fracionário, T é o período da arquitetura (em segundos) e P_{MEDIA} é a potência total da arquitetura (em Watts).

$$E_{op} = C * T * P_{MEDIA} \tag{5.1}$$

Os resultados mostram uma redução de potência das arquiteturas usando operadores aritméticos de codificação híbrida, quando comparado a mesma arquitetura usando operadores aritméticos convencionais, com codificação binária. A redução de potência é de 3,4% na ar-

quitetura do filtro de croma e de 14,98% na arquitetura do filtro de luma, em média para os seis vídeos avaliados. A redução de potência também leva a uma redução de energia, pois o atraso não é significativamente afetado. A redução de potência se dá devido à menor atividade de chaveamento provocada pela codificação híbrida, que reduz a potência dinâmica dos operadores aritméticos associados, reduzindo a potência das arquiteturas desenvolvidas com estes operadores.

Tabela 5.1: Resultados de potência e energia por operação da arquitetura do filtro de interpolação de luma.

Vídeo	**Resultado**	**Tipo**	**Binário**	**Híbrido**	**Redução (%)**
BasketballDrill	Potência (μW)	Leakage	7.307	6.933	5,11
		Dinâmica	89.694	79.311	11,57
		Total	97.001	86.244	11,08
	Energia (pJ)	Total	6.735	5.988	11,08
BasketballPass	Potência (μW)	Leakage	7.291	6.828	6,35
		Dinâmica	87.070	73.743	15,30
		Total	94361	80571	14,61
	Energia (pJ)	Total	6.552	5.594	14,61
BlowingBublles	Potência (μW)	Leakage	7.246	6.862	5,29
		Dinâmica	84.405	75.520	10,52
		Total	91651	82382	10,11
	Energia (pJ)	Total	6.364	5.720	10,11
BQSQuare	Potência (μW)	Leakage	7.335	6.834	6,83
		Dinâmica	91.024	76.420	16,04
		Total	98359	83254	15,35
	Energia (pJ)	Total	6.830	5.781	15,35
PartyScene	Potência (μW)	Leakage	7.301	6.807	6,76
		Dinâmica	89.168	73.670	17,38
		Total	96469	80477	16,57
	Energia (pJ)	Total	6.698	5.588	16,57
RaceHorses	Potência (μW)	Leakage	7.306	6.835	6,44
		Dinâmica	88.617	73.982	16,51
		Total	95923	80817	15,74
	Energia (pJ)	Total	6.660	5.611	15,74
Redução (%) da Potência Total (Média dos vídeos)					14,98

As tabelas 5.3 e 5.4 apresentam os resultados de número de células e área (de células e de interconexões, ou seja, as nets) das arquiteturas dos filtros de luma e croma, respectivamente. Com relação ao número de células, no caso da arquitetura do filtro de interpolação de croma houve redução. Já na arquitetura para filtro de interpolação de luma, não houve redução significativa. Contudo, como o foco de otimização da ferramenta de síntese e também o objetivo da

Tabela 5.2: Resultados de potência e energia por operação da arquitetura do filtro de interpolação de croma

Vídeo	Resultado	Tipo	Binário	Híbrido	Redução (%)
BasketballDrill	Potência (μW)	Leakage	27.705	22.845	17,54
		Dinâmica	70.259	78.007	-11,02
		Total	97.964	100.852	-2,94
	Energia (pJ)	Total	1.1338	1.1673	//
BasketballPass	Potência (μW)	Leakage	27.687	22.718	17,94
		Dinâmica	70.193	71.448	-1,78
		Total	97.880	94.166	3,79
	Energia (pJ)	Total	1.1329	1.0899	//
BlowingBublle	Potência (μW)	Leakage	27.686	22.665	18,13
		Dinâmica	70.224	69.606	0,88
		Total	97910	92271	5,75
	Energia (pJ)	Total	1.1332	1.0679	//
BQSQuare	Potência (μW)	Leakage	27.700	22.778	17,76
		Dinâmica	70.233	75.838	-7,98
		Total	97932	98616	-0,69
	Energia (pJ)	Total	1.1335	1.1414	//
PartyScene	Potência (μW)	Leakage	27.690	22.720	17,94
		Dinâmica	70.231	72.188	-2,78
		Total	97921	94908	3,07
	Energia (pJ)	Total	1.1333	1.0985	//
RaceHorses	Potência (μW)	Leakage	27.686	22.663	18,14
		Dinâmica	70.189	70.813	-0,88
		Total	97874	93475	4,49
	Energia (pJ)	Total	1.1328	1.0819	//
Redução (%) da Potência Total (Média dos vídeos)					3,43

codificação híbrida é a redução de potência dinâmica, e não de área, o fato da não redução de área do circuito não é esperado.

Tabela 5.3: Resultados de área da arquitetura do filtro luma

	Binário	Híbrido	Redução (%)
Número de células	753	741	1.59%
Area em células (μm^2)	1,112	1,071	3.68%
Área em interconexões (μm^2)	2,887	2,832	1.90%

A comparação dos resultados de potência obtidos pelas arquiteturas desenvolvidas neste trabalho podem ser feitas somente com os trabalhos [Kalali and Hamzaoglu 2014] e [Afonso et al. 2015] que mostram resultado de energia e potência. Considerando o trabalho de [Kalali and Hamzaoglu 2014], fica difícil a comparação, já que estamos comparando uma arquitetura

Tabela 5.4: Resultado de área da arquitetura do filtro croma

	Binário	**Híbrido**	**Aumento (%)**
Número de células	1,153	1,222	5.98%
Area em células (μm^2)	1,479	1,431	-3.24%
Área em interconexões (μm^2)	4,143	4,386	5.86%

sequencial com uma arquitetura paralela. Mesmo assim, os resultados de [Kalali and Hamzaoglu 2014] apontam um consumo energético de aproximadamente 1500 μJ enquanto as nossas arquiteturas para os filtros de croma e luma consomem 1000 μJ e 5000 μJ, respectivamente. Na arquitetura proposta em [Afonso et al. 2015] a dissipação de potência apresentada é de 48,67 mW enquanto a arquitetura proposta neste trabalho é de 89.003 mW, pois no trabalho de [Afonso et al. 2015] a síntese foi realizada para 0,72 V enquanto nosso trabalho utiliza 0,9 V. Desta forma, não é justa a comparação pois as tensões de alimentação são diferentes, o que influencia no resultado de potência.

5.1.2 Resultados da Arquitetura para a Soma das Diferenças Absolutas

5.1.2.1 Trabalhos Relacionados

Os trabalhos de [Nalluri, Alves and Navarro 2013] e [Nalluri, Alves and Navarro 2014] apresentam três arquiteturas de hardware SAD com diferentes níveis de paralelismo. O trabalho de [Dinh et al. 2016] propõe uma arquitetura SAD de tamanho de bloco variável para estimação de movimento. No entanto, as arquiteturas mostram resultados de síntese para *Field Programmable Gate Array* (FPGA), que é menos eficiente em consumo energético do que as células padrão e não pode ser diretamente comparado com o nosso trabalho. O trabalho em [Seidel, Moraes and Güntzel 2013] apresenta uma arquitetura SAD configurada para *pel-decimation*, que reduz a complexidade e a potência, mas diminui a qualidade do vídeo. Todos os trabalhos relacionados empregam somadores de codificação binária usados em arquiteturas SAD.

5.1.2.2 Resultados de Síntese e Discussões

A arquitetura foi sintetizada com frequência alvo de 400 MHZ e alimentação de 1,1 V. A metodologia para estimação de resultados de potência foi realizada utilizando os seguintes vídeos: *Blowing Bubbles, BQTerrace, Nebuta Festival, Race Horses, BQSquare, e Steam Locomotive Train*.

Como resultado final da síntese física, a arquitetura utiliza 1.238 células e ocupa uma área de 2.056 μm^2. Como comparação, sínteses foram realizadas utilizando os seguintes somadores: somador da ferramenta, utilizando a macro-função "+", referido por "Ferramenta"; somador híbrido proposto por [Costa 2002], referido como Somador Original e o somador híbrido A proposto neste trabalho. Os resultados são apresentados na tabela 5.5. A tabela 5.5 mostra os resultados de potência de *leakage*, *dinâmica* e total da arquitetura de SAD utilizando estas três opções de somadores.

Tabela 5.5: Resultados de potência (em μW) da arquitetura de SAD.

Vídeo	Somador	Leakage	Dinâmica	Total	Redução (%) Somador A	Aumento (%) Somador B
Blowing Bubble	**Ferramenta**	14.362	410.760	425.151	7.33	8.91
	Original	17.136	428.405	445.541	11.57	3.92
	Somador A	16.427	377.557	393.984	//	17.52
	Somador B	17.498	428.706	463.039	14.91	//
RaceHorses	**Ferrramenta**	14.815	412.587	427.402	7.41	12.60
	Original	17.298	446.298	463.595	14.64	3.80
	Somador A	16.879	378.819	395.697	0	21.62
	Somador B	17.660	446.599	481.255	17.77	0
BQSquare	**Ferramenta**	14.570	375.310	389.880	6.95	13.99
	Original	17.383	409.301	426.685	14.98	4.15
	Somador A	16.604	346.162	362.766	0	22.51
	Somador B	17.745	409.602	444.430	18.37	0
Steam Locomotive Train	**Ferramenta**	14.410	417.390	431.800	7.46	11.28
	Original	17.198	445.750	462.948	13.69	3.79
	Somador A	16.459	383.094	399.553	0	20.26
	Somador B	17.560	446.051	480.508	16.84	0
BQTerrace	**Ferramenta**	14.360	410.834	425.194	7.34	8.86
	Original	17.579	427.350	444.929	11.45	4.03
	Somador A	16.427	377.557	393.984	0	17.48
	Somador B	17.941	427.651	462.870	14.88	0
Nebuta Festival	**Ferramenta**	14.641	419.973	434.614	7.69	10.59
	Original	17.442	445.449	462.871	13.32	3.84
	Somador A	16.705	384.474	401.179	0	19.81
	Somador B	17.784	445.750	480.655	16.53	0

Como pode ser notado os somadores (versão A e Versão B) tem comportamento oposto um do outro. Enquanto a versão A reduz o consumo energético a versão B aumenta o consumo. Isto justifica-se pelo comportamento do dado de entrada, largura de bits e arquitetura no qual foi aplicada os somadores.

Os resultados de área e número de células são apresentados na tabela 5.6.

Tabela 5.6: Resultados de área da aquitetura de SAD.

Somador	Celulas	Área
Ferramenta	913	1752
Original	1191	2011
Somador A	1238	2056
Somador B	1100	2002

A comparação da arquitetura proposta com as arquiteturas apresentadas em [Nalluri, Alves and Navarro 2013], [Nalluri, Alves and Navarro 2014] e [Dinh et al. 2016] não pode ser feita já que os resultados apresentados neste trabalho são para ASIC, enquanto os trabalhos aqui citados sintetizam as arquiteturas apenas para FPGA, portanto não é uma comparação justa a ser feita. O trabalho de [Seidel, Moraes and Güntzel 2013] apresenta resultados para ASIC para uma tecnologia mais antiga (90 nm). Além disto, a arquitetura de SAD em [Seidel, Moraes and Güntzel 2013] tem menor paralelismo do que a nossa, portanto a comparação de potência é difícil. Além disso, nenhum dos trabalhos relacionados usa dados de vídeo real para estimativa de potência. O uso de dados de vídeo real para estimativa de potência é mais confiável do que o uso de vetores de entrada aleatórios, especialmente considerando que nosso trabalho é focado em redução energética.

Os resultados de síntese mostram que a arquitetura SAD usando o somador híbrido A reduz a dissipação de energia em 7,42 % e 12,97 % (em média de todos os vídeos avaliados) quando comparado com a arquitetura SAD usando o somador da ferramenta de síntese e o somador original proposto por [Costa 2002]. A redução energética é justificada pela adoção da codificação híbrida e os operadores aritméticos híbridos, diminuindo assim a atividade de chaveamento da arquitetura, ao custo de um pequeno custo no aumento em área e número de células.

5.1.3 Resultados da Arquitetura para a Quantização

5.1.3.1 Trabalhos Relacionados

Como comparação utilizamos os trabalhos dos autores [Dias, Roma and Sousa 2015] e [Pastuszak and Abramowski 2016] . [Dias, Roma and Sousa 2015] propõe uma arquitetura unificada de quantização e quantização inversa, que opera na frequência de 374 MHz atingindo um throughput de 265 Mpixels/s, sendo suficiente para processar 4k (3840x2160 pixels 30 fps). A tecnologia empregada para o desenvolvida do circuito foi em ASIC com em 90 nm utilizando Vdd igual a 1.2 V. Já o trabalho de [Pastuszak and Abramowski 2016] propõe uma arquitetura

completa para predição intra. Foi desenvolvida utilizando a tecnologia 90nm para FPGA operando com uma frequência de 400 MHz atingindo a resolução máxima de 3640x2160 pixel 30 fps. No trabalho de [Pastuszak and Abramowski 2016] não existe estimativa de *power* isolada somente para arquitetura de quantização e quantização inversa, pois o autor implementou toda a arquitetura de predição intra. Em ambos trabalhos estudados fica difícil a comparação já que empregam tecnologias de tamanho de células diferente da nossa e não utilizam vídeos reais para estimação energética.

5.1.3.2 Resultados de síntese e discussões

A síntese da arquitetura de quantização foi feita para a frequência de 266 MHz com alimentação de 1,25 V. Resultados de síntese mostraram que a arquitetura usando o somador híbrido A ocupa 4.582 células e uma área de 4882 μm^2. Já o somador B utilizou 4.429 células e ocupou uma área de 4.708 μm^2. As arquiteturas foram validadas com dados de entradas reais dos seguintes vídeos: *BasketballDrive, BasketballPass, BlowingBubbles, BQMall, BQSquare, BQTerrace, Cactus, Kimono, NebutaFestival, RaceHorses, RaceHorsesc* e *SteamLocomotive-Train*.

A tabela 5.8 mostra os resultados de potência (*leakage, dinâmica* e total) da arquitetura unificada (operando com a quantização e quantização inversa) usando operadores aritméticos binários e híbridos para cada sequência de vídeo. Os resultados são mostrados para as duas versões dos somadores híbridos propostos. Os resultados em negrito mostram as maiores reduções energéticas usando os Somadores Híbridos A e B propostos neste trabalho.

Tabela 5.7: Resultado de área da arquitetura de quantização.

Somador	**Células**	**Área**
Binário	3136	3563
Somador A	3482	3823
Somador B	3383	3707

Tabela 5.8: Resultado de potência (em μW) da arquitetura de quantização.

Video	Somador	Quantização				Quantização Inversa			
		Leakage	Dinâmica	Total	Redução %	Leakage	Dinâmica	Total	Redução %
Basketball Drive	**Binário**	**180.855**	**810.252**	**991.107**	**-**	**180.838**	**704.201**	**885.039**	**-**
	Somador A	**176.967**	**671.979**	**848.946**	**14.343**	**177.202**	**607.067**	**784.269**	**11.385**
	Somador B	**175.539**	**654.366**	**829.905**	**16.264**	**175.384**	**592.276**	**767.660**	**13.260**
BasketballPass	Binário	180.805	758.701	939.506	-	181.013	667.787	848.800	-
	Somador A	177.405	648.647	826.052	12.075	177.497	603.262	780.759	8.016
	Somador B	175.969	629.241	805.210	14.294	175.612	587.103	762.715	10.141
Blowing Bubbles	Binário	180.724	767.962	948.686	-	180.661	681.974	862.635	-
	Somador A	177.436	665.727	843.163	11.123	177.205	627.105	804.311	6.761
	Somador B	175.977	646.979	822.956	13.253	175.595	610.159	785.754	8.912
BQMall	**Binário**	**180.841**	**796.185**	**977.026**	**-**	**180.639**	**709.834**	**890.474**	**-**
	Somador A	**176.800**	**673.022**	**849.822**	**13.019**	**177.166**	**615.065**	**792.231**	**11.032**
	Somador B	**175.432**	**654.853**	**830.286**	**15.019**	**175.343**	**599.422**	**774.766**	**12.993**
BQSquare	Binário	180.797	757.867	938.664	-	180.776	691.674	872.450	-
	Somador A	177.346	675.851	853.198	9.105	177.548	638.293	815.841	6.488
	Somador B	175.959	657.149	833.107	11.245	175.816	622.792	798.608	8.463
BQTerrace	Binário	180.757	796.148	976.905	-	180.479	704.403	884.882	-
	Somador A	177.256	659.656	836.912	14.330	177.204	621.057	798.260	9.789
	Somador B	175.780	642.102	817.882	16.278	175.509	605.706	781.215	11.71
Cactus	**Binário**	**180.852**	**817.639**	**998.491**	**-**	**180.781**	**710.616**	**891.398**	**-**
	Somador A	**177.085**	**690.845**	**867.929**	**13.075**	**177.326**	**617.901**	**795.227**	**10.788**
	Somador B	**175.623**	**674.269**	**849.892**	**14.882**	**175.470**	**604.135**	**779.605**	**12.541**
Kimono	Binário	180.917	760.774	941.691	-	180.982	668.618	849.599	-
	Somador A	176.980	646.187	823.167	12.586	177.186	602.363	779.549	8.245
	Somador B	175.691	629.984	805.675	14.443	175.428	587.483	762.910	10.203
Nebuta Festival	Binário	180.752	785.923	966.675	-	180.788	682.086	862.874	-
	Somador A	177.142	655.572	832.714	13.857	177.008	615.216	792.224	8.187
	Somador B	175.761	639.433	815.194	15.670	175.288	600.676	775.964	10.072
Race Horses	Binário	180.840	803.457	984.297	-	180.796	703.726	884.522	-
	Somador A	177.049	684.626	861.676	12.457	177.301	627.145	804.446	9.053
	Somador B	175.663	664.974	840.636	14.595	175.494	611.901	787.395	10.980
Race HorsesC	Binário	180.917	760.776	941.693	-	180.982	668.624	849.606	-
	Somador A	176.979	648.369	825.349	12.354	177.186	604.957	782.144	7.940
	Somador B	175.690	631.992	807.683	14.230	175.428	589.904	765.333	9.919
Steam Locomotive Train	**Binário**	**180.852**	**817.639**	**998.491**	**-**	**180.781**	**710.616**	**891.398**	**-**
	Somador A	**177.085**	**690.845**	**867.929**	**13.075**	**177.326**	**617.901**	**795.227**	**10.788**
	Somador B	**175.623**	**674.269**	**849.892**	**14.882**	**175.470**	**604.135**	**779.605**	**12.541**
Média %		Somador A	12.617	Somador B	14.588	Somador A	9.039	Somador B	10.677

Quando selecionada a quantização direta, a arquitetura usando os somadores híbridos A e B, propostos neste trabalho, reduz a dissipação de potência em 12,617 % e 15,888 % do total, respectivamente, em média para as 12 sequências de vídeo, em comparação com a arquitetura usando operadores binários. A redução da dissipação de potência média da quantização inversa é 9,039 % e 10,677 % quando são usados os somadores híbridos A e B, respectivamente, quando comparada à arquitetura que utiliza operadores binários. As diferenças na redução da dissipação de potência entre quantização e quantização inversa são ocasionadas pelo uso de mais um operador híbrido no *datapath* (operador absoluto) que aumenta a disparidade entre as atividades de comutação de arquiteturas híbridas e binárias.

5.2 Resumo do capítulo

Este capítulo apresentou três arquiteturas para módulos do codificador HEVC utilizando operadores aritméticos de codificação híbrida propostos neste trabalho e apresentados na literatura. Com base nos resultados de potência e energia das arquiteturas, observa-se a potencialidade da codificação híbrida e do uso de operadores aritméticos de codificação híbrida para redução de potência e energia de módulos de codificação de vídeo em hardware.

6 CONCLUSÃO

Este trabalho apresentou um estudo sobre conceitos de codificação de vídeo, do novo padrão HEVC, e sobre a codificação híbrida e operadores aritméticos híbridos. O trabalho mostrou o projeto de arquiteturas para módulos do padrão de codificação de vídeo HEVC e o projeto e avaliação do uso de operadores aritméticos de codificação híbrida nestes módulos. Este trabalho propôs dois novos somadores híbridos, chamados de Somador A e Somador B.

Foi apresentada uma arquitetura para o filtro de interpolação fracionário de luminância e de crominância proposto no HEVC, aplicando operadores aritméticos de codificação híbrida propostos na literatura. Foram propostas também uma arquitetura de SAD, utilizando um dos somadores híbrido proposto, e uma nova arquitetura de quantização na qual foram aplicados os dois novos somadores híbridos.

A utilização da codificação híbrida nos operandos e dos operadores aritméticos de codificação híbrida nas arquiteturas mostrou resultados satisfatórios para projetos de baixa potência. As arquiteturas utilizando os operadores aritméticos híbridos reduzem a dissipação de potência na média dos vídeos avaliados em comparação com as mesmas arquiteturas utilizando somadores da ferramenta. Os somadores híbridos propostos também mostraram redução de potência quando comparados ao somadores híbrido existente na literatura.

Em alguns casos, como na arquitetura para o filtro de interpolação, existem limitações quanto ao desempenho, devido à natureza sequencial da arquitetura. Entretanto, no caso das arquiteturas para o cálculo de SAD e para Quantização, obteve-se resultados satisfatórios em termos de desempenho, dissipação de potência e consumo energético. Em alguns resultados de arquiteturas usando operadores híbridos, ocorre um pequeno aumento de área, devido ao fato de que a síntese foi otimizada para redução de potência, ao invés de ser otimizada para redução de área. Pode-se concluir que a codificação híbrida traz benefícios no projeto de operadores e arquiteturas para aceleradores de vídeo de baixa potência.

6.1 Trabalhos Futuros

Como trabalhos futuros, sugere-se o desenvolvimento de novas arquiteturas para módulos do codificador HEVC usando os operadores aritméticos híbridos, tais como o SSD. Pode-se utilizar também os novos somadores híbridos propostos em arquiteturas existentes para o filtro de interpolação (baseado em operações de soma e deslocamento), o SATD, e a DCT. Além disto, outra frente de pesquisa que se abre é a exploração de diferentes valores de m nos operadores

aritméticos e seu impacto na redução de potência das arquiteturas.

6.2 Aceitação de artigo científico no tema do trabalho

No contexto deste trabalho, foi aceito para publicação um artigo na conferência IEEE Latin American Symposium on Circuits and Systems (LASCAS) 2017, com o título **Low Power Sum of Absolute Differences Architecture Using Novel Hybrid Adder**.

REFERÊNCIAS

AFONSO, V. **Desenvolvimento de uma Arquitetura para Estimacao de Movimento Fracionario Segundo o Padrão Emergente HEVC**. [S.l.], 2013.

AFONSO, V. et al. Memory-aware and high-throughput hardware design for the hevc fractional motion estimation. In: **Proceedings of the 28th Symposium on Integrated Circuits and Systems Design**. New York, NY, USA: ACM, 2015. (SBCCI '15), p. 11:1–11:6. ISBN 978-1-4503-3763-2. Available from Internet: <http://doi.acm.org/10.1145/2800986.2801017>.

BOSSEN, F. Common test conditions and software configurations. In: **JCT-VC document no. L1100**. [S.l.: s.n.], 2013.

BROSS, B. et al. **High efficiency video coding text specification**. [S.l.]: draft, 2013.

BUDAGAVI, M.; SULLIVAN, G. J. **High Efficiency Video Coding (HEVC): Algorithms and Architectures**. Springer International Publishing, 2014. (Integrated Circuits and Systems). ISBN 9783319068947. Available from Internet: <https://books.google.com.br/books?id=oTR3oAEACAAJ>.

CARVALHO, M. G. Implementação hardware/software da estimação de movimento segundo o padrão h. 264. Universidade Federal do Rio Grande do Norte, 2008.

COLOR Model Color Sub-Sampling. 2012. Available from Internet: <http://blog.abelcine.com/2011/06/09/hd-formats-color-model-color-sub-sampling/>.

CORREA, G. R. **Computational Complexity Reduction and Scaling for High Efficiency Video Encoders**. Thesis (PhD) — Universidade de Coimbra, 2014.

CORRÊA, M. M.; SCHOENKNECHT, M. T.; AGOSTINI, L. V. A high performance hardware architecture for the h. 264/avc half-pixel motion estimation refinement. In: ACM. **Proceedings of the 23rd symposium on Integrated circuits and system design**. [S.l.], 2010. p. 151–156.

COSTA, E. d. **Operadores Aritmeticos de Baixo Consumo para Arquiteturas de Circuitos DSP. 2002**. Thesis (PhD) — Tese (Doutorado em Ciencia da Computacao—Programa de Pos-Graduacao em Computacao-UFRGS, Porto Alegre, RS, 2002.

COSTA, E. da; MONTEIRO, J.; BAMPI, S. A new array architecture for signed multiplication using gray encoded radix-2m operands. **Integration, the VLSI Journal**, Elsevier, v. 40, n. 2, p. 118–132, 2007.

DIAS, T.; ROMA, N.; SOUSA, L. High performance ip core for hevc quantization. In: **2015 IEEE International Symposium on Circuits and Systems (ISCAS)**. [S.l.: s.n.], 2015. p. 2828–2831. ISSN 0271-4302.

DINH, V. N. et al. High speed sad architecture for variable block size motion estimation in hevc encoder. In: IEEE. **Communications and Electronics (ICCE), 2016 IEEE Sixth International Conference on**. [S.l.], 2016. p. 195–198.

DINIZ, C. M. **Dedicated and Reconfigurable Hardware Accelerators for High Efficiency Video Coding Standard**. 141 p. Thesis (PhD) — Universidade Federal do Rio Grande do Sul. Instituto de Informática. Programa de Pós-Graduação em Computação., 2015. Available from Internet: <http://www.lume.ufrgs.br/handle/10183/118394>.

FERREIRA, M. A. **Porque motivo os televisores 4K ainda não são uma boa compra! – PCManias.com**. 2016. Available from Internet: <http://www.pcmanias.com/porque-motivo-os-televisores-4k-ainda-nao-sao-uma-boa-compra/>.

Fonseca De Oliveira, J. F.; Sampaio De Alencar, M. Padrão HEVC – Novas Tecnologias para Aplicações de Elevadas Taxas de Compressão de Vídeo. **REVISTA DE TECNOLOGIA DA INFORMAÇÃO E COMUNICAÇÃO**.

GONZALEZ, R.; WOODS, R. **Processamento Digital De Imagens**. ADDISON WESLEY BRA, 2014. ISBN 9788576054016. Available from Internet: <https://books.google.com.br/books?id=r5f0RgAACAAJ>.

GONçALVES, G. W. **Desenvolvimento de um IP core para DCT e Quantização segundo padrão HEVC: Projeto em Electronic System Level**. Dissertação, 2014.

HUANG, C.-T. et al. Hevc interpolation filter architecture for quad full hd decoding. In: **Visual Communications and Image Processing (VCIP), 2013**. [S.l.: s.n.], 2013. p. 1–5.

ITU. **International Telecommunication Union. H.265 Recommendation. SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS Infrastructure of audiovisual services – Coding of moving video**. [S.l.], 2013.

KALALI, E.; HAMZAOGLU, I. A low energy hevc sub-pixel interpolation hardware. In: IEEE. **2014 IEEE International Conference on Image Processing (ICIP)**. [S.l.], 2014. p. 1218–1222.

KAO, C.-Y.; KUO, H.-C.; LIN, Y.-L. High performance fractional motion estimation and mode decision for h. 264/avc. In: IEEE. **2006 IEEE International Conference on Multimedia and Expo**. [S.l.], 2006. p. 1241–1244.

LAI, Y.-K.; CHEN, L.-F.; HUANG, S.-Y. Hybrid parallel motion estimation architecture based on fast top-winners search algorithm. **IEEE Transactions on Consumer Electronics**, IEEE, v. 56, n. 3, p. 1837–1842, 2010.

MENDEL, G. **Qual a resolução em megapixels do olho humano?** 2013. Available from Internet: <https://darwinismo.wordpress.com/2013/07/18/qual-a-resolucao-em-megapixels-do-olho-humano/>.

MIGLIARI, I. Créditos. **Revista Mackenzie de Engenharia e Computação**, v. 14, n. 1, 2015.

MOREIRA, T. Conceitos e fundamentos do hevc/h.265. 2015.

NALLURI, P.; ALVES, L. N.; NAVARRO, A. A novel sad architecture for variable block size motion estimation in hevc video coding. In: IEEE. **System on Chip (SoC), 2013 International Symposium on**. [S.l.], 2013. p. 1–4.

NALLURI, P.; ALVES, L. N.; NAVARRO, A. High speed sad architectures for variable block size motion estimation in hevc video coding. In: IEEE. **Image Processing (ICIP), 2014 IEEE International Conference on**. [S.l.], 2014. p. 1233–1237.

OLIVEIRA, J. Padrao hevc- novas tecnologias para aplicação de elevadas taxas de compreensão de video. 2014.

OLIVEIRAJ.B. **Arquitetura Distribuida para Transcodificação de Vídeo com Alto Desempenho**. Dissertação, 2013.

PASTUSZAK, G.; ABRAMOWSKI, A. Algorithm and architecture design of the h.265/hevc intra encoder. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 26, n. 1, p. 210–222, Jan 2016. ISSN 1051-8215.

PASTUSZAK, G.; TROCHIMIUK, M. Architecture design and efficiency evaluation for the high-throughput interpolation in the hevc encoder. In: IEEE. **Digital System Design (DSD), 2013 Euromicro Conference on**. [S.l.], 2013. p. 423–428.

PAULO, F. de S. **Netflix supera 65 milhões de assinantes e já vale mais que a GM**. 2016. Available from Internet: <http://www1.folha.uol.com.br/mercado/2015/07/1656808-netflix-atinge-marca-de-65-milhoes-de-assinantes-no-mundo-todo.shtml>.

PEREIRA, R. **Analisando objetivamente a qualidade de um vídeo**. 2008. Available from Internet: <https://rafaelspereira.wordpress.com/tag/psnr/>.

RICHARDSON, I. E. **THE H.264 Advanced Video Compression Standard**. [S.l.]: John Wiley & Sons, 2010.

RICHARDSON, I. E. **The H. 264 advanced video compression standard**. [S.l.]: John Wiley & Sons, 2011.

SEIDEL, I.; MORAES, B. G. de; GÜNTZEL, J. L. A low-power configurable vlsi architecture for sum of absolute differences calculation. In: IEEE. **Circuits and Systems (LASCAS), 2013 IEEE Fourth Latin American Symposium on**. [S.l.], 2013. p. 1–4.

SEIDEL, I. et al. Análise do impacto de pel decimation na codificação de vídeos de alta resolução. 2014.

SULLIVAN, G. J. et al. Overview of the high efficiency video coding (hevc) standard. **IEEE Transactions on circuits and systems for video technology**, IEEE, v. 22, n. 12, p. 1649–1668, 2012.

VANNE, J. et al. Comparative rate-distortion-complexity analysis of hevc and avc video codecs. **IEEE Trans. Circuits Syst. Video Technol.**, v. 22, n. 12, p. 1885–1898, Dec 2012. ISSN 1051-8215.

ZHOU, W.; ZHOU, X.; LIAN, X. An efficient interpolation filter vlsi architecture for hevc. In: **2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**. [S.l.: s.n.], 2015. p. 1106–1110. ISSN 1520-6149.

ANEXO A — TABELA DE COEFICIENTES

As tabelas abaixo mostram os coeficientes na representação binária e híbrida utilizadas no filtro chroma e luma.

Tabela A.1: Representação dos coeficientes do filtro Chroma em binário e híbrido

1/8 (chroma)	**-2**	**58**	**10**	**-2**
Binario	1111111110	0000101000	0000001010	1111111110
Hibrido	1010101011	0000101111	0000001111	1010101011
2/8 (chroma)	**-4**	**54**	**-16**	**-2**
Binario	1111111100	0000110110	0000010000	1111111110
Hibrido	1010101000	0000100111	0000010000	1010101011
3/8 (chroma)	**-6**	**46**	**28**	**-4**
Binario	1111111010	0000101110	0000011100	1111111100
Hibrido	1010101111	0000111011	0000011000	1010101000
4/8 (chroma)	**-4**	**36**	**36**	**-4**
Binario	1111111100	0000100100	0000100100	1111111100
Hibrido	1010101000	0000110100	0000110100	1010101000
5/8 (chroma)	**-4**	**28**	**46**	**-6**
Binario	1111111100	0000011100	0000101110	1111111010
Hibrido	1010101000	0000011000	0000111011	1010101111
6/8 (chroma)	**-2**	**16**	**54**	**-4**
Binario	1111111110	1111111010	0000110110	1111111100
Hibrido	1010101011	1010101111	0000100111	1010101000
7/8 (chroma)	**-2**	**10**	**58**	**-2**
Binario	1111111110	0000001010	0000101000	1111111110
Hibrido	1010101011	0000001111	0000101111	1010101011

Tabela A.2: Representação dos coeficientes do filtro Luma em binário e híbrido

1/4 (luma)	**-1**	**4**	**-10**	**58**	**17**	**-5**	**1**	
Binario	1111111111	0000000100	1111110110	0000101000	0000010001	1111111011	0000000001	
Hibrido	1010101010	0000000100	1010100111	0000101111	0000010001	1010101110	0000000001	
2/4 (luma)	**-1**	**4**	**-11**	**40**	**40**	**-11**	**4**	**-1**
Binario	1111111111	0000000100	1111110101	0000101000	0000101000	1111110101	0000000100	1111111111
Hibrido	1010101010	0000000100	1010100101	0000111100	0000111100	1010100101	0000000100	1010101010
3/4 (luma)		**1**	**-5**	**17**	**58**	**-10**	**4**	**-1**
Binario		0000000001	1111111011	0000010001	0000101000	1111110110	0000000100	1111111111
Hibrido		0000000001	1010101110	0000010001	0000101111	1010100111	0000000100	1010101010