UNIVERSIDADE FEDERAL DO PAMPA

LUANA VIEIRA MARTINEZ BONATTO

DESENVOLVIMENTO DE ARQUITETURAS DIGITAIS PARA O MÓDULO DE TRANSFORMADAS, SEGUNDO O PADRÃO HEVC DE CODIFICAÇÃO DE VÍDEO

Bagé
2016

LUANA VIEIRA MARTINEZ BONATTO

DESENVOLVIMENTO DE ARQUITETURAS DIGITAIS PARA O MÓDULO DE TRANSFORMADAS, SEGUNDO O PADRÃO HEVC DE CODIFICAÇÃO DE VÍDEO

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Computação da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Computação.

Orientador: Prof. MSc. Fábio Luís Livi Ramos

Bagé
2016

B699d Bonatto, Luana Vieira Martinez
Desenvolvimento de arquiteturas digitais para o módulo de transformadas, segundo o padrão HEVC de codificação de vídeo / Luana Vieira Martinez Bonatto.
74 p.

Trabalho de Conclusão de Curso(Graduação)--Universidade Federal do Pampa, ENGENHARIA DE COMPUTAÇÃO, 2016.
"Orientação: Fábio Luís Livi Ramos".

1. codificação de vídeo. 2. HEVC. 3. transformadas. 4. DCT. 5. baixo consumo energético. I. Título.

LUANA VIEIRA MARTINEZ BONATTO

DESENVOLVIMENTO DE ARQUITETURAS DIGITAIS PARA O MÓDULO DE TRANSFORMADAS, SEGUNDO O PADRÃO HEVC DE CODIFICAÇÃO DE VÍDEO

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Computação da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel Engenharia de Computação.

Trabalho de Conclusão de Curso defendido e aprovado em: 07 de dezembro de 2016.

Banca examinadora:

_____

Prof. MSc. Fábio Luís Livi Ramos
Orientador
UNIPAMPA

_____

Prof. MSc. Julio Saraçol Domingues Júnior
UNIPAMPA

_____

Prof. Dr. Bruno Silveira Neves
UNIPAMPA

AGRADECIMENTO

Primeiramente gostaria de agradecer à minha família materna, em especial a minha avó Iolanda por ter cuidado de mim e me dado a oportunidade de estudar. Não poderia deixar de agradecer a duas pessoas que não estão mais aqui, a minha bisavó Eva pela troca de cuidados na minha infância e a minha mãe Ivana, que mesmo com pouquíssimo tempo de vida ao meu lado, sei que sempre cuidou de mim de onde ela está.

Gostaria de agradecer também ao meu namorado César pelo carinho, motivação, companheirismo e incentivo. É muito importante saber que tenho alguém com quem posso contar nos momentos bons e ruins.

Um grande agradecimento ao meu orientador Fábio Ramos, primeiramente pela oportunidade do trabalho e por ter me auxiliado e direcionado no entendimento e desenvolvimento deste trabalho.

Agradeço à UNIPAMPA pela oportunidade de estudo e aos professores pelos ensinamentos ao longo da graduação, principalmente aos professores do curso de Engenharia de Computação, que foram essenciais para o meu crescimento acadêmico. Um agradecimento especial ao professor Érico Amaral pelo empenho dedicado em motivar-me à participar de suas pesquisas, incentivando-me à escrever artigos científicos e à participar dos eventos da área de computação.

Agradeço também aos meus colegas e amigos Lesley, Leonardo, Camila e Maurício pelo apoio e amizade durante o curso.

E a todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigada.

“A única maneira de descobrir os limites do possível é ir além deles até o impossível”.

Arthur C. Clarke

RESUMO

Com o aumento significativo de tecnologias que fazem uso de vídeo digital, a codificação de vídeo surge dando suporte ao armazenamento e transmissão desse tipo de dado. As técnicas de codificação de vídeo apresentam como objetivo principal, reduzir a representatividade necessária para processar um vídeo, de maneira que esse processo não prejudique a qualidade visual do mesmo. Sendo assim, este trabalho apresenta o desenvolvimento de uma arquitetura para o Módulo de Transformada Discreta do Cosseno (DCT – *Discrete Cosine Transform*) 1-D (1-Dimensão) do codificador de vídeo digital do padrão H.265 (HEVC). Dessa maneira, esta proposta consiste na implementação de uma estrutura em hardware digital que seja capaz de processar os multi-tamanhos (4x4, 8x8, 16x16 e 32x32) de matrizes residuais, por meio de um desenvolvimento puramente combinacional. Como resultado, busca-se ganho de eficiência energética com a inserção de uma técnica de baixo consumo, baseada na análise do comportamento estatístico do uso das transformadas em sequência reais de vídeo, sem prejuízos consideráveis para o desempenho da arquitetura. Para a validação funcional é utilizado o *software* ModelSim da Mentor Graphics, e resultados de frequência, área e consumo de energia são obtidos por meio da síntese ASIC para a biblioteca de 65nm da ST, utilizando a ferramenta RTL Compiler da Cadence.

Palavras-Chave: codificação de vídeo, HEVC, transformadas, DCT, baixo consumo energético.

ABSTRACT

There has been an increasing demand for higher quality video, considering the high amount of electronic devices that process digital video, resulting in increasingly high resolutions. For that purpose, video coding techniques are used, which have as main goal the reduction of the required representation to process a digital video, implying in no loss of the visual quality of the transmitted and stored video. This work proposes a 1-D architecture Transform module (Discrete Cosine Transform) of the H.265 (HEVC) digital video encoder. Furthermore, the proposed work implements a digital hardware structure able to process all the four sizes (4x4, 8x8, 16x16 and 32x32) of residual matrixes, by using a purely combinational fashion. As result, energetic efficiency is achieved by inserting a low-power technique, based on a statistical analysis of the use of these transforms when playing real videos segments, without significant prejudice to the architecture performance. The functional validation was made using Mentor Graphics software Modelsim, and the frequency, area and power consumption results were obtained for ASIC 65 nm ST gate library synthesis, using Cadence RTL Compiler tool.

Keywords: video coding, HEVC, transforms, DCT, *low-power*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

1D – 1 Dimensão
2D – 2 Dimensões
ASIC – Application Specific Integrated Circuit
AVC – Advanced Video Coding
Cb – Chrominance Blue
CCITT – Comité Consultatif International Téléphonique et Télegraphique
CMOS – Complementary Metal Oxide Semiconductor
CMYK – Cyan, Magenta, Yellow, blacK
codec – COder/DECoder
CPO – Critical Path Optimization
Cr – Chrominance Red
CTB – Coding Block Tree
CTU – Coding Tree Unit
CU – Coding Unit
CV – Coeficiente de Variação
DCT – Discrete Cossine Transform
DPI – Dots Per Inch
DST – Discrete Sine Transform
DVD – Digital Versatile Disc
FPGA – Field Programmable Gate Array
FullHD – Full High Definition
HD – High Definition
HEVC – High Efficiency Video Coding
HM – HEVC Test Model
HVS – Human Visual System
IDCT – Inverse Discrete Cossine Transform
IEC – International Eletrotechnical Comission
ISO – International Organization os Standardization
ITU-T – International Telecommunication Union
JCT-3V – Join Collaborative Team on 3D Video Coding
JCT-VC – Join Collaborative Team on Video Coding
MMCM – Multiplierless Multiple Constant Multiplication

MPEG – Moving Picture Experts Group

N-p – N-point

N-pt – N-point

PPI – Pixel Per Inch

PU – Prediction Unit

RExt – Range Extension Format

RGB – Red, Green, Blue

RTL – Register Transfer Level

SCC – Screen Content

SHVC – Scalable HEVC

TSMC – Taiwan Semiconductor Manufacturing Company

TU – Transform Unit

UHD – Ultra High Definition

UNIPAMPA – Universidade Federal do Pampa

VCEG – Video Coding Experts Group

VHDL – VHSIC Hardware Description Language

VHSIC – Very High Speed Integrated Circuits

WQXGA – Quad Extended Graphics Array

Y – Luminance

YCbCr – Luminance, Chrminance Blue, Chrominance Red

SUMÁRIO

1 INTRODUÇÃO

Atualmente, tem-se percebido um aumento significativo de tecnologias que fazem uso de vídeo. Neste cenário, está presente uma vasta gama de produtos eletrônicos, tais como, computador pessoal ou portátil, *smartphones*, DVD (*Digital Versatile Disc*) *players*, câmeras e filmadoras digitais, televisores UHD (*Ultra High Definition*), ou seja, os mais diversos utilitários (SZE *et al*., 2014). A principal motivação para a crescente utilização de vídeo ocorre pela popularidade dos serviços ofertados por empresas como Adobe, Amazon, Apple, HBO, Hulu, Microsoft, Netflix e YouTube, que representam, com *streaming* de vídeo, uma fatia correspondente à 50% do tráfego da Internet (SUMMERS *et al.,* 2016).

Para otimizar a transmissão destes *streamings* em um canal de comunicação com largura de banda limitada, uma das alternativas é a utilização de técnicas de codificação de vídeo, cujo objetivo é reduzir o *bitstream* necessário para representar o vídeo sem que haja perda significativa de informação. Essa técnica é explorada devido à propriedade específica da redundância de informação entre o mesmo quadro e quadros diferentes dentre uma mesma sequência de vídeo (SZE *et al*., 2014). Os sinais de vídeo que não utilizam nenhum método de compressão geram uma grande quantidade de dados, o que poderia inviabilizar o seu armazenamento e transmissão de forma eficiente, sendo assim, é observável a importância da aplicação de técnicas de codificação de vídeo. O HEVC (*High Efficiency Video Coding* – H.265) é o estado da arte dos padrões de codificação de vídeo. Seu principal diferencial em relação ao padrão anterior (H.264/AVC - *Advanced Video Coding*) é o melhor desempenho de compressão, cerca de 50% mais eficiente (SEIDEL *et al*., 2016).

Os grandes esforços a fim de encontrar soluções de implementação para os padrões de compressão de vídeo são focados em soluções em hardware. Assim esperando-se que com a arquitetura dedicada do *codec* (codificador e decodificador), diferentemente dos codificadores em software, alcance taxas de processamento em tempo real para vídeos de alta definição (RAMOS, 2010).

O processo de transformada, que vem sendo utilizado nos últimos padrões de codificação de vídeo, é uma técnica de manipulação de *bits* usualmente aplicada para rearranjar os dados de uma forma mais eficiente. Para isso, o bloco de transformadas visa organizar os resíduos gerados pelos processos de predição (isto é, as diferenças

entre os blocos codificados e preditos), posteriormente codificados pelo bloco de entropia.

Neste contexto, o presente trabalho aborda a construção de um bloco residual da DCT 1-D do codificador de vídeo digital no padrão H.265 (HEVC). A proposta consiste na implementação de uma estrutura em hardware digital que seja capaz de processar os quatro tamanhos (4x4, 8x8, 16x16 e 32x32) de matrizes residuais, por meio de um desenvolvimento puramente combinacional. Com isso, a arquitetura é implementada em linguagem de descrição de hardware VHDL (*VHSIC High Description Language*), e propõe melhorias em hardware, como a reutilização de componentes, reduzindo o caminho crítico e ganho de eficiência energética.

1.1 Motivação

Levando em consideração a Lei de Moore, a cada ano o poder computacional é dobrado, sendo assim, os dispositivos eletrônicos estão se tornando cada vez mais baratos e eficientes. No entanto, a medida que os dispositivos se tornam menores, a capacidade do poder da bateria diminui, com isso, existe a necessidade de fornecer conteúdos multimídia de qualidade e com uma menor taxa de *bits*, implicando na redução do consumo de energia (MINALLAH *et al.*, 2015).

Com o aumento de aplicações baseadas em tecnologias de vídeo-transmissão, surgem desafios relacionados à concepção de soluções que suportem de forma eficiente o grande fluxo de *streaming* de vídeo. Neste contexto, diferentes soluções vêm sendo exploradas pela comunidade científica, dentre elas, arquiteturas específicas para os módulos de codificação de vídeo. No entanto, identifica-se potencial de desenvolvimento científico no que se refere ao ganho de eficiência energética, de espaço e de frequência. Assim, este trabalho propõe a concepção de uma solução em hardware, buscando o *trade-off* entre as variáveis potência, área, caminho crítico e frequência durante o projeto da arquitetura alvo.

1.2 Objetivos

Este trabalho apresenta como objetivo geral desenvolver e validar uma arquitetura para o Módulo de Transformadas 1-D (Transformada Discreta do Cosseno) do codificador de vídeo no padrão HEVC. Esta arquitetura deve ser capaz de

processar os multi-tamanhos (4x4, 8x8, 16x16 e 32x32) de transformadas buscando reduzir o consumo energético.

1.2.1 Objetivos Específicos

Com o intuito de alcançar um maior detalhamento dos objetivos específicos, abaixo eles estão destacados individualmente.

I. Realizar um estudo referente ao estado da arte do padrão HEVC;
II. Buscar e analisar Módulos de Transformadas no contexto de vídeo digital;
III. Realizar estudo e análise do Software de Referência alusivo ao padrão de interesse e extrair dados do mesmo;
IV. Propor a otimização do módulo de transformadas no que se refere à consumo energético;
V. Desenvolver arquitetura digital do Módulo de Transformadas, aplicando técnicas de *low-power*;
VI. Simular e validar funcionalmente a arquitetura desenvolvida utilizando o Software de Referência como parâmetro.
VII. Sintetizar a arquitetura para ASIC (*Application Specific Integrated Circuit)* avaliando resultado de área e frequência a partir da descrição VHDL;
VIII. Analisar o desempenho em relação ao consumo de energia da arquitetura sintetizada.

1.3 Estrutura do Texto

O texto está estruturado em sete capítulos e um anexo, em conformidade com os assuntos que relatam o problema de pesquisa

Além deste capítulo de Introdução, Objetivos e Motivação, o Capítulo 2 consiste em uma revisão bibliográfica dos conceitos de Codificação de Vídeo Digital, com denominação de Conceitos Gerais e Revisão da Literatura. Este capítulo está dividido na seção 2.1 Conceitos de vídeo, seção esta que abrange os principais conceitos que compõem uma imagem e vídeo digital, ela está subdividida em quatro subseções. Na seção 2.2 Codificação de Vídeo, contempla os assuntos mais técnicos da codificação

de vídeo, iniciando com alguns conceitos e finalizando com os padrões existentes, o padrão atual e o processo de codificação utilizada no H.265 – HEVC. Esta seção está subdividida em três subseções.

No Capítulo 3, é efetuada uma explanação mais ampla, com foco no módulo de interesse deste trabalho. Este capítulo divide-se em 3.1 Módulos de Transformadas, que ramifica-se em duas subseções.

Tendo em vista a importância de um trabalho estruturado, o capítulo 4, intitulado como Metodologia, explana sobre os processos de pesquisa, desenvolvimento e validação do trabalho. Neste capítulo também é feita uma introdução às ferramentas utilizadas neste desenvolvimento.

O Capítulo 5 apresenta a arquitetura das transformadas, sendo estruturado em subseções, como segue: 5.1 Transformada Discreta do Cosseno – Software de Referência, 5.2 Transformada Discreta do Cosseno para economia de recursos, onde são apresentadas as principais arquiteturas implementadas.

Encontram-se no capítulo 6, os Resultados e Discussões, apresentando na subseção 6.1 Resultados de síntese ASIC e na 6.2 Conclusão, onde são apresentadas as considerações dos resultados obtidos e comparados com os trabalhos correlatos.

Por fim, no capítulo 7 Conclusão, se detém na conclusão geral do trabalho, expondo a visão obtida em relação a teoria e a prática.

Ao final, o Anexo A apresenta a metodologia de validação das arquiteturas.

2 CONCEITOS GERAIS E REVISÃO DE LITERATURA

Neste capítulo, será abordado alguns conceitos sobre a representação e a compressão do vídeo digital, também será efetuada uma breve introdução ao padrão de compressão de vídeo H.265/HEVC.

2.1 Conceitos de Vídeo

Esta subseção é dedicada para a apresentação de conceitos básicos de vídeo digital. Os conceitos explorados não são estritamente aplicados ao padrão, sendo de conhecimento básico do funcionamento do mesmo.

2.1.1 Vídeo Digital

Um vídeo pode ser facilmente definido em relação ao termo digital, como sendo uma tecnologia de processamento de sinais eletrônicos, com o intuito de capturar, armazenar, transmitir ou indicar movimento. A representação de um vídeo digital é efetuada através de uma sucessão de imagens com certa velocidade. O sistema de visão humano percebe as imagens, como sendo um sinal de duas dimensões. Sendo assim, o olho humano é capaz de identificar em torno de vinte imagens por segundo, fazendo com que a visão de uma pessoa crie uma alusão de imagem animada, implicando em um vídeo (ROSA, 2010). A figura 1 demonstra uma sequência de imagens, indicando a sensação de movimento.

Figura 1 - Sequência de imagens com alusão ao movimento

Fonte: Repositório digital da Calaveras Images[1]

[1] Disponível em: < http://www.calaverasimages.com/Animals > Acesso em maio de 2016

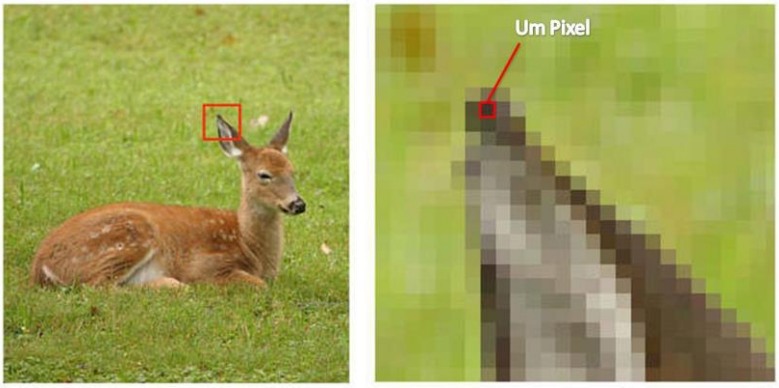
De acordo com Poynton (2012), a fluidez de um vídeo é explicitada pelo número de imagens visualizadas por segundo, termo conhecido como taxa de quadros ou do inglês *frame rate*. É visível a necessidade da transmissão de uma sequência de imagens para se obter a sensação de movimento. Uma sequência de imagens possui duas dimensões espaciais e uma terceira dimensão do sinal de vídeo que se refere ao tempo. As dimensões espaciais são representadas por um número finito de elementos, cada um com uma localização e valor específico. O uso de cor em análise de imagem e compressão está se tornando cada vez mais popular em vários domínios da transformação e transmissão de imagem (PARKER *et al.*, 2013).

A técnica aplicada à transferência de informações multimídia através de uma rede é caracterizada como taxa de transmissão, ou do inglês, *streaming*, onde esta denominação indica o fluxo contínuo de dados ou conteúdos multimídia que trafegam em um canal de comunicação da rede. Com a demanda de alta resolução em dispositivos eletrônicos de uso diário (como por exemplo, televisores UHD e smartphones), os distribuidores de serviço de rede tendem a preocupar-se com a largura de banda que satisfaça a necessidade dos consumidores (OLIVEIRA *et al.*, 2014). Sinais de vídeo moderno, como dito por Parker *et al.,* (2013), são representados, armazenados e transmitidos digitalmente, e para isso é indispensável a utilização de técnicas de processamento de dados do vídeo, que é a manipulação do redimensionamento do tamanho de armazenamento do vídeo.

2.1.2 Pixel

Para Kou (1995), em computação, uma imagem digital é representada através de uma matriz de números de elementos de imagem, de modo que, cada elemento compõe o conjunto de todos os componentes necessários para representar as cores. Toda e qualquer imagem é composta por um elemento no seu nível mais básico, onde tal elemento é denominado *pixel*. O *pixel* é um componente essencial para todas as imagens digitais, sendo ele uma unidade de informação lógica que armazena as informações de cor e intensidade (DINIZ, 2009). Sua denominação surgiu da junção das palavras “*PICture*” e “*ELement*”. É possível combinar milhões de *pixels* para criar uma imagem detalhada e aparentemente contínua, como pode ser observado na figura 2.

Figura 2 - Representação de um pixel



Fonte: Repositório digital da Xtech Commerce[2]

Kou (1995), explana que, cada *pixel* pode estar representando apenas uma cor por vez, porém, pelo fato deles serem tão pequenos, muitas vezes se misturam para formar vários tons e misturas de cores. Para se obter uma boa precisão de cor, o *pixel* depende de sua cadeia de *bits*, ou seja, quanto maior for a quantidade de *bits* do *pixel*, maior o número de cores que podem ser representadas por ele. Uma imagem bem detalhada depende de quantos *pixels* ela contém. Com relação à precisão da representação, as imagens são classificadas em três categorias:

- **Preto e branco:** Onde cada *pixel* é representado por um único *bit*.
- **Escala de cinza:** Cada *pixel* é representado por luminância (intensidade luminosa), normalmente representada com 8 *bits*.
- **Colorida:** É composta por múltiplos componentes. Cada *pixel* de cor pode ser representado por luminância e crominância (intensidade de cor).

Para especificações de tamanho de imagem, são utilizados os termos PPI (*Pixel Per Inch*) que significa *pixel* por polegada e DPI (*Dots Per Inch*) que são pontos por polegada. Esses termos são utilizados para relacionar a unidade teórica dos *pixels* com uma resolução visual real. Um *pixel* por polegada descreve exatamente a quantidade de *pixels* que uma imagem contém por cada polegada de distância na horizontal e na vertical. Já os pontos por polegada denotam a quantidade de pontos

[2] Disponível em: < http://suporte.xtechcommerce.com/hc/pt-br/articles/201198509 > Acesso em maio de 2016.

presentes por polegada. A utilização de diversos pontos para criar cada um dos *pixels* é um processo conhecido como *dithering*[3].

2.1.3 Espaço de Cores

O espaço de cor representa um modelo matemático usado para descrever cada cor a partir de fórmulas, ou seja, de forma análoga, é uma maneira de organizar as cores em um *pixel*. Isso é efetuado obedecendo a gama de cores que podem ser processadas por algum meio, seja por um monitor, utilizando o padrão RGB (*Red-Green-Blue*, Vermelho-Verde-Azul) ou impressora com o padrão CMYK (*Cyan-Magenta-Yellow-blacK,* Ciano-Magenta-Amarelo-Preto). As diversas maneiras de se representar as informações de cor se baseiam na fisiologia do sistema visual humano e na teoria das cores. A classificação dos formatos de vídeo digital portou a base da transmissão analógica, com formatos muito similares.

Conforme Midha *et al.* (2014), os esquemas de formatos de cores mais utilizados no ramo digital são ramificados em YCbCr (*Luminance, Chrominance Blue, Chrominance Red)*, e RGB, onde cada um tem suas próprias propriedades. O YCbCr é o mesmo padrão YUV utilizado em imagem analógica, sendo ele amplamente utilizado em vídeo e compressão de imagem digital.

No formato RGB, um *pixel* é formado pelas cores primárias de um vídeo digital, as quais são respectivamente vermelho, verde e azul. Segundo Agostini (2007), os componentes R, G e B apresentam um grau elevado de correlação com a informação de luminância, tal informação que é identificada pelo olho humano com muito mais intensidade do que a informação de cor. Sendo assim, um espaço de cor que tenha a luminância como uma das componentes principais é privilegiado, pois permite que diferentes ferramentas de compressão possam ser aplicadas para luminância e crominância separadamente (DINIZ, 2009).

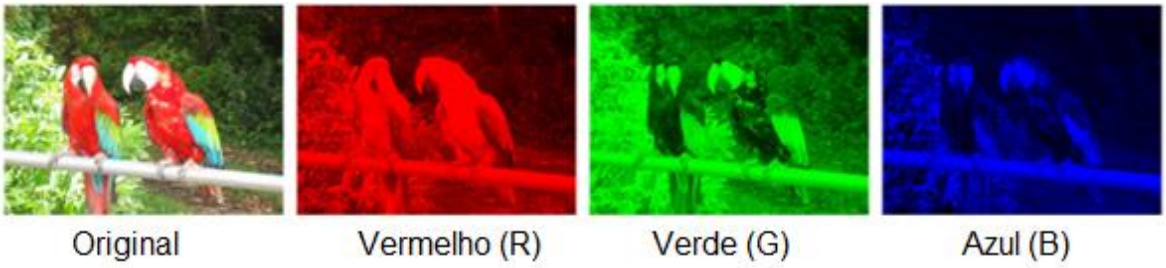
Já o sistema de cores do formato YCbCr representa um esquema de codificação de cores para imagens naturais na qual a luminância e a crominância são separados. A luminância se refere aos componentes em escala de cinza de um sinal de vídeo, que tem como função controlar a intensidade da luz, enquanto a crominância é um componente de cor do sinal de vídeo.

---

[3] Disponível em: < http://www.cambridgeincolour.com/pt-br/tutoriais/digital-camera-pixel.htm > Acesso em maio de 2016.

A figura 3 apresenta uma imagem decomposta em seus componentes RGB (vermelho, verde e azul), e a figura 4 mostra a decomposição da imagem em seus componentes YCbCr (luminância e crominância), respectivamente.

Figura 3 - Decomposição da imagem no formato RGB



Fonte: Disponível no repositório digital da Microsoft[4]

Figura 4 - Decomposição da imagem no formato YCbCr

Fonte: Disponível no repositório digital da Microsoft[4]

2.1.4 Sub-amostragem de Cores

Uma etapa do processo de compressão de vídeos digitais, é a diminuição das informações da cor do vídeo, ou seja, a redução da resolução (sub-amostragem). Isso é possível separando as informações do vídeo em luminância e crominância. A crominância é menos sensível para a visão humana do que a luminância, e esta sensibilidade reduzida é pouco perceptível. Para realizar essa redução, as informações do espaço de cor RGB do vídeo são transformadas para o espaço de cor YCbCr, e posteriormente, as informações de Cb e Cr têm sua taxa de amostragem reduzidas dos componentes de crominância em relação aos componentes de luminância. A esta redução dá-se o nome de sub-amostragem de crominância (POYNTON, 2012).

[4] Disponível em <https://msdn.microsoft.com/en-us/library/windows/desktop/dn424131(v=vs.85).aspx> Acesso em maio de 2016.

Pelo fato do olho humano apresentar menor sensibilidade as variações de cores do que as variações de intensidade de luz, o formato YCbCr permite a codificação das informações do vídeo de luminância (Y) em largura de banda total, enquanto para a crominância (Cb e Cr) pode-se utilizar a metade da banda da luminância. Um sistema que define a cor através de um valor de luminância e dois valores de crominância é chamado de sistema luma-croma.

Segundo Midha *et al*., (2014), a conversão de R, G, B para o espaço de YCbCr é a chave para a compressão sem perdas. A transformação pode ser realizada através de cálculos matemáticos, como exemplificado na tabela 1.

Tabela 1 - Cálculos de transformações de espaços de cores.

Transformação RGB para YCbCr
$Y = (0{,}299\ \times R) + (0{,}587\ \times G) + (0{,}114\ \times B)$
$Cb = (-0{,}16875\ \times R) - (0{,}33126\ \times G) + (0{,}5\ \times B) + 128$
$Cr = (0{,}5\ \times R) - (0{,}41869\ \times G) - (0{,}08131\ \times B) + 128$

Fonte: Adaptado de Midha (2014)

Existem diversas maneiras de relacionar os componentes de crominância com o componente de luminância na sub amostragem. Segundo Richardson (2004), os formatos mais comuns são o 4:4:4, o 4:2:2 e o 4:2:0.

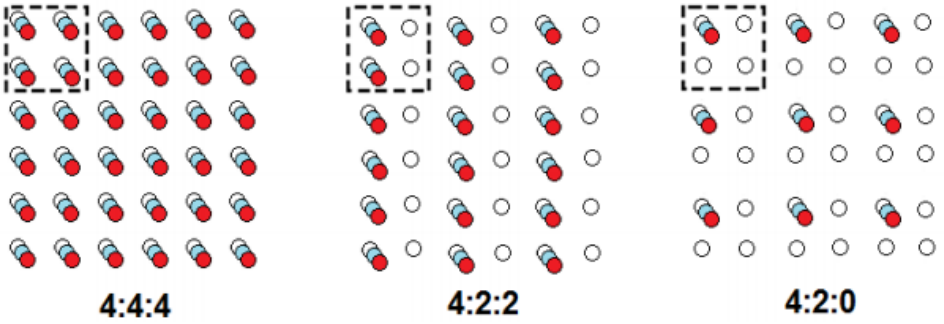
- **Formato 4:4:4**: Para cada quatro amostras de luminância (Y), existem outras quatro amostras de crominância azul (Cb) e outras quatro de crominância vermelha (Cr). Por este motivo os três componentes de cor apresentam a mesma quantidade de informação de luminância e crominância, equivalente ao modelo RGB. Neste caso, diz-se que a sub amostragem não foi aplicada.
- **Formato 4:2:2:** Para cada quatro amostras de luminância (Y) da direção horizontal, existem apenas duas amostras de crominância azul (Cb) e duas de crominância vermelha (Cr). Neste caso, as amostras de crominância possuem a mesma resolução vertical das amostras de luminância, mas possuem metade da resolução horizontal.
- **Formato 4:2:0:** Para cada uma das quatro amostras de luminância (Y), são associados alternadamente apenas uma amostra de crominância azul (Cb) e

> uma amostra de crominância vermelha (Cr). Neste caso, as amostras de crominância possuem metade da resolução horizontal e metade da resolução vertical do que as amostras de luminância.

Segundo informações coletadas no suporte da Microsoft, e considerando que cada componente de cor do *pixel* originalmente era representado em 8 *bits*, na sub amostragem no 4:4:4 são necessários 24 *bits*, enquanto na 4:2:2 são necessários, em média, 16 *bits* para representação de cada *pixel*. Já a sub amostragem 4:2:0 necessita apenas de 12 *bits* por *pixel*, em média. Quanto menor a quantidade de *bits* por *pixel* em cada sub-amostragem, maior a redução da crominância do vídeo[5]. As sub-amostragens 4:4:4 e 4:2:2 são usualmente aplicadas a codificação de vídeo nos últimos padrões quando se trata do enfoque na alta fidelidade dos vídeos, permitindo assim, maior resolução de crominância e maior precisão de bits por amostras, como cita Hung (2007).

Assim, uma imagem pode ser dita como uma sobreposição de três imagens, sendo elas, uma preta e branca e duas com as cores, no formato YUV, como pode ser visto na figura 5.

Figura 5 - Sub Amostragem no formato YUV



Fonte: Afonso (2012)

A sub-amostragem, reduz o detalhe em componentes de cor (ou crominância), mas não implica em perda de nitidez a uma distância razoável de visualização (CARTAJENA, 2014). Segundo Rosa (2010), a sub amostragem de cor aumenta a eficiência da compressão, considerando que uma parte da informação da imagem é eliminada. De acordo com Afonso (2012), essas técnicas permitem economizar 33%

[5] https://support.microsoft.com/pt-br/kb/294880

no número de *bits* no formato YUV 4:2:2 e 50% o formato YUV 4:2:0 (TEIXEIRA, 2014).

2.2 Codificação de Vídeo

Sabe-se que um arquivo multimídia original requer um longo período de tempo para ser processado, além de necessitar de um grande espaço de armazenamento, bem como de uma grande largura de banda para a transmissão (MIDHA *et al*., 2014). Conforme Rosa (2010), com isso surge a necessidade de utilização de técnicas de compressão de vídeo, com foco na redução da quantidade de dados a ser transmitida e/ou armazenada.

No contexto da codificação de vídeo, Richardson (2004), diz que é possível classificar a compressão de dados em duas técnicas:

- **Com perdas de informação (*lossy compression*):** esta técnica codifica os dados, de modo que a reconstrução dos mesmos resulta em alguns dados distintos do original, porém é possível alcançar altas taxas de compressão. Esse processo admite perdas de informação em seu processo de codificação que são imperceptíveis ao olho humano.

- **Sem perdas de informação (*lossless compression*):** esta técnica codifica os dados de maneira que possam ser reconstruídos de forma idêntica aos dados originais de antes da codificação, ou seja, não admite perdas de informação.

2.2.1 Redundâncias

Diz-se que uma cena é amostrada em um ponto de maneira a produzir um quadro (*frame*), sendo o quadro, a representação de uma cena visual completa em certo ponto no tempo. Cada amostragem se repete em intervalos periódicos, para produzir a sensação de movimento. Uma cena de vídeo natural é composta por múltiplos objetos onde, cada um deles apresenta característica individual (como por exemplo, forma, profundidade, textura e iluminação) (MIDHA, 2014).

A captura de uma imagem define a amostragem espacial do vídeo, sendo que, quanto mais *pixels* esta imagem é representada, implica em uma maior resolução

desta. O intervalo de captura entre uma imagem e outra é definido por amostragem temporal, e da mesma maneira, quanto maior a taxa de captura das imagens, ou seja, quanto maior a quantidade de capturas de imagens em um certo tempo, tem-se uma cena mais suave em relação ao movimento de um quadro para o outro (RICHARDSON, 2004).

Pelo fato da amostragem se repetir em intervalos periódicos e pela grande quantidade de *pixels* presentes em cada quadro, esse processo apresenta redundâncias, que são conceituadas como dados que não contribuem com novas informações relevantes para representação da imagem (AGOSTINI, 2007). Segundo Richardson (2004), a compressão de dados é aplicada através do tratamento dessas redundâncias de imagens de um vídeo, isto é, tratar a imagem de maneira a elencar os componentes que se julgam necessários para a representação fiel dos dados. Para se obter a compactação sem perdas, ou seja, com que os dados reconstruídos na saída do decodificador sejam uma cópia fielmente perfeita dos dados originais, diz-se que o dado apresenta redundâncias estatísticas. No entanto, com este tipo de compactação, não é possível alcançar um nível de compressão eficiente e, dessa maneira, a compressão com perdas é necessária para tornar possível o alcance de uma maior qualidade de compressão.

Sistemas de compressão de vídeo com perdas se baseiam no princípio da remoção de redundância subjetiva, que se refere aos elementos de uma imagem ou uma sequência de vídeo que podem ser removidos sem afetar significantemente a percepção humana de qualidade visual.

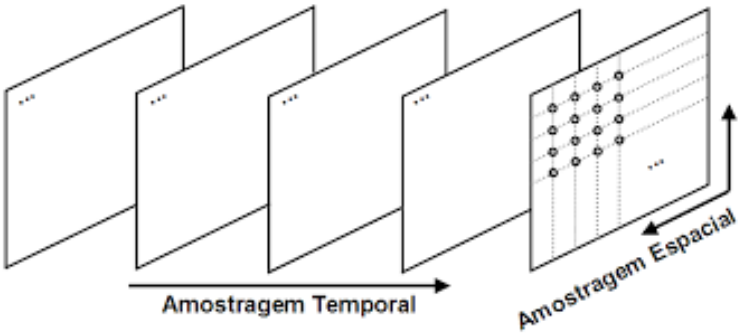
Grande parte dos métodos de codificação de vídeo explora os dois tipos de redundâncias, onde elas são denominadas temporais e espaciais (DINIZ, 2009):

- **Redundâncias Temporais:** No domínio do tempo, é observável a alta correlação (semelhança) entre quadros de vídeo que foram capturados em uma sequência de tempo. Temporalmente, diz-se que os quadros adjacentes (quadros sucessivos na ordem do tempo) são diversas vezes altamente correlacionados, especialmente se a taxa de amostragem temporal (taxa de quadros) é alta. Também é chamada de redundância inter-quadro.
- **Redundâncias Espaciais:** No domínio espacial, diz que há uma enorme correlação entre os *pixels* (amostras) que estão localizados de maneira

próxima, ou seja, com outras palavras, os valores de amostras vizinhos são frequentemente muito semelhantes.

Estes dois tipos de amostragem, tanto no tempo, como no espaço, são indicados na figura 6.

Figura 6 - Imagens capturadas em sequência



Fonte: Repositório Institucional Universidade de Brasília

O processo de captura de imagens demonstrando movimento, envolve amostrar o espaço e o tempo. Sendo assim, de acordo com Hung (2007), em regiões contíguas de imagem normalmente não apresentam grandes alterações, o mesmo ocorre para a mesma região em quadros diferentes em instantes de tempo distintos. Como consequência disso, as amostras espaciais e temporais de um vídeo apresentam alta correlação entre si, sendo este um fator que implica em melhoria de eficiência em um vídeo comprimido.

2.2.2 Compressão de Vídeo Digital

Para Silva (2014), a codificação de vídeo pode ser abstratamente descrita como um processo de análise de quadros e regiões dentro deste, buscando informações redundantes que de alguma maneira podem ser comprimidas através da exploração de redundâncias.

O termo genérico “codificação de vídeo”, define o processo de compressão e descompressão de um sinal. A compressão de vídeo é um processo também conhecido como compactação de dados, apresentando a finalidade de reduzir o número de *bits* de sua representação, como já mencionado. Um par complementar de sistemas é utilizado no processo de compressão, sendo ele composto por um

compressor (codificador) e um descompressor (decodificador). O codificador de vídeo tem como funcionalidade, converter os dados de entrada em uma forma de representação menor antes da realização da transmissão ou o armazenamento do mesmo, e isto acaba gerando uma cadeia de *bits* (*bitstream*), enquanto o decodificador converte o dado da forma reduzida de volta à uma representação dos dados no formato de vídeo original. Este par (codificador e decodificador) é frequentemente dito como um *codec* (COdificador/DECodificador).

Segundo Beltrão *et al*., (2011), nos dias atuais, acredita-se que as técnicas de codificação de vídeo forneçam soluções eficientes para a representação de dados de vídeo com uma forma mais compacta e robusta, impactando positivamente no que se refere ao fator de armazenamento e transmissão, ou seja, implicando em menores custos em termos de largura de banda e consumo de energia.

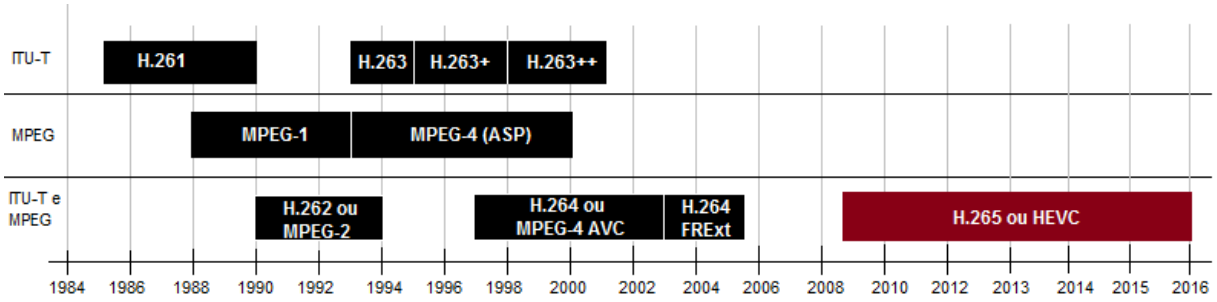
2.2.3 Padronização das Técnicas de Compressão de Vídeo

Tendo em vista as principais aplicações do vídeo digital, sabe-se que tanto em questão de armazenamento, como de transmissão são utilizados diversos dispositivos eletrônicos com esta finalidade. Como dito por Manoel (2007), para tornar possível a interoperabilidade entre os inúmeros dispositivos de fabricantes distintos utilizando técnicas de compactação de imagem, é necessário utilizar uma padronização dessas técnicas, de maneira que possuam especificações em comum, no que compete à compressão e descompressão do dado.

O impulso para a padronização das técnicas de codificação de vídeo digital se deu em torno do ano de 1980. Os principais órgãos envolvidos neste processo de padronização são o ITU-T *Video Coding Experts Group* (VCEG) e o ISO/IEC *Moving Picture Experts Group* (MPEG). O ITU-T era usualmente conhecido por Comitê Internacional Responsável pela Telegrafia e Telefonia (CCITT - *Comité consultatif international téléphonique et télégraphique*). O MPEG é um grupo de projeto das instituições ISO (*International Organization for* Standardization) e IEC *(International Eletrotechnical Comission*) (MANOEL, 2007).

Na figura 7, está representado um mapa cronológico de diversos padrões de codificação desenvolvidos pelos órgãos envolvidos no processo, tanto de maneira conjunta, como individualmente.

Figura 7 - Evolução dos padrões



Fonte: Própria autora (2016)

Uma diferença significativa entre as diferentes gerações de padrões de codificação de vídeo é que eles fornecem conjuntos distintos de codificação para um bloco de amostras. Porém, existe uma semelhança de característica entre os últimos padrões de codificação, que é a abordagem de codificação híbrida, baseada em blocos (SZE, 2014).

O modo de codificação determina se o bloco de amostras é comprimido explorando redundância espacial ou redundância temporal. Para um conteúdo ser codificado, os quadros de vídeo são particionados em blocos a serem codificados individualmente. Sendo assim, um *codec* é contemplado por diversos módulos, sendo cada um responsável por aplicações referentes compressão e descompressão do vídeo processado.

A eficiência de um padrão de codificação híbrido baseado em blocos depende de vários aspectos de projeto. No entanto, a principal fonte de melhoria entre um padrão e outro, é o aumento do número de possibilidades suportadas para codificação de uma imagem ou um bloco de amostras.

De acordo com a ITU-T (2015), as normas de codificação a partir do H.262 até o H.264 são utilizados atualmente para armazenar e transmitir conteúdo de vídeo de alta definição (HD – *High Definition*). No entanto, devido à popularidade do vídeo de alta definição e o crescimento exponencial pelo *Ultra High Definition* (UHD), com resoluções de luminância cada vez mais altas, surgiu o projeto do H.265, com foco em vídeos de alta resolução. Embora, a codificação de vídeo HD e UHD sejam um aspecto relevante ao projeto do HEVC, o padrão foi desenvolvido com foco em fornecer uma melhoria na eficiência de codificação em relação ao seu antecessor o H.264/AVC em todas as aplicações de codificação de vídeo existentes (SZE *et al*., 2014).

2.2.4 High Efficiency Video Coding – HEVC

Progressos significativos foram alcançados na área de codificação de vídeo, o que pode ser constatado com o último padrão lançado, considerando o estado da arte em codificação de vídeo, o H.265 ou também denominado como Codificação de Vídeo de Alta Eficiência (*High Efficiency Video Coding*) ou simplesmente HEVC. Este padrão alcançou a aprovação da norma em 2013, como ITU-T H.265 e ISO/IEC 23008-2. Ele também é particionado em extensões de desenvolvimento, de modo que o responsável por desenvolver as extensões para este padrão é a Equipe Colaborativa Conjunta sobre Codificação de Vídeo (*Join Collaborative Team on Video* - JCT-VC) (SULLIVAN *et al*., 2013). O escopo deste grupo foi estendido para continuar trabalhando em extensões de distribuição *Range Extension Format* (RExt), *Scalable* HEVC (SHVC) e de Conteúdo de Codificação de Tela (SCC – *Screen Content*). Já a Equipe Colaborativa Conjunta em Desenvolvimento de Extensão de Codificação de Vídeo 3D (JCT-3V) foi estabelecida para trabalhar em *multiview* e codificação de extensões do HEVC e outros padrões de codificação de vídeo em 3D. O Software de Referência do padrão foi desenvolvido juntamente com o projeto da norma, ele é caracterizado como sendo um modelo de teste do HEVC, recebendo a alcunha de HM[6].

O HEVC visa equacionar dois problemas principais, um referente à resolução do vídeo e o outro em relação ao uso de arquiteturas de processamento paralelo. Como consequência disso, ele apresenta ganho significativo em relação ao padrão de codificação de vídeo anterior, o H.264, onde este ganho é estimado em torno de 50% de melhoria em sua eficiência de compactação, sem apresentar perdas significativas de qualidade da imagem à visualização humana (SULLIVAN, 2012). Ou seja, este processo, reduz em até a metade a taxa de *bits* para representar o vídeo, utilizando informações redundantes do mesmo.

De acordo com Minallah *et al.* (2015), o HEVC define uma sintaxe padrão para simplificar a implantação e maximizar a interoperabilidade. Todavia, certos aspectos da criação deste padrão requerem mais processamento em comparação com o padrão antecessor (H.264). Aspectos como o aumento do tamanho dos blocos de

[6] HEVC Test Model, HM 16 [Online]
Available https://hevc.hhi.fraunhofer.de/svn/svnHEVCSoftware/tags/HM-16.6.

transformadas, tamanho das entradas de transformadas e o processo de predição são fatores que exigem mais processamento neste padrão em relação ao H.264.

Conforme os dados retirados do repositório digital do HEVC, o padrão está em sua terceira versão:

- A primeira versão do padrão foi finalizada em abril de 2013.
- A segunda versão incluindo extensões *rext*, SHVC e MV-HEVC foi finalizado em outubro de 2014.
- A terceira versão incluindo extensão 3D-HEVC foi finalizada em fevereiro de 2015.

2.2.4.1 Taxa de Amostragem

O HEVC apresenta três tipos de perfis com diversos requisitos de aplicação, que são denominados como sendo: *Main, Main 10 e Main Still Picture*. Esses perfis definem a sintaxe e as características da codificação que pode ser utilizada em determinado conteúdo de vídeo (SZE, 2014).

- ***Main*:** Este perfil representa os dados de vídeo com 8 *bits* por amostra e utiliza a representação típica com um sinal no formato 4:2:0.
- ***Main* 10:** Este perfil varia sua representação dos dados de vídeo entre 8 e 10 *bits* por amostra, fornecendo maior fidelidade de cores.
- ***Main Still Picture*:** Este perfil é um subconjunto de capacidades do Perfil Main, normalmente, utilizado para extração de sequências de vídeo instantâneas ou para fotografia em câmeras.

Houve uma redução da quantidade de perfis em relação a padrões anteriores, fornecendo assim, um valor máximo de interoperabilidade entre dispositivos. Os perfis variam de acordo com a profundidade de *bits* permitidas por cor. Neste contexto, níveis de capacidade são definidos para estabelecer a resolução da imagem, a taxa de quadros, a taxa de bits, entre outros aspectos. A especificação do padrão, atualmente, define 13 níveis, que são conjuntos de restrições que indicam o desempenho necessário para produzir o *bitstream* do perfil especificado.[7]

[7] Repositório Digital < http://pt.tvyvideo.com> Acessado em maio de 2016.

Em conformidade com o relatório técnico da ITU-T (2015), existe uma grande variação da quantidade de quadros por segundo para cada nível de *bitstream* referente ao codificador do padrão H.265. Os níveis, por sua vez, podem ser classificados de acordo a sua resolução, as resoluções mais baixas são compostas pelos níveis de 1 a 3 e resoluções mais altas, incluem os níveis de 4 à 6.2, que definem os requisitos para resoluções mais altas, como por exemplo um vídeo no formato 8K. Esse formato representa uma resolução UHD, referindo-se à resolução horizontal, apresentando 8000 *pixels,* sendo esta 16 vezes maior do que a resolução Full HD. As especificações de cada nível em relação a taxa de amostragem (imagens por segundo) são apresentadas na tabela 2.

Tabela 2 - Classificação dos níveis em relação a taxa de amostragem

Nível	**Resolução Mínima (*pixels*)**	**Resolução Máxima (*pixels*)**	**Taxa de Amostragem para Resolução Mínima (blocos/segundo)**	**Taxa de Amostragem para Resolução Máxima (blocos/segundo)**
1	128x96	176x144	33	15
2	128x96	352x288	225	30
2.1	128x96	640x360	300	30
3	128x96	960x540	300	30
3.1	128x96	1280x720	300	33
4	128x96	2048x1080	300	30
4.1	128x96	2048x1080	300	60
5	128x96	4096x2160	300	30
5.1	128x96	4096x2160	300	60
5.2	128x96	4096x2160	300	120
6	128x96	8192x4320	300	30
6.1	128x96	8192x4320	300	60
6.2	128x96	8192x4320	300	120

Fonte: Adaptado de ITU-T (2015)

2.2.4.2 Particionamento por Árvore de Codificação

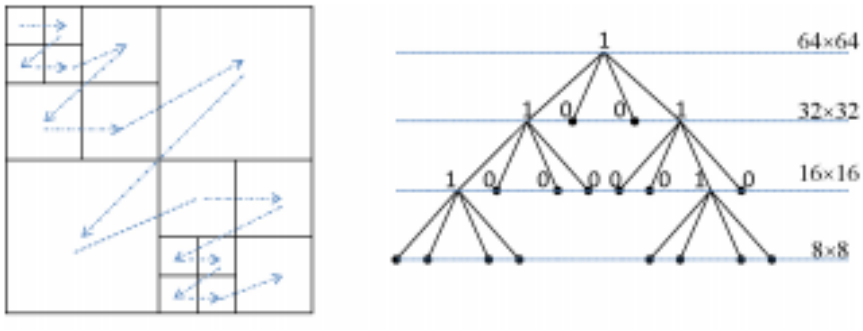
Um dos fatores essenciais para o ganho de desempenho no HEVC foi a introdução de estruturas de bloco maiores, apresentando mecanismos flexíveis e de maneira hierárquica da forma de particionamento dos blocos, em um esquema denominado como *TreeBlock Partitioning*, comumente denominado como *Quadtree*. Essa estrutura é análoga ao macrobloco utilizado nos padrões anteriores, que consistia em um bloco de amostras 16x16 de luminância e dois blocos correspondentes de amostras de crominância (SULLIVAN *et al*., 2013).

O HEVC é dividido em várias CTUs (*Coding Tree Unit*) que por definição podem variar seu tamanho de 8x8 até 64x64. Dentro da CTU, existe uma estrutura *quadtree* construída, permitindo mais flexibilidade para o particionamento da mesma. Dessa maneira, cada nó folha da árvore de codificação é denominada Unidade de Codificação (CU – *Coding Unit*), que define qual tipo de predição será aplicada. Cada CU pode ter várias Unidades de Predição (PU – *Prediction Unit*) e Unidades de Transformadas (TU – *Transform Unit*) (KIM, 2012).

A norma deste padrão define que cada CTU pode ser codificada de maneiras distintas, sendo dividida recursivamente em quatro CUs menores e de mesmo tamanho até chegar-se na última camada. Conforme Kim (2013), cada CU permite flexibilidade, com a finalidade de alcançar a melhor relação possível entre a eficiência de compressão e a qualidade de representação, sendo que cada CU pode ser subdividido em PUs durante o processo de predição e em TUs na fase de transformadas. Segundo Beltão (2011), essa possibilidade de escolha arbitrária do tamanho da CU permite uma otimização do *codec* para vários conteúdos, aplicações e dispositivos.

A divisão da CTU em CUs é feita com o intuito de separar regiões distintas de um quadro com redundâncias que podem ser exploradas pelo processo de predição. Este processamento ocorre de forma recursiva, como consequência uma CTU de 64x64 é dividido em quatro CTUs de 32x32, de maneira que cada um deles, pode ser subdividido em 16x16 CTUs. Esse processo é repetido até que a CU de menor tamanho seja alcançada, neste caso, 8x8, ou quando uma condição de término seja satisfeita, formando implicitamente uma *quadtree* (KIM, 2013).

Figura 8 - Exemplo de partição e ordem de processamento de CTU

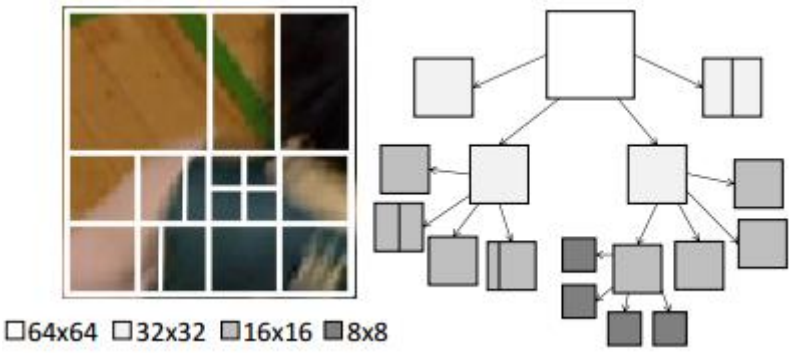


Fonte: Kim (2012)

Na figura 8, o bloco representa uma CU, sendo que uma CTU é dividida em 16 CUs, que apresentam diferentes tamanhos e posições. A estrutura em árvore, mostra a codificação da estrutura por partição CTU do bloco. A numeração na árvore diz se a CU ainda é dividida, e a ordem de processamento de cada CU é dada pela linha pontilhada do bloco.

Cada subdivisão se refere a raiz da primeira estrutura do bloco de particionamento *quadtree*, que são por definição, os blocos de codificação (CTBs – *Coding Tree Blocks*) equivalentes às CTUs. Outro exemplo deste particionamento pode ser visto como segue na figura 9, onde é possível perceber que cada subdivisão pode apresentar variações no dimensionamento de tamanho, como por exemplo 64x32, 64x48, entre outras variações (BOSSEN *et al*., 2016).

Figura 9 - Partição e dimensionamento na estrutura *quadtree*



Fonte: Silva (2014)

Estes CTBs apresentam número idêntico para ambos os componentes de luminância e crominância de imagem (assumindo um formato de vídeo não-monocromático[8]).

Quanto maior o tamanho da CTU, melhor a eficiência alcançada na codificação, porém, isso pode influenciar no aumento do atraso do *codec*, aumentar os requisitos de memória e a complexidade computacional do processo de codificação. Mas esses aumentos podem ser superados, pois o codificador pode determinar o tamanho da CTU, com foco na aplicação alvo (SZE *et al*., 2014).

2.2.4.3 Blocos Operacionais do padrão HEVC

Seguindo o propósito de codificação de vídeo híbrida baseada em blocos, a figura 10 apresenta o fluxo de codificação, onde cada imagem é subdividida em blocos na Unidade de Árvore de Codificação e, posteriormente, é feita a decisão sobre o processo de predição a ser utilizando, onde as redundâncias são exploradas.

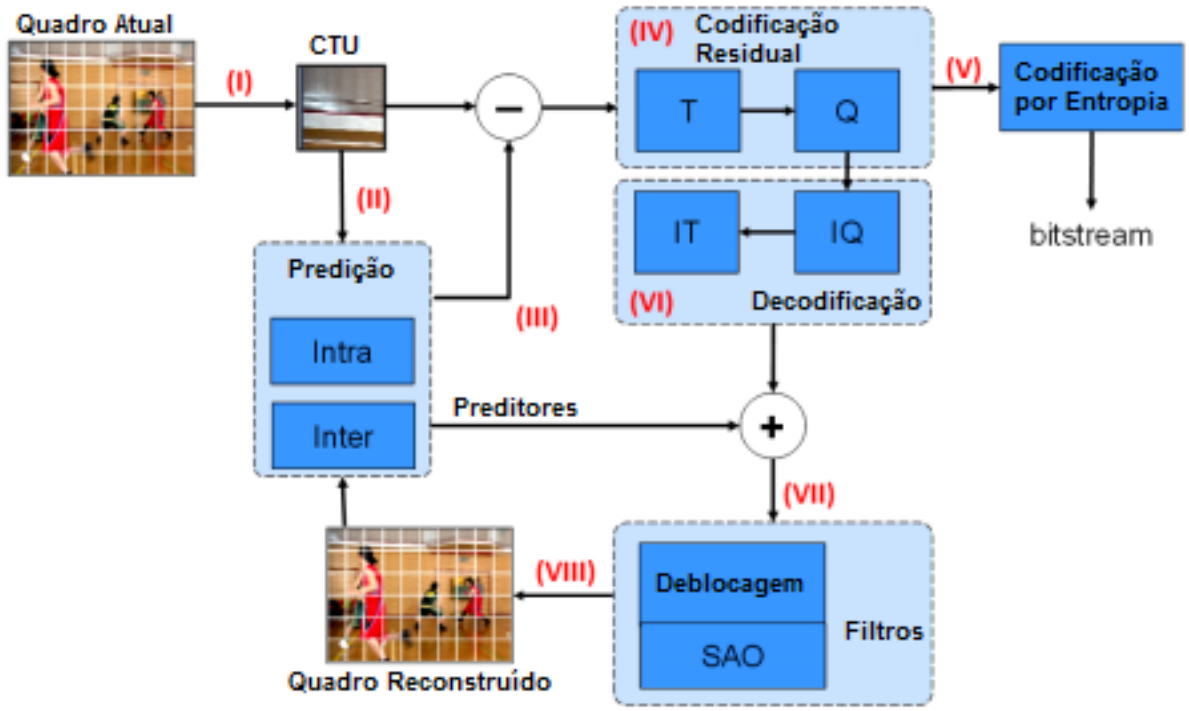
Nesta etapa, é aplicado a cada um dos quadros a predição intra quadro ou inter quadro. O quadro gerado pela predição é então diferenciado do quadro original de entrada na CTU, produzindo um erro, que por sua vez, é conceituado como resíduo. Esses resíduos são enviados para o bloco de Transformadas, onde são codificados através de redundâncias e posteriormente é aplicado o processo de Quantização nos coeficientes de transformada, resultando no descarte de informações menos relevantes para o sistema visual humano. Na saída gerada pelo bloco de Quantização, além dos demais elementos sintáticos (isto é, dados vindos das demais etapas anteriores do codificador, como inter predição por exemplo) é aplicada a codificação de entropia, no intuito de reduzir ainda mais o *bitstream* gerado pelo codificador de vídeo digital, a fim de utilizar essa corrente de bits final para armazenamento ou transmissão.

Por sua vez, os dados gerados pela Quantização também são enviados para um estágio de decodificação, que apresentam como principal funcionalidade, reconstruir o resíduo e adicioná-lo às amostras processadas pelo bloco de predição. O bloco resultante, é filtrado, com a finalidade de remover artefatos visuais gerados nesse processo de codificação, que insere perdas. Após a filtragem, esta informação é

[8] Que apresenta somente uma cor, sem variação.

armazenada como um quadro reconstruído que será utilizado como referência para as próximas iterações do processo de codificação (SULLIVAN, 2013).

Figura 10 - Laço de Codificação do padrão H.265



Fonte: Adaptado de Silva (2014)

3 MÓDULO DE TRANSFORMADAS

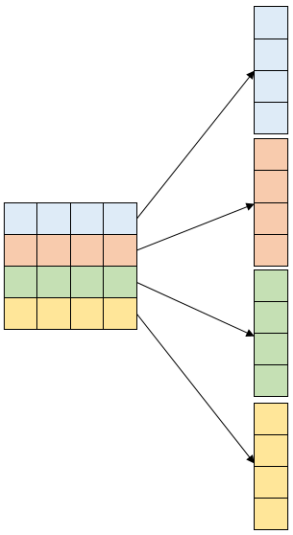
O resíduo produzido pela diferença entre o quadro da CTU atual e o quadro resultante do processo de predição é enviado para o processo de codificação que pode ser referenciado como codificação residual onde, durante esse processo, as transformadas são aplicadas aos blocos.

3.1 Módulo de Transformadas

De acordo com Teixeira (2014), o processo de transformada é responsável por traduzir o número de *bits* de *pixel* para o domínio da frequência. Isso é feito dividindo a imagem em quadros e executando uma função sobre todos os *pixels* residuais do quadro, ou seja, sobre cada um desses blocos é aplicada uma função de transformada.

No cenário da codificação de vídeo, as transformadas são aplicadas ao *pixel* residual resultante da predição inter ou intra quadro. No codificador, o quadro de uma imagem é dividido em blocos quadrados de tamanho NxN, onde $N = 2^M$, e M é um inteiro. Cada bloco residual é entrada para as transformadas diretas de duas dimensões NxN. As transformadas de duas dimensões podem ser implementadas como transformadas separáveis pelo uso de N-pontos em uma dimensão da transformada para cada linha e cada coluna separadamente, como representado na figura 11.

Figura 11 – Divisão da codificação residual para as Transformadas



Fonte: Própria autora (2016)

Esse procedimento resulta em NxN coeficientes da transformada (*coeff*), que são então submetidos posteriormente ao processo de quantização (SZE, 2014).

Segundo Kalali (2014), as TUs para a Transformada Discreta do Cosseno (DCT – *Discrete Cosine Transform*) no padrão H.264 realizavam processamento em apenas tamanhos de TUs de 4x4 e 8x8. No entanto, no padrão H.265 é de comum utilização tamanhos de 4x4, 8x8, 16x16 e 32x32, que são aproximações de precisão finita à transformada. Com isso, o HEVC ganha em melhor compactação e em consumo de energia. O HEVC também utiliza a Transformada Discreta do Seno (DST – *Discrete Sine Transform*) para tamanho 4x4, em certos casos de predição intra quadro.

O fato do padrão abranger vários tamanhos de transformadas, que variam de $4 - pontos$ até $32 - pontos$, melhora a eficiência da codificação, porém, implica em um aumento da complexidade de implementação da arquitetura (SZE *et al*., 2014). Segundo Wang *et al*. (2014), para alcançar uma boa eficiência computacional, são utilizadas operações de multiplicação, adição/subtração e operações de deslocamentos.

3.1.1 Transformada do Cosseno Discreta (DCT)

Uma transformada comumente usada é a Transformada Discreta do Cosseno (DCT), já mencionada, baseada na Transformada de *Fourier* e é muito aplicada em sua forma original, podendo também apresentar modificações que reduzem a complexidade de implementação e uma melhor compressão (TEIXEIRA, 2014).

Os N_s coeficientes de transformadas de uma N-pontos de transformada discreta do cosseno em 1D aplicados às amostras de entrada u_i podem ser expressados como na equação (1):

$$v_i = \sum_{j=0}^{N=1} u_j c_{ij} \tag{1}$$

Onde $i = 0, \dots, N-1$. Os elementos c_{ij} da DCT da matriz transformada são definidos na equação (2).

$$c_{ij} = \frac{P}{\sqrt{N}} cos\left[\frac{\pi}{N}\left(j + \frac{1}{2}\right) i\right] \tag{2}$$

Onde $i, j = 0, \dots, N-1$ e onde P é igual a 1 e a $\sqrt{2}$, para $i = 0$ e $i > 0$, respectivamente.

As transformadas maiores contribuem entre 6,7% - 10,1% de redução da taxa de *bits* para resolução de 1920x1080 e sequências de vídeo maiores, e para as transformadas menores, aumentam a precisão em 0,3% - 1,24%. O HEVC proporciona vários novos recursos, no entanto, levanta diversos desafios de implementação em hardware, tais como (TIKEKAR *et al*., 2014):

- O padrão utiliza unidades de trabalho, como as TUs que variam de 4x4 à 32x32 *pixels* residuais. Essa variação de tamanho, implica em uma lógica de controle mais complexa, sendo que cada TU pode assumir diferentes números de ciclos para o seu processamento.
- Da mesma forma, como no H.264/AVC, a transformada 2D no HEVC é particionada em transformadas 1-D ao longo de colunas e linhas. Esse processo aumenta o número de multiplicações, visto que as operações são replicadas. Além disso, com o aumento de precisão da transformada no HEVC, dobra o custo de cada multiplicação. Sendo assim, a complexidade computacional é afetada, tanto em relação a área como em energia.

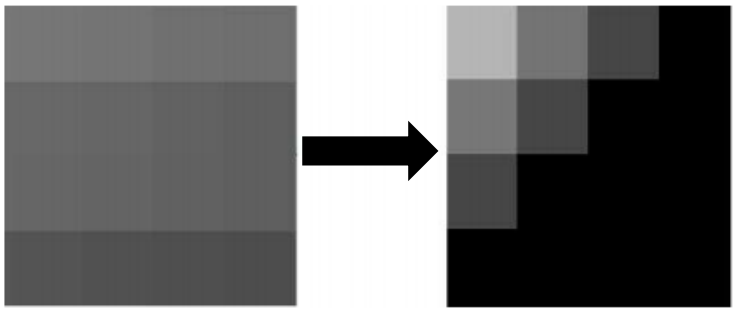
Os coeficientes de multiplicação das transformadas utilizados nas multiplicações de matrizes, são derivados da equação (3):

$$d_{ij}^{N} = d_{i\left(\frac{32}{N}\right),j}^{32} \tag{3}$$

Onde $i, j = 0, \ldots, N-1$, e N representa a própria matriz de transformada, podendo ser N= 4, 8 ,16 ou 32.

Através dos cálculos das Transformadas utilizando a estrutura citada acima, os valores dos *pixels* são organizados como demonstrado na figura 12.

Figura 12 - Organização dos bits após o processo de transformada



Fonte: Ramos (2010)

Os coeficientes superiores e à esquerda da matriz da transformada têm os valores mais altos em relação aos inferiores e à direita. Como consequência disso, o cálculo da transformada resulta em valores que necessitam de maior representatividade de *bits* para serem codificados do que o bloco original (TEIXEIRA, 2014). Entretanto, a forma que ele reorganiza os resíduos facilita e melhora a eficiência da codificação de entropia, que é o passo final no codificador completo HEVC.

3.1.2 Trabalhos Correlatos

A partir de uma pesquisa exploratória foi possível elencar as variáveis de interesse de cada autor buscado na literatura, para fins de análise e comparação ao final deste trabalho. Na tabela 3, os trabalhos correlatos foram identificados de acordo com o seu nível (arquitetural ou algorítmico), em relação ao objetivo e em relação ao bloco de desenvolvimento.

Tabela 3 - Variáveis avaliadas pelos trabalhos correlatos

	Nível	Objetivo	Bloco
(a)	Arquitetural	Redução do caminho crítico	DCT
(b)	Arquitetural	Redução de área	DCT e IDCT
(c)	Algorítmico	Redução de área	DCT
(d)	Arquitetural	Redução de consumo de energia e área	IDCT
(e)	Arquitetural	Aumento de desempenho	DCT

Fonte: Própria autora (2016)

(a) Alto rendimento e baixo custo de hardware na DCT do padrão HEVC

Do *et al*., (2014), propôs alterações na implementação base do bloco de Transformadas apresentado no modelo de referência, visando a minimização do caminho crítico. Essa minimização se dá pela conversão de cálculos de multiplicações por deslocamentos seguidos de soma com reuso de componentes estruturados na estrutura em software denominada *Partitial Butterfly* presente no *software* de referência. Após as conversões, é realizada a organização das operações em uma árvore de adição para cada multiplicação, assegurando o menor caminho crítico. Ao final, reorganiza as operações para maximizar as operações em comum, reduzindo a

quantidade total de operações. Essas minimizações do caminho crítico são realizadas com a implementação da Otimização do Caminho Crítico (CPO – *Critical Path Optimization*).

O autor também utilizou a estrutura *Butterfly*, separando a arquiteturas em partes pares e ímpares, de acordo com a semelhança. Para as partes ímpares, foi utilizada a técnica do Menor Múltiplo de Constante de Multiplicação (MMCM) para substituir as multiplicações por adições com deslocamentos reduzindo o número de operações ou a decomposição das matrizes ímpares em matrizes esparsas facilitando a implementação.

De acordo com o autor, a estrutura da arquitetura com a utilização de CPOs alcança caminhos críticos menores em comparação com uma estrutura baseada no *software* de referência do padrão. Como resultados, também é apresentada uma redução de 80% do caminho crítico da menor transformada individualmente, a 4x4. Já para a 8x8, 16x16 e 32x32, foi alcançada uma redução de aproximadamente 17%, 25% e 22% nas respectivas transformadas. Com as reduções alcançadas, o autor conclui que também há redução de tempo de processamento. No entanto, não foi realizada a análise baseada em síntese, mas unicamente em componentes de alto nível, não apresentando resultados para ASIC, por exemplo.

(b) Unificação das Transformadas Diretas e Inversas

Segundo Budagavi *et al*., (2012) os núcleos de transformadas do padrão HEVC apresentam várias propriedades de simetria que podem ser aplicadas para reduzir o custo de implementação.

- Simetria par-ímpar (*even-odd*) na matriz de transformada.
- As transformadas das matrizes 16-pt, 8-pt e 4-pt são subconjuntos da transformada de 32-pt. Com isso, transformadas de menor porte estão embarcadas dentro da transformada maior, sem requerer implementação separada, necessariamente.
- A decomposição em e*ven-odd*, também é referida como sendo uma estrutura *Butterfly* parcial no desenvolvimento do HEVC de transformadas de N-pt de uma entrada N-pt, como segue: somar ou subtrair os valores de entrada, gerando valores intermediários $N - pt$.

- Calcular a parte par da saída usando o subconjunto de transformada $\frac{N}{2} \times \frac{N}{2}$ obtida a partir das linhas da matriz transformada.
- Calcular a parte ímpar da saída usando o subconjunto $\frac{N}{2} \times \frac{N}{2}$ obtida a partir das linhas ímpares da matriz transformada.

Em seu trabalho, o autor expõe a implementação de uma arquitetura DCT e IDCT (*Inverse Discrete Cossine Transform*) unificada no padrão HEVC. Para isso, foram aplicadas as propriedades de simetria existente entre as transformadas deste padrão, que possibilitaram a redução da área em torno de 45%.

Através da comparação de equações referentes às transformadas direta e inversa, percebeu-se que a parte par da matriz direta e inversa são idênticas e, da mesma maneira, através de análise de equações, chegou-se também à afirmação de que a parte ímpar da matriz direta e inversa são idênticas. Dessa maneira, foi possível desenvolver uma arquitetura unificada de transformadas que compartilha o mesmo bloco de hardware no seu processo. O projeto foi sintetizado para a tecnologia CMOS (*Complementary Metal Oxide Semiconductor*) de 45 nm, alcançando frequência de 250 MHz para o desenvolvimento unificado.

(c) Solução DCT em software

Conceição *et al.* (2012), propõe uma solução da Transformada Discreta do Cosseno de 32-pt no padrão HEVC em *software*. O objetivo do trabalho é apresentar um software que encontre entre cinco bilhões de possibilidades dentre as melhores combinações de operações usadas no processamento de DCT 1-D de 32-pt, a fim de obter o máximo de operações compartilhadas.

O autor desenvolveu a solução em *software* em linguagem de programação C, onde o mesmo foi particionado em sete módulos, os quais cada um representa um processo de busca pelo maior número de compartilhamento de operações dentre as dezesseis equações aplicadas a parte ímpar da transformada. A validação algorítmica foi realizada utilizando dados extraídos do modelo de referência do padrão. Os resultados apresentados neste trabalho pelo autor são parciais, pois não havia alcançado o processamento de todas as combinações, tendo em vista que existem mais de cinco bilhões de possibilidades. No entanto, com o *software* foi possível

alcançar apenas 37 combinações de operações, enquanto manualmente verificou-se uma combinação que resultou em 104 operações, pelo menos.

(d) Eficiência energética e alta vazão dos multi-tamanhos da Transformada Inversa Discreta do Cosseno no padrão HEVC

De acordo com Conceição *et al.* (2014), é possível combinar eficiência energética e alta vazão na IDCT no HEVC. O autor fez uso dos quatro tamanhos de transformadas utilizadas neste padrão no desenvolvimento de uma arquitetura de *hardware* capaz de alcançar processamento em tempo real, com 30 quadros por segundo. Neste trabalho foram utilizados vídeos de alta resolução e foi explorado o alto nível de paralelismo. Como parte dos objetivos do trabalho, também é apresentada uma alternativa de baixo custo em termos de consumo de *hardware* e dissipação de potência.

A arquitetura foi implementada de maneira puramente combinacional, utilizando abordagens de multiplicadores menores e aplicando um algoritmo de reutilização de operações e equações compartilhadas. A taxa de processamento de alta resolução foi alcançada por meio da exploração de paralelismo inerente ao algoritmo IDCT. Em relação ao baixo custo (consumo e dissipação de potência), o desenvolvimento utilizou um sistema integrado para todos os tamanhos de transformadas, possibilitando o ganho nessas variáveis.

A síntese da arquitetura foi feita para o FPGA (*Field Programmable Gate Array*) da Altera Stratix V e para ASIC com a tecnologia TSMC (*Taiwan Semiconductor Manufacturing Company*) 90 nm. Os resultados de síntese da arquitetura com FPGA apresentaram bons resultados, apresentando 25% de redução de hardware comparado à trabalhos da literatura, devido a abordagem dos multi-tamanhos de transformadas. Para ASIC, a arquitetura apresentou-se capaz de decodificar vídeos UHD a 30 quadros por segundo, além dos resultados de dissipação de potência entre 33,8 mW e 339,2 mW.

(e) Projeto de Hardware Modular da DCT no HEVC

Goebel *et al*. (2016), desenvolveu uma arquitetura da DCT com o objetivo de processar todos os tamanhos definidos no padrão HEVC com uma taxa de transferência constante de 32 coeficientes por ciclo, sendo capaz de lidar com

subdivisões homogêneas ou heterogêneas da CU. O projeto desenvolvido pelo o autor é capaz de realizar oito transformadas 4x4, ou quatro 8x8, ou duas 16x16, ou uma 32x32 em paralelo. A arquitetura também realiza operações individuais das transformadas, sendo uma por chamada.

A estrutura modular proposta pelo autor faz uso de dois blocos básicos caracterizados como “geral” e “*butterfly*”. No texto foi exemplificada a transformada 8x8, contendo seis blocos de *butterfly* e seis de geral, além de utilizar dez multiplexadores 2:1 para ser possível processar tanto transformadas 8x8 como 4x4. Na mesma estrutura também foi utilizado um somador seletivo (SA) na última etapa. Este somador é responsável por executar a soma que compõe a 8x8. Já para a maior transformada do padrão (32x32), são utilizados na arquitetura 86 blocos gerais e 30 blocos *butterfly*, juntamente com 180 multiplexadores 2:1 e 42 blocos SA. O SA da 32x32 e 16x16 pode conter somadores e subtratores definidos por análise matemática.

Para a validação da arquitetura, foi utilizado dados do *software* de referência (HM 16.0), aplicando-os como estímulos da arquitetura. Os valores foram obtidos a partir dos cinco primeiros quadros de cinco Condições de Teste Comum, sendo estas, sequências de vídeo HD de 1080 pontos. As sequências de vídeo utilizadas foram a *BascketballDrive*, *BQTerrace*, *Quimono*, *ParkScene* e *Cacto*. A estimativa de energia foi realizada considerando os Parâmetros de Quantização (QPs) de 22, 27, 32 e 37.

A síntese da arquitetura multi-modular foi realizada para ASIC com a ferramenta da Cadence RTL Compiler utilizando a tecnologia Nangate de 45 nm com biblioteca de células padrão. O consumo de energia total média da DCT alcançada no projeto foi de 24,2mW e um coeficiente de variação (CV) de 8,8%. No entanto, o valor de consumo médio obtido foi considerando o chaveamento default de 10% da ferramenta. A validação funcional foi realizada na ferramenta ModelSim, considerando os dados de entrada extraídos do Modelo de Referência (HM).

4 METODOLOGIA

A partir da fase decisória, envolvendo a escolha do tema desta pesquisa e definição do problema a ser abordado, chegou-se a fase construtiva, de modo a ser explanado o plano para a execução da base teórica até chegar-se na fase redacional, que se refere à análise da funcionalidade do que é proposto como desenvolvimento. Sendo assim, neste item será descrito um conjunto de abordagens técnicas e processos utilizados para formular a solução e resolver o problema proposto, de uma maneira sistemática, como apresentado na diagramação de atividades da Figura 13.

a) Estudo e Análise:

É vista como etapa inicial desta proposta a realização de uma pesquisa bibliográfica, acerca do padrão HEVC. Com esta pesquisa aplicada, obtém-se bibliografias relevantes ao tema, servindo de auxílio na implementação do Blocos de Transformadas.

A etapa seguinte consiste em uma pesquisa exploratória, envolvendo investigações sobre arquiteturas que utilizam Módulos de Transformadas no padrão HEVC. Dessa forma, o trabalho identifica quais são as alternativas de soluções mais apropriadas para aplicar no desenvolvimento desses módulos, visando sempre um baixo consumo de energia e manutenção de alto desempenho.

b) Desenvolvimento da arquitetura:

Como ponto de partida para a implementação do bloco funcional almejado, é utilizado o *Software* de Referência (HM) do padrão HEVC. Nele é aplicado um plano de estudo, utilizando o mesmo para a extração de dados. Com isso o modelo, contribui para a descrição VHDL da arquitetura e para a respectiva validação.

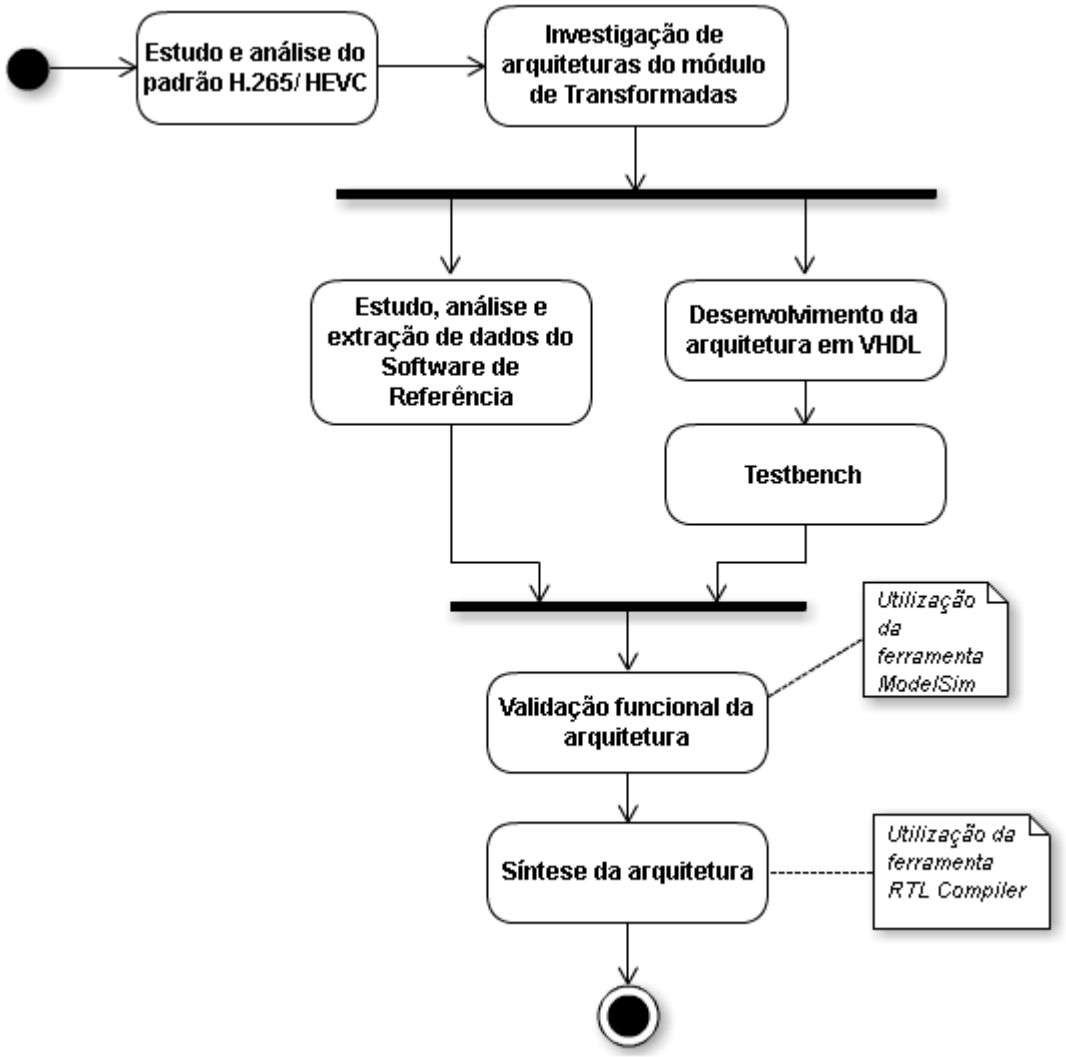
c) Simulação e Validação:

A etapa seguinte inicia um momento de pesquisa experimental, onde é feita a validação funcional da arquitetura projetada. Para isso, é desenvolvido o *testbench* para o bloco foco neste trabalho. Dessa maneira, o intuito geral é validar a arquitetura, fazendo uso dos dados de entrada coletados do *software* do padrão via sequências de vídeo reais, e realizando uma comparação de saídas geradas tanto no *software* quanto no *testbench* criado para a arquitetura. Com o resultado desta pesquisa quantitativa, o bloco pode ser validado através de simulação funcional, utilizando a ferramenta ModelSim da Mentor Graphics.

d) Síntese e Análise de Consumo

Com a validação concluída, a próxima etapa foi sintetizar a arquitetura para ASIC, de modo a verificar variáveis de interesse, tais como: frequência, caminho crítico e análise do seu consumo de energia. Para essa etapa, foi utilizada a ferramenta RTL Compiler da Cadence para a síntese e análise de desempenho da arquitetura para ASIC, fazendo uso da biblioteca de células da ST de 65 nm.

Figura 13 - Diagrama de atividades proposto



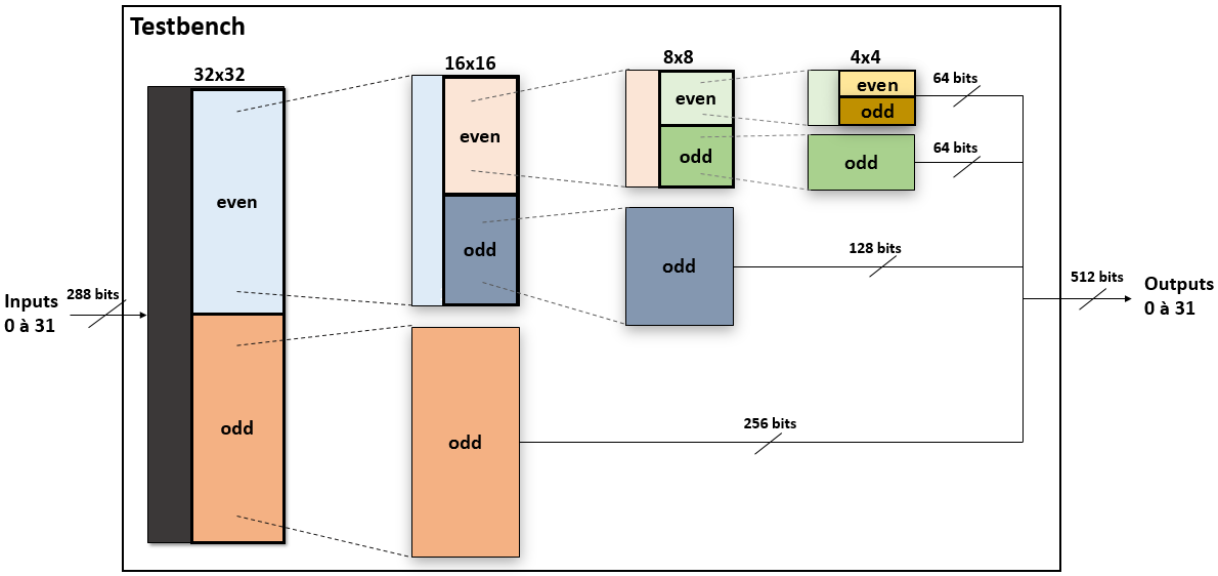
Fonte: Própria autora (2016)

5 ARQUITETURA DAS TRANSFORMADAS

O Bloco de Transformadas é uma parte crítica do *codec*, como cita Rithe *et al.* (2012), devido ao fato do aumento da eficiência de codificação estar fortemente relacionada com maior complexidade do módulo de transformadas, com seus multi-tamanhos. Com isso percebe-se que as arquiteturas desenvolvidas neste trabalho, implicam em um aumento de complexidade. No entanto, visam a reutilização de componentes, reduzindo o caminho crítico e o consumo de energia.

A estrutura *Butterfly* presente no *software* de referência foi desenvolvida no padrão HEVC com o intuito de reduzir componentes, sendo assim, ela elenca semelhanças entre as matrizes de transformadas e como consequência decompõe a matriz em uma parte par (*even*) e outra parte ímpar (*odd*), que juntas contemplam o *Partial Butterfly* (PB) indicado no modelo do *codec* HEVC. A estrutura par se refere às propriedades de simetria encontradas entre partes das matrizes de tamanhos distintos e a parte ímpar, compete a cálculos dependentes apenas da transformada atual. O fluxo da arquitetura é expresso na figura 14, onde as entradas são representadas pelos resíduos expressos por *Inputs* e as saídas por *Outputs*.

Figura 14 - Fluxo de partição das transformadas



Fonte: Própria autora (2016)

A troca de cores entre as transformadas informa a dependência das menores em relação às maiores. A parte inferior e independente da arquitetura implica nos cálculos realizados na parte ímpar, indicado por "*odd*" na figura.

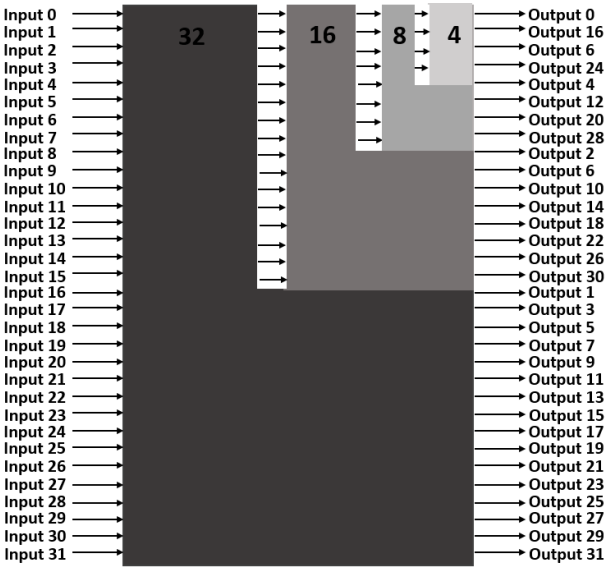
Para este trabalho foram consideradas algumas propriedades de simetria que auxiliam na redução de custo de implementação, citadas no trabalho de Budagavi *et al*. (2012), que estão descritas na subseção 3.1.2. Para a decomposição da parte par e da parte ímpar da *butterfly*, foram aplicados genericamente três etapas, como o mesmo autor também expõe em seu trabalho, considerando transformadas de $N - pt$ de uma entrada $N - pt$.

O projeto de transformadas, foi implementado em três etapas, em um primeiro momento foi analisada e implementada a estrutura genérica que contempla a DCT 1D caracterizada no modelo em *software* do padrão. A etapa seguinte visou o desenvolvimento de uma arquitetura baseada na proposta de Do *et al*. (2014), a fim de analisar por meio da síntese as variáveis área e dissipação de potência, pois o autor apresenta apenas resultados considerando o caminho crítico. Contudo, percebeu-se a expressiva quantidade de operações que não necessitariam estar em funcionamento em determinadas execuções desta arquitetura, fazendo com o que o consumo energético não seja eficiente. Sendo assim, optou-se por desenvolver uma arquitetura similar, porém, capaz de desativar operadores desnecessários para o instante da execução.

As três versões da DCT implementadas contemplam a ideia de redundância e de encapsulamento de transformadas para fins de análise comparativa das variáveis de interesse. Foram utilizados *shifts left* e somadores/subtratores para a implementação da DCT, com o intuito de almejar a melhor eficiência da arquitetura, visto que essas operações resultam em um circuito menor e mais rápido quando comparado a um circuito multiplicador.

A alternativa de encapsular os blocos de transformadas, como indica a figura 15, reutiliza o hardware já implementado em transformadas menores, de maneira que essas transformadas são componentes instanciados nas maiores.

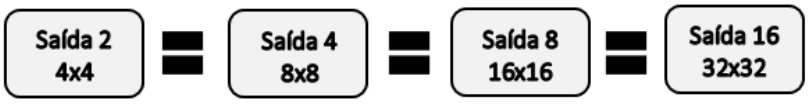
Figura 15 - Arquitetura embarcada



Fonte: Própria autora (2016)

As saídas, como é possível notar na mesma figura 15, apresentam-se de maneira desordenada. Isso ocorre, devido ao fato das transformadas menores representarem exatamente as saídas pares das maiores. Dessa maneira, as saídas das matrizes residuais são múltiplos de 2, ou seja, cada saída da transformada deve-se multiplicar por dois a ordem de saída. Para exemplificar, é possível visualizar a saída 2 da transformada 4x4, como exposto na figura 16, e qual seria a respectiva saída para as demais transformadas.

Figura 16 - Equivalência de saídas



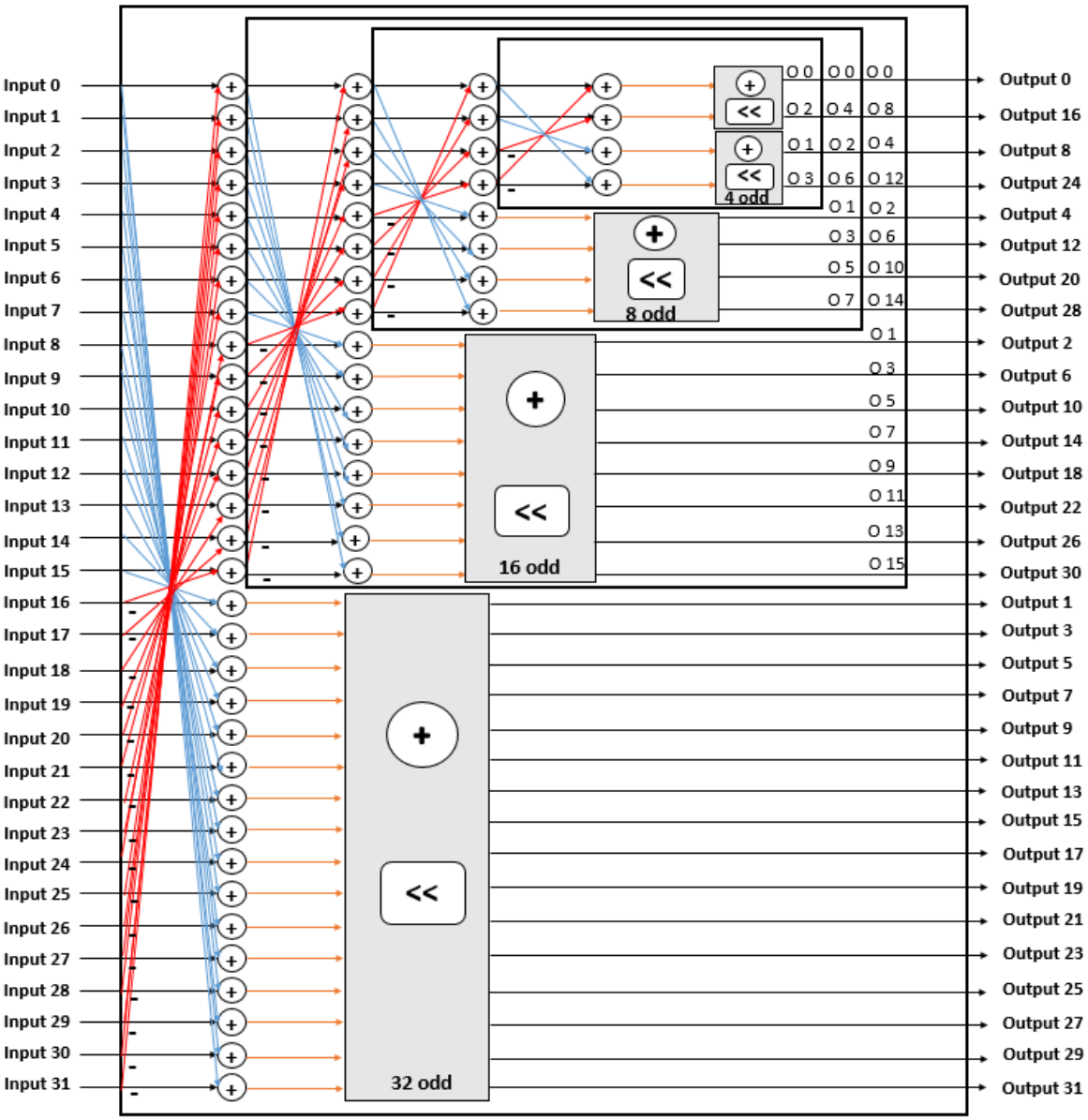
Fonte: Própria autora (2016)

5.1 Transformada Discreta do Cosseno – Software de Referência

Como fase inicial do desenvolvimento da DCT, foi implementada a arquitetura de transformadas utilizando a estrutura *Partitial Butterfly*, que é a DCT 1-D presente no codificador em *software*. Este desenvolvimento foi realizado apenas com a finalidade de entendimento do funcionamento do bloco de transformadas, envolveu a implementação dos multi-tamanhos de matrizes residuais de transformadas: 4x4, 8x8,

16x16 e 32x32. De forma geral, o vetor de resíduos processados pelo bloco de transformada 32x32 são representados na figura 17 pelos sinais de *Input* entre 0 e 31, cada um com 9 *bits,* e as saídas residuais pelos sinais de *Output* entre 0 e 31, cada um com 16 *bits*. Este aumento de *bits* entre a entrada e a saída da arquitetura ocorre devido as operações de somas e multiplicações, dessa maneira, aumentando em pelo menos 1 *bit* a cada operação.

Figura 17 - Estrutura *Partitial Butterfly*



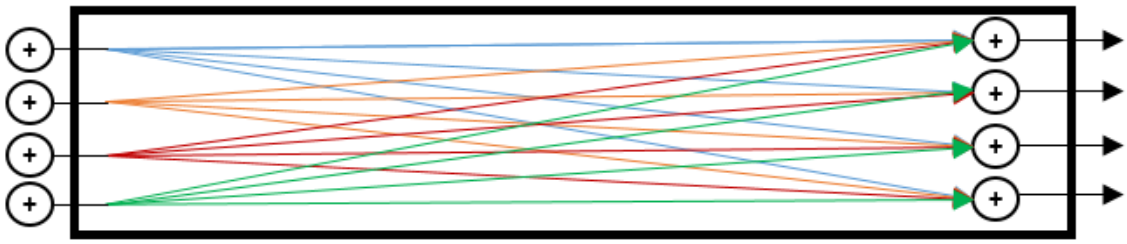
Fonte: Própria autora (2016)

A estrutura exposta na mesma figura 17, representa também as saídas das transformadas menores, indicadas pelos sinais “O” seguido de sua ordem de saída.

Fica perceptível que a parte superior das arquiteturas são representadas por uma sequência de somadores conectados à arquitetura subsequente de menor tamanho, denominada parte par. Já a parte inferior de todas as estruturas é denominada como ímpar, devido a sua independência em relação a outras transformadas de tamanho menor.

Os blocos representados por um somador e um *shift left*, como é possível observar, recebem os sinais gerados por todos os somadores da parte ímpar de cada transformada. Estes blocos são responsáveis pelos processos de deslocamentos seguidos de operações de somas dos operadores padrão do VHDL. Para exemplificar, a figura 18 apresenta as ligações deste processo em uma transformada 8x8, onde cada linha representa um sinal deslocado a esquerda representando a multiplicação por um coeficiente com sinal que foi extraído das chamadas a funções no software de referência.

Figura 18 - Deslocamentos seguidos de somas



Fonte: Própria autora (2016)

Na tabela 4 encontram-se os coeficientes de multiplicação retirados das funções da transformada 32x32 a partir do software de referência. Esses coeficientes são utilizados ao longo da execução dos quatro tamanhos de transformadas.

Tabela 4 - Coeficientes de Multiplicação da DCT 32x32

Multiplicadores
$64, 83, 36, 89, 90, 87, 80, 70, 57, 43, 25, 9, 67, 82, 85,$
$88, 78, 73, 61, 54, 46, 38, 31, 22, 13, 4, 75, 50, 18$

Fonte: Própria autora (2016)

Como estes valores de multiplicadores são aplicados aos multi-tamanhos de transformadas, foi necessário extrair o resultado de todas as saídas das

transformadas para vislumbrar de maneira correta as conexões com estes coeficientes de multiplicação.

Considerando um sinal genérico "*sig*" como exemplo, a equação (4) apresenta o processo de deslocamento utilizado nos cálculos de transformadas, suprindo a utilização de multiplicadores.

$$sig \times 90 = (sig \ll 6) + (sig \ll 4) + (sinal \ll 3) + (sig \ll 1) \qquad (4)$$

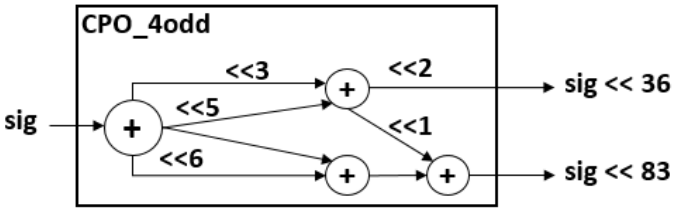
5.2 Transformada Discreta do Cosseno para economia de recursos

- **Versão 1: Do *et al*. (2014) buscando redução do caminho crítico**

Em um segundo momento, o desenvolvimento da arquitetura baseou-se na proposta de Do *et al.* (2014), que faz o reuso de componentes entre as transformadas e na transformada propriamente dita, utilizando-os como blocos operacionais separados. Para a primeira aplicação do algoritmo DCT na matriz de transformada 4x4, são realizadas duas operações básicas de soma e duas de subtração com os valores de entrada desta transformada. Para não haver perda de representatividade, a cada operação de soma é necessário estender em sinal os operandos, aumentando-os em um *bit*.

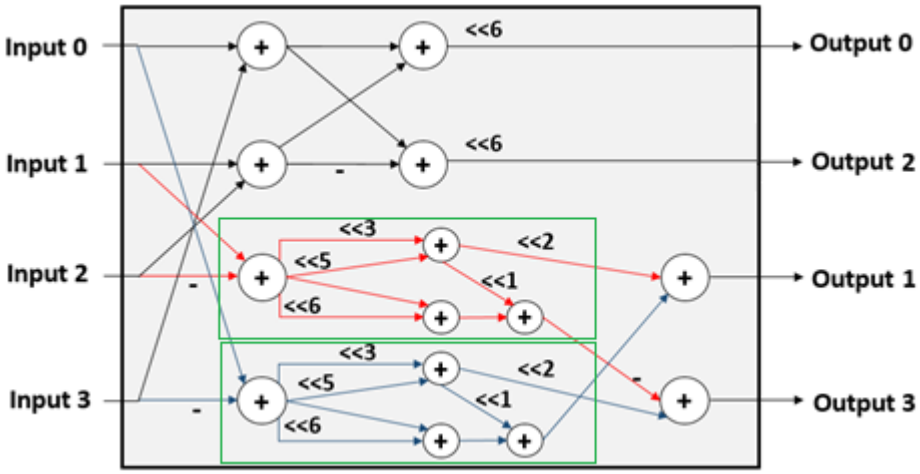
Na sequência são realizadas operações de *shift left* com as saídas pares. Para este procedimento, os sinais são estendidos já em 16 *bits*, independente do coeficiente de multiplicação que será aplicado ao sinal, resultando nos dezesseis *bits* de saída. Já para as saídas ímpares, é utilizado um componente denominado CPO4odd, responsável por calcular os deslocamentos da parte ímpar da estrutura, recebendo uma entrada de 11 *bits* devido ao procedimento de extensão em sinal aplicado a cada operando antes de serem atribuídos aos somadores/subtratores. O sinal de entrada do componente é estendido em sinal para 16 *bits* e após realiza todas as operações de deslocamento, resultando em duas saídas de dezesseis *bits*. O procedimento de extensão em sinal dentro do componente CPO é aplicado à todas as CPOs desta arquitetura. Este componente é instanciado duas vezes e, por fim, são realizadas as operações de soma/subtração, contemplando as saídas residuais ímpares desta transformada. A CPO4odd está representada na figura 19 e as conexões entre os somadores e as CPOs estão indicadas na figura 20.

Figura 19 - CPO4odd



Fonte: Própria autora (2016)

Figura 20 - Transformada 4x4

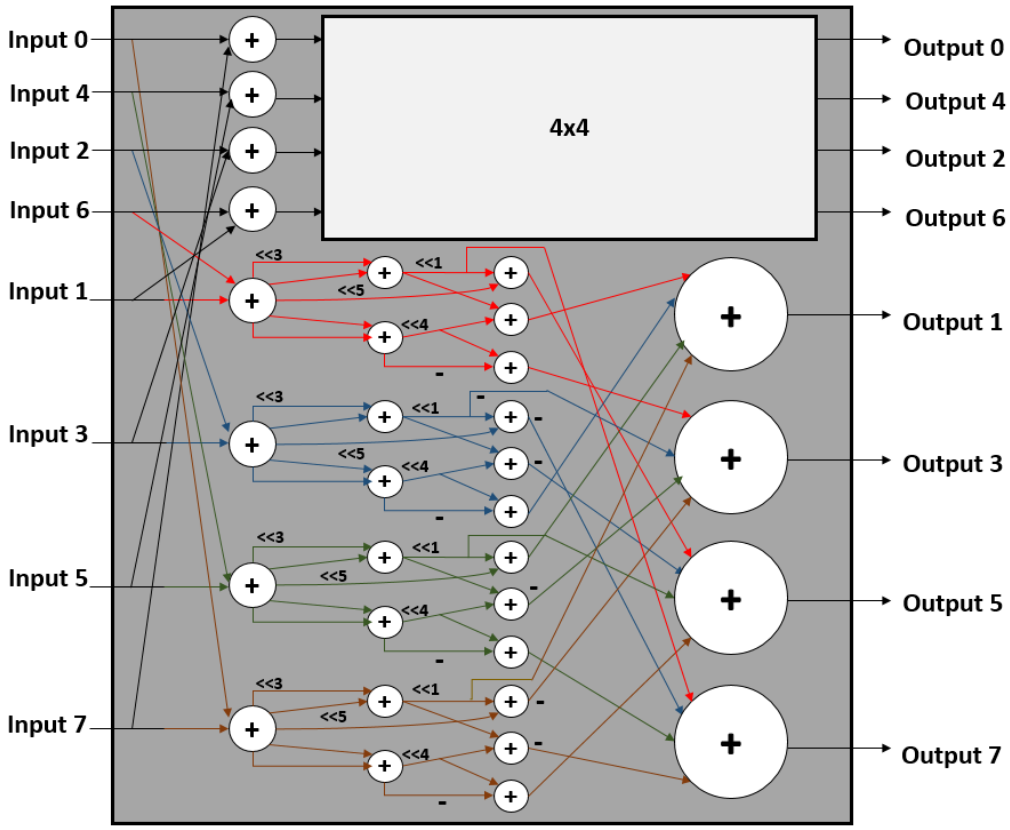


Fonte: Própria autora (2016)

Na matriz de transformada 8x8 são implementadas quatro somas e quatro subtrações dos valores de entrada. Após essas operações básicas, a parte *even* da transformada 8x8 é mapeada para as entradas do componente da transformada 4x4, que por sua vez gera as saídas pares da transformada 8x8. A parte ímpar da matriz 8x8 implica em um processo semelhante ao da 4x4, conectando através de sinais o resultado das somas em CPOs denominadas CPO8odd, como pode ser observado na figura 21, que recebem uma entrada de 11 *bits* e geram quatro saídas de dezesseis bits cada uma delas, representando os cálculos de deslocamentos à esquerda que são somados/subtraídos por um operador. O componente CPO8dd é instanciado quatro vezes dentro da transformada 8x8, cada um deles recebendo um sinal interno de posição distinta já calculado anteriormente. Sendo assim, o somador final recebe como entrada, uma saída de cada uma dessas CPOs, gerando as saídas ímpares da transformada 8x8. Este processo fica visível na figura 21, no entanto não é trivial saber onde são conectadas as saídas das CPOs. Por este motivo, foi necessário analisar as funções diretamente no software de referência. A nomenclatura utilizada para os

sinais de entrada, saídas e valores intermediários da CPO8odd presentes na imagem representam sinais residuais.

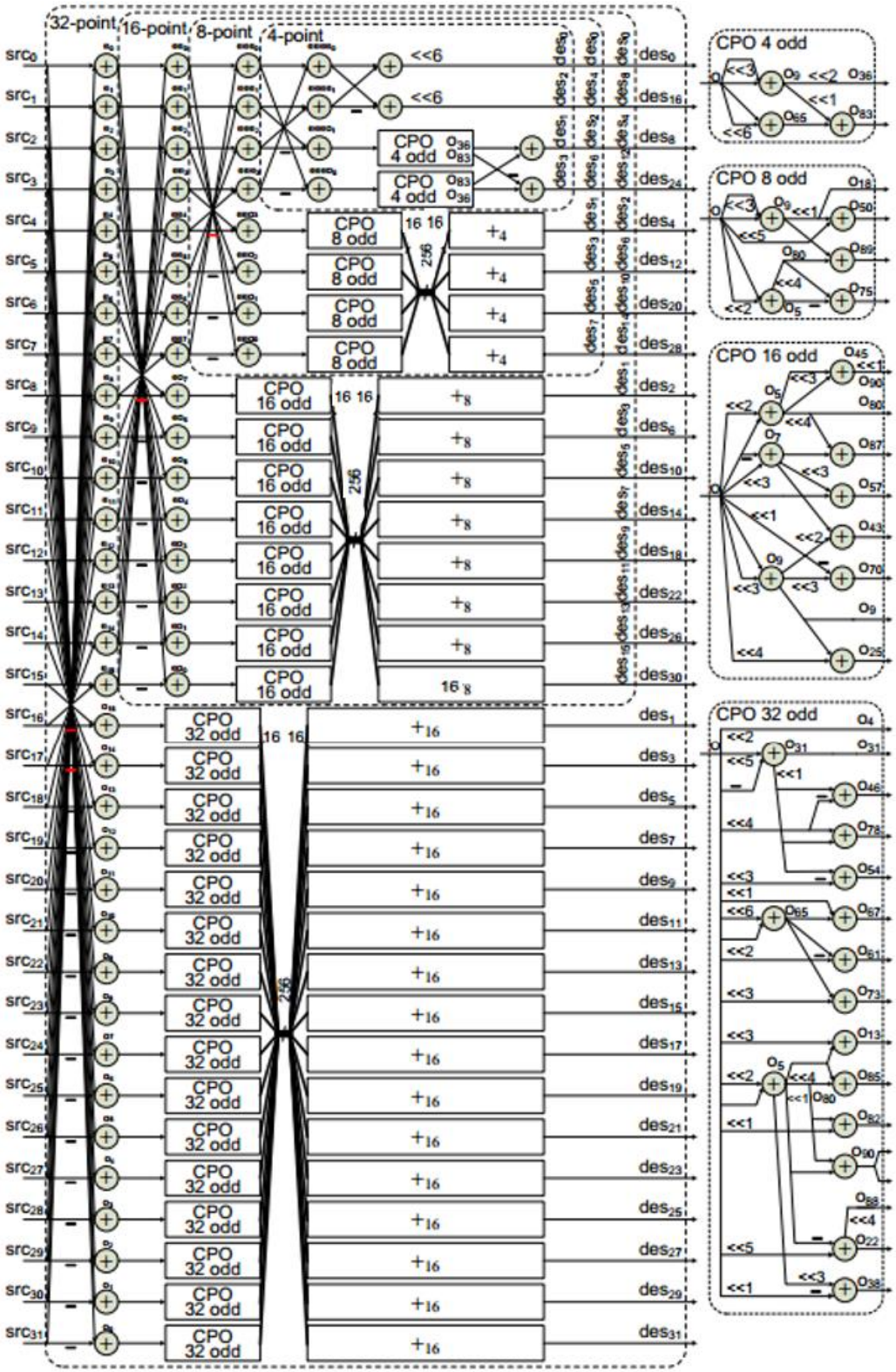
Figura 21 - Transformada 8x8



Fonte: Própria autora (2016)

A próxima transformada de tamanho 16x16, realiza operações similares às anteriores, com oito somas e oito subtrações, sendo as saídas das oito somas utilizadas como entradas na instância do bloco da matriz 8x8, enquanto cada uma das saídas das oito subtrações são entradas para cada uma das oito CPOs instanciadas neste componente. A estrutura da CPO16odd está representada na figura 22, junto ao restante da arquitetura, de maneira que similarmente como as outras transformadas já citadas, são conectadas ao somador maior, que tem por funcionalidade realizar a soma/subtração dos sinais de saída das respectivas CPOs, resultando cada uma dessas operações em uma saída ímpar referente à matriz 16x16.

Figura 22 - Arquitetura total da DCT



Fonte: Do (2014)

Para a última aplicação da transformada DCT total (32x32) proposta por Do *et al*. (2014), utiliza a *Butterfly*, com todas as respectivas dezesseis somas e dezesseis

subtrações. Sendo a parte par desse componente, os valores de entrada para o bloco da matriz de transformada 16x16, e a parte ímpar a entrada das CPOs32odd. Cada uma das dezesseis CPOs utilizadas gera dezesseis saídas de dezesseis *bits*, de maneira que cada uma dessas saídas está interligada a um sinal que realiza as somas/subtrações das respectivas saídas de cada CPO. Com esse processo, obtém-se o bloco de transformadas máximo do padrão HEVC, de 32x32, como pode ser visto na figura 22, com os resíduos de entrada sinalizados por *src* e os de saída como *des.* O sinal de entrada das CPOs, representado por “o”, é gerado pelo somador antecedente a este bloco, e as saídas deste componente representados por “o” seguido de um número, representando o sinal de entrada da CPO multiplicado por este valor.

Como citado anteriormente, as saídas vistas pela transformada 32x32 são desordenadas em relação aos outros tamanhos de transformadas. Dessa maneira, foi necessário considerar a existência de multiplexadores em todas as saídas da arquitetura, com exceção da saída zero que se mantém localizada na mesma posição independente da execução. No entanto, como a ordem visual da numeração das saídas não interfere nas funcionalidades da mesma, está lógica foi descrita junto ao *testbench* da arquitetura.

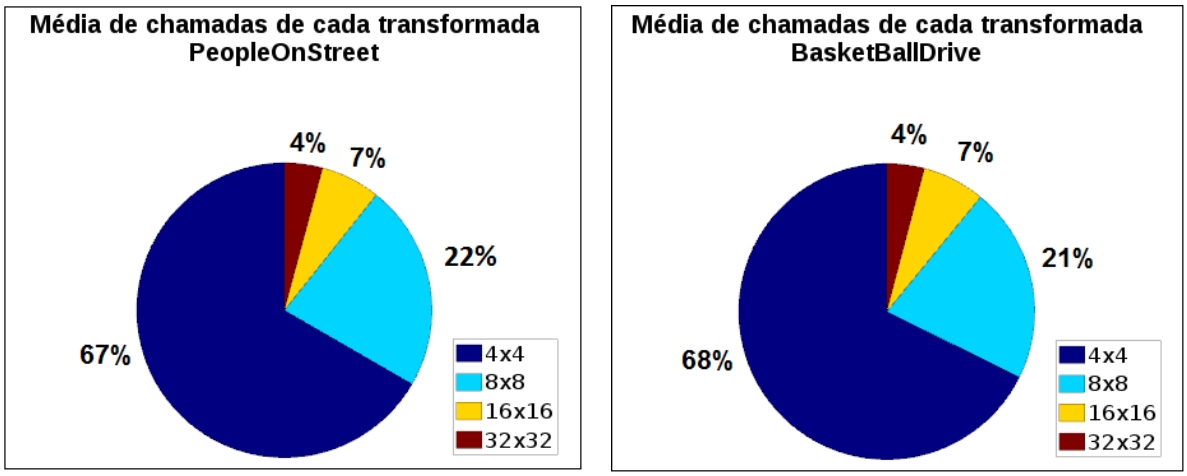
- **Análise do Software de Referência**

O procedimento de coleta de dados e análise dos mesmos, foi realizado para as sequências de vídeo teste de alta resolução, com tamanhos de 2560x1600 (WQXGA) e 1920x1080 (FullHD). As sequências de vídeo utilizadas foram a *PeopleOnStreet (*WQXGA) e *BasketballDrive* (FullHD), executadas nos perfis *LowDelay* e *RandomAccess*, com valores de QP iguais a 22 e 37. Os perfis de execução e variações do parâmetro de quantização utilizados neste trabalho, são configurações de teste recomendadas na norma pelos autores do padrão HEVC, tendo como intuito a equalização das variáveis analisadas nas arquiteturas. Devido a granularidade de chamadas do bloco de transformadas, implicando em inúmeras escritas em arquivo, os vídeos foram executados para um quadro. Esta execução apresenta-se de relevância, considerando que a função de transformadas é aplicada a cada *pixel* residual presente nos blocos de um quadro, particionados pela árvore de codificação.

No arquivo em que são realizadas as chamadas das transformadas, foi implementado em linguagem de programação C++ uma estrutura de dados que

realiza a contagem de execuções das transformadas em sequência, ou seja, foi criado um contador em formato de pilha. Esta estrutura grava em um arquivo texto, o número de vezes em sequência que cada transformada foi executada. Para fins de análise, foi utilizada a ferramenta Octave, versão 4.0.3, onde foi implementado *scripts* que analisam o arquivo gerado pelo *software* de referência, resultando no número total de chamadas de cada transformada. Como os valores mantiveram-se próximos, foi calculada uma média desse número total de execuções de cada transformada para cada sequência de vídeo, expresso em porcentagem nas figuras 23 (a) e (b).

Figura 23 - Média de execuções de cada transformada

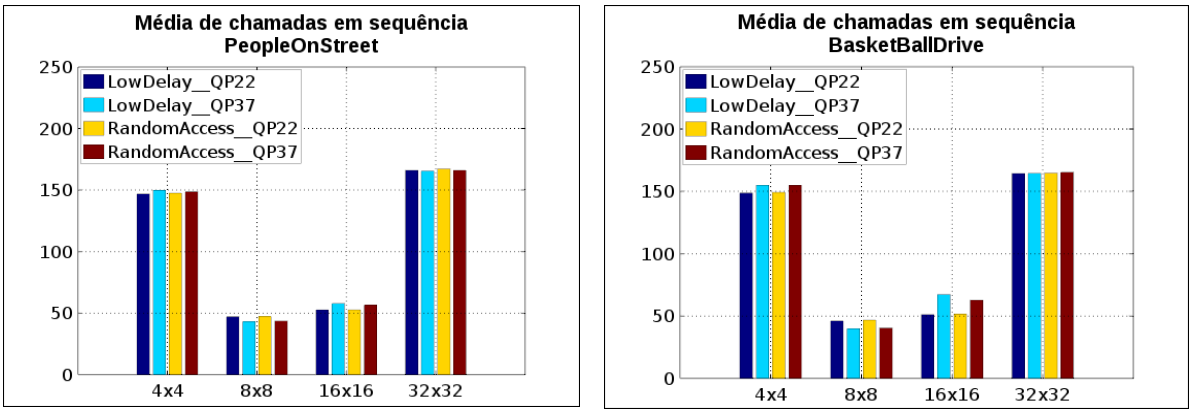


(a) PeopleOnStreet (b) BasketBallDrive

Fonte: Própria autora (2016)

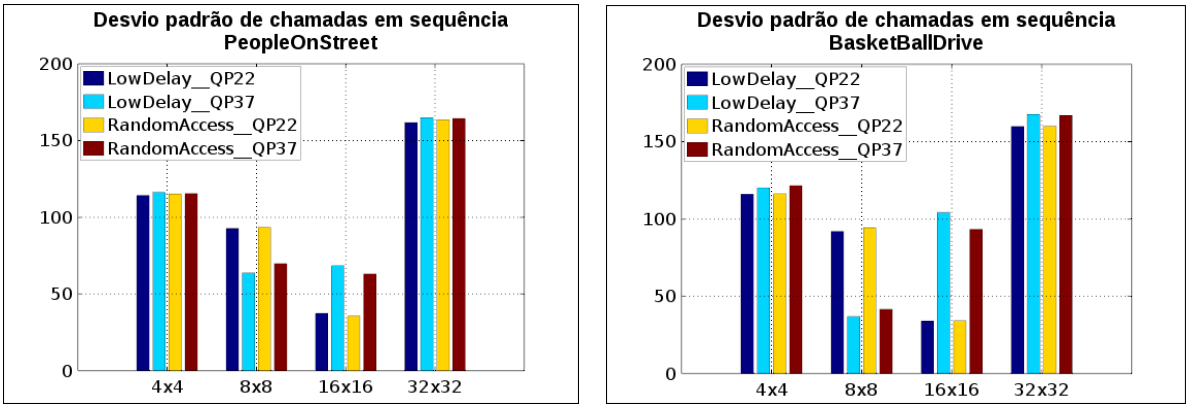
Também foi analisada a média total de chamadas em sequência, indicadas nas figuras 24 (a) e (b) e por fim o cálculo do desvio padrão nas figuras 25 (a) e (b). Todas os procedimentos foram realizados para dois valores de parâmetro de quantização (QP) e para os dois vídeos nos perfis *LowDelay* e *RandomAccess*.

Figura 24 - Média de chamadas em sequência de cada transformada



(a) PeopleOnStreet (b) BasketBallDrive

Figura 25 - Desvio padrão de chamadas em sequência de cada transformadas



(a) PeopleOnStreet (b) BasketBallDrive

Fonte: Própria autora (2016)

Os resultados obtidos por meio destas análises estatísticas, expressam a granularidade de execuções da transformada 4x4 e 8x8 em relação aos outros tamanhos de transformada. Em média, como é possível analisar na figura 23 (a) e (b), a transformada 4x4 é executada entre 67% e 68% nessas duas sequências de teste, respectivamente. A imagem 24 (a) e (b) expressa o número médio de vezes em que cada transformada é chamada em sequência, apresentando valores como no caso da 4x4, em torno de 150 vezes, a 8x8 variando perto das 48 vezes, a 16x16 entre 50 e 65 vezes e a 32x32 170 vezes. Também foi possível analisar o desvio padrão, expresso na imagem 25 (a) e (b) entre as chamadas em sequência, apresentando valores significativos principalmente para as transformadas 8x8 e 16x16.

Dessa maneira, torna-se de grande expressividade a necessidade de um bloco de inibição da atividade de chaveamento de algumas partes da arquitetura que não

necessitam estar em funcionamento enquanto as transformadas menores estão executando, principalmente no caso de quando a 4x4 é a necessária naquele momento. O objetivo de utilizar um bloco de inibição em algumas partes da arquitetura, de acordo com a transformada que é executada, é reduzir a dissipação de potência, resultando em um menor consumo de energia e também impedir que resíduos de entrada sejam enviados para somadores que não devem operar continuamente.

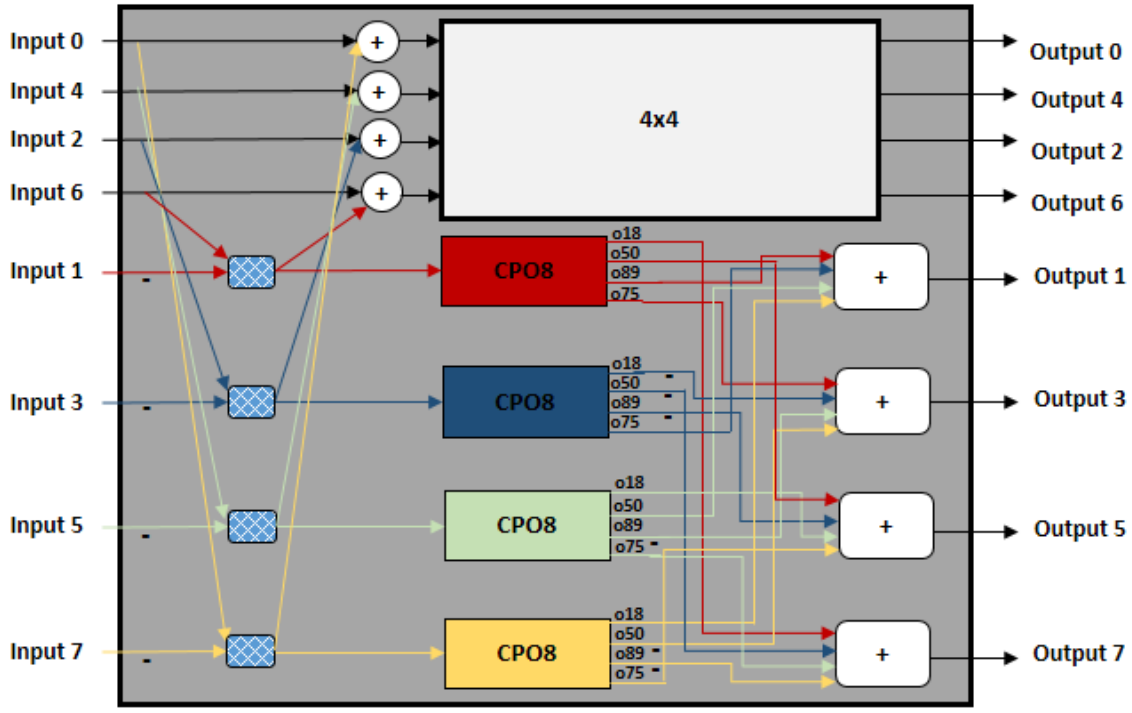
- **Versão 2: Inibição da atividade de chaveamento buscando eficiência energética**

A terceira versão implementada dos blocos de transformadas replicou a arquitetura apresentada anteriormente devido a redução do caminho crítico e reuso de componentes. A partir dos resultados obtidos nas análises estatísticas, percebeu-se o grande fluxo de chamadas da transformada 4x4, tornando as estruturas das demais transformadas desnecessárias durante essas execuções. Com isso, surge a necessidade de inibir o chaveamento de operações indesejadas, buscando reduzir a dissipação de potência e impedir que sinais com representatividade sejam processados, gerando saídas indevidas. Para obter-se o ganho energético nesta arquitetura, foi aplicada a técnica de baixo consumo denominada *operand isolation* em cada operador que não precisa estar ativo ao processar as transformadas menores. O *operand isolation* é uma técnica utilizada para conservação de energia, como o próprio nome indica, sua funcionalidade é aplicada para isolar partes do circuito com lógica combinacional (CORREALE, 1995).

Com isso, para facilitar a visualização das conexões, a figura 26 representa a transformada 8x8, com o bloco de *operand isolation* antecedendo a entrada no bloco de CPO, e também ilustra as ligações das saídas das CPOs aos somadores finais. As saídas representadas neste bloco, estão dispostas de acordo com a matriz 8x8, com finalidade de ilustração.

A conexão dos blocos de isolamento é demonstrada na figura 27, por meio de blocos distintos da arquitetura. Nesta imagem, foi destacado com a cor amarela as transformadas acopladas, e as CPOs implementadas são equivalentes às apresentadas no trabalho de Do *et al*. (2014).

Figura 26 - Exemplo de conexões ente CPOs e somadores



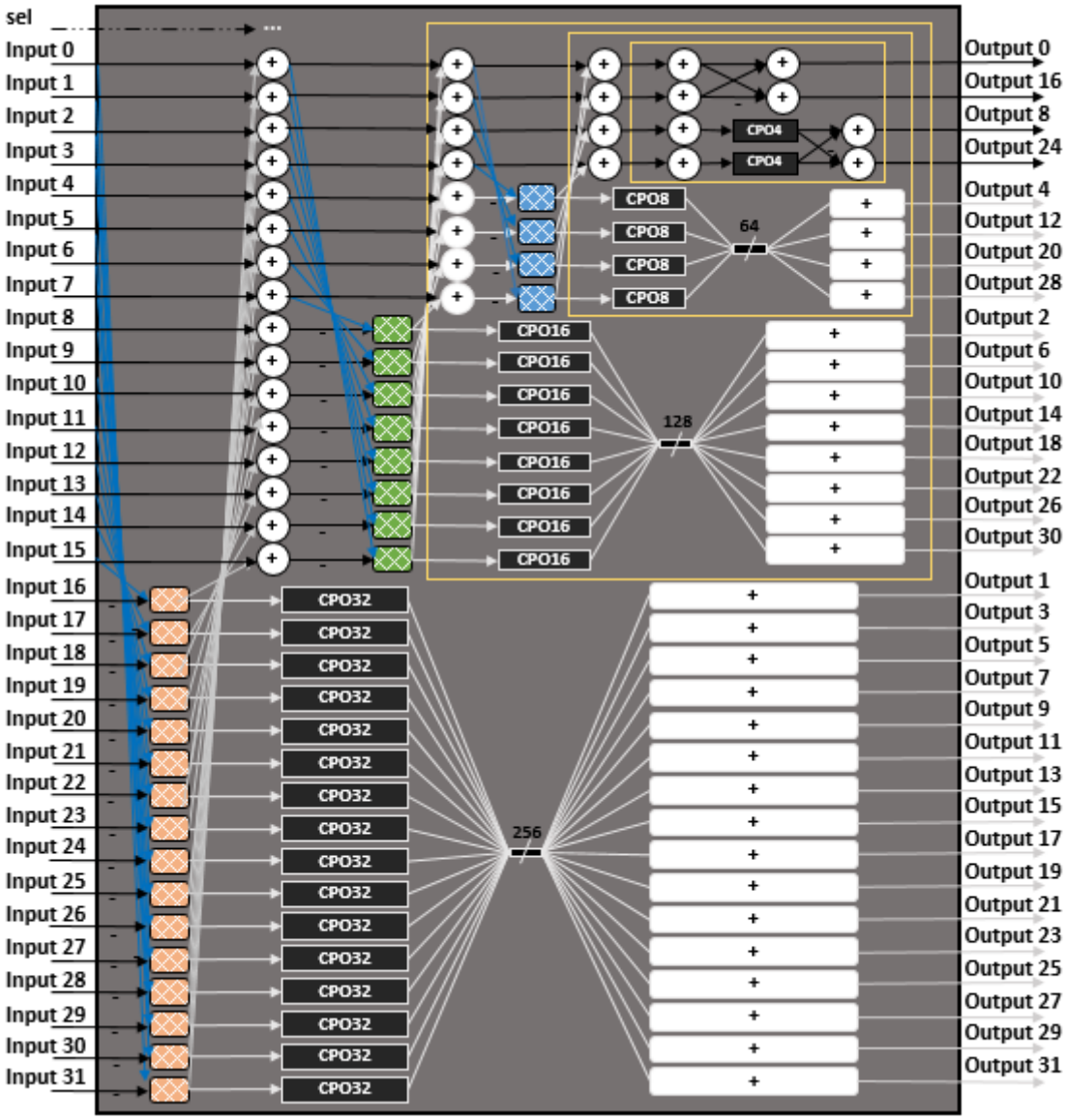
Fonte: Própria autora (2016)

Na figura 27 as estruturas indicadas pela cor cinza claro, representam todos os sinais e blocos operacionais em que foram desligados por meio dos blocos de isolamento. Como é possível observar, o primeiro sinal de entrada da arquitetura é o seletor (*sel*), possuindo 2 *bits* de entrada (*sel*(0) e *sel*(1)*).* Os dois *bits* indicam a transformada que será executada, como explicitado abaixo:

- *sel* = 00, Transformada 4x4;
- *sel* = 01, Transformada 8x8;
- *sel* = 10, Transformada 16x16;
- *sel* = 11, Transformada 32x32;

Sendo assim, o *operand isolalation* é ativo por demanda de acordo com o bloco ativador da lógica de cada transformada. Para isso, foi estruturada a tabela verdade, identificando a operação lógica aplicada de cada uma em conformidade com a tabela 5.

Figura 27 - Arquitetura total *Partitial Butterfly* com *Operand Isolation*



Fonte: Própria autora (2016)

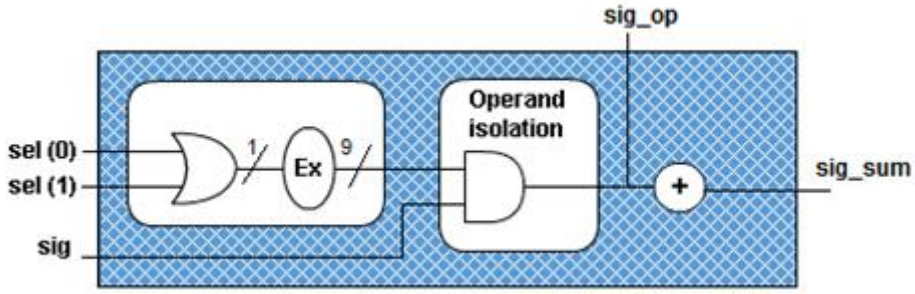
Tabela 5 - Tabela verdade da lógica de isolamento

sel(0)	sel(1)	8x8	16x16	32x32
0	0	0	0	0
0	1	1	0	0
1	0	1	1	0
1	1	1	1	1

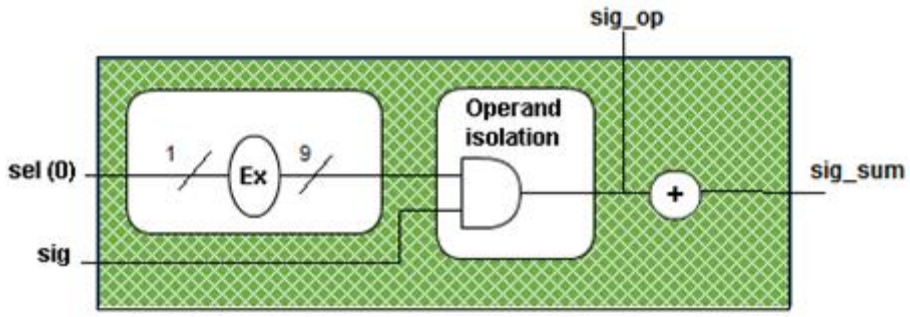
Fonte: Própria autora (2016)

Devido ao fato da transformada 4x4 ser executada em todas as chamadas, é dispensável a utilização desta técnica para ela. A figura 28 (a), (b) e (c) demonstra a lógica de isolamento identificada pela tabela verdade, conectada a uma porta AND para cada *bit* de entrada do somador. O sinal “sig_op” representa o sinal que sobe para os outros blocos da arquitetura e o “sig_sum” é o sinal que segue pela parte impar.

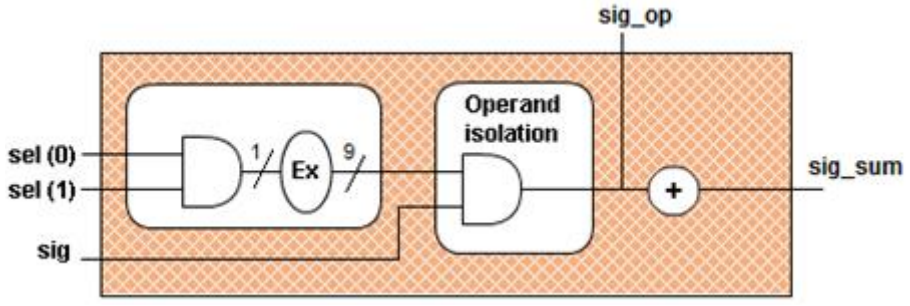
Figura 28 - Bloco de isolamento do chaveamento



(a) Bloco de isolamento da 8x8



(b) Bloco de isolamento da 16x16



(c) Bloco de isolamento da 32x32

Fonte: Própria autora (2016)

6 RESULTADOS E DISCUSSÕES DO BLOCO DE TRANSFORMADAS

Neste capítulo serão apresentados os resultados obtidos com desenvolvimento das arquiteturas. Resultados estes alcançados por meio dos dados extraídos do software de referência, com a validação e análise funcional da arquitetura de transformadas proposta, como explanado no anexo 1, e do processo de síntese.

6.1 Resultados de síntese ASIC

Para a síntese ASIC foi utilizada a ferramenta RTLCompiler da Cadence, e configurada para a biblioteca de células para a tecnologia de 65nm da ST. A partir da biblioteca de células determinada, a arquitetura foi mapeada de acordo com as células de portas lógicas disponíveis.

A síntese foi realizada para a arquitetura proposta por Do *et al.* (2014), desenvolvida neste trabalho sem e com o *operand isolation*, visto que o autor não havia apresentado resultados de síntese, apenas análises de alto nível, como o caminho crítico. Para a realização da síntese, usou-se os valores de tensão à 0,95V e temperatura a 125ºC, representando o pior caso para síntese. Dessa maneira, foi possível obter resultados em relação a frequência máxima e área das arquiteturas desenvolvidas. Para o cálculo da área total equivalente, para fins de comparação com outros trabalhos, foi realizada uma divisão da área total do circuito pela a área de 2,08 μm² da menor célula lógica NAND de duas entradas presente na biblioteca de células, referente a tecnologia para síntese ASIC. Dentre a área resultante das arquiteturas, percebe-se que houve um aumento devido a inserção da lógica de *operand isolation*, implicando, também, no aumento do caminho crítico, e consequentemente resultando em uma menor frequência.

Para encontrar o *throughput* das arquiteturas, foi calculada a quantidade de coeficientes por ciclo, como segue na equação (5).

$$\frac{\text{Coeff}}{\text{ciclo}} = \big((0{,}67 \times 4) + (0{,}22 \times 8) + (0{,}07 \times 16) + (0{,}04 \times 32)\big) = 6{,}84 \qquad (5)$$

O primeiro valor se refere a porcentagem dos resultados apresentados no gráfico da figura 23, sendo valores médios do total de execuções de cada transformada. Já o segundo valor representa a quantidade de dados que cada transformada é capaz de

processar por vez, ou seja, a quantidade de saídas que cada uma fornece por ciclo de operação. Para este cálculo foi considerado apenas resultados da sequência de vídeo *PeopleOnStreet*, devido ao fato deste ser de maior resolução e apresentar valores próximos aos do *BasketballDrive.*. Sendo assim, foi calculado o *throughput* de cada arquitetura desenvolvida, multiplicando os coeficientes por ciclo pela frequência atingida em cada arquitetura. Esse cálculo é demonstrado nas equações (6) referente a estrutura com aplicação da técnica de baixo consumo de energia e (7) sem aplicação da técnica. Os resultados gerais de síntese para ambas arquiteturas estão expostos na tabela 6.

$$Throughput = 6{,}84 \times 185{,}5 = 1268{,}82\ coeficientes\ por\ segundo \quad (6)$$

$$Throughput = 6{,}84 \times 191{,}39 = 1309{,}11\ coeficientes\ por\ segundo \quad (7)$$

Tabela 6 - Resultados de área e frequência das arquiteturas desenvolvidas

	Desenvolvida (*operand isolation*)	**Desenvolvida (Do *et al*. (2014))**
Frequência máxima (MHz)	185,5	191,39
Área (Kgates)	124,76	121,96
Coeficientes por ciclo	6,84	6,84
***Throughput* (Coeficientes por segundo)**	1268,82	1309,11

Fonte: Própria autora (2016)

As sequências de vídeo utilizadas para validar e obter resultados da arquitetura podem ser observadas na tabela 7, onde também estão indicados os valores de QP aplicados para executar as sequências, com valor de frequência definido no testbench para rodar à 166 MHz (frequência próxima da máxima frequência alcançadas por ambas versões da arquitetura). A tabela 7 também apresenta os resultados obtidos em relação a dissipação de potência. Percebe-se que os resultados finais da arquitetura de transformadas que faz uso de *operand isolation* foram significativos, chegando a reduzir a dissipação em mais de 5x, comparado a arquitetura que não aplica a técnica.

Tabela 7 - Resultados de síntese ASIC

Vídeos	Perfil	QP	Potência (mW) 166MHz	Otimização (vezes)	*Operand Isolation*
BasketballDrive	*LowDelay*	22	503,9243	#	Não
		37	506,6948	#	
	RandomAccess	22	504,0402	#	
		37	508,0538	#	
BasketballDrive	*LowDelay*	22	93,2093	5,41	Sim
		37	94,4515	5,36	
	RandomAccess	22	94,0166	5,36	
		37	93,6359	5,43	
PeopleOnStreet	*LowDelay*	22	501,9617	#	Não
		37	506,8859	#	
	RandomAccess	22	502,0028	#	
		37	506,4361	#	
PeopleOnStreet	*LowDelay*	22	92,9996	5,40	Sim
		37	93,1852	5,44	
	RandomAccess	22	93,0037	5,40	
		37	93,0647	5,44	

Fonte: Própria autora (2016)

Na tabela 8, são apresentadas as comparações em relação à área, tecnologia para síntese, frequência, dissipação de potência, coeficientes por ciclo e *throughput* entre a arquitetura proposta e as encontradas na literatura.

Tabela 8 - Comparação de resultados com trabalhos correlatos

	Desenvolvida	**Budagavi (2012)**	**Do (2014)**	**Conceição (2014)**	**Goebel (2016)**
Transformada	**DCT**	DCT+IDCT	**DCT**	IDCT	DCT
Gate Count (K)	**124,8**	156	**121,96***	238,2	97,3
Tecnologia (nm)	**ST 65**	45	**ST 65**	TSMC 90	Nangate 45
Frequência (MHz)	**185,5**	250	**191,39***	118,96	50
Estímulos	**Reais**	#	**Reais**	*Default*	*Default*
Dissipação de potência (mW)	**93**	#	**508***	339,2	24,2
Coeficientes por ciclo	**6,84**	6,84	**6,84***	#	32
***Throughput* (coeficientes por segundo)**	**1268,82**	1730	**1309,11***	541,90	1600

* Arquitetura proposta por Do *et al.* (2014) desenvolvida neste trabalho.

Fonte: Própria autora (2016)

Comparando os resultados apresentados na tabela 8, percebe-se que existe um *trade-off* em relação as variáveis analisadas. Ao avaliar os valores obtidos entre a arquitetura desenvolvida com e sem o bloco de isolamento, nota-se que em relação a área não apresentou diferença tão significativa, resultando em um aumento de 2,33%. Este aumento já era esperado em consequência da inserção de um bloco de isolamento antecedendo cada bloco de CPO presente na arquitetura. Já em comparação com o resultado apresentado por Budagavi *et al.* (2012) a arquitetura desenvolvida com o *operand isolation* apresentou redução em 25% em seu valor de área; em relação ao trabalho de Conceição *et al*. (2014) resultou na redução de 90,86% e comparado à Goebel *et al*. (2016), apresentou aumento de 28,26% em área.

Apenas Goebel *et al*. (2016) apresentou valor de coeficientes por ciclo, no entanto, considerando os valores estatísticos de execuções das transformadas e a frequência, foi possível estimar o *throughput* da arquitetura proposta por Budagavi *et al*. (2012). Goebel *et al*. (2016) apresenta o valor fixo de 32 coeficientes por ciclo, enquanto as arquiteturas desenvolvidas apresentam 6,84 coeficientes por ciclo. Sendo assim, o hardware de Goebel *et al*. (2016) citado tem um *throughput* de 1600 coeficientes por segundo, ou seja, 26,1% a mais do que a arquitetura com o *operand isolation*. O resultado estimado para Budagavi *et al*. (2012) apresenta *throughput* 34,77% maior. No entanto, o valor alcançado nas arquiteturas proposta neste trabalho resulta em um bom desempenho da arquitetura, se aproximando dos valores dos trabalhos correlatos.

No que se refere a dissipação de potência, foco principal neste trabalho, a arquitetura desenvolvida com a aplicação da técnica de baixo consumo, apresentou um resultado de baixo consumo, com 93mW. Comparando este resultado com a mesma arquitetura sem a aplicação da técnica, apresentou redução da dissipação de potência em mais de 5x. Já em comparação com resultados expostos no trabalho de Conceição *et al*. (2014), houve redução de 3,6x (mesmo sendo focado nas transformadas inversas, os operadores tendem a ter granularidade parecida com as transformadas diretas). O resultado de consumo de Goebel *et al*. (2016) chegou a 24,2mW, valor consideravelmente menor do que a arquitetura proposta e outras encontradas na literatura. No entanto, cabe ressaltar que a análise da dissipação de potência realizada por Conceição *et al*. (2014) e Goebel *et al.* (2016) utilizaram a atividade de chaveamento *default* inferidos pela ferramenta de análise, não resultando em valores considerados realísticos, ao contrário do presente trabalho, onde foram utilizadas sequências de vídeos reais para a análise.

7 CONCLUSÃO

Este trabalho apresentou um estudo seguido de conceitos referentes a vídeo digital, compressão de vídeo, ao padrão H.265/HEVC e bloco de transformadas, visto que este, destaca-se por ser um dos mais novos padrões de codificação de vídeo. Foram descritos os principais módulos do *codec* HEVC, e seus respectivos funcionamentos. Neste mesmo embasamento teórico, percebeu-se a elevada complexidade computacional que este padrão apresenta, principalmente quando comparado aos padrões anteriores.

Após os estudos e análises do padrão e do bloco de transformadas, a arquitetura foi descrita em linguagem de descrição de *hardware* VHDL e sintetizada para ASIC, com tecnologia de ST de 65 nm utilizando a ferramenta RTLCompiler da Cadence. Como resultados de síntese, foi obtido uma frequência de 191,5 MHz na arquitetura descrita sem aplicação de técnicas de *low-power* e 185,5 MHz na arquitetura com *operand isolation*, diferença praticamente desprezível no que tange o *throughput* das arquiteturas, com resultados de 1268,82 e 1309,11 coeficientes por segundo. Ambas as arquiteturas desenvolvidas foram validadas utilizando resíduos de sequências de vídeo de teste reais gerados pelo modelo do *codec*.

Dessa maneira, presente trabalho explorou técnicas de implementação visando implementação *low-power,* utilizando um modelo da arquitetura. Conclui-se então, que foi possível atingir ganho considerável no que se trata de consumo energético para processar matrizes residuais de uma sequência de vídeo qualquer, considerando os resultados obtidos para os dois valores de QP (22 e 37). Além disso, a arquitetura proposta também explorou os múltiplos tamanhos de matrizes residuais, proporcionando com o isolamento de algumas partes da arquitetura.

Com os resultados apresentados no capítulo anterior, conclui-se que este trabalho se destacou, apresentando maior potencial para aplicações *low-power*, visto que atingiu redução de mais de 5x de dissipação de potência, resultando na dissipação de 93mW, implicando na redução do consumo de energia.

REFERÊNCIAS

AFONSO, Vladimir. **Desenvolvimento de Arquiteturas para Estimação de Movimento Fracionária Segundo o Padrão HEVC.** Dissertação de Mestrado. Universidade Federal de Pelotas, 2012.

AGOSTINI, Luciano Volcan. **Desenvolvimento de Arquiteturas de Alto Desempenho Dedicadas à Compressão de Vídeo Segundo o Padrão H. 264/AVC**. Tese de Doutorado. Universidade Federal do Rio Grande do Sul, 2007.

BELTRÃO, Gabriel; Rangel Arthur; Iano Yuzu. Introdução ao HEVC – High Efficiency Video Coding. In: SIMPÓSIO DE PROCESSAMENTO DE SINAIS, 2, 2011, Campinas. **Anais eletrônicos**... Recife: UNICAMP, 2011. Disponível em: <http://www.sps.fee.unicamp.br/sps2011/proceedings_sps2011/Gabriel_Coding_SPS2011.pdf>. Acesso em: 5 maio, 2016.

BOSSEN, F. *et al*. HM Software Manual. **Document: JCTVC-Software Manual**. 2016.

BUDAGAVI, M. *et al*. Unified forward+ inverse transform architecture for HEVC. In: IEEE International Conference on Image Processing (ICIP**)**. 2012. **Anais...** IEEE, 2012. p. 209-212.

CARTAJENA, Juan Jesus Cruz. **Codificação de vídeo utilizando modelos de texturas**. Dissertação de Mestrado. Universidade de Brasília, 2014.

CONCEIÇÃO, Ruhan; Souza, Claudio; Jeske Ricardo; Porto Marcelo; Zatt Bruno; Agostini Luciano. Power efficient and high troughtput multi-size IDCT targeting UHD HEVC decoders. In: IEEE. International Symposium on Circuits and Systems (ISCAS). **Anais…** 2014. p. 1925-1928.

CONCEICAO, R. *et al*. Software Solution for Hardware Optimization of 1-D DCT of the HEVC. In: Student Forum**. Anais…** 2012.

CORREALE, A. Jr. Overview of the power minimization techniques employed in the IBM PowerPC 4xx embedded controllers. In: Proceedings of the 1995 international symposium on Low power design. **Anais...** ACM, 1995. p. 75-80.

DINIZ, C. M. **Arquitetura de hardware dedicada para a predição intra-quadro em codificadores do padrão H. 264/AVC de compressão de vídeo**. Dissertação de Mestrado – Universidade Federal do Rio Grande do Sul. 2009.

DO, T. T. T. *et al*. High-throughput and low-cost hardware-oriented integer transforms for HEVC. In: IEEE International Conference on Image Processing (ICIP). **Anais…** IEEE, 2014**.** p. 2105-2109.

GOEBEL, J. *et al*. An HEVC multi-size DCT hardware with constant throughput and supporting heterogeneous CUs. In: IEEE International Symposium on Circuits and Systems (ISCAS). **Anais...** 2016. p. 2202-2205.

HUNG, E. M. **Compensação de movimento utilizando módulos multi-escala e forma variável em um CODEC de vídeo híbrido**. Dissertação de Mestrado – Universidade de Brasília. 2007.

ITU-T Study Group *et al*. **Series H: Audiovisual And Multimedia Systems Infrastructure of audiovisual services – Coding of moving**, 2015.

KALALI, E. *et al*. A low energy HEVC inverse transform hardware. **IEEE Transactions on Consumer Electronics**. 2014. v.60, p. 754-761.

KIM, I. *et al*. Block partitioning structure in the HEVC standard. **IEEE Transactions on Circuits and Systems for Video Technology**. 2012. v.22, p. 1697-1706.

KIM, I. *et al*. High efficiency video coding (hevc) test model 10 (hm10) encoder description. In: **Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 12th Meeting: Geneva**. 2013. p. 126-138.

KOU, W. **Digital image compression: algorithms and standards**. In**:** Springer Science & Business Media. 1995.

MANOEL, E. T. M. *et al*. **Codificação de vídeo H. 264: estudo de codificação mista de macroblocos**. Dissertação de Mestrado – Universidade Federal de Santa Catarina. 2007.

MIDHA, S. *et al*. Smriti. Analysis of RGB and YCbCr color spaces using wavelet transform. In: IEEE International Advanced Computing Conference (IACC). **Anais…** 2014**.** p. 1004-1007.

MINALLAH, N *et al.*. Performance Analysis of H. 265/HEVC (High-Efficiency Video Coding) with Reference to Other Codecs. In: 2015 13th International Conference on Frontiers of Information Technology (FIT). **Anais...** IEEE, 2015. p. 216-221.

OLIVEIRA, J. F. F *et al*. **Padrão HEVC–Novas Tecnologias para Aplicações de Elevadas Taxas de Compressão de Vídeo**. Instituto de Estudos Avançados em Comunicação (IACOM). 2014.

PARKER, M. *et al*. **Digital video processing for engineers: a foundation for embedded systems design**. Newnes, 2012.

POYNTON, C. **Digital video and HD: Algorithms and Interfaces**. Elsevier. 2012.

RAMOS, F. L. L. **Arquitetura para o algoritmo CAVLC de codificação de entropia segundo o padrão H. 264/AVC**. Dissertação de Mestrado – Universidade Federal do Rio Grande do Sul 2010.

RICHARDSON, I. E. **H. 264 and MPEG-4 video compression: video coding for next-generation multimedia**. John Wiley & Sons, 2004.

RITHE, R. *et al*. Quad full-hd transform engine for dual-standard low-power video coding. In: IEEE International Solid-State Circuits Conference (ISSCC), 2012. **Anais...** IEEE, 2012, p. 2724-2736.

ROSA, V. S. **Arquiteturas de hardware dedicadas para codificadores de vídeo H. 264: filtragem de efeitos de bloco e codificação aritmética binária adaptativa a contexto.** Tese de Doutorado – Universidade Federal do Rio Grande do Sul, 2010.

SEIDEL, I. *et al*. Energy-efficient SATD for beyond HEVC. In: IEEE International Symposium Circuits and Systems (ISCAS). **Anais...** 2016, p. 802-805.

SILVA, A. M. C. **Um Estudo Sobre o Padrão H. 264/AVC de Compressão de Vídeo**. Dissertação de Doutorado - Universidade Católica De Pelotas. 2007.

SILVA, M. G. **Computational Effort Analysis and Control in High Efficiency Video Coding**. Tese de Doutorado Universidade Federal do Rio Grande do Sul, 2014

SULLIVAN, G. J*. et al*. Overview of the high efficiency video coding (HEVC) standard. **IEEE Transactions on Circuits and Systems for Video Technology**, 2012, v.22, p. 1649-1668.

SULLIVAN, G. J. *et al*. Standardized extensions of high efficiency video coding (HEVC). **IEEE Journal of Selected Topics in Signal Processing**, 2013,v.7, p. 1001-1016.

SUMMERS, Jim et al. Characterizing the workload of a netflix streaming video server. In: IEEE International Symposium on Workload Characterization (IISWC), 2016. **Anais...** IEEE, 2016. p. 1-12.

SZE, V. *et al*.. **High Efficiency Video Coding (HEVC)**. Springer. 2014.

TEIXEIRA, G. D. **Desenvolvimento de uma arquitetura de hardware de um estimador de vetores de movimento de precisão sub-pixel seguindo o padrão HEV**. Dissertação de Doutorado – Universidade Federal do Rio Grande do Sul, 2014

TIKEKAR, M. *et al*. Energy and area-efficient hardware implementation of HEVC inverse transform and dequantization. In: International Conference on Image Processing (ICIP), 2014. **Anais…** IEEE, 2014. p. 2100-2104.

WANG, H. *et al*. Predicting zero coefficients for High Efficiency Video Coding. In: IEEE International Conference on Multimedia and Expo (ICME), 2014**. Anais...** IEEE, 2014 p. 1-6.

APÊNDICE A - Metodologia para Validação do Módulo de Transformadas

Para tornar efetiva a validação funcional dos blocos de transformadas desenvolvidos, foi necessário gerar estímulos de entrada para que ocorresse o processamento dos dados. Dessa maneira, foi necessário obter uma fonte de dados válidos.

Em decorrência disso, foi utilizado o *software* de referência (JCT-VC) do *codec* H.265/HEVC, comumente denominado como Modelo Referência (HM). Após ser realizado um estudo deste *software* desenvolvido em linguagem de programação C++, foi identificado o arquivo que implementa o bloco de interesse neste trabalho. Sendo assim, foram inseridas estruturas no mesmo código programado, com a finalidade de obter impressões no formato binário destes dados em um arquivo texto, como indicado pela figura 1.

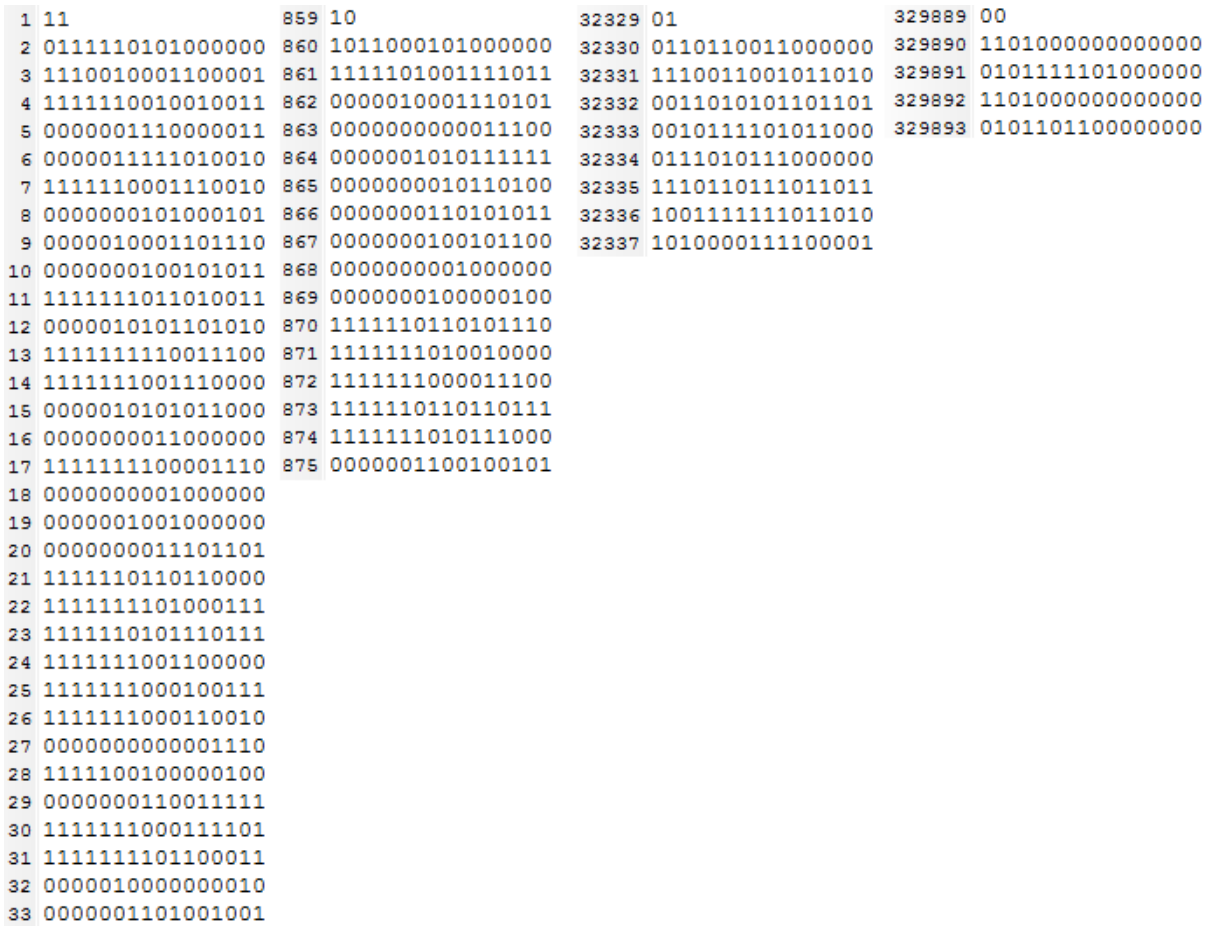
Figura 1 – Exemplo de dados de entrada extraídos do HM

```
 1 11          859 10          32329 01          329889 00
 2 110111011   860 110111011   32330 110111011   329890 010000100
 3 110111010   861 110111010   32331 110111010   329891 110000101
 4 110111010   862 110111010   32332 110111010   329892 110000101
 5 110111010   863 110111010   32333 110111010   329893 010000100
 6 110111100   864 110111100   32334 110111100
 7 110111101   865 110111101   32335 110111101
 8 110111111   866 110111111   32336 110111111
 9 111000000   867 111000000   32337 111000000
10 110111111   868 110111111
11 110111100   869 110111100
12 110111011   870 110111011
13 110111101   871 110111101
14 110111101   872 110111101
15 110111111   873 110111111
16 110111111   874 110111111
17 110111110   875 110111110
18 110111101
19 110111110
20 110111101
21 110111101
22 110111101
23 110111111
24 111000001
25 111000011
26 111000010
27 111000000
28 111000000
29 111000000
30 110111111
31 111000000
32 110111100
33 110111010
```

Fonte: Própria autora (2016)

Dessa maneira, foi possível obter valores residuais de entrada e uma *flag* (utilizada na arquitetura como um seletor) indicativa de qual transformada são estes resíduos. Com isso, foi utilizada a ferramenta ModelSim HDL da Altera, versão 10.1 para realizar as simulações com o *testbench* descrito em VHDL, capaz de ler o arquivo texto com as entradas de cada sequência de vídeo obtidos pelo *software*. Além das simulações realizadas nas arquiteturas, também foi inserida ao *testbench* a funcionalidade de gravar as saídas em um arquivo texto da mesma maneira que os resultados do *software*, como segue na figura 2.

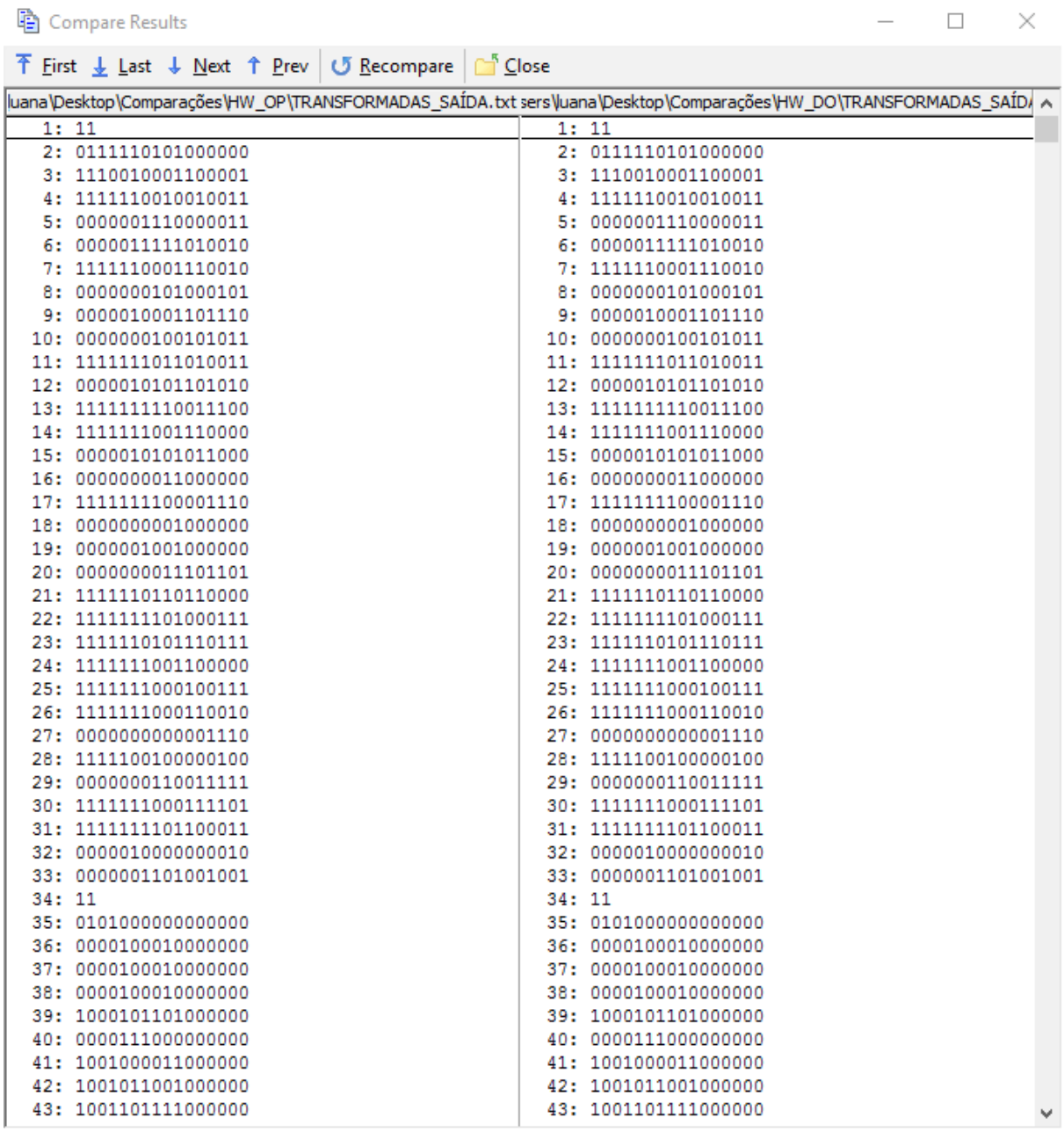
Figura 2 – Exemplo de saídas geras



Fonte: Própria autora (2016)

Ao final dessas etapas, o ambiente de validação estava estruturado, faltando apenas a comparação entre as duas saídas de dados (HDL e HM) para concretizar a validação funcional do bloco. No processo de comparação entre os dois arquivos, foi utilizada a ferramenta ConTEXT, como explicitado na figura 3.

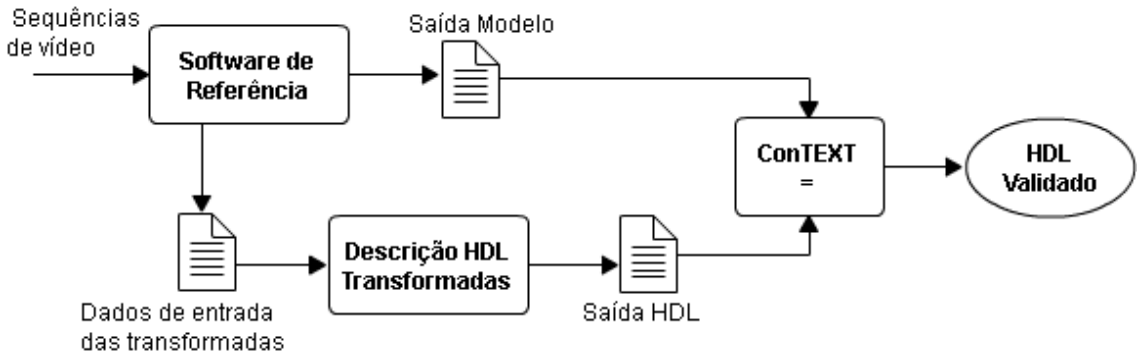
Figura 3 – Comparação entre as arquiteturas, indicando igualdade



Fonte: Própria autora (2016)

Após a execução do procedimento explanado, o ambiente de validação estava definido, resultando na verificação funcional da arquitetura totalmente de acordo com o esperado em relação ao modelo em *software*. Com isso, a metodologia aplicada ao ambiente de simulação é demonstrada na figura 4.

Figura 4 – Modelo de validação das arquiteturas



Fonte: Própria autora (2016)