UNIVERSIDADE FEDERAL DE PELOTAS
Programa de Pós-Graduação em Computação

Dissertação

Desenvolvimento de Hardware para a Transformada Rotacional 8x8 com Foco na Codificação de Vídeos Digitais de Altíssima Resolução

Henrique Avila Vianna

Pelotas, 2012

HENRIQUE AVILA VIANNA

Desenvolvimento de Hardware para a Transformada Rotacional 8x8 com Foco na Codificação de Vídeos Digitais de Altíssima Resolução

Dissertação apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Mestre em Ciência da Computação

Orientador: Prof. Dr. Luciano Volcan Agostini

Pelotas, 2012

Dados de catalogação na fonte:
Ubirajara Buddin Cruz – CRB-10/901
Biblioteca de Ciência & Tecnologia - UFPel

V617d Vianna, Henrique Avila
Desenvolvimento de hardware para a transformada rotacional 8x8 com foco na codificação de vídeos digitais de altíssima resolução / Henrique Avila Vianna. – Pelotas, 2012. – 79f. : il. color. – Dissertação (Mestrado). Programa de Pós-Graduação em Ciência da Computação. Área de concentração em Sistemas Digitais e Embarcados. Universidade Federal de Pelotas. Centro de Desenvolvimento Tecnológico. Pelotas, 2012. - Orientador Luciano Volcan Agostini.

1.Informática. 2.Codificação de vídeo. 3. High Efficiency Video Coding (HEVC). 4.Transformada rotacional. 5.Sistemas digitais. 6.Projeto em FPGA. 7.Vídeos de alta definição. I.Agostini, Luciano Volcan. II.Título.

CDD: 006.7

Banca examinadora:

.....
Prof. Dr. Luciano Volcan Agostini (Orientador)

.....
Prof. Dr. Sergio Bampi

.....
Prof. Dr. Júlio Carlos Balzano de Mattos

.....
Prof. Dr. Felipe de Souza Marques

Aos meus pais.

AGRADECIMENTOS

Quero agradecer, em primeiro lugar, aos meus pais, Vilma e Antonio, pela educação que me deram e pelos valores que me ensinaram desde criança. Pelas oportunidades que me proporcionaram, de estudar, de conhecer novos lugares e culturas, e por me incentivarem a continuar sempre me aprimorando.

À minha mãe, em especial, por insistir incansavelmente durante quase 18 anos para que eu fizesse o mestrado. Taí mãe, demorou, mas saiu! :)

Ao meu pai, pela aquisição daquele Apple II, que, além de ter sido uma fascinante ferramenta de aprendizado durante minha infância, me fez descobrir uma profissão. Por ter sido meu primeiro professor de computação, e por compartilhar comigo seu entusiasmo por áudio e vídeo digital. Todo esse *background* teve um papel importante na minha escolha por essa linha de pesquisa e no meu sucesso no mestrado.

À minha namorada Virginia, agradeço pelo carinho e apoio incondicionais que foram fundamentais para que eu superasse diversos obstáculos e desafios nesses últimos dois anos, não só no mestrado mas também na minha vida pessoal e profissional. Pela parceria em todos os momentos, por suportar minhas chatices e manias e, ainda assim, estar sempre ao meu lado. Obrigado por tudo, Boti! > 3<

À minha irmã Carolina e ao tio Jonca, a quem considero um irmão mais velho, agradeço pela amizade e companhia desde a infância. E por saber que posso contar sempre com vocês!

Ao professor Luciano Agostini, meu orientador, quero registrar meu profundo agradecimento pela paciência, serenidade e objetividade com as quais sempre tratou minhas inúmeras dúvidas e angústias durante o desenvolvimento deste trabalho. Agradeço pelas orientações e ensinamentos, pelo constante incentivo e, acima de tudo, por acreditar na minha capacidade, mesmo quando eu próprio a questionei. Desde o primeiro contato com o prof. Agostini, ficaram evidentes para mim sua paixão pela pesquisa e pela docência, e seu entusiasmo pelo trabalho que desenvolve. Ao fazer parte do seu dia-a-dia de trabalho, percebi que esse entusiasmo é contagiante e incita em todos que trabalham com ele o desejo de dar o melhor de si, de encarar novos desafios e superar limitações. Luciano, foi uma honra

e um privilégio ter sido teu orientando. Espero que eu tenha conseguido, com este trabalho, retribuir um pouco da tua dedicação.

Aos professores do PPGC, em especial àqueles dos quais tive o prazer de ser aluno, — professores Júlio Mattos, Leomar da Rosa Jr., Ricardo Araújo, Marilton de Aguiar, Paulo Ferreira Jr., Felipe Marques, Denis Franco e Simone Cavalheiro — agradeço pela dedicação e comprometimento com o curso, que transpareceram na excelência das aulas e demais atividades didáticas realizadas. Graças a essa equipe de professores, o mestrado foi uma experiência ainda mais enriquecedora e gratificante para mim.

Ao Guilherme Corrêa, doutorando em Portugal, por desbravar os caminhos do HEVC e apresentar o novo padrão aos colegas do GACI. E pela sugestão da ROT como um dos possíveis objetos de minha investigação no mestrado.

Ao Gustavo Sanchez, bolsista IC do GACI, pela ajuda na implementação das primeiras arquiteturas.

Ao Ricardo Jeske, colega de mestrado, pelas proveitosas trocas de ideias desde os primeiros trabalhos de disciplinas.

À Martha Maia, secretária do PPGC, pela gentileza e diligência habituais no auxílio aos alunos com relação às demandas administrativas do curso.

Ao prof. Adenauer Yamin, pelas valiosas orientações durante minha preparação para a seleção do mestrado.

E, finalmente, muito obrigado a todos os colegas e amigos que me apoiaram durante essa jornada e que contribuíram, mesmo que indiretamente, para o sucesso deste trabalho.

“Do. Or do not. There is no try.”

Yoda

RESUMO

VIANNA, Henrique A. **Desenvolvimento de Hardware para a Transformada Rotacional 8x8 com Foco na Codificação de Vídeos Digitais de Altíssima Resolução**. 2012. 79f. Dissertação (Mestrado em Ciência da Computação). Universidade Federal de Pelotas, Pelotas.

A Transformada Rotacional (ROT) é uma das novas ferramentas propostas para o padrão emergente de codificação de vídeo HEVC. O objetivo desta ferramenta de codificação é obter maior compactação da energia presente na matriz de coeficientes da transformada principal, melhorando a eficiência da codificação de entropia e minimizando o erro de quantização. Arquiteturas de hardware dedicadas à codificação e decodificação de vídeo são essenciais para garantir o desempenho necessário com baixo consumo de energia e potência, fatores especialmente críticos em dispositivos móveis e portáteis. Este trabalho apresenta uma investigação da ROT com foco no desenvolvimento de soluções em hardware para esta transformada. O trabalho detalha a exploração algorítmica realizada para simplificar as equações, visando a implementação em hardware. São apresentadas três versões arquiteturais para as transformadas ROT direta e inversa, gerando diferentes alternativas de desempenho em termos de taxa de processamento e consumo de hardware. As arquiteturas foram descritas em VHDL e sintetizadas para um FPGA da família Stratix III. Os resultados mostram que todas as versões da arquitetura são capazes de processar vídeos até a resolução 4K UHD (3840x2160 pixels) a 30 quadros por segundo. A versão com a maior taxa de processamento obteve uma frequência máxima de operação de 215,01 MHz. Essa versão da arquitetura atinge uma taxa de processamento de 1,72 bilhão de amostras por segundo, permitindo o processamento de vídeos até a resolução 8K UHD (7680x4320 pixels) a uma taxa de 30 quadros por segundo.

Palavras-chave: Codificação de Vídeo, HEVC, Transformada Rotacional, Sistemas Digitais, Projeto em FPGA.

ABSTRACT

VIANNA, Henrique A. **Desenvolvimento de Hardware para a Transformada Rotacional 8x8 com Foco na Codificação de Vídeos Digitais de Altíssima Resolução**. 2012. 79f. Dissertação (Mestrado em Ciência da Computação). Universidade Federal de Pelotas, Pelotas.

The Rotational Transform (ROT) is one of the novel tools proposed for the HEVC emergent video coding standard. The main goal of this coding tool is to achieve higher energy compaction of the main transform coefficient matrix and thus improve entropy coding and minimize quantization error. Dedicated hardware architectures for video coding and decoding are essential to guarantee the necessary performance with low power and energy consumption, which are especially critical resources on portable and mobile devices. This work presents an investigation of the ROT focusing on the development of hardware solutions for the transform. The work explains in details the algorithmic exploration targeting hardware implementation. Three architectural versions are presented for the forward and inverse ROT transforms, generating different performance alternatives considering processing rate and hardware consumption. The architectures were described in VHDL and synthesized for a Stratix III FPGA. Results show that all versions of the architecture are capable of processing videos up to the resolution 4K UHD (3840x2160 pixels) at 30 frames per second. The version with the highest processing rate achieved a maximum operation frequency of 215.01 MHz. This version of the architecture reaches a processing rate of 1.72 billion samples per second, allowing it to process videos up to the resolution 8K UHD (7680x4320 pixels) at 30 frames per second.

Keywords: Video Coding, HEVC, Rotational Transform, Digital Systems, FPGA Based Design.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ALUT	*Adaptive Look-Up Table*
AVC	*Advanced Video Coding*
CABAC	*Context Adaptive Binary Arithmetic Coding*
CTB	*Coding Tree Block*
CTU	*Coding Tree Unit*
CU	*Coding Unit*
DCT	*Discrete Cosine Transform*
DF	*Deblocking Filter*
DST	*Discrete Sine Transform*
DVD	*Digital Versatile Disc*
FPGA	*Field-Programmable Gate Array*
GACI	Grupo de Arquiteturas e Circuitos Integrados
Gbps	Gigabits por segundo
HD	*High Definition*
HEVC	*High Efficiency Video Coding*
HM	*HEVC Test Model*
IEC	*International Electrotechnical Commission*
ISO	*International Organization for Standardization*
ITU-R	*International Telecommunication Union – Radiocommunication Sector*
ITU-T	*International Telecommunication Union – Telecommunication Standardization Sector*
JCT-VC	*Joint Collaborative Team on Video Coding*
MPEG	*Motion Picture Experts Group*
PU	*Prediction Unit*
QHD	*Quad High Definition*
RDOQ	*Rate Distortion Optimized Quantization*
ROT	*Rotational Transform*
SAO	*Sample Adaptive Offset*

TMuC	*Test Model under Consideration*
TU	*Transform Unit*
UFPel	Universidade Federal de Pelotas
UHD	*Ultra High Definition*
VCEG	*Video Coding Experts Group*
VHDL	*VHSIC Hardware Description Language*
VHSIC	*Very High Speed Integrated Circuit*

SUMÁRIO

1 INTRODUÇÃO

O vídeo digital tornou-se um importante recurso de entretenimento, comunicação e educação, através de diversos dispositivos de uso cotidiano, como a televisão digital, tocadores de DVD e Blu-ray, computadores pessoais, câmeras fotográficas e filmadoras, celulares e outros dispositivos móveis e portáteis capazes de reproduzir, criar e compartilhar vídeos.

Representar um vídeo digitalmente requer uma enorme quantidade de bits, que precisam ser transmitidos ou armazenados, dependendo da aplicação. Sem utilizar qualquer técnica de compressão, um filme de duas horas em resolução *Full HD* (1920x1080 pixels) ocuparia cerca de 1,25 terabyte de memória, o equivalente a 25 discos Blu-ray ou 147 DVDs. Para transmitir esse volume de dados em tempo real seria necessária uma conexão de rede com velocidade de 1,5 Gbps.

Além disso, a popularização dos *displays* de alta definição tem feito crescer a demanda por resoluções mais elevadas e maior fidelidade de imagem nos vídeos digitais (ISO/IEC, 2011). O formato UHDTV (ITU-R, 2012), em fase final de padronização, deve atingir uma resolução de até 7680x4320 pixels a 120 quadros por segundo, o que representa um volume de dados até 64 vezes maior em relação ao formato *Full HD*.

Os compressores de vídeo exploram o elevado grau de similaridade entre as imagens (ou quadros) que compõem um vídeo, codificando essas informações de forma a evitar a transmissão ou armazenamento de dados redundantes (AGOSTINI, 2007). Características pouco perceptíveis ao olho humano, como variações tênues de luminosidade e cor, também podem ser descartadas no processo de codificação, aumentando a compressão ao custo de uma pequena perda de qualidade na imagem.

A eficiência de um padrão de codificação de vídeo geralmente é medida por sua capacidade de compressão, para uma dada qualidade de imagem. Uma codificação mais eficiente permite transmitir ou armazenar mais conteúdo com a mesma qualidade, ou aumentar a resolução ou fidelidade do vídeo utilizando a mesma quantidade de bits. Assim, dada a importância dos vídeos digitais na

atualidade, existe um esforço constante da academia e da indústria em desenvolver novos e mais eficientes padrões de codificação.

A compressão de um vídeo é obtida através de uma série de algoritmos aplicados em diferentes etapas do codificador. Os padrões atuais de codificação de vídeo, tais como o MPEG-2 (ISO/IEC, 1996) e o H.264/AVC (ITU-T, 2003) realizam um processamento baseado em blocos. Nesse modelo, cada quadro do vídeo é divido em blocos de tamanho regular (por exemplo 8x8 pixels), que podem ser, eventualmente, divididos em blocos menores. Cada bloco é individualmente processado pelos diferentes módulos do codificador. O decodificador é responsável por realizar as operações inversas para reconstruir o bloco original. A Figura 1.1 apresenta um diagrama de blocos com os principais módulos utilizados na codificação e decodificação de um vídeo.

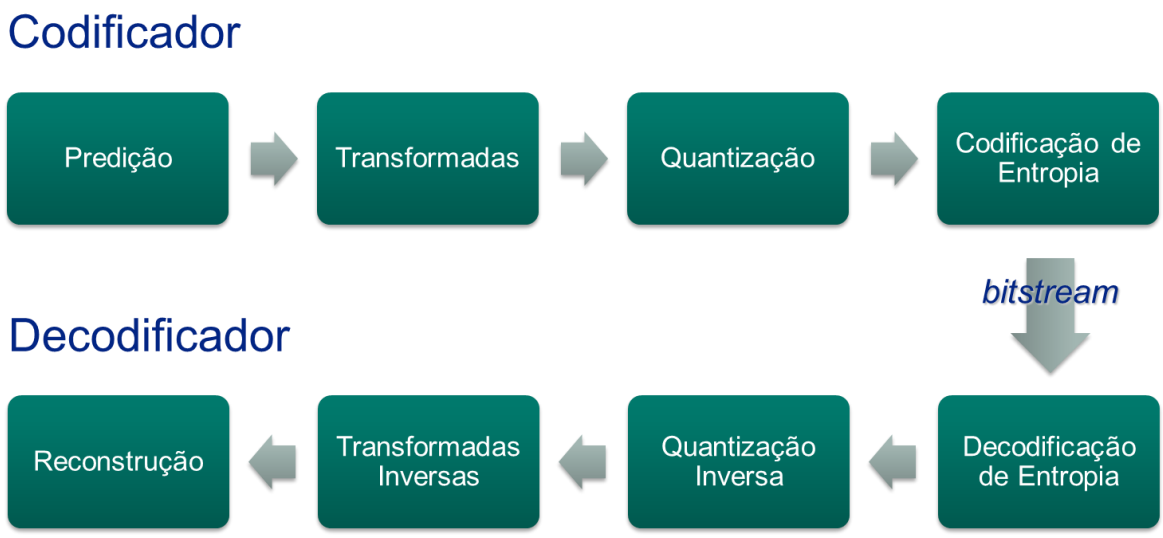

Figura 1.1: Principais módulos do codificador e do decodificador de vídeo.

No codificador, o primeiro módulo é o da predição, que tem por objetivo reduzir a redundância espacial e temporal de dados, fazendo uma predição do bloco que está sendo codificado. Na predição podem ser utilizadas amostras vizinhas já codificadas dentro do mesmo quadro (predição espacial ou intra-quadro) ou blocos semelhantes de outros quadros previamente codificados (predição temporal ou inter-quadros). A ideia é gerar uma aproximação mais precisa possível do bloco atual, reutilizando ao máximo as amostras já codificadas. O resultado desse processo é

um bloco de resíduos, que são as diferenças entre os valores preditos e os valores originais das amostras no bloco processado.

O resíduo da predição é, então, submetido ao módulo das transformadas, onde são aplicadas transformadas espaciais, como a DCT (Transformada Discreta do Cosseno), com o objetivo de concentrar a informação no bloco de resíduos. O resultado da DCT é um bloco de coeficientes, que podem ser vistos como “pesos” para um conjunto de padrões-base, compostos por combinações horizontais e verticais de funções cosseno. Geralmente, a DCT consegue concentrar a maior parte da energia (variações de informação) do bloco de resíduos em poucos desses coeficientes. Por exemplo, se todo um bloco de 16x16 pixels for formado pelo mesmo valor, apenas um coeficiente da DCT diferente de zero é necessário para representar o bloco inteiro. No decodificador, a transformada inversa é capaz de reconstruir o bloco original combinando os padrões-base da DCT multiplicados pelos respectivos coeficientes (RICHARDSON, 2010). A Figura 1.2 apresenta os padrões-base utilizados para a DCT 8x8.

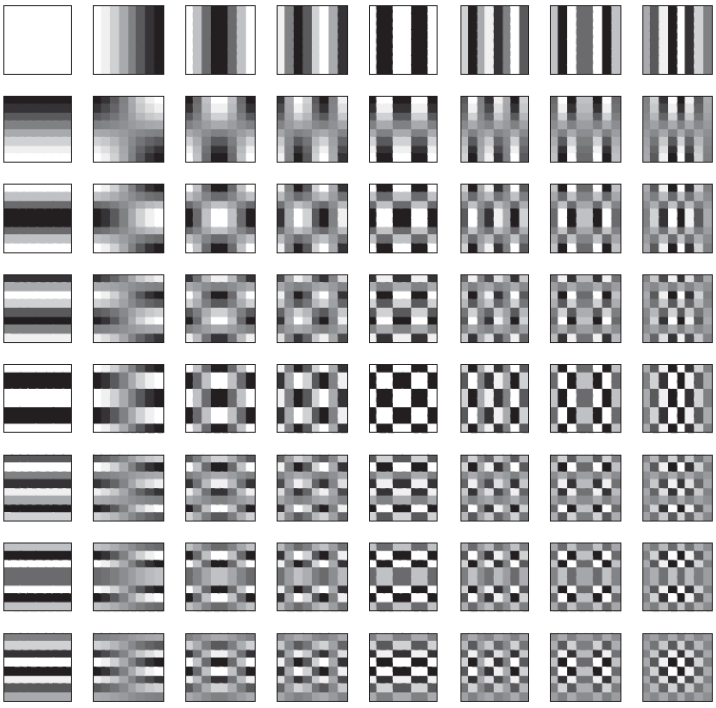

Figura 1.2: Padrões-base da DCT 8x8 (RICHARDSON, 2010)

O passo seguinte na codificação é a aplicação da quantização, que realiza uma divisão inteira sobre os elementos da matriz de coeficientes da transformada e elimina ou atenua os coeficientes de menor relevância para o sistema visual

humano. Assim, o resultado típico da quantização é uma matriz esparsa. Esse é um processo com perdas, uma vez que a quantização inversa não é capaz de restaurar exatamente os valores originais. O nível de quantização geralmente é um parâmetro ajustável no codificador, afetando diretamente a qualidade do vídeo comprimido e a sua taxa de compressão.

Finalmente, é realizada a codificação de entropia do bloco, reduzindo o número de bits utilizados para representar os diferentes símbolos na matriz quantizada. Em especial, as sequências de zeros tendem a ser completamente eliminadas, gerando uma elevada taxa de compressão.

A elevada complexidade computacional associada à soma desses processos é o principal obstáculo para se atingir maiores taxas de compressão. Entretanto, à medida que a tecnologia evolui e o poder computacional dos dispositivos aumenta, torna-se possível definir novas ferramentas de compressão.

Com este objetivo em mente, especialistas do VCEG (*Video Coding Experts Group*) do ITU-T (ITU, 2011) e do MPEG (*Motion Picture Experts Group*) da ISO/IEC (ISO, 2011) uniram esforços para elaborar um padrão para a próxima geração de codificadores de vídeo. A equipe oriunda dessa união foi denominada *Joint Collaborative Team on Video Coding* (JCT-VC) (ITU, 2012). Em janeiro de 2010 o grupo lançou uma chamada de propostas (ITU-T; ISO/IEC, 2010), convidando empresas, universidades e organizações que desenvolvem tecnologia de codificação de vídeo a submeterem seus trabalhos. As submissões deveriam incluir codificações de 18 sequências de vídeo fornecidas para testes, com diferentes resoluções e limites de taxa de bits.

Durante o mês de março de 2010 foram realizados testes de qualidade subjetiva, onde os vídeos submetidos foram comparados com versões de referência, codificadas no padrão H.264/AVC.

A primeira reunião do JCT-VC foi realizada em abril de 2010, para avaliar as submissões em resposta à chamada de propostas. Nessa reunião o nome do projeto ficou definido como *High Efficiency Video Coding* (HEVC). As características mais importantes das propostas que apresentaram melhor desempenho nos testes foram identificadas e combinadas em uma primeira implementação em software, chamada de *Test Model under Consideration* (TMuC). O TMuC tinha como objetivo facilitar o estudo das diferentes ferramentas de codificação implementadas, bem como servir de *framework* para a investigação de novas ferramentas.

Após a terceira reunião do JCT-VC, realizada em outubro de 2010, foi lançado o primeiro *draft* do padrão HEVC. Foi disponibilizada também a versão 1.0 do software de referência, que passou a ser chamado *HEVC Test Model* (HM) e incorporou apenas as principais ferramentas do TMuC.

A partir de então, o JCT-VC vem realizando reuniões periódicas, a cada três ou quatro meses. No intervalo de tempo entre duas reuniões são realizados os *Core Experiments* (CE) que agrupam ferramentas de uma mesma categoria, como codificação de entropia, transformadas, predição intra, etc. O CE7, por exemplo, tem sido o grupo de experimentos responsável por investigar as transformadas alternativas propostas para o HEVC. A cada reunião, são avaliados os resultados dos últimos experimentos realizados, e os avanços obtidos são adicionados às novas versões do *draft* e do software de referência, lançadas sempre algumas semanas após o término de cada reunião. Novas ferramentas e propostas de otimizações também continuam sendo submetidas para avaliação do grupo. Todos os documentos, como propostas submetidas e relatórios de reuniões e experimentos, estão disponíveis ao público em geral (JCT-VC, 2011a, 2011b).

A principal meta do HEVC é reduzir em aproximadamente 50% a taxa de bits necessária em relação ao H.264/AVC, mantendo a mesma qualidade subjetiva da imagem (SULLIVAN; WIEGAND, 2009). Outro requisito é que o padrão atenda às necessidades de uma vasta gama de dispositivos e aplicações, desde *webcams* até cinema digital de ultra-alta-definição, com resoluções até 8Kx4K pixels. Para manter a complexidade computacional compatível com as diversas tecnologias-alvo, o HEVC deverá oferecer diferentes configurações de operação, que permitam balancear complexidade e capacidade de compressão.

1.1 Motivação para este Trabalho

O padrão H.264/AVC, lançado em 2003 e atual estado-da-arte em codificação de vídeo, conseguiu atingir seu objetivo de dobrar a taxa de compressão em relação ao padrão anterior, o MPEG-2. Entretanto, a demanda por maiores resoluções e melhor qualidade de imagem nos vídeos digitais continua crescendo, enquanto as redes de comunicação de dados, especialmente as tecnologias sem fio, continuam sendo um fator limitante para a transmissão de vídeos HD com alta qualidade (ISO/IEC, 2011).

O HEVC está sendo desenvolvido com o objetivo de obter um ganho significativo de compressão em relação ao H.264/AVC e, assim, atender às demandas da próxima geração de vídeos digitais. Para atingir esse objetivo, diversas mudanças estruturais e novas ferramentas algorítmicas estão sendo propostas e investigadas, num esforço conjunto da indústria e da academia, que reúne centenas de pesquisadores ao redor do mundo.

Uma das novas ferramentas propostas para o HEVC é a Transformada Rotacional (ROT), uma transformada secundária aplicada seletivamente pelo codificador após a DCT, com o objetivo de melhor compactar a energia nos blocos de resíduos da predição intra-quadro e melhorar a codificação de entropia. Na versão atual do padrão, a transformada ROT não está inserida, principalmente em função da complexidade adicional introduzida no codificador. Entretanto, esta ferramenta apresentou resultados interessantes de ganho de compressão e, por isso, é muito provável que seja inserida nos novos padrões de codificação, caso realmente permaneça fora do HEVC.

Este trabalho está focado na investigação e no desenvolvimento de soluções inovadoras de hardware para as transformadas ROT propostas para o padrão emergente HEVC. A motivação para o desenvolvimento deste trabalho originou-se de três fatores principais: (i) a relevância dos vídeos digitais na atualidade, nas áreas de entretenimento, comunicação e educação; (ii) a importância do desenvolvimento de hardware dedicado à codificação e decodificação de vídeo, especialmente para dispositivos móveis; e (iii) a possibilidade de trabalhar com um tema extremamente atual e completamente aberto a novas contribuições.

1.2 Objetivos

Arquiteturas de hardware dedicadas à codificação e decodificação de vídeo são essenciais para garantir o desempenho necessário ao processamento de vídeos em tempo real, com baixo consumo de energia e potência, fatores especialmente críticos em dispositivos móveis e portáteis.

Até onde se tem conhecimento, não existem trabalhos na literatura abordando a implementação em hardware da Transformada Rotacional. Assim, o objetivo deste trabalho é investigar o conjunto de transformadas que compõem a proposta da Transformada Rotacional, visando o desenvolvimento de arquiteturas eficientes para

o cálculo da ROT, que possam ser utilizadas em codificadores e decodificadores em hardware do futuro padrão HEVC ou de outros padrões que incorporem esta ferramenta.

1.3 Organização do Texto

Este trabalho está dividido em seis capítulos. No segundo capítulo é realizada uma introdução ao padrão HEVC, apresentando suas principais características e alguns conceitos que serão necessários para melhor compreensão deste trabalho.

O terceiro capítulo apresenta a fundamentação teórica da Transformada Rotacional, a evolução da proposta até a versão utilizada neste trabalho e a exploração realizada nos algoritmos da ROT, visando a implementação em hardware. O quarto capítulo apresenta as arquiteturas desenvolvidas para a ROT direta e inversa. O quinto capítulo relata as metodologias utilizadas na síntese e na validação das arquiteturas, e apresenta os resultados obtidos. Por fim, o sexto capítulo apresenta as conclusões desta dissertação.

2 PADRÃO HEVC

O padrão HEVC adota o esquema híbrido de codificação em blocos já consagrado em padrões anteriores, baseado na predição com compensação de movimento, transformadas e codificação de entropia de alta eficiência (KIM et al., 2012). Este capítulo apresenta uma visão geral das características do HEVC, no atual estado em que se encontra o padrão. As informações aqui apresentadas baseiam-se no *HEVC Text Specification Draft 8* (BROSS et al., 2012) e no *HEVC Test Model 8 Encoder Description* (KIM et al., 2012), que são as versões mais recentes publicadas até a finalização deste trabalho.

Na primeira reunião do JCT-VC, em abril de 2010, foi definido o *Test Model under Consideration* (TMuC), um modelo de codificador e decodificador em software a ser utilizado como referência nos testes do novo padrão. Na terceira reunião, em outubro de 2010, o JCT-VC definiu a primeira versão do *HEVC Test Model* (HM1). O HM1 incorporou apenas as principais ferramentas do TMuC, reduzindo substancialmente a complexidade do software de referência. A partir de então, a cada reunião subsequente uma nova versão do modelo de testes é apresentada, incorporando otimizações que proporcionam melhor desempenho do que a versão anterior, em termos da relação entre eficiência de codificação e complexidade computacional.

2.1 Configurações do Codificador

A oitava versão do modelo de testes do HEVC (HM8) foi definida na décima reunião do JCT-VC, realizada de 11 a 20 de julho de 2012. Duas configurações básicas foram definidas para utilização nos testes: *Main* e *High Efficiency*. A configuração *Main* contém as principais ferramentas de codificação, enquanto a configuração *High Efficiency* permite a utilização de ferramentas adicionais, visando máxima eficiência de compressão, ao custo de maior complexidade computacional.

Dentro de cada uma dessas configurações, outras três configurações foram definidas para especificar as restrições estruturais dos vídeos codificados: *Intra Only, Random Access* e *Low Delay*. Na configuração *Intra Only* todos os quadros do vídeo são codificados utilizando apenas predição intra-quadro. A configuração *Random Access* visa possibilitar pontos frequentes de acesso aleatório no vídeo codificado, estipulando a utilização de quadros tipo I em intervalos de, no máximo, 1,1 segundo e definindo uma estrutura hierárquica que limita o número de quadros que podem ser utilizados como referência na predição inter-quadros (SULLIVAN; OHM, 2010a). Na configuração *Low Delay*, não é permitida a reordenação de quadros entre o processamento e a saída do codificador, e somente quadros já exibidos podem ser usados como referência na predição inter-quadros, visando comunicações interativas em tempo real, com baixo atraso.

2.2 Elementos Estruturais

Com o intuito de obter maior compressão em vídeos de alta resolução, uma das principais estratégias empregadas pelo HEVC foi a adoção de estruturas de blocos maiores, com mecanismos mais flexíveis de subdivisão (SULLIVAN; OHM, 2010b). Outra inovação foi a introdução de unidades de particionamento específicas para os módulos de predição (as *Prediction Units*) e de transformadas (as *Transform Units*). A seguir, serão apresentados os principais elementos estruturais do HEVC.

2.2.1 *Coding Tree Units* (CTUs) e *Coding Tree Blocks* (CTBs)

Cada quadro do vídeo é dividido em uma sequência de unidades elementares, de mesmo tamanho, chamadas *Coding Tree Units* (CTUs). Uma CTU consiste em um bloco de NxN amostras de luminância e os dois blocos de crominância correspondentes (KIM et al., 2012). Cada bloco é individualmente chamado de *Coding Tree Block* (CTB).

As CTUs substituem o conceito de macrobloco utilizado em padrões anteriores de codificação, como o H.264/AVC, e já foram chamadas também de *Treeblocks* e *Largest Coding Units*, em versões anteriores do *draft* de especificação do HEVC. Na versão atual, o tamanho máximo do bloco de luminância de uma CTU é de 64x64 amostras.

2.2.2 *Slices* e *Tiles*

Um *slice* é especificado como uma unidade de agrupamento de dados de codificação para fins de transmissão. *Slices* são projetados para serem decodificáveis de forma independente. Assim, a predição não pode utilizar dados de *slices* diferentes e a codificação de entropia é reiniciada a cada *slice*. Um *slice* consiste de um cabeçalho e uma sequência de blocos em ordem de codificação. A Figura 2.1 demonstra o particionamento de uma imagem em dois *slices*.

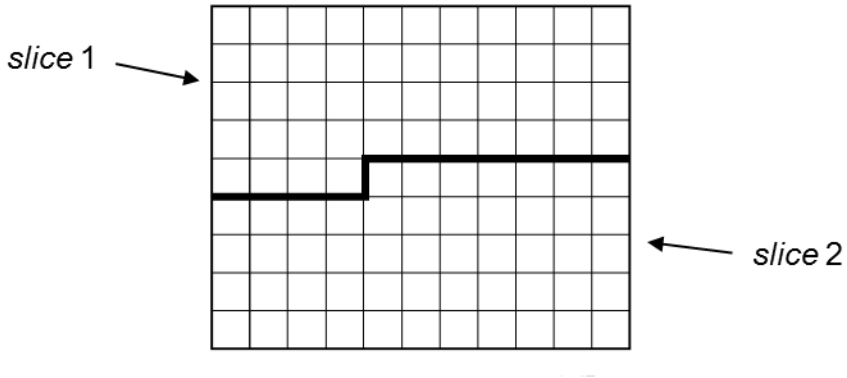

Figura 2.1: Imagem de 11x9 blocos particionada em dois *slices*

Tiles, assim como os *slices*, são sequências de blocos de codificação. Os *tiles*, entretanto, são sempre retangulares e contêm um número inteiro de CTBs em ordem de varredura na tela (*raster scan*). A Figura 2.2 ilustra o particionamento de uma imagem em três *tiles*.

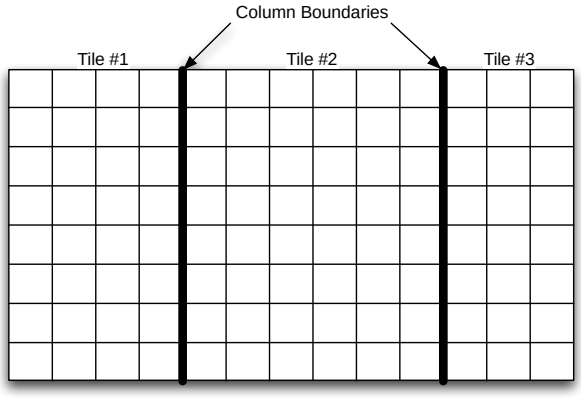

Figura 2.2: Imagem de 13x8 blocos particionada em três *tiles* (BROSS et al., 2012)

Um *tile* pode consistir de blocos contidos em mais de um *slice*, e um *slice* pode consistir de blocos contidos em mais de um *tile*. Entretanto, pelo menos uma das seguintes condições deve ser satisfeita para cada *slice* e *tile*: (i) todos os blocos de um *slice* pertencem ao mesmo *tile*; (ii) todos os blocos de um *tile* pertencem ao mesmo *slice* (BROSS et al., 2012).

2.2.3 *Coding Units* (CUs)

As *Coding Units* (CUs) são as unidades básicas utilizadas na codificação. As CUs são sempre quadradas e seu tamanho máximo é o tamanho da CTU. Cada CU pode ser recursivamente subdividida em quatro blocos do mesmo tamanho, até o limite mínimo de 8x8 amostras de luminância, formando uma estrutura de árvore quaternária (CORRÊA, 2011), ou *quadtree,* conforme apresentado na Figura 2.3.

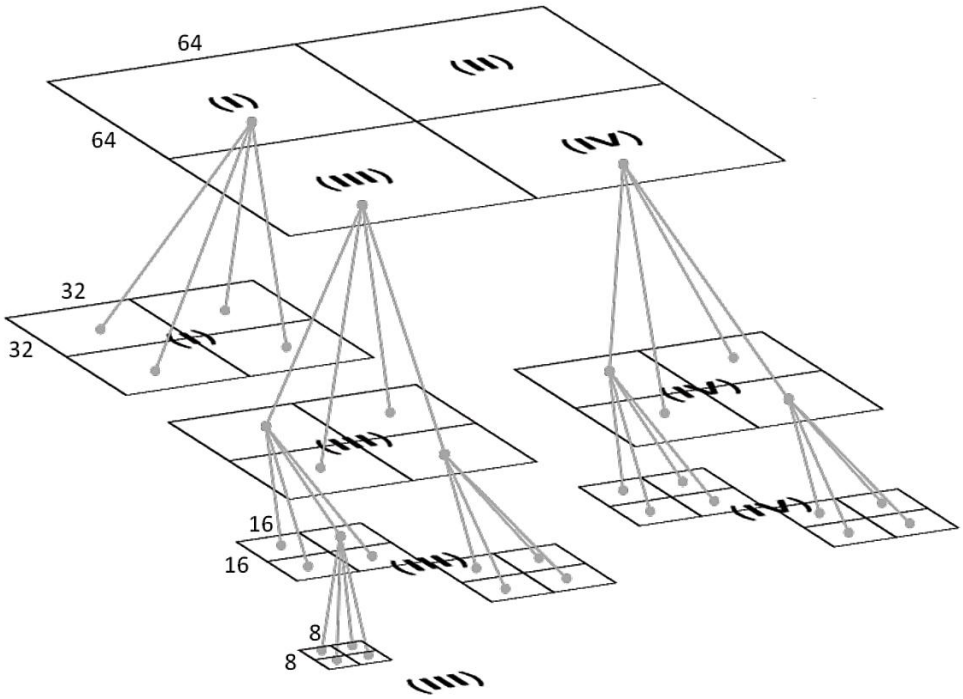

Figura 2.3: Exemplo de subdivisão recursiva de CUs em árvores quaternárias (CORRÊA, 2011)

2.2.4 *Prediction Units* (PUs)

As *Prediction Units* (PUs) são as unidades básicas que transportam as informações dos processos de predição. Cada CU pode conter uma ou mais PUs.

A predição inter-quadros permite PUs quadradas ou retangulares, a fim de facilitar o casamento com bordas de objetos nas imagens (KIM et al., 2012). Considerando o tamanho da CU como 2Nx2N, as PUs podem assumir tamanhos 2Nx2N, 2NxN, Nx2N e NxN, sendo o menor tamanho permitido 8x4 ou 4x8 amostras de luminância. Uma inovação em relação a padrões anteriores é a possibilidade de partições assimétricas em uma das dimensões da PU, com tamanhos N/2 e 3N/2. A divisão NxN e as divisões assimétricas só são permitidas para CUs maiores do que 8x8 (KIM et al., 2012).

A Figura 2.4 apresenta um exemplo dos formatos possíveis de subdivisão de PUs, considerando uma CU de tamanho 32x32. Os formatos assimétricos são demonstrados nas Figuras 2.4(e), 2.4(f), 2.4(g) e 2.4(h), com suas respectivas nomenclaturas.

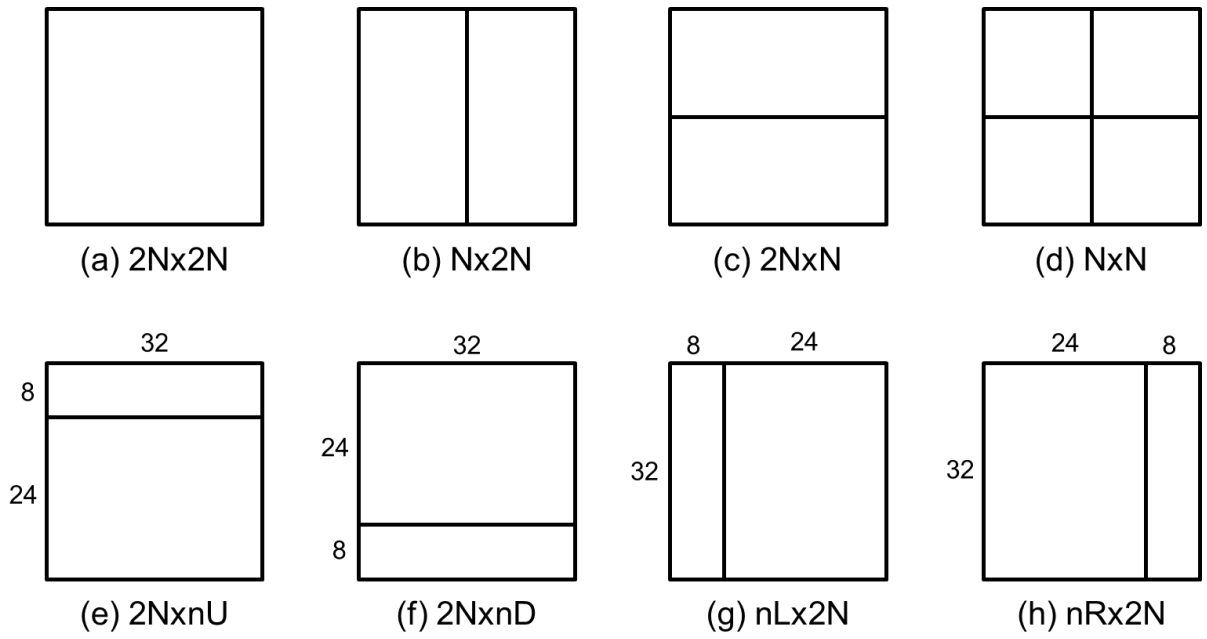

Figura 2.4: Formatos possíveis de subdivisão de PUs, para uma CU 32x32

Para a predição intra-quadro são suportadas apenas PUs quadradas, no formato 2Nx2N (mesmo tamanho da CU) ou NxN.

2.2.5 *Transform Units* (TUs)

As *Transform Units* (TUs) são unidades básicas utilizadas pelos processos de transformadas e quantização. Cada CU pode ter uma ou mais TUs e múltiplas TUs também podem ser organizadas hierarquicamente em uma estrutura de árvore quaternária. As TUs transportam para o módulo das transformadas os dados resultantes do processamento das PUs, ou seja, os resíduos gerados pela predição intra-quadro ou inter-quadros. As TUs são sempre quadradas e podem ter um tamanho máximo de 32x32 e mínimo de 4x4 amostras de luminância.

Para cada CU processada, o codificador avalia todos os tamanhos possíveis de TU e monta uma árvore final com a combinação de transformadas que oferece o melhor resultado de compressão. Um exemplo de subdivisão de TUs é apresentado na Figura 2.5. Neste exemplo, uma CU de 32x32 amostras foi dividida em quatro TUs de 16x16 amostras. Três dessas TUs 16x16 foram novamente divididas em quatro TUs de 8x8 amostras. Por fim, uma TU 8x8 foi dividida em quatro TUs de 4x4 amostras. Assim, a árvore gerada pelo codificador com a melhor combinação de transformadas para essa CU, utilizou uma transformada 16x16, onze transformadas 8x8 e quatro transformadas 4x4.

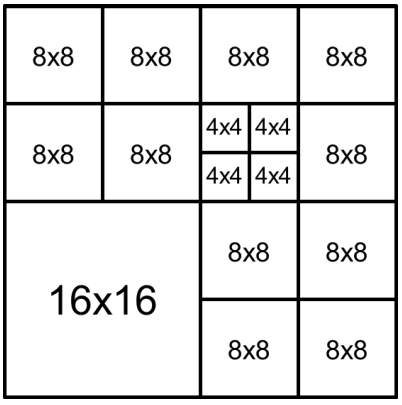

Figura 2.5: Exemplo de subdivisão de TUs em árvores quaternárias

2.3 Ferramentas de Codificação

Além das modificações estruturais apresentadas na seção anterior, novas ferramentas de codificação também estão sendo investigadas e incorporadas ao HEVC, à medida que os trabalhos de padronização avançam. Nesta seção, serão apresentadas as principais ferramentas de codificação presentes na atual versão do modelo de testes do HEVC.

2.3.1 Predição Intra-quadro

A predição intra-quadro, ou simplesmente intra, utiliza amostras de PUs vizinhas à atual, previamente codificadas, para gerar as amostras preditas da PU atual. Quadros que utilizam a predição intra em todas as PUs são chamados de quadros tipo I. Os quadros tipo I são especialmente importantes, pois, por não dependerem de informações de outros quadros, permitem a sincronização do sinal de vídeo (ao trocar o canal em uma transmissão de TV digital, por exemplo) e o acesso aleatório dentro de um arquivo de vídeo.

Cada PU codificada com a predição intra deve indicar um modo de predição para os componentes de luminância e um modo de predição para os componentes de crominância. Todas as TUs dentro de uma PU devem usar o mesmo modo para cada componente (KIM et al., 2012).

Para os componentes de luminância podem ser utilizados até 35 modos de predição direcional, incluindo os modos DC e plano, conforme ilustrado na Figura 2.6. O número de modos efetivamente disponíveis para cada PU dependerá do tamanho da PU. As setas indicam a posição relativa da primeira amostra utilizada na predição em relação à PU atual. Amostras vizinhas no mesmo ângulo de inclinação são extrapoladas ou interpoladas para compor as amostras da PU predita.

No modo DC, todas as amostras da PU predita são compostas pelo valor médio das amostras vizinhas à borda superior e à borda esquerda da PU. O modo Plano utiliza uma função linear para interpolar as amostras vizinhas às bordas da PU, gerando as amostras preditas.

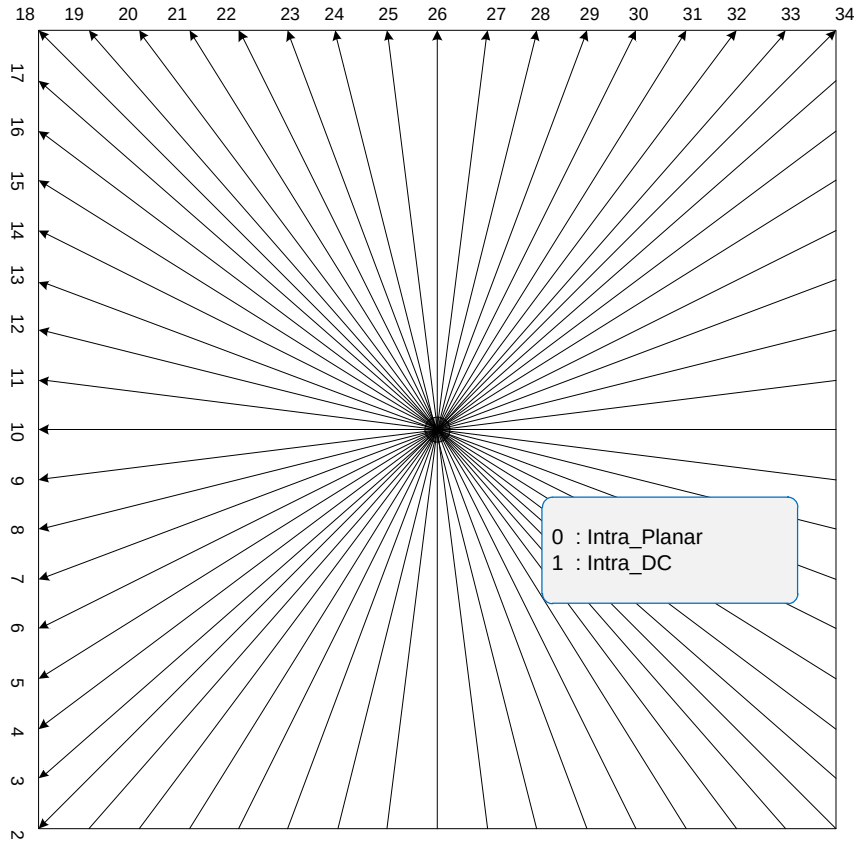

Figura 2.6: Modos de predição direcional da predição intra-quadro (BROSS et al., 2012)

Para os componentes de crominância, o codificador pode escolher entre 5 modos de predição: plano, DC, horizontal, vertical e uma cópia direta do modo de predição utilizado para os componentes de luminância da PU.

2.3.2 Predição Inter-quadros

Cada PU codificada pela predição inter-quadros, ou simplesmente inter, deve conter um conjunto de parâmetros de movimento, composto de um vetor de movimento, o índice do quadro de referência e o *flag* de utilização de uma lista de quadros de referência. Estes parâmetros podem ser sinalizados de forma explícita ou implícita (KIM et al., 2012). Quando uma CU é codificada no modo *skip*, ela será composta de apenas uma PU cujos vetores de movimento, índices de quadros de referência e *flags* de utilização da lista de quadros de referência serão obtidos pela técnica de *Motion Merge* (KIM et al., 2012).

A técnica de *Motion Merge* consiste em encontrar uma PU vizinha codificada pela predição inter-quadros cujos parâmetros de movimento (vetor de movimento, índice do quadro de referência, *flag* de uso da lista de quadros de referência) possam ser utilizados como parâmetros da PU atual. O codificador pode selecionar os melhores parâmetros entre múltiplos candidatos, formados por PUs espacialmente e temporalmente vizinhas, e transmitir o índice correspondente ao candidato escolhido.

A *Motion Merge* não é restrita ao modo *skip*. Para qualquer PU codificada pela predição inter, o codificador tem a liberdade de transmitir explicitamente os parâmetros de movimento ou utilizar a técnica de *Motion Merge*.

Quando os parâmetros de movimento são explicitamente transmitidos, outra técnica pode ser utilizada, chamada *Motion Vector Prediction* (KIM et al., 2012). Esta técnica explora a correlação espaço-temporal dos vetores de movimento entre PUs vizinhas. Uma lista de vetores candidatos é construída a partir das PUs acima e à esquerda da PU atual e de PUs temporalmente vizinhas. O codificador seleciona o melhor candidato e transmite apenas o índice correspondente na lista.

No decodificador, a compensação de movimento é realizada com precisão de 1/4 de pixel para amostras de luminância e 1/8 de pixel para amostras de crominância. Para o cálculo das posições fracionárias são utilizados filtros de interpolação baseados na DCT, com 8 *taps* para luminância e 4 *taps* para crominância (KIM et al., 2012).

2.3.3 Transformadas

A transformada principal (*core transform*) utilizada no HEVC é uma aproximação inteira da DCT, já consagrada em padrões anteriores de codificação de vídeo. A DCT está disponível nos tamanhos 4x4, 8x8, 16x16 e 32x32 amostras, o que é uma novidade do HEVC, já que padrões anteriores usavam, no máximo, dois tamanhos de transformadas: 4x4 e 8x8 (RICHARDSON, 2010). Especificamente para TUs 4x4 de resíduos da predição intra, pode ser aplicada também a DST (Transformada Discreta do Seno) para os componentes de luminância.

Transformadas adicionais, como a Transformada Rotacional (ROT) e as Transformadas Secundárias Dependentes de Modo (MD-ST), estão sendo

investigadas através de experimentos, com o objetivo de melhorar a concentração de energia nas matrizes de coeficientes resultantes da DCT.

2.3.4 Quantização e Codificação de Entropia

A técnica de quantização otimizada para taxa-distorção (RDOQ) é utilizada, de forma semelhante ao H.264/AVC, para selecionar a quantização dos coeficientes considerando o impacto no *bitrate* e na qualidade (KIM et al., 2012).

Para a codificação de entropia, o HEVC utiliza uma versão aperfeiçoada do método de codificação aritmética binária adaptativa ao contexto (CABAC), já utilizado no H.264/AVC. Uma das melhorias realizadas no CABAC, em relação ao H.264/AVC, foi a paralelização do processo, através da técnica chamada *Wavefront Parallel Processing* (KIM et al., 2012). Propostas ainda estão sendo investigadas com o intuito de aperfeiçoar outras características, como: uso de memória, taxa de processamento e mecanismo de atualização de probabilidades.

2.3.5 Filtros

Dois processos de filtragem são aplicados após a reconstrução da imagem, com o objetivo de obter melhor qualidade visual: o *Deblocking Filter* e o *Sample Adaptive Offset*.

O *Deblocking Filter* (DF) visa reduzir o efeito de bloco causado pelo processo de transformada e quantização do codificador (CORRÊA, 2011). Filtros horizontais e verticais são aplicados às bordas de blocos 8x8 selecionados para filtragem, tanto para componentes de luminância quanto de crominância. Diferentemente do padrão H.264/AVC, bordas de blocos 4x4 não são filtradas, para reduzir a complexidade computacional.

O processo do DF envolve bordas de CUs, bordas de TUs e bordas de PUs, porém nem todas as bordas envolvidas são efetivamente filtradas. A intensidade da borda é calculada a partir do modo de predição intra, da existência de coeficientes de transformada não-zero e das informações de movimento, como número de vetores de movimento e diferença entre vetores de movimento. Com base na intensidade da borda e em outros dois parâmetros, derivados do parâmetro de quantização (QP) empregado nos blocos envolvidos, uma das seguintes opções é

selecionada para cada borda: sem filtragem, filtragem fraca e filtragem forte (KIM et al., 2012).

O *Sample Adaptive Offset* (SAO) é aplicado opcionalmente após o DF. O propósito do SAO é classificar pixels em categorias e reduzir a distorção entre a imagem original e a imagem reconstituída, aplicando um valor de compensação (*offset*) aos pixels de cada categoria. São utilizadas duas técnicas diferentes para classificação dos pixels: *Edge Offset* e *Band Offset* (KIM et al., 2012).

Edge Offset classifica cada pixel de acordo com dois pixels vizinhos, avaliando as características direcionais de bordas na imagem. O valor do pixel é compensado, de acordo com a direção detectada (0, 45, 90 ou 135 graus) e a diferença entre o pixel avaliado e os pixels vizinhos.

A técnica de *Band Offset* classifica todos os pixels de um bloco em 32 faixas uniformes, de acordo com a intensidade do pixel. A seguir, quatro faixas adjacentes são agrupadas ajustando o deslocamento entre as faixas, e o grupo é referenciado pelo valor inicial da faixa mais à esquerda. O codificador deverá procurar o agrupamento que resulte na maior redução de distorção.

Além destes dois filtros, um terceiro filtro, chamado de *Adaptive Loop Filter* (ALF) foi considerado para ser inserido no HEVC, tendo sido recentemente excluído do padrão (KIM et al., 2012).

3 TRANSFORMADA ROTACIONAL

A Transformada Rotacional (ROT) faz parte do trabalho apresentado por McCann et al. (2010) em resposta à chamada de propostas do JCT-VC. A ROT é recomendada como uma transformada adicional, a ser aplicada após a DCT sobre os componentes de luminância de resíduos da predição intra. Os autores da proposta argumentam que a DCT tem um desempenho sub-ótimo para resíduos com determinadas características como, por exemplo, componentes diagonais acentuados, que não podem ser representados eficientemente pelos padrões-base da DCT (MCCANN et al., 2010).

Este capítulo apresenta, inicialmente, os conceitos teóricos da Transformada Rotacional proposta para o padrão HEVC. Em seguida, é apresentada a exploração algorítmica realizada neste trabalho para a simplificação das equações originais, visando a implementação da transformada em hardware.

3.1 Definições Teóricas

A proposta da Transformada Rotacional apresenta uma solução de baixa complexidade que pode ser aplicada a TUs de 8x8, 16x16 e 32x32 amostras de luminância. A proposta inicial previa ainda uma versão da transformada exclusiva para TUs de tamanho 4x4, mas essa opção foi removida dos trabalhos subsequentes.

A Figura 3.1 ilustra o contexto de aplicação da ROT no codificador. Após a aplicação da DCT, as TUs de tamanhos 8x8, 16x16 e 32x32 são submetidas à ROT. No caso de TUs maiores do que 8x8, apenas um bloco de 8x8 amostras do canto superior esquerdo da TU é processado pela ROT. Isso se deve ao fato de que os coeficientes mais significativos da DCT encontram-se concentrados nessa região de baixas frequências.

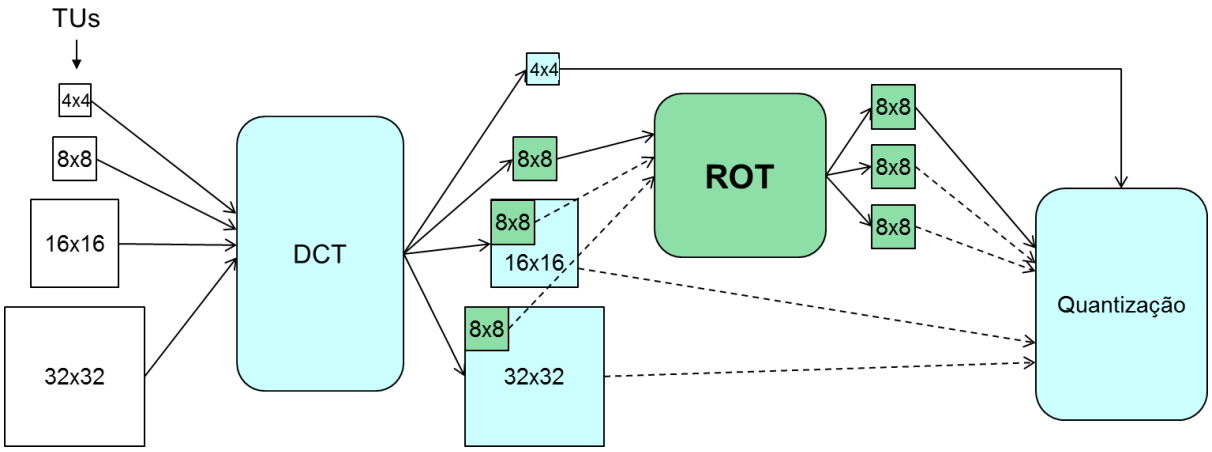

Figura 3.1: Contexto de aplicação da ROT, para diferentes tamanhos de TU

No decodificador, é calculada a ROT inversa, reconstruindo os coeficientes DCT originais.

Utilizando matrizes compostas de rotação de Givens (GIVENS, 1958) e ângulos de rotação selecionados, a Transformada Rotacional promove uma troca parcial de energia entre as linhas e colunas da matriz de coeficientes resultante da DCT, melhorando a compactação de energia e minimizando o erro de quantização (MA et al., 2011a).

A ROT é essencialmente um produto de três matrizes. A equação da ROT direta é apresentada em (1).

$$W = R_h \cdot M \cdot R_v^T \tag{1}$$

Onde *M* é um bloco 8x8 de coeficientes da DCT, R_h e R_v são as matrizes da ROT horizontal e ROT vertical respectivamente e *W* é o bloco de coeficientes transformados.

De forma análoga, a ROT inversa é obtida através da equação em (2).

$$M = R_v^T \cdot W \cdot R_h \tag{2}$$

Onde *W* é um bloco 8x8 de coeficientes previamente processados pela ROT direta, R_v e R_h são as matrizes da ROT vertical e ROT horizontal respectivamente e *M* é o bloco com os coeficientes DCT originais.

As matrizes R_v e R_h são definidas como produtos de matrizes de rotação, conforme apresentado em (3) e (4).

$$R_v = R_z(\theta_1)R_x(\theta_2)R_z(\theta_3) \tag{3}$$

$$R_h = R_z(\theta_4)R_x(\theta_5)R_z(\theta_6) \tag{4}$$

As matrizes $R_x(\theta)$ e $R_z(\theta)$ são matrizes compostas de rotação de Givens e realizam rotações sobre dois pares de eixos simultaneamente. Essas matrizes estão definidas em (5) e (6).

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 & 0 & 0 & 0 & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cos(\theta) & -\sin(\theta) & 0 & 0 \\ 0 & 0 & 0 & 0 & \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

$$R_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 & 0 & 0 & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos(\theta) & -\sin(\theta) & 0 & 0 & 0 \\ 0 & 0 & 0 & \sin(\theta) & \cos(\theta) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (6)$$

Conforme pode-se observar em (3) e (4), são utilizados seis diferentes ângulos nas matrizes de rotação. Com o intuito de minimizar a complexidade computacional no lado do codificador, McCann et al (2010) propuseram quatro conjuntos pré-definidos de ângulos de rotação, de modo que os elementos não-zero das matrizes R_h e R_v pudessem ser previamente calculados e armazenados como constantes. A estrutura final das matrizes R_h e R_v proposta por McCann et al (2010) está apresentada em (7) e (8).

$$R_h = \begin{bmatrix} C_1 & C_2 & C_3 & 0 & 0 & 0 & 0 & 0 \\ C_4 & C_5 & C_6 & 0 & 0 & 0 & 0 & 0 \\ C_7 & C_8 & C_9 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{10} & C_{11} & C_{12} & 0 & 0 \\ 0 & 0 & 0 & C_{13} & C_{14} & C_{15} & 0 & 0 \\ 0 & 0 & 0 & C_{16} & C_{17} & C_{18} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (7)$$

$$R_v = \begin{bmatrix} C_{19} & C_{20} & C_{21} & 0 & 0 & 0 & 0 & 0 \\ C_{22} & C_{23} & C_{24} & 0 & 0 & 0 & 0 & 0 \\ C_{25} & C_{26} & C_{27} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{28} & C_{29} & C_{30} & 0 & 0 \\ 0 & 0 & 0 & C_{31} & C_{32} & C_{33} & 0 & 0 \\ 0 & 0 & 0 & C_{34} & C_{35} & C_{36} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (8)$$

Os quatro conjuntos de ângulos de rotação permitem melhor adaptação da transformada para blocos de resíduos com diferentes características. Para cada TU, o codificador deverá testar as quatro opções de transformada e escolher aquela que resultar no melhor custo taxa-distorção após a quantização. Essa opção é informada ao decodificador através do sinal indexROT = 1,...,4 codificado no *bitstream*. O codificador pode, também, optar por não utilizar a ROT, informando indexROT = 0.

A Figura 3.2 demonstra como a ROT transfere energia de maneira diferente, dependendo do conjunto de ângulos utilizado. A Figura 3.2(a) apresenta uma matriz de entrada com coeficientes da DCT, a Figura 3.2(b) apresenta a matriz resultante da aplicação da ROT com indexROT = 1 e a Figura 3.2(c) apresenta a matriz resultante da ROT com indexROT = 4. Blocos mais escuros indicam coeficientes mais próximos de zero, enquanto blocos mais claros representam coeficientes com maior valor absoluto. É possível perceber, especialmente na Figura 3.2(c), a concentração de elementos com maior amplitude em um menor número de coeficientes, com o aumento de coeficientes iguais ou próximos a zero, em relação à matriz original.

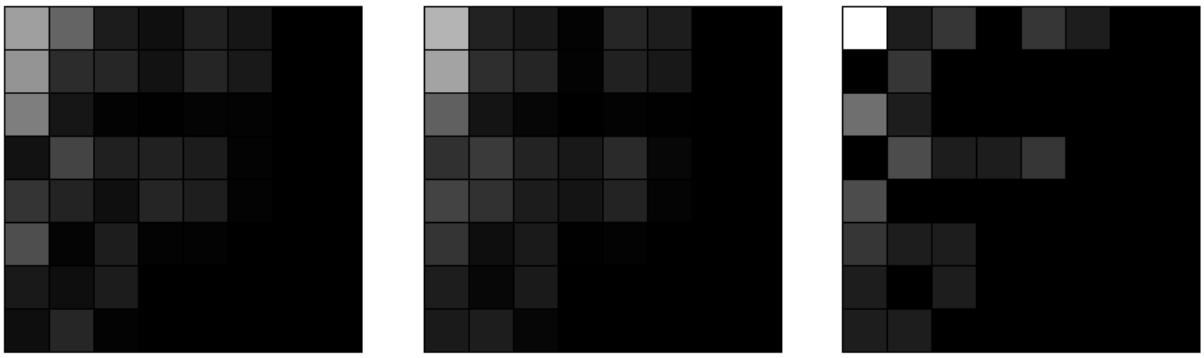

Figura 3.2: Exemplo de compactação de energia com a aplicação da ROT: (a) matriz de entrada com coeficientes DCT; (b) matriz de saída para indexROT = 1; (c) matriz de saída para indexROT = 4

A ROT foi implementada nas versões iniciais do software de referência do HEVC (JCT-VC, 2011c), até o TMuC 0.9. Eram utilizadas duas matrizes de constantes, uma para a ROT direta e outra para a ROT inversa. Cada matriz era composta por 4x36 elementos, contemplando as 36 constantes ilustradas em (7) e (8) para os quatro diferentes conjuntos de ângulos de rotação. As constantes da ROT direta utilizavam 12 bits de precisão e as da ROT inversa 8 bits.

A partir da versão HM 1.0, apenas a DCT foi mantida no módulo das transformadas, enquanto as transformadas alternativas, incluindo a ROT, eram investigadas mais a fundo através do grupo de experimentos CE7 (*Core Experiment 7*) (COHEN; YEO; JOSHI, 2010) e novas otimizações foram sendo propostas.

Na terceira reunião do JCT-VC, Fernandes (2010) apresentou uma proposta de redução de complexidade para a ROT, fatorando cada uma das matrizes de rotação em um produto de matrizes. Esta técnica, conhecida como *lifting*, é explicada a seguir.

Considerando uma matriz de rotação $R(\theta)$, definida em (9).

$$R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{9}$$

A matriz $R(\theta)$ pode ser fatorada em um produto de matrizes de *lifting*, como apresentado em (10).

$$R(\theta) = \begin{bmatrix} 1 & u \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ v & 1 \end{bmatrix} \begin{bmatrix} 1 & w \\ 0 & 1 \end{bmatrix} \tag{10}$$

Os elementos *u*, *v* e *w* estão definidos em (11) e (12).

$$u = w = -\tan\left(\frac{\theta}{2}\right) \tag{11}$$

$$v = \sin(\theta) \tag{12}$$

A mesma técnica é aplicada então às matrizes de rotação compostas $R_x(\theta)$ e $R_z(\theta)$ utilizadas na ROT, e as matrizes fatoradas são apresentadas em (13) e (14).

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & u & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & u & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & v & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & v & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & w & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & w & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (13)$$

$$R_z(\theta) = \begin{bmatrix} 1 & u & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & u & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ v & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & v & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & w & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & w & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (14)$$

A redução de complexidade é obtida aproximando-se os valores de *u*, *v* e *w* a racionais na forma $k/2^m$, ou seja, um inteiro sobre uma potência de 2. Dessa forma, as constantes utilizadas nas multiplicações serão sempre números inteiros e a divisão por uma potência de 2 é implementada simplesmente com o deslocamento de bits à direita, sem custo adicional de hardware.

Por exemplo, considerando um dos ângulos de rotação propostos, θ = 0,439062 (em radianos). O valor de *v* nesse caso seria sin(θ) = 0,425091. Multiplicando-se este valor por 32 (2^5) chega-se ao valor 13,6. Este resultado é aproximado para o inteiro mais próximo e obtém-se 14 como a constante *v*.

Outra vantagem apontada pelos autores dessa proposta é a perfeita inversibilidade da transformada utilizando as mesmas matrizes de constantes, bastando inverter o sinal dos elementos para o cálculo da ROT inversa.

Na sexta reunião do JCT-VC, realizada em julho de 2011, Ma et al. (2011a) apresentaram resultados de um experimento utilizando essa técnica, com aproximação dos elementos de *lifting* a racionais na forma k/32. Assim, cada multiplicação necessita apenas 5 bits de precisão e torna-se facilmente implementável com baixo custo de hardware, utilizando-se somas e deslocamentos.

Os experimentos apresentados pelos autores apontam uma redução no BD-rate (BJØNTEGAARD, 2001) em torno de 1% para a configuração *Intra Only*, e em

torno de 0,5% para a configuração *Random Access*. Esses resultados são corroborados também por Joshi (2011).

Ma et al. (2011a) apresentaram, também, as matrizes de constantes propostas e um pseudocódigo detalhado para o cálculo da ROT direta e inversa, com a técnica de *lifting*. Essa proposta mostrou-se ideal para implementação em hardware e foi a principal referência para o desenvolvimento das arquiteturas apresentadas no decorrer deste trabalho.

As Tabelas 3.1, 3.2 e 3.3 apresentam os valores das constantes *u*, *v* e *w*, respectivamente, utilizadas nas matrizes vertical e horizontal, para as quatro opções de ROT.

Tabela 3.1: Valores da constante ***u*** utilizada nas matrizes de *lifting*.

	Vertical			Horizontal			indexROT
u	-7	-7	5	-4	-2	-2	1
	4	-22	9	-7	-4	5	2
	-3	-5	-3	2	-29	5	3
	-10	-6	-4	-1	-29	6	4

Tabela 3.2: Valores da constante ***v*** utilizada nas matrizes de *lifting*.

	Vertical			Horizontal			indexROT
v	14	13	-10	8	3	4	1
	-8	30	-17	13	7	-10	2
	5	9	6	-5	32	-10	3
	18	11	8	1	32	-12	4

Tabela 3.3: Valores da constante ***w*** utilizada nas matrizes de *lifting*.

	Vertical			Horizontal			indexROT
w	-7	-7	5	-4	-1	-2	1
	4	-22	9	-7	-4	5	2
	-3	-5	-3	2	-29	5	3
	-10	-6	-4	-1	-29	6	4

Comparando-se as Tabelas 3.1 e 3.3 pode-se observar que as constantes *u* e *w* diferem apenas na quinta posição para indexROT = 1. Embora *u* e *w* ambos representem -tan(θ/2), como visto em (11), Fernandes (2010) justifica que

aproximações para inteiros diferentes resultariam em um melhor desempenho da transformada.

Entretanto, em um trabalho suplementar, Ma et al. (2011b) propõem a utilização dos mesmos valores de *u* para *w*, o que reduziria o espaço de memória necessário para o armazenamento das matrizes de constantes. Os resultados apresentados demonstram que a eficiência da transformada não é afetada. Essa foi também a última proposta relacionada à ROT apresentada ao JCT-VC.

As arquiteturas apresentadas neste trabalho foram desenvolvidas segundo essa proposta mais recente, considerando as constantes *u* e *w* idênticas em todos os casos, sendo representadas pelos valores contidos na Tabela 3.1.

3.2 Exploração Algorítmica

Com base no pseudocódigo apresentado por Ma et al. (2011a) e complementado por Ma et al. (2011b), realizou-se um estudo preliminar dos algoritmos propostos. Nessa fase, os algoritmos da ROT direta e inversa foram implementados em linguagem Javascript, a fim de obter-se uma interface visual simples que permitisse analisar o comportamento das quatro diferentes alternativas da ROT, bem como verificar o correto funcionamento dos algoritmos.

Em seguida, as equações das transformadas horizontais e verticais foram isoladas dos algoritmos. Foram identificados nove estágios distintos de cálculos para cada etapa da transformada (horizontal e vertical), apresentando uma estrutura bastante uniforme.

A Tabela 3.4 apresenta as equações originais da ROT direta horizontal, extraídas dos algoritmos em Ma et al. (2011a). As constantes utilizadas nas multiplicações correspondem ao conjunto de ângulos de rotação indicado por indexROT = 1, considerando as constantes atualizadas apresentadas em Ma et al. (2011b). A entrada da ROT direta horizontal é uma linha (oito amostras) do bloco 8x8 no canto superior esquerdo da TU sendo processada. A saída é uma linha de oito amostras, com os coeficientes transformados. Nas equações, W0 a W5 representam as entradas e S0 a S5 as saídas da transformada. As entradas W6 e W7 são propagadas diretamente para a saída, sem modificações e, portanto, não aparecem nas equações.

Tabela 3.4: Equações originais da ROT Direta Horizontal, para indexROT = 1

Estágio 1	Estágio 2	Estágio 3
a0 = W0 + (-4 * W1 >>5) a3 = W3 + (-4 * W4 >>5)	b1 = W1 + (8 * a0 >>5) b4 = W4 + (8 * a3 >>5)	c0 = a0 + (-4 * b1 >>5) c3 = a3 + (-4 * b4 >>5)
Estágio 4	**Estágio 5**	**Estágio 6**
d1 = b1 + (-2 * W2 >>5) d4 = b4 + (-2 * W5 >>5)	S2 = W2 + (3 * d1 >>5) S5 = W5 + (3 * d4 >>5)	f1 = d1 + (-2 * S2 >>5) f4 = d4 + (-2 * S5 >>5)
Estágio 7	**Estágio 8**	**Estágio 9**
g0 = c0 + (-2 * f1 >>5) g3 = c3 + (-2 * f4 >>5)	S1 = f1 + (4 * g0 >>5) S4 = f4 + (4 * g3 >>5)	S0 = g0 + (-2 * S1 >>5) S3 = g3 + (-2 * S4 >>5)

A Tabela 3.5 apresenta as equações originais da ROT inversa vertical, com as constantes de indexROT = 1. A entrada da ROT inversa vertical é uma coluna (oito amostras) do bloco 8x8 no canto superior esquerdo da TU sendo processada. A saída é uma coluna de oito amostras, com os coeficientes transformados. Assim como nas demais transformadas, as entradas W6 e W7 são propagadas para a saída sem modificações.

Tabela 3.5: Equações originais da ROT Inversa Vertical, para indexROT = 1

Estágio 1	Estágio 2	Estágio 3
g0 = W0 - (5 * W1 >>5) g3 = W3 - (5 * W4 >>5)	f1 = W1 - (-10 * g0 >>5) f4 = W4 - (-10 * g3 >>5)	c0 = g0 - (5 * f1 >>5) c3 = g3 - (5 * f4 >>5)
Estágio 4	**Estágio 5**	**Estágio 6**
d1 = f1 - (-7 * W2 >>5) d4 = f4 - (-7 * W5 >>5)	S2 = W2 - (13 * d1 >>5) S5 = W5 - (13 * d4 >>5)	b1 = d1 - (-7 * S2 >>5) b4 = d4 - (-7 * S5 >>5)
Estágio 7	**Estágio 8**	**Estágio 9**
a0 = c0 - (-7 * b1 >>5) a3 = c3 - (-7 * b4 >>5)	S1 = b1 - (14 * a0 >>5) S4 = b4 - (14 * a3 >>5)	S0 = a0 - (-7 * S1 >>5) S3 = a3 - (-7 * S4 >>5)

As equações da ROT direta e inversa com indexROT 2, 3 e 4 são bastante similares às equações apresentadas nas Tabelas 3.4 e 3.5 e, por isso, foram omitidas no texto. O Apêndice A apresenta as equações para todos os índices da ROT direta e inversa.

Comparando-se as Tabelas 3.4 e 3.5 pode-se observar que as equações da ROT, tanto vertical quanto horizontal, possuem uma estrutura uniforme, sendo compostas por uma soma (na ROT direta) ou subtração (na ROT inversa), uma multiplicação e um deslocamento de 5 bits à direita. Para algumas opções de indexROT são realizadas, em determinados estágios, operações adicionais simples

que visam assegurar a estabilidade numérica dos algoritmos (MA et al., 2011a). Essas operações se resumem à troca de valores entre dois coeficientes da transformada, com a inversão de sinal de um deles.

Como todas as multiplicações presentes nas equações envolvem uma constante, essas operações podem ser substituídas por somas e deslocamentos. Com isso, obtém-se menor custo em hardware e maior desempenho em relação à implementação de um multiplicador genérico.

Nas equações onde a constante é negativa, a soma ou subtração principal pode ser invertida, eliminando a necessidade de uma operação adicional para inversão de sinal. Esta simplificação, entretanto, gera uma diferença de arredondamento no deslocamento de bits à direita, quando o operando do deslocamento não é múltiplo de 2^n (onde *n* é o número de bits deslocados). Essa diferença deve-se à característica assimétrica da codificação binária em complemento de dois. A Tabela 3.6 ilustra o problema com uma equação da ROT inversa.

Tabela 3.6: Diferença de arredondamento, entre operandos positivos e negativos, no deslocamento de bits à direita

Equação original	Equação simplificada
f1 = W1 - (-10 * g0 >>5)	f1 = W1 + (10 * g0 >>5)
para g0 = 16:	
f1 = W1 - (-10 * 16 >>5) f1 = W1 - (-160 >>5) -160_{10} >>5 = 10110000_2 >>5 = 11111011_2 = -5_{10} f1 = W1 - (-5) = W1 + 5	f1 = W1 + (10 * 16 >>5) f1 = W1 + (160 >>5) 160_{10} >>5 = 01010000_2 >>5 = 00000101_2 = 5_{10} f1 = W1 + 5 (resultado correto)
para g0 = 5:	
f1 = W1 - (-10 * 5 >>5) f1 = W1 - (-50 >>5) -50_{10} >>5 = 11001110_2 >>5 = 11111110_2 = -2_{10} f1 = W1 - (-2) = W1 + 2	f1 = W1 + (10 * 5 >>5) f1 = W1 + (50 >>5) 50_{10} >>5 = 00110010_2 >>5 = 00000001_2 = 1_{10} f1 = W1 + 1 (correção necessária: +1)

Pode-se observar, na Tabela 3.6, que a constante na equação original é negativa, então a subtração original foi substituída por uma soma na equação simplificada. É possível observar que quando o operando do deslocamento é

múltiplo de 32, todos os bits descartados no deslocamento são “0” e o resultado é equivalente nas duas equações. Entretanto, quando há truncamento, o valor negativo é uma unidade maior, em módulo, do que o positivo e, nesse caso, o resultado da equação simplificada precisa ser corrigido. Essa operação “+1” extra pode ser implementada sem custo adicional de hardware, através da entrada com um no *carry-in* do somador principal.

Considerando as otimizações expostas nos parágrafos anteriores, realizou-se a simplificação das equações visando a implementação em hardware. A Tabela 3.7 apresenta as equações simplificadas, sem multiplicações, da ROT direta horizontal, utilizando as constantes para indexROT = 1. Como a maioria das constantes, nesse caso, é uma potência de 2, quase todas as multiplicações foram resolvidas com um único deslocamento, com exceção do estágio 5 onde foram necessários somadores adicionais. As somas com constantes negativas foram substituídas por subtrações, como pode ser observado nos estágios 1, 3, 4, 6, 7 e 9. A operação “+1” presente nessas equações deve ser efetuada apenas se algum dos bits descartados no deslocamento à direita for igual a 1, para corrigir o arredondamento, conforme discutido acima.

Tabela 3.7: Equações simplificadas da ROT Direta Horizontal, para indexROT = 1

Estágio 1	**Estágio 2**	**Estágio 3**
a0 = W0 - ((W1>>3) + 1) a3 = W3 - ((W4>>3) + 1)	b1 = W1 + (a0>>2) b4 = W4 + (a3>>2)	c0 = a0 - ((b1>>3) + 1) c3 = a3 - ((b4>>3) + 1)
Estágio 4	**Estágio 5**	**Estágio 6**
d1 = b1 - ((W2>>4) + 1) d4 = b4 - ((W5>>4) + 1)	S2 = W2 + ((d1<<1)+d1 >>5) S5 = W5 + ((d4<<1)+d4 >>5)	f1 = d1 - ((S2>>4) + 1) f4 = d4 - ((S5>>4) + 1)
Estágio 7	**Estágio 8**	**Estágio 9**
g0 = c0 - ((f1>>4) + 1) g3 = c3 - ((f4>>4) + 1)	S1 = f1 + (g0>>3) S4 = f4 + (g3>>3)	S0 = g0 - ((S1>>4) + 1) S3 = g3 - ((S4>>4) + 1)

A Tabela 3.8 apresenta as equações simplificadas, sem multiplicações, da ROT inversa vertical, considerando as constantes de indexROT = 1. As subtrações com constantes negativas foram substituídas por somas, como observado nos estágios 2, 4, 6, 7 e 9. Nesses estágios, novamente, a operação “+1” é efetuada apenas se algum dos bits descartados no deslocamento à direita for igual a 1.

Tabela 3.8: Equações simplificadas da ROT Inversa Vertical, para indexROT = 1

Estágio 1	**Estágio 2**	**Estágio 3**
g0 = W0 - ((W1«2)+W1 »5) g3 = W3 - ((W4«2)+W4 »5)	f1 = W1 + ((g0«3)+(g0«1) »5) + 1 f4 = W4 + ((g3«3)+(g3«1) »5) + 1	c0 = g0 - ((f1«2)+f1 »5) c3 = g3 - ((f4«2)+f4 »5)
Estágio 4	**Estágio 5**	**Estágio 6**
d1 = f1 + ((W2«3)-W2 »5) + 1 d4 = f4 + ((W5«3)-W5 »5) + 1	S2 = W2 - ((d1«3)+(d1«2)+d1 »5) S5 = W5 - ((d4«3)+(d4«2)+d4 »5)	b1 = d1 + ((S2«3)-S2 »5) + 1 b4 = d4 + ((S5«3)-S5 »5) + 1
Estágio 7	**Estágio 8**	**Estágio 9**
a0 = c0 + ((b1«3)-b1 »5) + 1 a3 = c3 + ((b4«3)-b4 »5) + 1	S1 = b1 - ((a0«4)-(a0«1) »5) S4 = b4 - ((a3«4)-(a3«1) »5)	S0 = a0 + ((S1«3)-S1 »5) + 1 S3 = a3 + ((S4«3)-S4 »5) + 1

As equações simplificadas para os três demais indexROT possuem uma estrutura bastante similar ao que foi apresentado nas Tabelas 3.7 e 3.8 e foram omitidas no texto principal desta dissertação, mas estão apresentadas nos Apêndices B e C, para a ROT direta e inversa, respectivamente.

4 ARQUITETURAS DESENVOLVIDAS

O objetivo do desenvolvimento arquitetural para a transformada ROT foi gerar arquiteturas com desempenho suficiente para o processamento em tempo real de vídeos com resoluções superiores à *Full HD*, visando atender as demandas da próxima geração de vídeos digitais.

Com base nas equações apresentadas no Apêndice B (Tabelas B.1 e B.2), foram desenvolvidas arquiteturas combinacionais para as quatro opções da transformada direta. Essas arquiteturas foram, então, combinadas em uma única arquitetura configurável, a fim de otimizar o consumo de hardware. Por fim, foi desenvolvida uma versão dessa arquitetura configurável com um *pipeline* de nove estágios, visando maior taxa de processamento.

A mesma metodologia foi utilizada para o desenvolvimento das arquiteturas para as transformadas inversas. Todas as arquiteturas trabalham com amostras de 16 bits, tanto na entrada quanto na saída, conforme especificado na proposta da ROT (MA et al., 2011a).

Os resultados parciais deste trabalho foram publicados em (VIANNA et al., 2012a, 2012b; VIANNA; AGOSTINI, 2012). Uma descrição detalhada dessas publicações encontra-se no Apêndice D.

Este capítulo apresenta, inicialmente, a implementação do *buffer* de transposição, necessário em todas as versões das arquiteturas. Em seguida, são apresentados os detalhes específicos de cada arquitetura desenvolvida.

4.1 Implementação do *Buffer* de Transposição

Conforme apresentado no capítulo 3, o algoritmo da ROT é dividido em duas etapas, horizontal e vertical. Na etapa horizontal, a matriz de entrada é processada linha a linha, enquanto a etapa vertical processa as entradas coluna a coluna. A ordem de aplicação das etapas depende se a transformada é direta ou inversa, mas de qualquer forma, um *buffer* de transposição é necessário para armazenar a matriz

intermediária, gerada em um sentido na primeira etapa, e fornecer esses dados no sentido oposto para a segunda etapa.

A Figura 4.1 apresenta um diagrama parcial do *buffer* de transposição desenvolvido neste trabalho. Este buffer foi implementado com 64 registradores de 16 bits, dispostos como uma matriz de oito linhas por oito colunas e interligados entre si através de multiplexadores.

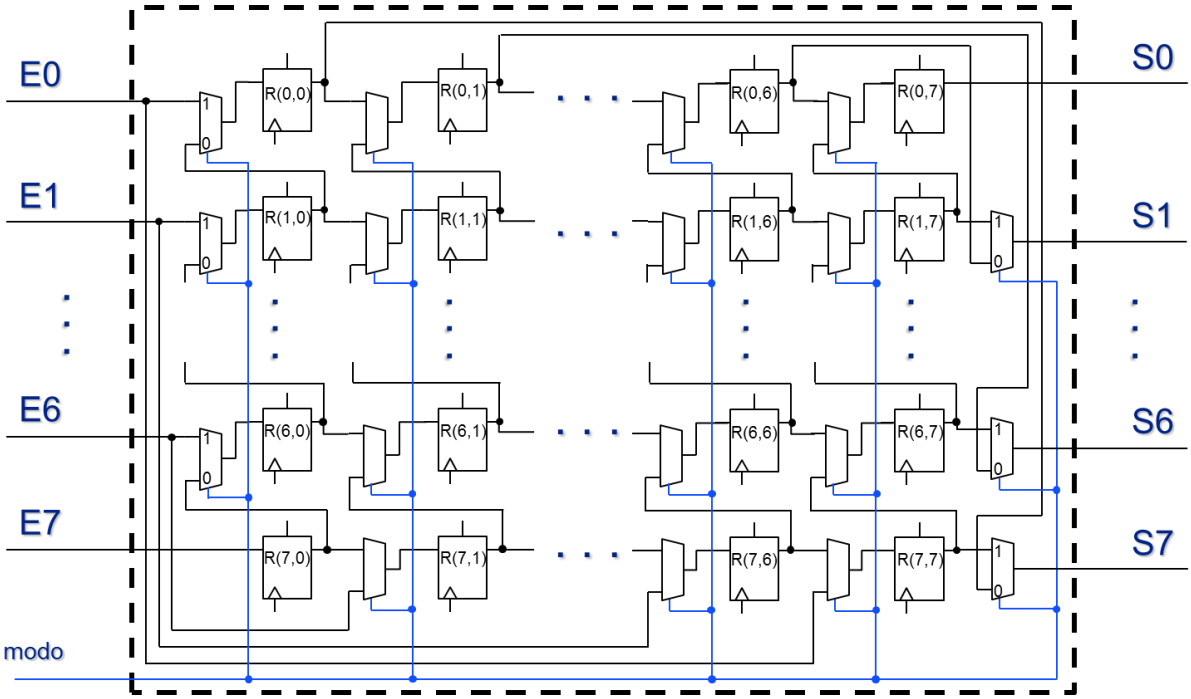

Figura 4.1: Detalhe da estrutura utilizada na implementação do *buffer* de transposição

O *buffer* recebe na entrada oito amostras de 16 bits, indicadas por E0 a E7 na Figura 4.1, o sinal de controle “modo”, de 1 bit, e os sinais de *clock* e *reset*, cujas conexões não são apresentadas na figura, para facilitar a visualização. Na saída, são fornecidas oito amostras de 16 bits, indicadas na Figura 5.1 por S0 a S7.

O sinal “modo” comanda os multiplexadores que controlam as conexões de entrada e saída, bem como o fluxo de dados dentro do *buffer*. Quando modo = 1, o *buffer* opera no modo “colunas”: as entradas são conectadas à coluna de registradores da esquerda e a cada ciclo de *clock* o conteúdo dos registradores é deslocado uma coluna para a direita. As saídas recebem os dados da coluna de registradores mais à direita. Quando modo = 0, o *buffer* opera no modo “linhas”: as entradas são conectadas à linha inferior de registradores e a cada ciclo de *clock* o

conteúdo dos registradores é deslocado uma linha acima. As saídas são realizadas através da linha superior de registradores.

Após oito ciclos de *clock*, uma matriz completa está armazenada no *buffer* e o valor do sinal “modo” é, então, invertido. A matriz que foi armazenada em linhas nos ciclos anteriores é, agora, lida em colunas e vice-versa, realizando assim a transposição necessária para a etapa seguinte. Simultaneamente, novos dados de entrada são armazenados nos registradores liberados a cada deslocamento.

4.2 Arquiteturas para as Transformadas Diretas e Inversas

Esta seção do texto irá apresentar as arquiteturas desenvolvidas para as transformadas ROT diretas e inversas em suas três versões: paralela, configurável e configurável com *pipeline*. Como as arquiteturas das transformadas diretas e inversas em cada uma das três versões possuem um modelo arquitetural bastante similar, estas arquiteturas estão descritas de forma agrupada nas próximas três subseções.

4.2.1 Arquiteturas Paralelas para a ROT Direta e Inversa

A primeira solução arquitetural desenvolvida para a ROT foi baseada em arquiteturas combinacionais independentes. Com base nas equações simplificadas, apresentadas no Apêndice B (Tabelas B.1 e B.2), foram desenvolvidas quatro arquiteturas para a ROT direta, uma para cada opção de indexROT.

Cada arquitetura está dividida em uma etapa horizontal e uma etapa vertical, com o *buffer* de transposição apresentado na seção 4.1 entre as etapas. Em cada etapa, são processadas oito amostras por ciclo de *clock*, o que equivale a uma linha da matriz de entrada ou uma coluna da matriz intermediária. Cada arquitetura executa as operações necessárias para o cálculo das equações de uma opção de indexROT.

A Figura 4.2 apresenta uma visão parcial da arquitetura combinacional desenvolvida para a ROT direta horizontal correspondente a indexROT = 1. As entradas são indicadas por W0 a W7 e as saídas por S0 a S7. Uma barreira de registradores é utilizada para sincronizar as amostras de entrada. É possível

visualizar, na Figura 4.2, os operadores utilizados para realizar os cálculos dos estágios 1 a 4 e do estágio 9, conforme organização apresentada na Tabela B.1.

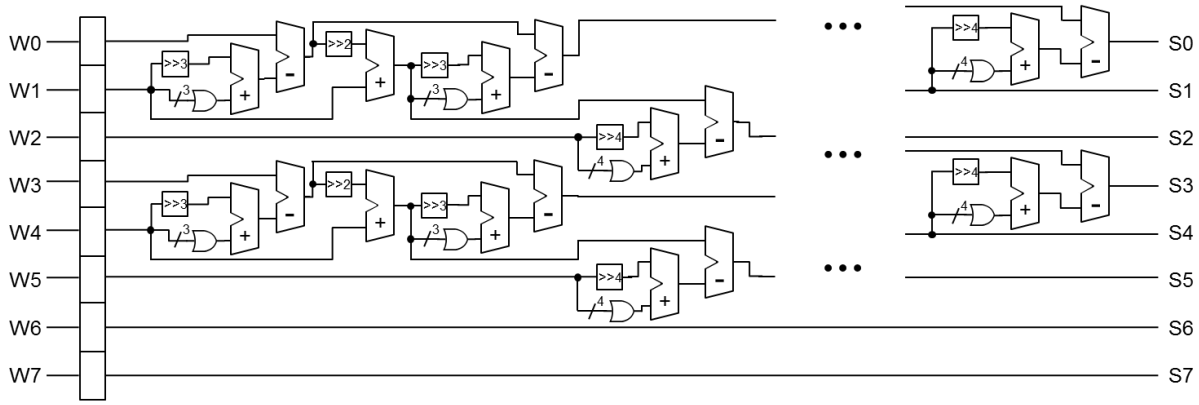

Figura 4.2: Visão parcial da arquitetura combinacional desenvolvida para a ROT direta horizontal com indexROT = 1

Como pode ser observado na Figura 4.2, as duas equações de cada estágio são calculadas em paralelo, utilizando-se dois conjuntos de operadores. Nas equações onde é necessária a correção de arredondamento, utilizam-se somadores alimentados por portas lógicas OR que testam os bits descartados nos deslocamentos à direita, conforme discutido na seção 3.2.

As arquiteturas para as demais opções de indexROT, tanto da etapa horizontal quanto vertical, são extremamente semelhantes, seguindo o mesmo modelo arquitetural apresentado na Figura 4.2.

As quatro arquiteturas da ROT direta horizontal e as quatro arquiteturas da ROT direta vertical foram então combinadas em uma única arquitetura, compartilhando o mesmo *buffer* de transposição. A Figura 4.3 apresenta um diagrama da arquitetura completa. As entradas desta arquitetura são oito amostras da matriz de entrada e o sinal indexROT. O valor de indexROT, informado apenas no início de cada TU, é armazenado em registradores dedicados (indicados na Figura 4.3 por indexH e indexV) para que o valor correto esteja disponível, de forma estável, durante os oito ciclos de processamento das etapas horizontal e vertical.

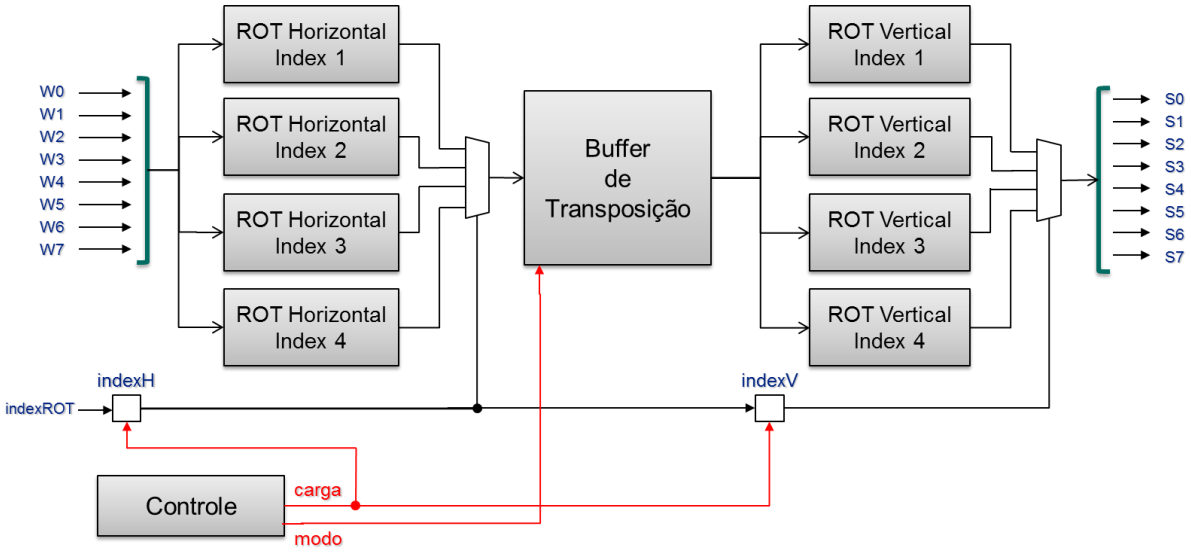

Figura 4.3: Diagrama da arquitetura paralela para a ROT direta

Pode-se observar, na Figura 4.3, que as amostras de entrada são conectadas às entradas das quatro arquiteturas horizontais, que realizam os cálculos das quatro opções de transformada em paralelo. Multiplexadores na entrada do *buffer* de transposição selecionam os resultados que devem ser armazenados, comandados pelo valor do registrador indexH. A saída do *buffer* alimenta as entradas das quatro arquiteturas verticais, e multiplexadores na saída selecionam os resultados da transformada que corresponde ao valor de indexROT, armazenado no registrador indexV.

Após oito ciclos de *clock*, uma matriz 8x8 completa é processada por cada etapa, e dados de uma nova TU começam a chegar na entrada da etapa horizontal. A matriz intermediária, que está armazenada no *buffer* de transposição e será processada pela etapa vertical, pertence à TU anterior e, portanto, o valor de indexROT daquela TU precisa ser preservado. Assim, o registrador indexV recebe o valor armazenado em indexH e indexH recebe o novo valor de indexROT da entrada. A atualização dos registradores indexH e indexV acontece a cada oito ciclos de *clock*, habilitada pelo sinal de carga emitido pela parte de controle da arquitetura.

O controle é também responsável pela geração do sinal que define o modo de operação do *buffer* de transposição, conforme apresentado na seção 4.1.

O controle da arquitetura foi implementado através de uma máquina de estados finitos com nove estados. O estado 0 representa o ciclo inicial de espera

para a sincronização das amostras de entrada e os estados 1 a 8 são os estados de operação normal, representando os oito ciclos de *clock* necessários para completar o processamento de uma matriz 8x8 por cada etapa. No estado 8 é realizada a atualização dos registradores indexH e indexV, o valor do sinal “modo” é invertido e a máquina de estados retorna ao estado 1.

A arquitetura paralela da ROT apresenta uma latência de nove ciclos de *clock*, sendo oito ciclos utilizados pela etapa horizontal para processar as oito linhas da matriz de entrada, e um ciclo adicional para o processamento da primeira coluna pela etapa vertical da arquitetura. O caminho crítico da arquitetura encontra-se na etapa vertical, contendo 24 somadores em série.

O mesmo modelo arquitetural foi utilizado para o desenvolvimento das arquiteturas para a ROT inversa, resultando em uma solução bastante semelhante. A Figura 4.4 apresenta um diagrama geral da arquitetura paralela desenvolvida para a ROT inversa. Pode-se observar, na Figura 4.4, que a principal diferença encontra-se na ordem de aplicação das etapas da transformada. Na ROT inversa, a transformada vertical é calculada primeiro e a horizontal em seguida.

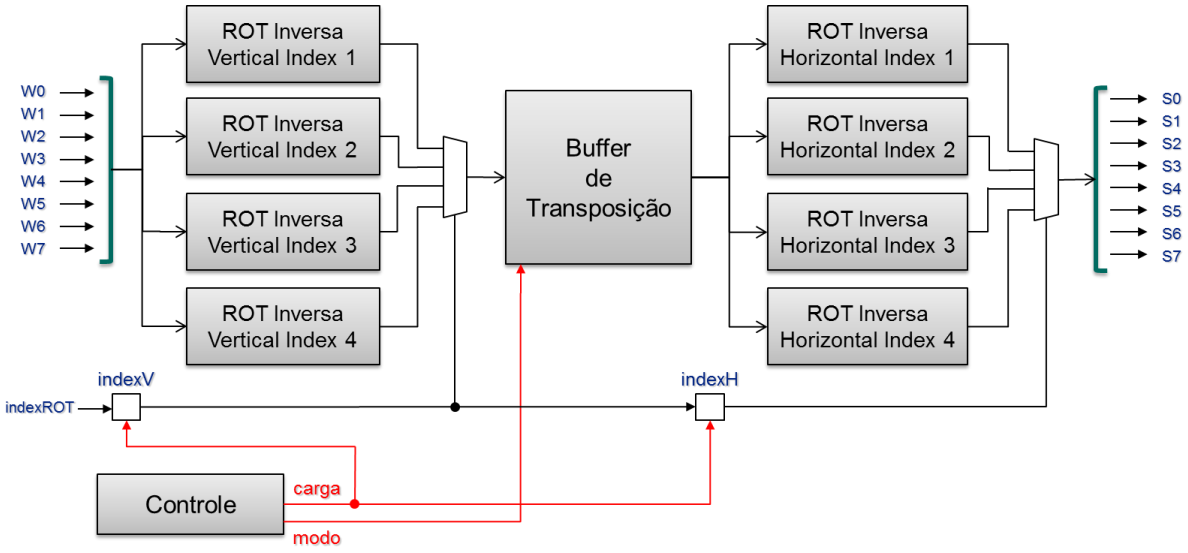

Figura 4.4: Diagrama da arquitetura paralela para a ROT inversa

Outra diferença reside na operação principal das equações que, no caso da ROT inversa, é uma subtração, conforme pode ser observado nas equações apresentadas no Apêndice A (Tabelas A.3 e A.4). Nas equações utilizadas para a implementação em hardware, apresentadas no Apêndice C (Tabelas C.1 e C.2),

quando a constante multiplicadora é negativa, a subtração principal é substituída por uma soma e o sinal da constante é invertido, podendo ocasionar novamente uma diferença de arredondamento no deslocamento à direita. Nesse caso, entretanto, a correção pode ser implementada sem complexidade adicional, utilizando-se um somador completo para realizar a soma principal e alimentando-se com “1” a entrada de *carry-in* do somador. Com essa implementação, o caminho crítico da arquitetura é reduzido de 24 para 19 somadores em série. As demais características são idênticas às da arquitetura da ROT direta, com latência de nove ciclos de *clock* e a mesma máquina de estados na parte de controle.

4.2.2 Arquiteturas Configuráveis para a ROT Direta e Inversa

A segunda arquitetura desenvolvida foi uma alternativa configurável para a arquitetura paralela, buscando-se uma solução mais otimizada em termos de consumo de hardware.

Nesta nova arquitetura, as quatro opções da transformada foram combinadas em apenas um bloco horizontal e um bloco vertical. Dentro de cada bloco, multiplexadores selecionam os operandos de cada somador/subtrator, de acordo com a opção de indexROT, otimizando o número de operadores utilizados.

A Figura 4.5 apresenta um diagrama da arquitetura configurável para a ROT direta. Nesta arquitetura, a etapa horizontal recebe, além das amostras de entrada, o valor de indexROT, através do registrador indexH. Esse valor será utilizado para comandar os multiplexadores internos e executar as operações correspondentes à opção de transformada desejada.

As saídas da etapa horizontal seguem diretamente para o *buffer* de transposição. A etapa vertical da arquitetura recebe como entradas as amostras lidas do *buffer* e o valor de indexROT, através do registrador indexV. As saídas da etapa vertical já são as saídas da própria arquitetura.

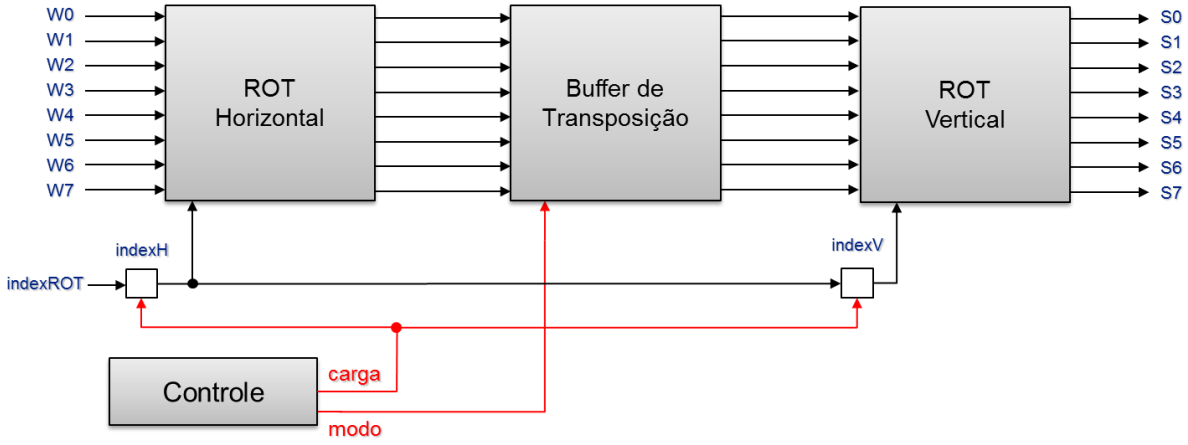

Figura 4.5: Diagrama da arquitetura configurável para a ROT direta

A Figura 4.6 apresenta uma visão parcial da etapa horizontal da arquitetura configurável combinacional para a ROT direta. É possível visualizar, na Figura 4.6, os dois primeiros estágios de cálculo, cada um capaz de resolver quatro equações, conforme o valor de indexROT.

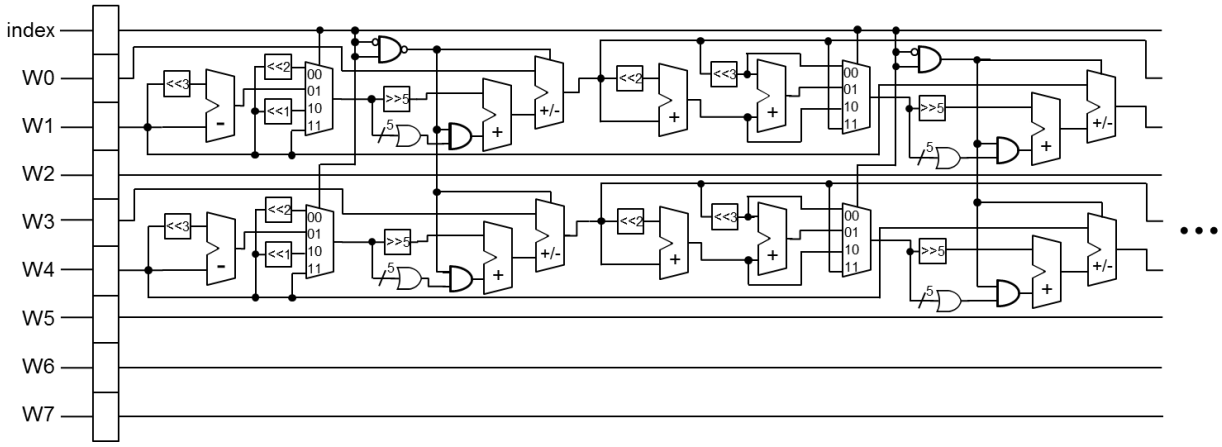

Figura 4.6: Detalhe da arquitetura configurável combinacional desenvolvida para a ROT direta horizontal

Pode-se observar, na Figura 4.6, a adição de multiplexadores 4:1 que selecionam uma das entradas do somador/subtrator principal de cada estágio, comandados pelo valor de indexROT. Portas lógicas AND e NAND são utilizadas para testar os bits individuais de indexROT e controlar o modo de operação dos componentes somadores/subtratores. A saída dessas portas também habilita a correção de arredondamento, através de outra porta AND na entrada do somador.

Para a correção de arredondamento, são avaliados os bits descartados no deslocamento à direita em cada estágio de cálculo. Em função das otimizações realizadas nas equações simplificadas, em um mesmo estágio encontram-se equações com diferentes deslocamentos de bits à direita. No caso das arquiteturas configuráveis, isso traria uma complexidade adicional, pois seria necessário testar diferentes conjuntos de bits. Para simplificar o processo de teste, as equações de cada estágio foram reescritas com um deslocamento fixo de cinco bits à direita. Desse modo, o deslocamento é realizado uma única vez, na saída de cada multiplexador, e são testados sempre cinco bits para verificar a necessidade ou não do arredondamento, conforme pode ser observado na Figura 4.6.

A Tabela 4.1 apresenta as equações simplificadas do primeiro estágio da ROT direta horizontal e as equações reescritas com o deslocamento fixo de 5 bits à direita, onde é possível observar a estrutura regular obtida.

Tabela 4.1: Equações do primeiro estágio da ROT direta horizontal reescritas com deslocamento fixo de cinco bits à direita

indexROT	Equações simplificadas	Equações com deslocamento fixo
1	a0 = W0 - ((W1>>3) + 1)	a0 = W0 - ((W1<<2 >>5) + 1)
2	a0 = W0 - (((W1<<3)-W1 >>5) + 1)	a0 = W0 - (((W1<<3)-W1 >>5) + 1)
3	a0 = W0 + (W1>>4)	a0 = W0 + (W1<<1 >>5)
4	a0 = W0 - ((W1>>5) + 1)	a0 = W0 - ((W1 >>5) + 1)

O caminho crítico desta arquitetura é de 29 somadores e nove multiplexadores em série. O aumento no número de somadores totais, em relação à arquitetura paralela, deve-se ao fato de que os estágios de cálculo apresentam complexidades diferentes de acordo com o indexROT selecionado. Na arquitetura paralela, cada indexROT é processado de forma independente, enquanto que na arquitetura configurável sempre será processado o pior caso de cada estágio.

A parte de controle não necessitou alterações em relação à arquitetura apresentada na seção anterior, sendo utilizada exatamente a mesma máquina de estados.

Para o desenvolvimento da arquitetura configurável para a ROT inversa foi utilizada a mesma metodologia, agrupando-se as quatro opções da transformada em blocos vertical e horizontal. As entradas da arquitetura são processadas,

primeiramente, pela etapa vertical, sendo os resultados armazenados no *buffer* de transposição e, então, processados pela etapa horizontal.

Dentro de cada bloco, utilizou-se uma estrutura semelhante à apresentada na Figura 4.6, implementando-se as equações apresentadas no Apêndice C (Tabelas C.1 e C.2). Foi empregada, também, a técnica de reescrita das equações apresentada na Tabela 4.1, para uniformização do deslocamento à direita e otimização dos componentes utilizados para avaliar a necessidade de aplicação do arredondamento.

O caminho crítico da arquitetura para a ROT inversa é equivalente ao da arquitetura para a ROT direta, com 29 somadores e nove multiplexadores em série.

4.2.3 Arquiteturas Configuráveis com Pipeline para a ROT Direta e Inversa

A terceira arquitetura desenvolvida teve como objetivo atingir a taxa de processamento necessária para processar, em tempo real, vídeos na resolução 8K UHD, a mais alta resolução definida até o momento para vídeos digitais.

Utilizando a organização apresentada no Apêndice B (Tabelas B.1 e B.2) foi possível implementar uma arquitetura com um *pipeline* de nove estágios para as transformadas horizontal e vertical. Cada estágio do *pipeline* explora o paralelismo processando duas equações simultaneamente.

Como base desta arquitetura foi utilizada a arquitetura configurável combinacional, porém os estágios de operações foram separados por barreiras de registradores. Assim como nas arquiteturas anteriores, oito amostras são processadas a cada ciclo de *clock*, porém, devido à forte dependência de dados entre as operações, apenas dois novos resultados são gerados a cada estágio, como pode ser observado nas equações apresentadas no Apêndice B (Tabelas B.1 e B.2).

A Figura 4.7 apresenta uma visão parcial da arquitetura configurável com *pipeline* desenvolvida para a ROT direta horizontal. Pode-se observar, na Figura 4.7, as barreiras temporais que utilizam oito registradores de 16 bits para propagar as amostras de entrada e os resultados intermediários, e um registrador de dois bits para armazenar o valor de indexROT. O valor correto de indexROT precisa acompanhar cada conjunto de amostras dentro do *pipeline*, pois amostras de TUs distintas estarão sendo processadas por diferentes estágios a cada ciclo de *clock*.

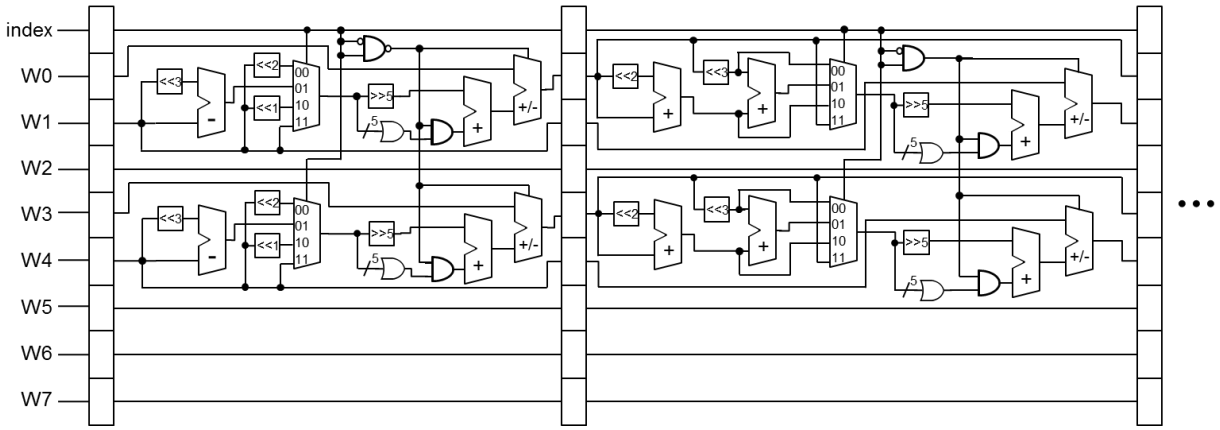

Figura 4.7: Detalhe da arquitetura configurável com *pipeline* desenvolvida para a ROT direta horizontal

Um diagrama geral da arquitetura configurável com *pipeline* é apresentado na Figura 4.8.

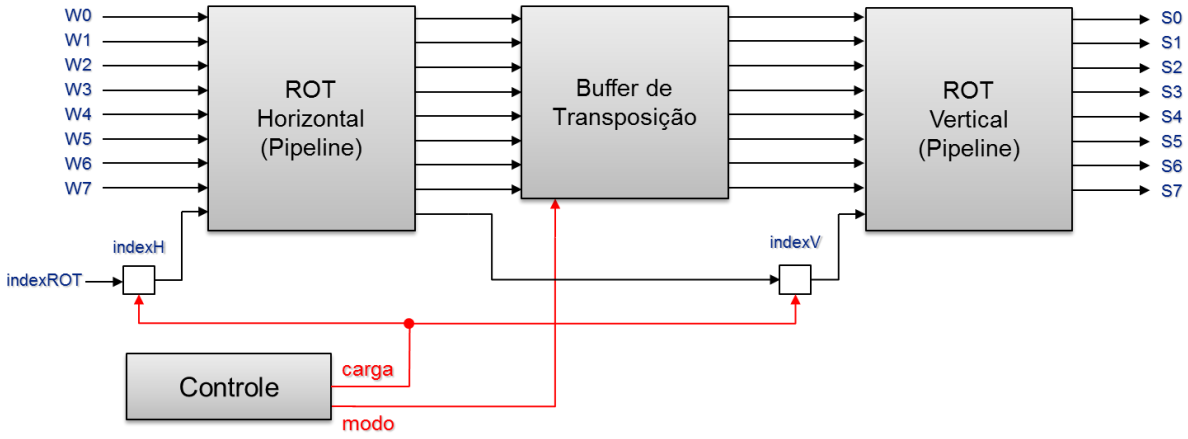

Figura 4.8: Diagrama da arquitetura configurável com *pipeline* para a ROT direta

O valor de indexROT é propagado junto com as amostras de entrada através de ambos *pipelines.* A etapa horizontal tem uma saída adicional, em relação à arquitetura combinacional, que é o valor de indexROT associado às amostras que estão sendo entregues na saída. Essa saída alimenta o registrador indexV.

A parte de controle utiliza uma máquina de estados finitos com 17 estados, muito semelhante à apresentada na seção 4.2.1, com os estados 0 a 8 representando os ciclos de latência até que os primeiros resultados válidos sejam

gerados pela etapa horizontal, e os estados 9 a 16 representando o laço de operação normal após o preenchimento do *pipeline*.

O caminho crítico desta arquitetura é de quatro somadores e um multiplexador em série, o que corresponde ao estágio de maior complexidade. A latência da arquitetura é de 25 ciclos de *clock*, sendo oito ciclos para preencher o *pipeline* da etapa horizontal, oito ciclos para a etapa horizontal processar todas as linhas da primeira matriz, e mais nove ciclos até que os primeiros resultados válidos sejam entregues na saída da etapa vertical.

A arquitetura para a transformada inversa também toma como base a versão configurável combinacional e segue a organização das equações apresentadas no Apêndice C (Tabelas C.1 e C.2). Barreiras de registradores são introduzidas entre cada estágio de cálculo, criando um *pipeline* de nove estágios para cada etapa da transformada.

As demais características da arquitetura para a ROT inversa, como latência, caminho crítico e parte de controle são idênticas às da arquitetura para a ROT direta, devido à simetria das transformadas.

5 SÍNTESE, VALIDAÇÃO E RESULTADOS

Este capítulo descreve, inicialmente, a metodologia utilizada na síntese das arquiteturas desenvolvidas. Em seguida, apresenta-se a metodologia empregada na validação dessas arquiteturas. Por fim, são apresentados e discutidos os resultados obtidos, em termos de utilização de hardware e taxa de processamento das arquiteturas implementadas.

5.1 Metodologia de Síntese

As arquiteturas apresentadas no capítulo 5 foram descritas em VHDL e sintetizadas para o FPGA EP3SL50F484C2, da família Stratix III da Altera, utilizando o software Quartus II (ALTERA, 2012), versão 9.1sp2 Web Edition.

Com o intuito de acelerar o processo de descrição das diversas arquiteturas propostas, procurou-se criar códigos VHDL bastante modularizados, potencializando a reutilização de código.

Um dos primeiros componentes elementares a ser descrito foi um registrador genérico de “n” bits. Esse componente foi instanciado na descrição do *buffer* de transposição e, posteriormente, na descrição das barreiras de registradores das arquiteturas com *pipeline*.

Para as arquiteturas paralelas, apresentadas na seção 4.2.1, a arquitetura de cada indexROT foi descrita em dois componentes independentes, um para a etapa horizontal e um para a vertical. No arquivo VHDL principal da arquitetura foram instanciados os quatro componentes horizontais, os quatro componentes verticais, o *buffer* de transposição e uma barreira de registradores para sincronizar as entradas. Também foram descritos, no arquivo principal, os multiplexadores que selecionam as saídas de cada etapa da arquitetura e a máquina de estados responsável pelos sinais de controle.

A parte operativa foi descrita dentro dos componentes horizontais e verticais de cada indexROT. Foram criados componentes individuais para cada constante única presente nas equações da transformada. Cada componente possui duas

entradas, que correspondem às variáveis da equação, e uma saída. Como a cada estágio são calculadas duas equações idênticas, alterando apenas as entradas, cada componente foi instanciado duas vezes por estágio, bastando mapear corretamente suas entradas e saídas. Considerando ainda que algumas constantes são utilizadas em mais de um estágio, essa modularização representou um ganho de tempo significativo na descrição das arquiteturas, evitando a reescrita de código redundante.

Esta metodologia também simplificou substancialmente a depuração dos códigos VHDL. Cada componente era descrito e imediatamente testado, o que praticamente eliminou a necessidade de correções nas arquiteturas finalizadas.

Uma vez que a tecnologia alvo para as arquiteturas são FPGAs, as equações simplificadas foram descritas de forma comportamental, deixando maiores otimizações por conta da ferramenta de síntese.

A Figura 5.1 apresenta um exemplo de arquitetura sintetizada, para o componente responsável pelo cálculo das equações com a constante -29, utilizada nos estágios 4 e 6 das arquiteturas horizontais diretas e inversas, para indexROT 3 e 4. As entradas são representadas por *e0* e *e1*, e a saída por *s0*. É possível observar, na Figura 6.1, que foram utilizados quatro somadores e uma porta “OU” de cinco entradas para esse componente.

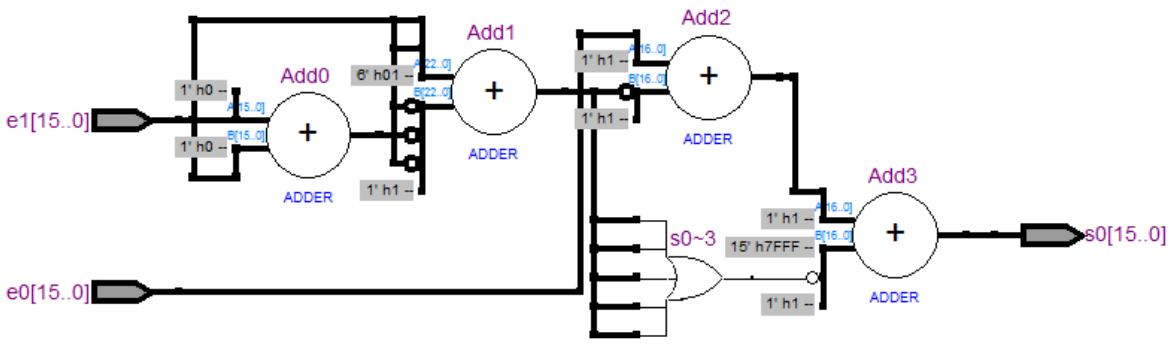

Figura 5.1: Arquitetura sintetizada para o componente que calcula as equações com a constante -29

Para as arquiteturas configuráveis, as quatro opções de indexROT foram englobadas em apenas um componente horizontal e um vertical. Para cada estágio de cálculo, conforme os Apêndices B e C (Tabelas B.1, B.2, C.1 e C.2), foram criados componentes individuais. Cada um desses componentes contempla quatro

equações, que correspondem às opções de indexROT para aquele determinado estágio. A cada ciclo de *clock*, uma das equações é calculada pelo componente que recebe como entradas, além das duas variáveis, o valor de indexROT. Dentro do componente, multiplexadores comandados pelo indexROT selecionam as operações corretas a serem realizadas.

Cada componente foi instanciado duas vezes por estágio, para resolver as duas equações em paralelo, assim como descrito anteriormente. Além disso, devido ao padrão regular de aplicação das constantes nas equações, os componentes dos estágios 1, 4 e 7 puderam ser reutilizados nos estágios 3, 6 e 9, respectivamente, tanto na etapa horizontal como na vertical.

A Figura 5.2 apresenta a arquitetura sintetizada para o componente do primeiro estágio da ROT direta horizontal configurável. É possível observar, na Figura 5.2, que a solução gerada pela ferramenta de síntese apresenta algumas diferenças em relação à arquitetura proposta, apresentada na Figura 4.5. Em especial, pode-se perceber a utilização de dois somadores em paralelo no lugar do somador/subtrator genérico proposto, e a aplicação da correção de arredondamento que é controlada por um multiplexador adicional na saída, ao invés da porta AND. Conforme já comentado, os componentes foram descritos de forma comportamental para agilizar o desenvolvimento das arquiteturas.

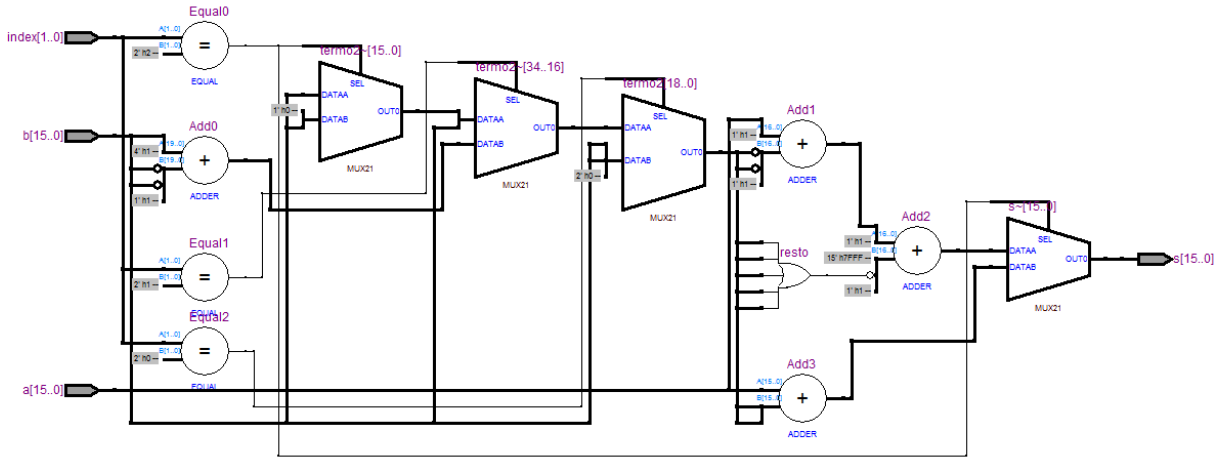

Figura 5.2: Arquitetura sintetizada para o componente do primeiro estágio da ROT direta horizontal configurável

Para as arquiteturas configuráveis com *pipeline* foi definido um novo componente, composto de um registrador de 2 bits, para o sinal indexROT, e oito

registradores de 16 bits, para as amostras. Cada estágio de cálculo foi conectado ao estágio seguinte através de uma instância deste componente, criando-se, assim, as barreiras temporais do *pipeline*.

Devido à modularização utilizada, os componentes dos estágios de cálculo puderam ser aproveitados diretamente das arquiteturas configuráveis combinacionais, sem modificações. A parte de controle foi alterada, para incluir na máquina de estados os ciclos de latência do *pipeline*.

5.2 Metodologia de Validação

Para a validação das arquiteturas foram utilizados dados extraídos do software de referência do HEVC. A versão do software utilizada foi a HM5.0-ROT, uma modificação da versão 5.0 oficial, com o código da ROT adicionado. Os códigos-fonte foram gentilmente cedidos pelos autores da ROT (Zhan Ma, Felix Fernandes, Elena Alshina e Alexander Alshin), após contato por e-mail, uma vez que essa versão não está disponibilizada publicamente no repositório do HM. Os autores da proposta da ROT são pesquisadores da empresa Samsung Electronics e mostraram-se bastante interessados na execução deste projeto.

Primeiramente, foram identificados, no código-fonte, os métodos responsáveis pela ROT direta e inversa. A seguir, foram adicionadas funções para capturar, em formato texto, os dados de entrada e saída das transformadas. As modificações foram realizadas nos métodos TComTrQuant::RotTransformLI2 e TComTrQuant::InvRotTransformLI2, no arquivo fonte TComTrQuant.cpp.

O código-fonte com as alterações foi compilado e a sequência de vídeo “riverbed” (XIPH, 2012) foi processada pelo módulo codificador, na configuração *Intra Only*, capturando-se em um arquivo texto os dados referentes aos blocos processados pela ROT direta. O vídeo codificado foi, então, processado pelo módulo decodificador, para captura dos dados referentes à ROT inversa. Foram capturados, em arquivos texto, 4.000 blocos processados pelas transformadas diretas e 4.000 blocos processados pelas transformadas inversas.

Em seguida, os arquivos texto foram processados por um script PHP, para organizar os dados para utilização nos *testbenches*. Por fim, quatro arquivos texto foram gerados: entrada da ROT direta, saída da ROT direta, entrada da ROT inversa e saída da ROT inversa.

Os *testbenches* das arquiteturas foram descritos em VHDL, lendo as amostras dos arquivos texto de entrada e direcionando as saídas das arquiteturas para novos arquivos texto, no mesmo formato dos arquivos de saída gerados pelo script. Para rodar as simulações foi utilizado o software ModelSim-Altera (ALTERA, 2012), versão Starter Edition 6.5b.

Os arquivos gerados pelos *testbenches* foram comparados com os arquivos de saída do software de referência, utilizando o comando *fc* da linha de comando do Windows. Nenhuma diferença foi encontrada, indicando que o comportamento das arquiteturas está de acordo com a ROT implementada no software de referência. Assim, as arquiteturas foram consideradas validadas.

5.3 Descrição dos Resultados Atingidos

No total, seis arquiteturas foram sintetizadas e avaliadas, sendo três para a ROT direta e três para a ROT inversa, conforme apresentado na seção 4.2. Esta seção apresenta, primeiramente, os resultados obtidos pelas arquiteturas desenvolvidas para a ROT direta, em termos de consumo de hardware e taxa de processamento. Em seguida, são apresentados os resultados para as arquiteturas da ROT inversa.

Para a ROT direta, foram avaliadas as seguintes arquiteturas: (i) arquitetura paralela (descrita na seção 4.2.1); (ii) arquitetura configurável (descrita na seção 4.2.2) e (iii) arquitetura configurável com *pipeline* de nove estágios (descrita na seção 4.2.3).

A Tabela 5.1 apresenta os resultados de síntese das arquiteturas da ROT direta, considerando o consumo de hardware em termos de utilização de ALUTs e registradores dedicados alocados pelo FPGA alvo (Stratix III EP3SL50F484C2 da Altera).

Pode-se observar na Tabela 5.1 que a arquitetura paralela foi a que utilizou o maior número de ALUTs. A arquitetura configurável obteve uma redução de aproximadamente 31% no número de ALUTs, em relação à arquitetura paralela, graças ao reuso dos operadores em cada estágio de cálculo. A arquitetura configurável com *pipeline*, por outro lado, apresentou um aumento de quase 180% no número de registradores dedicados, por conta das barreiras temporais entre os estágios do *pipeline*. No entanto, essa maior utilização de registradores foi de certa

forma compensada por uma redução no número de ALUTs. Isso pode ser atribuído a uma melhor otimização obtida pela ferramenta de síntese na alocação de células lógicas.

Tabela 5.1: Resultados comparativos do consumo de hardware entre as arquiteturas desenvolvidas para a ROT Direta

Arquitetura	ALUTs	Registradores Dedicados
Paralela	5.308	1.166
Configurável	3.647	1.166
Configurável com *pipeline*	3.057	3.258

Dispositivo EP3SL50F484C2

A seguir, são comparados os desempenhos das diferentes alternativas arquiteturais. A Tabela 5.2 apresenta a frequência máxima de operação obtida, a respectiva taxa de processamento em milhões de amostras por segundo e a latência de cada arquitetura.

Tabela 5.2: Resultados comparativos de desempenho entre as arquiteturas desenvolvidas para a ROT Direta

Arquitetura	Frequência (MHz)	Taxa de Processamento (MAmostras/s)	Latência (ciclos)
Paralela	44,58	356,64	9
Configurável	33,17	265,36	9
Configurável com *pipeline*	215,01	1.720,08	25

Dispositivo EP3SL50F484C2

Como pode ser observado na Tabela 5.2, a arquitetura configurável apresentou uma redução de desempenho de aproximadamente 25% em relação à arquitetura paralela, em função da complexidade adicional para resolver as quatro equações em cada estágio de cálculo. A arquitetura paralela e a arquitetura configurável sem *pipeline* apresentam, ambas, uma latência de 9 ciclos de *clock*, enquanto a arquitetura com *pipeline* possui uma latência de 25 ciclos. Com uma

frequência de operação de 215,01 MHz e processando oito amostras por ciclo, a arquitetura configurável com *pipeline* atingiu um desempenho suficiente para processar 1,7 bilhão de amostras por segundo, mais de seis vezes a taxa de processamento obtida pela arquitetura configurável sem *pipeline*.

Para a ROT inversa as mesmas arquiteturas foram avaliadas: (i) arquitetura paralela; (ii) arquitetura configurável e (iii) arquitetura configurável com *pipeline* de nove estágios.

A Tabela 5.3 apresenta os resultados de síntese das arquiteturas da ROT inversa, considerando o consumo de hardware em termos de utilização de ALUTs e registradores dedicados alocados pelo FPGA alvo.

Tabela 5.3: Resultados comparativos do consumo de hardware entre as arquiteturas desenvolvidas para a ROT Inversa

Arquitetura	**ALUTs**	**Registradores Dedicados**
Paralela	4.689	1.166
Configurável	3.568	1.166
Configurável com *pipeline*	3.057	3.255

Dispositivo EP3SL50F484C2

Conforme os resultados apresentados na Tabela 5.3, observa-se que a arquitetura configurável obteve a menor utilização de lógica, com uma redução de quase 24% no número de ALUTs em relação à arquitetura paralela. A arquitetura configurável com *pipeline* foi a que menos utilizou ALUTs, mas utilizou um maior número de registradores, conforme esperado, com resultados praticamente idênticos aos da arquitetura equivalente para a ROT direta.

Os resultados de desempenho das arquiteturas desenvolvidas para a ROT inversa são apresentados na Tabela 5.4.

Nos resultados apresentados na Tabela 5.4, pode-se observar que a arquitetura paralela obteve uma taxa de processamento quase 20% superior à da arquitetura configurável. A maior frequência de operação e respectiva taxa de processamento foi obtida pela arquitetura configurável com *pipeline*, superando 1,5 bilhão de amostras processadas por segundo.

Tabela 5.4: Resultados comparativos de desempenho entre as arquiteturas desenvolvidas para a ROT Inversa.

Arquitetura	Frequência (MHz)	Taxa de Processamento (MAmostras/s)	Latência (ciclos)
Paralela	46,79	374,32	9
Configurável	39,14	313,12	9
Configurável com *pipeline*	195,54	1.564,32	25

Dispositivo EP3SL50F484C2

A seguir apresenta-se uma análise do desempenho necessário para o processamento de vídeos em tempo real, e estimativas de quadros por segundo que as arquiteturas desenvolvidas são capazes de atingir.

Tomando-se como base a decodificação de vídeo *Full HD*, com resolução de 1920x1080 a 30 quadros por segundo e na configuração *Intra Only* (todos os quadros do tipo I), a arquitetura da ROT deverá ser capaz de processar, no pior caso, 62,2 milhões de amostras de luminância por segundo, caso todas as TUs sejam de tamanho 8x8. Para TUs de 16x16 ou 32x32, a ROT opera apenas sobre um bloco de 8x8 amostras, no canto superior esquerdo da TU, fazendo com que só parte das amostras do quadro necessite ser processada pela arquitetura. Ainda na configuração *Intra Only*, mas com todas as TUs de tamanho 32x32, a ROT necessitaria processar somente 1/16 do total de amostras, ou cerca de 3,8 milhões de amostras por segundo para a resolução *Full HD*. Este seria o melhor caso para a ROT considerando a configuração *Intra Only*.

Na configuração *Random Access*, mais comumente utilizada, apenas um a cada aproximadamente 30 quadros é do tipo I. Além disso, a codificação intra-quadro pode também ser usada na compressão dos blocos dos demais tipos de quadros, mas ela é pouco utilizada neste cenário. Como a ROT é aplicada apenas sobre resíduos da predição intra-quadro, na configuração *Random Access* o número de amostras que necessitam ser efetivamente processadas pela ROT diminui expressivamente.

Seguindo esse raciocínio, a Tabela 5.5 apresenta a estimativa de quadros por segundo que as arquiteturas desenvolvidas para a ROT inversa, utilizadas no decodificador, são capazes de processar, para diferentes resoluções de vídeo. Os

resultados apresentados consideram sempre o pior caso, com todos os quadros do vídeo utilizando apenas predição intra-quadro e todas as TUs com tamanho 8x8. As resoluções consideradas são: *Full HD* ou 1080p (1920x1080 pixels), QHD (2560x1440 pixels), 4K UHD (3840x2160 pixels) e 8K UHD (7680x4320 pixels). Também é apresentada na Tabela 5.5 a frequência mínima necessária para as arquiteturas processarem 30 e 60 quadros por segundo em cada uma das resoluções avaliadas. A frequência mínima para atingir uma determinada taxa de quadros por segundo é função somente do número de amostras processadas por ciclo. Por isso, para uma dada resolução, a frequência mínima para atingir uma determinada taxa de quadros por segundo é comum para todas as arquiteturas desenvolvidas, já que todas elas são capazes de processar oito amostras por ciclo.

Tabela 5.5: Estimativa de quadros por segundo processados pelas arquiteturas da ROT Inversa

Resolução	Arquitetura Paralela (qps)	Arquitetura Configurável (qps)	Arquitetura Configurável com Pipeline (qps)	Frequência para 30qps (MHz)	Frequência para 60qps (MHz)
1080p	180,5	151,0	754,3	7,78	15,56
QHD	101,5	84,9	424,3	13,83	27,66
4K UHD	45,1	37,7	188,6	31,10	62,20
8K UHD	11,2	9,4	47,1	124,42	-

Nos resultados apresentados na Tabela 5.5 é possível constatar que a arquitetura com *pipeline* é capaz de superar as taxas de processamento necessárias para atingir 30 quadros por segundo, mesmo quando processando vídeos de altíssima resolução, no formato 8K UHD. A arquitetura com *pipeline* também é capaz de processar mais de 120 quadros por segundo para resolução 4K UHD e isso mostra que esta solução é capaz de atingir requisitos extremos de taxa de quadros para esta resolução, já que algumas aplicações exigem tais taxas. As arquiteturas sem *pipeline*, tanto a paralela quanto a configurável, também obtiveram um desempenho bastante elevado, superando os 30 quadros por segundo até a resolução 4K UHD.

Caso a resolução-alvo seja a *Full HD* (1080p), todas as arquiteturas são capazes de processar vídeos em tempo real, com uma baixa frequência de

operação, de apenas 7,78 MHz, conforme apresentado na Tabela 5.5. Essa é uma característica desejável para dispositivos móveis e portáteis, já que a operação em baixa frequência permite reduzir o consumo de energia e potência.

No lado do codificador, estimar a taxa de processamento necessária para se atingir um determinado número de quadros por segundo é uma tarefa bem mais complexa. Isso porque, a cada etapa da codificação, o codificador precisa tomar uma série de decisões que afetam as etapas seguintes, criando uma árvore de decisão com um elevado número de possibilidades. Algumas dessas decisões incluem: qual a melhor estratégia de particionamento para cada CU; qual o melhor modo de predição a ser utilizado em cada PU; na predição intra-quadro qual dos 35 modos direcionais utilizar; qual opção da ROT utilizar para cada TU, e assim por diante.

Em uma abordagem do tipo “força bruta” o codificador precisaria testar todas as combinações possíveis, do particionamento do bloco até a codificação de entropia, para optar por aquela combinação que resulte na melhor taxa-distorção ao final do processo. Técnicas de controle de complexidade, geralmente baseadas em heurísticas, podem auxiliar na tomada de decisões do codificador, visando limitar o número de operações necessárias e atingir a taxa de processamento desejada.

Com base nos resultados apresentados na Tabela 5.2 é possível fazer uma avaliação semelhante à que foi feita para a ROT inversa, quando considerada isoladamente. A Tabela 5.6 apresenta estimativas de quadros por segundo para as diferentes arquiteturas desenvolvidas para a ROT Direta. Considerou-se, novamente, um cenário de pior caso, com todos os quadros utilizando somente a predição intra-quadro, e todas as TUs de tamanho 8x8.

Pode-se observar, nos resultados apresentados na Tabela 5.6, que todas as versões da arquitetura são capazes de superar os 30 quadros por segundo até a resolução 4K UHD. A arquitetura configurável com *pipeline* supera os 30 quadros por segundo até a resolução 8K UHD e também atinge 120 quadros por segundo na resolução 4K UHD. Estas estimativas contemplam o cálculo de uma das opções da ROT por TU, considerando a utilização de alguma heurística para decidir o indexROT a ser aplicado.

Tabela 5.6: Estimativa de quadros por segundo processados pelas arquiteturas da ROT Direta

Resolução	Arquitetura Paralela (qps)	Arquitetura Configurável (qps)	Arquitetura Configurável com Pipeline (qps)
1080p	171,9	127,9	829,5
QHD	96,7	71,9	466,6
4K UHD	42,9	31,9	207,3
8K UHD	10,7	7,9	51,8

Finalmente, cabe ressaltar que o FPGA utilizado na síntese das arquiteturas não é um dispositivo de última geração. Dispositivos mais atuais são capazes de atingir frequências de operação mais elevadas, proporcionando taxas de processamento ainda maiores para as arquiteturas, o que pode ser vantajoso especialmente para o codificador.

Até onde se tem conhecimento, este é o primeiro trabalho publicado na literatura sobre a ROT implementada em hardware. Assim, não foi possível comparar a solução apresentada com outros trabalhos.

6 CONCLUSÕES

Esta dissertação apresentou soluções arquiteturais para o cálculo da Transformada Rotacional (ROT), uma nova ferramenta proposta para o padrão emergente de codificação de vídeo HEVC. A ROT é uma transformada secundária utilizada para compactar melhor a energia das matrizes de coeficientes resultantes da DCT, aumentando a taxa de compressão. A ROT é aplicada apenas sobre amostras de luminância de TUs de tamanho 8x8, 16x16 e 32x32, porém o tamanho da transformada é sempre 8x8. A proposta da ROT define quatro conjuntos de ângulos de rotação, que fornecem ao codificador quatro opções distintas de transformada, para melhor adequação a diferentes tipos de resíduos.

O texto apresentou uma introdução ao HEVC, a fim de contextualizar a contribuição deste trabalho e permitir sua integral compreensão. Nessa introdução, foram descritos os novos elementos estruturais do futuro padrão e as principais ferramentas de codificação incorporadas até o momento da escrita deste trabalho.

Em seguida, foi apresentada a fundamentação teórica da Transformada Rotacional, onde foram discutidos detalhadamente os objetivos e conceitos da ROT, bem como a evolução da proposta original, destacando-se os trabalhos que fundamentaram o desenvolvimento das arquiteturas apresentadas nesta dissertação.

Relatou-se, então, a exploração algorítmica realizada para a simplificação das equações originais, visando a implementação da transformada em hardware com alto desempenho e baixo custo. Essas equações foram a base para o desenvolvimento das arquiteturas para a ROT direta e inversa.

As arquiteturas desenvolvidas foram implementadas em três versões, uma versão paralela, uma versão configurável e uma versão configurável com um *pipeline* de nove estágios para cada etapa da transformada. As arquiteturas foram descritas em VHDL e sintetizadas para um FPGA Stratix III da Altera.

O principal objetivo do HEVC é obter alta eficiência de codificação para vídeos de resoluções elevadas, acima do *Full HD*. Dessa forma, resoluções mais elevadas também foram foco deste trabalho. As arquiteturas com *pipeline* atingiram

taxas de processamento suficientes para processar, em tempo real, vídeos de altíssima resolução, como os formatos 8K UHD (7680x4320 pixels) a 30 quadros por segundo ou 4K UHD (3840x2160 pixels) a 120 quadros por segundo. As arquiteturas sem *pipeline* obtiveram desempenho suficiente para processar vídeos até a resolução 4K UHD (3840x2160 pixels) a 30 quadros por segundo e até QHD (2560x1440 pixels) a 60 quadros por segundo. As altas taxas de processamento obtidas também são vantajosas considerando-se resoluções mais baixas. Considerando-se, por exemplo, vídeos *Full HD* (ou 1080p), todas as arquiteturas propostas são capazes de realizar o processamento em tempo real com uma frequência de apenas 7,78 MHz.

Assim, foi possível concluir que este trabalho atingiu os objetivos propostos, pois foram geradas três diferentes versões em hardware das transformadas ROT direta e inversa, que são capazes de processar vídeos de resolução muito elevada em tempo real. Além disso, o custo do hardware utilizado foi bastante baixo, especialmente para as versões configuráveis da arquitetura.

REFERÊNCIAS

AGOSTINI, L.V. **Desenvolvimento de Arquiteturas de Alto Desempenho Dedicadas à Compressão de Vídeo Segundo o Padrão H.264/AVC**. 2007. 172f. Tese (Doutorado em Computação) – Programa de Pós-Graduação em Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre.

ALTERA CORPORATION. Design Software Support. Disponível em: <http://www.altera.com/support/software/>. Acesso em: ago. 2012.

BJØNTEGAARD, G. Calculation of average PSNR differences between RD-curves. **ITU-T SG16/Q6 Meeting,** 13. Austin, 2001. VCEG-M33.

BROSS, B. et al. High Efficiency Video Coding (HEVC) Text Specification Draft 8. **JCT-VC Meeting,** 10. Estocolmo, 2012. JCTVC-J1003_d7.

COHEN, R.; YEO C.; JOSHI, R. CE7: Alternative Transforms. **JCT-VC Meeting,** 3. Guangzhou, 2010. JCTVC-C507.

CORRÊA, G. **Controle de Complexidade Computacional em Codificadores de Vídeo de Alta Eficiência**. 2011. 96f. Projeto de Tese (Doutorado em Engenharia Eletrotécnica e de Computadores) – Faculdade de Ciências e Tecnologia, Universidade de Coimbra, Coimbra, Portugal.

FERNANDES, F. Low Complexity Rotational Transform. **JCT-VC Meeting,** 3. Guangzhou, 2010. JCTVC-C096.

GIVENS, J.W. Computation of plane unitary rotations transforming a general matrix to triangular form. **SIAM Journal of Applied Mathematics**, v.6, n.1, p.26-50, 1958.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. ISO Home Page. Disponível em: <http://www.iso.org>. Acesso em: dez. 2011.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO/IEC 13818-2 - MPEG-2 Part 2:** Generic coding of moving pictures and associated audio information: Video, 1996.

INTERNATIONAL TELECOMMUNICATION UNION. ITU-T Home Page. Disponível em: <http://www.itu.int/ITU-T/>. Acesso em: dez. 2011.

INTERNATIONAL TELECOMMUNICATION UNION. **ITU-T Recommendation H.264/AVC (05/03):** advanced video coding for generic audiovisual services. [S.l.], 2003.

INTERNATIONAL TELECOMMUNICATION UNION. **ITU-R Draft Recommendation BT.6/18(Rev.1) (08/12):** Parameter values for UHDTV systems for production and international programme exchange. [S.l.], 2012.

INTERNATIONAL TELECOMMUNICATION UNION. Joint Collaborative Team on Video Coding (JCT-VC). Disponível em: <http://www.itu.int/en/ITU-T/studygroups/com16/video/Pages/jctvc.aspx>. Acesso em: ago. 2012.

ISO/IEC. Vision, Applications and Requirements for High Efficiency Video Coding (HEVC), **ISO/IEC JTC1/SC29/WG11**, W11872. 2011.

ITU-T; ISO/IEC. Joint Call for Proposals on Video Compression Technology, **ITU-T Q6/16**, VCEG-AM91. 2010.

JCT-VC. JCT-VC Document Archive. Disponível em: <http://wftp3.itu.int/av-arch/jctvc-site/>. Acesso em: dez. 2011.

JCT-VC. JCT-VC Document Management System. Disponível em: <http://phenix.int-evry.fr/jct/>. Acesso em: nov. 2012.

JCT-VC. HEVC Reference Software. Disponível em: <https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/>. Acesso em: dez. 2011.

JOSHI, R. CE7: Crosscheck of tool 4 - Samsung's rotational transform (JCTVC-F294). **JCT-VC Meeting,** 6. Torino, 2011. JCTVC-F559.

MA, Z. et al. CE 7: Experimental Results for the Rotational Transform. **JCT-VC Meeting,** 6. Torino, 2011. JCTVC-F294.

MA, Z. et al. Non CE 7: Supplementary Results for the Rotational Transform. **JCT-VC Meeting,** 7. Genebra, 2011. JCTVC-G591.

MCCANN, K. et al. Samsung’s Response to the Call for Proposals on Video Compression Technology. **JCT-VC Meeting,** 1. Dresden, 2010. JCTVC-A124.

KIM, I. et al. HM8: High Efficiency Video Coding (HEVC) Test Model 8 Encoder Description. **JCT-VC Meeting,** 10. Estocolmo, 2012. JCTVC-J1002.

RICHARDSON, I. **The H.264 Advanced Video Compression standard**. 2nd ed. Chichester: John Wiley and Sons, 2010.

SULLIVAN, G.; OHM, J. Meeting report of the first meeting of the Joint Collaborative Team on Video Coding (JCT-VC). **JCT-VC Meeting,** 1. Dresden, 2010. JCTVC-A200.

SULLIVAN, G.; OHM, J. Recent developments in standardization of high efficiency video coding (HEVC). **Proceedings of SPIE - The International Society for Optical Engineering** Vol. 7798, [np], 2010.

SULLIVAN, G.; WIEGAND, T. Draft Requirements for next-generation video coding project. **ITU-T Study Group 16**, VCEG-AL96. 2009.

VIANNA, H.A. et al. High Performance Hardware Architectures for the Inverse Rotational Transform of the Emerging HEVC Standard. In: IEEE ICIP 2012 - IEEE International Conference on Image Processing, 2012, Orlando. **Proceedings...** Los Alamitos: IEEE, 2012. p.189-192.

VIANNA, H.A.; AGOSTINI, L.V. Projeto de Hardware para a Transformada Rotacional do Padrão Emergente de Codificação de Vídeo HEVC. In: XIV Encontro de Pós-Graduação UFPel, 2012, Pelotas. **Anais do...** Pelotas: UFPel, 2012.

VIANNA, H.A. et al. Very High Throughput FPGA Design for Vertical Rotational Transform of HEVC Emergent Video Coding Standard. In: SPL 2012 - VIII Southern Programmable Logic Conference, 2012, Bento Gonçalves. **Proceedings...** Piscataway: IEEE, 2012. p.69-74.

XIPH.ORG FOUNDATION. Test sequences. Disponível em: <http://media.xiph.org/ldv/pub/test_sequences/1080p/>. Acesso em: nov. 2012.

APÊNDICE A EQUAÇÕES ORIGINAIS DA ROT DIRETA E INVERSA

Tabela A.1: Equações originais da ROT Direta Horizontal

indexROT	Estágio 1	Estágio 2
1	a0 = W0 + (-4 * W1 >>5) a3 = W3 + (-4 * W4 >>5)	b1 = W1 + (8 * a0 >>5) b4 = W4 + (8 * a3 >>5)
2	a0 = W0 + (-7 * W1 >>5) a3 = W3 + (-7 * W4 >>5)	b1 = W1 + (13 * a0 >>5) b4 = W4 + (13 * a3 >>5)
3	a0 = W0 + (2 * W1 >>5) a3 = W3 + (2 * W4 >>5)	b1 = W1 + (-5 * a0 >>5) b4 = W4 + (-5 * a3 >>5)
4	a0 = W0 + (-1 * W1 >>5) a3 = W3 + (-1 * W4 >>5)	b1 = W1 + (1 * a0 >>5) b4 = W4 + (1 * a3 >>5)
indexROT	Estágio 3	Estágio 4
1	c0 = a0 + (-4 * b1 >>5) c3 = a3 + (-4 * b4 >>5)	d1 = b1 + (-2 * W2 >>5) d4 = b4 + (-2 * W5 >>5)
2	c0 = a0 + (-7 * b1 >>5) c3 = a3 + (-7 * b4 >>5)	d1 = b1 + (-4 * W2 >>5) d4 = b4 + (-4 * W5 >>5)
3	c0 = a0 + (2 * b1 >>5) b1 = -W2 W2 = b1 c3 = a3 + (2 * b4 >>5) b4 = -W5 W5 = b4	d1 = b1 + (-29 * W2 >>5) d4 = b4 + (-29 * W5 >>5)
4	c0 = a0 + (-1 * b1 >>5) b1 = -W2 W2 = b1 c3 = a3 + (-1 * b4 >>5) b4 = -W5 W5 = b4	d1 = b1 + (-29 * W2 >>5) d4 = b4 + (-29 * W5 >>5)
indexROT	Estágio 5	Estágio 6
1	S2 = W2 + (3 * d1 >>5) S5 = W5 + (3 * d4 >>5)	f1 = d1 + (-2 * S2 >>5) f4 = d4 + (-2 * S5 >>5)
2	S2 = W2 + (7 * d1 >>5) S5 = W5 + (7 * d4 >>5)	f1 = d1 + (-4 * S2 >>5) f4 = d4 + (-4 * S5 >>5)
3	S2 = W2 + (32 * d1 >>5) S5 = W5 + (32 * d4 >>5)	f1 = d1 + (-29 * S2 >>5) f4 = d4 + (-29 * S5 >>5)
4	S2 = W2 + (32 * d1 >>5) S5 = W5 + (32 * d4 >>5)	f1 = d1 + (-29 * S2 >>5) f4 = d4 + (-29 * S5 >>5)
indexROT	Estágio 7	Estágio 8
1	g0 = c0 + (-2 * f1 >>5) g3 = c3 + (-2 * f4 >>5)	S1 = f1 + (4 * g0 >>5) S4 = f4 + (4 * g3 >>5)
2	g0 = c0 + (5 * f1 >>5) g3 = c3 + (5 * f4 >>5)	S1 = f1 + (-10 * g0 >>5) S4 = f4 + (-10 * g3 >>5)
3	g0 = c0 + (5 * f1 >>5) g3 = c3 + (5 * f4 >>5)	S1 = f1 + (-10 * g0 >>5) S4 = f4 + (-10 * g3 >>5)
4	g0 = c0 + (6 * f1 >>5) g3 = c3 + (6 * f4 >>5)	S1 = f1 + (-12 * g0 >>5) S4 = f4 + (-12 * g3 >>5)
indexROT	Estágio 9	
1	S0 = g0 + (-2 * S1 >>5) S3 = g3 + (-2 * S4 >>5)	
2	S0 = g0 + (5 * S1 >>5) S3 = g3 + (5 * S4 >>5)	
3	S0 = g0 + (5 * S1 >>5) S3 = g3 + (5 * S4 >>5)	
4	S0 = g0 + (6 * S1 >>5) S3 = g3 + (6 * S4 >>5)	

Tabela A.2: Equações originais da ROT Direta Vertical

indexROT	Estágio 1	Estágio 2
1	a0 = W0 + (-7 * W1 >>5) a3 = W3 + (-7 * W4 >>5)	b1 = W1 + (14 * a0 >>5) b4 = W4 + (14 * a3 >>5)
2	a0 = W0 + (4 * W1 >>5) a3 = W3 + (4 * W4 >>5)	b1 = W1 + (-8 * a0 >>5) b4 = W4 + (-8 * a3 >>5)
3	a0 = W0 + (-3 * W1 >>5) a3 = W3 + (-3 * W4 >>5)	b1 = W1 + (5 * a0 >>5) b4 = W4 + (5 * a3 >>5)
4	a0 = W0 + (-10 * W1 >>5) a3 = W3 + (-10 * W4 >>5)	b1 = W1 + (18 * a0 >>5) b4 = W4 + (18 * a3 >>5)
indexROT	Estágio 3	Estágio 4
1	c0 = a0 + (-7 * b1 >>5) c3 = a3 + (-7 * b4 >>5)	d1 = b1 + (-7 * W2 >>5) d4 = b4 + (-7 * W5 >>5)
2	c0 = a0 + (4 * b1 >>5) b1 = -W2 W2 = b1 c3 = a3 + (4 * b4 >>5) b4 = -W5 W5 = b4	d1 = b1 + (-22 * W2 >>5) d4 = b4 + (-22 * W5 >>5)
3	c0 = a0 + (-3 * b1 >>5) c3 = a3 + (-3 * b4 >>5)	d1 = b1 + (-5 * W2 >>5) d4 = b4 + (-5 * W5 >>5)
4	c0 = a0 + (-10 * b1 >>5) c3 = a3 + (-10 * b4 >>5)	d1 = b1 + (-6 * W2 >>5) d4 = b4 + (-6 * W5 >>5)
indexROT	Estágio 5	Estágio 6
1	S2 = W2 + (13 * d1 >>5) S5 = W5 + (13 * d4 >>5)	f1 = d1 + (-7 * S2 >>5) f4 = d4 + (-7 * S5 >>5)
2	S2 = W2 + (30 * d1 >>5) S5 = W5 + (30 * d4 >>5)	f1 = d1 + (-22 * S2 >>5) f4 = d4 + (-22 * S5 >>5)
3	S2 = W2 + (9 * d1 >>5) S5 = W5 + (9 * d4 >>5)	f1 = d1 + (-5 * S2 >>5) f4 = d4 + (-5 * S5 >>5)
4	S2 = W2 + (11 * d1 >>5) S5 = W5 + (11 * d4 >>5)	f1 = d1 + (-6 * S2 >>5) f4 = d4 + (-6 * S5 >>5)
indexROT	Estágio 7	Estágio 8
1	g0 = c0 + (5 * f1 >>5) g3 = c3 + (5 * f4 >>5)	S1 = f1 + (-10 * g0 >>5) S4 = f4 + (-10 * g3 >>5)
2	g0 = c0 + (9 * f1 >>5) g3 = c3 + (9 * f4 >>5)	S1 = f1 + (-17 * g0 >>5) S4 = f4 + (-17 * g3 >>5)
3	g0 = c0 + (-3 * f1 >>5) g3 = c3 + (-3 * f4 >>5)	S1 = f1 + (6 * g0 >>5) S4 = f4 + (6 * g3 >>5)
4	g0 = c0 + (-4 * f1 >>5) g3 = c3 + (-4 * f4 >>5)	S1 = f1 + (8 * g0 >>5) S4 = f4 + (8 * g3 >>5)
indexROT	Estágio 9	
1	S0 = g0 + (5 * S1 >>5) S3 = g3 + (5 * S4 >>5)	
2	S0 = g0 + (9 * S1 >>5) S3 = g3 + (9 * S4 >>5)	
3	S0 = g0 + (-3 * S1 >>5) S3 = g3 + (-3 * S4 >>5)	
4	S0 = g0 + (-4 * S1 >>5) S3 = g3 + (-4 * S4 >>5)	

Tabela A.3: Equações originais da ROT Inversa Vertical

indexROT	Estágio 1	Estágio 2
1	g0 = W0 - (5 * W1 >>5) g3 = W3 - (5 * W4 >>5)	f1 = W1 - (-10 * g0 >>5) f4 = W4 - (-10 * g3 >>5)
2	g0 = W0 - (9 * W1 >>5) g3 = W3 - (9 * W4 >>5)	f1 = W1 - (-17 * g0 >>5) f4 = W4 - (-17 * g3 >>5)
3	g0 = W0 - (-3 * W1 >>5) g3 = W3 - (-3 * W4 >>5)	f1 = W1 - (6 * g0 >>5) f4 = W4 - (6 * g3 >>5)
4	g0 = W0 - (-4 * W1 >>5) g3 = W3 - (-4 * W4 >>5)	f1 = W1 - (8 * g0 >>5) f4 = W4 - (8 * g3 >>5)
indexROT	Estágio 3	Estágio 4
1	c0 = g0 - (5 * f1 >>5) c3 = g3 - (5 * f4 >>5)	d1 = f1 - (-7 * W2 >>5) d4 = f4 - (-7 * W5 >>5)
2	c0 = g0 - (9 * f1 >>5) c3 = g3 - (9 * f4 >>5)	d1 = f1 - (-22 * W2 >>5) d4 = f4 - (-22 * W5 >>5)
3	c0 = g0 - (-3 * f1 >>5) c3 = g3 - (-3 * f4 >>5)	d1 = f1 - (-5 * W2 >>5) d4 = f4 - (-5 * W5 >>5)
4	c0 = g0 - (-4 * f1 >>5) c3 = g3 - (-4 * f4 >>5)	d1 = f1 - (-6 * W2 >>5) d4 = f4 - (-6 * W5 >>5)
indexROT	Estágio 5	Estágio 6
1	S2 = W2 - (13 * d1 >>5) S5 = W5 - (13 * d4 >>5)	b1 = d1 - (-7 * S2 >>5) b4 = d4 - (-7 * S5 >>5)
2	S2 = W2 - (30 * d1 >>5) S5 = W5 - (30 * d4 >>5)	b1 = S2 S2 = d1 - (-22 * S2 >>5) b4 = S5 S5 = d4 - (-22 * S5 >>5)
3	S2 = W2 - (9 * d1 >>5) S5 = W5 - (9 * d4 >>5)	b1 = d1 - (-5 * S2 >>5) b4 = d4 - (-5 * S5 >>5)
4	S2 = W2 - (11 * d1 >>5) S5 = W5 - (11 * d4 >>5)	b1 = d1 - (-6 * S2 >>5) b4 = d4 - (-6 * S5 >>5)
indexROT	Estágio 7	Estágio 8
1	a0 = c0 - (-7 * b1 >>5) a3 = c3 - (-7 * b4 >>5)	S1 = b1 - (14 * a0 >>5) S4 = b4 - (14 * a3 >>5)
2	a0 = c0 - (4 * b1 >>5) S2 = -S2 a3 = c3 - (4 * b4 >>5) S5 = -S5	S1 = b1 - (-8 * a0 >>5) S4 = b4 - (-8 * a3 >>5)
3	a0 = c0 - (-3 * b1 >>5) a3 = c3 - (-3 * b4 >>5)	S1 = b1 - (5 * a0 >>5) S4 = b4 - (5 * a3 >>5)
4	a0 = c0 - (-10 * b1 >>5) a3 = c3 - (-10 * b4 >>5)	S1 = b1 - (18 * a0 >>5) S4 = b4 - (18 * a3 >>5)
indexROT	Estágio 9	
1	S0 = a0 - (-7 * S1 >>5) S3 = a3 - (-7 * S4 >>5)	
2	S0 = a0 - (4 * S1 >>5) S3 = a3 - (4 * S4 >>5)	
3	S0 = a0 - (-3 * S1 >>5) S3 = a3 - (-3 * S4 >>5)	
4	S0 = a0 - (-10 * S1 >>5) S3 = a3 - (-10 * S4 >>5)	

Tabela A.4: Equações originais da ROT Inversa Horizontal

indexROT	Estágio 1	Estágio 2
1	g0 = W0 - (-2 * W1 >>5) g3 = W3 - (-2 * W4 >>5)	f1 = W1 - (4 * g0 >>5) f4 = W4 - (4 * g3 >>5)
2	g0 = W0 - (5 * W1 >>5) g3 = W3 - (5 * W4 >>5)	f1 = W1 - (-10 * g0 >>5) f4 = W4 - (-10 * g3 >>5)
3	g0 = W0 - (5 * W1 >>5) g3 = W3 - (5 * W4 >>5)	f1 = W1 - (-10 * g0 >>5) f4 = W4 - (-10 * g3 >>5)
4	g0 = W0 - (6 * W1 >>5) g3 = W3 - (6 * W4 >>5)	f1 = W1 - (-12 * g0 >>5) f4 = W4 - (-12 * g3 >>5)
indexROT	Estágio 3	Estágio 4
1	c0 = g0 - (-2 * f1 >>5) c3 = g3 - (-2 * f4 >>5)	d1 = f1 - (-2 * W2 >>5) d4 = f4 - (-2 * W5 >>5)
2	c0 = g0 - (5 * f1 >>5) c3 = g3 - (5 * f4 >>5)	d1 = f1 - (-4 * W2 >>5) d4 = f4 - (-4 * W5 >>5)
3	c0 = g0 - (5 * f1 >>5) c3 = g3 - (5 * f4 >>5)	d1 = f1 - (-29 * W2 >>5) d4 = f4 - (-29 * W5 >>5)
4	c0 = g0 - (6 * f1 >>5) c3 = g3 - (6 * f4 >>5)	d1 = f1 - (-29 * W2 >>5) d4 = f4 - (-29 * W5 >>5)
indexROT	Estágio 5	Estágio 6
1	S2 = W2 - (3 * d1 >>5) S5 = W5 - (3 * d4 >>5)	b1 = d1 - (-2 * S2 >>5) b4 = d4 - (-2 * S5 >>5)
2	S2 = W2 - (7 * d1 >>5) S5 = W5 - (7 * d4 >>5)	b1 = d1 - (-4 * S2 >>5) b4 = d4 - (-4 * S5 >>5)
3	S2 = W2 - (32 * d1 >>5) S5 = W5 - (32 * d4 >>5)	b1 = S2 S2 = d1 - (-29 * S2 >>5) b4 = S5 S5 = d4 - (-29 * S5 >>5)
4	S2 = W2 - (32 * d1 >>5) S5 = W5 - (32 * d4 >>5)	b1 = S2 S2 = d1 - (-29 * S2 >>5) b4 = S5 S5 = d4 - (-29 * S5 >>5)
indexROT	Estágio 7	Estágio 8
1	a0 = c0 - (-4 * b1 >>5) a3 = c3 - (-4 * b4 >>5)	S1 = b1 - (8 * a0 >>5) S4 = b4 - (8 * a3 >>5)
2	a0 = c0 - (-7 * b1 >>5) a3 = c3 - (-7 * b4 >>5)	S1 = b1 - (13 * a0 >>5) S4 = b4 - (13 * a3 >>5)
3	a0 = c0 - (2 * b1 >>5) S2 = -S2 a3 = c3 - (2 * b4 >>5) S5 = -S5	S1 = b1 - (-5 * a0 >>5) S4 = b4 - (-5 * a3 >>5)
4	a0 = c0 - (-1 * b1 >>5) S2 = -S2 a3 = c3 - (-1 * b4 >>5) S5 = -S5	S1 = b1 - (1 * a0 >>5) S4 = b4 - (1 * a3 >>5)
indexROT	Estágio 9	
1	S0 = a0 - (-4 * S1 >>5) S3 = a3 - (-4 * S4 >>5)	
2	S0 = a0 - (-7 * S1 >>5) S3 = a3 - (-7 * S4 >>5)	
3	S0 = a0 - (2 * S1 >>5) S3 = a3 - (2 * S4 >>5)	
4	S0 = a0 - (-1 * S1 >>5) S3 = a3 - (-1 * S4 >>5)	

APÊNDICE B EQUAÇÕES SIMPLIFICADAS DA ROT DIRETA

Tabela B.1: Equações simplificadas da ROT Direta Horizontal

indexROT	Estágio 1	Estágio 2
1	a0 = W0 - ((W1>>3) + 1) a3 = W3 - ((W4>>3) + 1)	b1 = W1 + (a0>>2) b4 = W4 + (a3>>2)
2	a0 = W0 - (((W1<<3)-W1 >>5) + 1) a3 = W3 - (((W4<<3)-W4 >>5) + 1)	b1 = W1 + ((a0<<3)+(a0<<2)+a0 >>5) b4 = W4 + ((a3<<3)+(a3<<2)+a3 >>5)
3	a0 = W0 + (W1>>4) a3 = W3 + (W4>>4)	b1 = W1 - (((a0<<2)+a0 >>5) + 1) b4 = W4 - (((a3<<2)+a3 >>5) + 1)
4	a0 = W0 - ((W1>>5) + 1) a3 = W3 - ((W4>>5) + 1)	b1 = W1 + (a0>>5) b4 = W4 + (a3>>5)
indexROT	Estágio 3	Estágio 4
1	c0 = a0 - ((b1>>3) + 1) c3 = a3 - ((b4>>3) + 1)	d1 = b1 - ((W2>>4) + 1) d4 = b4 - ((W5>>4) + 1)
2	c0 = a0 - (((b1<<3)-b1 >>5) + 1) c3 = a3 - (((b4<<3)-b4 >>5) + 1)	d1 = b1 - ((W2>>3) + 1) d4 = b4 - ((W5>>3) + 1)
3	c0 = a0 + (b1>>4) b1 = 0 - W2 W2 = b1 c3 = a3 + (b4>>4) b4 = 0 - W5 W5 = b4	d1 = b1 - (((W2<<5)-((W2<<1)+W2) >>5) + 1) d4 = b4 - (((W5<<5)-((W5<<1)+W5) >>5) + 1)
4	c0 = a0 - ((b1>>5) + 1) b1 = 0 - W2 W2 = b1 c3 = a3 - ((b4>>5) + 1) b4 = 0 - W5 W5 = b4	d1 = b1 - (((W2<<5)-((W2<<1)+W2) >>5) + 1) d4 = b4 - (((W5<<5)-((W5<<1)+W5) >>5) + 1)
indexROT	Estágio 5	Estágio 6
1	S2 = W2 + ((d1<<1)+d1 >>5) S5 = W5 + ((d4<<1)+d4 >>5)	f1 = d1 - ((S2>>4) + 1) f4 = d4 - ((S5>>4) + 1)
2	S2 = W2 + ((d1<<3)-d1 >>5) S5 = W5 + ((d4<<3)-d4 >>5)	f1 = d1 - ((S2>>3) + 1) f4 = d4 - ((S5>>3) + 1)
3	S2 = W2 + d1 S5 = W5 + d4	f1 = d1 - (((S2<<5)-((S2<<1)+S2) >>5) + 1) f4 = d4 - (((S5<<5)-((S5<<1)+S5) >>5) + 1)
4	S2 = W2 + d1 S5 = W5 + d4	f1 = d1 - (((S2<<5)-((S2<<1)+S2) >>5) + 1) f4 = d4 - (((S5<<5)-((S5<<1)+S5) >>5) + 1)
indexROT	Estágio 7	Estágio 8
1	g0 = c0 - ((f1>>4) + 1) g3 = c3 - ((f4>>4) + 1)	S1 = f1 + (g0>>3) S4 = f4 + (g3>>3)
2	g0 = c0 + ((f1<<2)+f1 >>5) g3 = c3 + ((f4<<2)+f4 >>5)	S1 = f1 - (((g0<<3)+(g0<<1) >>5) + 1) S4 = f4 - (((g3<<3)+(g3<<1) >>5) + 1)
3	g0 = c0 + ((f1<<2)+f1 >>5) g3 = c3 + ((f4<<2)+f4 >>5)	S1 = f1 - (((g0<<3)+(g0<<1) >>5) + 1) S4 = f4 - (((g3<<3)+(g3<<1) >>5) + 1)
4	g0 = c0 + ((f1<<2)+(f1<<1) >>5) g3 = c3 + ((f4<<2)+(f4<<1) >>5)	S1 = f1 - (((g0<<3)+(g0<<2) >>5) + 1) S4 = f4 - (((g3<<3)+(g3<<2) >>5) + 1)
indexROT	Estágio 9	
1	S0 = g0 - ((S1>>4) + 1) S3 = g3 - ((S4>>4) + 1)	
2	S0 = g0 + ((S1<<2)+S1 >>5) S3 = g3 + ((S4<<2)+S4 >>5)	
3	S0 = g0 + ((S1<<2)+S1 >>5) S3 = g3 + ((S4<<2)+S4 >>5)	
4	S0 = g0 + ((S1<<2)+(S1<<1) >>5) S3 = g3 + ((S1<<2)+(S1<<1) >>5)	

Tabela B.2: Equações simplificadas da ROT Direta Vertical

indexROT	Estágio 1	Estágio 2
1	a0 = W0 - (((W1<<3)-W1 >>5) + 1) a3 = W3 - (((W4<<3)-W4 >>5) + 1)	b1 = W1 + ((a0<<4)-(a0<<1) >>5) b4 = W4 + ((a3<<4)-(a3<<1) >>5)
2	a0 = W0 + (W1>>3) a3 = W3 + (W4>>3)	b1 = W1 - ((a0>>2) + 1) b4 = W4 - ((a3>>2) + 1)
3	a0 = W0 - (((W1<<1)+W1 >>5) + 1) a3 = W3 - (((W4<<1)+W4 >>5) + 1)	b1 = W1 + ((a0<<2)+a0 >>5) b4 = W4 + ((a3<<2)+a3 >>5)
4	a0 = W0 - (((W1<<3)+(W1<<1) >>5) + 1) a3 = W3 - (((W4<<3)+(W4<<1) >>5) + 1)	b1 = W1 + ((a0<<4)+(a0<<1) >>5) b4 = W4 + ((a3<<4)+(a3<<1) >>5)
indexROT	Estágio 3	Estágio 4
1	c0 = a0 - (((b1<<3)-b1 >>5) + 1) c3 = a3 - (((b4<<3)-b4 >>5) + 1)	d1 = b1 - (((W2<<3)-W2 >>5) + 1) d4 = b4 - (((W5<<3)-W5 >>5) + 1)
2	c0 = a0 + (b1>>3) b1 = 0 - W2 W2 = b1 c3 = a3 + (b4>>3) b4 = 0 - W5 W5 = b4	d1 = b1 - (((W2<<4)+(W2<<2)+(W2<<1) >>5) + 1) d4 = b4 - (((W5<<4)+(W5<<2)+(W5<<1) >>5) + 1)
3	c0 = a0 - (((b1<<1)+b1 >>5) + 1) c3 = a3 - (((b4<<1)+b4 >>5) + 1)	d1 = b1 - (((W2<<2)+W2 >>5) + 1) d1 = b1 - (((W2<<2)+W2 >>5) + 1)
4	c0 = a0 - (((b1<<3)+(b1<<1) >>5) + 1) c3 = a3 - (((b4<<3)+(b4<<1) >>5) + 1)	d1 = b1 - (((W2<<2)+(W2<<1) >>5) + 1) d4 = b4 - (((W5<<2)+(W5<<1) >>5) + 1)
indexROT	Estágio 5	Estágio 6
1	S2 = W2 + ((d1<<3)+(d1<<2)+d1 >>5) S5 = W5 + ((d4<<3)+(d4<<2)+d4 >>5)	f1 = d1 - (((S2<<3)-S2 >>5) + 1) f4 = d4 - (((S5<<3)-S5 >>5) + 1)
2	S2 = W2 + ((d1<<5)-(d1<<1) >>5) S5 = W5 + ((d4<<5)-(d4<<1) >>5)	f1 = d1 - (((S2<<4)+(S2<<2)+(S2<<1) >>5) + 1) f4 = d4 - (((S5<<4)+(S5<<2)+(S5<<1) >>5) + 1)
3	S2 = W2 + ((d1<<3)+d1 >>5) S5 = W5 + ((d4<<3)+d4 >>5)	f1 = d1 - (((S2<<2)+S2 >>5) + 1) f4 = d4 - (((S5<<2)+S5 >>5) + 1)
4	S2 = W2 + ((d1<<3)+(d1<<1)+d1 >>5) S5 = W5 + ((d4<<3)+(d4<<1)+d4 >>5)	f1 = d1 - (((S2<<2)+(S2<<1) >>5) + 1) f4 = d4 - (((S5<<2)+(S5<<1) >>5) + 1)
indexROT	Estágio 7	Estágio 8
1	g0 = c0 + ((f1<<2)+f1 >>5) g3 = c3 + ((f4<<2)+f4 >>5)	S1 = f1 - (((g0<<3)+(g0<<1) >>5) + 1) S4 = f4 - (((g3<<3)+(g3<<1) >>5) + 1)
2	g0 = c0 + ((f1<<3)+f1 >>5) g3 = c3 + ((f4<<3)+f4 >>5)	S1 = f1 - (((g0<<4)+g0 >>5) + 1) S4 = f4 - (((g3<<4)+g3 >>5) + 1)
3	g0 = c0 - (((f1<<1)+f1 >>5) + 1) g3 = c3 - (((f4<<1)+f4 >>5) + 1)	S1 = f1 + ((g0<<2)+(g0<<1) >>5) S4 = f4 + ((g3<<2)+(g3<<1) >>5)
4	g0 = c0 - ((f1>>3) + 1) g3 = c3 - ((f4>>3) + 1)	S1 = f1 + (g0>>2) S4 = f4 + (g3>>2)
indexROT	Estágio 9	
1	S0 = g0 + ((S1<<2)+S1 >>5) S3 = g3 + ((S4<<2)+S4 >>5)	
2	S0 = g0 + ((S1<<3)+S1 >>5) S3 = g3 + ((S4<<3)+S4 >>5)	
3	S0 = g0 - (((S1<<1)+S1 >>5) + 1) S3 = g3 - (((S4<<1)+S4 >>5) + 1)	
4	S0 = g0 - ((S1>>3) + 1) S3 = g3 - ((S4>>3) + 1)	

APÊNDICE C EQUAÇÕES SIMPLIFICADAS DA ROT INVERSA

Tabela C.1: Equações simplificadas da ROT Inversa Vertical

indexROT	Estágio 1	Estágio 2
1	g0 = W0 - ((W1<<2)+W1 >>5) g3 = W3 - ((W4<<2)+W4 >>5)	f1 = W1 + ((g0<<3)+(g0<<1) >>5) + 1 f4 = W4 + ((g3<<3)+(g3<<1) >>5) + 1
2	g0 = W0 - ((W1<<3)+W1 >>5) g3 = W3 - ((W4<<3)+W4 >>5)	f1 = W1 + ((g0<<4)+g0 >>5) + 1 f4 = W4 + ((g3<<4)+g3 >>5) + 1
3	g0 = W0 + ((W1<<1)+W1 >>5) + 1 g3 = W3 + ((W4<<1)+W4 >>5) + 1	f1 = W1 - ((g0<<2)+(g0<<1) >>5) f4 = W4 - ((g3<<2)+(g3<<1) >>5)
4	g0 = W0 + (W1>>3) + 1 g3 = W3 + (W4>>3) + 1	f1 = W1 - (g0>>2) f4 = W4 - (g3>>2)
indexROT	Estágio 3	Estágio 4
1	c0 = g0 - ((f1<<2)+f1 >>5) c3 = g3 - ((f4<<2)+f4 >>5)	d1 = f1 + ((W2<<3)-W2 >>5) + 1 d4 = f4 + ((W5<<3)-W5 >>5) + 1
2	c0 = g0 - ((f1<<3)+f1 >>5) c3 = g3 - ((f4<<3)+f4 >>5)	d1 = f1 + ((W2<<4)+(W2<<2)+(W2<<1) >>5) + 1 d4 = f4 + ((W5<<4)+(W5<<2)+(W5<<1) >>5) + 1
3	c0 = g0 + ((f1<<1)+f1 >>5) + 1 c3 = g3 + ((f4<<1)+f4 >>5) + 1	d1 = f1 + ((W2<<2)+W2 >>5) + 1 d4 = f4 + ((W5<<2)+W5 >>5) + 1
4	c0 = g0 + (f1>>3) + 1 c3 = g3 + (f4>>3) + 1	d1 = f1 + ((W2<<2)+(W2<<1) >>5) + 1 d4 = f4 + ((W5<<2)+(W5<<1) >>5) + 1
indexROT	Estágio 5	Estágio 6
1	S2 = W2 - ((d1<<3)+(d1<<2)+d1 >>5) S5 = W5 - ((d4<<3)+(d4<<2)+d4 >>5)	b1 = d1 + ((S2<<3)-S2 >>5) + 1 b4 = d4 + ((S5<<3)-S5 >>5) + 1
2	S2 = W2 - ((d1<<5)-(d1<<1) >>5) S5 = W5 - ((d4<<5)-(d4<<1) >>5)	b1 = S2 S2 = d1 + ((S2<<4)+(S2<<2)+(S2<<1) >>5) + 1 b4 = S5 S5 = d4 + ((S5<<4)+(S5<<2)+(S5<<1) >>5) + 1
3	S2 = W2 - ((d1<<3)+d1 >>5) S5 = W5 - ((d4<<3)+d4 >>5)	b1 = d1 + ((S2<<2)+S2 >>5) + 1 b4 = d4 + ((S5<<2)+S5 >>5) + 1
4	S2 = W2 - ((d1<<3)+(d1<<1)+d1 >>5) S5 = W5 - ((d4<<3)+(d4<<1)+d4 >>5)	b1 = d1 + ((S2<<2)+(S2<<1) >>5) + 1 b4 = d4 + ((S5<<2)+(S5<<1) >>5) + 1
indexROT	Estágio 7	Estágio 8
1	a0 = c0 + ((b1<<3)-b1 >>5) + 1 a3 = c3 + ((b4<<3)-b4 >>5) + 1	S1 = b1 - ((a0<<4)-(a0<<1) >>5) S4 = b4 - ((a3<<4)-(a3<<1) >>5)
2	a0 = c0 - (b1>>3) S2 = 0 - S2 a3 = c3 - (b4>>3) S5 = 0 - S5	S1 = b1 + (a0>>2) + 1 S4 = b4 + (a3>>2) + 1
3	a0 = c0 + ((b1<<1)+b1 >>5) + 1 a3 = c3 + ((b4<<1)+b4 >>5) + 1	S1 = b1 - ((a0<<2)+a0 >>5) S4 = b4 - ((a3<<2)+a3 >>5)
4	a0 = c0 + ((b1<<3)+(b1<<1) >>5) + 1 a3 = c3 + ((b4<<3)+(b4<<1) >>5) + 1	S1 = b1 - ((a0<<4)+(a0<<1) >>5) S4 = b4 - ((a3<<4)+(a3<<1) >>5)
indexROT	Estágio 9	
1	S0 = a0 + ((S1<<3)-S1 >>5) + 1 S3 = a3 + ((S4<<3)-S4 >>5) + 1	
2	S0 = a0 - (S1>>3) S3 = a3 - (S4>>3)	
3	S0 = a0 + ((S1<<1)+S1 >>5) + 1 S3 = a3 + ((S4<<1)+S4 >>5) + 1	
4	S0 = a0 + ((S1<<3)+(S1<<1) >>5) + 1 S3 = a3 + ((S4<<3)+(S4<<1) >>5) + 1	

Tabela C.2: Equações simplificadas da ROT Inversa Horizontal

indexROT	Estágio 1	Estágio 2
1	g0 = W0 + (W1>>4) + 1 g3 = W3 + (W4>>4) + 1	f1 = W1 - (g0 >>3) f4 = W4 - (g3 >>3)
2	g0 = W0 - ((W1<<2)+W1 >>5) g3 = W3 - ((W4<<2)+W4 >>5)	f1 = W1 + ((g0<<3)+(g0<<1) >>5) + 1 f4 = W4 + ((g3<<3)+(g3<<1) >>5) + 1
3	g0 = W0 - ((W1<<2)+W1 >>5) g3 = W3 - ((W4<<2)+W4 >>5)	f1 = W1 + ((g0<<3)+(g0<<1) >>5) + 1 f4 = W4 + ((g3<<3)+(g3<<1) >>5) + 1
4	g0 = W0 - ((W1<<2)+(W1<<1) >>5) g3 = W3 - ((W4<<2)+(W4<<1) >>5)	f1 = W1 + ((g0<<3)+(g0<<2) >>5) + 1 f4 = W4 + ((g3<<3)+(g3<<2) >>5) + 1
indexROT	Estágio 3	Estágio 4
1	c0 = g0 + (f1>>4) + 1 c3 = g3 + (f4>>4) + 1	d1 = f1 + (W2>>4) + 1 d4 = f4 + (W5>>4) + 1
2	c0 = g0 - ((f1<<2)+f1 >>5) c3 = g3 - ((f4<<2)+f4 >>5)	d1 = f1 + (W2>>3) + 1 d4 = f4 + (W5>>3) + 1
3	c0 = g0 - ((f1<<2)+f1 >>5) c3 = g3 - ((f4<<2)+f4 >>5)	d1 = f1 + ((W2<<5)-((W2<<1)+W2) >>5) + 1 d4 = f4 + ((W5<<5)-((W5<<1)+W5) >>5) + 1
4	c0 = g0 - ((f1<<2)+(f1<<1) >>5) c3 = g3 - ((f4<<2)+(f4<<1) >>5)	d1 = f1 + ((W2<<5)-((W2<<1)+W2) >>5) + 1 d4 = f4 + ((W5<<5)-((W5<<1)+W5) >>5) + 1
indexROT	Estágio 5	Estágio 6
1	S2 = W2 - ((d1<<1)+d1 >>5) S5 = W5 - ((d4<<1)+d4 >>5)	b1 = d1 + (S2>>4) + 1 b4 = d4 + (S5>>4) + 1
2	S2 = W2 - ((d1<<3)-d1 >>5) S5 = W5 - ((d4<<3)-d4 >>5)	b1 = d1 + (S2>>3) + 1 b4 = d4 + (S5>>3) + 1
3	S2 = W2 - d1 S5 = W5 - d4	b1 = S2 S2 = d1 + ((S2<<5)-((S2<<1)+S2) >>5) + 1 b4 = S5 S5 = d4 + ((S5<<5)-((S5<<1)+S5) >>5) + 1
4	S2 = W2 - d1 S5 = W5 - d4	b1 = S2 S2 = d1 + ((S2<<5)-((S2<<1)+S2) >>5) + 1 b4 = S5 S5 = d4 + ((S5<<5)-((S5<<1)+S5) >>5) + 1
indexROT	Estágio 7	Estágio 8
1	a0 = c0 + (b1>>3) + 1 a3 = c3 + (b4>>3) + 1	S1 = b1 - (a0>>2) S4 = b4 - (a3>>2)
2	a0 = c0 + ((b1<<3)-b1 >>5) + 1 a3 = c3 + ((b4<<3)-b4 >>5) + 1	S1 = b1 - ((a0<<3)+(a0<<2)+a0 >>5) S4 = b4 - ((a3<<3)+(a3<<2)+a3 >>5)
3	a0 = c0 - (b1>>4) S2 = 0 - S2 a3 = c3 - (b4>>4) S5 = 0 - S5	S1 = b1 + ((a0<<2)+a0 >>5) + 1 S4 = b4 + ((a3<<2)+a3 >>5) + 1
4	a0 = c0 + (b1>>5) + 1 S2 = 0 - S2 a3 = c3 + (b4>>5) + 1 S5 = 0 - S5	S1 = b1 - (a0>>5) S4 = b4 - (a3>>5)
indexROT	Estágio 9	
1	S0 = a0 + (S1>>3) + 1 S3 = a3 + (S4>>3) + 1	
2	S0 = a0 + ((S1<<3)-S1 >>5) + 1 S3 = a3 + ((S4<<3)-S4 >>5) + 1	
3	S0 = a0 - (S1>>4) S3 = a3 - (S4>>4)	
4	S0 = a0 + (S1>>5) + 1 S3 = a3 + (S4>>5) + 1	

APÊNDICE D PUBLICAÇÕES GERADAS A PARTIR DESTE TRABALHO

Resultados parciais do trabalho apresentado nesta dissertação foram publicados nos anais dos seguintes eventos:

- (VIANNA, SANCHEZ, PORTO, AGOSTINI) – High Performance Hardware Architectures for the Inverse Rotational Transform of the Emerging HEVC Standard. 2012 IEEE International Conference on Image Processing (ICIP 2012). Orlando, Florida, EUA. (Qualis A1)
- (VIANNA, ANDERSSON, SANCHEZ, AGOSTINI) - Very High Throughput FPGA Design for Vertical Rotational Transform of HEVC Emergent Video Coding Standard. VIII Southern Programmable Logic Conferece (SPL 2012). Bento Gonçalves, RS. (Qualis B4)
- (VIANNA, AGOSTINI) – Projeto de Hardware para a Transformada Rotacional do Padrão Emergente de Codificação de Vídeo HEVC. XIV Encontro de Pós-Graduação da UFPel (ENPOS 2012). Pelotas, RS.