PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Arquitetura de *Hardware* Reconfigurável Paralela Dedicada para a Implementação da SA-DCT

Amanda Regina Mascarenhas Diniz

Belo Horizonte

2008

FICHA CATALOGRÁFICA
Elaborada pela Biblioteca da Pontifícia Universidade Católica de Minas Gerais

Diniz, Amanda Regina Mascarenhas
D585a Arquitetura de *hardware* reconfigurável paralela dedicada para a implementação da SA-DCT / Amanda Regina Mascarenhas Diniz. Belo Horizonte, 2008.
91f. : Il.

Orientador: Carlos Augusto Paiva da Silva Martins
Co-orientadora: Flávia Magalhães Freitas Ferreira
Dissertação (Mestrado) – Pontifícia Universidade Católica de Minas Gerais. Programa de Pós-Graduação em Engenharia Elétrica

1. Programação orientada a objetos (Computação). 2. Arquitetura de computador. 3. Sistemas de controle ajustável. 4. Processamento paralelo (Computação). I. Martins, Augusto Paiva da Silva. II. Ferreira, Flávia Magalhães Freitas. III. Pontifícia Universidade Católica de Minas Gerais. Programa de Pós-Graduação em Engenharia Elétrica. IV.Título.

CDU: 681.3.06

Amanda Regina Mascarenhas Diniz

Arquitetura de *Hardware* Reconfigurável Paralela Dedicada para a Implementação da SA-DCT

Dissertação apresentada ao Curso de Pós-Graduação em Engenharia Elétrica da Pontifícia Universidade Católica de Minas Gerais como requisito parcial para obtenção do grau de Mestre em Engenharia Elétrica.

Orientador: Dr. Carlos Augusto Paiva da Silva Martins

Co-orientadora: Dra. Flávia Magalhães Freitas Ferreira

Belo Horizonte
2008

Ao Cláudio, o meu amor; aos meus pais, os meus grandes mestres e amigos e aos meus queridos irmãos, cunhadas e sobrinhos.

AGRADECIMENTOS

Não me conformo em colocar apenas o meu nome na autoria deste trabalho. Com certeza, não faria um rascunho sequer razoável, se não fosse a contribuição de muitas pessoas nessa empreitada. Assim, gostaria de deixar aqui as minhas reverências e o meu muito obrigado:

- ao Prof. Dr. Carlos Augusto Paiva da Silva Martins pelo incentivo, apoio e dedicação a esta pesquisa. Sem dúvida não encontraria em livro algum o que você me ensinou durante estes anos de convivência e trabalho em equipe.
- à Profa. Dra. Flávia Magalhães Freitas pelo interesse com que acompanhou esta pesquisa, contribuindo, a todo momento, com seus conhecimentos.
- aos meus amáveis pais pelo eterno carinho e dedicação. Faltam-me palavras para expressar o grande amor, admiração e gratidão que sinto por vocês.
- ao Cláudio, meu grande amor, sempre ao meu lado, incentivando e compreendendo. E aos seus pais, Lídia e Tadeu.
- aos meus queridos irmãos e cunhadas tão presentes em minha vida e aos meus adoráveis e encantadores sobrinhos que a cada dia me surpreendem mais com as suas conquistas. Quando estou perto de vocês, entro num mundo encantado de sonhos e brincadeiras.
- às minhas amigas de todas as horas, Denise e Adriana. Como é bom ter vocês sempre por perto para desabar, rir bastante e até mesmo chorar um pouquinho.
- aos amigos com quem tive a honra de trabalhar, Chico, Cássio, P. J., Franco e colegas do LSDC. O que seria de mim sem o conhecimento compartilhado com vocês?
- aos amigos das horas de descontração no mestrado, Breno, Leandro, Luíz, Léia, Celso, Samuel e Josy. Tenham a certeza de que vocês tornaram muito mais prazeroso este trabalho.
- aos colegas da Cemig, por me propiciarem um ambiente de aprendizado, solidariedade e companherismo, amenizando a correria e o stress do dia-a-dia.

- a todos os professores, funcionários e amigos do PPGEE sempre tão educados e atenciosos comigo.

- a Deus, por tornar tudo isto possível, por me proporcionar grandes oportunidades, por me cercar de pessoas muito queridas, por iluminar o meu caminho.

RESUMO

Esta dissertação trata da codificação adaptativa à forma (*Shape Adaptive Discrete Cosine Transform* - SA-DCT) que é uma das técnicas utilizadas no processamento dos blocos de borda na codificação orientada por objeto. Essa técnica, apesar de apresentar grande relação sinal ruído para médias e altas taxas de compressão e para blocos de borda pouco preenchidos por pixels do objeto (independente da taxa de bits), dificilmente é implementada em tempo real, devido à sua grande complexidade computacional. Nesse contexto, foi proposta uma arquitetura de *hardware* que implemente a SA-DCT, utilizando computação reconfigurável em conjunto com técnicas de paralelismo no processamento dos dados. Nos resultados de simulação, verificou-se que é possível reduzir bastante o tempo de processamento da SA-DCT com a utilização da arquitetura proposta, o que possibilitará a sua implementação em tempo real, conforme desejado. Verificou-se também que a utilização dos recursos de reconfigurabilidade torna a arquitetura mais flexível, aumenta a tolerância a falhas no circuito digital e possibilita uma redução no custo de prototipação dos projetos.

Palavras chaves: SA-DCT, computação reconfigurável, paralelismo, arquitetura de *hardware*, codificação orientada por objeto, desempenho, flexibilidade, tolerância a falhas.

ABSTRACT

This dissertation deals with Shape Adaptive Discrete Cosine Transform (SA-DCT), which is one of image processing techniques used in the processing of edge blocks in the object-oriented coding. This technique presents high values of PSNR (Peak Signal-to-Noise Ratio) for medium and high compression rates and also for edge blocks filled with few pixels of the object (regardless of bit rates). Despite these advantages, the SA-DCT is hardly ever implemented in real-time conditions due to its computational complexity. In this context, an architecture of hardware that implements this technique using reconfigurable computing and parallelism in the SA-DCT algorithm implementation to reduce processing times was proposed. The simulation results showed that it is possible to reduce considerably the SA-DCT processing times by using the architecture proposed, which would make the implementation of the SA-DCT in real time possible, as desired. It was also verified that the use of reconfigurability resources makes the architecture more flexible, increases its tolerance to failures in digital circuit and may reduce the project prototyping costs.

Key words: SA-DCT, reconfigurable computing, paralelism, hardware architecture, object-oriented coding technique, high performance, flexibility.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

1D-DCT - *Unidimensional Discrete Cosine Transform* (Transformada Discreta do Cosseno Unidimensional)

2D-DCT - *Bidimensional Discrete Cosine Transform* (Transformada Discreta do Cosseno Bidimensional)

ASIC - *Application Specific Integrated Circuits* (Circuito Integrado de Aplicação Específica)

CLB - *Configurable Logic Block*

CMOS - *Complementary Metal Oxide Semiconductor*

CPLD - *Complex Programmable Logic Devices* (Dispositivo Lógico Programável Complexo)

CSD - *Canonical-Signed-Digit*

CSHM - *Computation Sharing Multiplier*

dB - Decibel

DCTs-VS - *Discrete Cosine Transform - Variable Size* (Transformadas Discreta do Cosseno de Tamanho Variável)

DDL - *Description Definition Language*

DDG - *Data Dependence Graph* (Grafo de Dependência de Dados)

DMIF - *Delivery Multimidia Integration Framework*

DVD - *Digital Video Disc*

EI - *Extension/Interpolation*

EI-DCT - *Extension/Interpolation* - DCT

EMQ - Erro Médio Quadrático

FPGA - *Field Programmable Gate Array*

GPP - *General Purpose Processor* (Processadores de Propósito Geral)

HDL - *Hardware Description Language* (Linguagem de Descrição *Hardware*)

HDTV - *High Definition Television* (Televisão de Alta Definição)

HF - *Hardware* Fixo

HP - *Hardware* Programável

IEC - *International Electrotechnical Commission*

IOB - *Input Output Block* (Bloco de Entrada e Saída)

ISE - *Integrated Software Environment*

ISO - *International Organization for Standardization* ITU - *International Telecommunication Union*

LPE - *Low Pass Extrapolation*

LUT - *Lookup table*

MIMD - *Multiple Instruction Stream Multiple Data Stream*

MPEG - *Motion Picture Experts Group*

NTSC - *National Television System(s) Committee*

PAL - *Phase Alternate Lines*

PLD - *Programmable Logic Device* (Dispositivo Lógico Programável)

PSNR - *Peak Signal-to-Noise Ratio* (Relação Sinal Ruído de Pico)

RLE - *Run Length Encoding* (Código de Comprimento de Corrida)

RPR - Razão Pico-Ruído (dB)

SA-DCT - *Shape-Adaptive DCT* (DCT Adaptativa à Forma)

SDTV - *Standard Definition Television* (Televisão de Definição Padrão)

SIMD - *Single Instruction Stream Multiple Data Stream*

VHDL - VHSIC + *Hardware Description Language*

VHSIC - *Very High Speed Integrated Circuits*

XML - *eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo, apresentam-se o contexto em que a pesquisa se insere, o problema motivador, os objetivos e as metas a serem atingidas no presente trabalho. Em seguida, são apresentadas a justificativa e o escopo da pesquisa e, por fim, é exposta a organização geral do texto.

1.1 Contexto da pesquisa

Os estudos das técnicas de compressão digital de imagens estáticas e dinâmicas têm experimentado um crescimento significativo nas últimas décadas. Isso se deve à utilização cada vez maior da imagem nos mais diversos campos da ciência, entretenimento e tecnologia. Entre as diversas áreas de aplicação envolvendo processamento de imagens, podem ser citadas, a título de ilustração, algumas de grande relevância, como:

- Sistemas de televisão digital;
- Imageamento médico para diagnóstico, como ultrassonografia, tomografia computadorizada e ressonância magnética;
- Sistemas de segurança, como videomonitoramento e controle de acesso por reconhecimento de impressões digitais, face ou íris;
- Controle de qualidade nas indústrias;
- Sistemas de videoconferência;
- Transmissões de informação via rede telefônica convencional ou móvel;
- Sensoriamento remoto utilizado em medições de fronteiras e previsão climática;
- Astronomia;
- Internet e vídeos digitais.

A utilização da imagem digital está se tornando tão rotineira que os meios de transmissão e as memórias usados para transmitir e armazenar as imagens precisam, a todo momento, ser redimensionados para acompanhar tal crescimento. Nesse contexto, surgem várias técnicas de compressão que têm como objetivo diminuir a quantidade de bits utilizada na representação das

imagens, conservando suas informações básicas, de tal forma que as informações removidas não sejam perceptíveis pelo olho humano.

Com o objetivo de padronizar todas essas técnicas, alguns orgãos como a *International Organization for Standardization* (ISO), o *International Telecommunication Union* (ITU) e o *International Electrotechnical Commission* (IEC) desenvolveram padrões que, atualmente, são adotados em todas as aplicações que utilizam técnicas de compressão de imagens. Entre esses vários padrões, destaca-se o padrão de compressão de vídeo MPEG-4 (JTC1/SC29/WG11, March 1998) desenvolvido pelo *Motion Picture Experts Group* (MPEG) da ISO/IEC.

O padrão MPEG-4 foi desenvolvido a partir de 1994, tornando-se uma recomendação internacional em 1999, para ser um padrão para aplicações de televisão, cinema e internet. Esse padrão consiste de partes relacionadas que podem ser implementadas individualmente ou combinadas com outras partes, as quais serão melhor explicadas no Capítulo 2. A parte 2 do padrão MPEG-4 trata dos tópicos relacionados à codificação visual orientada por objeto e será o enfoque desta pesquisa, devido ao fato de esse tipo de codificação proporcionar, além da melhoria da qualidade subjetiva da imagem, facilidades decorrentes da manipulação independente dos objetos dentro de um mesmo quadro.

A codificação orientada por objeto manipula, de forma independente, os objetos de uma cena de vídeo (*Video Objects* - VOs), que são caracterizados por sua forma, textura e movimento. Nessa codificação, cada objeto da imagem é inscrito em uma região retangular, que é então particionada em blocos. Existem três tipos diferentes de blocos: os que não possuem nenhum pixel do objeto, chamados de blocos transparentes, os que estão situados completamente dentro do objeto, conhecidos como blocos cheios e os que são parcialmente preenchidos por pixels do objeto, chamados blocos de borda (JAIN, A., 1989).

A codificação dos blocos de borda na codificação orientada por objeto pode ser tratada utilizando-se duas abordagens distintas:

- A Transforma Discreta do Cosseno (*Discrete Cosine Transform* - DCT) baseada em blocos, que associa a utilização de algoritmos de extrapolação (FRANKE, U.; MESTER, R.; AACH, T., 1988; KAUP, A.; AACH, T., 1994; KAUP, A., 1999) à DCT Bidimensional (2D-DCT) (AHMED, N.; NATRAJAN, T.; RAO, K. R., 1984);

- A codificação adaptativa à forma, que utiliza o algoritimo da *Shape Adaptive DCT* (SA-DCT) (SIKORA, T.; MAKAI, B., 1995).

Ambas abordagens transformam os pixels no domínio espacial da imagem, que normalmente apresentam uma alta correlação, em coeficientes mais descorrelacionados no domínio das freqüências espaciais. Isso possibilita uma significativa diminuição dos dados a serem pro-

cessados, permitindo a redução da memória de armazenamento e da faixa de passagem do canal de transmissão.

Esta pesquisa tratará da codificação adaptativa à forma, visto que essa técnica apresenta maior Relação Sinal Ruído de Pico (*Peak Signal-to-Noise Ratio* - PSNR) para médias e altas taxas de compressão e também apresenta maiores PSNR para blocos de borda pouco preenchidos por pixels do objeto, independente da taxa de bits (FERREIRA, F. M. F., 2004). Além disso, a SA-DCT carece de codificadores que a implementem em tempo real, devido à sua grande complexidade computacional, à inexistência de algoritmos rápidos e ao alto tempo necessário para o seu processamento.

1.2 Problema motivador

O problema motivador desta pesquisa é a necessidade de tornar o algoritmo da SA-DCT mais flexível no que diz respeito ao tempo de processamento, à demanda de recursos lógicos e à qualidade da imagem reconstruída. É muito importante ter esse tipo de flexibilidade no processamento de imagens, uma vez que os parâmetros arquiteturais podem variar de acordo com os requisitos de cada aplicação e com as variações nas cargas de trabalho (quantidade e comprimento das colunas/linhas de pixels do objeto em cada bloco de borda e quantidade de bits por coeficiente). Dessa forma, investiga-se a possibilidade de utilizar computação reconfigurável e paralelismo considerando as várias etapas do algoritmo da SA-DCT que serão mapeadas em diferentes unidades do nível mais alto da arquitetura, denominado nível sistêmico. Com o objetivo de validar o ganho obtido com a utilização da computação reconfigurável e do paralelismo, serão utilizados diferentes blocos de borda (cargas de trabalho) que, em função das suas características, possuem maior ou menor paralelismo intrínseco (operações sem dependências, que podem ser executadas em paralelo) e por isso possuem arquiteturas ideais ou ótimas diferentes.

Atualmente, existem algumas abordagens de implementação da 2D-DCT e da SA-DCT que utilizam computação reconfigurável para diminuir a quantidade de recursos lógicos, o tempo de processamento e/ou o consumo de potência. Uma abordagem propõe a utilização de *Field Programmable Gate Arrays* (FPGAs) para se alterar a estrutura lógica interna da 2D-DCT, como compartilhamento de multiplicadores (PARK, J.; KWON, S.; ROY, K., 2002) ou redução do número de bits utilizados no cálculo dos coeficientes (PARK, J.; ROY, K., 2004; PARK, J.; CHOI, J. H.; ROY, K., 2006), de modo que se possa escolher entre produzir uma melhor qualidade de imagem ou reduzir o consumo de potência. Já uma outra abordagem explora a computação reconfigurável para optar-se entre diferentes técnicas de codificação de imagens, utilizando o recurso de reconfiguração para implementar quatro algoritmos de compressão em tempo real (SCHONER, B. et al., 1995).

É importante destacar que, na literatura pesquisada, também foram encontrados alguns trabalhos que propõem a implementação de técnicas de codificação de imagens por transformada em FPGAs, mas não exploram os recursos de reconfigurabilidade do dispositivo (HERON, J.; TRAINOR, D.; WOODS, R., 1997; MOHD-YUSOF, Z.; SULEIMAN, I.; ASPAR, Z., 2000; CARNEIRO, C. A. et al., 2006).

1.3 Objetivos

O objetivo principal desta pesquisa consiste em propor e projetar uma arquitetura de *hardware* reconfigurável paralela dedicada para a implementação da SA-DCT, utilizando paralelismo e computação reconfigurável no processamento dos dados, possibilitando assim, aumentar o desempenho e a flexibilidade da arquitetura, bem como a tolerância a falhas no circuito digital.

Entre os objetivos intermediários, podem ser indicados:

- Estudar os algoritmos de codificação de textura orientada por objeto que podem seguir uma dentre duas abordagens: DCT baseada em blocos e SA-DCT;
- Estudar o estado da arte de computação reconfigurável e dos dispositivos reconfiguráveis;
- Realizar uma revisão bibliográfica dos diversos trabalhos desenvolvidos utilizando algoritmos de codificação de textura orientada por objeto em conjunto com técnicas de computação reconfigurável e paralelismo;
- Analisar as vantagens e desvantagens de se implementar a SA-DCT, usando técnicas de paralelismo e/ou computação reconfigurável em FPGAs ;
- Analisar o desempenho, a flexibilidade, a tolerância a falhas e o consumo de recursos lógicos na implementação da arquitetura proposta e projetada em dispositivos reconfiguráveis.

1.4 Metas

A meta principal desta dissertação é a proposição de uma arquitetura de *hardware* reconfigurável e paralela dedicada para a implementação da SA-DCT, utilizando paralelismo no processamento das colunas e das linhas de um mesmo bloco de borda. Espera-se que, com a utilização do paralelismo, obtenha-se um ganho de desempenho em relação às arquiteturas já existentes (GAUSE, J.; CHEUNG, P. Y. K.; LUK, W., 2000, 2004) e que, com a utilização da computação reconfigurável, aumente-se a flexibilidade e a tolerância a falhas, além de reduzir o

tempo de projeto e o custo da implementação. Para se atingir essa meta, é necessário que se alcancem também metas intermediárias que possibilitarão o desenvolvimento e a validação da arquitetura, listadas a seguir:

- Implementação da unidade da SA-DCT que demanda maior desempenho, utilizando uma linguagem de descrição de *hardware* - HDL (*Hardware Description Language*) (ASHENDEN, P. J., 1998);
- Teste, verificação e análise, usando simulação, da arquitetura da unidade da SA-DCT que demanda maior desempenho;
- Preparação e submissão de artigos em congressos nacionais, internacionais e em periódicos das diferentes áreas relacionadas à pesquisa.

1.5 Justificativa

A justificativa para o desenvolvimento desta pesquisa é a possibilidade de aumentar o desempenho, a flexibilidade e a tolerância a falhas no processo de codificação de imagens orientada por objeto adaptativa à forma.

A flexibilidade é uma característica muito importante na arquitetura da SA-DCT, visto que essa característica possibilitará, a qualquer momento, aumentar o grau de paralelismo do *harware* no processamento do algoritmo da SA-DCT. Isso é de fundamental importância na implementação dessa técnica de codificação, visto que a maioria das aplicações não utiliza o algoritmo da SA-DCT devido ao seu elevado tempo de processamento. A característica da reconfigurabilidade também permitirá aumentar a tolerância a falhas na arquitetura, pois se ocorrer algum problema em parte do circuito digital, o mesmo poderá ser implementado em outra área do dispositivo.

A relevância desta pesquisa é a consideração dos aspectos relacionados à implementação em *hardware*, utilizando paralelismo na implementação das técnicas de codificação de imagens por transformada, fato não observado em grande parte dos trabalhos desenvolvidos no Brasil. Na pesquisa bibliográfica realizada até a presente data, observou-se também que existem poucos trabalhos que utilizam técnicas de codificação de imagens por transformadas empregando conceitos de computação reconfigurável, principalmente no que diz respeito às técnicas de reconfiguração dinâmica e parcial (GAUSE, J.; CHEUNG, P. Y. K.; LUK, W., 2000, 2004). Na maioria das vezes, isso ocorre devido às limitações dos próprios dispositivos, que apresentam um tempo de reconfiguração elevado, e à ausência de ferramentas de projeto e *softwares* com custos mais acessíveis e com uma interface mais amigável.

1.6 Escopo

Nesta pesquisa, propõe-se uma arquitetura de *hardware* que implemente a SA-DCT, utilizando computação reconfigurável e paralelismo, para a codificação de imagens acromáticas. É importante destacar que não está sendo proposta uma nova técnica de codificação e sim uma nova arquitetura que implementa uma técnica já existente, visando a um aumento no desempenho do processamento da imagem, na flexibilidade e na tolerância a falhas.

Para validar a arquitetura proposta, optou-se por sintetizar e implementar a unidade da arquitetura que implementa as DCTs de tamanho variável em paralelo, num ambiente de projeto que permita mapear, rotear e gerar os arquivos de configuração, parciais ou totais. Optou-se por sintetizar essa unidade, por ser ela a que demanda maior tempo de processamento na implementação da SA-DCT e maior complexidade computacional. A verificação e a validação dessa unidade serão realizadas por sucessivas simulações e a sua implementação deverá apresentar características de independência do dispositivo utilizado (família, fabricante) e ser reconfigurável estaticamente. É importante mencionar que os conceitos e técnicas de computação reconfigurável propostos na implementação dessa unidade são os mesmos propostos na implementação das outras unidades.

Como embasamento teórico para a realização desta pesquisa, serão utilizados os conhecimentos de várias áreas, tais como processamento digital de imagens, projeto de sistemas digitais, arquitetura de computadores e computação reconfigurável.

1.7 Organização do texto

No Capítulo 2 desta dissertação, inicialmente, será apresentada uma revisão do estado da arte dos principais conceitos relacionados à teoria de compressão, bem como uma breve descrição dos principais padrões. Em seguida, serão abordadas a codificação de imagens baseada em blocos e as técnicas de codificação orientada por objeto. Feito isso, apresentam-se os principais conceitos de computação reconfigurável, dispositivos reconfiguráveis e linguagem de descrição de *hardware*. Finalmente, descrevem-se alguns trabalhos correlatos que tratam da implementação de técnicas de codificação de imagens em *hardware*.

No Capítulo 3, inicialmente, são descritos os requisitos esperados da arquitetura de *hardware* reconfigurável paralela dedicada para a implementação da SA-DCT. Em seguida, é apresentada a arquitetura proposta, descrevendo todas as unidades que a constituem. Por fim, são explicados o funcionamento e o tipo de codificação da arquitetura, bem como os dispositivos utilizados na sua implementação.

No Capítulo 4 é apresentada a verificação funcional da unidade da arquitetura da SA-DCT que demanda maior desempenho. Para isso, inicialmente, são definidas algumas configurações

e alguns tipos de blocos de borda utilizados para verificar o seu correto funcionamento. Em seguida, apresentam-se os resultados das simulações obtidos para cada bloco de borda, bem como uma verificação e validação de todos os coeficientes gerados por essa unidade. Feito isso, é realizada uma análise dos resultados de desempenho obtidos, comparando características como tempo de processamento e consumo de recursos lógicos de cada configuração dessa unidade para todos os tipos de blocos de borda utilizados. Por último, apresentam-se algumas representações estatísticas (agrupamentos de blocos de bordas nos quais se predominam determinadas características) que são utilizadas para avaliar o ganho real da utilização da computação reconfigurável e do paralelismo na arquitetura proposta.

No Capítulo 5 é apresentada a discussão dos resultados obtidos, comparando os possíveis ganhos alcançados com a utilização da arquitetura proposta aos ganhos obtidos nos trabalhos relacionados. Neste capítulo também comenta-se o alcance dos objetivos e das metas propostos nesta pesquisa. Em seguida, apresentam-se as principais contribuições desta dissertação e, por fim, são listados os possíveis trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo, inicialmente apresenta-se uma revisão do estado da arte dos principais conceitos relacionados à teoria de compressão, assim como uma breve descrição dos principais padrões existentes. Em seguida, na Seção 2.3, aborda-se a codificação de imagens baseada em blocos, bem como as técnicas de codificação orientada por objeto. Feito isso, apresentam-se os principais conceitos de computação reconfigurável, dispositivos reconfiguráveis e linguagem de descrição de *hardware*. Por fim, na última seção, descrevem-se alguns trabalhos correlatos que tratam da implementação de técnicas de codificação de imagens em *hardware*.

2.1 Compressão

O processo de compressão de imagens explora a redundância de informação existente nos sinais. Através desse processo, é possível reduzir a quantidade de dados necessária para representar uma imagem. Do ponto de vista matemático, isto corresponde a transformar uma matriz de pixels de duas dimensões num conjunto de dados estatisticamente descorrelacionado. Esse processo acontece antes do armazenamento ou transmissão da imagem, atendendo aos requisitos de faixa de passagem do meio de transmissão e de quantidade de memória de armazenamento (GONZALEZ, R. C.; WOODS, R. E., 1993).

No processo de compressão, são exploradas três redundâncias básicas de dados: a redundância de codificação, a redundância interpixels e a redundância psicovisual.

A redundância de codificação ocorre quando os pixels de uma imagem são codificados com mais bits de codificação do que o absolutamente necessário para representar cada pixel. Normalmente, esse fato ocorre quando os códigos atribuídos a um conjunto de eventos não foram escolhidos de forma a tirar toda a vantagem das probabilidades dos eventos.

A redundância interpixels está diretamente relacionada às correlações entre os pixels de uma imagem. Isso ocorre, porque o valor de qualquer pixel pode ser razoavelmente previsível a partir do valor dos seus vizinhos, uma vez que a informação carregada especificamente por cada pixel é relativamente pequena.

A redundância psico-visual consiste nas informações que visualmente não são relevantes. Essas informações podem ser eliminadas, porque a informação em si não é essencial para o processamento visual humano. O processo de eliminação desses dados resulta em uma perda de informação quantitativa (GONZALEZ, R. C.; WOODS, R. E., 1993).

No processo de compressão pode haver, ou não, perda de informação. Quando não há perda, esse processo é conhecido como compactação, caso contrário, o próprio termo “com-

pressão" é utilizado. As técnicas de compressão podem ser classificadas de acordo com diferentes critérios, entre os quais se destacam a capacidade de reconstrução da informação original, após a utilização de uma técnica de compressão, além de questões referentes à taxa de bits que pode ser constante (*Constant Bits Rate* - CBR) ou variável (*Variable Bits Rate* - VBR)

Quando existe a necessidade da reconstrução exata da informação original, as técnicas de compressão utilizadas são conhecidas como *lossless* (sem perda). Essas técnicas são normalmente utilizadas em programas de computador e em documentos médicos, onde a qualidade e a fidelidade da imagem são imprescindíveis. A técnica de compressão sem perda possui uma taxa de compressão baixa. Entende-se por taxa de compressão (C_R) a razão entre a quantidade de dados originais (n_1) e a quantidade de dados após a compressão (n_2), isto é:

$$C_R = \frac{n_1}{n_2} \tag{2.1}$$

Assim, uma taxa de compressão prática, tal como 50 (ou 50:1), significa dizer que o primeiro conjunto de dados (n_1) possui 50 bits para cada 1 bit no segundo conjunto (n_2). Nas compressões sem perda, as taxas normalmente são de 2:1 ou 3:1. Dentre as técnicas *lossless* mais utilizadas, podem ser citadas:

- Codificação de Comprimento de Corrida (*Run Length Encoding* - RLE) - é uma técnica de codificação de fonte discreta, cujos símbolos correspondem ao endereço inicial de cada sequência de 1s (ou 0s), seguidos do comprimento daquela sequência (JAIN, A., 1989).

- Codificação de Huffman - baseia-se na criação de um código ótimo para um conjunto de símbolos e respectivas probabilidades, sob a restrição de que os símbolos sejam codificados um por vez. Uma vez que o código tenha sido criado, a codificação e/ou decodificação é realizada através de "*look-up tables*"(HUFMAN, D. A., 1952).

- Codificação aritmética - baseia-se na construção de uma tabela onde são listadas as probabilidades do símbolo lido ser cada um dos possíveis símbolos. O algoritmo de codificação aritmética consiste em representar a probabilidade da ocorrência de cada caracter de acordo com intervalos cumulativos. Isto é, assume-se que a mensagem pertença ao intervalo [0,1] e nele identifica-se um sub-intervalo ao qual corresponde o primeiro símbolo lido. Para cada símbolo subseqüente, subdivide-se o intervalo atual em sub-intervalos proporcionais às probabilidades da tabela de intervalos, encontrando-se novamente o intervalo que corresponde ao próximo símbolo. No final do processo, obtém-se um intervalo que corresponde à probabilidade da ocorrência de todos os símbolos na ordem correta. Essa técnica elimina a associação entre símbolos individuais e palavras (códigos de comprimento inteiro) e, com isso, é capaz de praticamente igualar o número médio

de bits necessários para codificar um símbolo à entropia da fonte[1], em todos os casos (GONZALEZ, R. C.; WOODS, R. E., 1993).

- Algoritmo de Lempel-Ziv - associa códigos de comprimento fixo a palavras de comprimento variável, permitindo também reduzir dependências inter-pixels (ZIV, J.; LEMPEL, A., 1977). É um esquema patenteado, atualmente utilizado em GIF, TIFF e PDF.

Quando se deseja apenas uma boa aproximação da informação original, utilizam-se técnicas de compressão com perdas, também chamadas de *lossy*. Essas técnicas são normalmente utilizadas quando reduzir o tamanho da imagem é mais importante que obter qualidade na imagem, como em alguns tipos de compressão de áudio, imagens e vídeo digital. Essa técnica possui uma razão de compressão alta.

Dentre as técnicas *lossy*, destacam-se:

- *Pulse Code Modulation* (PCM) - consiste na aplicação da quantização escalar sobre os pixels da imagem.

- *Diferencial Pulse Code Modulation* (DPCM) - codificação preditiva, que realiza a quantização e a codificação dos erros de estimação.

- Codificação por transformada (foco deste trabalho) - utiliza-se uma transformada linear reversível para mapear um bloco da imagem em um bloco de coeficientes de transformadas, que são então quantizados e codificados. No caso da maioria das imagens, um número significativo de coeficientes tem pequenas magnitudes, podendo ser quantizados grosseiramente (ou descartados), produzindo pouca distorção na imagem (GONZALEZ, R. C.; WOODS, R. E., 1993).

- Codificação interquadros - a redundância entre quadros sucessivos, em uma seqüencia temporal de imagens, é reduzida pela utilização de uma abordagem de codificação previsora ou por transformada (ROESE, J. A.; PRATT, W. K.; ROBINSON, G. S., 1977).

- Codificação híbrida (HABIBI, A., 1974) - baseia-se na combinação de uma Transformada 2D (espacial) com uma predição (temporal), muito útil na compressão de imagens em movimento.

- Quantização vetorial (LINDE, Y.; BUZO, A.; GRAY, R. M., 1980) - a imagem é decomposta em vetores (contendo pixels ou coeficientes de transformadas) que são mapeados em um

[1] A entropia da fonte de símbolos define um número teórico mínimo de bits por símbolos necessários para codificar mensagens geradas por essa fonte, assumindo que os sucessivos símbolos emitidos da fonte são estatisticamente independentes.

dicionário finito de possíveis vetores, que posteriormente são codificados (GONZALEZ, R. C.; WOODS, R. E., 1993).

- Codificação em sub-bandas - a imagem é filtrada para produzir um conjunto de imagens sub-amostradas (com diferentes frequências espaciais) que podem ser individualmente codificadas por DPCM (WOODS, J. W.; O'NEIL, S. D., 1986).

A Figura 2.1 mostra um modelo de sistema de compressão genérico. Esse sistema consiste de dois blocos estruturais distintos: um codificador e um decodificador. Uma imagem de entrada é alimentada no codificador, que cria um conjunto de símbolos a partir desses dados. Depois da transmissão ao longo do canal, a representação codificada é alimentada no decodificador, onde uma imagem reconstruída é gerada (GONZALEZ, R. C.; WOODS, R. E., 1993). Dependendo de quais técnicas de compressão serão utilizadas, *lossless* ou *lossy*, a imagem reconstruída pode ou não ser uma réplica da imagem original.

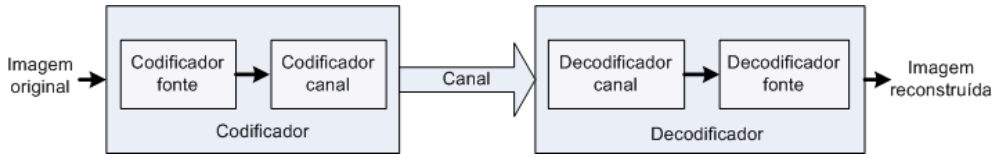


Figura 2.1: Modelo de sistema de compressão genérico

O codificador é composto por um codificador fonte, responsável pela redução ou eliminação das redundâncias psicovisual, interpixel e de codificação na imagem de entrada e por um codificador canal, responsável por aumentar a imunidade ao ruído de saída do codificador através da inserção de uma forma de redundância controlada nos dados codificados. Por sua vez, o decodificador inclui um decodificador canal seguido de um decodificador fonte. Se o canal for livre de ruídos, o codificador e o decodificador canal são omitidos (GONZALEZ, R. C.; WOODS, R. E., 1993). As Figuras 2.2 e 2.3, extraídas de (GONZALEZ, R. C.; WOODS, R. E., 1993), mostram os estágios de um codificador fonte e de um decodificador fonte, respectivamente.

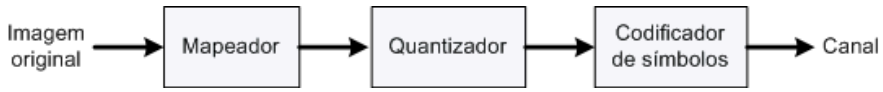


Figura 2.2: Codificador fonte - figura extraída de (GONZALEZ, R. C.; WOODS, R. E., 1993)

No primeiro estágio do processo de codificação fonte, o mapeador transforma os dados de entrada num formato projetado para reduzir as redundâncias interpixels nas imagens de entrada. Essa operação é geralmente reversível e pode ou não reduzir a quantidade de dados necessários para representar a imagem. A codificação por comprimento de corrida é um

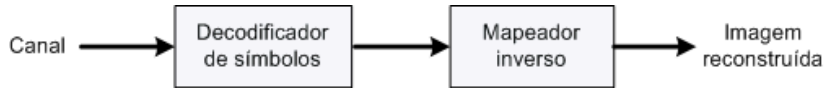


Figura 2.3: Decodificador fonte - figura extraída de (GONZALEZ, R. C.; WOODS, R. E., 1993)

exemplo de um mapeamento que resulta diretamente em compressão de dados neste estágio, ao contrário, por exemplo, da codificação por transformadas (GONZALEZ, R. C.; WOODS, R. E., 1993).

O quantizador reduz as redundâncias psicovisuais da imagem de entrada. Essa operação é irreversível, assim ela é omitida quando se deseja uma compressão livre de erros.

O codificador de símbolos cria um código de comprimento fixo ou variável para representar a saída do quantizador e mapeia a saída de acordo com o código, reduzindo assim as redundâncias de codificação. Essa operação, naturalmente, é reversível. Após completar o passo de codificação de símbolos, a imagem de entrada foi processada para remover cada uma das três redundâncias descritas anteriormente nessa seção.

O decodificador fonte mostrado na Figura 2.3 contém apenas o decodificador de símbolos e o mapeador inverso. Esses blocos desempenham, respectivamente, as operações inversas dos blocos codificador de símbolos e mapeador do codificador-fonte. Como a quantização resulta em perda irreversível de informação, um bloco quantizador inverso não é incluído no modelo de decodificador-fonte genérico (GONZALEZ, R. C.; WOODS, R. E., 1993).

2.2 Padrões de compressão de imagem

Conforme visto anteriormente, existe uma grande variedade de técnicas e possibilidades para se realizar a compressão de imagens, vídeos ou mídias. Com o objetivo de padronizar todas essas possibilidades, os órgãos ISO, ITU e IEC desenvolveram alguns padrões que pudessem ser utilizados mundialmente nos meios de comunicação, na indústria e no meio acadêmico. A seguir serão explicados, sucintamente, alguns dos principais padrões normalizados junto a esses órgãos. Será dado um enfoque maior ao padrão MPEG-4, por ser esse padrão foco desta pesquisa.

2.2.1 *Joint Pictures Expert Group* - JPEG

JPEG é um acrônimo de *Joint Photographic Experts Group*, que é o nome original do comitê que escreveu o padrão. O grupo foi organizado em 1986, emitindo um padrão em 1992, que foi aprovado em 1994 como ISO 10918-1, com o título "*Digital Compression and Coding of Continuous-tone Still Images*". Junto ao ITU, esse padrão possui a referência ITU-T

Recommendation T-81.

O padrão JPEG foi desenvolvido para comprimir imagens estáticas e de tom contínuo (como fotografias), não tendo um bom desempenho em imagens que apresentem descontinuidades. Esse padrão contempla técnicas de compressão com perdas e sem perdas. Os sistemas com perdas utilizam a DCT e são os mais empregados atualmente, devido à sua maior capacidade de compressão em relação ao sistemas de compressão sem perdas. Já a compressão sem perdas utiliza a codificação preditiva (DPCM), para formar os resíduos que são codificados com um código de comprimento variável: Huffman ou Aritmético.

2.2.2 H.261

Essa padronização foi projetada para aplicações de videoconferência, onde os vídeos são transmitidos por linhas T1, com atraso de transmissão de menos de 150 ms. A linha T1 foi introduzida pelo sistema Bell, para comunicações de voz digital, e é composta de vinte e quatro canais de 64 kbps multiplexados no tempo, amostrados e codificados em um sinal PCM de 1.544 Mbit/s.

A padronização H.261 estende a abordagem de compressão do tipo JPEG baseada em DCT, objetivando incluir métodos para redução de redundâncias inter-quadros. Essa padronização comprime um quadro inicial (ou de referência), usando uma abordagem do tipo JPEG baseada em DCT e reconstrói o quadro comprimido. Em seguida, estima o movimento dos objetos entre o quadro corrente e o quadro anterior reconstruído e decide, com base na quantidade de movimento detectada, pela compressão no modo *intraframe* (sem utilização de informação de quadros vizinhos já codificados) do quadro corrente ou no modo *interframe*, pela utilização desse tipo de informação (GONZALEZ, R. C.; WOODS, R. E., 1993).

2.2.3 *Moving Picture Experts Group* - MPEG

Os padrões MPEG são muito utilizados na compressão de vídeo e têm sido adotados como padrões internacionais desde 1993. Como os filmes contêm tanto sons quanto imagens, o MPEG pode compactar igualmente áudio e vídeo (TANENBAUM, A. S., 2003).

2.2.3.1 MPEG-1

O primeiro padrão do grupo MPEG a ser criado foi o MPEG-1 (ISO/IEC 11172). Seu objetivo era codificar áudio e vídeo para aplicações multimídia em CD-ROM, utilizando uma taxa máxima de 1,5 $Mbits/s$.

O MPEG-1 tem três partes: áudio, vídeo e sistema. Os codificadores de áudio e vídeo funcionam de maneira independente e sincronizam-se no receptor, com um *clock* de 90 kHz.

Esse padrão explora dois tipos de redundância: espacial e temporal. A redundância espacial pode ser reduzida pela simples codificação *intraframe* de cada quadro. Já a redundância temporal é reduzida na estimação do movimento entre quadros consecutivos. A saída do MPEG-1 consiste em quatro tipos de quadros:

- *Intraframe* (*I-Frame*) - quadro comprimido de forma independente dos outros quadros e codificado com o JPEG;
- *Predictive frame* (*P-Frame*) - diferença, bloco a bloco, entre o quadro atual e a sua previsão, baseada no quadro anterior;
- *Bidirectional frame* (*B-Frame*) - diferença entre o quadro atual e a sua previsão, baseada no quadro anterior e no quadro seguinte;
- *DC-coded* (*D-Frame*) - médias de blocos usadas para o avanço rápido. Cada entrada de quadro D é apenas o valor médio de um bloco, sem maiores codificações, o que facilita a exibição em tempo real.

O MPEG-1 foi projetado, considerando-se resoluções de vídeo da ordem de 360 x 280 pixels a uma taxa de 30 quadros por segundo, o que era insuficiente para muitas aplicações. Com isso, tornou-se necessário buscar um aperfeiçoamento do MPEG-1, resultando, em 1994, em um novo padrão, conhecido como MPEG-2.

2.2.3.2 MPEG-2

O padrão MPEG-2 (ISO/IEC 13818) foi projetado originalmente para compactar vídeos com qualidade de difusão em uma faixa de 4 a 6 $Mbits/s$, sendo transmitido em um canal de difusão do tipo *National Television System(s) Committee* (NTSC) ou *Phase Alternate Lines* (PAL) . Mais tarde, o MPEG-2 foi expandido para aceitar resoluções mais altas, incluindo a tecnologia HDTV. Atualmente, ele forma a base para o DVD e para a televisão digital por satélite (TANENBAUM, A. S., 2003).

A codificação MPEG-2 é semelhante à codificação MPEG-1, com quadros I, P e B, entretanto essa codificação não possui os quadros D. Além disso, a DCT utiliza um bloco de 10x10, em lugar de um bloco de 8x8. O MPEG-2 aceita tanto imagens progressivas quanto entrelaçadas, enquanto o MPEG-1 admite somente imagens progressivas. O MPEG-2 utiliza quatro níveis de resolução (ao contrário do MPEG-1 que aceita apenas 1): baixa (352x240), principal (720x480), alta 1440 (1440x1152) e alta (1920x1080). A resolução baixa destina-se aos aparelhos de videocassete, de forma a manter compatibilidade retroativa com o MPEG-1. A resolução principal se destina aos sistemas de difusões PAL e NTSC e as outras duas são destinadas à HDTV (TANENBAUM, A. S., 2003).

2.2.3.3 MPEG-4

O padrão MPEG-4 (ISO/IEC 14496) foi finalizado em outubro de 1998 e tornou-se um padrão internacional, no início de 1999. O MPEG-4 foi desenvolvido para aplicações multimídias e tem, como meta principal, a integração da produção, da distribuição e do acesso ao conteúdo multimídia. Esse padrão combina as características típicas dos outros padrões MPEG com a representação e a manipulação individual dos objetos audiovisuais em uma cena. Os objetos audiovisuais podem ser naturais (capturados a partir de câmeras e microfones) ou sintéticos (gerados por computador, como gráficos, animações e música computacional) e, no caso específico dos objetos visuais, 2D ou 3D.

Com o objetivo de facilitar o desenvolvimento de novas formas baseadas em conteúdo de comunicação, de acesso e de manipulação de dados audiovisuais digitais, o grupo MPEG identificou oito funcionalidades e as agrupou em três classes (PEREIRA, F.; EBRAHIMI, T., 2002), conforme descrito a seguir:

1. Interatividade baseada em conteúdo:

   (a) Acesso e organização eficientes de dados audiovisuais, através de ferramentas para indexação, busca, navegação, carga, descarga e deleção.

   (b) Fornecimento de sintaxe e métodos de codificação que permitam manipular conteúdo, sem a necessidade de transcodificação.

   (c) Codificação de dados híbridos (sintéticos e naturais).

   (d) Acesso aleatório, dentro de um intervalo de tempo e com granulação fina, a partes de uma seqüência audiovisual (quadros de vídeo, por exemplo).

2. Compressão eficiente:

   (a) Codificação eficiente para a transmissão de dados audiovisuais, em redes com restrições de faixa de passagem e de memória de armazenamento.

   (b) Codificação de múltiplas visões e trilhas sonoras de uma mesma cena.

3. Acesso Universal:

   (a) Robustez em ambientes que trabalham com baixa taxa de bits, sob condições de erro severas, como por exemplo, redes sem fio. Nesse cenário, sistemas MPEG-4 devem fornecer recursos de tolerância a erros.

   (b) Capacidade de alcançar escalabilidade em conteúdo, resolução espacial, resolução temporal, qualidade e complexidade.

O padrão MPEG-4 foi dividido em dez partes, sendo algumas padrões internacionais e outras, relatórios técnicos (PEREIRA, F.; EBRAHIMI, T., 2002). Essas partes são descritas a seguir:

- Parte 1 (ISO/IEC 14496-1) - Sistemas. Especifica a descrição de cenas, multiplexação, sincronização, armazenamento local temporário (*buffer*), gerenciamento e proteção de propriedades intelectuais.
- Parte 2 (ISO/IEC 14496-2) - Visual (foco deste trabalho). Especifica a codificação, decodificação e manipulação, de forma independente, dos objetos visuais naturais e sintéticos, que são caracterizados por sua forma, textura e movimento.
- Parte 3 (ISO/IEC 14496-3) - Áudio. Especifica a codificação de objetos de áudio naturais e sintéticos.
- Parte 4 (ISO/IEC 14496-4) - Teste de Compatibilidade. Define as condições para testar as implementações de ferramentas MPEG-4.
- Parte 5 (ISO/IEC 14496-5) - *Software* de Referência. Informações para auxiliar o desenvolvimento de ferramentas reais compatíveis com o padrão.
- Parte 6 (ISO/IEC 14496-6) - *Delivery Multimedia Integration Framework* (DMIF). Define um protocolo de sessão para gerenciamento do fluxo multimídia sobre as tecnologias de distribuição.
- Parte 7 (ISO/IEC 14496-7) - *Software* de Referência Visual Otimizado (relatório técnico). Consiste em um *software* otimizado para ferramentas visuais como, estimação de movimento rápida e estimativa de movimento global rápida.
- Parte 8 (ISO/IEC 14496-8) - Transporte de Conteúdo MPEG-4 sobre Redes IP. Especifica o mapeamento do conteúdo MPEG-4, para protocolos baseados em IP.
- Parte 9 (ISO/IEC 14496-9) - Descrição de *Hardware* de Referência (relatório técnico). Inclui descrições de ferramentas MPEG-4 em linguagem de descrição de *hardware*, facilitando o desenvolvimento de ferramentas baseadas em dispositivos reconfiguráveis (assunto deste trabalho).
- Parte 10 (ISO/IEC 14496-10) - Codificação de Vídeo Avançada. Especifica a sintaxe de vídeo e ferramentas de codificação baseadas na especificação H.264 da ITU-T.

2.2.3.4 MPEG-7

O padrão MPEG-7 (ISO/IEC 15938) visa a padronizar o modo de descrever o conteúdo audiovisual (documentos apenas de texto não fazem parte dos objetivos desse padrão). Isso é feito através da especificação de um conjunto de descritores (*Descriptors* - Ds) que são usados na representação dos diversos tipos de informação multimídia. A definição de um novo descritor ou esquema de descrição é feita usando uma linguagem específica, chamada de linguagem de definição de descrição (*Description Definition Language* - DDL). Uma descrição MPEG-7 pode ter duas formas: textual em XML (*eXtensible Markup Language*), adequada para busca, edição e filtragem de dados, ou binária, adequada para armazenamento e transmissão (MARTÍNEZ, J. M., 2004).

Os descritores MPEG-7 são usados para compor uma descrição das características do objeto. Tais características incluem informações de índice (título, autor etc), de arquivo (formato, tipo de compressão etc) e informações sobre o significado semântico do objeto (personagens, objetos ou local da cena). Esses descritores não dependem da maneira com que o conteúdo descrito foi codificado ou armazenado. Existem várias aplicações que podem se beneficiar dos recursos fornecidos pelo padrão MPEG-7, entre elas podem ser destacadas: as bibliotecas digitais e os sistemas de informação geográfica. É importante ressaltar que não faz parte dos objetivos do MPEG-7 desenvolver algoritmos e ferramentas de busca (MARTÍNEZ, J. M., 2004).

2.2.3.5 MPEG-21

O MPEG-21 (ISO/IEC 21000) é o padrão mais recente do grupo MPEG e ainda é objeto de estudos e propostas. Esse padrão visa a definir a tecnologia necessária para dar suporte a usuários que desejam acessar, consumir, alterar ou manipular itens digitais de forma eficiente, transparente e interoperável. O MPEG-21 é baseado em dois conceitos essenciais: a definição de uma unidade fundamental de distribuição e transação (*Digital Item*) e o conceito dos Usuários (*Users*) que interagem ou fazem uso dessa unidade (BORMANS, J.; HILL, K., 2002).

2.3 Codificação orientada por objeto

A codificação orientada por objeto é uma das técnicas mais importantes utilizadas pelo padrão MPEG-4 parte 2. Conforme mencionado na seção anterior, esse padrão manipula, de forma independente, os objetos de uma cena de vídeo (*Video Objects* - VOs), caracterizados por sua forma, textura e movimento.

Nessa codificação, cada objeto da imagem é inscrito em uma região retangular, que é então particionada em blocos. Existem três tipos diferentes de blocos: os que não possuem ne-

nhum pixel do objeto, chamados de blocos transparentes, os que estão situados completamente dentro do objeto, conhecidos como blocos do objeto e os que pertencem parcialmente ao objeto, chamados blocos de borda (AHMED, N.; NATRAJAN, T.; RAO, K. R., 1984).

No processo de codificação, os blocos transparentes podem ser omitidos e os blocos do objeto são codificados utilizando a técnica 2D-DCT (AHMED, N.; NATRAJAN, T.; RAO, K. R., 1984) convencional. Já os blocos de borda, entretanto, utilizam uma técnica de codificação diferenciada. Conforme mencionado no Capítulo 1, existem basicamente dois modos de codificar esses blocos:

1. A região externa ao objeto no bloco de borda é totalmente preenchida com o uso de técnicas de extrapolação (FRANKE, U.; MESTER, R.; AACH, T., 1988; KAUP, A.; AACH, T., 1994; KAUP, A., 1999), sendo o bloco resultante codificado com a 2D-DCT padrão (AHMED, N.; NATRAJAN, T.; RAO, K. R., 1984);

2. O bloco de borda é codificado usando a SA-DCT (SIKORA, T.; MAKAI, B., 1995).

Nas subseções seguintes, serão explicadas a codificação 2D-DCT padrão e as duas técnicas utilizadas normalmente para codificar os blocos de borda, as técnicas de extrapolação e a SA-DCT, embora o foco desta pesquisa seja apenas esta última.

2.3.1 *Bidimensional Discrete Cosine Transform* (2D-DCT)

A 2D-DCT (AHMED, N.; NATRAJAN, T.; RAO, K. R., 1984) é uma das principais técnicas de codificação utilizada pelos padrões de compressão de imagens. A DCT converte o sinal do domínio espacial para o domínio das freqüências espaciais, com a aplicação de algumas operações matemáticas sobre os dados. Com o objetivo de reduzir a complexidade e o tempo de processamento da DCT, essa técnica normalmente divide a imagem em blocos de NxN pixels (geralmente de 8x8), conduzindo a uma transformada DCT também de NxN pixels. Cada pixel contém uma informação numérica que corresponde ao valor da sua textura. Para imagens acromáticas (sem cor), essa informação corresponderia ao valor da luminância (JAIN, A., 1989). A Figura 2.4 ilustra o particionamento da imagem em blocos de $8x8$ pixels.

Em uma imagem, os pixels adjacentes, geralmente, apresentam alta correlação e por isso, após a transformação, a maior parte da informação fica concentrada nos coeficientes de baixa freqüência (AHMED, N.; NATRAJAN, T.; RAO, K. R., 1984). Isso permite descartar ou realizar uma quantização mais grosseira sobre os coeficientes que apresentarem pouca informação, principalmente sobre aqueles correspondentes às altas freqüências espaciais, para as quais o sistema visual humano é menos sensível (JAIN, A., 1989). Desse modo, a 2D-DCT torna-se uma importante técnica utilizada para minimizar a demanda de recursos computacionais necessários à

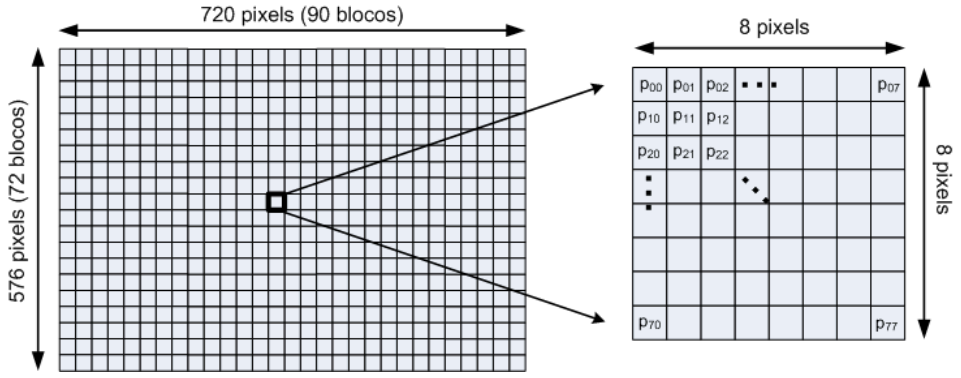


Figura 2.4: Imagem particionada em blocos de 8*x*8 pixels

manipulação digital das imagens. A Figura 2.5 mostra a localização dos coeficientes resultantes de uma transformada 2D-DCT sobre um bloco de tamanho 8x8 de uma imagem. Nesta figura, é importante ressaltar a concentração da energia no coeficiente DC (primeiro coeficiente) e a variação das frequências espaciais ao longo do bloco (quanto maior o índice das colunas, maior a freqüência horizontal, quanto maior o índice das linhas maior a freqüência vertical).

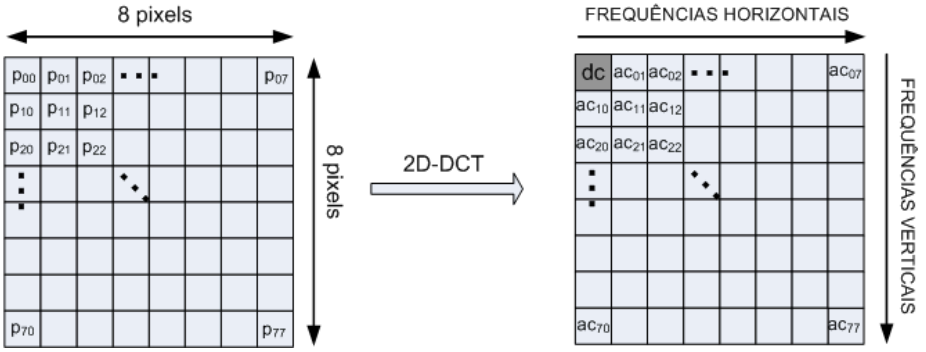


Figura 2.5: Localização dos coeficientes resultantes da 2D-DCT

A equação utilizada para o cálculo da 2D-DCT está apresentada abaixo:

$$T_{ij} = c_{ij} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} V_{yx} \cos \frac{(2y+1)i\pi}{2*N} \cos \frac{(2x+1)j\pi}{2*N}, \quad em \ \ que \ \ 0 \leq i,j \leq N-1 \qquad (2.2)$$

onde T_{ij} representa o coeficiente no domínio das freqüências espaciais, localizado na linha i e coluna j do bloco, N representa a dimensão do bloco, V_{yx} é o componente na linha y e coluna x da matriz de entrada e c_{ij} é dado por:

$$c_{ij} = \begin{cases} \sqrt{\frac{1}{N}} & se \quad i = 0 \\ \\ \sqrt{\frac{2}{N}} & se \quad i \neq 0 \end{cases} \tag{2.3}$$

Para diminuir a complexidade computacional do cálculo da 2D-DCT, é necessária a utilização de algoritmos rápidos para sua implementação. A contribuição mais significativa nesse sentido foi dada por (ARAI, Y.; AGUI, T.; NAKAJIMA, M., 1988). Essa contribuição explora a propriedade da separabilidade da 2D-DCT, isso é, o cálculo pode ser realizado, aplicando-se primeiro a DCT unidimensional (1D-DCT) sobre as colunas da matriz NxN de entrada e, a seguir, calculando-se novamente a 1D-DCT sobre as linhas da matriz resultante do primeiro processamento. Essa propriedade é muito empregada em implementações da 2D-DCT em *hardware*, uma vez que o número de operações e a quantidade de *hardware* utilizado é menor do que nas implementações em que a 2D-DCT é calculada diretamente. A 1D-DCT é dada por:

$$T_i = c_i \sum_{x=0}^{N-1} V_x \cos \frac{(2x+1)i\pi}{2N} \tag{2.4}$$

onde T_i representa os coeficientes no domínio das freqüências espaciais, resultantes da 1D-DCT, V_x representa a coluna de dados e c_i é dado por:

$$c_i = \begin{cases} \sqrt{\frac{1}{N}} & se \quad i = 0 \\ \\ \sqrt{\frac{2}{N}} & se \quad i \neq 0 \end{cases} \tag{2.5}$$

2.3.2 Métodos de extrapolação

O método mais simples de extrapolação consiste no preenchimento dos pixels externos ao objeto nos blocos de borda com zeros. Nas codificações *interframes*, que usam informações de outros quadros em uma seqüência para prever um quadro a partir dos seus vizinhos, esse método resulta em uma taxa de distorção razoável, considerando a sua baixa complexidade. No caso de codificação *intraframes*, que usam informação apenas do quadro corrente, esse método resulta em descontinuidades nas bordas do objeto, devido à grande diferença entre as intensidades dos tons de cinza presentes na imagem acromática original e o tom utilizado no preenchimento. Isso resulta numa baixa concentração de energia e num grande número de coeficientes de altas freqüências, após a aplicação da 2D-DCT (BOON, C. S.; TAKAHASHI, J.; KADONO, S., 1996).

As descontinuidades nas bordas podem ser reduzidas, ao se utilizar o algoritmo *Simplified Expanded Arbitrary-Shaped DCT* (SEA-DCT) (XIE, T.; WENIG, C.; HE, Y., 1997), que con-

siste em atribuir aos pixels externos ao objeto o valor médio dos pixels do bloco pertencentes à região segmentada.

Franke (FRANKE, U.; MESTER, R.; AACH, T., 1988), em 1988, também propôs um método de extrapolação que resulta numa grande concentração de energia em poucos coeficientes DCT. Esse trabalho foi generalizado para transformadas arbitrárias em (KAUP, A.; AACH, T., 1994) e um algoritmo numérico simplificado foi proposto para a estimação dos coeficientes de transformadas ótimos. A desvantagem dos métodos propostos por (FRANKE, U.; MESTER, R.; AACH, T., 1988; KAUP, A.; AACH, T., 1994) é que eles envolvem operações nos domínios espaciais e das freqüências, gerando um aumento significativo da carga computacional.

Com o objetivo de obter um método de extrapolação computacionalmente mais simples, com o mínimo de descontinuidades nas bordas, Kaup em 1999 (KAUP, A., 1999) propôs uma função que gera valores de pixels de preenchimento próximos, ao mesmo tempo, dos valores da borda e dos valores dos pixels do objeto. Esse método recebeu o nome de *Low Pass Extrapolation* (LPE). A operação realizada por (KAUP, A., 1999) para calcular o valor do pixel de preenchimento em cada ponto é dada por:

$$f^{(k+1)}(i,j) = f^{(k)}(i,j) + \frac{1}{4}\Delta f^{(k)}(i,j) \tag{2.6}$$

em que

$$\Delta f^{(k)} = f^{(k)}(i,j-1) + f^{(k)}(i-1,j) + f^{(k)}(i,j+1) + f^{(k)}(i+1,j) - 4f^{(k)}(i,j) \tag{2.7}$$

sendo k = momento presente e k+1 = momento futuro.

Uma vez que cada pixel da região de preenchimento é continuamente substituído pelo valor médio dos seus vizinhos, a informação do nível de cinza do objeto da imagem rapidamente se propaga dentro da área a ser preenchida e as descontinuidades das bordas dos segmentos da imagem são suavizadas. Se um ou mais dos quatro pixels da vizinhança estão fora do bloco da imagem, esses pixels são desconsiderados para a operação da média e os fatores 1/4 e 4 são modificados, proporcionalmente, em função disso. Nas codificações *intraframes*, ao se comparar a LPE com o preenchimento com zeros, observa-se um ganho na taxa de distorção entre 0,5 e 1 dB (KAUP, A., 1999).

Com o objetivo de minimizar o aumento do erro médio quadrático gerado pela inclusão dos dados de preenchimento, tenta-se delimitar o bloco da imagem num retângulo tão pequeno quanto possível. Isso reduz a quantidade de coeficientes a serem codificados. Entretanto, esse retângulo tem que ser recalculado no receptor, o que aumenta a complexidade computacional no decodificador.

2.3.3 *Shape Adaptive - DCT* (SA-DCT)

A SA-DCT foi desenvolvida por Sikora e Makai (SIKORA, T.; MAKAI, B., 1995), em 1995, com o objetivo de codificar objetos de forma arbitrária. Essa transformada é baseada em um conjunto pré-definido de funções-base 1D-DCT separáveis e representa um bom compromisso entre complexidade de implementação, eficiência de codificação e compatibilidade com as técnicas DCT existentes. A implementação em *hardware* da SA-DCT, para a codificação de objetos de forma arbitrária, é mais complexa e possui uma demanda computacional maior do que a implementação da 2D-DCT padrão. Entretanto, a SA-DCT normalmente apresenta uma PSNR significativamente melhor, além de apresentar funcionalidades relativas à manipulação independente dos objetos. Na 2D-DCT, em um bloco de imagem NxN (em que N representa a dimensão do bloco), a transformada é sempre aplicada sobre o conjunto dos pixels de cada coluna e, em seguida, sobre os coeficientes de cada linha resultantes do processamento anterior, ou vice-versa. Já na SA-DCT, a dimensão da transformada a ser aplicada depende do número de pixels do objeto presentes em cada linha e em cada coluna, apresentando uma maior complexidade, devido à não homogeneidade, e demandando mais recursos de *hardware*.

No algoritmo da SA-DCT, a imagem é separada em blocos NxN adjacentes e somente os blocos inteiramente contidos em um objeto ou região segmentada são codificados, usando-se a 2D-DCT padrão. Os blocos de borda de dimensão NxN são transformados em segmentos verticais de comprimento variável L ($1 \leq L \leq N$), que contêm apenas pixels pertencentes ao objeto. Esses segmentos verticais são deslocados em direção a borda superior do bloco, em seguida, uma transformada unidimensional de tamanho L pré-definida é aplicada a cada segmento. Num segundo estágio, os coeficientes verticais resultantes são alinhados pelo índice, isto é, todos os segmentos verticais são deslocados em direção a borda esquerda do bloco, resultando em segmentos horizontais de tamanho variável L. Esses segmentos, finalmente, são processados na direção horizontal, aplicando-se as transformadas unidimensionais de tamanho L, mencionadas anteriormente, sobre cada um deles (SIKORA, T.; MAKAI, B., 1995).

A Figura 2.6 ilustra as transformações da região de suporte no processamento da SA-DCT: (a) bloco de borda original, com sete segmentos verticais de tamanhos 1, 2, 2, 5, 4, 5 e 3, respectivamente; (b) alinhamento dos segmentos verticais com a borda superior do bloco, para o primeiro processamento unidimensional da DCT na direção vertical; (c) alinhamento dos coeficientes resultantes do primeiro processamento com a borda esquerda, para posterior processamento unidimensional na direção horizontal. É importante salientar que o número de coeficientes resultantes da SA-DCT equivale ao número de pixels do objeto presentes no bloco.

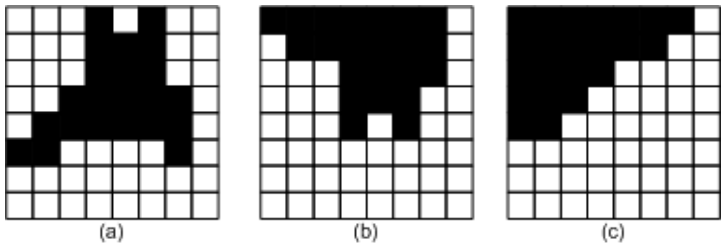


Figura 2.6: Transformações da região de suporte no processamento da SA-DCT

2.4 Computação reconfigurável

Atualmente, observa-se a existência de uma grande quantidade de problemas complexos que geram uma grande demanda de recursos computacionais necessários no armazenamento, recuperação, transmissão e processamento de informações. Para resolver esses problemas, normalmente utilizam-se dois tipos de soluções: as soluções implementadas em *Hardware* Fixo (HF), ou *Hardwired*, e as soluções implementadas em *Hardware* Programável (HP), através de *Software* (SW). Entretanto, ainda se observam alguns problemas relacionados a desempenho, flexibilidade, custo e tolerância a falhas (MARTINS, C. A. P. S. et al., 2003).

Nesse contexto, a computação reconfigurável surge como uma solução intermediária que tenta reduzir ou eliminar as deficiências das soluções que usam o paradigma de *hardware* fixo e as deficiências das soluções que usam o paradigma de *hardware* programável (MARTINS, C. A. P. S. et al., 2003). Ela possibilita a obtenção de uma maior flexibilidade, em relação ao paradigma de *hardware* fixo, e um maior desempenho, em relação ao paradigma de *software* (HP+SW), além de proporcionar outros fatores de motivação importantes, como uma significativa melhora de eficiência, custo, generalidade e tolerância a falhas no circuito digital (COMPTON, K.; HAUCK, S., 2002).

As arquiteturas reconfiguráveis são formadas por blocos (módulos) lógicos que constituem as unidades funcionais de processamento, armazenamento, comunicação ou entrada e saída de dados. Essas arquiteturas podem ser puramente reconfiguráveis ou híbridas (mistas), sendo que essas últimas utilizam os modelos de *hardware* fixo e/ou *hardware* programável, em conjunto com o modelo de *hardware* reconfigurável.

Entre as principais características das arquiteturas reconfiguráveis, podem ser destacadas:

- Granularidade fina, média ou grossa;
- Arranjo unidimensional, bidimensional, *pipeline* ou *crossbar*;
- Capacidade de realizar uma ou várias configurações;

- Reconfiguração estática ou dinâmica, parcial ou total, local ou remota;
- Configuração em tempo de compilação ou em tempo de execução;
- Modelo de computação monoprocessador ou multiprocessador, *Single Instruction Stream Multiple Data Stream* (SIMD) ou *Multiple Instruction Stream Multiple Data Stream* (MIMD);
- Propósito geral ou específico;
- Modelo de implementação de solução reconfigurável, mista, fixa ou programável.

A granularidade está relacionada à diferença, em termos de simplicidade/complexidade, do dispositivo reconfigurável. Dispositivos com granularidade fina são indicados para aplicações de manipulação no nível dos bits (*lookup table - LUT*), enquanto os de granularidade grossa são indicados para aplicações que envolvem computações mais complexas, como manipulações de imagens e caminho de dados com largura de vários bits (TODMAN, T. J. et al., 2005). O arranjo unidimensional, bidimensional, *pipeline* ou *crossbar* está relacionado à estrutura ou topologia dos blocos construtivos reconfiguráveis.

A reconfiguração estática está relacionada a configuração do dispositivo antes desse começar a fazer a computação dos dados e a manutenção do seu estado e funcionalidades, enquanto se necessite dessas características. Caso seja necessário reconfigurar, para que se tenha outra funcionalidade, o dispositivo pára de executar as operações e é reconfigurado, para depois voltar a executar novamente as operações de computação. Já a reconfiguração dinâmica acontece, quando o dispositivo necessita de alguma nova funcionalidade, reconfigurando uma área que não esteja processando nenhum dado, realizando-se uma reconfiguração parcial, paralelamente à execução das respectivas operações de computação. Esse tipo de reconfiguração também é conhecido como *on-the-fly* (HAUCK, S.; DEHON, A., 2008). Dispositivos da família Virtex da Xilinx (XILINX, 2008) e AT40K da Atmel (ATMEL, 2007) apresentam essas características, sendo os dispositivos Xilinx Virtex os mais utilizados na implementação de Sistemas Dinamicamente Reconfiguráveis (SDRs), principalmente pela disponibilização de ferramentas de suporte ao projeto. A Figura 2.7 mostra um sistema dinamicamente reconfigurável.

Nas arquiteturas dos sistemas computacionais reconfiguráveis, muitos ou quase todos os conceitos e níveis de abstrações arquiteturais tradicionais (não reconfiguráveis), como algoritmos, linguagens, compiladores, sistemas operacionais, arquiteturas e microarquiteturas podem continuar existindo. Entretanto, uma arquitetura reconfigurável organiza e implementa a computação de maneira diferente. Ao invés de processar uma função através de um conjunto de instruções executadas seqüencialmente ao longo do tempo, como em um processador, as arquiteturas reconfiguráveis geralmente processam a função através de unidades funcionais con-

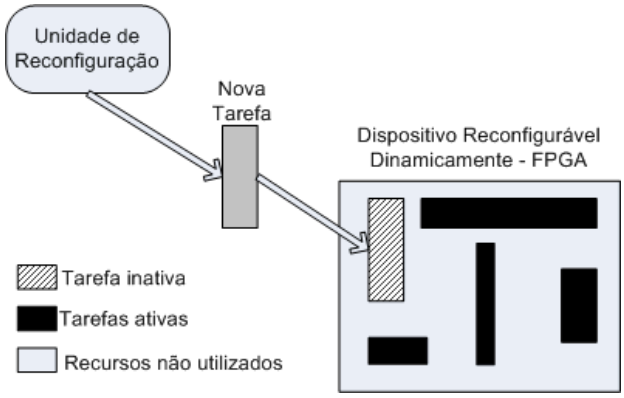


Figura 2.7: Sistema dinamicamente reconfigurável

figuradas no espaço (unidades lógicas mapeadas em diferentes blocos construtivos básicos, dentro dos dispositivos reconfiguráveis, como os FPGAs). Tem-se, portanto, computação paralela, envolvendo diferentes unidades funcionais que geram e consomem resultados intermediários. A diferença, portanto, está na computação temporal/seqüencial das operações ou tarefas *versus* computação espacial/paralela das operações ou tarefas (MARTINS, C. A. P. S. et al., 2003).

Em termos mais específicos, o alvo dos principais sistemas reconfiguráveis são as aplicações que apresentam inerente paralelismo de dados, alta regularidade e grandes requisitos de vazão. Alguns exemplos dessas aplicações são: compressão de vídeo, processamento gráfico e de imagens, multimídia e encriptação de dados (MARTINS, C. A. P. S. et al., 2003).

2.4.1 Dispositivos reconfiguráveis

Os *Field Programmable Gate Arrays* (FPGAs) são dispositivos programáveis em campo, ou seja, podem ter sua configuração alterada sem que tenham que ser retirados do circuito ou equipamento. Esses dispositivos são utilizados na implementação dos sistemas computacionais reconfiguráveis. A capacidade atual dos FPGAs está na faixa dos milhões de portas lógicas (HAUCK, S.; DEHON, A., 2008).

Normalmente, os FPGAs são compostos de uma matriz de elementos reconfiguráveis. Cada elemento deve ser primeiro configurado, antes de ser utilizado para realizar alguma computação (operação). Para tal, existem os bits de configuração (*bitstreams*) dos dispositivos, que o configuram e o reconfiguram, determinando a função que o mesmo irá desempenhar, a partir do momento em que é configurado ou reconfigurado (COMPTON, K.; HAUCK, S., 2002). No momento da configuração, são especificadas, além da função que cada elemento reconfigurável irá desempenhar, as portas de entrada ou de saída de cada elemento da matriz. Essa configuração das portas gera um outro tipo de configuração, que é a de roteamento dos dados. Esse roteamento é de grande importância para o desempenho e utilização do dispositivo reconfigu-

rável, pois quanto melhor o roteamento, melhor a utilização da área do dispositivo e melhor o desempenho conseguido na execução das funções configuradas (TODMAN, T. J. et al., 2005).

Um FPGA típico possui uma arquitetura interna composta de uma matriz de blocos lógicos configuráveis (*Configurable Logic Blocks* - CLBs), cercados por uma rede de interconexão programável, formada de blocos de interconexão. Circundando todo o circuito, existem os blocos de entrada e saída (*Input Output Blocks* - IOBs), que também são programáveis e que servem como interface entre o mundo exterior e a lógica interna (TODMAN, T. J. et al., 2005). Atualmente, também existem nestes dispositivos uma área de memória interna (*block* RAM) e os blocos de gerenciamento de *clock* (*Digital Clock Manager* - DCM). Os blocos de gerenciamento de *clock* fornecem calibração e soluções digitais para a distribuição, a multiplicação, a divisão e o deslocamento de fase para sinais de *clock* (XILINX, 2008). A Figura 2.8, retirada de (XILINX, 2008), mostra a arquitetura interna de um FPGA.

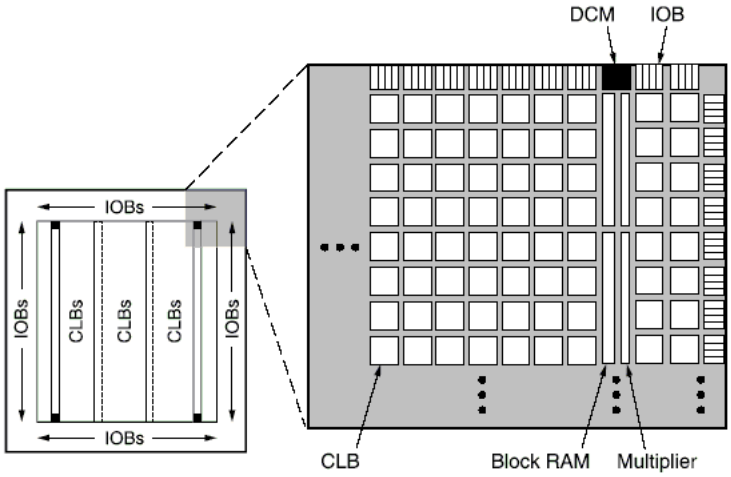


Figura 2.8: Arquitetura interna de um FPGA (XILINX, 2008)

A arquitetura de um CLB varia de família para família e de fabricante para fabricante, mas, basicamente, é composta de pontos de entrada que se conectam a blocos que implementam funções puramente combinacionais (*Lookup tables* - LUTs), a multiplexadores, que direcionam o fluxo dos sinais internamente ao CLB, e a registradores (tipicamente *flip-flops*), que estão ligados às saídas e também podem realimentar as entradas dos geradores de funções combinacionais. Todos os elementos são reconfiguráveis e propiciam uma grande flexibilidade para a implementação de funções. A rede de interconexão programável é composta de diferentes tipos de segmentos de conexão, capazes de interligar a maioria das entradas e saídas dos CLBs entre si e aos IOBs. Isso tudo permite que circuitos complexos, máquinas de estado e algoritmos sejam implementados nos FPGAs (MARTINS, C. A. P. S. et al., 2003).

Entre os dispositivos reconfiguráveis comerciais mais utilizados no Brasil, destacam-se

os fabricados pelas empresas: Xilinx (XILINX, 2008), Altera (ALTERA, 2008), Actel (ACTEL, 2008) e Atmel (ATMEL, 2007).

2.5 Linguagem de descrição de *hardware* - VHDL

O termo VHDL consiste num acrônimo das seguintes siglas: VHSIC + HDL, em que VHSIC significa *Very High Speed Integrated Circuits* e a sigla HDL significa *Hardware Description Language* (ASHENDEN, P. J., 1998). A idéia da criação de uma linguagem de descrição de *hardware* partiu do Departamento de Defesa dos Estados Unidos da América (DoD) na década de 80. Nessa época, os projetos eletrônicos eram de grande importância militar e existiam dezenas de fornecedores envolvidos, assim o DoD estava preocupado com as questões de portabilidade, documentação e compreensibilidade dos projetos. Nesse contexto, a linguagem VHDL surgiu para descrever e documentar de forma eficiente o circuito, possibilitando a todos os fornecedores entender o funcionamento de suas partes, padronizando a comunicação. Assim, inicialmente essa linguagem substituiu os complexos manuais que descreviam o funcionamento dos ASICs, facilitando o entendimento dos diagramas esquemáticos. Mais tarde, o VHDL também passou a ser utilizado na descrição, síntese, simulação, teste, verificação formal e, em alguns casos, compilação do software (ASHENDEN, P. J., 1998).

Um código em VHDL pode ser escrito usando, basicamente, dois modelos de descrição: estrutural e comportamental (SMITH, D. J., 1997). No modelo estrutural, a organização física e topológica do sistema é descrita, portanto é necessário o conhecimento detalhado do projeto do *hardware*. Isso quer dizer que são especificadas as entradas e saídas, os componentes lógicos, a interligação deles e os sinais que compõem o sistema. Existem bibliotecas em VHDL que contêm entidades que podem ser usadas nos projetos, como somadores, contadores e multiplicadores. Já no modelo comportamental, não é necessário descrever a organização física e topológica do sistema, somente o comportamento, isto é, as suas funções. Um programa que utiliza esse tipo de descrição possui o mesmo formato de um programa fonte escrito numa linguagem de programação de alto nível, como C++. Essa abordagem diminui a necessidade de conhecer o projeto do *hardware*, aumentando a facilidade de desenvolvimento do sistema. No entanto, os sistemas gerados a partir desse tipo de descrição podem não ser tão otimizados em questões de desempenho e consumo de recursos lógicos, quanto os sistemas descritos em VHDL estrutural.

A linguagem VHDL oferece vários benefícios, dentre os quais, podem ser citados:

- Independência do estilo de implementação e da tecnologia utilizada no projeto;
- Facilidade na atualização dos projetos;

- Diferentes alternativas de implementação;
- Verificação do comportamento do sistema digital, através de simulação;
- Possibilidade de se considerarem os atrasos comuns aos circuitos digitais no projeto.

2.6 Implementação de técnicas de codificação de imagens em *hardware*

Nesta seção, são apresentados os principais trabalhos correlatos, entre os quais se destacam (GAUSE, J.; CHEUNG, P. Y. K.; LUK, W., 2000, 2004), como os mais relacionados a esta pesquisa.

Schoner (SCHONER, B. et al., 1995) propôs dois sistemas de compressão de vídeo que utilizam FPGAs. O primeiro sistema é limitado a um único algoritmo de compressão (Transformada Wavelet), apresenta baixa complexidade e usa reprogramabilidade num único FPGA. Nesse sistema, são realizadas otimizações no algoritmo, diminuindo a complexidade da implementação que explora reprogramabilidade. Já o segundo sistema pode implementar quatro algoritmos de compressão (2D-DCT, Transformada Wavelet, filtragem bidimensional e quantização vetorial) em tempo real. Isso foi possível através da combinação do FPGA com um processador de sinal de vídeo, o qual realiza operações que não precisam de reprogramabilidade, economizando recursos de *hardware* do FPGA.

Heron (HERON, J.; TRAINOR, D.; WOODS, R., 1997) implementou uma 2D-DCT com o dispositivo XC6200 da Xilinx. Foram utilizadas organização de dados apropriada (aritmética distribuída) e uma arquitetura que usa *pipeline* e paralelismo no processamento dos dados. O projeto alcançou uma taxa de 16,7 quadros por segundo (*frames per second* - fps). Como o FPGA utilizado tem capacidade para implementar até três circuitos de DCT, pode-se atingir, utilizando paralelismo, uma performance de 50,1 fps, o que é bastante considerável, visto que uma 2D-DCT, implementada em um processador Pentium 233, resulta em apenas 6,5 fps.

Le (LE, T.; GLESNER, M., 1999, 2000) propôs uma arquitetura da 1D-DCT, satisfazendo os requisitos de escalabilidade e modularidade para $2 \leq N \leq 8$. Essa arquitetura consiste na combinação de módulos recursivos de segunda ordem capazes de implementar uma arquitetura escalável, para implementar a 1D-DCT de tamanho variável, com o objetivo de implementar a SA-DCT. Nesse projeto reconfigurável, pode-se optar por uma arquitetura que apresente um baixo desempenho com alta modularidade (N variável) e grande consumo de recursos lógicos, ou uma arquitetura com grande economia de recursos lógicos, mas limitada a comprimentos fixos de N. As duas arquiteturas são descritas em VHDL e sintetizadas utilizando uma tecnologia CMOS (*Complementary Metal Oxide Semiconductor*) de duas camadas. Nesse trabalho, Le

não empregou técnicas de paralelismo, sua arquitetura processa um módulo da DCT de cada vez.

Yusof (MOHD-YUSOF, Z.; SULEIMAN, I.; ASPAR, Z., 2000) propôs a implementação da 2D-DCT e da 2D-IDCT através da utilização de algoritmos rápidos utilizando FPGA. A implementação da DCT/IDCT é baseada no algoritmo proposto por (CHEN, W. H.; SMITH, C. H.; FRALICK, S. C., 1991). Para reduzir a complexidade do *hardware*, técnicas de aritmética distribuída foram utilizadas. A 2D-DCT/IDCT foi implementada usando o dispositivo FLEX l0Kl00 FPGA/CPLD (100K gates CPLD) da Altera, com uma freqüência de *clock* de 11 MHz e 13,55 MHz, respectivamente. O projeto final consumiu apenas 50% do total das portas disponíveis no dispositivo Flexl0k100 (ALTERA, 2008).

Gause (GAUSE, J.; CHEUNG, P. Y. K.; LUK, W., 2000) propôs duas arquiteturas reconfiguráveis para implementar a SA-DCT: uma estática e outra dinâmica. As duas arquiteturas utilizam um grafo de dependência de dados (*Data Dependence Graph* - DDG) que representa os sinais de entrada e a computação requerida para calcular um sinal de saída dependente dos parâmetros de entrada, que são variáveis.

A arquitetura proposta para o cálculo da DCT de tamanho variável L (em que L representa o comprimento das colunas/linhas) baseia-se em dois módulos principais. No primeiro módulo, são realizados o deslocamento e a contagem dos pixels de cada coluna ou linha. Já no segundo módulo, uma matriz L x L de coeficientes constantes é multiplicada pelo vetor que contém os dados de entrada deslocados. Na arquitetura reconfigurável dinamicamente, enquanto o primeiro módulo é estático, isto é, o mesmo módulo é usado para todos os sinais de entrada da SA-DCT, o segundo é dinâmico, sendo sua estrutura variável em tempo real, de acordo com o valor de L. Assim, o processo de reconfiguração do dispositivo, para o cálculo da matriz DCT-L específica, repete-se para todas as colunas e linhas, até que a SA-DCT tenha sido completamente realizada.

Através da reestruturação do DDG (GAUSE, J.; CHEUNG, P. Y. K.; LUK, W., 2000) e da exploração do compartilhamento de alguns recursos do FPGA para diferentes entidades da SA-DCT, foi demonstrado que a área requerida é bastante reduzida, quando se utiliza reconfiguração dinâmica. Entretanto, também foi verificado que os FPGAs existentes no mercado apresentam um tempo de reconfiguração bastante longo, quando comparado com o tempo gasto em uma SA-DCT completa, o que tornou inviável a reconfiguração dinâmica para essa arquitetura. Esse tempo de reconfiguração dos FPGAs é crítico, uma vez que essa arquitetura é reconfigurada, toda vez que se vai processar a DCT, em uma coluna com tamanho diferente da anterior. Para um bloco com oito colunas e oito linhas de tamanhos diferentes, por exemplo, serão necessárias dezesseis reconfigurações para realizar a SA-DCT completa.

O projeto estático da SA-DCT proposto por (GAUSE, J.; CHEUNG, P. Y. K.; LUK, W., 2000)

para o MPEG-4 foi implementado no dispositivo FLEX 10K130E da Altera. O circuito possui uma frequência de 47 MHz, com uma vazão de um valor de coeficiente, a cada dois ciclos de *clock*, ou seja, uma taxa de 24 $Mpixels/s$. Uma SA-DCT completa foi processada num tempo de 4,47 μs.

Park (PARK, J.; KWON, S.; ROY, K., 2002) apresentou uma arquitetura de DCT reconfigurável de baixa potência. Essa arquitetura é baseada em compartilhamento de multiplicadores (*Computation Sharing Multiplier* - CSHM) e elimina a computação redundante na multiplicação de vetores por escalares. Nesse projeto reconfigurável, pode-se optar por uma maior qualidade da imagem, ou um menor consumo de potência. Os resultados experimentais mostraram que a DCT reconfigurável, usando CSHM, pode melhorar o consumo de potência em até 40%, sem uma degradação considerável da qualidade da imagem. Para medir objetivamente essa degradação, Park comparou o erro médio quadrático (EMQ) obtido, utilizando os coeficientes DCTs originais, com uma precisão de 8 bits, com os coeficientes modificados utilizando a técnica proposta. Quando se utilizaram 256 valores em tons de cinza, o EMQ resultante entre os dois conjuntos de coeficientes foi de 9, o que geralmente é aceitável num processo de compressão de imagens.

Já em (PARK, J.; ROY, K., 2004; PARK, J.; CHOI, J. H.; ROY, K., 2006), Park apresentou um projeto de DCT reconfigurável, que alcança uma redução considerável da complexidade computacional no seu cálculo, com uma mínima degradação na qualidade da imagem. Os trabalhos (PARK, J.; ROY, K., 2004; PARK, J.; CHOI, J. H.; ROY, K., 2006) são baseados na modificação da quantidade de bits utilizada pelos vetores base da DCT. Eles propuseram uma arquitetura reconfigurável dinamicamente, que permite não só trabalhar com um maior número de bits e obter uma qualidade melhor da imagem, como também trabalhar com menos bits e, conseqüentemente, reduzir o consumo de potência.

Em (GAUSE, J.; CHEUNG, P. Y. K.; LUK, W., 2004), Gause propôs também modelos simples para representar objetos de forma arbitrária e mapeá-los dentro de projetos de *hardware* específicos para aqueles objetos. Vários projetos e estratégias de reconfiguração foram investigados para mapear esses modelos, visando a um mapeamento eficiente das tarefas de processamento de vídeo adaptativo à forma, para uma dada arquitetura de computação reconfigurável. (GAUSE, J.; CHEUNG, P. Y. K.; LUK, W., 2004) demonstrou que um algoritmo, com um número pequeno de configurações, pode ser mais eficiente quando for implementado utilizando configuração estática ou multiconfigurações, enquanto que um projeto, empregando reconfiguração parcial ou dinâmica, será mais apropriado quando o número de configurações for relativamente maior. Devido à indisponibilidade de ferramentas de projeto que suportam reconfiguração dinâmica, as arquiteturas propostas por (GAUSE, J.; CHEUNG, P. Y. K.; LUK, W., 2004) foram reconfiguradas estaticamente.

Mohamed (MOHAMED, T. S.; BADAWY, W., 2004) propôs o projeto e a implementação de uma plataforma de *hardware* e *software* integrados, para processamento de imagens. A idéia é integrar um processador de propósito geral (*host*) com um elemento de processamento de propósito específico (FPGA). Essa plataforma é versátil, porque é configurada em tempo real, tem baixo consumo de potência e requer um baixo processamento do *host*. O *software* do *host* é responsável por configurar o dispositivo em tempo real, enviar os blocos dos dados de entrada e receber os resultados da compressão. A principal vantagem dessa plataforma é evitar que o *hardware* tenha que ser projetado novamente, quando se quiser alterar a técnica de codificação utilizada. Já a desvantagem é o elevado *overhead* gerado pela grande quantidade de portas de entrada e saída. Como estudo de caso para testar a plataforma proposta, configurou-se o FPGA em tempo real, com uma 2D-DCT. O dispositivo realizou a 2D-DCT, em um bloco de tamanho 8x8, com sucesso, utilizando 64 ciclos de *clock* a uma freqüência de 60 MHz.

Em (CARNEIRO, C. A. et al., 2006), Carneiro (CARNEIRO, C. A. et al., 2006) propõe uma arquitetura paralela parametrizável da 2D-DCT, para atender aplicações que requerem compressão de vídeo em tempo real. Nessa arquitetura, Carneiro utiliza dispositivos reconfiguráveis para implementar módulos da 1D-DCT, que podem ser instanciados de acordo com os requisitos da aplicação. O dispositivo utilizado na arquitetura proposta foi o CYCLONE II da Altera. Nesse trabalho, (CARNEIRO, C. A. et al., 2006) apresentou duas possíveis implementações da arquitetura: uma, que utiliza dois módulos da 1D-DCT e atende os propósitos do padrão de televisão *Standard Definition Television* (SDTV), e outra, que utiliza quatro módulos da 1D-DCT e atende os requisitos do padrão *High Definition Television* (HDTV). A primeira arquitetura pode obter uma taxa de até 40 $Mpixels/s$ e a segunda, uma taxa de até 80 $Mpixels/s$.

3 ARQUITETURA DE *HARDWARE* RECONFIGURÁVEL PARALELA DA SA-DCT

Neste capítulo, inicialmente, apresenta-se uma análise dos requisitos que se pretende atender com a nova arquitetura. Em seguida é mostrada a arquitetura proposta para implementar a SA-DCT, enfatizando-se, principalmente, a unidade reconfigurável paralela que implementa as DCTs de tamanho variável. Por último são abordados a codificação da arquitetura dessa unidade em VHDL e a sua implementação em dispositivos reconfiguráveis.

3.1 Análise dos requisitos da arquitetura da SA-DCT

A arquitetura a ser proposta tem como meta atender alguns requisitos, como desempenho, flexibilidade e tolerância a falhas, que podem variar de acordo com a aplicação e as características das imagens a serem processadas. Para alcançar esses requisitos, optou-se por utilizar os recursos de paralelismo e reconfiguração considerando as várias etapas do algoritmo da SA-DCT, mencionadas no Capítulo 2, que foram mapeadas em diferentes unidades do nível mais alto da arquitetura, denominado nível sistêmico.

O paralelismo será utilizado no processamento simultâneo de várias colunas e linhas de cada bloco de borda da imagem com o intuito de diminuir o tempo de execução da SA-DCT e, conseqüentemente, viabilizar o emprego dessa técnica para um número maior de aplicações. A arquitetura possuirá um grau de paralelismo variável, de acordo com a demanda de cada aplicação. Já a utilização da reconfiguração tem o intuito de tornar a arquitetura mais flexível (adaptável para cada tipo de imagem e requisitos da aplicação), visto que ela permite alterar algumas de suas características, como, por exemplo, a quantidade total de módulos de processamento da 1D-DCT de cada tamanho ou o número de bits utilizados no cálculo dos coeficientes. Com o aumento da flexibilidade, espera-se, por exemplo, poder fazer a opção entre ter-se uma maior qualidade da imagem processada, em detrimento de um maior consumo de recursos lógicos, ou vice-versa. As técnicas de reconfiguração também possibilitam uma maior tolerância a falhas no circuito digital, pois no caso de defeito em alguma parte do dispositivo, o circuito pode ser implementado em outra área do chip. Além disso, com a computação reconfigurável pode-se obter uma significativa redução de custos, uma vez que o recurso de reconfigurabilidade permite a utilização do mesmo *hardware* para diferentes versões do produto.

Inicialmente, para verificar o correto funcionamento da arquitetura proposta para a SA-DCT, optou-se por simular e implementar em FPGA somente a unidade que demanda maior desempenho nessa arquitetura. O FPGA é programável em campo, assim, os parâmetros dessa unidade

poderão ser alterados, via reconfiguração, de acordo com as características de cada bloco de borda e/ou de cada aplicação, sem que a unidade tenha quer ser retirada do sistema. Deve-se ressaltar que a implementação deverá apresentar características de independência do dispositivo utilizado (família, fabricante) e ser reconfigurável estaticamente.

3.2 Arquitetura proposta

A Figura 3.1 apresenta a arquitetura sistêmica da SA-DCT proposta, considerando um fluxo hipotético de dados de processamento para fins ilustrativos. As unidades reconfiguráveis paralelas que compõe essa arquitetura são:

- Unidade reconfigurável paralela (URP) de alinhamento dos pixels na borda superior.
- URP de detecção do comprimento das colunas.
- URP de processamento das DCTs unidimensionais de tamanho variável - *Unidimensionals Discrete Cosine Transforms - Variable Size* (1Ds-DCTs-VS). Nesta dissertação, com o objetivo de facilitar a compreensão do texto, optou-se por chamar as 1Ds-DCTs de tamanho variável somente de DCTs-VS. É importante mencionar que na arquitetura proposta existem duas unidades independentes de processamento das DCTs-VS, uma fará o processamento das colunas e a outra fará o processamento das linhas.
- URP de alinhamento dos pixels na borda esquerda.
- URP de detecção do comprimento das linhas.

Conforme mostra a Figura 3.1, a unidade de gerenciamento de reconfiguração recebe as informações referentes aos requisitos das aplicações e às características dos blocos de borda (comprimento das colunas do bloco original e linhas do bloco após o deslocamento dos segmentos verticais em direção à borda superior do bloco). Com essas informações e com base na quantidade de recursos lógicos disponíveis no *hardware*, essa unidade realizará a configuração das demais unidades de processamento. A escolha da configuração mais adequada é muito importante, visto que algumas aplicações requerem altas taxas de processamento, como, por exemplo, videoconferência e controle de veículos pilotados remotamente em aplicações militares e espaciais. Outras aplicações, por sua vez, priorizam custos de implementação, como transmissão de imagem por celulares e jogos eletrônicos de baixa resolução. Existem, ainda, aquelas aplicações que priorizam a qualidade das imagens, como o diagnóstico por imagens médicas.

A unidade de gerenciamento de reconfiguração também é responsável pela escolha dos tipos de configurações das unidades de processamento a serem utilizados: parcial e dinâmica ou

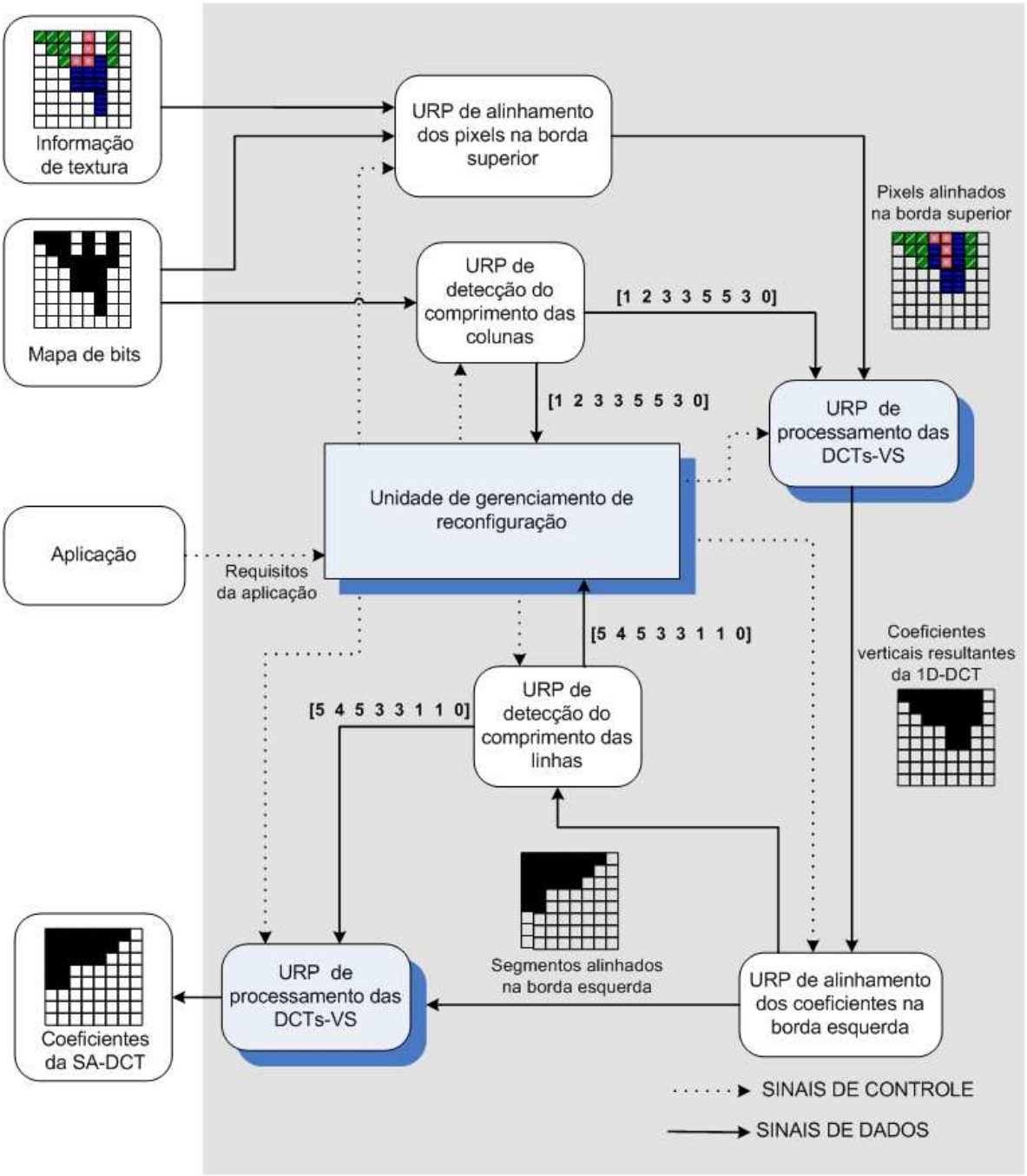


Figura 3.1: Diagrama em blocos da arquitetura de *Hardware* reconfigurável paralela dedicada para a implementação da SA-DCT

total e estática. Assim, a arquitetura pode sofrer uma reconfiguração parcial dinâmica, trocando-se apenas o *bitstream* enviado para configurar uma das unidades ou utilizar configurações pré-determinadas dessas unidades. A unidade de gerenciamento de reconfiguração pode ser implementada de quatro maneiras diferentes: *hardware* interno ao dispositivo reconfigurável (FPGA), *hardware* externo, *software* interno ou *software* externo ao dispositivo.

A URP de detecção do comprimento das colunas utiliza a informação contida no mapa de bits (informação de forma), para determinar o comprimento de cada coluna. O mapa de bits indica as posições do bloco de borda que possuem pixels do objeto, preenchidas com o bit 1, e as que não possuem, preenchidas com o bit 0. Uma vez determinado o comprimento das colunas, transmite-se essa informação para a unidade de gerenciamento de reconfiguração que fará a configuração da URP de processamento das DCTs-VS. Com a utilização da computação reconfigurável e do paralelismo na URP de detecção do comprimento das colunas, pode-se realizar a contagem dos pixels pertencentes ao objeto nas oito colunas (no caso de blocos de tamanho 8x8), simultaneamente, obtendo-se um ganho no tempo de processamento de até oito vezes. É importante notar que nessa unidade são implementados basicamente contadores e que o seu algoritmo demanda um tempo de processamento muito curto. Assim, caso haja necessidade de se economizar recursos lógicos, pode-se optar por diminuir o grau de paralelismo dessa unidade.

A URP de alinhamento dos pixels na borda superior realiza apenas o deslocamento dos segmentos verticais, obtendo-se assim os segmentos de comprimento variável que sofrerão, cada um deles, a primeira transformação 1D na URP de processamento das DCTs-VS. Assim como a URP de detecção do comprimento das colunas, essa unidade também pode realizar o deslocamento das oito colunas de pixels simultaneamente, obtendo-se um ganho no tempo de processamento de até oito vezes.

A URP de processamento das DCTs-VS, responsável pela implementação das diversas DCTs de tamanho variável, é a mais complexa e a que demanda maior desempenho na execução da SA-DCT. Com a utilização da computação reconfigurável e do paralelismo nessa unidade, tornou-se possível o processamento de várias 1Ds-DCTs de tamanhos diferentes (dependentes do tamanho de cada coluna de pixels do bloco de borda a ser processado) ao mesmo tempo. Observou-se que ao diminuir o tempo de processamento dessa unidade, obtém-se uma significativa melhora no desempenho global da SA-DCT. Assim, como se dispunha de pouco tempo para implementar todo o algoritmo da SA-DCT utilizando os recursos de paralelismo e reconfigurabilidade, optou-se, inicialmente, por implementar esses recursos na URP de processamento das DCTs-VS. A arquitetura interna da unidade reconfigurável paralela de processamento das DCTs-VS será melhor explicada na próxima seção.

A URP de detecção do comprimento das linhas determina o comprimento de cada linha, após o alinhamento dos segmentos verticais na borda superior. Essas informações serão utilizadas pela unidade de gerenciamento de reconfiguração para configurar a URP de processamento das DCTs-VS responsável pela transformação horizontal. Como nas outras unidades, através da utilização da computação reconfigurável e do paralelismo, pode-se obter um ganho no tempo de processamento de até oito vezes nessa unidade.

A URP de alinhamento dos coeficientes na borda esquerda alinha os coeficientes obtidos, após o primeiro processamento 1D sobre as colunas, pelos seus índices. Os segmentos horizontais de tamanho variável são então processados por outra URP de processamento das DCTs-VS para, finalmente, gerarem os coeficientes finais da SA-DCT. Assim como a URP de alinhamento dos pixels na borda superior, essa unidade também pode obter uma redução no seu tempo de processamento de até oito vezes, com a utilização da computação reconfigurável e do paralelismo.

3.3 Unidade reconfigurável paralela de processamento das DCTs-VS

A URP de processamento das DCTs-VS é necessariamente implementada em *hardware* interno ao dispositivo programável e pode ser configurada estaticamente ou dinamicamente. Optou-se inicialmente por realizar a implementação em *hardware* para evitar os atrasos de processamento inerentes às aplicações implementadas em *software*. Da mesma forma, optou-se por *hardware* interno para evitar os atrasos de propagação que poderiam ocorrer caso a implementação fosse externa ao dispositivo.

A implementação da arquitetura da URP de processamento das DCTs-VS consiste em vários módulos que implementam as DCTs unidimensionais de tamanhos L, sendo $1 \leq L \leq N$ (para blocos de borda de dimensão NxN). Nesta dissertação, faz-se N=8, que é a dimensão típica dos blocos de imagem no padrão MPEG-4. A figura 3.2 mostra uma URP de processamento das DCTs-VS com os possíveis módulos de processamento da 1D-DCT de tamanho L.

Esses módulos implementam as 1Ds-DCTs utilizando processamento de vetores. O cálculo consiste na utilização de uma matriz de constantes (A) obtidas através da utilização da seguinte fórmula:

$$A(i,j) = k * \cos[i * (j + \frac{1}{2}) * \frac{\pi}{N}], \ \ em\ que\ 0 \leq \ i,j \leq N-1 \tag{3.1}$$

em que $k = \sqrt{\frac{1}{2}}$ para $i = 0$ e $k = 1$, caso contrário.

As linhas dessa matriz são multiplicadas pelos valores dos dados de entrada e os resultados dessas multiplicações são armazenados no próprio dispositivo, para, posteriormente, serem

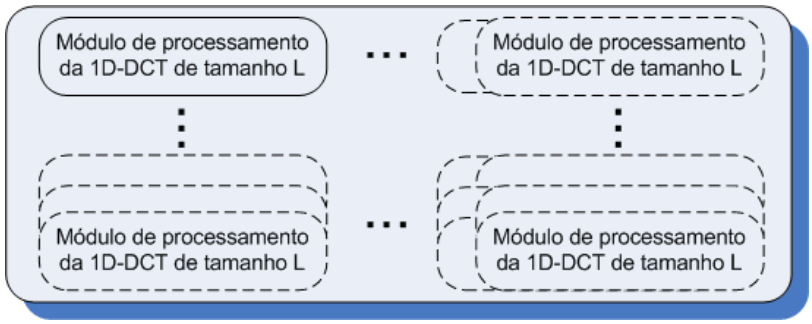


Figura 3.2: URP de processamento das DCTs-VS com os possíveis módulos de processamento da 1D-DCT de tamanho L

somados entre si, resultando num coeficiente da DCT-VS. Desse modo, o coeficiente de índice i ($0 \leq i \leq N-1$) da DCT, Z_i, resulta da soma de todos os resultados das multiplicações da i-ésima linha da matriz das constantes da DCT-VS (matriz A) pelo vetor de dados (X), ou seja:

$$Z_i = A_i X, \quad em\ que\ 0 \leq i \leq N-1 \tag{3.2}$$

No cálculo de uma 1D-DCT de tamanho 2, por exemplo, os dois coeficientes da DCT, Z_0 e Z_1, resultam de uma multiplicação das duas linhas da matriz A pelo vetor dos dados de entrada, $Z = AX$, que podem ser calculados como se segue:

$$A = \begin{bmatrix} 23170 & 23170 \\ 23170 & -23170 \end{bmatrix} \qquad X = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}$$

$$Z_0 = 23.170(x_0 + x_1)$$
$$Z_1 = 23.170x_0 - 23.170x_1 = 23.170(x_0 - x_1)$$

O diagrama em blocos do algoritmo utilizado na implementação do módulo da DCT de tamanho 2 é mostrado na Figura 3.3 a seguir.

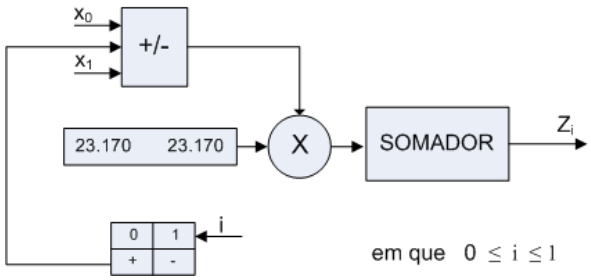


Figura 3.3: Algoritmo utilizado na implementação do módulo da 1D-DCT de tamanho 2

É importante notar que as operações de soma e subtração são determinadas pelo valor

do índice das linhas (i) da matriz A e comutam toda vez que o índice muda de valor.

Já no caso do cálculo de uma DCT de tamanho 6, por exemplo, os seis coeficientes Z_i (em que $0 \leq i \leq 5$) da 1D-DCT resultam, respectivamente, da multiplicação de cada linha da matriz A pelo vetor dos dados de entrada, como mostrado a seguir:

$$A = \begin{bmatrix} 13377 & 13377 & 13377 & 13377 & 13377 & 13377 \\ 18274 & 13377 & 4897 & -4897 & -13377 & -18274 \\ 16384 & 0 & -16384 & -16384 & 0 & 16384 \\ 13377 & -13377 & -13377 & 13377 & 13377 & -13377 \\ 9459 & -18919 & 9459 & 9459 & -18919 & 9459 \\ 4897 & -13377 & 18274 & -18274 & 13377 & -4897 \end{bmatrix} \quad X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

$Z_0 = 13.377(x_0 + x_1 + x_2 + x_3 + x_4 + x_5)$

$Z_1 = 18.274x_0 + 13.377x_1 + 4.897x_2 - 4.897x_3 - 13.377x_4 - 18.274x_5 = 18.274(x_0 - x_5) + 13.377(x_1 - x_4) + 4.897(x_2 - x_3)$

$Z_2 = 16.384x_0 + 0x_1 - 16.384x_2 - 16.384x_3 + 0x_4 + 16.384x_5 = 16.384(x_0 + x_5) + 0(x_1 + x_4) - 16.384(x_2 + x_3)$

$Z_3 = 13.377x_0 - 13.377x_1 - 13.377x_2 + 13.377x_3 + 13.377x_4 - 13.377x_5 = 13.377(x_0 - x_5) - 13.377(x_1 - x_4) - 13.377(x_2 - x_3)$

$Z_4 = 9.459x_0 - 18.919x_1 + 9.459x_2 + 9.459x_3 - 18.919x_4 + 9.459x_5 = 9.459(x_0 + x_5) - 18.919(x_1 + x_4) + 9.459(x_2 + x_3)$

$Z_5 = 4.897x_0 - 13.377x_1 + 18.274x_2 - 18.274x_3 + 13.377x_4 - 4.897x_5 = 4.897(x_0 - x_5) - 13.377(x_1 - x_4) + 18.274(x_2 - x_3)$

O diagrama em blocos do algoritmo utilizado na implementação do módulo da DCT de tamanho 6 é mostrado na Figura 3.4, a seguir.

Inicialmente, pensou-se numa arquitetura que implementasse a SA-DCT para imagens acromáticas. Por isso, a arquitetura proposta para a unidade que implementa a 1D-DCT de tamanho variável realiza apenas o processamento da componente Y (luminância), uma das três componentes do sinal de vídeo que é responsável pelos diversos tons de cinza da imagem. Nesta implementação, utilizou-se um sinal de *clock* de 80 MHz e obteve-se um coeficiente a cada pulso de clock, resultando numa taxa mínima de 80 $Mpixels/s$.

Conforme mencionado anteriormente, a URP de processamento das DCTs-VS pode ser reconfigurada para implementar os módulos de acordo com os requisitos da aplicação e com os comprimentos das colunas de cada bloco de borda. A Figura 3.5 mostra algumas possibilidades de configuração da URP de processamento das DCTs-VS para determinado bloco de borda.

Caso a aplicação demande um alto desempenho, pode-se utilizar um módulo da DCT dedicado para cada coluna, conforme mostra a Figura 3.5(a), em que a URP de processamento das DCTs-VS possui dois módulos da DCT de tamanho 6 e três módulos da DCT de tamanho 2.

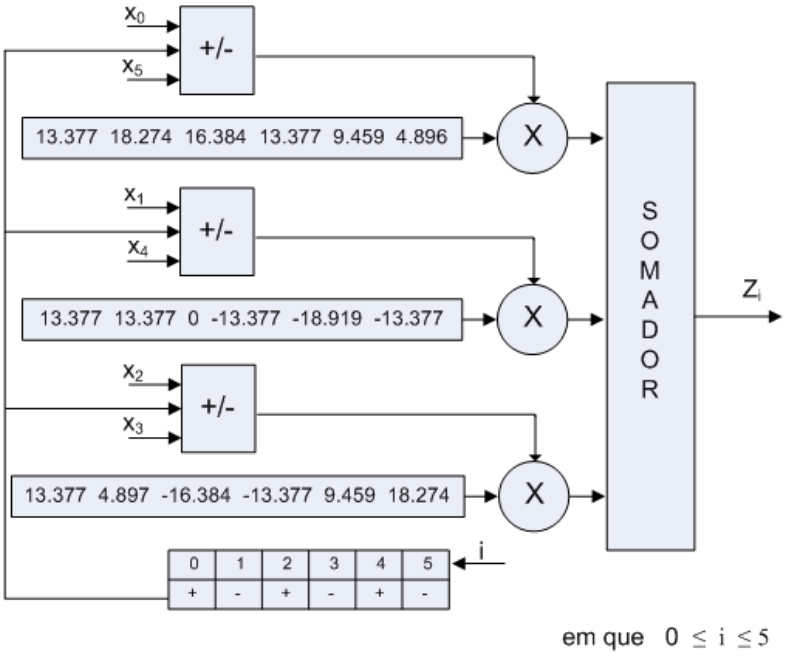


Figura 3.4: Algoritmo utilizado na implementação do módulo da 1D-DCT de tamanho 6

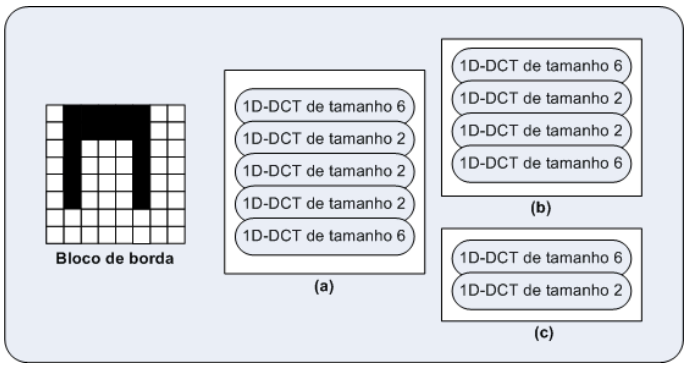


Figura 3.5: Possíveis configurações de uma URP de processamento das DCTs-VS para um bloco de borda

Logo, todas as colunas do bloco mostrado na Figura 3.5(a) poderão ser processadas em paralelo. Caso a aplicação não só demande desempenho, mas também priorize o custo, pode-se utilizar a unidade da Figura 3.5(b), que concilia um pouco esses dois requisitos. Já para uma aplicação que priorize somente o custo, em que o desempenho não é tão importante, pode-se utilizar uma URP de processamento das DCTs-VS que possua apenas um módulo que execute a DCT de tamanho 6 e um módulo que execute a DCT de tamanho 2, como mostra a Figura 3.5(c).

Neste trabalho, optou-se por utilizar apenas três das inúmeras configurações da arquitetura de *hardware* proposta para essa unidade, cujas denominações estão relacionadas ao grau de paralelismo, que é o principal parâmetro arquitetural reconfigurável da arquitetura da SA-DCT.

Na primeira configuração proposta, chamada de Configuração Serial, a URP de processamento das DCTs-VS possui apenas um módulo da DCT para cada tamanho L, as entradas de pixels são seriais, coluna a coluna, e cada módulo é executado um após o outro. É importante observar que, embora já esteja configurado um módulo para cada tamanho da DCT, esse processamento é realizado serialmente. A Figura 3.6 apresenta o funcionamento sistêmico dessa configuração.

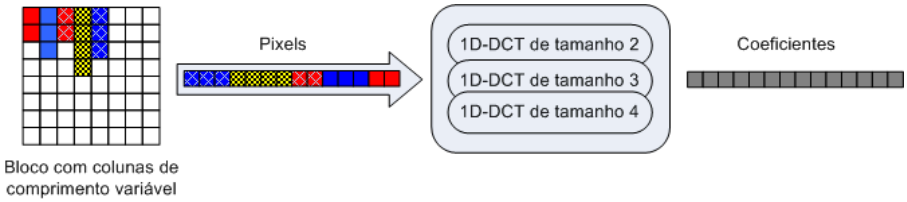


Figura 3.6: Funcionamento da Configuração Serial da URP de processamento das DCTs-VS

Na segunda configuração proposta, chamada de Configuração Paralela, a URP de processamento das DCTs-VS também possui apenas um módulo para cada tamanho L ($1 \leq L \leq 8$), entretanto todos os módulos são executados simultaneamente, isto é, todas as colunas de pixels que possuem comprimentos diferentes são processadas em paralelo. O seu objetivo é utilizar o maior grau de paralelismo possível sem, contudo, aumentar a quantidade de recursos lógicos utilizados. A Figura 3.7 mostra um diagrama simplificado do funcionamento dessa configuração.

Já na terceira configuração, chamada de Configuração Ideal, a URP de processamento das DCTs-VS possui quantos módulos (para cada tamanho L) forem necessários para processar todas as colunas do bloco de borda em paralelo, inclusive as que possuem o mesmo comprimento. A Figura 3.8 mostra o diagrama de funcionamento da Configuração Ideal.

No último caso, a demanda de recursos computacionais é muito maior, pois para realizar o processamento de todas as colunas em paralelo, é necessário que se tenha uma configuração específica da URP para cada bloco de borda, visto que, normalmente, cada um deles possui

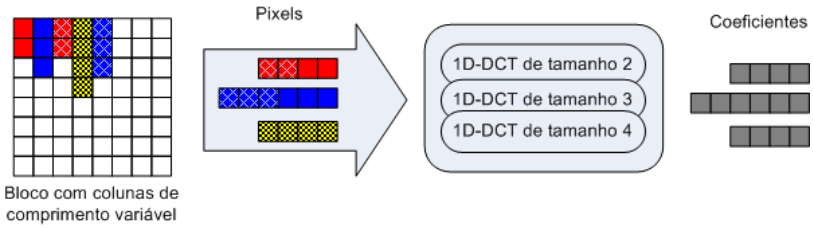


Figura 3.7: Funcionamento da Configuração Paralela da URP de processamento das DCTs-VS

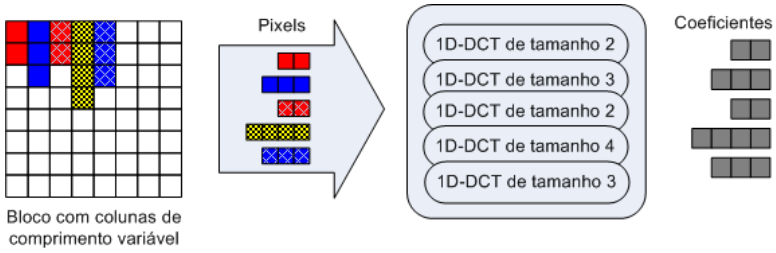


Figura 3.8: Funcionamento da Configuração Ideal da URP de processamento das DCTs-VS

uma quantidade e localização dos pixels diferente. Isso é inviável na maioria das vezes, tendo em vista a grande quantidade de configurações e, conseqüentemente, o elevado consumo de recursos lógicos. Assim, com o objetivo de viabilizar essa proposta, aplicam-se os conceitos e técnicas de computação reconfigurável, pois, com a utilização dessas técnicas, é possível reconfigurar integralmente ou parcialmente o dispositivo, com a configuração mais adequada para cada bloco, dispensando dessa forma um elevado número de configurações.

Através de pesquisas e estudos na tentativa de viabilizar a utilização das técnicas de computação reconfigurável dinâmica na implementação da URP de processamento das DCTs-VS, constatou-se que os dispositivos existentes no mercado (FPGAs) ainda possuem um tempo elevado de reconfiguração (ALTERA, 2008; XILINX, 2008). Isso pode ser bastante crítico, visto que, em determinados casos, o tempo de configuração é mais elevado do que o tempo de processamento da 1D-DCT. Nesses casos, para que o tempo de reconfiguração não seja um inconveniente, pode-se utilizar o recurso da reconfiguração parcial e dinâmica. Esse recurso consiste na reconfiguração de algumas áreas do dispositivo, enquanto outras permanecem em execução, conforme mencionado no Capítulo 2 desta dissertação. Dessa forma, pode-se, por exemplo, configurar parte do dispositivo com uma determinada configuração da URP de processamento das DCTs-VS, enquanto outra parte do dispositivo estiver executando o processamento de um bloco.

Caso não seja viável utilizar o processo de reconfiguração dinâmica parcial, em função do custo mais elevado dos dispositivos que apresentam tais funcionalidades, pode-se optar por configurar estaticamente algumas das unidades mais utilizadas pelos blocos de borda da imagem

que será processada ou utilizar multiconfigurações. Gause em seu trabalho (GAUSE, J.; CHEUNG, P. Y. K.; LUK, W., 2004) propôs modelos simples para representar objetos de forma arbitrária e mapeá-los dentro de projetos de *hardware* específicos para aquele objeto. Assim, se a utilização da reconfiguração dinâmica parcial for inviável em função do custo dos dispositivos, pode-se, por exemplo, optar por utilizar os modelos propostos por (GAUSE, J.; CHEUNG, P. Y. K.; LUK, W., 2004) para representar e mapear os objetos dentro de projetos de *hardware* específicos. Nesse caso, ocorre uma perda no desempenho, se comparado àquele alcançado utilizando uma Configuração Ideal para cada bloco, em que todas as colunas são processadas de uma só vez. Para tanto, é importante fazer uma análise da relação entre o custo computacional e o desempenho para cada aplicação, definindo assim o tipo de configuração a ser utilizada.

3.4 Codificação da arquitetura em VHDL

A linguagem escolhida para descrever, simular e sintetizar a arquitetura proposta para a SA-DCT foi o VHDL. O motivo dessa escolha foi a flexibilidade oferecida por essa linguagem na descrição do comportamento de circuitos eletrônicos digitais, uma vez que suporta diferentes tipos de dados e permite estruturar o projeto de tal forma que ele possa ser facilmente adaptado para outros ambientes de desenvolvimento, independentemente da tecnologia e do dispositivo utilizado na implementação. Além disso, o VHDL tem a capacidade de descrever o comportamento e o desempenho para um circuito sob teste, na forma comumente referida como *test bench* (código em VHDL que descreve as entradas de dados e sinais, *clock*, *reset*, *enable*, usado para verificar o correto funcionamento do circuito).

É importante salientar que a descrição do comportamento/lógica de um modelo em VHDL é composta de duas partes principais: a declaração da entidade e o corpo arquitetural. O primeiro define as portas de entrada e saída e o segundo descreve a arquitetura do modelo que pode ser comportamental, estrutural ou uma combinação desses dois. As Figuras 3.9 e 3.10 mostram, respectivamente, a entidade e o corpo arquitetural da descrição em VHDL de uma URP de processamento das DCTs-VS para um bloco de borda composto por colunas de tamanho 6 e 2, utilizando a Configuração Paralela.

A Figura 3.11 mostra o corpo arquitetural da descrição em VHDL de uma URP de processamento das DCTs-VS que possui dois módulos que executam a DCT de tamanho 2 e dois módulos que executam a DCT de tamanho 6.

Neste trabalho, como a meta principal é a proposta de uma arquitetura de *hardware* reconfigurável e paralela dedicada para a implementação da SA-DCT, preocupou-se mais com aspectos macro da arquitetura, permitindo assim a escolha do modelo comportamental para a descrição do projeto em VHDL. O VHDL utilizado também ficou restrito ao subconjunto do

```
ENTITY DCTs_66222266 IS
PORT (

      CLK_v                    : IN std_logic;
      RST_2                    : IN std_logic;
      RST_6                    : IN std_logic;
      xin_2                   : IN std_logic_vector(7 downto 0);    -- entrada de 8 bits
      xin_6                   : IN std_logic_vector(7 downto 0);    -- entrada de 8 bits
      dct_1d_2                 : OUT std_logic_vector(11 downto 0); -- saída de 12 bits
      dct_1d_6                 : OUT std_logic_vector(11 downto 0); -- saída de 12 bits
      rdy_out_2               : OUT std_logic;
      rdy_out_6               : OUT std_logic);
END DCTs_66222266;
```

Figura 3.9: Entidade da URP de processamento das DCTs-VS com uma instância do módulo que processa a 1D-DCT de tamanho 6 e uma instância do módulo que processa a 1D-DCT de tamanho 2

```
ARCHITECTURE logic OF DCTs_66222266 IS

component dct6 is
    port (
      CLK                     : IN std_logic;
      RST                     : IN std_logic;
      xin                     : IN std_logic_vector(7 downto 0);   --  8 bit input.
      dct                     : OUT std_logic_vector(11 downto 0);
      rdy_out                 : OUT std_logic);
end component;

component dct2 is
    port (
      CLK                     : IN std_logic;
      RST                     : IN std_logic;
      xin                     : IN std_logic_vector(7 downto 0);   --  8 bit input.
      dct                     : OUT std_logic_vector(11 downto 0);
      rdy_out                 : OUT std_logic);
end component;

begin
inst_dct6   : component dct6
     port map (
      CLK           =>   CLK_v,
      RST           =>   RST_6,
      xin           =>   xin_6 ,
      dct           =>   dct_1d_6,
      rdy_out       =>   rdy_out_6 );

inst_dct2 : component dct2
     port map (
      CLK           =>   CLK_v,
      RST           =>   RST_2,
      xin           =>   xin_2,
      dct           =>   dct_1d_2,
      rdy_out       =>   rdy_out_2 );

end logic;
```

Figura 3.10: Corpo arquitetural da URP de processamento das DCTs-VS com uma instância do módulo que processa a 1D-DCT de tamanho 6 e uma instância do módulo que processa a 1D-DCT de tamanho 2

```
ARCHITECTURE logic OF DCTs_66222266 IS

component dct6 is
    port (
      CLK                   : IN std_logic;
      RST                   : IN std_logic;
      xin                   : IN std_logic_vector(7 downto 0);   --  8 bit input.
      dct                   : OUT std_logic_vector(11 downto 0);
      rdy_out               : OUT std_logic);
end component;

component dct2 is
    port (
      CLK                   : IN std_logic;
      RST                   : IN std_logic;
      xin                   : IN std_logic_vector(7 downto 0);   --  8 bit input.
      dct                   : OUT std_logic_vector(11 downto 0);
      rdy_out               : OUT std_logic);
end component;

begin

inst_dct6_1 : component dct6
     port map (
      CLK           =>   CLK_v,
      RST           =>   RST_6_1,
      xin           =>   xin_6_1,
      dct           =>   dct_1d_6_1,
      rdy_out       =>   rdy_out_6_1 );

inst_dct6_2 : component dct6
     port map (
      CLK           =>   CLK_v,
      RST           =>   RST_6_2,
      xin           =>   xin_6_2,
      dct           =>   dct_1d_6_2,
      rdy_out       =>   rdy_out_6_2 );

inst_dct2_1 : component dct2
     port map (
      CLK           =>   CLK_v,
      RST           =>   RST_2_1,
      xin           =>   xin_2_1,
      dct           =>   dct_1d_2_1,
      rdy_out       =>   rdy_out_2_1 );

inst_dct2_2 : component dct2
     port map (
      CLK           =>   CLK_v,
      RST           =>   RST_2_2,
      xin           =>   xin_2_2,
      dct           =>   dct_1d_2_2,
      rdy_out       =>   rdy_out_2_2 );

end logic;
```

Figura 3.11: Corpo arquitetural da URP de processamento das DCTs-VS com duas instâncias do módulo que processa a 1D-DCT de tamanho 6 e duas instâncias do módulo que processa a 1D-DCT de tamanho 2

VHDL que é sintetizável, para que a arquitetura possa ser implementada em circuitos reais.

3.5 Implementação em FPGA

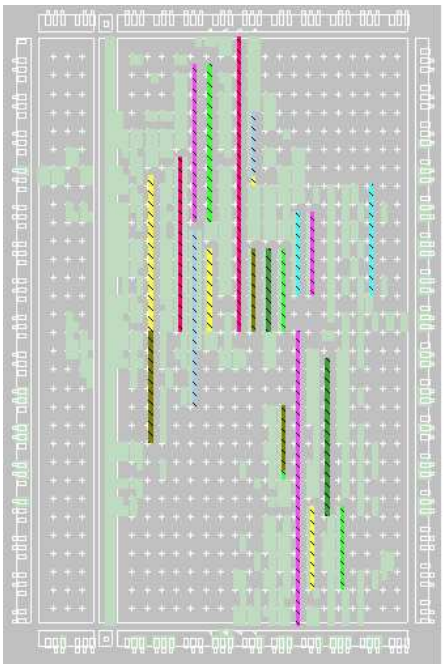
A implementação da arquitetura proposta é baseada na utilização de dispositivos lógicos reconfiguráveis, FPGAs, devido à sua demanda por capacidade de reconfiguração após a fabricação do dispositivo e à necessidade de alto desempenho. Esses dispositivos lógicos reconfiguráveis oferecem desempenho comparável com o oferecido por dispositivos de propósito específico (*Application Specific Integrated Circuits* - ASICs) integração de mais funções, tal como processador e memória, reduzido consumo de potência e custos competitivos (MARTINS, C. A. P. S. et al., 2003). Para simular, sintetizar e implementar a arquitetura proposta, utilizou-se um FPGA da família Spartan 3 da Xilinx, mais especificamente o XC3s1000-5fg320. Esse FPGA possui 17.280 células lógicas organizadas em 1.920 CLBs (*Configurable Logic Blocks*), uma RAM interna de 432K e um espaço de armazenamento distribuído de 120K, 24 multiplicadores dedicados e 391 portas de entrada e saída.

Optou-se por utilizar os dispositivos da Xilinx, em função das suas qualidades e vantagens em relação aos dispositivos dos demais fabricantes (ACTEL, 2008; ALTERA, 2008; ATMEL, 2007). Essas vantagens podem ser comprovadas na série de dispositivos Virtex-V, otimizada para alta velocidade e baixo consumo de potência, que inclui processadores Power PC e dispositivos seriais de entrada e saída RocketIO (XILINX, 2008). Entre outras vantagens, está o conjunto de ferramentas do pacote ISE, que possibilita a simulação, mapeamento, posicionamento, roteamento e geração de arquivos de configuração, parciais ou totais, numa interface de fácil manipulação. Nessa arquitetura não se verificou a necessidade de utilizar um dispositivo com as características da série Virtex-V mas, mesmo assim, optou-se pela Xilinx devido ao seu conjunto de ferramentas que facilita bastante o desenvolvimento dos projetos.

Neste trabalho utilizou-se uma ferramenta, chamada *Floorplanner*, que permite visualizar e editar a localização das constantes, dos recursos lógicos e do roteamento, além de alocar as portas automaticamente num projeto modular. Através da utilização dessa ferramenta, foi possível visualizar os recursos lógicos consumidos para cada uma das configurações implementadas. Pode-se verificar, por exemplo, que a Configuração Ideal normalmente consome uma quantidade significativamente maior de recursos, quando comparada com a Configuração Serial ou Paralela. Essa ferramenta também está disponível no pacote ISE da Xilinx.

A Figura 3.12 mostra o *floorplanning* Pós *Place & Route* da Configuração Serial, apresentada na Seção 3.3, para um bloco de borda que possui quatro colunas de tamanho 6 e quatro colunas de tamanho 2, num FPGA XC3S50 da família Spartan 3 da Xilinx.

Para mostrar a diferença de recursos lógicos consumidos numa Configuração Serial e



Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	409	1,536	26%	
Number of 4 input LUTs	455	1,536	29%	
Logic Distribution				
Number of occupied Slices	374	768	48%	
Number of Slices containing only related logic	374	374	100%	
Number of Slices containing unrelated logic	0	374	0%	
Total Number 4 input LUTs	543	1,536	35%	
Number used as logic	455			
Number used as a route-thru	88			
Number of bonded IOBs	45	124	36%	
IOB Flip Flops	16			
Number of MULT18X18s	4	4	100%	
Number of GCLKs	1	8	12%	
Total equivalent gate count for design	23,651			
Additional JTAG gate count for IOBs	2,160			

Figura 3.12: Quantidade de recursos lógicos consumidos na Configuração Serial e o *floorplanning* para o dispositivo XC3S50

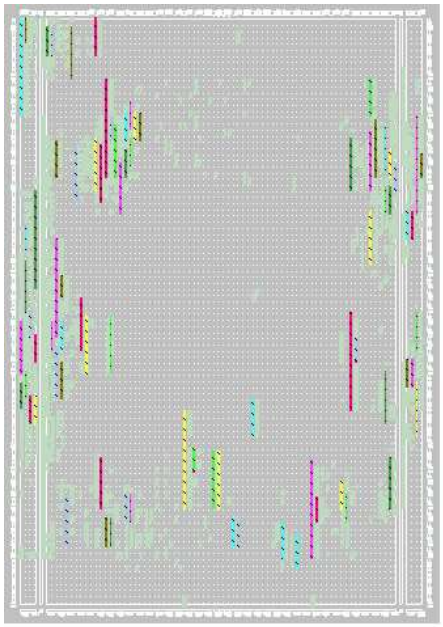
numa Configuração Ideal, tentou-se obter o *floorplanning* Pós *Place & Route* da Configuração Ideal para o bloco com quatro colunas de tamanho 6 e quatro de tamanho 2 no mesmo FPGA XC3S50. Entretanto isso não foi possível, porque esse FPGA não possui a quantidade de recursos lógicos necessários para se implementar a Configuração Ideal. A Tabela 3.1 mostra a quantidade de recursos lógicos disponíveis no dispositivo XC3S50 e a quantidade de recursos lógicos necessários para a implementação da Configuração Ideal. Essa tabela foi extraída do relatório de "Status do Projeto" do *software* da Xilinx.

A implementação da Configuração Ideal também não foi possível nos dispositivos XC3S200 e XC3S400 da família Spartan 3, devido à quantidade de recursos lógicos necessários nessa configuração. O menor dispositivo da família Spartan 3 que pode ser utilizado para a implementação dessa configuração foi o XC3S1000. Na Figura 3.13 pode-se visualizar a quantidade de recursos lógicos consumidos nessa configuração e o *floorplanning* dessa configuração para o dispositivo XC3S1000.

Com o objetivo de se comparar visualmente a quantidade de recursos lógicos consumidos na Configuração Serial e na Ideal, optou-se por mostrar também o *floorplanning* da Configuração Serial para um dispositivo XC3S1000 da família Spartan 3. A Figura 3.14 mostra a quantidade de recursos lógicos consumidos na Configuração Serial e o *floorplanning* dessa configuração para o dispositivo XC3S1000.

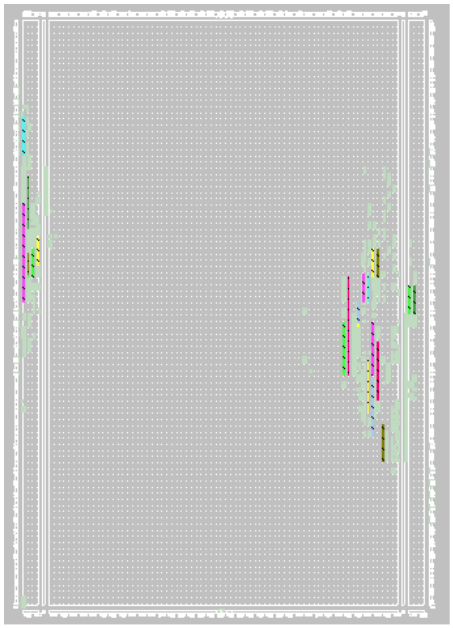
Tabela 3.1: Recursos lógicos disponíveis no dispositivo XC3S50 e recursos necessários na Configuração Ideal

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	1,628	1,536	105%	OVERMAPPED
Number of 4 input LUTs	2,396	1,536	155%	OVERMAPPED
Logic Distribution				
Number of occupied Slices	1,656	768	215%	
Number of Slices containing only related logic	1,524	1,656	92%	
Number of Slices containing unrelated logic	132	1,656	7%	
Total Number 4 input LUTs	2,780	1,536	180%	OVERMAPPED
Number used as logic	2,396			
Number used as a route-thru	384			
Number of bonded IOBs	177	124	142%	OVERMAPPED
IOB Flip Flops	64			
Number of MULT18X18s	4	4	100%	
Number of GCLKs	1	8	12%	
Total equivalent gate count for design	53,279			
Additional JTAG gate count for IOBs	8,496			



Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	1,636	15,360	10%	
Number of 4 input LUTs	1,820	15,360	11%	
Logic Distribution				
Number of occupied Slices	1,498	7,680	19%	
Number of Slices containing only related logic	1,498	1,498	100%	
Number of Slices containing unrelated logic	0	1,498	0%	
Total Number 4 input LUTs	2,172	15,360	14%	
Number used as logic	1,820			
Number used as a route-thru	352			
Number of bonded IOBs	177	221	80%	
IOB Flip Flops	64			
Number of MULT18X18s	16	24	66%	
Number of GCLKs	1	8	12%	
Total equivalent gate count for design	94,595			
Additional JTAG gate count for IOBs	8,496			

Figura 3.13: Quantidade de recursos lógicos consumidos na Configuração Ideal e o *floorplanning* para o dispositivo XC3S1000



Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	409	15,360	2%	
Number of 4 input LUTs	455	15,360	2%	
Logic Distribution				
Number of occupied Slices	374	7,680	4%	
Number of Slices containing only related logic	374	374	100%	
Number of Slices containing unrelated logic	0	374	0%	
Total Number 4 input LUTs	543	15,360	3%	
Number used as logic	455			
Number used as a route-thru	88			
Number of bonded IOBs	45	221	20%	
IOB Flip Flops	16			
Number of MULT18X18s	4	24	16%	
Number of GCLKs	1	8	12%	
Total equivalent gate count for design	23,651			
Additional JTAG gate count for IOBs	2,160			

Figura 3.14: Quantidade de recursos lógicos consumidos na Configuração Serial e o *floorplanning* para o dispositivo XC3S1000

4 RESULTADOS

Neste capítulo é apresentada a verificação funcional da unidade reconfigurável paralela de processamento das DCTs-VS. Inicialmente, definem-se algumas configurações e alguns tipos de blocos de borda para verificar o correto funcionamento da arquitetura dessa unidade. Em seguida, apresentam-se os resultados das simulações obtidos para cada tipo de bloco de borda, bem como uma verificação e validação de todos os coeficientes gerados pela URP de processamento das DCTs-VS, através do cálculo do erro médio quadrático entre o bloco de borda original e o seu equivalente reconstruído. Feito isso, é realizada uma análise dos resultados obtidos, comparando características da arquitetura, como tempo de processamento e consumo de recursos lógicos, de cada configuração, para todos os tipos de blocos de borda analisados. Por último, apresentam-se algumas representações estatísticas (agrupamentos de blocos de borda, nos quais se predominam determinadas características) que são utilizadas para avaliar o ganho real da utilização da computação reconfigurável na variação do grau de paralelismo da arquitetura proposta.

4.1 Configurações e blocos de borda utilizados na verificação da URP de processamento das DCTs-VS

Com o objetivo de verificar a funcionalidade e estimar o desempenho da arquitetura proposta, optou-se por desenvolver e implementar a URP de processamento das DCTs-VS, com os três tipos de configurações mencionados na Seção 3.3 (Configurações Serial, Paralela e Ideal), embora existam várias outras possibilidades. Conforme explicado no Capítulo 3, a URP de processamento das DCTs-VS é responsável pelo processamento 1D da DCT em cada coluna ou linha de tamanho variável dos blocos de borda da imagem. O principal objetivo da implementação dessa arquitetura é mostrar o ganho de desempenho alcançado quando se aumenta o grau de paralelismo no processamento dos segmentos de dados, em relação às implementações feitas utilizando as arquiteturas existentes na literatura (GAUSE, J.; CHEUNG, P. Y. K.; LUK, W., 2000).

Conforme mencionado na Seção 3.3, na Configuração Serial as entradas dos dados são seriais, segmento a segmento, e o processamento também, embora já se encontrem configurados no dispositivo os módulos que executam o processamento de cada DCT. Na Configuração Paralela, todos os segmentos de dados que possuem comprimentos diferentes são processadas em paralelo. E, na última configuração proposta, chamada de Configuração Ideal, todos os segmentos são processados em paralelo, inclusive os que possuem o mesmo comprimento. Nessa configuração, possivelmente, o tempo de processamento diminuirá, em detrimento de um au-

mento considerável de recursos lógicos.

Com o objetivo de verificar qual configuração seria mais apropriada para cada tipo de bloco de borda (blocos com poucos ou muitos pixels do objeto distribuídos homegeneamente ou não nos segmentos), foram usados três grupos de blocos de borda na simulação de cada configuração da URP de processamento das DCTs-VS. Nesse estudo, considerou-se apenas a transformação 1D das colunas (segmentos verticais).

A escolha desses três grupos de teste foi realizada com base nos conceitos do método de planejamento de experimentos (GOUPY, J., 1988) que consiste em variações planejadas de determinados critérios, como por exemplo, a quantidade de pixels e a homogeneidade da distribuição desses nas colunas. Para tanto, o primeiro grupo de teste contém blocos que possuem poucos pixels do objeto, sendo que o primeiro bloco de borda possui pixels distribuídos uniformemente em todas as colunas, o segundo possui a maior parte dos pixels concentrada em poucas colunas e o terceiro apresenta uma distribuição completamente desigual. A Figura 4.1 mostra um esboço do primeiro grupo de teste.

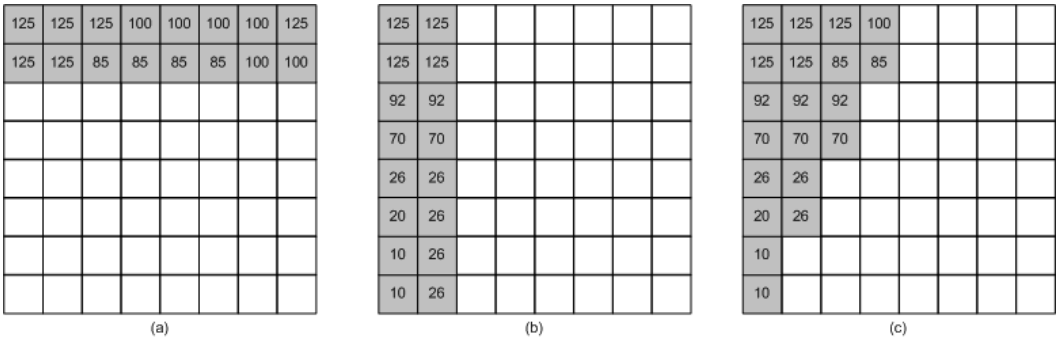


Figura 4.1: Grupo 1 - Blocos de borda que possuem poucos pixels do objeto

No primeiro e no segundo blocos da Figura 4.1, não há possibilidade de realizar o processamento das colunas em paralelo, considerando-se que exista a restrição de somente um módulo de processamento para cada comprimento da DCT (Configuração Paralela). Isso só seria possível se a URP de processamento fosse reconfigurada para utilizar outro tipo de configuração como, por exemplo, a Configuração Ideal. Já no terceiro bloco, pode-se realizar o processamento de todas as colunas em paralelo, tanto com a Configuração Paralela quanto com a Configuração Ideal, uma vez que cada coluna desse bloco possui um comprimento diferente.

Assim como no Grupo 1, os grupos de teste 2 e 3 também possuem um bloco de borda que apresenta uma distribuição mais uniforme dos pixels nas colunas, outro bloco que possui uma maior concentração dos pixels em algumas colunas e o terceiro que apresenta uma distribuição completamente desigual dos pixels nas colunas. A diferença entre o segundo e o terceiro grupo de teste é que o segundo grupo possui a metade dos seus pixels pertencentes ao

objeto e o terceiro possui a maior parte dos seus pixels pertencentes ao objeto. A Figura 4.2 mostra o segundo grupo de blocos de borda utilizado e a Figura 4.3, o terceiro.

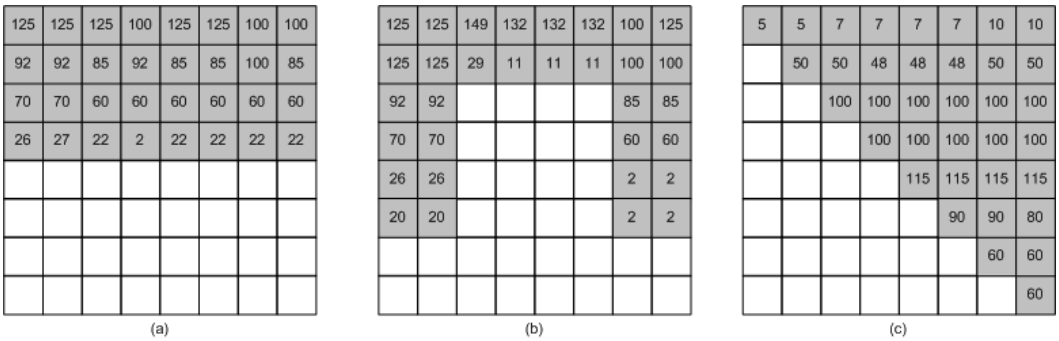


Figura 4.2: Grupo 2 - Blocos de borda que possuem a metade de seus pixels pertencentes ao objeto

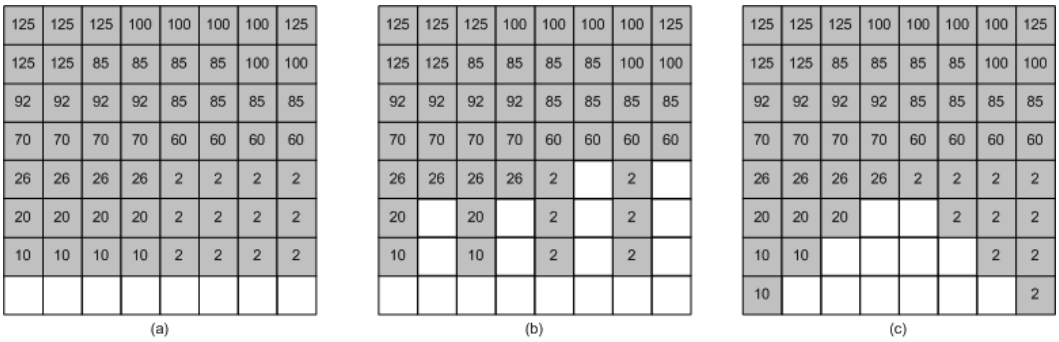


Figura 4.3: Grupo 3 - Blocos de borda que possuem a maior parte de seus pixels pertencentes ao objeto

Na Figura 4.2, pode-se notar que não há possibilidade de realizar o processamento das colunas do bloco “a” em paralelo, a menos que se utilize a Configuração Ideal da URP de processamento das DCTs-VS. Já nos blocos “b” e “c”, existe a possibilidade, caso se utilize a Configuração Paralela, de se processar duas colunas (no caso de bloco “b”) e oito colunas (no caso do bloco “c”), simultaneamente.

Como nos grupos anteriores, nos tipos de blocos apresentados na Figura 4.3 (Grupo 3), não há possibilidade de realizar o processamento das colunas do bloco “a” em paralelo, a menos que se utilize a Configuração Ideal da URP de processamento das DCTs-VS. Já nos blocos “b” e “c”, pode-se optar por utilizar a Configuração Paralela e processar algumas colunas desses blocos em paralelo.

Nos resultados das simulações (seção seguinte), pode-se verificar que existe uma configuração ótima para cada tipo de aplicação e para cada tipo de bloco de borda. Essa análise é muito importante, visto que algumas aplicações, conforme foi mencionado anteriormente, re-

querem altas taxas de processamento, enquanto outras priorizam custo de implementação e/ou tempo de prototipação.

4.2 Resultados das simulações

Para realizar a verificação funcional e de desempenho da URP de processamento das DCTs-VS, foram criados arquivos de testes que simulam a chegada dos pixels dos blocos de borda na arquitetura dessa unidade. Assim, através do Xilinx ISE Simulator, foi possível simular os resultados de cada configuração proposta para a arquitetura (Configurações Serial, Paralela e Ideal) em todos os grupos de blocos de borda apresentados anteriormente.

As Figuras 4.4 e 4.5 mostram os resultados das simulações do processamento do bloco da Figura 4.1(a) para as configurações Serial e Ideal, respectivamente. Nesse caso não foi realizada a simulação utilizando a Configuração Paralela, visto que todas as colunas desse bloco possuem o mesmo tamanho. Logo, não há como processá-las em paralelo sem aumentar o número de módulos de processamento da 1D-DCT de tamanho 2.

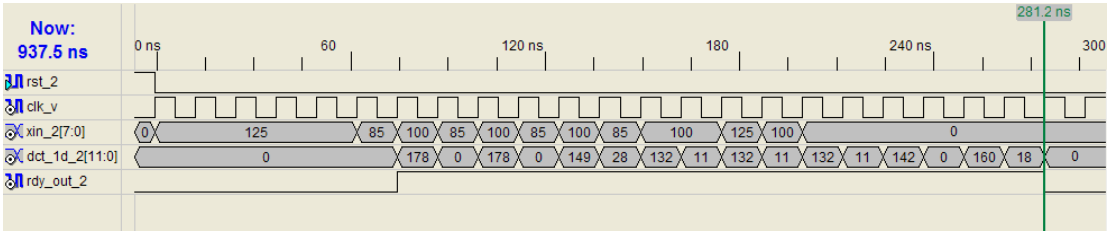


Figura 4.4: Resultado da simulação para o bloco de borda da Figura 4.1(a), utilizando a Configuração Serial

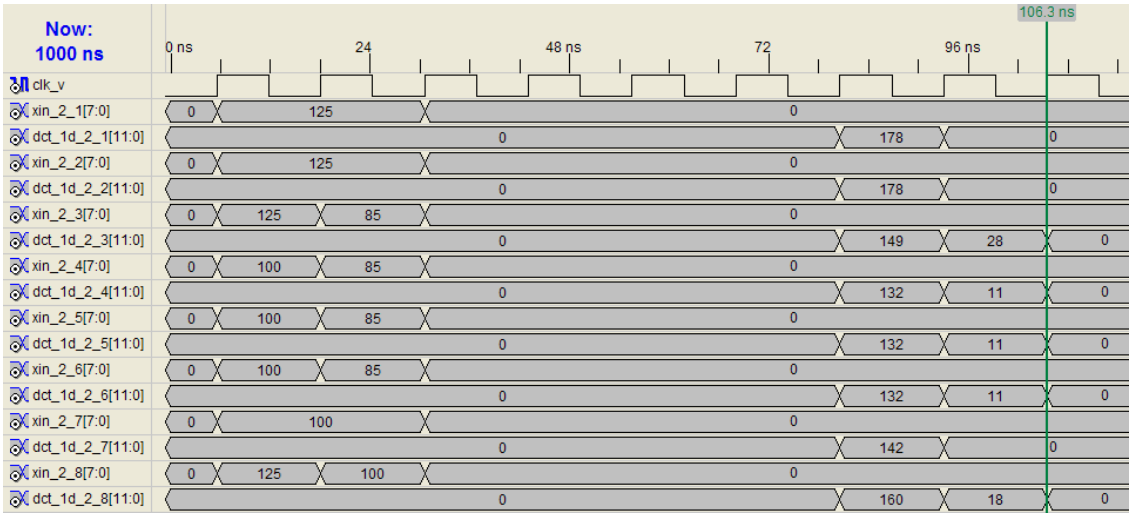


Figura 4.5: Resultado da simulação para o bloco de borda da Figura 4.1(a), utilizando a Configuração Ideal

No resultado da simulação da Configuração Serial, é importante observar que, inicialmente, entra o primeiro pixel da primeira coluna do bloco, seguido do segundo pixel da primeira coluna. Os pixels da segunda coluna do bloco só começam a chegar depois que todos os pixels da primeira coluna já estão sendo processados e, assim, sucessivamente. Nessa configuração os pixels são representados por *xin_L*, em que L representa o comprimento da coluna. Quando os coeficientes (dct_1d_L) começam a chegar, o sinal de habilitação (*rdy_out_L*) troca o seu nível lógico de 0 para 1 e permanece no nível lógico 1, enquanto estiver chegando dados válidos, caso contrário, o seu nível retorna para 0. Os sinais de resete (*rst_L*) e de habilitação (*rdy_out_L*) somente foram mostrados na simulação da Figura 4.4, porque a sua visualização tornaria as figuras muito grandes, de difícil entendimento, visto que existe um sinal de resete e um sinal de habilitação para cada módulo de processamento da 1D-DCT.

Na simulação referente à Configuração Ideal, apresentada na Figura 4.5, todos os primeiros pixels de cada coluna chegam simultaneamente na URP de processamento das DCTs-VS e o processamento desses pixels também é paralelo. Nessa configuração os segmentos verticais de pixels são representados por *xin_L_i*, em que L representa o comprimento da coluna e i ($1 \leq i \leq N$, para blocos de dimensão NxN) representa o número da coluna. Assim, no resultado da simulação, pixels de mesmo índice em cada coluna (representadas por *xin*_2_1, *xin*_2_2,..., *xin*_2_7, *xin*_2_8) chegam na URP de processamento das DCTs-VS simultaneamente. Os segmentos de coeficientes verticais ($dct\_1d\_2\_i$), resultantes da transformação 1D de cada coluna de tamanho 2, podem ser vistos nos sinais chamados de $dct\_1d\_2\_1$, $dct\_1d\_2\_2$,..., $dct\_1d\_2\_7$, $dct\_1d\_2\_8$.

Conforme pode-se verificar através dos resultados das simulações, o tipo de configuração não interfere nos valores finais da 1D-DCT (na seção seguinte será apresentado o erro médio quadrático obtido com esses resultados). A grande diferença está no tempo de processamento e na taxa de pixels por segundo alcançada por cada configuração. Na Configuração Serial, o tempo de processamento foi de 281,2 ns e na Configuração Ideal foi de 106,3 ns (menos da metade do tempo utilizado na Configuração Serial). Já a taxa de pixels por segundo (desprezando-se o tempo de latência) foi de $80 Mpixels/s$ na Configuração Serial e de $640 Mpixels/s$ na Configuração Ideal (oito vezes maior). Conforme era de se esperar, a taxa de pixels por segundo é diretamente proporcional ao grau de paralelismo das colunas.

Normalmente, ao se falar de desempenho da DCT, tende-se a considerar somente a taxa de pixels por segundo e a desconsiderar o tempo de latência. No caso específico deste trabalho, os módulos de processamento da 1D-DCT variam com o comprimento das colunas. Desse modo, optou-se por apresentar e analisar o tempo de processamento considerando o tempo de latência de cada módulo, conforme feito anteriormente, devido ao fato de que, em alguns casos, ao se processar colunas com comprimentos diferentes, ocorre um atraso significativo

entre os resultados de uma coluna e os da outra, isto é, não há um fluxo contínuo de pixels. Isso acontece, por exemplo, quando se processa uma coluna de tamanho 8, seguidamente a uma coluna de tamanho 2. Como a coluna de tamanho 2 possui um tempo de processamento muito pequeno, existe um atraso entre o último resultado da coluna de tamanho 2 e o primeiro coeficiente resultante da coluna de tamanho 8.

As Figuras 4.6 e 4.7 mostram os resultados das simulações do processamento do bloco de borda da Figura 4.1(b) para as Configurações Serial e Ideal, respectivamente. Nesse caso também não foi realizada a simulação utilizando a Configuração Paralela, visto que as duas colunas desse bloco possuem o mesmo tamanho e, portanto, a Configuração Paralela seria equivalente à Configuração Serial.

Na Configuração Serial, o tempo de processamento foi de 368,8 ns e na Configuração Ideal foi de 268,8 ns. Já a taxa de pixels por segundo foi de 80 Mpixels/s na Configuração Serial e de 160 Mpixels/s na Configuração Ideal, ou seja, duas vezes maior.

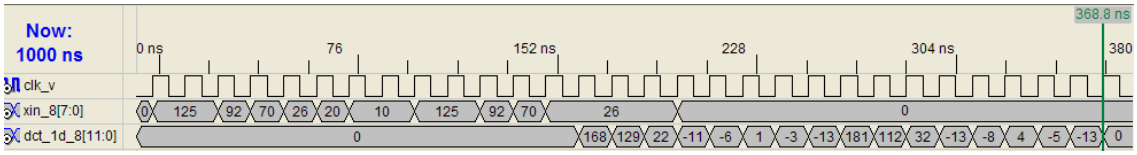


Figura 4.6: Resultado da simulação para o bloco de borda da Figura 4.1(b) utilizando a Configuração Serial

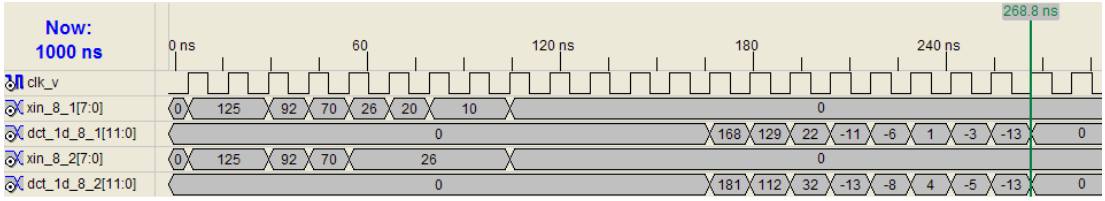


Figura 4.7: Resultado da simulação para o bloco de borda da Figura 4.1(b) utilizando a Configuração Ideal

Pode-se observar que, embora os dois primeiros blocos do Grupo 1, representados nas Figuras 4.1 (a) e (b) possuam a mesma quantidade de pixels (16 pixels), ao se aplicar a Configuração Ideal ao invés da Configuração Serial nos dois, observa-se uma redução no tempo de processamento e um aumento na taxa de pixels significativamente maior no bloco da Figura 4.1(a) do que no bloco da Figura 4.1(b). Isso ocorreu porque o bloco da Figura 4.1(b) possui uma possibilidade máxima de paralelismo igual a dois, isto é, não adianta utilizar mais de dois módulos de processamento, visto que o bloco possui apenas duas colunas de pixels pertencentes ao objeto. Já o bloco da Figura 4.1(a) apresenta a possibilidade de se utilizar um maior grau de paralelismo, visto que esse bloco possui seus pixels distribuídos num maior número de colunas.

As Figuras 4.8 e 4.9 mostram os resultados da simulação do bloco de borda da Figura 4.1(c) para a Configuração Serial e Paralela, respectivamente. Como nesse bloco cada coluna possui um tamanho distinto, não há diferença entre a Configuração Ideal e a Paralela, visto que, na Configuração Paralela, a URP de processamento das DCTs-VS possui um módulo para cada comprimento de coluna.

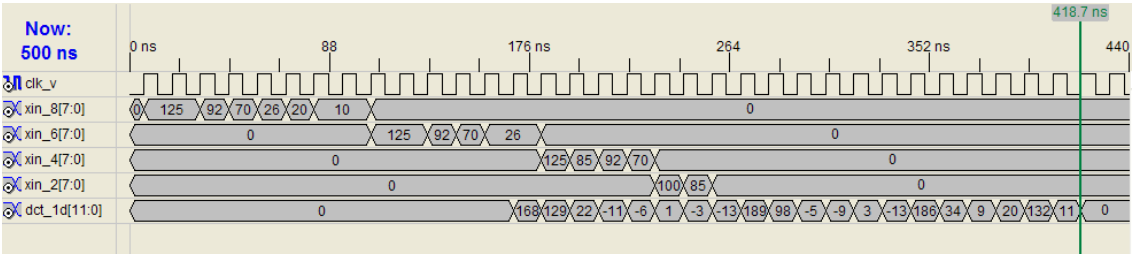


Figura 4.8: Resultado da simulação para o bloco de borda da Figura 4.1(c) utilizando a Configuração Serial

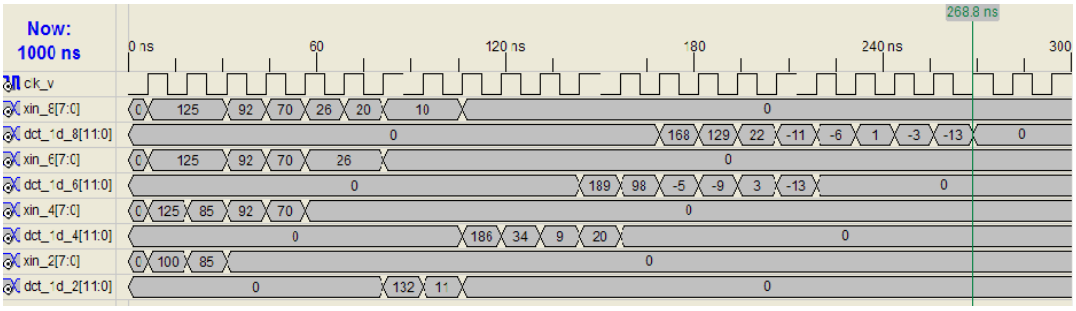


Figura 4.9: Resultado da simulação para o bloco de borda da Figura 4.1(c) utilizando a Configuração Paralela

Na Configuração Serial, obteve-se um tempo de processamento de 418,7 ns e na Configuração Paralela, de 268,8 ns. Já a taxa de pixels por segundo foi de 80 Mpixels/s na Configuração Serial e de 320 Mpixels/s na Configuração Paralela, ou seja, quatro vezes maior, conforme era de se esperar, tendo em vista que o grau de paralelismo aumentou quatro vezes.

As Figuras 4.10 e 4.11 mostram os resultados das simulações do bloco de borda da Figura 4.2(a) para as Configurações Serial e Ideal, respectivamente. Nesse caso, como todas as colunas possuem comprimento igual a 4, não existem diferenças entre os resultados obtidos nas Configurações Paralela e Serial. Em algumas simulações, optou-se por mostrar apenas o seu final, visto que, às vezes, o processamento de todos os pixels demanda um tempo elevado, o que torna a simulação muito extensa e de difícil visualização numa única tela.

Na Configuração Serial, obteve-se um tempo de processamento de 518,8 ns e, na Configuração Ideal, de 156,3 ns. Já a taxa de pixels por segundo foi de 80 Mpixels/s na Configuração Serial e de 640 Mpixels/s na Configuração Ideal, ou seja, oito vezes maior. Isso ocorreu porque,

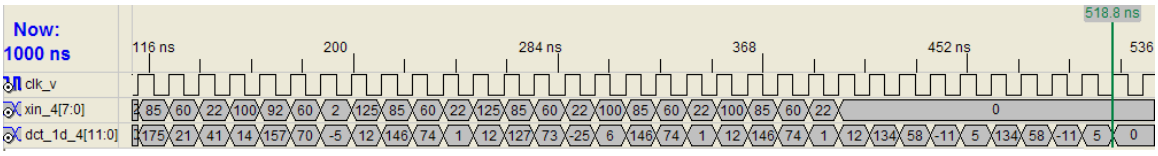


Figura 4.10: Resultado da simulação para o bloco de borda da Figura 4.2(a) utilizando a Configuração Serial

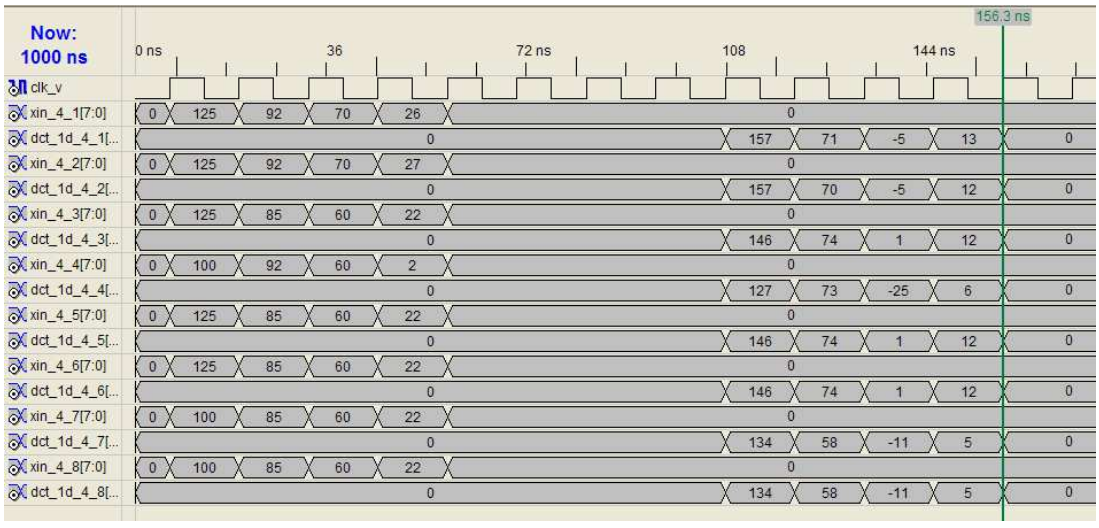


Figura 4.11: Resultado da simulação para o bloco de borda da Figura 4.2(a) utilizando a Configuração Ideal

na Configuração Ideal, utilizou-se uma URP de processamento das DCTs-VS com oito módulos que executam a DCT de tamanho 4 e, na Configuração Serial, a unidade de processamento possuía apenas um módulo que executava a DCT de tamanho 4, ou seja, o grau de paralelismo aumentou oito vezes.

As Figuras 4.12, 4.13 e 4.14 mostram os resultados das simulações do bloco de borda da Figura 4.2(b) para as Configurações Serial, Paralela e Ideal, respectivamente.

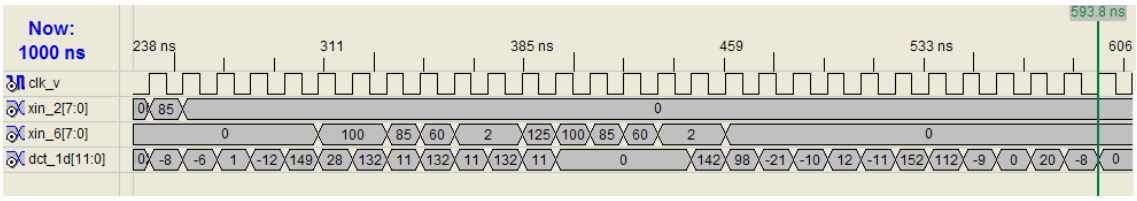


Figura 4.12: Resultado da simulação para o bloco de borda da Figura 4.2(b) utilizando a Configuração Serial

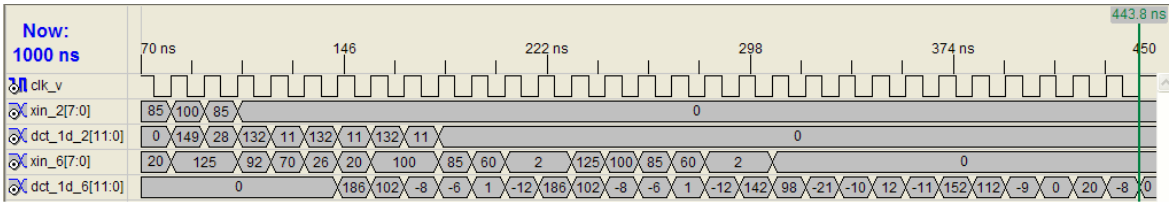


Figura 4.13: Resultado da simulação para o bloco de borda da Figura 4.2(b) utilizando a Configuração Paralela

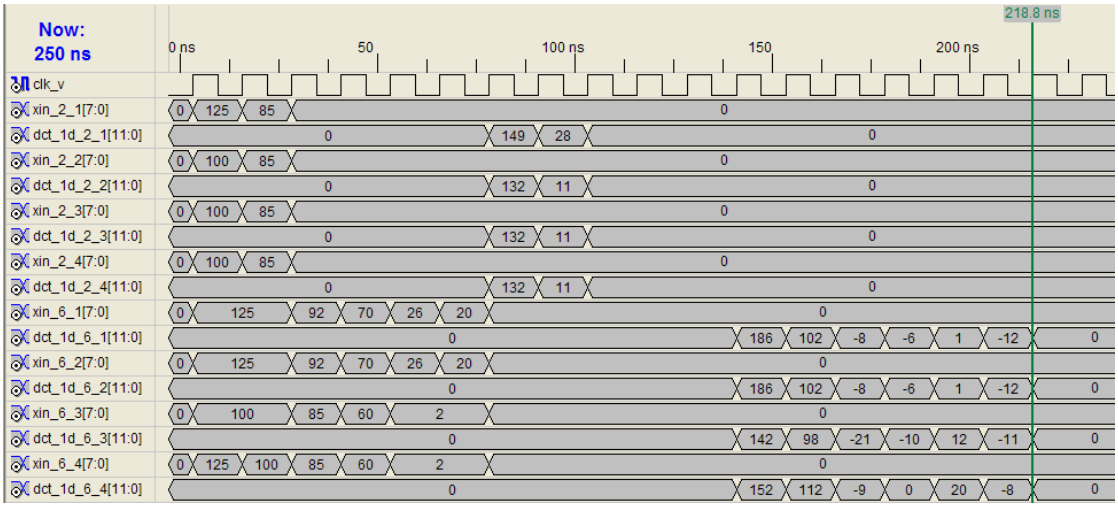


Figura 4.14: Resultado da simulação para o bloco de borda da Figura 4.2(b) utilizando a Configuração Ideal

Através de uma análise desses três resultados de simulação, verifica-se um ganho significativo no tempo de processamento e na taxa de pixels/segundo da Configuração Paralela em

relação à Configuração Serial e também da Configuração Ideal em relação às Configurações Paralela e Serial. Na Configuração Serial, obteve-se um tempo de processamento igual a 593,8 ns, na Configuração Paralela, o tempo de processamento foi de 443,8 ns e na Configuração Ideal, foi de 218,8 ns. Já a taxa de pixels por segundo foi de 80 Mpixels/s na Configuração Serial, 160 Mpixels/s na Configuração Paralela e de 640 Mpixels/s na Configuração Ideal, ou seja, quatro vezes maior que a Configuração Paralela e oito vezes maior que a Configuração Serial, conforme o aumento do grau de paralelismo.

As Figuras 4.15 e 4.16 mostram os resultados das simulações do bloco de borda da Figura 4.2(c) para as Configurações Serial e Paralela, respectivamente. Como nesse bloco cada coluna possui um tamanho diferente, a restrição de se ter apenas um módulo de processamento de cada tamanho L não altera o desempenho. Logo, não há diferença entre as Configurações Ideal e a Paralela.

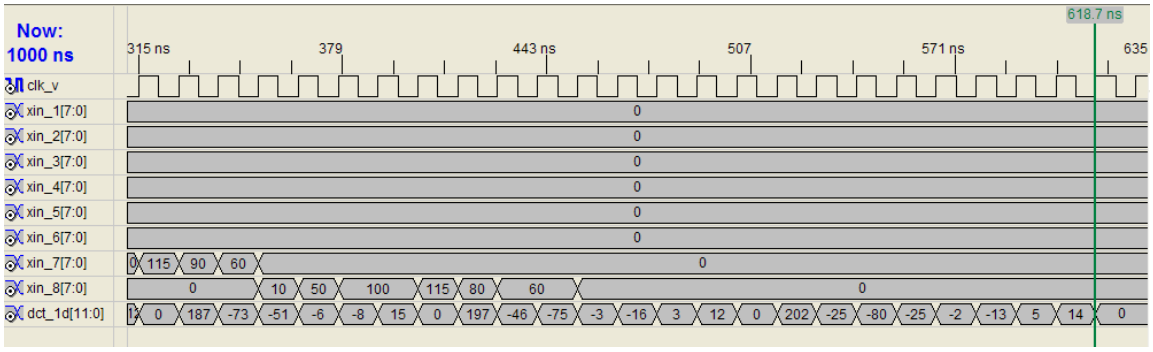


Figura 4.15: Resultado da simulação para o bloco de borda da Figura 4.2(c) utilizando a Configuração Serial

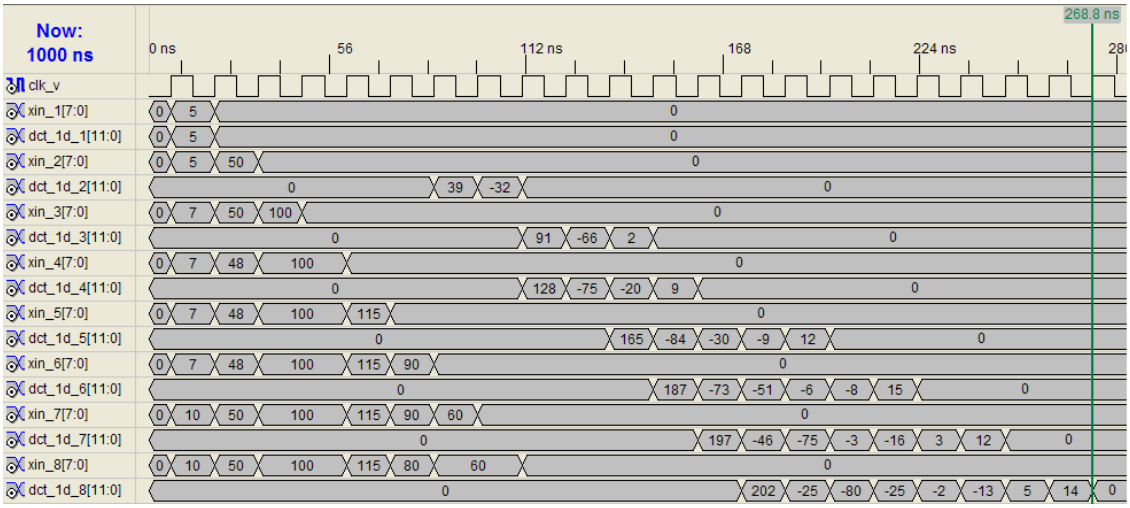


Figura 4.16: Resultado da simulação para o bloco de borda da Figura 4.2(c) utilizando a Configuração Paralela

As Figuras 4.17 e 4.18 mostram os resultados das simulações do bloco de borda da Figura 4.3(a) para as Configurações Serial e Ideal, respectivamente. Nesse caso também não foi realizada a simulação utilizando a Configuração Paralela, visto que todas as colunas desse bloco possuem o mesmo tamanho.

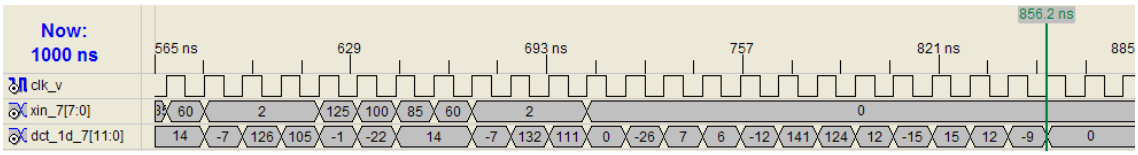


Figura 4.17: Resultado da simulação para o bloco de borda da Figura 4.3(a) utilizando a Configuração Serial

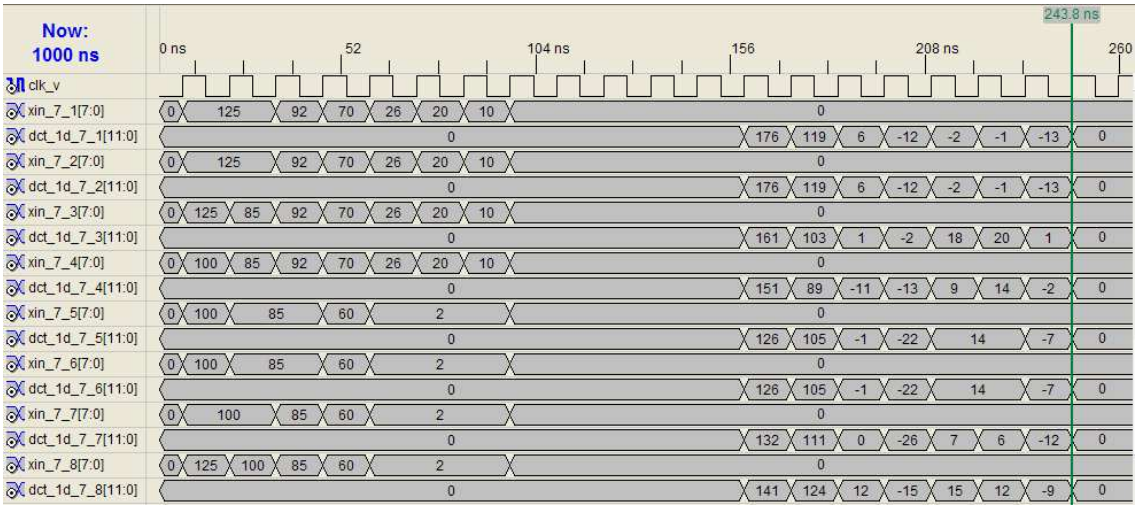


Figura 4.18: Resultado da simulação para o bloco de borda da Figura 4.3(a) utilizando a Configuração Ideal

Na Configuração Serial, obteve-se um tempo de processamento de 856,2 ns e na Configuração Ideal, de 243,8 ns. É importante notar que, quando comparamos o ganho obtido no tempo de processamento da Configuração Ideal em relação ao ganho da Configuração Serial para o bloco da Figura 4.3(a) e para o bloco da Figura 4.1(a), que possuía poucos pixels em cada coluna, observa-se que o ganho da Configuração Ideal em relação ao ganho da Configuração Serial é significativamente maior para o bloco da Figura 4.3(a). Logo, pode-se concluir que a utilização da Configuração Ideal é ainda mais indicada para blocos de borda que possuem muitas colunas com elevada quantidade de pixels do objeto e que, conseqüentemente, possuem um tempo de processamento maior. Contudo, com relação à taxa de pixels por segundo, não se observaram diferenças quando se variou a quantidade de pixels do objeto em cada coluna. Nota-se que a taxa de pixels por segundo, conforme mencionado anteriormente, varia apenas com a quantidade de colunas a serem processadas simultaneamente (grau de paralelismo). Para

o bloco de borda da Figura 4.3(a), obteve-se uma taxa de 80 Mpixels/s na Configuração Serial e de 640 Mpixels/s na Configuração Ideal, exatamente igual ao resultado da simulação do bloco da Figura 4.1(a).

As Figuras 4.19, 4.20 e 4.21 mostram os resultados das simulações do bloco de borda da Figura 4.3(b) para as Configurações Serial, Paralela e Ideal, respectivamente.

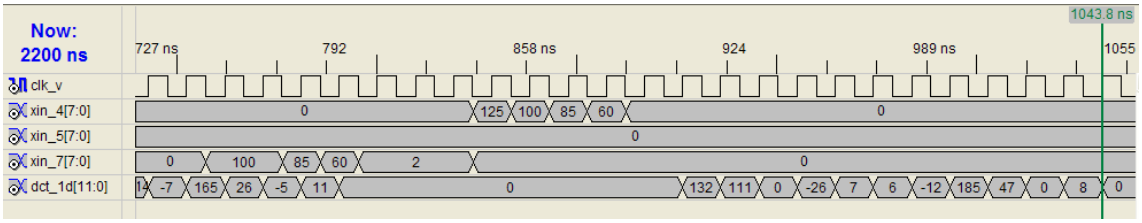


Figura 4.19: Resultado da simulação para o bloco de borda da Figura 4.3(b) utilizando a Configuração Serial

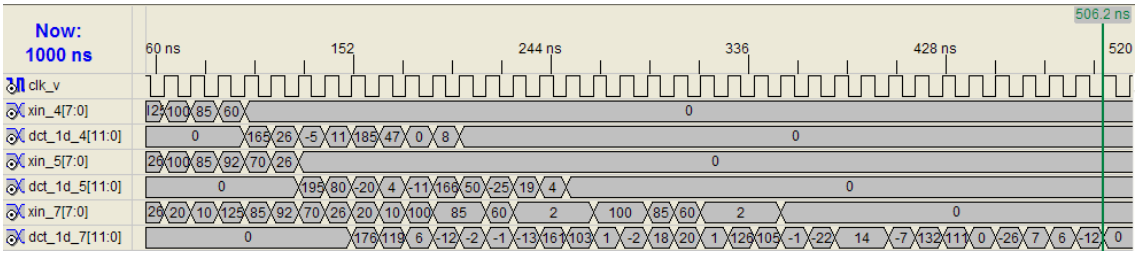


Figura 4.20: Resultado da simulação para o bloco de borda da Figura 4.3(b) utilizando a Configuração Paralela

Na Configuração Serial, obteve-se um tempo de processamento de 1043,8 ns, na Configuração Paralela, de 506,2 ns e na Configuração Ideal, de 243,8 ns. Já a taxa de pixels por segundo foi de 80 Mpixels/s na Configuração Serial, 240 Mpixels/s na Configuração Paralela e 640 Mpixels/s na Configuração Ideal. Através de uma análise desses três resultados de simulação, verifica-se um ganho significativo no tempo de processamento e na taxa de pixels/segundo da Configuração Ideal em relação às Configurações Paralela e Serial.

As Figuras 4.22, 4.23 e 4.24 mostram os resultados das simulações do bloco de borda da Figura 4.3(c) para as Configurações Serial, Paralela e Ideal. Obteve-se um tempo de processamento de 843,8 ns e uma taxa de 80 Mpixels/s na Configuração Serial, um tempo de 368,8 ns e uma taxa de 320 Mpixels/s na Configuração Paralela e um tempo de 268,7 ns e uma taxa de 640 Mpixels/s na Configuração Ideal.

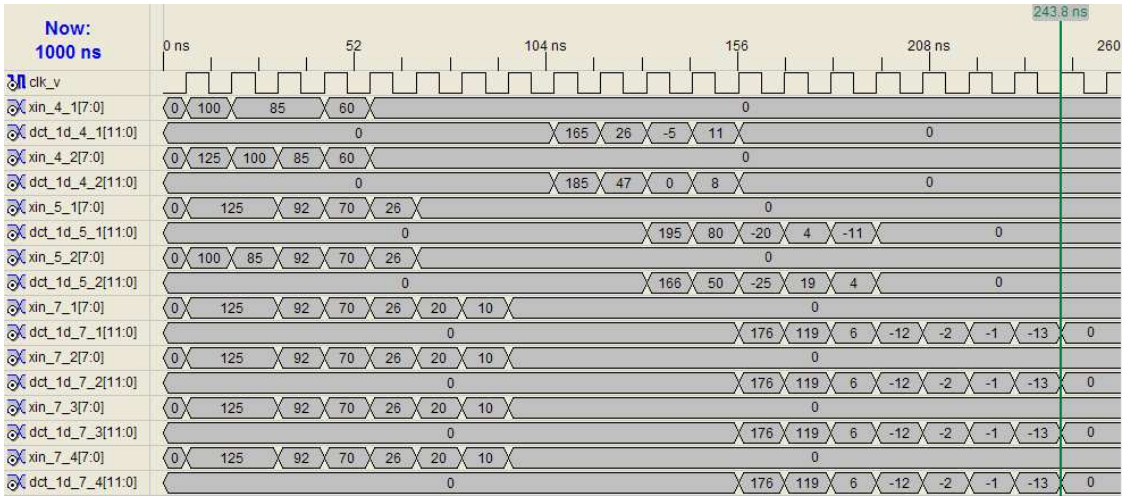


Figura 4.21: Resultado da simulação para o bloco de borda da Figura 4.3(b) utilizando a Configuração Ideal

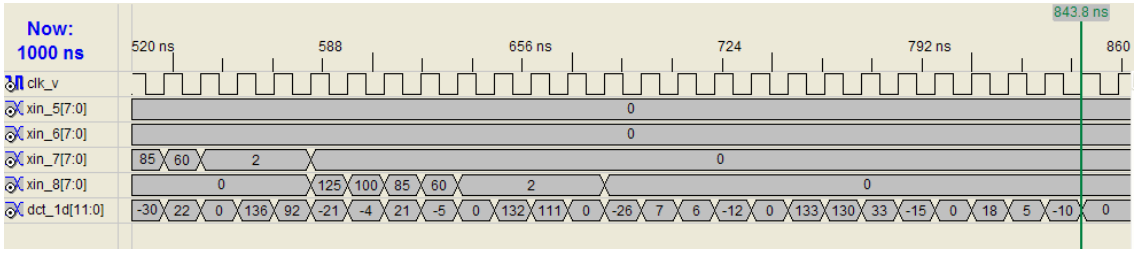


Figura 4.22: Resultado da simulação para o bloco de borda da Figura 4.3(c) utilizando a Configuração Serial

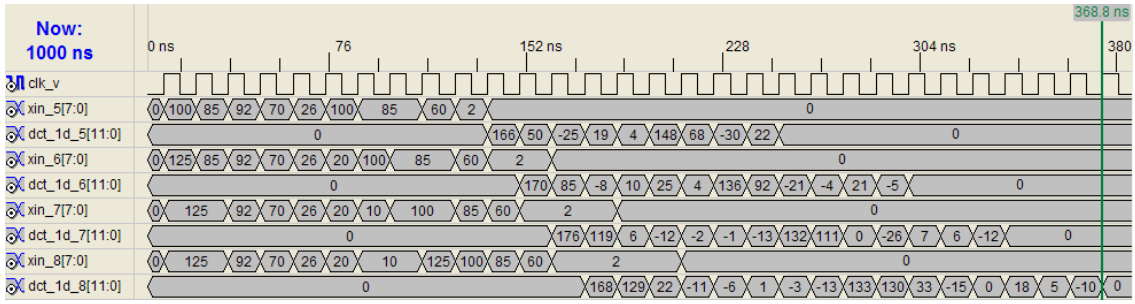


Figura 4.23: Resultado da simulação para o bloco de borda da Figura 4.3(c) utilizando a Configuração Paralela

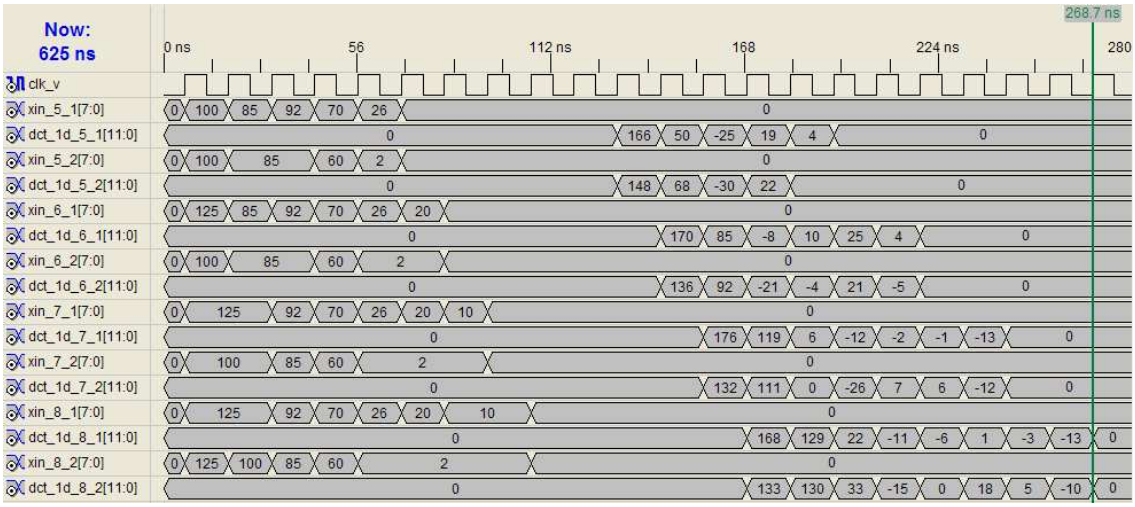


Figura 4.24: Resultado da simulação para o bloco de borda da Figura 4.3(c) utilizando a Configuração Ideal

4.3 Validação dos coeficientes gerados pela arquitetura da DCT-VS

Com o objetivo de validar todos os resultados obtidos na simulação da arquitetura reconfigurável, verificando a precisão dos módulos que implementam as 1Ds-DCTs, foi feito o cálculo do erro médio quadrático (EMQ) de todos os blocos de borda analisados, calculado conforme a seguinte fórmula:

$$EMQ = \frac{1}{M}\sum_{i}\sum_{j}(p_{ij} - \hat{p}_{ij})^2, \tag{4.1}$$

em que M representa o número de pixels do objeto no bloco e p_{ij} e $\hat{p_{ij}}$ são, respectivamente, o pixel original e o pixel reconstruído.

Para se calcular os pixels reconstruídos, aplicou-se a função "idct.m" do DSP Blockset[1] do Matlab aos coeficientes obtidos na simulação da arquitetura reconfigurável (representados com 12 bits). A Tabela 4.1 mostra o erro médio quadrático decorrente do processamento de cada bloco de borda dos três grupos, independente da configuração.

Através de uma análise dos resultados dessa tabela, pode-se afirmar que os módulos que implementam as DCTs estão corretos, visto que os valores de erro médio quadrático são decorrentes de truncamentos realizados na implementação em *hardware*. Esse erro ainda poderá ser reduzido, caso se aumente a quantidade de bits utilizada para calcular e representar os coeficientes. Esse parâmetro também pode ser alterado através da reconfiguração da arquitetura que implementa a SA-DCT, pois a variação dos números de bits é uma possibilidade importante da

[1]O DSP Blockset fornece um conjunto de funções para concepção, simulação e protótipos de sistemas de processamento digital de sinais.

Tabela 4.1: Erro médio quadrático decorrente da implementação da 1D-DCT

	Blocos	EMQ
1^o **grupo**	1 (a)	0,58
	1 (b)	0,34
	1 (c)	0,26
2^o **grupo**	2 (a)	0,10
	2 (b)	0,34
	2 (c)	0,30
3^o **grupo**	3 (a)	0,47
	3 (b)	0,30
	3 (c)	0,27

arquitetura proposta.

4.4 Análise quantitativa de desempenho

Conforme se verificou nos resultados de simulação apresentados e comentados na Seção 4.2, a utilização do paralelismo no processamento das colunas reduziu significativamente o tempo gasto na execução das 1Ds-DCTs de tamanho variável. As Tabelas 4.2 e 4.3 apresentam uma síntese dos tempos de processamento e das taxas de pixels, utilizando as três configurações propostas para cada tipo de bloco de borda.

Tabela 4.2: Tempo de processamento de cada bloco usando os três tipos de configuração

	Blocos	Config. Serial	Config. Paralela	Config. Ideal
1^o **grupo**	1 (a)	281,2 ns	281,2 ns	106,3 ns
	1 (b)	368,8 ns	368,8 ns	268,8 ns
	1 (c)	725 ns	268,8 ns	268,8 ns
2^o **grupo**	2 (a)	518,8 ns	518,8 ns	156,3 ns
	2 (b)	593,8 ns	443,8 ns	218,8 ns
	2 (c)	618,7 ns	268,8 ns	268,8 ns
3^o **grupo**	3 (a)	856,2 ns	856,2 ns	243,8 ns
	3 (b)	1043,8 ns	506,2 ns	243,8 ns
	3 (c)	843,8 ns	368,8 ns	268,7 ns

Através de uma análise dos tempos de processamento, verificou-se que quanto maior o número de colunas que apresentam o mesmo comprimento, maior é a redução no tempo de processamento ao se utilizar um maior grau de paralelismo. Na Configuração Ideal, por exemplo, consegue-se reduzir o tempo de processamento em até oito vezes. Esse ganho torna-se ainda mais representativo, quando consideramos os blocos de borda que possuem uma grande

Tabela 4.3: Taxa de pixels de cada bloco usando os três tipos de configuração

	Blocos	Config. Serial	Config. Paralela	Config. Ideal
1^o **grupo**	1 (a)	80 Mpixels/s	80 Mpixels/s	640 Mpixels/s
	1 (b)	80 Mpixels/s	80 Mpixels/s	160 Mpixels/s
	1 (c)	80 Mpixels/s	320 Mpixels/s	320 Mpixels/s
2^o **grupo**	2 (a)	80 Mpixels/s	80 Mpixels/s	640 Mpixels/s
	2 (b)	80 Mpixels/s	160 Mpixels/s	640 Mpixels/s
	2 (c)	80 Mpixels/s	640 Mpixels/s	640 Mpixels/s
3^o **grupo**	3 (a)	80 Mpixels/s	80 Mpixels/s	640 Mpixels/s
	3 (b)	80 Mpixels/s	240 Mpixels/s	640 Mpixels/s
	3 (c)	80 Mpixels/s	320 Mpixels/s	640 Mpixels/s

quantidade de pixels do objeto, uma vez que esses blocos possuem um tempo de latência e um tempo de processamento maior. Isso pode ser visualizado claramente na Tabela 4.4, que relacionou o *speed up* das Configurações Paralela e Ideal em relação à Configuração Serial.

Tabela 4.4: *Speed up* das configurações Paralela e Ideal em relação a Configuração Serial

	Blocos	Config. Serial / Config. Paralela	Config. Serial / Config.Ideal
1^o **grupo**	1 (a)	1	2,6
	1 (b)	1	1,4
	1 (c)	2,7	2,7
2^o **grupo**	2 (a)	1	3,3
	2 (b)	1,7	3,5
	2 (c)	4,8	4,8
3^o **grupo**	3 (a)	1	3,5
	3 (b)	3,2	6,7
	3 (c)	4,5	6,2

4.5 Representações estatísticas

Pela impossibilidade de se utilizar imagens reais, visto que foram simuladas configurações para apenas nove tipos de blocos de borda diferentes, optou-se por criar representações estatísticas para se avaliar o ganho de desempenho obtido com as configurações simuladas. Para formar essas representações, utilizou-se a metodologia do planejamento de experimentos (GOUPY, J., 1988), criando agrupamentos de blocos de borda nos quais se predominassem blocos com determinadas características. Cada uma dessas representações foi constituída por 300 blocos de borda, tendo sido utilizadas seis representações distintas, com as seguintes características:

- Representação 1: maioria dos blocos com poucos pixels do objeto concentrados em poucas colunas;
- Representação 2: maioria dos blocos com poucos pixels do objeto distribuídos em várias colunas;
- Representação 3: maioria dos blocos com metade dos pixels do objeto concentrados em poucas colunas;
- Representação 4: maioria dos blocos com metade dos pixels do objeto distribuídos em várias colunas;
- Representação 5: maioria dos blocos com muitos pixels do objeto concentrados em poucas colunas;
- Representação 6: maioria dos blocos com muitos pixels do objeto distribuídos em várias colunas.

A Tabela 4.5 apresenta a quantidade percentual dos vários tipos de blocos de borda em cada representação estatística.

Tabela 4.5: Constituição das representações estatísticas

		Representações					
	Bloco de borda	**1**	**2**	**3**	**4**	**5**	**6**
1^o **grupo**	1 (a)	80%	0	5%	5%	5%	5%
	1 (b)	0	0	0	0	0	0
	1 (c)	0	80%	5%	5%	5%	5%
2^o **grupo**	2 (a)	5%	5%	80%	0	5%	5%
	2 (b)	0	0	0	0	0	0
	2 (c)	5%	5%	0	80%	5%	5%
3^o **grupo**	3 (a)	5%	5%	5%	5%	80%	0
	3 (b)	0	0	0	0	0	0
	3 (c)	5%	5%	5%	5%	0	80%

A Tabela 4.6 mostra o tempo aproximado, em milisegundos, gasto em cada configuração, no processamento dos 300 blocos de cada representação estatística. Esse tempo foi obtido por estimação, através de uma média aritmética ponderada dos resultados obtidos na simulação independente de cada bloco de borda.

A Tabela 4.7 mostra o *speed up* da Configuração Paralela e Ideal, em relação à Configuração Serial, para cada representação estatística. Através de uma análise dessa tabela, pode-se observar, para a Configuração Paralela, um ganho maior nas Representações 2, 4 e 6, as quais

Tabela 4.6: Tempo de processamento para cada representação estatística em ms

	Representações					
Configurações	**1**	**2**	**3**	**4**	**5**	**6**
Serial	81	93	123	135	187,5	175,5
Paralela	67,8	34,8	112,9	30,4	178,8	49,8
Ideal	10,1	19,1	15,8	17,3	23,9	22,4

tinham como característica principal a maioria de seus blocos com grande quantidade de colunas de tamanhos diferentes. Já para as representações estatísticas que possuíam a maioria dos blocos com colunas de mesmo tamanho, não se observou um ganho significativo no desempenho da Configuração Paralela em relação à Serial. Nesse caso o ganho obtido foi de aproximadamente 1,11, conforme era de se esperar, visto que não houve um aumento significativo do grau de paralelismo da Configuração Paralela em relação a Configuração Serial. Através dessa tabela também pode-se observar um ganho na Configuração Ideal em relação a Configuração Serial praticamente igual a oito no processamento dos blocos que possuem oito colunas de pixels. Conforme mencionado anteriormente, pode-se concluir que o ganho da Configuração Ideal em relação à Serial é proporcional à quantidade de colunas (independente dos seus comprimentos) da maioria dos blocos que constituem a representação. Na Representação 2, por exemplo, o ganho de 4,8 aproxima-se da quantidade de colunas (quatro) da maioria dos seus blocos (80 % dos blocos).

Tabela 4.7: *Speed up* das Configurações Paralela e Ideal em relação a Serial para cada representação estatística

	Representações					
	1	**2**	**3**	**4**	**5**	**6**
Config. Serial / Config. Paralela	1,19	2,67	1,09	4,44	1,05	3,52
Config. Serial / Config. Ideal	8	4,86	7,76	7,78	7,84	7,83

4.6 Consumo de recursos lógicos

A redução do tempo de processamento é alcançada em detrimento de um aumento dos recursos lógicos utilizados. As Tabelas 4.8, 4.9 e 4.10 mostram a quantidade de recursos lógicos consumidos por cada tipo de bloco de borda nas Configurações Serial, Paralela e Ideal, respectivamente. Essas tabelas foram construídas com base nos relatórios gerados pela ferramenta de síntese do ambiente de desenvolvimento da Xilinx.

Pode-se verificar, através de uma análise das Tabelas 4.8 e 4.9, que a Configuração Serial consumiu a mesma quantidade de recursos lógicos da Configuração Paralela e esta última apresentou, para alguns tipos de blocos, um desempenho muito melhor. Assim, é possível concluir que a Configuração Serial não oferece nenhuma vantagem em relação às outras duas configurações. É importante notar também que a Configuração Ideal, embora tenha reduzido consideravelmente o tempo de processamento da DCT, apresentou um consumo de recursos lógicos significativamente maior. Logo, é necessário analisar cada aplicação, antes de escolher a configuração do *hardware*. No caso de aplicações como videoconferência e controle de veículos pilotados remotamente em aplicações militares e espaciais, que requerem alto desempenho (visto que o processamento das imagens normalmente deve ocorrer em tempo real) justifica-se a utilização da Configuração Ideal. Já para aplicações que não demandam altas taxas de processamento, mas que priorizam custo de implementação, a Configuração Paralela é mais apropriada, visto que essa configuração consumiu a mesma quantidade de recursos lógicos que a Configuração Serial.

Tabela 4.8: Recursos lógicos consumidos na Configuração Serial

Bloco	***Slices***[2]	***Flip Flops***	**LUTs**	**IOBs**	**Multiplexadores**	***Clocks***
Grupo 1 (a)	112	114	150	23	1	1
Grupo 1 (b)	328	384	402	23	4	1
Grupo 1 (c)	896	1009	1126	89	10	1
Grupo 2 (a)	182	199	238	23	2	1
Grupo 2 (b)	384	425	486	45	4	1
Grupo 2 (c)	1642	1902	2067	177	19	1
Grupo 3 (a)	319	377	421	23	4	1
Grupo 3 (b)	753	873	982	67	9	1
Grupo 3 (c)	1171	1370	1483	89	14	1

[2]Um *slice* na família Spartan 3 da Xilinx é formado por duas LUTs e dois elementos de armazenamento dedicado (flip-flops ou latches). O termo *slice* é utilizado no projeto de FPGAs para facilitar a escolha do dispositivo com base na demanda de recursos lógicos do projeto.

Tabela 4.9: Recursos lógicos consumidos na Configuração Paralela

Bloco	*Slices*	*Flip Flops*	LUTs	IOBs	Multiplexadores	*Clocks*
Grupo 1 (a)	112	114	150	23	1	1
Grupo 1 (b)	328	384	402	23	4	1
Grupo 1 (c)	896	1009	1126	89	10	1
Grupo 2 (a)	182	199	238	23	2	1
Grupo 2 (b)	384	425	486	45	4	1
Grupo 2 (c)	1642	1902	2067	177	19	1
Grupo 3 (a)	319	377	421	23	4	1
Grupo 3 (b)	753	873	982	67	9	1
Grupo 3 (c)	1171	1370	1483	89	14	1

Tabela 4.10: Recursos lógicos consumidos na Configuração Ideal

Bloco	*Slices*	*Flip Flops*	LUTs	IOBs	Multiplexadores	*Clocks*
Grupo 1 (a)	892	912	1200	177	8	1
Grupo 1 (b)	656	768	804	45	8	1
Grupo 1 (c)	896	1009	1126	89	10	1
Grupo 2 (a)	1459	1592	1904	177	16	1
Grupo 2 (b)	1536	1700	1944	177	16	1
Grupo 2 (c)	1642	1902	2067	177	19	1
Grupo 3 (a)	2552	3016	3368	177	32	1
Grupo 3 (b)	2145	2500	2806	177	26	1
Grupo 3 (c)	2343	2740	2966	177	28	1

4.7 Considerações finais

Através de uma análise dos resultados, pode-se observar um ganho significativo de desempenho ao utilizar-se a Configuração Ideal para blocos que possuem grande quantidade de colunas de mesmo tamanho. Entretanto, verifica-se que essa configuração apresenta um consumo de recursos lógicos significativamente maior, assim, faz-se necessário analisar os requisitos de cada aplicação, antes de escolher a configuração do *hardware*.

Analisando as tabelas de *speed up* e de consumo de recursos lógicos, pode-se concluir que a Configuração Paralela oferece vantagens em relação à Serial em todos os aspectos, uma vez que ela apresenta maior desempenho em alguns tipos de blocos e consome a mesma quantidade de recursos lógicos da Configuração Serial.

A computação reconfigurável torna a arquitetura mais flexível (adaptável para cada tipo de imagem e requisitos da aplicação), visto que ela permite alterar algumas de suas características, como a quantidade total de módulos de processamento de cada tamanho ou a quantidade de bits utilizados no cálculo e na representação dos coeficientes. A computação reconfigurável também possibilita uma maior tolerância a falhas no circuito digital, pois no caso de defeito em alguma parte do dispositivo, o circuito pode ser implementado em outra área do chip que não esteja realizando nenhum processamento. Além disso, as técnicas de reconfigurabilidade possibilitam reduzir o custo de prototipação, uma vez que, no desenvolvimento do projeto, o mesmo *hardware* pode suportar diferentes versões do produto.

5 CONCLUSÕES

Neste capítulo, inicialmente, apresenta-se a discussão dos resultados obtidos, comparando os possíveis ganhos alcançados com a utilização da arquitetura proposta aos ganhos obtidos nos trabalhos relacionados. Feito isso, comenta-se o alcance dos objetivos e das metas propostos no início da pesquisa. Em seguida, são apresentadas as principais contribuições desta dissertação. E, por fim, são listados alguns possíveis trabalhos futuros.

5.1 Discussão dos resultados

Este trabalho propôs uma arquitetura de *hardware* reconfigurável paralela dedicada para a implementação da SA-DCT. Essa arquitetura foi projetada com o objetivo de viabilizar a implementação dessa técnica, uma vez que ela proporciona maior PSNR do que a 2D-DCT implementada em conjunto com métodos de extrapolação, para as médias e as altas taxas de compressão e para blocos pouco preenchidos por pixels do objeto (independente da taxa).

A utilização de técnicas de paralelismo possibilitou uma redução no tempo de processamento dos dados e a utilização de técnicas de reconfigurabilidade tornou o algoritmo da SA-DCT mais flexível no que diz respeito ao tempo de processamento, à demanda de recursos lógicos e à qualidade da imagem. No Capítulo 3, essa arquitetura foi descrita em detalhes, mencionando como deveria ser a sua implementação, bem como o funcionamento das unidades responsáveis por implementar cada etapa do algoritmo da SA-DCT. Com o objetivo de validar a arquitetura proposta, optou-se por implementar e simular a unidade da SA-DCT responsável pela execução das DCTs de tamanho variável, porque, no algoritmo da SA-DCT, essa unidade é a que possui maior complexidade computacional e a que demanda maior tempo de processamento.

Os resultados de simulação da URP de processamento das DCTs-VS comprovam o aumento de desempenho, flexibilidade e tolerância a falhas obtidos na implementação do algoritmo da SA-DCT utilizando a arquitetura proposta. É importante enfatizar que os conceitos e técnicas de computação reconfigurável e paralelismo utilizados na implementação dessa unidade também deverão ser utilizados na implementação das outras unidades.

Assim, considerando a arquitetura de *hardware* reconfigurável paralela proposta e projetada para a implementação da SA-DCT (descrita no Capítulo 3) e os resultados obtidos (analisados no Capítulo 4), pode-se concluir que os objetivos propostos foram alcançados. Dessa forma, consideram-se os resultados satisfatórios, conforme apresentados no decorrer da pesquisa.

O trabalho que mais se aproximou da arquitetura proposta nesta dissertação foi a arquite-

tura desenvolvida por (GAUSE, J.; CHEUNG, P. Y. K.; LUK, W., 2000). Gause propôs duas arquiteturas reconfiguráveis para implementar a SA-DCT, uma estática e outra dinâmica. Entretanto, as duas implementam uma única 1D-DCT de cada vez. O trabalho (GAUSE, J.; CHEUNG, P. Y. K.; LUK, W., 2000) não aliou as vantagens da computação reconfigurável às técnicas de paralelismo, as quais são as principais responsáveis pela diminuição do tempo de processamento. Em seu trabalho (GAUSE, J.; CHEUNG, P. Y. K.; LUK, W., 2004), Gause menciona o problema referente ao elevado tempo de reconfiguração dos FPGAs existentes. Esse tempo torna-se bastante crítico para Gause, porque a sua arquitetura que utiliza reconfiguração dinâmica configura um módulo de processamento para determinada coluna de cada vez. Assim, para realizar o processamento de uma SA-DCT completa, Gause precisa realizar 16 reconfigurações.

Neste trabalho, a arquitetura dinâmica proposta difere da proposta por Gause (GAUSE, J.; CHEUNG, P. Y. K.; LUK, W., 2000), porque inicialmente a arquitetura recebe a informação de forma (mapa de bits), enviada através de 64 bits no caso de blocos de tamanho 8x8 (8 bits para cada coluna enviados em paralelo) e, em seguida, configura-se a URP de processamento das DCTs-VS que realizará o processamento de todos os segmentos verticais. Feito isso, realiza-se o deslocamento dos segmentos verticais para a borda esquerda, calcula-se o comprimento de cada linha e então configura-se a outra URP de processamento das DCTs-VS que fará o processamento dos segmentos horizontais. Assim, o tempo de reconfiguração deixa de ser um gargalo pois, ao invés de se ter 16 reconfigurações consecutivas, tem-se apenas duas reconfigurações e essas poderão ser realizadas, enquanto se estiver processando outro bloco, utilizando, nesse caso, os recursos da reconfiguração parcial.

A outra arquitetura desenvolvida por Gause (GAUSE, J.; CHEUNG, P. Y. K.; LUK, W., 2004), onde ele propôs modelos simples para representar objetos de forma arbitrária e a seguir mapeá-los dentro de projetos de *hardware* específicos para aquele objeto, aproxima-se da arquitetura proposta neste trabalho, nos casos de se utilizar reconfiguração estática. Entretanto, em nenhuma das duas arquiteturas propostas por Gause, utilizaram-se técnicas de paralelismo em conjunto com computação reconfigurável. Assim não foi possível comparar o desempenho obtido utilizando a arquitetura proposta neste trabalho com as arquiteturas propostas por Gause.

A arquitetura estática da SA-DCT proposta por Gause utiliza uma freqüência de 47 MHz com uma vazão de um valor a cada dois ciclos de clock, ou seja, uma taxa de 24 $Mpixels/s$. Já a arquitetura proposta para a SA-DCT deste trabalho possui uma vazão de um pixel a cada ciclo de *clock* e foi implementada utilizando um *clock* de 80 MHz, alcançando uma taxa de até 640 Mpixels/s. Caso se utilizasse uma freqüência de 47 MHz, conforme Gause, obter-se-ia uma taxa de até 192 $Mpixels/s$, ainda assim oito vezes maior que a obtida por Gause.

Como não se encontrou, na literatura pesquisada, trabalhos que utilizassem computação reconfigurável em conjunto com técnicas de paralelismo, visando a redução do tempo de pro-

cessamento da SA-DCT, não foi possível estabelecer comparações entre a arquitetura proposta e as outras arquiteturas existentes na literatura. Pode-se verificar apenas que, em relação à arquitetura tradicional da SA-DCT, obteve-se um ganho significativo, conforme verificou-se no Capítulo 4.

É importante ressaltar também que a maioria dos trabalhos encontrados na literatura estudam o algoritmo da 2D-DCT, são poucos os que estudam e buscam otimizar o algoritmo da SA-DCT. Talvez, justamente, por esse algoritmo apresentar um elevado tempo de processamento, sendo inviável a sua implementação, sem a utilização de técnicas de paralelismo.

5.2 Contribuições

As principais contribuições desta dissertação são apresentadas a seguir:

- Proposta e projeto da arquitetura de *hardware* reconfigurável paralela dedicada para a implementação da SA-DCT em um bloco NxN, possibilitando reduzir o tempo de processamento e aumentar a flexibilidade e a tolerância a falhas no circuito digital;
- Implementação e simulação da unidade da SA-DCT que implementa as DCTs de tamanho variável, utilizando uma linguagem de descrição de *hardware* (HDL) no ambiente de simulação da Xilinx;
- Análise do desempenho obtido com as diferentes configurações propostas para a URP de processamento das DCTs-VS;
- Preparação e submissão de artigos em congressos nacionais, internacionais e em periódicos das diferentes áreas relacionadas à pesquisa.

5.3 Trabalhos futuros

Esta pesquisa abre possibilidades para a realização de diversos trabalhos futuros, visto que na literatura existem poucos trabalhos que utilizam técnicas de codificação de imagens por transformadas empregando conceitos de paralelismo e computação reconfigurável, principalmente no que diz respeito às técnicas de reconfiguração dinâmica e parcial. Além disso, as pesquisas nessas áreas ainda são muito recentes nos trabalhos desenvolvidos no Brasil e no mundo. Dentre as propostas de realização de novos trabalhos, destacam-se:

- Implementação e simulação da URP de detecção dos pixels pertencentes ao objeto, utilizando reconfiguração estática;

- Implementação e simulação da URP de alinhamento dos pixels na borda superior, utilizando reconfiguração estática;
- Implementação e simulação da URP de alinhamento dos coeficientes na borda esquerda, utilizando reconfiguração estática;
- Implementação e simulação de todas as unidades da arquitetura da SA-DCT proposta, utilizando técnicas de reconfiguração parcial e dinâmica;
- Análise de desempenho utilizando reconfiguração dinâmica e parcial na implementação da arquitetura proposta para a SA-DCT.

REFERÊNCIAS

ACTEL. Em http://www.actel.com/. *Acessado em: 05 de abril*, 2008.

AHMED, N.; NATRAJAN, T.; RAO, K. R. **Discrete Cosine Transform**. *IEEE Transactions on Computing*, C-23, n. 1, p. 90–93, 1984.

ALTERA. Em http://www.altera.com/. *Acessado em: 01 de maio*, 2008.

AMARAL, A. M.; MARTINS, C. A. P. S. **HRA-MPSoC: Arquitetura Hierarquicamente Reconfigurável de Sistemas Multiprocessados em Chip.** *VIII Workshop em Sistemas Computacionais de Alto Desempenho. Porto Alegre : Sociedade Brasileira de Computação*, p. 137–144, 2007.

ARAI, Y.; AGUI, T.; NAKAJIMA, M. **A Fast DCT-SQ Scheme for Images**. *Transactions of IEICE*, E71, n. 11, p. 1095–1097, 1988.

ASHENDEN, P. J. **The Student's Guide to VHDL**. [S.l.]: Morgan Kaufmann Publishers, Inc, San Francisco, California, 1998.

ATMEL. Em http://www.atmel.com/. *Acessado em: 10 de novembro*, 2007.

BHASKARAN, V.; KONSTANTINIDES, K.; NAKAJIMA, M. **Image and Video Compression Standards Algorithms and Architectures**. [S.l.]: Massachusetts: Kluwer Academic Publishers, 2nd ed, 1999.

BOON, C. S.; TAKAHASHI, J.; KADONO, S. **Experimental Results of Content Based Texture Coding**. *ISO/IEC JTC1/SC29/WG11 Doc.MPEG96/1029*, 1996.

BORMANS, J.; HILL, K. **MPEG-21 Overview (ISO/IEC JTC1/SC29/WG11/N5231).** ***Organization internationale de normalisation ISO/IEC JTC1/SC29/WG11 coding of moving pictures and audio*, 2002.**

BROWN, S.; ROSE, J. **Architecture of FPGAs and CPLDs: A Tutorial**. *IEEE Transactions on Design and Test of Computers*, v. 13, n. 2, p. 42–57, 1996.

CARNEIRO, C. A. et al. **A Real-Time and Parametric Parallel Video Compression Architecture Using FPGA**. *The International Symposium on Parallel and Distributed Processing and Application (ISPA)*, p. 758–768, 2006.

CHEN, W. H.; SMITH, C. H.; FRALICK, S. C. **A Fast Computational Algorithm for the Discrete Cosine Transform**. *IEEE Transactions on Circuits and Systems*, v. 9, p. 1004–1009, 1991.

COMPTON, K.; HAUCK, S. **Reconfigurable Computing: A Survey of Systems and Software**. *ACM Computing Surveys*, v. 34, n. 1, 2002.

FERREIRA, F. M. F. **Contribuições às abordagens SA-DCT e DCT Baseada em Blocos para codificação de Imagens Orientadas por Objeto**. *Tese de Doutorado, Departamento de Engenharia Elétrica - PUC Rio*, 2004.

FRANKE, U.; MESTER, R.; AACH, T. **Constrained iterative restoration techniques: A powerful tool in region oriented texture coding**. *ISO/IEC JTC1/SC29/WG11 Doc.MPEG96/1029 Signal Processing IV: Theories and Application*, p. 1145–1148, 1988.

GAUSE, J.; CHEUNG, P. Y. K.; LUK, W. **Static and dynamic reconfigurable designs of a 2D shape-adaptive DCT**. *Lectures Notes in Computer Science*, p. 96–105, 2000.

GAUSE, J.; CHEUNG, P. Y. K.; LUK, W. **Reconfigurable Computing for Shape-Adaptive Video Processing**. *IEE Proceedings - Computers and Digital Techniques*, v. 151, n. 5, p. 313–320, 2004.

GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing**. [S.l.]: Addison-Wesley, Inc, 1993.

GOUPY, J. **La Méthode des Plans D'expériences**. [S.l.]: Paris: Dunod, 1988.

HABIBI, A. **Hybrid Coding of Pictorial Data**. *IEEE Trans. Comm.*, COM-22, n. 5, p. 614–624, 1974.

HAUCK, S.; DEHON, A. **Reconfigurable Computing: The Theory and Practice of FPGA-Based Computing**. *Morgan Kaufman*, 2008.

HERON, J.; TRAINOR, D.; WOODS, R. **Image Compression Algorithms Using Reconfigurable Logic**. *Signals, Systems and Computers Conference Record of the Thirty-First Asilomar Conference on*, v. 1, n. 1, p. 399–403, 1997.

HUFMAN, D. A. **A Method for the Construction of Minimum Redundancy Codes**. *Proc. IRE*, v. 40, n. 10, p. 1089–1101, 1952.

JAIN, A. **Fundamentals of Digital Image Processing**. [S.l.]: Prentice Hall. USA, 1989.

JTC1/SC29/WG11, I. **MPEG-4 Video Verification Model Version 11.0**. *Doc. N2172*, March 1998.

KAUP, A. **Object-Based Texture Coding of Moving Video in MPEG-4**. *IEEE Transactions on Circuits and Systems for Video Technology*, v. 9, n. 1, 1999.

KAUP, A.; AACH, T. **Segment-Oriented Coding of Textured Images Based on Successive Approximation**. In: . [S.l.: s.n.], 1994. v. 1, p. 197–200.

LE, T.; GLESNER, M. **Flexible Architectures for DCT of Variable-Length Targeting Shape-Adaptive Transform**. *IEEE Transactions on Circuits and Systems for Video Technology*, v. 10, n. 8, p. 1949–1952, 1999.

LE, T.; GLESNER, M. **Configurable VLSI-Architectures for both Standard DCT and Shape-Adaptive DCT in Future MPEG-4 Circuit Implementations**. *The IEEE International Symposium on Circuits and Systems, ISCAS*, v. 2, p. 461–464, 2000.

LINDE, Y.; BUZO, A.; GRAY, R. M. **An Algorithm for Vector Quantizer Design**. *IEEE Trans. Comm.*, COM-28, n. 1, p. 84–95, 1980.

MARTINS, C. A. P. S. et al. **Computação Reconfigurável: conceitos, tendências e aplicações**. In: *XXII Jornada de Atualização em Informática (JAI)*. Campinas, Brasil: [s.n.], 2003. v. 22, p. 339–388.

MARTÍNEZ, J. M. **MPEG-7 Overview (ISO/IEC JTC1/SC29/WG11/N6828).** ***Organization internationale de normalisation ISO/IEC JTC1/SC29/WG11 coding of moving pictures and audio*****, 2004.**

MOHAMED, T. S.; BADAWY, W. **Integrated Hardware-Software Platform for Image Processing Applications**. *IEEE International Workshop on System-on-Chip for Real-Time Applications*, p. 145–148, 2004.

MOHD-YUSOF, Z.; SULEIMAN, I.; ASPAR, Z. **Implementation of Two Dimensional Forward DCT and Inverse DCT using FPGA**. *TENCON 2000. Proceedings*, v. 3, p. 242–245, 2000.

PARK, J.; CHOI, J. H.; ROY, K. **Dynamic Bit Width Adaptation in DCT : Image Quality versus Computation Energy Trade off**. *Design and Test in Europe*, 2006.

PARK, J.; KWON, S.; ROY, K. **Low Power Reconfigurable DCT Design Based on Sharing Multiplication**. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Orlando, USA*, 2002.

PARK, J.; ROY, K. **A Low Power Reconfigurable DCT Architecture to trade off Image Quality for Computational Complexity**. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Montreal, Canada*, 2004.

PEREIRA, F.; EBRAHIMI, T. **The MPEG-4 Book**. [S.l.]: Prentice Hall, 2002.

ROESE, J. A.; PRATT, W. K.; ROBINSON, G. S. **Interframe Cosine Transform Image Coding**. *IEEE Trans. Comm.*, COM-25, p. 1329–1339, 1977.

SCHONER, B. et al. **Techniques for FPGA Implementation of Video Compression Systems**. *Third International ACM Symposium on Field-Programmable Gate Arrays (FPGA'95)*, p. 154–159, 1995.

SHEN, G.; ZENG, B.; LIOU, M. L. **A New Padding Technique for Coding of Arbitrarily-Shaped Image / Video Segments**. *IEEE International Conference on Image Processing (ICIP'99), Japan*, 1999.

SIKORA, T.; MAKAI, B. **Shape-Adaptive DCT for Generic Coding of Video**. *IEEE Transactions on Circuits and Systems for Video Technology*, v. 5, n. 1, p. 59–65, 1995.

SMITH, D. J. **HDL Chip Design: A practical Guide for Designing, Synthesizing and Simulating ASICs and FPGAs using VHDL or Verilog**. *Doone Publications*, 1997.

TANENBAUM, A. S. **Redes de Computadores**. [S.l.]: Elsevier, 2003.

TECHNOLOGY, M. Em http://www.model.com/. *Acessado em: 5 de novembro*, 2007.

TODMAN, T. J. et al. **Reconfigurable Computing Architectures and Design Methods**. *IEE Proceedings: Computer Digital Techniques*, v. 152, n. 2, p. 193–208, 2005.

WOODS, J. W.; O'NEIL, S. D. **Subband Coding of Images**. *IEEE Trans. Acoust. Speech Signal Proc.*, ASSP-35, n. 5, p. 1278–1288, 1986.

XIE, T.; WENIG, C.; HE, Y. **A New Approach to Arbitrary Shaped DCT**. *PCS´97, Berlin*, p. 97–101, 1997.

XILINX. Em http://www.xilinx.com/. *Acessado em: 20 de junho*, 2008.

ZIV, J.; LEMPEL, A. **A Universal Algorithm for Sequential Data Compression**. *IEEE Trans. Info. Theory*, IT-23, n. 3, p. 337–343, 1977.