Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Coordenação de Pós-Graduação em Ciência da Computação

Um estudo sobre consumo energético na indexação de conteúdo visual com síntese em alto nível usando *hardware* reconfigurável

Thiago Fonseca Meneses

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação

Linha de Pesquisa: Redes de Computadores e Sistemas Distribuídos

Elmar Uwe Kurt Melcher (Orientador)

Campina Grande, Paraíba, Brasil

M543e Meneses, Thiago Fonseca.
Um estudo sobre consumo energético na indexação de conteúdo visual com síntese em alto nível usando hardware reconfigurável / Thiago Fonseca Meneses. - Campina Grande, 2012.
77 f.

Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2012.
"Orientação : Prof. Dr. Elmar Uwe Kurt Melcher".
Referências.

1. Processamento de Imagem. 2. Energia. 3. FPGA. 4. Análise de Conteúdo. 5. Dissertação - Ciência da Computação. I. Melcher, Elmar Uwe Kurt. II. Universidade Federal de Campina Grande - Campina Grande (PB). III. Título

CDU 004.932(043)

"UM ESTUDO SOBRE CONSUMO ENERGÉTICO NA INDEXAÇÃO DE CONTEÚDO VISUAL COM SÍNTESE EM ALTO NÍVEL USANDO HARDWARE RECONFIGURÁVEL"

THIAGO FONSECA MENESES

DISSERTAÇÃO APROVADA EM 06/09/2012

ELMAR UWE KURT MELCHER, Dr.
Orientador(a)

JOSEANA MACÊDO FECHINE RÉGIS DE ARAÚJO, D.Sc
Examinador(a)

HERMAN MARTINS GOMES, Ph.D
Examinador(a)

JOSÉ ANTÔNIO GOMES DE LIMA, D.Sc
Examinador(a)

CAMPINA GRANDE - PB

Resumo

Atualmente, o maior impacto ambiental causado pelos computadores é o consumo de energia. Este trabalho estuda uma alternativa tecnológica, ao alto consumo de energia causado pelos computadores *desktop* e servidores, na execução de tarefas específicas, a exemplo da indexação de conteúdo visual. O principal objetivo da dissertação foi avaliar a eficiência energética, utilizando um método de indexação baseado na transformada *wavelet*, de um sistema CBIR (*Content Based Image Retrieval*) sobre a plataforma FPGA (*Field Programmable Gate Array*). Utiliza-se da eficiência energética como medida de avaliação sobre duas plataformas. Os resultados mostram que a plataforma FPGA oferece uma eficiência energética 758 vezes maior em relação ao sistema implementado na plataforma PC.

Abstract

Nowadays, the biggest impact on the environment caused by computers is the power consumption. This work studies an alternative technology to the high energy consumption caused by desktop computers and servers, in the execution of a certain task, for example, visual content indexing. The main goal of this dissertation was to evaluate the energy efficiency using a method of indexing based on wavelets of a CBIR (Content Based Image Retrieval) system implemented on a FPGA plataform. We use energy efficiency as metrics to compare the two platforms. Results show that FPGAs offer 758 times better energy efficiency than the system that we implemented previously on PC platform.

Agradecimentos

Agradeço a Deus por iluminar meu caminho, me dar saúde e força para seguir minha missão na terra.

Aos meus pais, especiais em minha vida, pela força, dedicação e pelo amor eterno.

Ao meu orientador, Prof. Elmar Melcher, pela paciência, oportunidade e conhecimento transmitidos ao longo desse trabalho. À professora Joseana, pelo apoio e ideias sugeridas.

Ao pessoal do LAD e Brazil-IP, pela ajuda e conhecimentos transmitidos. Aos meus colegas do mestrado pela amizade e momentos de alegria.

A minha namorada Raquel Lima pela ajuda, incentivo, e dedicação. E aos meus sogros Aluísio e Bernadete que me trataram como segundo filho aqui na Paraíba.

A todos amigos de Sergipe. Aos amigos que me receberam em Campina, Augusto e Welflen, e a todos os outros que conheci.

A todos que fazem parte da Universidade Federal de Campina Grande.

À Capes pelo apoio financeiro.

Conteúdo

Lista de Siglas

AR - *Augmented Reality* - Realidade Aumentada

ASIC - *Application Specific Integrated Circuit* - Circuito Integrado para Aplicação Específica

ASSP - *Aplication Specific Standard Product* - Produto Padrão de Aplicação Específica

CAD - *Computer-Aided Design* - Projeto Auxiliado por Computador

CBIR - *Content Based Image Retrieval* - Recuperação de Imagem Baseada em Conteúdo

CI - *Circuit Integrated* - Circuito Integrado

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

CPU - *Central Processing Unit* - Unidade de Processamento Central

CWT - *Continuous Wavelet Transform* - Transformada Wavelet Contínua

DSP - *Digital Signal Processing* - Processamento de Sinal Digital

EDA - *Electronic Design Automation* - Automação de Projeto Eletrônico

ELS - *Eletronic System Level* - Nível de Sistema Eletrônico

FPGA - *Field Programmable Gate Array* - Arranjo de Portas Programável em Campo

GIF - *Graphics Interchange Format* - Formato de Intercambio Gráfico

GPU - *Graphics Processing Unit* - Unidade de Processamento Gráfica

HLS - *High Level Synthesis* - Síntese de Alto Nível

HPC - *High Performace Computing* - Computação de Alto Desempenho

HSV - *Hue Saturation Value* - Matiz Saturação Brilho

JPEG - *Joint Photographic Experts Group*

LUT - *Look Up Table* - Tabela de Consulta

MPPA - *Massively Parallel Processor Arrays* - Arranjo de Processos Massivamente Paralelo

PC - *Personal Computer* - Computador Pessoal

RGB - *Red Green Blue* - Vermelho Verde Azul

RTL - *Register Transfer Level* - Nível de Transferência de Registrador

SPLD -*Simple Programmable Logic Devices* - Dispositivo Lógico Simples Programável

STFT -*Short-Time Fourier Transform* - Transformada de Fourier em Curto Tempo

VLSI - *Very Large Scale Integration* - Integração em Escala Muito Alta

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Com a expansão da internet, cada vez é maior a quantidade de conteúdo multimídia disponibilizados, tais como imagens, vídeos e áudio. Como exemplo, cita-se o *Youtube*, que recebe em seu domínio cerca de sessenta horas de vídeo a cada minuto (YOUTUBE, 2012). A popularização de *sites* como Flickr (FLICKR, 2012) e Facebook (FACEBOOK, 2012) também é responsável por grandes volumes de imagens disponível na web. Além disso, conteúdo visual está presente em diversas áreas: medicina, astronomia, robótica, mineralogia, sensoriamento remoto, entre outras (CSILLAGHY; HINTERBERGER; BENZ, 2000; WANG; CHI; FENG, 2002; SCHRODER et al., 2002; PAINTER et al., 2003; ERGEN; BAYKARA, 2010).

A necessidade de indexação para o acesso a essa categoria de documento é de fundamental importância para empresas e usuários. No caso específico das imagens e vídeo, surgem questões do tipo: como indexar e recuperar imagens e quadros de vídeo dentre bilhões de outros? Quais alternativas tecnológicas podem ser utilizadas para acelerar o processo de indexação sem comprometer o custo da solução?

Sistemas convencionais de banco de dados foram desenvolvidos para manipular dados textuais e numéricos e, para recuperar as informações, são feitas comparações diretas dos valores armazenados. Essa forma simples de recuperar informação não se aplica a dados multimídia como imagem, áudio e vídeo (WU, 1997).

A recuperação de imagens pode ser feita a partir de metadados previamente e manualmente anotados ou a partir de metadados obtidos por extração automática de características, sendo essas últimas armazenadas em um vetor de índice (WU, 1997; SAADATMAND-TARZJAN; MOGHADDAM, 2007). A primeira abordagem tem como vantagem maior a riqueza semântica

dos dados anotados. Em contrapartida, o custo de anotação das imagens é alto em escalas em que o conjunto de imagens ultrapassa milhões de imagens. Como segunda alternativa, foi proposta a técnica CBIR *(Content Based Image Retrieval)*. Nessa técnica, ao invés de utilizar anotações manuais para indexar as imagens, foi proposta a indexação baseada em conteúdo visual, descrito por características de baixo nível tais como cor (SWAIN; BALLARD, 1991; HUANG et al., 2001), forma (MAHMOUDI et al., 2003) e textura (STRICKER; DIMAI, 1997; RUI; HUANG; CHANG, 1999). Como afirma Schettini (SCHETTINI et al., 2001), a cor é a característica mais utilizada para indexar conteúdo visual. Para indexação e recuperação de vídeos, os principais desafios assemelham-se às técnicas de CBIR (HUANG et al., 2008).

A maioria dos sistemas utiliza informações do domínio espacial de pixels para extrair o vetor de características. Outra possibilidade consiste em utilizar uma transformada no domínio da frequência para extrair somente características mais importantes. Dentre as transformadas, está a transformada Wavelet que é utilizada na decomposição das imagens transformando-as para o domínio de espaço e frequência (JACOBS; FINKELSTEIN; SALESIN, 1995; WANG; LI; WIEDERHOLD, 2001).

Sistemas CBIR e de análise de imagens despertam interesse da mídia para aplicações do mundo real como evidenciam as publicações na *Scintific American* (MIRSKY, 2006), *Discovery News* (STAEDTER, 2006) e *CNN* (CNN, 2005). Como explica Datta *et. al.* (DATTA et al., 2008), espera-se que em um futuro próximo sistemas de busca por conteúdo visual atuem juntamente com a pesquisa em texto. Isso já acontece em parte com o sistema de busca do *Google Image Search* (MURPHY-CHUTORIAN; ROSENBERG, 2009), que em 2009 lançou a pesquisa por imagens semelhantes. Apesar disso, a busca por similaridade em vídeos parece um pouco distante, dentre outros fatores, uma das possíveis causas é o custo computacional exigido no processamento de vídeos contido em sítios como exemplo no *Youtube*.

Desde a década de 1990, sistemas CBIR têm sido pesquisados e desenvolvidos (RUI; HUANG; CHANG, 1999). Para prover acurácia e rápida recuperação do conteúdo, dois problemas devem ser solucionados: (i) a disparidade semântica entre conteúdo de baixo nível e conceitos de alto nível da imagem; (ii) o custo computacional demandado para análise de imagens, indexação, pesquisa e pelos algoritmos de aprendizagem (DATTA et al., 2008; YANG; KAMATA; AHRARY, 2009).

A otimização dos algoritmos utilizados na indexação de imagens apresenta uma série de

complicações quando comparada aos algoritmos utilizados na recuperação. Isso se deve pelo fato de que, uma vez que o algoritmo utilizado para indexar tenha sido modificado, toda a base de imagem tem que ser indexada novamente. Assim, o custo computacional é alto para executar melhorias no algoritmo utilizado na indexação, particularmente sobre uma grande base de imagens (SAADATMAND-TARZJAN; MOGHADDAM, 2007).

Trabalhos recentes utilizam computação em nuvem para prover escalabilidade (YANG; KAMATA; AHRARY, 2009). Contudo, nos últimos anos muita atenção tem sido dada ao impacto do uso dos computadores *desktop* e servidores. Sejam esses impactos ambientais causados pela fabricação e eliminação dos computadores como também, e mais importante, o consumo de energia (CALWELL; OSTENDORP, 2005). Como cita Calwell e Ostendorp (2005), nos dias atuais o maior impacto ao meio ambiente causado pelos computadores é o consumo de energia tendo como consequência a emissão de gases na atmosfera acelerando o processo do efeito estufa e da poluição do ar.

Como exemplo, em 2008, o valor gasto no contrato anual de energia da empresa *Google* foi maior do que toda infraestrutura da empresa, exceto terrenos e edifícios. Somente os servidores dessa empresa nos Estados Unidos, consumiram aproximadamente 2% de energia total gasta no mundo. A maior parte dessa energia é utilizada na climatização dos servidores. Com isso, grandes empresas estão migrando seus servidores para regiões frias, cujo objetivo é diminuir esse alto consumo de energia (GRUNDBERG; ROLANDER, 2011).

Previsões para o consumo energético de componentes eletrônicos é cada vez maior. Estima-se que em 2030 a internet aumentará em 30% o consumo atual de energia devido à demanda e popularização dos componentes e serviços conectados à internet (ECOINFORMATICA, 2011).

Diante dessa preocupação mundial surgem iniciativas como a *Climate Savers Computing Initiative* (CLIMATE, 2010), grupos sem fins lucrativos de consumidores, empresas e organizações, dedicadas a prover tecnologias inteligentes, cujo objetivo é reduzir o consumo energético dos computadores e periféricos e usar a energia de forma eficiente. Como exemplo, tem-se as tecnologias desenvolvidas pelas empresas de processadores, como a *SpeedStep* da Intel e *Cool'n'Quiet* da AMD, que operam na mudança de frequência de *clock* dos processadores na execução de tarefas, minimizando o consumo de energia e a dissipação de calor.

Tradicionalmente, ao longo dos anos, os sistemas computacionais têm sido desenvolvidos em *software* executando tarefas em um processador de uso geral, ou em um *hardware* customizado, assim como em uma arquitetura mista alocando subtarefas específicas em *hardware* e deixando outras tarefas em *software*. No tipo de arquitetura desenvolvida em *software*, pode-se citar como benefício a flexibilidade e facilidade no desenvolvimento do sistema. Mas, quando comparada a um sistema desenvolvido em *hardware*, a execução do sistema é muito onerosa, com impacto direto no consumo de energia, além de poder ser ineficiente, uma vez que instruções do processador não são otimizadas na realização de tarefas específicas (RIBEIRO, 2002).

O desenvolvimento de *hardware* tem sido simplificado e aperfeiçoado durante as últimas décadas. Essa simplificação se deu graças à evolução dos circuitos digitais que passaram de transistores para circuitos integrados VLSI *(Very Large Scale Integration)*, da criação de ferramentas EDA *(Eletronic Design Automatation)*, do surgimento das linguagens de descrição de *hardware* (HDL) consolidadas no meio acadêmico e industrial (RIBEIRO, 2002) e do surgimento de dispositivos de *hardware* configuráveis (VOTANO; PARHAM; HALL, 2004) que conferem mais flexibilidade aos projetos de *hardware*.

Como afirma Datta (DATTA et al., 2008), uma arquitetura em *hardware* para os sistemas CBIR se faz necessário e essencial para futuras aplicações de sistemas de recuperação de imagem. Essa arquitetura auxiliará sistemas desenvolvidos em *software* a solucionar problemas cujos benefícios vão do processamento paralelo e distribuído ao consumo energético. Com isso, uma arquitetura em *hardware* provê o uso de tecnologia inteligente e pode ser amplamente explorada nos sistemas de visão computacional.

1.1 Objetivos

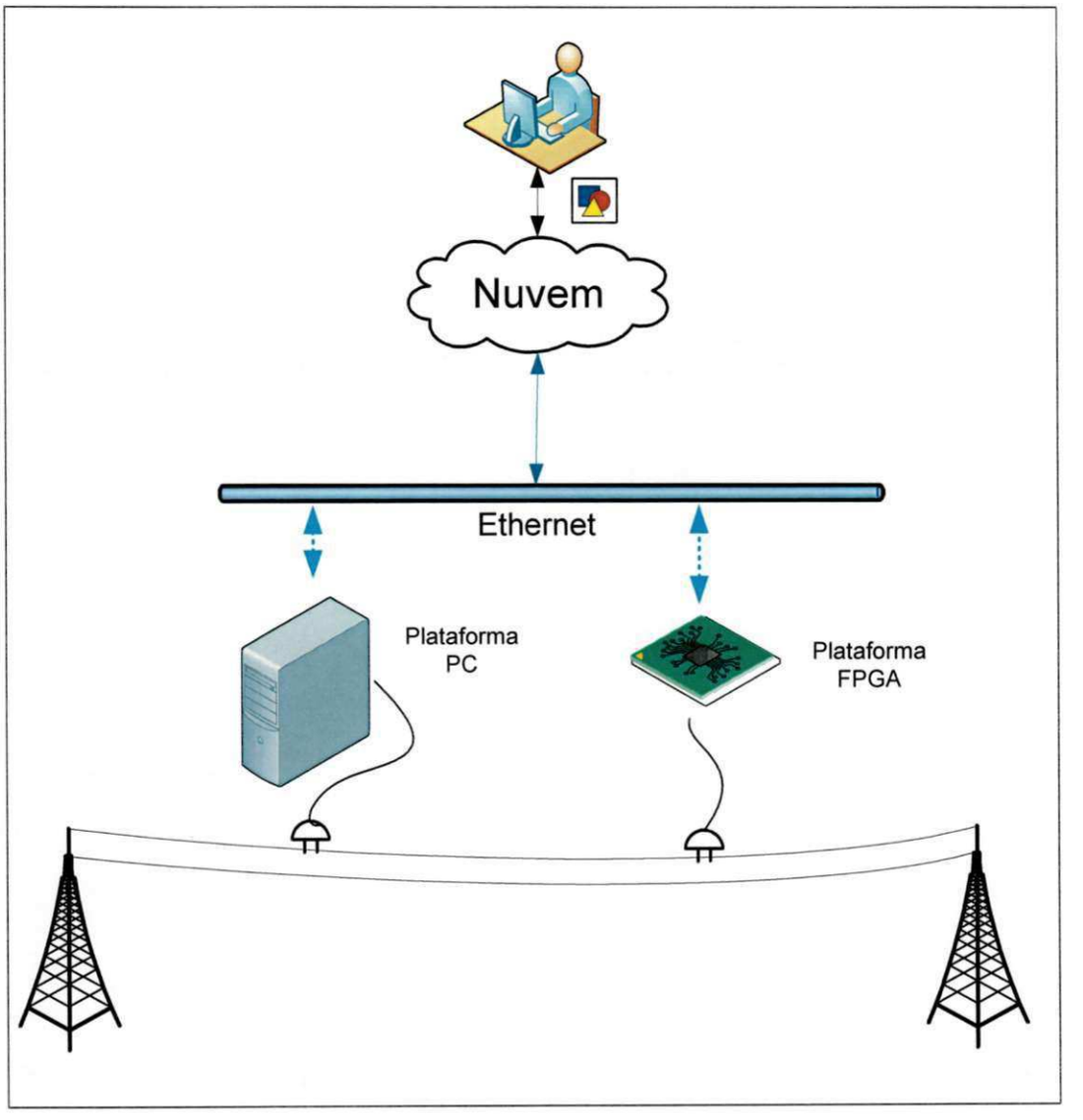
O trabalho consiste em comparar duas implementações do mesmo modelo de *software* com uso de tecnologias distintas: CPU e FPGA. Na plataforma PC, com uso de CPU, considera-se a implementação do sistema CBIR denominado *Guaatupi* (MENESES; FILHO; ARAUJO, 2010), implementado anteriormente a este trabalho. Avalia-se a eficiência energética dessa solução e compara-se com a solução desenvolvida neste trabalho com uso da plataforma FPGA.

Os sistemas estão conectados somente à rede de energia e à rede *Ethernet*. Os dados a se-

rem processados entram pela *Ethernet* e os resultados são enviados de volta pela mesma rede. Com o uso da plataforma FPGA, busca-se uma maior eficiência no processo de indexação de conteúdo visual, proporcionando a grandes empresas utilizar sistemas CBIR com grande volume de dados e reduzir o consumo energético, como também, propor uma alternativa computacional aos sistemas desenvolvidos sobre a plataforma PC.

Conforme ilustrada na Figura 1.1, compara-se o mesmo algoritmo nas duas plataformas: plataforma PC, com processador de propósito geral; plataforma FPGA, com circuito específico.

Figura 1.1: Diagrama do projeto.



1.1.1 Objetivos Específicos

O modelo de *software* utilizado como referencia neste trabalho foi o *Guaatupi*. Os objetivos específicos deste trabalho são:

1. Medir o consumo energético do sistema *Guaatupi* sobre a plataforma PC;
2. Fazer um estudo da arte e do consumo energético dos principais sistemas CBIR e de processamento de imagens;
3. Realizar um estudo sobre a Computação Reconfigurável com uso de FPGA e da ferramenta de Síntese de Alto Nível Catapult C;
4. Adequar o modelo utilizado na plataforma PC para plataforma FPGA;
5. Implementar o modelo na plataforma FPGA;
6. Medir e avaliar o consumo energético nas plataformas PC e FPGA;

1.2 Justificativas

Diante da evolução científica e tecnológica dos últimos séculos, o ser humano busca um lugar melhor para que todos possam viver. O uso da tecnologia traz melhorias, porem uma deterioração acelerada dos recursos naturais e poluição do meio ambiente.

Com a corrida pelo mercado, grandes empresas buscam inovar e oferecer uma maior qualidade dos seus produtos aos seus clientes e usuários, consumindo cada vez mais recursos ambientais e tendo gastos elevados para prover seus serviços. Além disso, os sistemas computacionais desenvolvido nos dias atuais contam com bases consolidadas de conhecimento, como por exemplo, na extração de características, inteligência artificial, reconhecimento de padrões e etc. Com isso, a escolha do sistema *Guaautpi* reflete essas duas vertentes dos sistemas atuais: (i) auxilia usuários e permite reduzir a exaustão na busca por conteúdo multimídia; (ii) utiliza a plataforma PC que oferece uma maior flexibilização no desenvolvimento ao custo elevado no consumo de energia. Com isso, neste trabalho será realizado um estudo do uso da plataforma FPGA como alternativa aos custos energéticos e computacionais utilizados na indexação visual do sistema *Guaatupi*.

Com a utilização de PC, predomina-se a utilização da arquitetura de von Neumann na execução dos algoritmos. Essa arquitetura tem como desvantagem a uniformização na execução dos algoritmos, além dos processadores estarem limitados ao um conjunto de instruções e unidades funcionais. Com frequências máximas de *clock* limitadas pelo calor dissipado, as indústrias de processadores investem na inclusão de vários processadores dentro de um mesmo *chip*. Contudo, o incremento no desempenho ocorre de maneira mais difícil. Dentre outras razões, têm-se o fato de que programadores não estão habituados a expressar paralelismo nas aplicações com proveito do fluxo de instruções, sendo essa tarefa, muitas vezes realizada pelo Sistema Operacional.

Por outra vertente, nos últimos anos com o crescimento da lógica programável, que permite a implementação de todo e qualquer algoritmo com a lógica de Boolean, e os esforços das indústrias EDA (*Electronic Design Automation*), aproxima-se cada vez mais o desenvolvimento de *hardware* com a abstração utilizada em *software*. Ferramentas como a *Catapult* (MENTOR, 2012), surgem como alternativas no mercado, possibilitando engenheiros de *software* escrever algoritmos para implementação em *hardware* e de maneira não muito distante da escrita em *software*. Com isso, explora-se o paralelismo inerente à aplicação com proveito de uma arquitetura subjacente como nos FPGA.

1.3 Estrutura da Dissertação

O restante desta dissertação está estruturado da seguinte forma:

1. **Capítulo 2**: Fundamentação teórica sobre os temas abordados neste trabalho. Citam-se as etapas no desenvolvimento de um projeto de *hardware*, assim como a evolução das ferramentas de Síntese de Alto Nível. Destacam-se os sistemas de recuperação de imagem por conteúdo.

2. **Capítulo 3**: Descrição do sistema de CBIR utilizado como referência. Implementação do sistema na plataforma FPGA.

3. **Capítulo 4**: Apresentação dos resultados obtidos do consumo energético das soluções na plataforma PC e FPGA.

4. **Capítulo 5**: Considerações finais e sugestões para trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Nesse capítulo, realiza-se uma revisão bibliográfica de alguns conceitos matemáticos utilizados por esse trabalho. Abordam-se o projeto de desenvolvimento de circuitos eletrônicos e as ferramentas utilizadas na geração de *hardware*. Explica-se a computação reconfigurável com uso de *FPGA*. E por fim, descrevem-se alguns sistemas de recuperação de imagem por conteúdo e trabalhos relacionados.

2.1 Introdução à Transformada Wavelet

As wavelets são ferramentas matemáticas usadas para decomposição hierárquica de funções. Wavelets permitem que qualquer função seja descrita em termos de forma global, permitindo ampliar detalhes restritos. Trata-se de uma técnica que provê representações em diferentes níveis de detalhes.

A primeira menção às wavelets aconteceu em 1909 com Alfred Haar. Mas somente 1985, através de um trabalho sobre processamento digital de imagens que Stephane Mallat trouxe a notoriedade do uso das mesmas. Daí então a matemática Ingrid Daubechies criou um conjunto de bases ortogonais de wavelet com suporte compacto formando a base atual paras as wavelets.

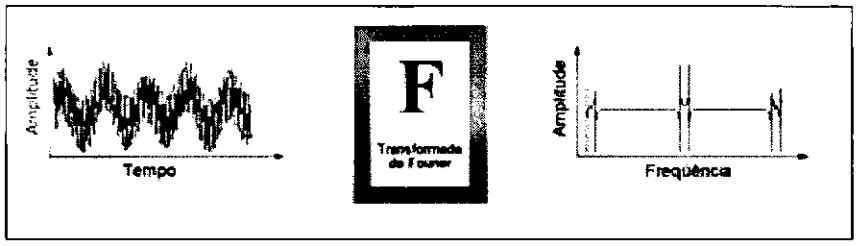
2.1.1 Transformada de Fourier

Uma das ferramentas matemáticas mais utilizadas ao longo dos anos é a transformada de Fourier. Quando é necessário representar um sinal, como, por exemplo, uma imagem decomposta em suas componentes de freqüência, existe uma série de ferramentas, sendo a mais utilizada a transformada de Fourier. Sua equação é dada por:

$$F(w) = \int_{-\infty}^{+\infty} f(t)e^{-iwt}dt \tag{2.1}$$

Trata-se de uma transformada integral que expressa uma função em termos de função bases (senos e cossenos) como soma ou integral de funções multiplicadas por coeficientes. A transformada de Fourier realiza uma mudança no domínio da função. Quando aplicada na área de processamento de sinais a transformada de Fourier é tipicamente utilizada para decompor um sinal nas suas componentes de freqüência e amplitudes.

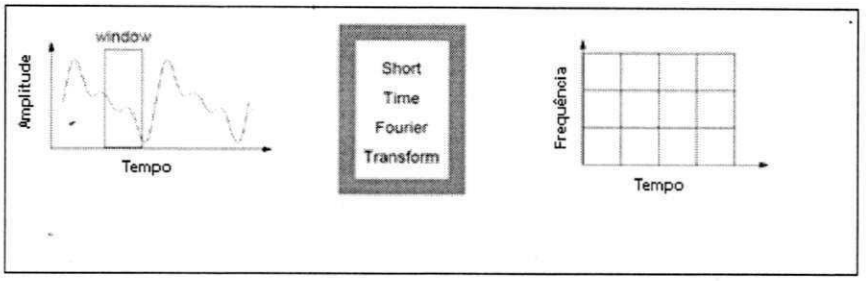
Figura 2.1: Transformada de Fourier.



No entanto, com a utilização da transformada de Fourier perdem-se informações temporais, ou seja, não se sabe quando um determinado evento ocorre. Na tentativa de contornar esse problema surgiu a técnica de "janelamento" proposta por Dennis Gabor.

Com essa nova técnica, também conhecida como *Short-Time Fourier Transform* (STFT), a transformada de Fourier é dividida em pequenas janelas de tempo. Apesar disso a STFT apresenta problemas na tentativa de obter valores de sinais no domínio de freqüência e tempo. Isso ocorre pelo tamanho fixo das janelas. Como ela decompõe o sinal em senos e cossenos que são infinitos, com a técnica de janelamento ocorre uma ruptura no tempo da função seno e cosseno dando origem a um problema denominado *aliasing*, ou seja, a reconstrução não perfeita do sinal. Outros problemas encontrados, tais como a definição do tamanho da janela, podem ser melhor observados no trabalho de Misiti et al. (MISITI; MISITI; OPPENHEIM, 1997).

Figura 2.2: Análise da Fourier em janela.

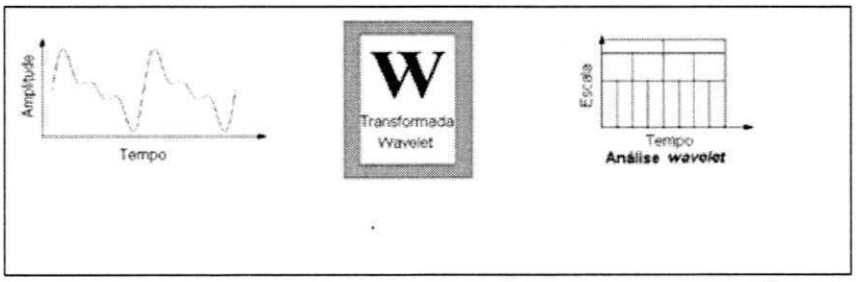


2.1.2 Transformada Wavelet

Como explica Daubechies (DAUBECHIES, 1992), a transformada wavelet provê uma ferramenta para localização de freqüência (ou escala) e tempo na análise de uma função.

A análise wavelet é o passo seguinte da transformada de Fourier no sentido que essa trás melhorias em aspectos não observados com a transformada de Fourier. Utiliza janelas dinâmicas sendo que as janelas maiores são aplicadas aos sinais de baixa frequência, e as janelas menores aplicadas às componentes de maior frequência.

Figura 2.3: Transformada wavelet.



Para uma função ser considerada uma wavelet, existem diversas condições que devem ser satisfeitas. Sendo classificadas em dois tipos: continuas e discretas (DAUBECHIES, 1992).

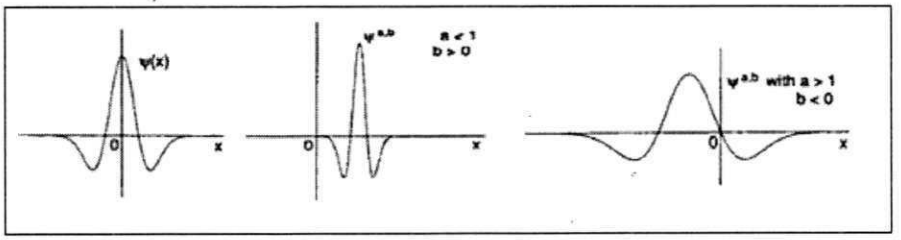
Assim como na Transformada de Fourier, em que uma função é decomposta em diversas frequências formadas por senos ou cossenos, na transformada wavelet a função é decomposta em diversas funções derivadas da chamada wavelet mãe. Existem diversos tipos de wavelet mãe onde o uso depende do sinal a ser analisado e do contexto da aplicação. Dentre essas, pode-se destacar: a wavelet de Haar; a família Daubechies (dbN); Biorthogonal; Coiflets; Symlets; Morlet; Chapéu Mexicano e Meyer.

Uma wavelet é uma função de duração efetivamente limitada que tem média igual a zero. Essa é uma característica importante que diferencia a análise baseada em wavelet da análise de Fourier (DAUBECHIES, 1992). A Transformada Contínua Wavelet ou CWT (*Continuous Wavelet Transform*) é definida como a integral no tempo da função geradora do sinal, sendo multiplicada pelas diferentes versões da wavelet mãe (representada pela letra grega Ψ) em infinitas escalas conforme a equação abaixo.

$$C(escala, posicao) = \int_{-\infty}^{+\infty} f(t)\psi(escala, posicao, t)\, dt \tag{2.2}$$

- **escala**: Indica a compressão ou dilatação da onda.
- **posicao**: Indica a posição da onda em relação ao sinal analisado, geralmente sendo relacionada ao eixo do tempo.

Figura 2.4: Wavelet de acordo com a escala (a) e posição (b).



Com isso, têm-se infinitos coeficientes da wavelet C de acordo com a escala e posição.

Wavelet de Haar

A transformada wavelet de Haar, cujo nome é designado ao criador Alfred Haar, é a mais simples e a primeira conhecida. Sua wavelet mãe pode ser descrita como:

$$\psi(t) = \begin{cases} -1, & se\ 0 \leq t < 1/2 \\ 1, & se\ 1/2 \leq t < 1 \\ 0, & outro \end{cases} \tag{2.3}$$

e sua função de escala:

$$\phi(t) = \begin{cases} 1, & 0 \leq t < 1 \\ 0, & outro \end{cases} \tag{2.4}$$

Entre suas principais propriedades estão:

- Qualquer função real contínua pode ser aproximada a partir de uma combinação linear de $\phi(t)\phi(2t)\phi(4t)...\phi(2^k t)$.
- A ortogonalidade é na forma:

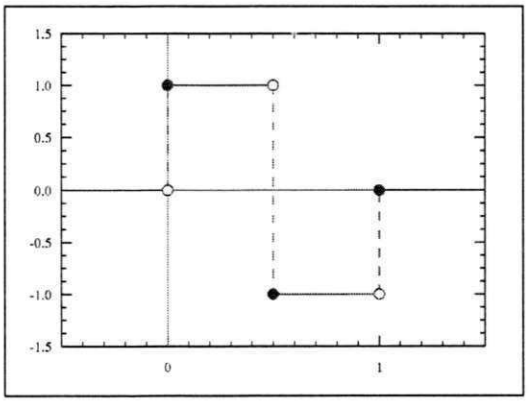
$$\int_{-\infty}^{+\infty} 2^m \psi(2^m t - n)\psi(2^{m1} - n1)\, dt \tag{2.5}$$

- Funções wavelet com diferentes escalas m têm uma função de relacionamento:

$$\phi(t) = \phi(2t) + \phi(2t - 1) \tag{2.6}$$

$$\psi(t) = \phi(2t) - \phi(2t - 1) \tag{2.7}$$

Figura 2.5: Wavelet Haar.



Para exemplificar a transformada wavelet de Haar, observa-se o vetor formado por quatro elementos:

$$[4\ 8\ 7\ 6] \tag{2.8}$$

O primeiro passo, para decompor esse vetor a partir da transformada de Haar, é calcular a média entre os pares adjacentes obtendo o seguinte vetor:

$$[6\ 6.5] \tag{2.9}$$

Essa operação também é representada pelo operador ($\downarrow 2$) cuja função é obter sub-amostras (do inglês, *downsampling*) e quando aplicado ao sinal reduz sua taxa de amostra pela metade (SCHNITER,).

Para recuperar os quatro valores iniciais necessita-se de informações adicionais que são obtidas através dos coeficientes de detalhe. Para os dois primeiros pares 4 e 8, o coeficiente de detalhe é obtido através de um valor que adicionado a 6 consegue-se obter 4 e subtraído de 6 obtêm-se o valor 8, logo esse coeficiente é -2, pois:

$$6 + (-2) = 4 \tag{2.10}$$

$$6 - (-2) = 8 \tag{2.11}$$

Prossegue-se sucessivamente para o restante dos pares adjacentes, tem-se:

$$6.5 + (0.5) = 7 \tag{2.12}$$

$$6.5 - (0.5) = 6 \tag{2.13}$$

Após calcular os coeficientes de detalhe, analisa-se o novo vetor formado pelas médias, nesse exemplo o vetor:

$$[6\ 6.5] \tag{2.14}$$

Aplica-se o processo novamente e encontra-se média igual a 6.25. O número que adicionado a 6.25 é igual a 6 é -0.25, logo se encontra o terceiro coeficiente pois ele está dentro da regra onde:

$$6.25 + (-0.25) = 6 \tag{2.15}$$

$$6.25 - (-0.25) = 6.5 \tag{2.16}$$

Após concluir o processo tem-se a transformada wavelet do sinal formada pelos coeficientes com informação global (a última média 6.25) mais os coeficientes de detalhe, sendo o vetor resultante:

$$[6.25 \ -0.25 \ -2 \ 0.5] \tag{2.17}$$

Ao analisar o vetor resultante observa-se que a aplicação da transformada wavelet é responsável por uma compressão de dados.

Transformada Discreta da Wavelet em 2D

No desenvolvimento desse trabalho, utilizou-se da decomposição da wavelet em duas dimensões sobre os sinais encontrados nos *pixels* das imagens. A transformada wavelet em duas dimensões pode ser realizada em duas formas: padrão e não padrão.

O funcionamento da transformada padrão é feito aplicando a transformada wavelet a cada linha obtendo o coeficiente de informação global na primeira posição e os demais coeficientes de detalhes de cada linha. Após esse primeiro passo, o processo se repete nas colunas da imagem resultante do processo anterior. A Figura 2.6 ilustra o processo de decomposição padrão.

Como explica Stollnitz, Derose e Salesin (STOLLNITZ; DEROSE, 1995), para a decomposição não padrão as operações alternam entre as linhas e as colunas. No primeiro passo horizontalmente, calcula-se a média e a diferença de pares de *pixels* com o valor do pixel em cada linha da imagem. O próximo passo é realizar o mesmo processo em relação à coluna. Por fim, para completar a decomposição, repetem-se os dois processos recursivamente no quadrante resultado obtendo a decomposição da imagem. A Figura 2.7 ilustra uma imagem após o processo de decomposição não padrão.

Figura 2.6: Wavelet Haar. Decomposição padrão.

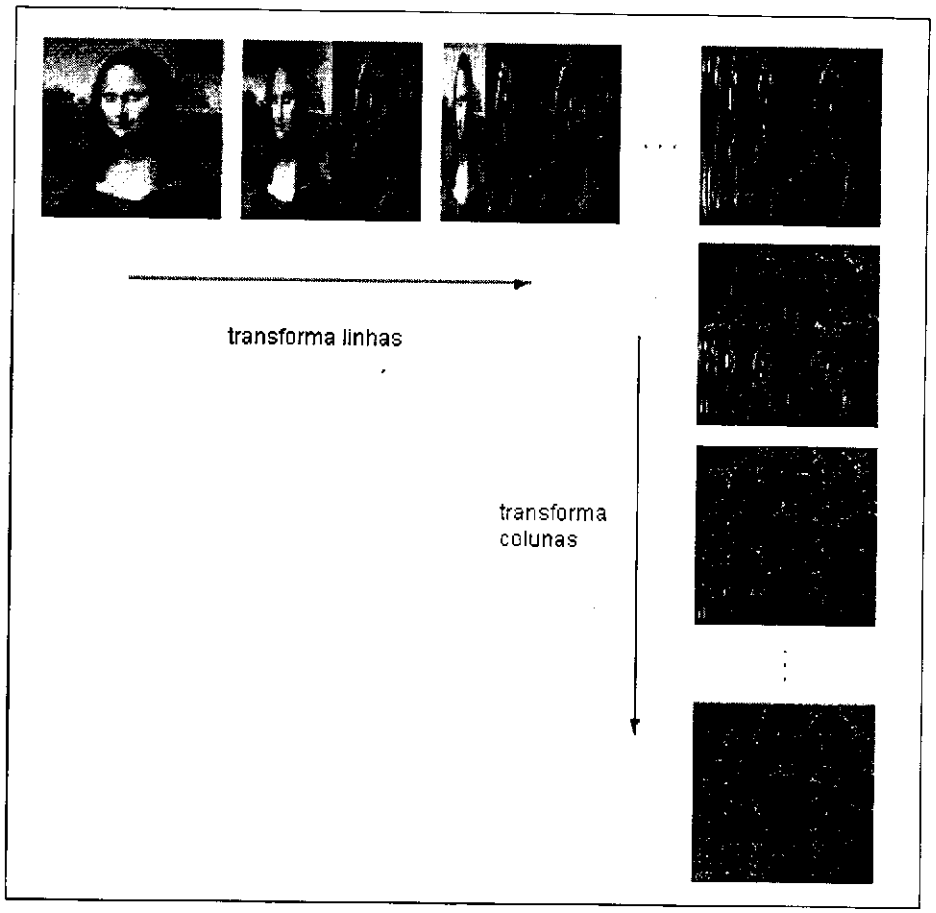
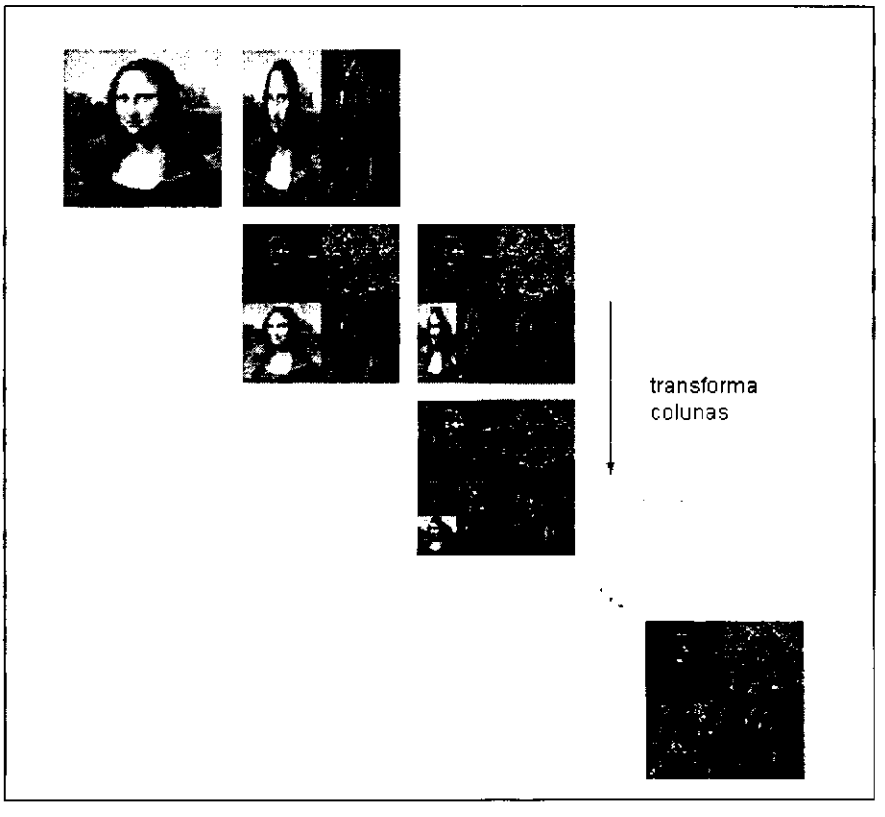


Figura 2.7: Wavelet Haar. Decomposição não padrão.



2.2 Projeto de *Hardware*

Uma das aplicações da eletrônica é a criação em grande escalá de computadores digitais. Desenvolver manualmente complexos circuitos eletrônicos torna-se inviável em grandes projetos. Para solucionar esse problema, técnicas de projeto auxiliado por computador denominadas CAD (*Computer-Aided Design*) surgiram com as ferramentas EDA (*Electronic Design Automation*). A primeira utilização dessas ferramentas vem do ano de 1967, quando um programa de computador foi utilizado para determinar as conexões entre transistores de um circuito integrado (MARTENS; GIELEN, 2008).

No início da década de 1980, surgiram empresas, como a Mentor Graphics, especializadas em ferramentas de suporte a desenvolvimento de projetos. Nos dias atuais, seus *softwares* que projetam circuitos integrados, permitem a automação de várias etapas de um projeto.

Com as inovações tecnológicas, produtos que utilizam semicondutores vêm sendo produzidos em grande escala. Esses são utilizados em diversas áreas, como na computação (computadores, consoles de jogos), comunicação (redes 802.11 a/b/g, Bluetooth), indústrias (automação de processos) e produtos eletrônicos (Mp3, câmeras digitais). Além disso, a miniaturização dos componentes resultou num maior desempenho com baixos custos. Assim, cada vez mais é possível aumentar as funcionalidades de um chip de silício tornando-o mais complexo. Esse crescimento exponencial da complexidade do circuito integrado (CI) é normalmente citado como lei de Moore (MOORE, 2006).

Antes da era da computação reconfigurável, dois métodos tradicionais eram utilizados para realizar uma computação: (i) processador de propósito geral, o mais flexível, em que o processador executa um conjunto de instruções para realizar uma determinada tarefa; (ii) um circuito integrado para uma aplicação específica, ou ASIC (*Application Specific Integrated Circuit*) (COMPTON, 2000).

Para elaboração de um ASIC, até meados da década de 1990, a metodologia utilizada para fabricação baseava-se na "captura e simulação". Em resumo, o departamento de desenvolvimento fornecia um conjunto de especificações e requisitos do produto a ser desenvolvido. Sem informações de como implementar esses requisitos, a equipe desenvolvia em diagramas de blocos a arquitetura do *chip*, servindo como uma especificação preliminar. Uma vez refinada e aprovada, a equipe de lógica e *layout* convertia cada função do bloco em lógica

ou em esquema de circuito. Após isso, simulava para verificar as funcionalidades, tempo, e cobertura (GAJSKI, 1990).

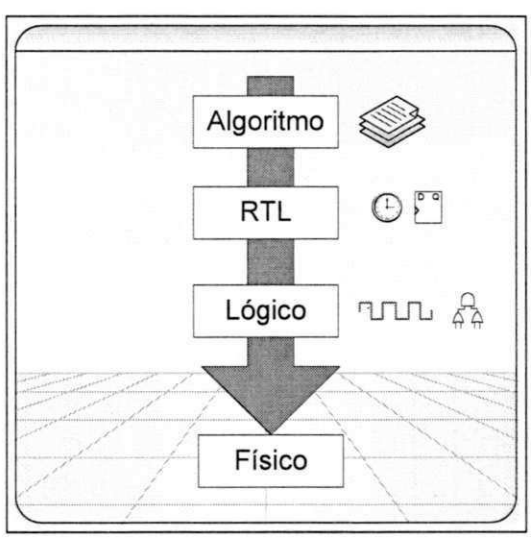
Somente anos mais tarde que a síntese lógica começou a ser parte essencial do processo de desenvolvimento, liderando uma evolução na metodologia "capture e simule" para "descreva e sintetize" (GAJSKI, 1990).

2.2.1 Etapas de um Projeto em *Hardware*

Com a complexidade cada vez maior dos chips, automatizar etapas do desenvolvimento com maior nível de abstração, com funcionalidades mais fáceis de serem entendidas, tem diversas vantagens. De acordo com Lin (LIN, 1997), dentre essas vantagens tem-se: (i) a redução do tempo de desenvolvimento do projeto; (ii) a possibilidade de explorar diferentes implementações que podem ser geradas e avaliadas rapidamente; (iii) a utilização de ferramentas de automação identificando requisitos e restrições não observadas pelo projetista.

O projeto de *hardware* é usualmente visto como uma sucessão de etapas com vários níveis de abstração: nível algorítmico, nível de transferência de registrador (RTL, *Register Transfer Level*), nível lógico e nível físico (DOULOS, 2012).

Figura 2.8: Etapas de um projeto de hardware.



Conforme se observa a partir da Figura 2.8, o nível superior é o algorítmico ou comportamental. Nele encontram-se estruturas conhecidas pelos programadores de *software*, como

laços, testes condicionais e atribuições. O sistema é descrito na forma de um algoritmo que calcula os valores de saída de acordo com os valores de entrada. Detalhes como *clock* e tempo são abstraídos desse nível, sendo o nível de transferência de registrador responsável por esse detalhamento.

No nível de transferência de registrador (RTL), a estrutura do sistema é composta por operadores combinacionais (soma, multiplicação) e elementos de memória (registradores, bancos de memórias, etc). O nível lógico é formado por portas lógicas e *flip-flops*, entre outros.

Por fim, o nível físico corresponde aos transistores (KHAN; SHOAB, 2011). Com isso, diferentes tipos de sínteses ocorrem no projeto de um *hardware* como a síntese de alto nível, a síntese lógica e a síntese física.

Em resumo, a síntese é o processo de tradução de uma descrição no nível de abstração mais elevado para um nível mais baixo (LIN, 1997). Conforme Forland (MCFARLAND; PARKER; CAMPOSANO, 1990), a tarefa de síntese de alto nível surge a partir de uma especificação comportamental de um sistema e um conjunto de restrições e objetivos a serem alcançados, para se chegar a uma estrutura que realize tal comportamento no nível de transferência de registrador (RTL).

2.3 Síntese de Alto Nível

A busca por uma maior produtividade dos projetistas trouxe uma maior automatização com níveis de abstração mais elevados. Como explica Coussy e Morawiec (COUSSY; MORAWIEC, 2008), essa maior produtividade pode ser oferecida pela ESL (*Eletronic System Level*), a partir do qual se faz co-projetos em *software* e *hardware* e síntese de alto nível.

Conforme Martin e Smitch (MARTIN; SMITH, 2009), a evolução da síntese de alto nível se divide em três gerações e uma pré-histórica.

2.3.1 Pré-histórica: Geração 0

Surgiu na década de 1970 com trabalhos sobre síntese e síntese de alto nível. Nessa época, as indústrias de produtos EDA (Calma, Applicon e ComputerVision) somente ofereciam síntese de *layout* físico. Um grupo formado por pesquisadores da Universidade de Carnegie Mellon

focaram seus trabalhos na especificação, simulação e síntese, ambos no nível de registrador e algorítmico.

Dentre esses trabalhos, encontra-se o de Mario Barbacci (BARBACCI, 1981), cujas anotações descrevem que em teoria se poderia "compilar" um conjunto específico de instruções de processador (utilizando a linguagem ISPS, *Intrustion Set Process Specfication*) em *hardware*. A linguagem ISPS foi considerada a primeira linguagem de descrição de *hardware*. Assim, surgiu a noção de síntese a partir de uma linguagem de alto nível, sendo conhecida anos mais tarde como um processo de geração automática de circuito de *hardware* a partir de uma "descrição comportamental".

Essa geração serviu de base para as futuras pesquisas, mas teve pouco impacto nas indústrias de EDA. Isso ocorreu pelo fato de que muitas indústrias estavam começando a adotar as recentes ferramentas CAD, sendo formada em uma época anterior as tecnologias de chip VLSI (*Very Large Scale Integration*) e antes mesmo do surgimento dos *softwares* comerciais de síntese.

2.3.2 Geração 1: 1980 - 1990

Considera-se a geração primária da síntese de alto nível. São explorados os principais conceitos, tendo impacto no campo com a pesquisa e apresentação de artigos científicos. Nela, os fundamentos da síntese de alto nível (HLS) foram decompostos em modelagem de *hardware* (*"hardware modeling"*), alocação de recursos (*"resource allocation"*), escalonamento (*"scheduling"*) e vinculação (*"binding"*) (COUSSY; MORAWIEC, 2008).

Na modelagem do *hardware*, a principal função é obter especificações, como um programa, disponibilizando uma descrição parcial e ordenada do projeto assim como concorrências utilizadas. Na alocação de recursos, determinam-se quais recursos serão utilizados e a quantidade necessária para construção final do circuito em *hardware*. Em seguida, vem o escalonamento, que gera ciclos de relógio específicos para as operações, cria uma máquina de estados finitos correspondente a essas atribuições e determina o tempo de cada operação em tempo de execução. Na vinculação, elabora-se um vínculo entre uma operação e um recurso (unidade funcional, memória etc), ou seja, as operações da unidade funcional são associadas de forma coerente com os resultados da alocação e do escalonamento. (COUSSY; MORAWIEC, 2008).

Como explica Martin e Smith (MARTIN; SMITH, 2009), embora vital na formação da base das futuras ferramentas comerciais, essa primeira geração falhou. Dentre os motivos, tem-se que a adoção da síntese no nível de transferência de registrador (RTL) estava nos primeiros passos, substituindo o método de captura de esquemático lógico para descrição com linguagem de *hardware* (HDL). Sobre essas circunstâncias, desenvolvedores estavam aprendendo como utilizar a síntese RTL de forma eficiente, e muitos ainda nem sequer tinham adotado.

Outro fator do fracasso foi a qualidade dos resultados. Essas ferramentas podiam gerar arquiteturas simples com extensiva alocação de recursos, vinculação primitiva e projetos difíceis de serem aceitos, sendo muitas ferramentas especializadas em processamento digital de sinais (DSP, *Digital Signal Processing*) (MARTIN; SMITH, 2009).

2.3.3 Geração 2: 1990 - 2000

Período em que empresas como Synopsys, Cadence e Mentor Graphs começaram a oferecer ferramentas de Síntese de Alto Nível (HLS). A Synopsys, dominante na síntese em RTL, tinha grande interesse ao lançar a ferramenta "*Behavioral Compiler*". O grupo Alta, da Cadence, ofereceu o "*Visual Architect*", orientada ao processamento de sinais.

Com a segunda geração, essas ferramentas foram motivo de grandes interesses, mas falharam diante do comércio e dos usuários. Dentre os motivos, cita-se a alocação errada de pessoas que deveriam usar as ferramentas de síntese de alto nível (MARTIN; SMITH, 2009). Estas por outro lado, não satisfizeram os critérios de prover qualidade nos resultados (área, desempenho) ao mesmo esforço, ou menor esforço com a mesma qualidade nos resultados, em relação aos usuários da síntese RTL, além de apresentar uma difícil curva de aprendizagem.

Outro motivo foi a utilização da linguagem de descrição de *hardware* (HDL) nessas ferramentas. Desenvolvedores de *software* não usavam HDL nos algoritmos desenvolvidos, necessitando alterar seus algoritmos para serem adequados.

A qualidade dos resultados foi outro motivo do fracasso dessa geração. Com difícil validação, resultados ruins eram obtidos quando se fazia uso dessas ferramentas para aplicações no domínio de controle. Por fim, a segunda geração também falhou por não oferecer uma ferramenta para desenvolvedores de *software*.

2.4 Computação Reconfigurável

No desenvolvimento de um dispositivo computacional, projetistas confrontam-se no problema que envolve a balança entre flexibilidade e eficiência. De um lado, aplicações específicas (ASICs), especializadas em realizar com alto desempenho a tarefa para a qual foram projetadas, com baixo consumo energético e com altas velocidades de *clock*. Por outro lado, os processadores programáveis, ou microprocessadores, que são programados para executar qualquer tipo de aplicação, são projetados com um conjunto de instruções predeterminadas e limitadas, controlados e organizados para realizar de forma sequencial qualquer tipo de computação, tornando-os menos eficientes.

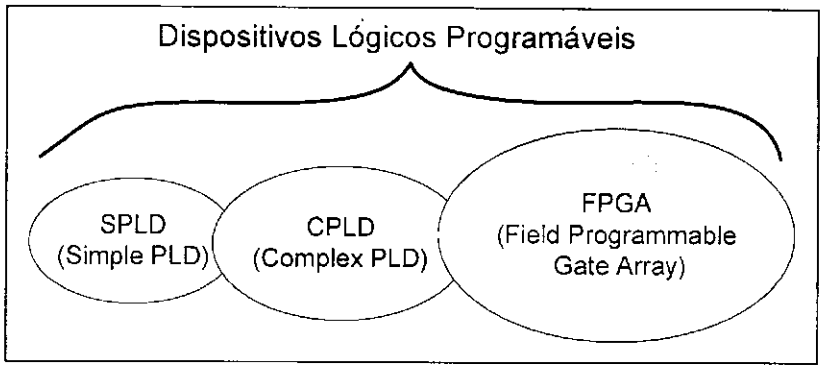
A fim de encontrar um meio termo entre flexibilidade e eficiência, surgiu a computação reconfigurável. Isso foi possível com a lógica programável, em que um *hardware* flexível pode ser programado dinamicamente, mais especificamente, ser estruturado para realizar uma computação com o fluxo de dados necessário. Deste modo, o *hardware* é estruturado para implementar diretamente as operações necessárias para a aplicação, como também, organizados para explorar mecanismos como concorrência, inerente à computação.

Atualmente, os dispositivos com lógica programável estão divididos em três categorias: SPLDs (*Simple Programmable Logic Devices*), CPLDs (*Complex Programmable Logic Devices*) e FPGAs (*Field Programmable Gate Array*). Cada dispositivo tem foco em aplicações específicas. Mas, em geral, qualquer funcionalidade desenvolvida para um dispositivo simples (SPLD), também pode ser implementada dentro de um dispositivo complexo (CPLD). O contrário não é válido, muitas funcionalidades complexas não podem ser desenvolvidas em dispositivos simples (VOTANO; PARHAM; HALL, 2004). Na Figura 2.9, ilustra-se essas categorias.

Com os FPGA, funcionalidades adicionais podem ser implementadas, tornando-os mais complexos. Com isto, alguns fatores podem influenciar na escolha do dispositivo, como, a necessidade de poder expandir as funcionalidades, custo, familiaridade com o dispositivo específico, entre outras.

Portanto, ao invés de computar uma função sequencial a partir de instruções de tempo, a computação reconfigurável utiliza unidades lógicas configuradas no espaço, que envolve diferentes unidades funcionais, envolvendo a computação paralela para gerar e consumir

Figura 2.9: Categorias dos dispositivos.



resultados.

2.5 Trabalhos Relacionados

2.5.1 Sistemas de Recuperação de Imagem Baseados em Conteúdo

Nesta seção são abordados os principais sistemas pesquisados. São descritos os sistemas clássicos criados ao longo dos anos cujo objetivo é recuperar informações a partir de técnicas que as descrevam visualmente. Nesta seção, aborda-se uma breve descrição desses sistemas e os tipos de características que são utilizadas.

Sistemas modernos de CBIR definem um conjunto de propriedades de baixo nível capazes de caracterizar o conteúdo das imagens e utilizá-las para fins de recuperação. Essas características devem ser simples (no sentido que um algoritmo de extração automática possa ser utilizado), mas bastante significativas (no sentido de que possam capturar de forma eficiente o conteúdo presente na imagem). Com isso, forma-se um vetor de características e uma função de similaridade, utilizados na recuperação de imagens. Para mais detalhes sobre o processo de extração de características recomenda-se a leitura da tese de Doutorado de Smitch (SMITH, 1997).

Diante da evolução dos sistemas CBIR nos últimos anos, pode-se classificar os sistemas de acordo com características globais e locais. Nos sistemas que utilizam as características globais, destacam-se os descritores gerais formados por cor, forma, textura e a aplicação de técnicas no domínio do tempo-frequência, como a utilização de Transformadas. Nos

sistemas baseados em regiões, busca-se "objetos" específicos com características particulares sobre a imagem. Esses são descritos por características locais que fragmentam a imagem num conjunto de regiões homogêneas (BARTOLINI, 2001). A seguir, serão descritos exemplos de sistemas CBIR de acordo com essa classificação.

Recuperação de Imagem a partir de Características Globais

QBIC

Desenvolvido pela IBM, o QBIC *(Query By Image Content)* é um dos sistemas mais conhecidos na área de recuperação de imagem baseada em conteúdo. Os dois objetos chave do QBIC são imagens e vídeos. Esse sistema extrai características computáveis como cor, formato, textura, movimentação de câmera, além de movimento de objetos (FLICKNER et al., 1995).

O sistema QBIC, para extrair características de cor, utiliza histograma de cor em diferentes representações de espaço de cor. Para forma, a extração é feita a partir de momentos invariantes que representam a forma em relação à área e circularidade em relação à orientação. Para textura, utilizam-se versões modificadas da proposta de Tamura (TAMURA; MORI; YAMAWAKI, 1978) de características fineza (*coarseness*), contraste (*contrast*) e direcionalidade (*directionality*).

O QBIC utiliza várias formas distintas de consultas: mediante uma imagem de consulta em que o sistema busca por imagens semelhantes ou iguais à enviada pelo usuário, a partir de um esboço gerado pelo usuário que descreve suas principais características ou por intermédio de seleção de padrões de cor e textura (FLICKNER et al., 1995).

Para recuperação de vídeo, o QBIC segmenta o vídeo em tomadas, e a partir dessas tomadas são extraídos inter-quadros candidatos a identificar a tomada. A partir desses quadros, o sistema trabalha com imagens estáticas extraindo suas características e armazenando em um banco de dados (FLICKNER et al., 1995).

As buscas são baseadas em similaridade vetorial, a partir da distância Euclidiana, nas quais são usados vetores que representam as características da imagem (cor, textura, forma) (FLICKNER et al., 1995).

Fast Multiresolution Image Query

Desenvolvido pelo Departamento de Ciência e Engenharia da Universidade de Washington, é um sistema que usa da técnica de CBIR, sendo o primeiro a utilizar técnica de multirresolução aplicada à recuperação de imagem a partir do conteúdo. A consulta de imagens em uma base de dados pode ser feita a partir de uma imagem com baixa resolução originada de um *scanner* ou câmera de vídeo, como também rascunhos desenhados pelo usuário (JACOBS; FINKELSTEIN; SALESIN, 1995).

Como explicam os autores Jacobs, Finkelstein e Salesin (JACOBS; FINKELSTEIN; SALESIN, 1995), o trabalho *Fast Multiresolution Image Query* utiliza a ferramenta matemática wavelet de Haar para decompor imagens, extraindo características para sua representação, indexação e recuperação. Essa representação foi denominada pelos autores de "assinatura da imagem".

Nas consultas, são utilizadas métricas experimentais para truncar e quantizar versões da wavelet decomposta e para comparar os coeficientes mais significativos da imagem de consulta com a imagem alvo (JACOBS; FINKELSTEIN; SALESIN, 1995). Entre estas métricas estão: o espaço de cor utilizado, tipo de wavelet, o tipo de decomposição (padrão ou não padrão), truncamento, quantização e normalização.

A base de imagens foi obtida a partir do uso de 1093 imagens de pinturas fixas de diversos artistas. Foram adicionadas imagens provenientes da web, a partir de um *crawler*, sendo indexadas imagens do tipo GIF (*Graphics Interchange Format*) formando uma base de 20.588 imagens para realização dos experimentos.

Os experimentos foram realizados de várias formas. Inicialmente, comparou-se as métricas desenvolvidas pelos autores com métricas de histograma, Gaussiana e *City-Block*, sobre diferentes resoluções, em três categorias: imagens digitalizadas, rascunhos e rascunhos por lembrança na memória do usuário. Nas três categorias, a técnica desenvolvida no trabalho obteve a maior quantidade de sucesso nas consultas. Por fim, foram realizados testes mais precisos sobre distorções aplicados às imagens, com mudanças na escala, rotação, translação, deslocamento de cor e todas combinadas.

Sistema de Rodrigues

Sistema desenvolvido no laboratório de Visão Computacional da Universidade Federal de Campina Grande, apresenta várias combinações na formação do descritor da imagem, com características de cor, forma e textura (RODRIGUES, 2008).

Para extrair as características da cor, diferentes espaços foram utilizados como RGB, HSV e YCbCr em diferentes números de *bins* por componente (128, 64, 32 e 16). Cada *bin* representa o espectro de cor pertencente a um intervalo, fundamental na criação de histogramas. Em relação à forma, foram utilizados momentos invariantes de Hu (TARR, 2000). Os métodos de Wavelets e LBP foram utilizados para extrair as características de textura. Após a extração de todas essas características, os dados são armazenados em um banco de dados juntamente com o endereço da imagem na Web. Esses dados, são utilizados posteriormente no treinamento de uma rede neural do tipo auto organizável (RODRIGUES, 2008), utilizando um tipo de característica específica para gerar um classificador.

Conforme explica Rodrigues (RODRIGUES, 2008), o processo utilizado nas consultas de imagens ocorre de forma diferente. O usuário faz uma seleção prévia de quais características serão extraídas da imagem. Com isso, de forma indireta, o usuário seleciona quais classificadores serão utilizados na recuperação da imagem. Se mais de uma característica for selecionada, ocorrerá uma combinação entre os classificadores. Ao final do processo, a rede neural retorna um conjunto de neurônios vencedores, isto é, que possuam menor erro de quantização em relação a imagem consultada, retornando uma lista de imagens decrescentes de acordo com as similaridades.

Foram realizados experimentos para validar o método desenvolvido e análise estatística considerando a opinião de usuários. Dentre os resultados, destacam-se os obtidos a partir da combinação de descritores com espaço de cor RGB com 32 componentes de cor, HSV com 16 componentes e extração a partir de LBP (*Local Binary Pattern*) e Wavelets. O sistema foi formado com um banco de imagens provenientes de um *Web Crawler* com aproximadamente 64.000 endereços de imagens do tipo JPEG (*Joint Photographic Experts Group*).

Recuperação de Imagem Baseada em Regiões

Netra

Desenvolvido pelo departamento de Engenharia Elétrica e de Computação da Universidade da Califórnia, o Netra é um sistema que utiliza cor, textura, forma e informações espaciais em regiões de imagens segmentadas (MA; MANJUNATH, 1999).

Para extrair as características de cor, o Netra utiliza uma representação quantizada do espaço de cor RGB em 256 cores. O algoritmo generalizado de Lloyd (DU; FABER; GUNZBURGER, 1999) é utilizado para agrupar regiões de cores homogêneas, cujo objetivo é representar regiões com poucas cores (MA; MANJUNATH, 1999).

Na extração de formas, o Netra detecta a representação das bordas de *pixeis* próximos sendo utilizados três tipos de detecção: (i) função de curvatura (*curvature function*), (ii) distância do centro (*centroid distance*) e (iii) função de coordenadas complexas (*complex coordinate function*) (MA; MANJUNATH, 1999). Com o estudo experimental, detectou-se que as descrições de forma possuem melhor desempenho utilizando a transformada de Fourier somente com coeficientes de amplitude e descartando as informações de fase (MA; MANJUNATH, 1999).

A extração de textura é baseada no projeto de Ma e Manjunath (MANJUNATH; MA, 1996) que utiliza de bancos de filtros da Wavelet de Gabor com múltiplas orientações e escalas (MA; MANJUNATH, 1999).

A consulta pode ser feita por regiões específicas da imagem, assim como pela imagem inteira. A busca por imagens similares é feita a partir da distância Euclidiana dos descritores de forma. O sistema contém 2.500 imagens da galeria de fotos do Corel (COREL,). Nos resultados experimentais, o trabalho somente demonstra a aplicação do modelo com exemplos.

Blobworld

O BlobWorld é um sistema que se propõe a realizar pesquisa por meio de objetos identificados em figuras. Ao selecionar um objeto em uma imagem, o sistema pesquisa e recupera as imagens que contém aquele objeto (CARSON et al., 2002).

Como explica Carson (CARSON et al., 2002), para segmentar cada imagem automatica-

mente utiliza-se um modelo com distribuição de cor, textura e posição dos objetos. Utiliza-se o algoritmo *Expectation Maximization* (EM) (DEMPSTER et al., 1977) para estimar os parâmetros do modelo. O resultado é um conjunto de *pixels* pertencente a um mesmo grupo que provê a segmentação da imagem. Após segmentar a imagem em regiões, uma descrição de cada região de cor e textura é produzida. Esse sistema utilizou cerca de 10.000 imagens para formar a base de dados para consultas (CARSON et al., 2002).

Nos experimentos, o sistema foi comparado com técnicas que utilizam informações de cor globais e de histogramas de cor. Realizaram-se 50 consultas em 10 categorias de objetos. Foram obtidos melhores resultados na distinção de objetos, mas houve falha na distinção de cenas.

Visualseek

Sistema desenvolvido pelo Laboratório de Imagens e Televisão avançada da Universidade de Columbia. Esse sistema utiliza regiões de cores como principal técnica para indexação e recuperação de imagem. Trata-se de um sistema híbrido que integra indexação de imagem com métodos de consulta espacial (SMITH; CHANG, 1997).

Nesse sistema, cada região da imagem é automaticamente extraída contendo informações sobre cor e propriedades espaciais como tamanho, localização e o relacionamento com outras regiões (SMITH; CHANG, 1997). Para representação das cores, utiliza-se o espaço de cor HSV (*Hue Saturation Value*) em vez do RGB (*Red Green Blue*) devido ao menor esforço utilizado para extrair informações como: cor, saturação e intensidade (SMITH; CHANG, 1997).

Para consulta, o usuário elabora um esboço a partir do qual regiões são parametrizadas, atribuindo a essas regiões padrões, como cor, localização espacial e tamanho. A procura por imagens similares ocorre com a junção de padrões por região, utilizando a distância Euclidiana, considerando as posições relativas e absolutas entre as regiões de cores. A imagem que possuir o maior número de regiões similares é considerada a mais próxima da imagem de consulta.

O experimento foi realizado em três categorias: localização espacial, regiões de cor e características globais. Os resultados indicam melhores resultados nas categorias de localização espacial e regiões de cor.

Outros trabalhos

Na pesquisa desta dissertação, foram encontrados diversos trabalhos que comparam algoritmos na plataforma FPGA. No quesito energia, a maioria aperfeiçoa as técnicas implementadas e realiza um comparativo, tanto em desempenho como em eficiência energética. Dentre as técnicas, destacam-se a otimização no acesso à memória e o baixo consumo. Para sistemas de recuperação visual com base no conteúdo, destaca-se o trabalho de Kotoulas e Andreadis (KOTOULAS; ANDREADIS, 2004). Poucos trabalhos realizam um comparativo entre diferentes plataformas. Apesar disso, acredita-se na tendência cada vez maior desse tipo de pesquisa. Dentre os trabalhos pesquisados, destacam-se:

- **Kotoulas e Andreadis (2004)** (KOTOULAS; ANDREADIS, 2004): utiliza-se a plataforma FPGA para obter desempenho na indexação e comparação de imagens. Nessa solução, utilizou-se o histograma de cor para extrair e representar as imagens. Nenhuma investigação é feita sobre o consumo energético.

- **Guimarães, Lima e Teixeira (2007)** (GUIMARAES et al., 2007): compara-se o desempenho computacional na implementação de algoritmo de processamento de imagem, mais especificamente na área de Realidade Aumentada (AR, *Augmented Reality*). Comparam-se implementações de algoritmos no nível de cinza, filtro de média, detecção de bordas, entre outras. Esses são implementados na plataforma PC e na plataforma FPGA. Somente o desempenho foi comparado entre as plataformas. Destaca-se a implementação do algoritmo de filtro 3×3. Esse teve um desempenho no tempo de processamento em trinta mil vezes em relação à implementação em *software*.

- **Atabany e Degenaar (2008)** (ATABANY; DEGENAAR, 2008): nesse trabalho, os autores usam de técnicas em *pipeline* para reduzir o consumo energético em relação a uma solução que usa processamento paralelo. O estudo de caso foi realizado com algoritmo de detecção de bordas em imagens (filtro de Sobel) sobre a plataforma FPGA. A técnica proposta dividiu o fluxo de dados em múltiplos processos em pipeline ao invés de uma solução paralelizada. A potência da solução paralelizada foi de 771 mW. Com redução em cerca de 45 %, a solução proposta pelos autores obteve uma redução na potência de aproximadamente 300 mW.

- **Thomas, Howes e Luk (2009)** (THOMAS; HOWES; LUK, 2009): realiza-se um comparativo na geração de números aleatórios em quatro plataformas: CPU (*Central Processing Unit*), GPU (*Graphics Processing Unit*), FPGA (*Field Programmable Gate Array*) e MPPA (*Massively Parallel Processor Arrays*). Avaliam-se o desempenho e a eficiência energética sobre as plataformas. Destaca-se a eficiência energética alcançada na plataforma FPGA. Essa foi cerca de duzentos e cinquenta vezes maior quando comparada a que utiliza CPU.

- **Kestur, Davis e Williams (2010)** (KESTUR; DAVIS; WILLIAMS, 2010): nesse trabalho, os autores executam um processamento massivo para avaliar a computação de alto desempenho ou HPC (*High Performace Computing*). Foram utilizados dados de cálculos de funções lineares (BLAS), muito utilizados em *benchmarks*, que avaliam o desempenho de novos dispositivos. Compararam-se os resultados com as plataformas CPU, GPU e FPGA. Como métrica de avaliação, foram utilizados o desempenho e a eficiência energética. A plataforma FPGA obteve o melhor resultado, com redução energética de aproximadamente trezentas vezes superior em relação às outras plataformas.

- **Fowers, Brown e Cooke (2012)** (FOWERS; BROWN; COOKE, 2012): compara-se o desempenho e a eficiência energética de uma aplicação que utiliza técnicas de janelamento em imagem sobre CPU, GPU e FPGA. Destaca-se nesse trabalho, uma redução energética na ordem de uma magnitude com a utilização de FPGA. Em relação ao desempenho, com o uso de FPGA chegou-se a um desempenho de até cinquenta e sete vezes maior. No pior dos casos, o experimento com imagens 1080 pontos com janela de 45 $\times$ 45 *pixels* a 30 quadros por segundo, a potência gasta pelo FPGA foi de 12 W contra os 8 KW e 3 KW com o uso de CPU e GPU, respectivamente. A potência foi calculada multiplicando o tempo de execução pelo consumo energético de cada plataforma.

Sobre os trabalhos aqui estudados, destaca-se a implementação de diferentes tipos de algoritmos sobre diferentes plataformas. Esta dissertação de mestrado, diferencia-se dos demais trabalhos na abordagem de uma aplicação prática de um sistema CBIR com uso da plataforma FPGA como alternativa energética.

Capítulo 3

Descrição do Sistema CBIR

Neste capítulo é descrito o sistema utilizado como referência, o *Guaatupi*. Descreve-se as etapas na implementação desse sistema sobre a plataforma FPGA.

3.1 Guaatupi: Sistema de Referência

Nesse capítulo, descreve-se um sistema de recuperação baseado em conteúdo visual. A aplicação desse sistema pode ser observada pelos resultados publicados no 16^{th} *Brazilian Symposium on Multimedia and the Web - Webmedia* (MENESES; FILHO; ARAUJO, 2010). O sistema, denominado *Guaatupi*, apresenta uma abordagem para recuperar imagens a partir de características visuais no cenário da *World Wide Web*. Foi proposta uma solução completa para indexar, remover cópias duplicadas originadas do processo de indexação e recuperar imagens a partir do conteúdo visual. O sistema Guaatupi, utiliza da técnica *Fast Multi Resolution* descrita no capítulo anterior. A escolha dessa técnica foi proveniente dos bons resultados alcançados com pouca complexidade.

3.1.1 Arquitetura do Sistema

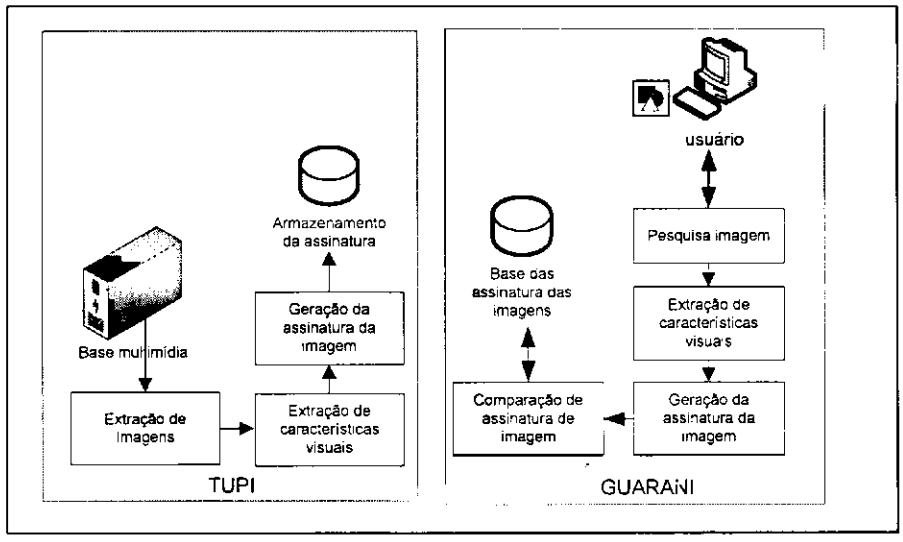
A arquitetura do sistema pode ser observada de forma resumida a partir da Figura 3.1. Esse sistema denominado *Guaatupi* foi dividido em duas partes: um sistema denominado de *Tupi*, que pode ser separado nos seguintes módulos:

- **Extração de imagens**, caracteriza pela extração das imagens de uma fonte multimídia,

tais como, um *web crawler* com uma lista de imagens a serem indexadas, ou imagens de um vídeo segmentado.

- **Extração das características visuais**, processa as características visuais utilizadas na assinatura da imagem.
- **Geração da assinatura da imagem**, seleciona os *pixels* da imagem utilizados como assinatura dessa imagem.

Figura 3.1: Arquitetura do Guaatupi.



Nesse sistema, na extração das imagens, foi utilizada a API do *Google Image* (MURPHY-CHUTORIAN; ROSENBERG, 2009) para formar a base de imagens do sistema. Buscou-se trabalhos de pintores conhecidos. Para cada consulta, foram indexadas 200 imagens. Nenhum controle foi feito sobre cópias duplicadas, resolução mínima e padrão de cor utilizado. O outro sistema, responsável por criar a interface com o usuário disponível a partir da web, foi denominado *Guarani*.

3.1.2 Extração das Características Visuais

Esse módulo é responsável pelo processamento da imagem e pela extração de características onde será gerada a assinatura da imagem. Como na maioria das técnicas utilizadas na visão

computacional, a recuperação de imagem por intermédio das características visuais é considerada uma tarefa relativamente complexa que envolve uma diversidade de fatores e muito processamento.

Nesse trabalho, utiliza-se a técnica *Fast Multiresolution Image Querying* proposta por Jacobs, Finkelstein e Salesin (JACOBS; FINKELSTEIN; SALESIN, 1995). Nessa estratégia, o usuário interage com uma imagem Q e, a partir dessa imagem, o sistema retorna a imagem T que mais se assemelha a do usuário. A busca é independente da resolução inicial da imagem, sendo redimensionada para uma resolução de 128x128 *pixels*. Para simplificar a transformada wavelet, a resolução da imagem deve atender ao requisito de ser uma potência de dois. Somente os coeficientes de maior valor absoluto são selecionados para formar a assinatura da imagem após ser decomposta pela transformada *wavelet*.

Como explica Jacobs, Finkelstein e Salesin (JACOBS; FINKELSTEIN; SALESIN, 1995) diversos fatores devem ser observados. A imagem de consulta é tipicamente muito diferente da imagem alvo permitindo distorções como cor, forma e posições de objetos pelo método de recuperação. Além desses fatores, métodos de pesquisa a partir de um rascunho devem contornar problemas como deslocamento de brilho e posição de cores e de objetos.

Métricas utilizadas para extração de assinatura

Como explica Jacobs, Finkelstein e Salesin (JACOBS; FINKELSTEIN; SALESIN, 1995), com a utilização das *Wavelets* identificaram-se componentes que deveriam ser estudados e a partir de experimentos, foi definido um conjunto de métricas que proporcionasse melhores resultados para recuperação de imagem a partir do conteúdo. Dentre essas métricas, cita-se o espaço de cor, o tipo de *wavelet*, o tipo de decomposição e a quantidade de coeficientes utilizados na assinatura. Dentre os espaços de cor experimentados (RGB, HSV e YIQ), chegou-se à conclusão de que o espaço de cor YIQ foi o que obteve melhores resultados.

Em relação ao tipo de *wavelet*, foi escolhida a *wavelet* de *Haar* por ser computacionalmente barata e simples para implementar. A partir da *wavelet* de *Haar*, a decomposição da imagem pode ser feita de duas formas: padrão e não padrão. Foi adotado o tipo de decomposição padrão por obter melhores resultados.

A imagem passa pelo processo de redimensionamento. Cada imagem capturada é transformada para uma resolução de 128x128 *pixels*. Essa resolução implica na análise de 16.384

coeficientes *wavelets*, em cada canal de cor.

O número de coeficientes utilizados para formar a assinatura da imagem foi de 60, sendo 20 para cada canal de cor. Dentre esses 20 coeficientes, são escolhidos os 10 maiores coeficientes positivos e os 10 menores negativos. Os valores dos coeficientes não são necessários, uma vez que a imagem não será reconstruída. Para formar a assinatura da imagem, utiliza-se o coeficiente de média do canal de cor e a posição do *pixel* dos coeficientes selecionados.

Métricas utilizadas na comparação de assinatura

Sejam Q a imagem rascunho ou imagem exemplo enviada pelo usuário, e T a imagem alvo ou imagem recuperada pelo sistema. A partir da imagem Q, identificou-se os primeiros coeficientes de média da imagem de cada canal, sendo representada por $Q[0,0]$. A imagem Q sofre o mesmo processo de decomposição a partir da transformada *wavelet*, separando-se os coeficientes responsáveis pela geração da assinatura. De acordo com Jacobs, Finkelstein e Salesin (JACOBS; FINKELSTEIN; SALESIN, 1995), a Equação 3.1 foi utilizada para comparação das assinaturas.

$$W_0 \left|Q\left[0,0\right] - T\left[0,0\right]\right| - \sum_{i,j:Q'\neq 0} W_{bin(i,j)} \left(Q'\left[i,j\right] = T'\left[i,j\right]\right) \tag{3.1}$$

Os coeficientes de detalhe $Q'[i,j]$ e $T'[i,j]$ são responsáveis por medir o grau de semelhança entre as imagens. A função $bin(i,j)$ provê uma forma de agrupar diferentes coeficientes em um número limitado de valores. Para cada conjunto de bin, os autores definiram pesos experimentalmente. A função bin é dada pela Equação 3.2.

$$bin(i,j) \leftarrow min\left\{max\left(i,j\right),5\right\} \tag{3.2}$$

Na Tabela 3.1 são apresentados os valores atribuídos pelos autores para os pesos dos coeficientes *wavelet*.

Para encontrar os pesos da Tabela 3.1, utilizou-se o modelo estatístico *logit*, formado por uma combinação linear de termos. Para mais detalhes sobre a técnica, vide Jacobs, Finkelstein e Salesin (JACOBS; FINKELSTEIN; SALESIN, 1995).

Tabela 3.1: Valores dos coeficientes wavelet (JACOBS; FINKELSTEIN; SALESIN, 1995).

	Imagem exemplo			Rascunho		
bin	w^y	w^i	w^q	w^y	w^i	w^q
0	4,04	15,14	22,62	5,00	19,21	34,37
1	0,78	0,92	0,40	0,83	1,26	0,36
2	0,46	0,53	0,63	1,01	0,44	0,45
3	0,42	0,26	0,25	0,52	0,53	0,14
4	0,41	0,14	0,15	0,47	0,28	0,18
5	0,31	0,07	0,38	0,30	0,14	0,27

3.1.3 Implementação

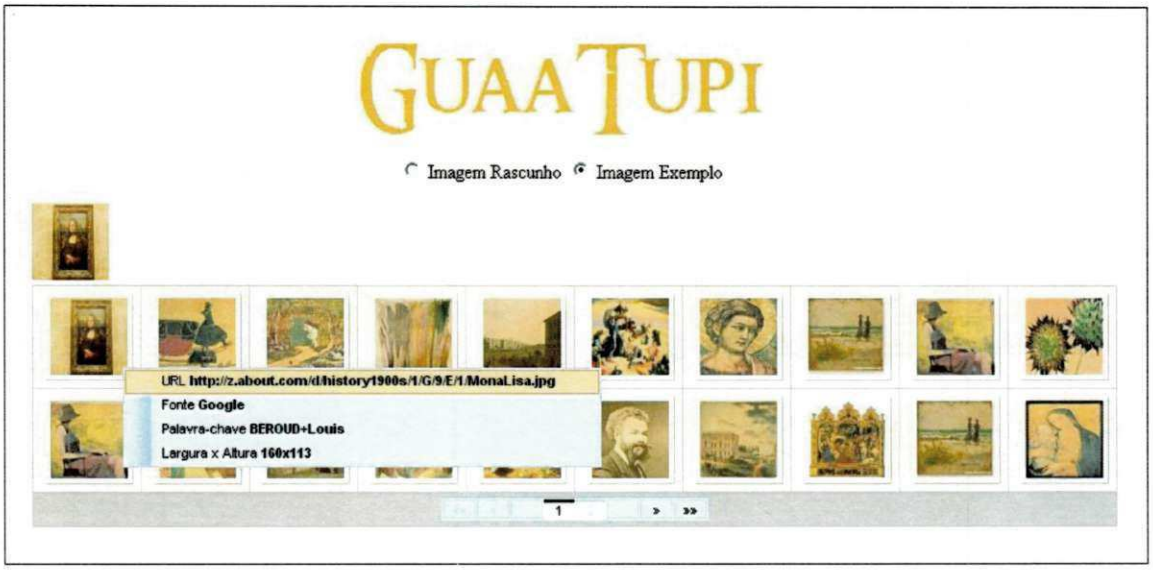
O sistema *Guaatupi* foi desenvolvido na linguagem de programação Java. Para a interface de busca de imagens, desenvolvida para web, utilizou-se a especificação JSF (*Java Server Faces*). O sistema de gerenciamento de banco de dados é o *PostgreSQL*. Na figura 3.2 está ilustrada a interface do sistema.

Foram implementadas duas formas de busca de imagem: rascunho ou exemplo. Para o método de busca, após a decomposição da imagem, um vetor é gerado contendo as informações mais importantes da imagem. Esse vetor é formado pelas médias dos canais YIQ. A partir dessas médias, são feitas os primeiros filtros na base de dados. Para **busca por rascunho**, o filtro retorna as imagens que estejam entre ± 50 da média do espaço de cor Y. Para as médias I e Q o valor de variação é de ± 60. Para a **busca por exemplo** o filtro na média Y é ± 25 e para as médias do canal I e Q ± 20.

Após o filtro das médias ser aplicado, utilizam-se as métricas para comparação de assinatura conforme explicado na Seção 3.1.2. Uma vez calculados os pesos com as métricas, utiliza-se o algoritmo de *QuickSort* para ordenar o resultado.

Finalmente, o sistema retorna as imagens mais semelhantes, podendo o usuário obter informações clicando sobre as imagens ou navegando pelos resultados. Para mais detalhes dos resultados experimentais alcançados, vide o Apêndice A.

Figura 3.2: Interface do sistema de recuperação de imagem. Na parte superior está a imagem enviada pelo usuário. Na grade encontra-se, o resultado do sistema contendo as imagens mais semelhantes.



Adequações para a implementação em *hardware*

O sistema desenvolvido para indexar imagens foi implementado em um computador de uso geral. Mais especificamente, o sistema foi desenvolvido na linguagem de programação *Java*. Assim como na maioria das linguagens utilizadas nos computadores, essa linguagem representa os números reais em ponto flutuante. Essa representação é definida pelo padrão *IEEE 754*. Dentre os benefícios representados por esse método, cita-se o maior alcance na representação dos números em detrimento à precisão. Este alcance na representação alivia o desenvolvedor da tarefa de se preocupar com o alcance dos valores das variáveis usados na codificação do problema.

No desenvolvimento deste trabalho, foram considerados somente os elementos funcionais do sistema *Guaatupi* a serem desenvolvidos na plataforma FPGA. O pré-processamento da imagem com a transformação do espaço de cor, e a transformação da resolução não foram considerados na implementação. A arquitetura resume-se em quatro partes:

- **Normalização**, utilizada na computação dos coeficientes *wavelet*, faz com que todas as *wavelets* computadas sejam ortogonais a cada outra. Sua aplicação enfatiza as diferenças entre diversas escalas na imagem.

- **Transformada wavelet**, calcula-se os coeficientes a partir dos valores da imagem. Para a transformada, o tipo de *wavelet* foi a de Haar com a decomposição padrão. Para mais detalhes sobre a transformada vide o Apêndice 2.1.

- **Extração dos coeficientes**, após a transformada *wavelet*, seleciona-se os maiores e menores coeficientes que serão utilizados na geração da assinatura da imagem.

- **Geração da assinatura da imagem**, um vetor é gerado contendo as informações de *pixel* mais importantes da imagem.

Após essas etapas, armazena-se as assinaturas para futuras pesquisas sobre a base de imagem, finalizando o processo de indexação. Na Figura 3.3, são ilustrados os aspectos considerados do sistema. Na "nuvem", estão as imagens e assinaturas provenientes do processo de indexação.

3.2 Implementação do Sistema na plataforma FPGA

Conforme citado em capítulos anteriores, a flexibilidade oferecida pelas ferramentas atuais de síntese no projeto de *hardware*, possibilita desenvolvedores projetar circuitos integrados complexos com custos de engenharia reduzidos. Com isso, o nível de abstração oferecido por essas ferramentas permite focar o algoritmo a ser implementado, ao invés de se preocupar como os circuitos que serão implementados. A implementação do sistema CBIR na plataforma FPGA é descrita neste capítulo.

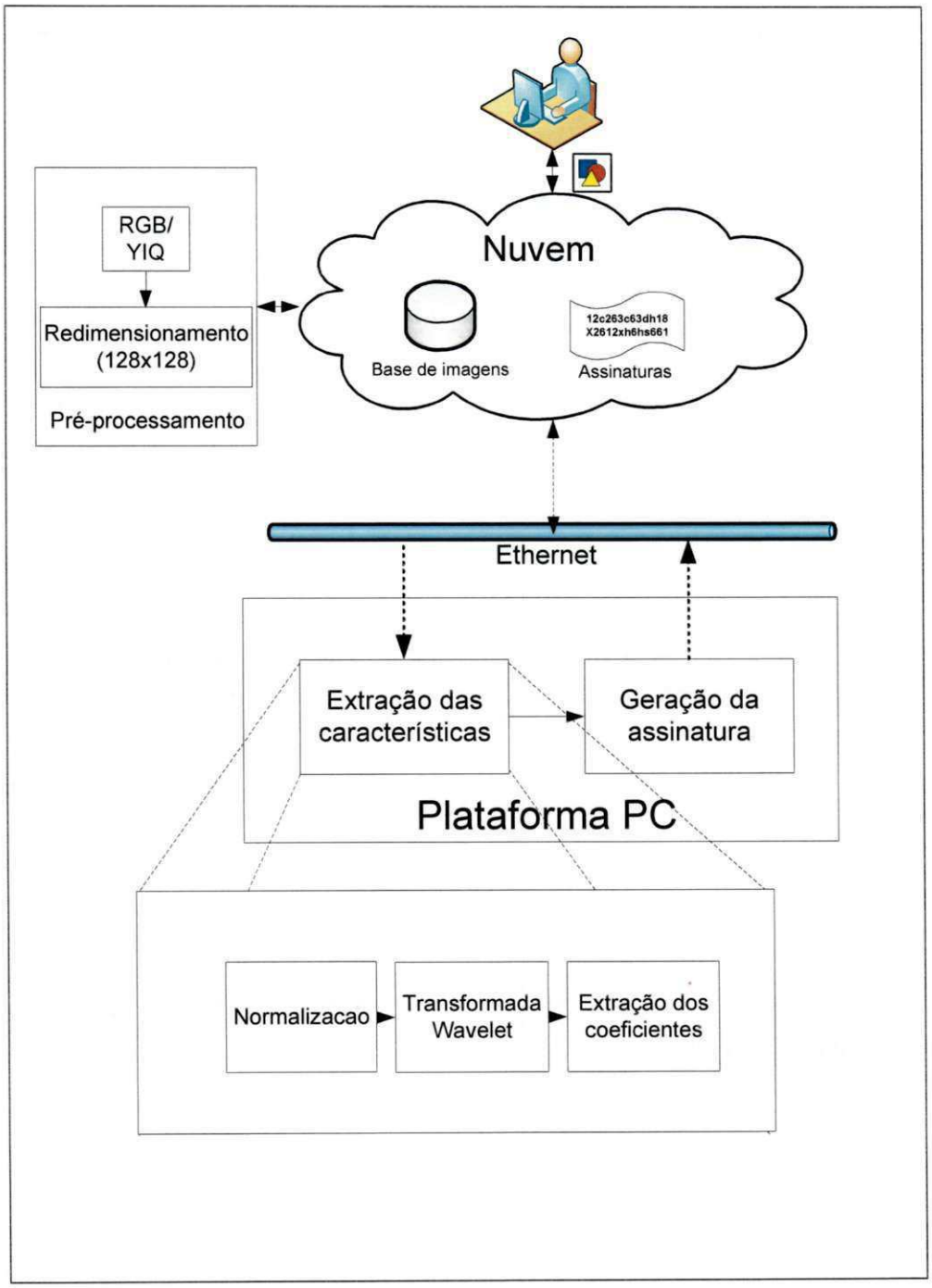
3.2.1 Arquitetura do Sistema

A Figura 3.4 ilustra a arquitetura do *Guaatupi* implementada na plataforma FPGA. Essa arquitetura é a mesma utilizada na plataforma PC. Todos os módulos são equivalentes.

3.2.2 Precisão de Bits

Em dispositivos que não possuem unidade de ponto flutuante, como o projeto desenvolvido neste trabalho, a representação dos números reais pode ser feita usando ponto fixo. Nesse método, define-se o ponto que não é alterado pela aplicação. Com isso, especifica-se o valor

Figura 3.3: Diagrama com a arquitetura do Guaatupi considerada na plataforma PC.



de "casas" decimais para parte inteira e fracionária (mantissa). Dessa forma, conforme se observa na Figura 3.5, o uso de ponto fixo delimita o espaço utilizado para representar o número inteiro e fracionário.

Para o exemplo *a)* da Figura 3.5, um bit para a parte inteira delimita apenas dois valores

Figura 3.4: Diagrama com a arquitetura na plataforma FPGA.

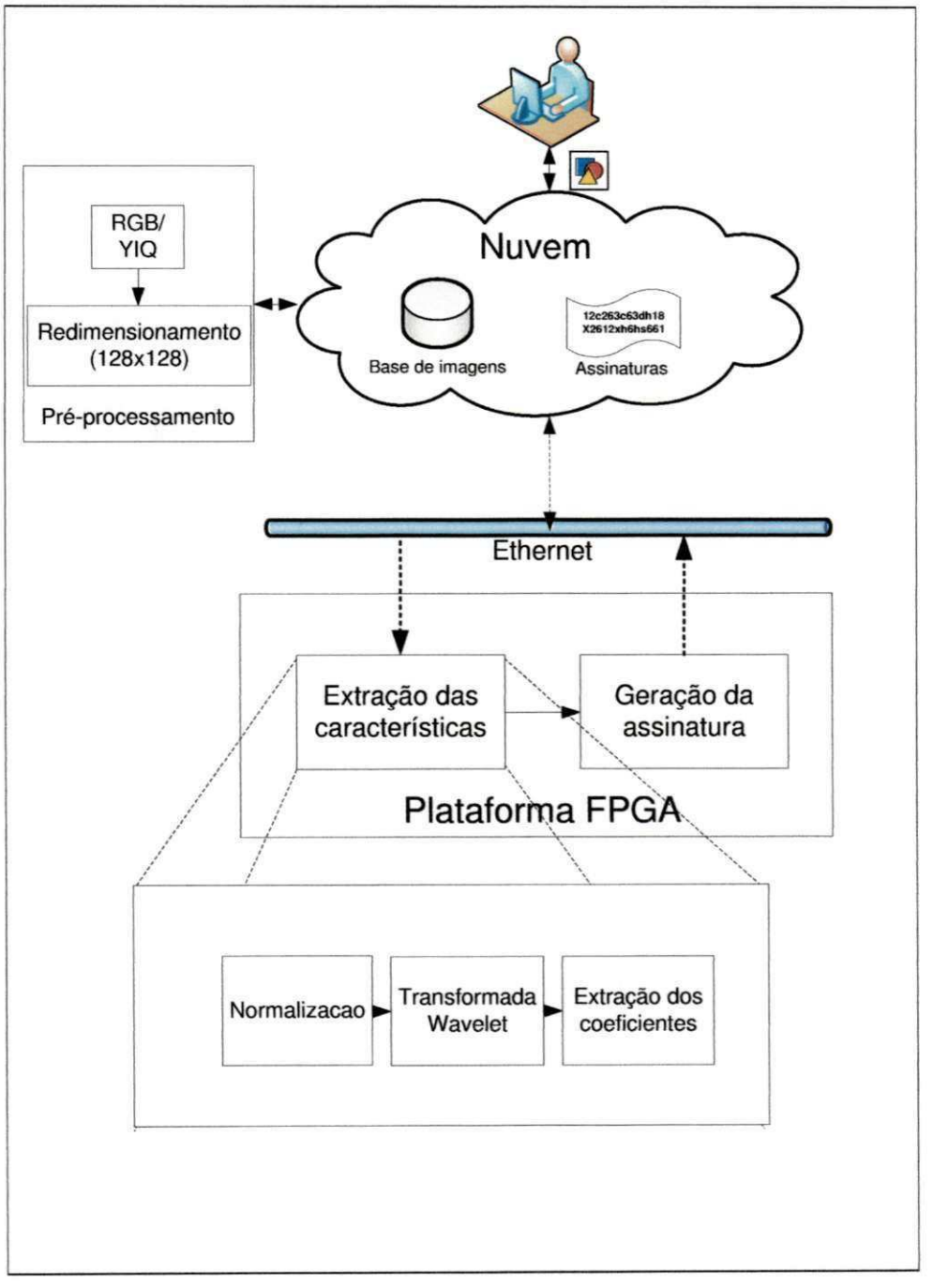
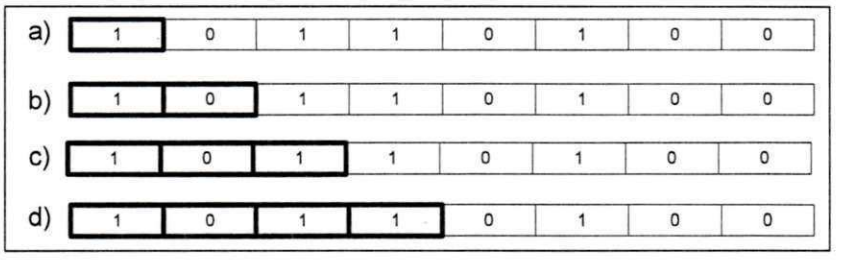


Figura 3.5: Exemplo de representação de valor em ponto fixo. Em negrito, a quantidade de "casas" utilizadas para representar a parte inteira.



inteiros (0 e 1). À medida que se aumenta o número de "casas" para representar a parte inteira, a parte fracionária delimita seu espaço, podendo gerar assim perda de informação em cálculos fracionários.

Unidades Experimentais

A compatibilidade entre o projeto desenvolvido na plataforma PC (*software*) com a plataforma em FPGA (*hardware*), depende da quantidade de bits utilizados na representação dos valores. Para isso, realizou-se um conjunto de experimentos para identificar a quantidade de bits necessários, cujos valores computados fossem equivalentes.

Foram selecionadas aleatoriamente 1200 imagens indexadas pelo sistema. Sendo utilizados 60 valores, provenientes do processo de extração de características visuais denominadas **assinatura da imagem**, para representar cada imagem, conforme explicado no capítulo anterior.

Inicialmente, para definir a parte inteira, foram analisados os intervalos de maior e menor valor das variáveis, assim como todas as variáveis intermediárias utilizadas nos cálculos. Para a parte inteira, fixou-se a quantidade de 9 bits de precisão com mais 1 de sinal, pois não foram necessários representar valores maiores do que 512. Para a parte fracionada, foram utilizados 8, 9, 10, 11, 12, 13 e 14 bits. Assim, o experimento foi realizado em 7 níveis com 18, 19, 20, 21, 22, 23 e 24 bits de acurácia. Para evitar perda de informação, utilizou-se o método de arredondamento e saturação nas variáveis (CONSTANTINIDES; CHEUNG; LUK, 2003).

Cada variável do sistema foi redefinida com a nova representação em bits. Para variáveis

internas, como contadores e variáveis temporárias, foram utilizados tamanhos adequados à representação, como exemplo, nas variáveis de 8 bits utilizadas como índice dos valores de *pixels* da imagem.

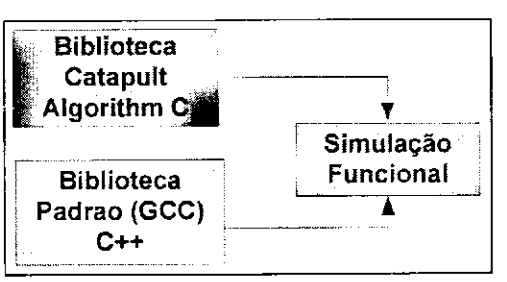
3.2.3 Simulação Funcional

O objetivo dessa etapa é simular as funcionalidades lógicas sem se preocupar com os atrasos de propagação dos sinais pelos componentes. Permite detectar erros nas funcionalidades antes da prototipação do projeto.

Inicialmente, toda estrutura do programa original em Java foi reescrita para a linguagem C++. Uma vez validado o processo de indexação na linguagem C++, utilizou-se as bibliotecas da ferramenta *Catapult C* na geração de código RTL sintetizado na plataforma FPGA.

Foi elaborado um ambiente de simulação com a interface utilizada no *Catapult C* com uso das bibliotecas necessárias para sintetizar o algoritmo sobre a plataforma FPGA. Com isso, todos os códigos desenvolvidos foram compilados e testados com o uso das bibliotecas da ferramenta *Catapult C*. Esse código desenvolvido foi adequada às restrições da ferramenta para poder ser sintetizável. Nela desenvolveu-se toda estrutura com instruções de *hardware* e a utilização de variáveis com precisão de bits adequada. Os resultados dessa interface foram comparados com a interface implementada em código C++ (GCC) do algoritmo original. Na figura 3.6 é ilustrado o contexto do processo utilizado na simulação funcional.

Figura 3.6: Processo da simulação funcional. Na caixa "Biblioteca Padrão (GCC) C++", o algoritmo original desenvolvido com o uso das bibliotecas do GCC. Na caixa "Biblioteca Catapult Algorithm C", o algoritmo sintetizado pela ferramenta Catapult C. A simulação funcional, compara o resultado entre as duas interfaces.



Como ilustrado na Tabela 3.2, na simulação funcional, os valores de *pixels* de uma ima-

gem foram submetidos às duas interfaces. Realizou-se o processo utilizado na indexação da imagem, comparando os valores das duas implementações. Apesar de os valores não serem exatamente iguais, considera-se uma aproximação que confirma a equivalência entre esses. Em "C++"têm-se os valores da solução na plataforma PC. Nos resultados "Catapult", o resultado equivalente, obtido pela simulação do algoritmo sintetizado na plataforma FPGA com 22 bits de precisão nos cálculos dos coeficientes.

Tabela 3.2: Resultado da simulação funcional. Lista de alguns coeficientes utilizados no processo de indexação.

Catapult	**C++**
0.459	0.460
-0.107	-0.107
0.017	0.017
-0.345	-0.346
0.010	0.011
-0.155	-0.154
-0.201	-0.201

No ambiente de configuração, as propriedades da ferramenta *Catapult C* têm impacto no desempenho e na qualidade final do projeto. Nenhum tipo de otimização sobre o algoritmo foi realizado. Com isso, o ambiente de estudo foi equivalente tanto na plataforma PC como na plataforma FPGA em quesitos de otimização.

3.2.4 Síntese de Alto Nível

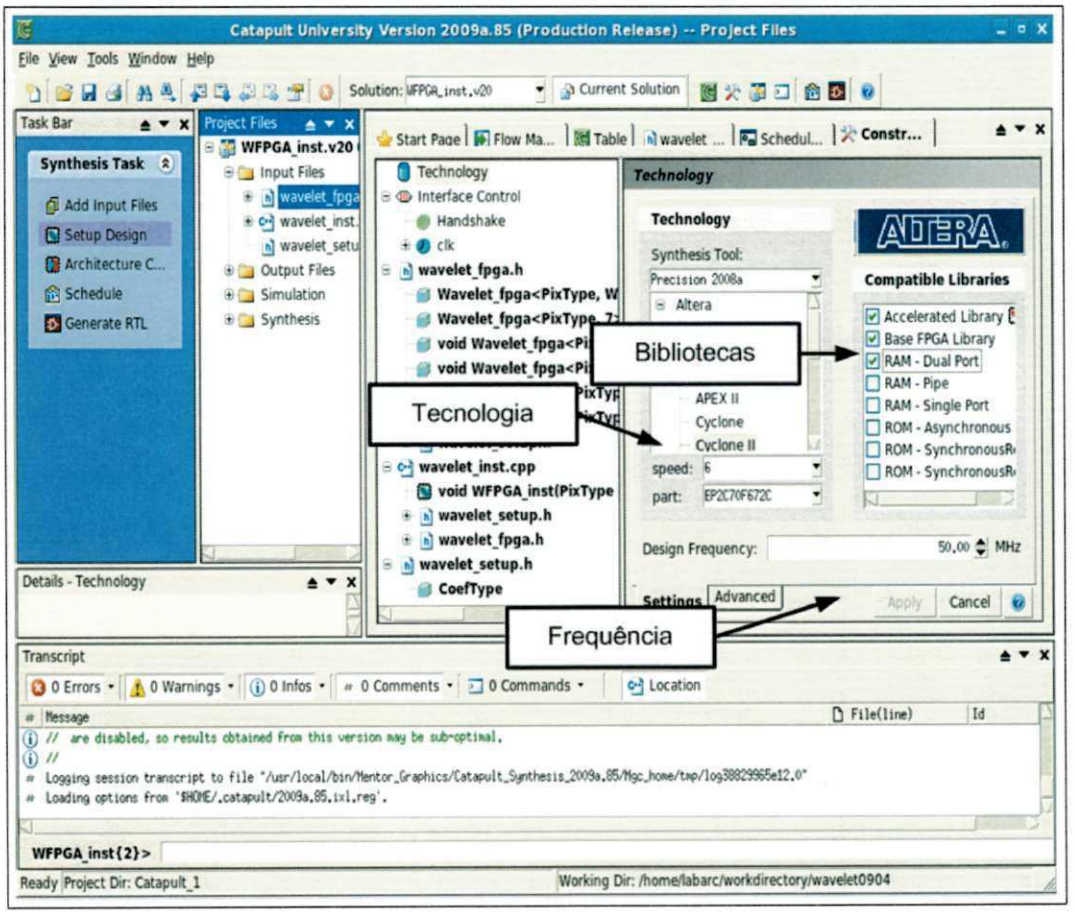
Para desenvolver o algoritmo na plataforma FPGA, utilizou-se da ferramenta de Síntese de Alto Nível Catapult C (MENTOR, 2012), da fabricante *Mentor Graphics*®.

A implementação do processo de indexação de conteúdo visual na plataforma FPGA foi realizada com a análise do algoritmo implementado sobre a plataforma PC. Algumas alterações foram realizadas na modelagem e no código. Isso foi feito para adequar as restrições impostas pela ferramenta de síntese. Com isso, o algoritmo desenvolvido aproxima-se da

implementação na plataforma PC.

Conforme ilustrado na Figura 3.7 da ferramenta Cataput, a síntese ocorre em várias etapas. Na opção **Setup Design**, configura-se o tipo de *hardware* e suas restrições. Define-se o tipo de arquitetura e os recursos que serão utilizados. Como ilustrado na Figura 3.7, utilizou-se a tecnologia da Altera Cyclone II (ALTERA, 2012), com frequência de *clock* de 50 MHz. Foi selecionado o recurso de memória RAM com portas duplas de leitura e escrita. A escolha dessa memória teve impacto na síntese do projeto. Isso ocorreu pelo fato de uma grande quantidade de leitura e escrita na memória no processamento da imagem.

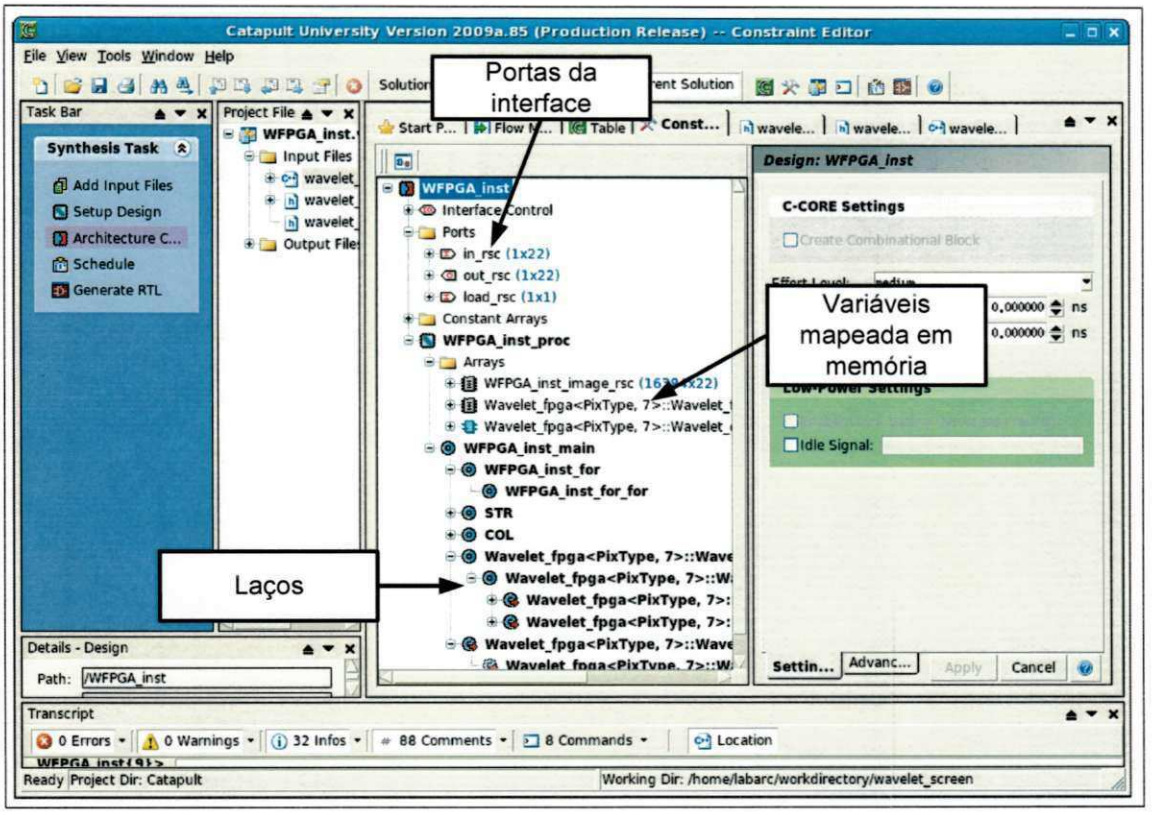
Figura 3.7: Etapas no procedimento de síntese da ferramenta Catapult.



O próximo passo é o **Architectural Constraints**. Nesse passo, são tratadas as restrições e otimizações sobre as portas, laços e variáveis. Como ilustrada na Figura 3.8, observa-se toda estrutura com laços, variáveis e recursos utilizados pelo dispositivo. Nas portas, encontram-se as variáveis utilizadas como entrada e saída do dispositivo. Sinais de *clock*

e *reset* são adicionados automaticamente pela ferramenta. Na parte de *Array*, destaca-se a memória *("WFPGA_inst_image_rsc")* formada por 16384 palavras (128x128 *pixels*) com tamanho de 22 bits.

Figura 3.8: Configurações da arquitetura do dispositivo no Catapult.

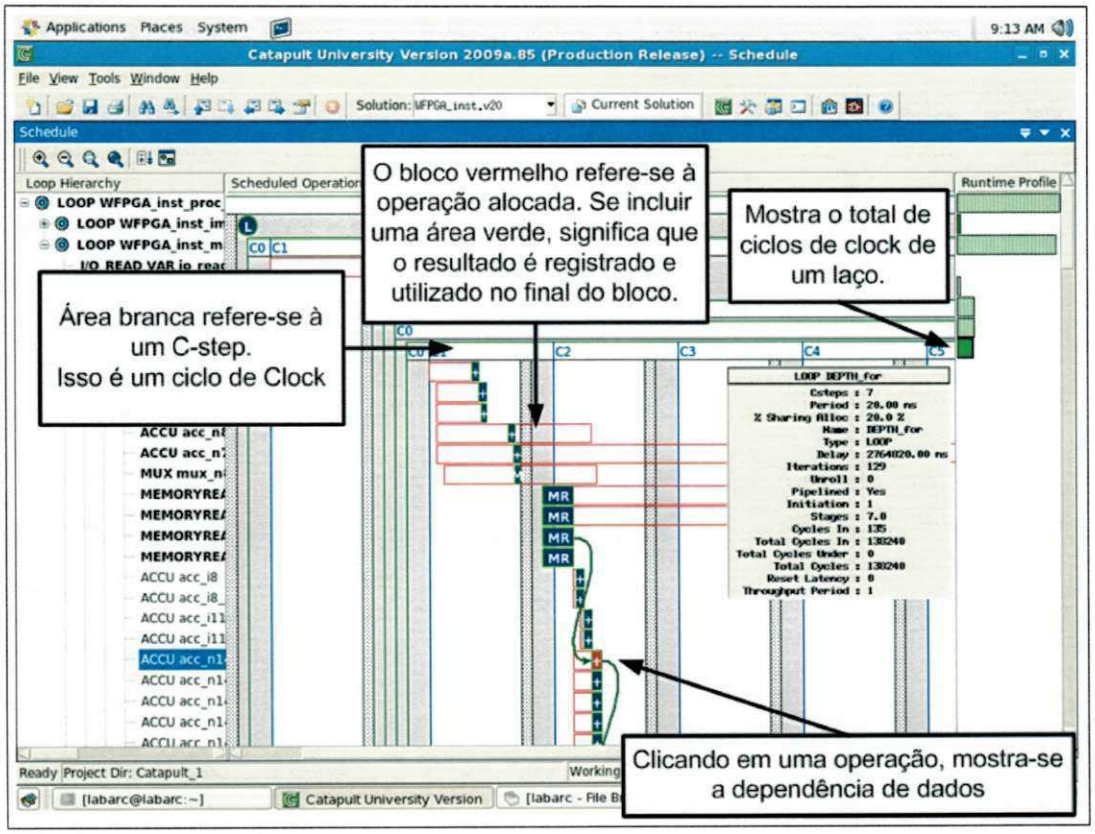


A parte de análise do projeto é realizada nos passos **Schedule** e **Generate RTL**. Na análise, questões como tempo, área e *throughput* garantem que a implementação corresponde à especificação do circuito. O passo **Schedule** refere-se à alocação de recursos, como multiplicadores, somadores, em ciclos de *clock*. Determina-se a latência e o *throughput* do projeto. A ferramenta mostra um gráfico com a descrição do projeto (Figura 3.9).

Por fim, o passo **Generate RTL** que gera código RTL sintetizável nas linguagens de Verilog, VHDL além de outros relatórios.

Portanto, o uso da ferramenta Catapult C oferece várias opções de parâmetros com descrição do código em alto nível. Para ser sintetizável, o código precisa seguir algumas restrições, como determinação do tamanho de memória. Outro exemplo, é o uso de recursos ilimitados, em que funções recursivas sem limite de operações caem no problema de alocação dinâmica

Figura 3.9: Gráfico produzido pela ferramenta no passo Schedule.



de memória (ZURITA, 2009).

3.2.5 Síntese Lógica

Na Síntese Lógica, caracteriza-se pelo refinamento da descrição RTL. Neles, características dos blocos lógicos são conhecidas. O artefato de saída desse processo é denominado de *Netlist*, que descreve as portas lógicas, *flip-flops* e a conectividade do circuito integrado.

Para realizar a Síntese Lógica, utilizou-se a ferramenta *Altera Quartus* ®(ALTERA, 2012). Cada quantidade de bits fracionários na representação dos valores do sistema foi sintetizada separadamente.

Capítulo 4

Apresentação e Análise dos Resultados

Nesse capítulo, são apresentados os resultados alcançados por esse trabalho de dissertação. Esses foram realizados na plataforma PC e na plataforma FPGA. Na plataforma PC, considera-se a solução implementada em *software* com os módulos equivalentes ao desenvolvido na plataforma FPGA. Ao final, realiza-se um comparativo referente ao consumo energético.

4.1 Metodologia

Nos experimentos aqui apresentados, utilizou-se a ferramenta de síntese em alto nível *Mentor Catapult C* para modelar e gerar o código RTL do sistema, e a ferramenta *PowerPlay Power Analysis Tool* da *Altera Quartus* para síntese lógica e física e para fornecer uma estimativa da potência utilizada pelo dispositivo. Cada configuração da implementação foi analisada separadamente, permitindo um estudo individual dos resultados.

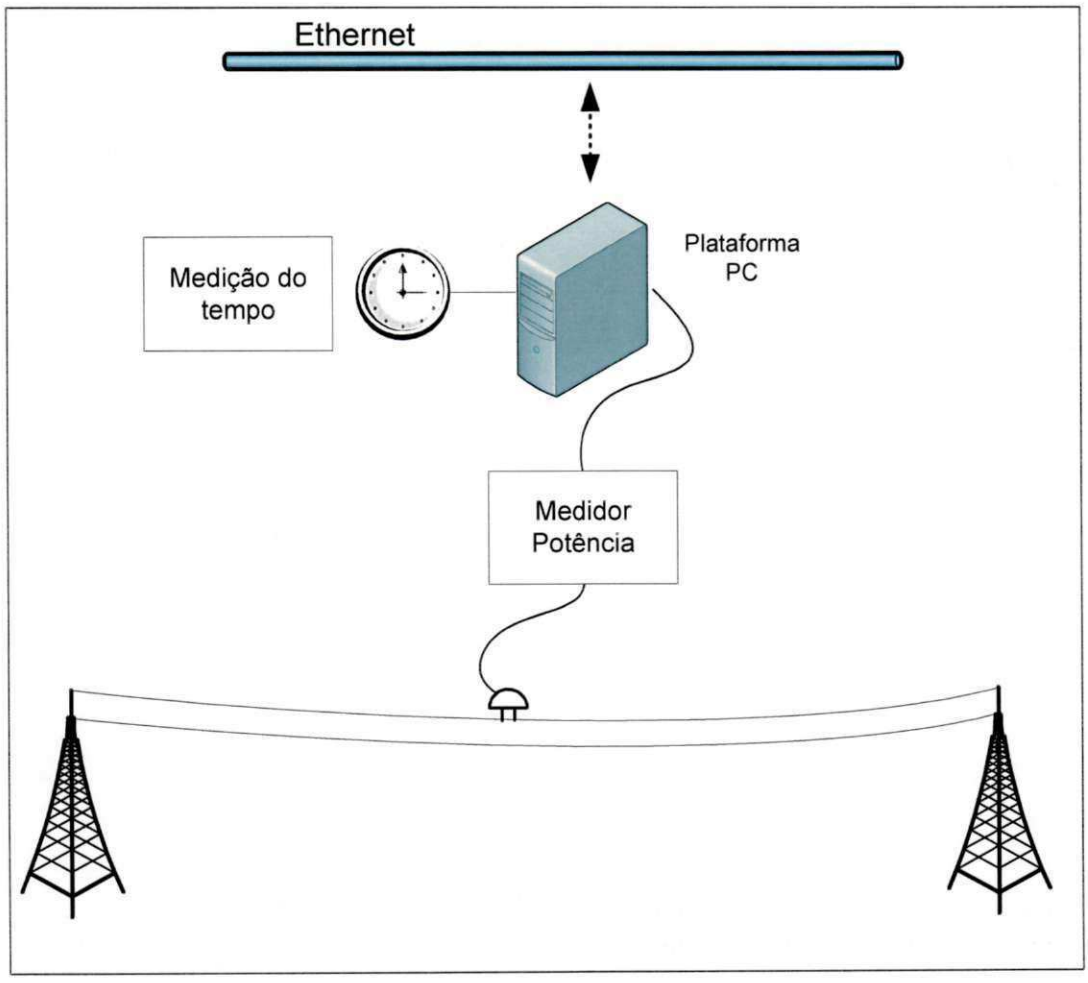
4.2 Experimento 1: Plataforma PC

Esse experimento foi realizado com 1200 imagens pré-processadas com resolução 128x128 *pixels*. Toda computação realizada na plataforma PC foi equivalente aos módulos desenvolvido na plataforma FPGA. A configuração do computador utilizado nos experimentos foi Intel Core 2 Duo, 2,4 GHz com 3GB espaço de memória RAM.

Inicialmente, calculou-se a potência gasta pelo computador no processamento de con-

teúdo visual. Esse teve em média uma potência de 103 Watts[1]. Conforme ilustrada na Figura 4.1, esse experimento foi realizado com o computador ligado a um medidor de potência na execução dos módulos de indexação do Guaatupi. Para buscar essa medida, anotaram-se as diversas potências de saída de um *no break*, calculando-se ao final a média das potências. Essa medida inclui os periféricos (interface *ethernet*, memória RAM) utilizados na aquisição e no processamento do conteúdo visual presente no computador. Para cada imagem, calculou-se o tempo de processamento.

Figura 4.1: Diagrama com os procedimentos realizados na plataforma PC.



Na tabela 4.1 é apresentado a sumarização do resultado obtido no tempo de processamento das imagens. Considera-se a mediana sobre os valores observados, uma vez que essa

[1]Experimento realizado no laboratório de Arquiteturas Dedicadas da Universidade Federal de Campina Grande - UFCG.

medida não é tão sensível aos *outliers* (valores muito menores e muito maiores).

Tabela 4.1: Sumarização do tempo de processamento das imagens no computador *PC* em segundos (s).

Mínimo	1º Quartil	Mediana	Média	3º Quartil	Máximo
0,0470	0,078	0,0935	0,0897	0,0940	0,1870

Com isso, o consumo energético é de:

$$E_{PC} = 103W \times 0,0935s \approx 9,63J \tag{4.1}$$

Portanto, esse valor refere-se à quantidade de energia utilizada para processar uma imagem no sistema de indexação, utilizando uma CPU com processador de uso geral.

4.3 Experimento 2: Plataforma FPGA

Para calcular a potência total utilizada na plataforma FPGA, considerou-se três aspectos: (i) a potência dissipada na fonte chaveada utilizada na alimentação do dispositivo, (ii) a interface de rede que alimenta o dispositivo com dados de entrada e saída, e (iii) o circuito que implementa o algoritmo de indexação. Na Figura 4.2, é ilustrada o procedimento realizado no cálculo da potência utilizada pela plataforma em FPGA. Ressalva-se que somente foi realizado a simulação do circuito sem a prototipagem na placa de FPGA.

No circuito que implementa o algoritmo de indexação, utilizou-se 22 bits para representar os valores internos. Esses são utilizados no processamento dos coeficientes da assinatura da imagem com precisão adequada. Para cada assinatura da imagem, contou-se a quantidade de valores diferentes. Sendo 0, nenhum valor (assinatura da imagem idêntica), e 60 todos os valores diferentes. Esse experimento foi realizado com o uso da biblioteca *Mentor Graphics Algorithmic C*.

Conforme se observa na Tabela 4.2, a sumarização dos dados mostra a média das diferenças entre as assinaturas das imagens. Buscou-se identificar diferenças estatísticas entre os experimentos.

Figura 4.2: Processo utilizado para calcular a energia gasta na plataforma em FPGA.

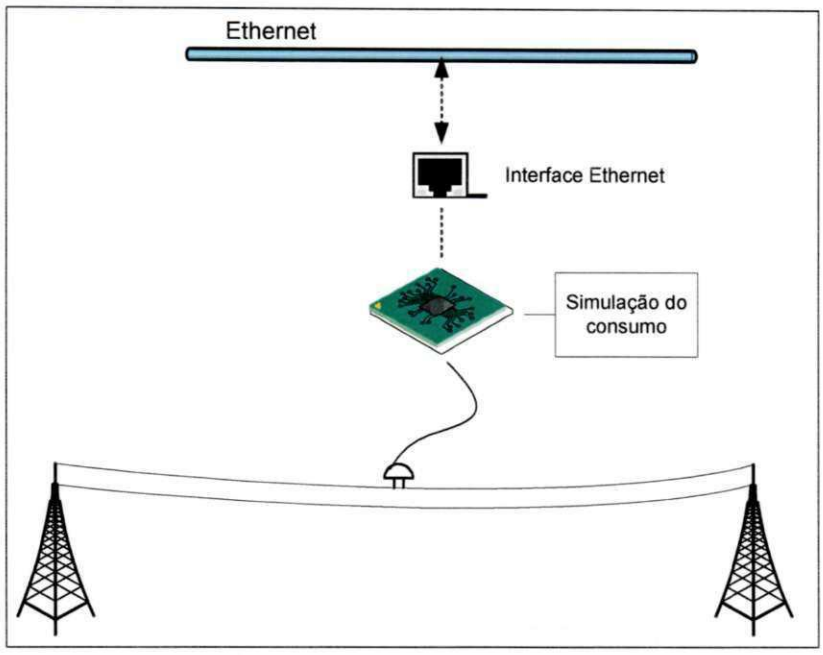


Tabela 4.2: Sumarização dos experimentos. Destacam-se os tratamentos de T21 ao T24 com médias muito próximas.

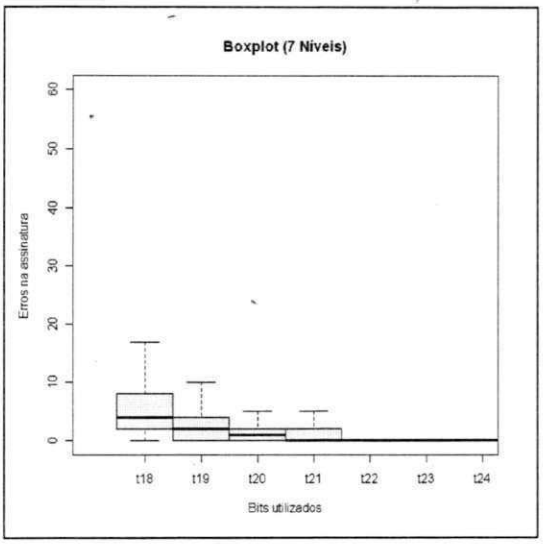
	Quantidade de bits							
	T18	T19	T20	T21	T22	T23	T24	**Média Geral**
Média (μ)	5,51	3,20	1,79	0,92	0,50	0,28	0,19	1,77
Desvio Padrão (σ)	4,87	3,58	2,55	1,76	1,33	0,98	0,96	

Como ilustrado na Figura 4.3, com o gráfico *boxplot* não foi possível determinar diferenças visuais entre os experimentos, uma vez que todos os intervalos de confiança se sobrepõem.

Com isso, como ilustra na Figura 4.4, foram criados gráficos de resíduos padronizados pelos valores teóricos de distribuição normal, uma vez que testes com dados normais são mais confiáveis. Foram analisados os gráficos entre os experimentos e realizados os testes de *Shapiro* e *Anderson Darlin*, não sendo encontrado um padrão normal nos dados das amostras.

Uma vez verificado que os dados não são normais, realizou-se o teste de *Kruskal-Wallis* a fim de encontrar diferença entre as médias. Nela, identificaram-se diferenças significativas

Figura 4.3: Boxplot dos experimentos com tamanho de bits.



entre os experimentos na faixa de 18 e 24 bits, ou seja, existe pelo menos uma média entre os sete grupos com diferenças significativas. Com isso, realizou-se um teste mais específico, o *Multiple Comparisons Kruskal* (JAIN, 2008). Conforme se observa a partir da Tabela 4.3, tem-se diferenças significativas entre os tratamentos T18 a T22. Contudo, entre os tratamentos de T22 a T24, as médias entre os erros são estatisticamente equivalentes. Com isso, utilizou-se de 22 bits nas variáveis do algoritmo de indexação sobre a plataforma em *hardware*.

Esses testes foram realizados com auxílio da ferramenta *R-Statistic* (HORNIK, 2012). Para mais informações sobre os testes estatísticos realizados, vide Raj Jain (JAIN, 2008).

Na Síntese Lógica foi comparada todos os tratamentos a fim de descobrir possível relação entre a potência, tempo e número de elementos lógicos. Conforme ilustrada na Tabela 4.4, observa-se uma relação crescente entre a quantidade de bits utilizada e a quantidade de elementos lógicos utilizados pelo dispositivo. Em contrapartida, pouca variação ocorre na potência dissipada e no tempo de latência do dispositivo.

Portanto, a utilização de 22 bits na representação dos valores do processo de indexação do conteúdo visual apresentou melhores resultados e com aproximadamente a mesma potência dissipada das versões comparadas.

Na transmissão dos valores da imagem, para o dispositivo, somente 16 bits são utilizados,

Figura 4.4: Gráfico da análise de resíduos. Esse gráfico demonstra a análise de normalidade em função da distribuição dos dados.

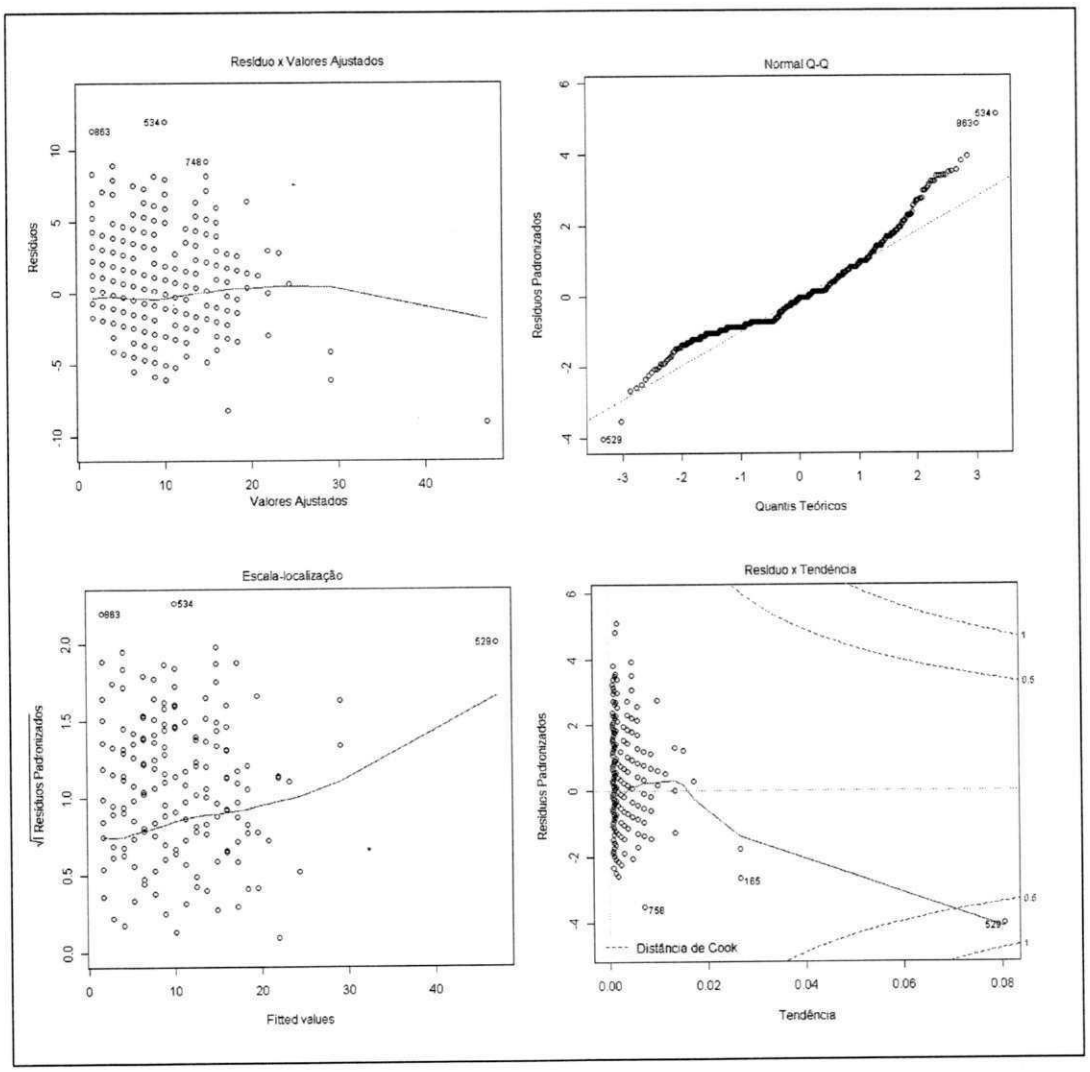


Tabela 4.3: Resultado do teste estatístico com os tratamentos de bits.

Tratamento	**Observação**	**Limiar crítico**	**Diferença**
T18-T19	1028.01	299.13	SIM
T18-T20	1993.80	299.13	SIM
T18-T21	2824.50	299.13	SIM
T18-T22	3378.54	299.13	SIM
T18-T23	3677.01	299.13	SIM
T18-T24	3836.88	299.13	SIM
T19-T20	965.79	299.13	SIM
T19-T21	1796.49	299.13	SIM
T19-T22	2350.53	299.13	SIM
T19-T23	2649.00	299.13	SIM
T19-T24	2808.87	299.13	SIM
T20-T21	830.69	299.13	SIM
T20-T22	1384.73	299.13	SIM
T20-T23	1683.20	299.13	SIM
T20-T24	1843.08	299.13	SIM
T21-T22	554.03	299.13	SIM
T21-T23	852.51	299.13	SIM
T21-T24	1012.38	299.13	SIM
T22-T23	298.47	299.13	NÃO
T22-T24	458.34	299.13	SIM
T23-T24	159.87	299.13	NÃO

Tabela 4.4: Sumarização da síntese do dispositivo em FPGA.

Num. Bits	Potência (W)	Tempo (ms)	Num. Elementos Lógicos
18	0,1979	16,4	5,295
19	0,1982	16,4	5,730
20	0,1984	16,4	5,796
21	0,1986	16,4	5,969
22	0,1989	16,4	6,026
23	0,1992	16,5	6,303
24	0,1994	16,5	6,682

uma vez que esses representam todos os valores utilizados pelo padrão YIQ. Para os dados de saída, somente 8 bits são necessários para representar a posição do *pixel* utilizado na assinatura da imagem.

A potência medida na implementação com uso de 22 bits foi de aproximadamente *0,199 W* (Tabela 4.4). Utilizando o joule como medida de energia, observa-se um consumo energético em Joules (J) de:

$$E_{FPGA} = 0,199W \times 0,0164s \approx 0,0033J \tag{4.2}$$

Esse valor equivale ao processamento em 1 canal de cor da imagem. A quantidade total utilizada no processamento da imagem é três vezes maior, totalizando um consumo energético de *0,0099 J*.

Para o controlador da interface *ethernet*, considera-se a potência para transmitir e receber dados, assim como, no estado *standby* (momento em que ocorre o processamento dos coeficientes da imagem). Como se encontra no *datasheet* (SHEET, 2005), a potência utilizada pelo controlador *ethernet* ao receber dados é de *0,125 W*. Quando se utiliza a interface para transmitir, a potência utilizada sobe para *0,303 W*. Vale ressaltar, que foi proposto para este trabalho usar diretamente quadros *ethernet*, sem pilha de protocolo TCP/IP. Dessa forma, a entrada/saída dos dados do FPGA se dá pela simples transferência da carga útil dos quadros *ethernet* entre o FPGA e o controlador *ethernet*. A energia gasta nesta transferência está inclusa no consumo do controlador *ethernet*.

Para enviar os valores dos *pixels* da imagem, considera-se o total de 16.384 valores em cada canal de cor. Cada valor é formado por 16 bits, que corresponde aos valores utilizados no padrão de cor YIQ. Portanto, para os três canais de cor, foram necessários 786.432 bits.

Considera-se uma largura de banda média 80 Mb/s da rede *ethernet*. Com isso, são necessários aproximadamente 0,009 segundos para transmitir os dados de entrada. Para os dados de saída, é enviado um valor de 22 bits com a média do canal de cor e 120 valores utilizado na assinatura da imagem. Cada valor é formado por 8 bits que corresponde às posições de *pixels* dos coeficientes selecionados após o processamento da imagem. Com isso, serão gastos 0,000012 segundos na transmissão dos 982 bits que formam a assinatura.

Portanto, conforme mostrada na Tabela 4.5, observa-se que os valores utilizados pela controladora *ethernet* influencia o total gasto na plataforma em FPGA.

Tabela 4.5: Sumarização da energia utilizada na plataforma FPGA.

	Potência (W)	Tempo (s)	Energia (J)
FPGA	0,597	0,0164	0,0099
Ethernet (Entrada)	0,125	0,0090	0,0011
Ethernet (Saída)	0,303	0,000012	0,000004
Ethernet (Standby)	0.069	0,0075	0,0005
		Total	**0,0115**

Considera-se que 10% da potência utilizada na fonte chaveada é desperdiçada na forma de calor. Portanto, o consumo total energético é de *0,0127 J* (NISHIMURA et al., 2008).

4.4 Análise dos Resultados

Esta seção são analisados os resultados obtidos nesse trabalho. Realiza-se um comparativo entre os resultados alcançados nessa dissertação.

O consumo energético calculado na plataforma PC foi de 758 vezes maior do que o consumo da plataforma FPGA. Na plataforma PC, com *clock* de 2,4 GHz, se gasta *43 nJ* a cada ciclo na execução do algoritmo de indexação. Para plataforma FPGA, com *clock* de 50 MHz, se gasta aproximadamente *12 nJ*. Isso equivale a um consumo quatro vezes maior a

cada ciclo na plataforma PC. Considera-se que a quantidade de transistores utilizados em um Core 2 Duo que é de 290 milhões (INTEL, 2012). Na solução em FPGA, têm-se cerca de 21 milhões de transistores, ou seja, o processador do PC tem quatorze vezes mais transistores. Na Tabela 4.6, é apresentada a sumarização da análise realizada entre as plataformas.

Tabela 4.6: Sumarização dos resultados obtidos nas plataformas PC e FPGA.

Plataforma	Potência (W)	Tempo (s)	Clock (MHz)	Energia (J)	Ciclos (milhões)	Energia/ciclo (nJ)	Transistores (milhões)
PC	103	0,0935	2,400	9,63	225	43	290
FPGA	0,6	0,0495	50	0,0127	2,5	12	21

Para processar uma imagem, a solução em PC utiliza aproximadamente 225.000.000 ciclos de *clock*. Na plataforma FPGA são utilizados 2.450.000 ciclos. Com isso, o PC precisa de 92 ciclos de *clock* para fazer o quê o FPGA faz em 1 ciclo.

Concluindo, o circuito implementado no FPGA é dedicado para fazer exatamente a operação específica que precisa ser feita em cada ciclo de *clock*. O processador do PC precisa de mais passos no seu circuito de propósito geral e envolve uma maior quantidade de transistores para realizar a mesma operação.

Capítulo 5

Considerações Finais e Sugestões para Trabalhos Futuros

Neste trabalho, foi realizado um estudo da utilização da plataforma *FPGA* como alternativa energética aos computadores PC. Um estudo de caso foi aplicado ao domínio da computação visual. Foi implementado na plataforma *FPGA* o mesmo algoritmo utilizado na plataforma PC. Ao final, foi comparada a energia gasta de ambas as soluções.

5.1 Desafios Encontrados

Nesta seção, são destacados os principais desafios superados no desenvolvimento deste trabalho. Discute-se o tempo de aprendizagem no desenvolvimento do projeto sobre as plataformas citadas.

Inicialmente, na análise da ferramenta de síntese de alto nível (*Mentor Catapult C*), buscou-se entender e verificar a utilização da ferramenta no desenvolvimento do projeto. Como se trata de uma ferramenta recente, pouco utilizada no Brasil, e sem nenhum material oficial de treinamento; utilizou-se o material de ajuda da ferramenta e grupos de discussão sobre o assunto. Com esse material, desenvolveu-se os primeiros programas simples (com divisores, raiz quadrada etc).

Foi realizada uma análise dos elementos lógicos utilizados pelo código gerado depois de sintetizado. Comparou-se com aplicações desenvolvidas manualmente em linguagem de descrição de *hardware*. Essa teve um incremento de aproximadamente 20% no número de

elementos lógicos. Com isso, viabilizou-se a utilização da ferramenta com ressalva de que mais estudos precisam ser realizados.

Após isso, desenvolveu-se um algoritmo de filtro de *Sobel* (GONZALEZ et al., 1998). Esse mais complexo e com processamento em tempo real. Na síntese lógica, a utilização da ferramenta *Altera Quartus* inicialmente apresentou erros. Dentre soluções para contornar o problema, alteraram-se as configurações da ferramenta Catapult C. Dentre essas, tem-se o incremento no número de registradores utilizados pela aplicação e a opção de gerar código em VHDL. Após isso, descobriu-se que o problema estava na memória gerada pela ferramenta *Catapult C*, não sintetizada pela ferramenta *Altera Quartus* (ferramenta não homologada). Essa somente inferiu memória com a utilização do código em VHDL ao invés de Verilog.

5.1.1 Tempo de Aprendizagem

Considera-se o tempo para aprender a linguagem de programação Java durante um curso de graduação. Com média de estudos de 4 horas por semana, com dois semestres de 4 meses em 1 ano, têm-se ao final de 1 ano cerca de 128 horas. Multiplica-se esse valor, pelo tempo médio de graduação de 4 anos, totalizando um estudo de 512 horas.

No desenvolvimento sobre a plataforma FPGA, considera-se uma rotina de estudos na média de 16 horas por semana. Considera-se o tempo de 10 meses para realizar o projeto. Com isso, têm-se ao final desse período um total de 640 horas.

Diante do novo paradigma no desenvolvimento em *hardware*, esse requer uma maior quantidade de estudos. Ressalvam-se ainda que, o conhecimento desenvolvido com a ferramenta de síntese de alto nível não exclui a necessidade de conhecimento nas linguagens de descrição de *hardware* como Verilog ou VHDL. Isso ocorre pelo fato que o código gerado pela ferramenta, em alguns casos, precisa ser adaptado para situações específicas.

5.2 Sugestões para Trabalhos Futuros

Os resultados aqui obtidos demonstram uma eficiência energética de 758 vezes maior da plataforma FPGA sobre a plataforma PC. Outros trabalhos como em (THOMAS; HOWES; LUK, 2009; KESTUR; DAVIS; WILLIAMS, 2010), também alcançaram eficiência na magnitude de centenas de vezes. Entretanto, ressalva-se que mais estudos precisam ser elaborados com

outros casos e aplicações. Dentre esses, destacam-se:

- Comparação com outras plataformas: Notebook, PDA, GPU.
- Implementação do software em C em substituição à Java.
- Implementação de outros algoritmos.
- Otimização dos algoritmos implementados em ambas as plataformas.

Bibliografia

ALTERA. *Altera Quartus*. 2012. Disponível em: <http://www.altera.com/products/software/quartus-ii/subscription-edition/qts-se-index.html>. Acesso em: 06 jul. 2012.

ATABANY, W.; DEGENAAR, P. Parallelism to reduce power consumption on FPGA spatiotemporal image processing. *2008 IEEE International Symposium on Circuits and Systems*, Ieee, p. 1476–1479, maio 2008. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4541708>.

BARBACCI, M. R. Instruction set processor specifications (isps): the notation and its applications. *IEEE Trans. Comput.*, IEEE Computer Society, Washington, DC, USA, v. 30, n. 1, p. 24–40, jan. 1981. ISSN 0018-9340. Disponível em: <http://dl.acm.org/citation.cfm?id=1963620.1963623>.

BARTOLINI, I. *Efficient and Effective Similarity Search in Image Databases*. Tese (Doutorado), 2001. Disponível em: <http://www-db.deis.unibo.it/research/papers/theses/bartolini.pdf>.

CALWELL, C.; OSTENDORP, P. 80 plus: a strategy for reducing the inherent environmental impacts of computers. In: IEEE. *Electronics and the Environment, 2005. Proceedings of the 2005 IEEE International Symposium on*. [S.l.], 2005. p. 151–156. ISBN 0780389107. ISSN 1095-2020.

CARSON, C. et al. Blobworld: Image segmentation using expectation-maximization and its application to image querying. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, IEEE, v. 24, n. 8, p. 1026–1038, 2002. ISSN 0162-8828.

CLIMATE. *Climate Savers Computing*. 2010. Disponível em: <http://www.climatesaverscomputing.org>. Acesso em: 21 nov 2010.

CNN. *Computer decodes Mona Lisa's smile.* 2005. Disponível em: <http://www.cnn.com/2005/TECH/12/16/mona.lisa.smile/index.html>. Acesso em: 12 de set 2012.

COMPTON, K. An introduction to reconfigurable computing. *IEEE Computer, Apr*, 2000. Disponível em: <http://www.dimap.ufrn.br/ ivan/reconfiguraveis/IntroductionReconfComputing.pdf>.

CONSTANTINIDES, G. A.; CHEUNG, P. Y. K.; LUK, W. Synthesis of saturation arithmetic architectures. *ACM Trans. Des. Autom. Electron. Syst.*, ACM, New York, NY, USA, v. 8, n. 3, p. 334–354, jul. 2003. ISSN 1084-4309. Disponível em: <http://doi.acm.org/10.1145/785411.785415>.

COREL. *Corel Image Database*. Disponível em: <http://carter.idiap.ch/databases.html>. 1999. Acesso em: 06 Ago 2012.

COUSSY, P.; MORAWIEC, A. *High-Level Synthesis From Algorithm to Digital Circuit*. Springer, 2008. ISBN 9781402085871. Disponível em: <http://books.google.com.br/books?id=lEWRGB5JlHkC>.

CSILLAGHY, A.; HINTERBERGER, H.; BENZ, A. Content-based image retrieval in astronomy. *Information Retrieval*, Springer, v. 3, n. 3, p. 229–241, 2000. ISSN 1386-4564.

DATTA, R. et al. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys (CSUR)*, ACM, v. 40, n. 2, p. 1–60, 2008. ISSN 0360-0300.

DAUBECHIES, I. *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, 1992. (Cbms-Nsf Regional Conference Series in Applied Mathematics). ISBN 9780898712742. Disponível em: <http://books.google.com.br/books?id=9t5SG06AiT0C>.

DEMPSTER, A. et al. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, Royal Statistical Society, v. 39, n. 1, p. 1–38, 1977. ISSN 0035-9246.

DOULOS. 2012. Disponível em: <http://www.doulos.com/knowhow/vhdl_designers_guide>. Acesso em: 03 jan. 2012.

DU, Q.; FABER, V.; GUNZBURGER, M. Centroidal Voronoi tessellations: applications and algorithms. *SIAM review*, JSTOR, v. 41, n. 4, p. 637–676, 1999. ISSN 0036-1445.

ECOINFORMATICA. 2011. Disponível em <http://www.catalogosustentavel.com.br>. Acesso em: 21 fev. 2011.

ERGEN, B.; BAYKARA, M. Content based medical image retrieval feature extraction of using statistical spatial methods for content based medical image retrieval. In: IEEE. *Signal Processing and Communications Applications Conference (SIU), 2010 IEEE 18th*. [S.l.], 2010. p. 692–695.

FACEBOOK. 2012. Disponível em: <http://www.facebook.com>. Acesso em: 05 jan. 2012.

FLICKNER, M. et al. Query by Image and Video Content: The QBIC System. *Computer*, Published by the IEEE Computer Society, p. 23–32, 1995. ISSN 0018-9162.

FLICKR. 2012. Disponível em: <http://www.flickr.com>. Acesso em: 12 de set 2012.

FOWERS, J.; BROWN, G.; COOKE, P. A performance and energy comparison of FPGAs, GPUs, and multicores for sliding-window applications. *Proceedings of the ACM/SIGDA*, p. 47–56, 2012. Disponível em: <http://dl.acm.org/citation.cfm?id=2145704>.

GAJSKI, D. Introduction to High-Level Synthesis. *Technology*, 1990.

GONZALEZ, R. C. et al. Image processing. *IEEE Transactions on Image Processing*, v. 7, n. 3, p. 359–369, 1998.

GRUNDBERG, S.; ROLANDER, N. For Data Center, Google Goes for the Cold. *Wall Street Journal*, September 2011. Disponível em <http://online.wsj.com/article/SB10001424053111904836104576560551005570810.html>. Acesso em: 09 jan. 2012.

GUIMARAES, G. F. et al. Fpga infrastructure for the development of augmented reality applications. In: *Proceedings of the 20th annual conference on Integrated circuits and systems design*. New York, NY, USA: ACM, 2007. (SBCCI '07), p. 336–341. ISBN 978-1-59593-816-9. Disponível em: <http://doi.acm.org/10.1145/1284480.1284568>.

HORNIK, K. *The R FAQ*. 2012. ISBN 3-900051-08-9. Disponível em: <http://CRAN.R-project.org/doc/FAQ/R-FAQ.html>.

HUANG, J. et al. *Image indexing using color correlograms*. [S.l.]: Google Patents, jun 2001. US Patent 6,246,790.

HUANG, Z. et al. Content-Based Video Scarch: is there a need, and is it possible? In: IEEE. *Information-Explosion and Next Generation Search, 2008. INGS'08. International Workshop on*. [S.l.], 2008. p. 12–19.

INTEL. *Microprocessor Quick Reference Guide*. 2012. Disponível em: <http://www.intel.com/pressroom/kits/quickreffam.htm>. Acesso em: 01 jun. 2012.

JACOBS, C. E.; FINKELSTEIN, A.; SALESIN, D. H. Fast multiresolution image querying. *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques - SIGGRAPH '95*, ACM Press, New York, New York, USA, v. 95, n. January, p. 277–286, 1995. Disponível em: <http://portal.acm.org/citation.cfm?doid=218380.218454>.

JAIN, R. *The Art Of Computer Systems Performance Analysis:*. Wiley India Pvt. Limited, 2008. ISBN 9788126519057. Disponível em: <http://books.google.com.br/books?id=eOR0kJjgMqkC>.

KESTUR, S.; DAVIS, J. D.; WILLIAMS, O. BLAS Comparison on FPGA, CPU and GPU. *2010 IEEE Computer Society Annual Symposium on VLSI*, Ieee, p. 288–293, jul. 2010. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5572788>.

KHAN, S.; SHOAB, K. *Digital Design of Signal Processing Systems: A Practical Approach*. John Wiley & Sons, 2011. ISBN 9780470741832. Disponível em: <http://books.google.com.br/books?id=EWJMg2jDLHAC>.

KOTOULAS, L.; ANDREADIS, I. Parallel Local Histogram Comparison Hardware Architecture for Content-Based Image Retrieval. *Journal of Intelligent and Robotic Systems*, v. 39, n. 3, p. 333–343, mar. 2004. ISSN 0921-0296. Disponível em: <http://www.springerlink.com/openurl.asp?id=doi:10.1023/B:JINT.0000021021.10780.af>.

LIN, Y.-L. Recent developments in high-level synthesis. *ACM Trans. Des. Autom. Electron. Syst.*, ACM, New York, NY, USA, v. 2, n. 1, p. 2–21, jan. 1997. ISSN 1084-4309. Disponível em: <http://doi.acm.org/10.1145/250243.250245>.

MA, W.; MANJUNATH, B. Netra: A toolbox for navigating large image databases. *Multimedia Systems*, Springer, v. 7, n. 3, p. 184–198, 1999. ISSN 0942-4962.

MAHMOUDI, F. et al. Image retrieval based on shape similarity by edge orientation autocorrelogram. *Pattern recognition*, Elsevier, v. 36, n. 8, p. 1725–1736, 2003. ISSN 0031-3203.

MANJUNATH, B.; MA, W. Texture Features for Browsing and Retrieval of Image Data. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, Published by the IEEE Computer Society, p. 837–842, 1996. ISSN 0162-8828.

MARTENS, E. S. J.; GIELEN, G. G. E. *High-Level Modeling and Synthesis of Analog Integrated Systems*. Tese (Doutorado), 2008.

MARTIN, G.; SMITH, G. High-level synthesis: Past, present, and future. *Design & Test of Computers, IEEE*, IEEE, v. 26, n. 4, p. 18–25, 2009. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5209959>.

MCFARLAND, M.; PARKER, A. C.; CAMPOSANO, R. The high-level synthesis of digital systems. *Proceedings of the IEEE*, v. 78, n. 2, p. 301–318, 1990. ISSN 00189219. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=52214>.

MENESES, T. F.; FILHO, C. A. P.; ARAUJO, R. W. M. Guaatupi : Um ambiente para indexação e recuperação de imagens da web sem redundância visual . In: *WebMedia 2010 - Artigos completos e Resumos*. Belo Horizonte: [s.n.], 2010. p. 1–4.

MENTOR. *Mentor Catapult C*. 2012. Disponível em: <http://www.calypto.com/catapult_c_synthesis.php>. Acesso em: 06 jul. 2012.

MIRSKY, S. *Computers get the picture. Sci Amer.* Nov 2006.

MISITI, M.; MISITI, Y.; OPPENHEIM, G. Wavelet Toolbox. *User's Guide*, 1997. Disponível em: <http://www.mathworks.fr/help/pdf_doc/wavelet/wavelet_ug.pdf>.

MOORE, G. E. Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp.114. *IEEE SolidState Circuits Newsletter*, [New York, NY]: Institute of Electrical and Electronics Engineers, v. 20, n. 3, p. 33–35, 2006. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4785860>.

MURPHY-CHUTORIAN, E.; ROSENBERG, C. *Similar Images graduates from Google Labs*. 2009. Disponível em: <http://googleblog.blogspot.com/2009/10/similar-images-graduates-from-google.html>. Acesso em: 08 jun. 2012.

NISHIMURA, K. et al. Novel double chopper type ac-dc power supply with reduced dc voltage ripple for pfc converter and modified circuit topologies. In: *Industrial Electronics, 2008. ISIE 2008. IEEE International Symposium on*. [S.l.: s.n.], 2008. p. 148 –153.

PAINTER, T. et al. Retrieval of subpixel snow-covered area and grain size from imaging spectrometer data. *Remote Sensing of Environment*, Elsevier, v. 85, n. 1, p. 64–77, 2003. ISSN 0034-4257.

RIBEIRO, A. A. d. L. *Reconfigurabilidade dinâmica e remota de FPGA*. Dissertação (Mestrado) — Universidade de São Paulo, São Carlos, 2002.

RODRIGUES, S. d. T. O. *Investigação de Técnicas para Extração de Características usando Redes GHSOM Aplicadas à Recuperação de Imagens por Conteúdo*. Dissertação (Mestrado) — Universidade Federal de Campina Grande, PB, 2008.

RUI, Y.; HUANG, T.; CHANG, S. Image Retrieval: Current Techniques, Promising Directions, and Open Issues. *Journal of visual communication and image representation*, Elsevier, v. 10, n. 1, p. 39–62, 1999. ISSN 1047-3203.

SAADATMAND-TARZJAN, M.; MOGHADDAM, H. A novel evolutionary approach for optimizing content-based image indexing algorithms. *Systems, Man, and Cybernetics,*

Part B: Cybernetics, IEEE Transactions on, IEEE, v. 37, n. 1, p. 139–153, 2007. ISSN 1083-4419.

SCHETTINI, R. et al. A survey of methods for colour image indexing and retrieval in image databases. *Color Imaging Science: Exploiting Digital Media*, Citeseer, p. 183–211, 2001.

SCHNITER, P. *Downsampling*. Disponível em: <http://cnx.org/content/m10441/2.12/>. Acesso em: 03 jan. 2012.

SCHRODER, M. et al. Interactive learning and probabilistic retrieval in remote sensing image archives. *Geoscience and Remote Sensing, IEEE Transactions on*, IEEE, v. 38, n. 5, p. 2288–2298, 2002. ISSN 0196-2892.

SHEET, D. DM9000A Ethernet Controller with General Processor Interface DM9000A. *Control*, 2005.

SMITH, J. R. *Integrated spatial and feature image systems: Retrieval, analysis and compression.* Tese (Doutorado) — DigitalCommons@Columbia, jan. 01 1997. Disponível em: <http://digitalcommons.libraries.columbia.edu/dissertations/AAI9723852>.

SMITH, J. R.; CHANG, S. F. VisualSEEk: a fully automated content-based image query system. In: ACM. *Proceedings of the fourth ACM international conference on Multimedia.* [S.l.], 1997. p. 87–98. ISBN 0897918711.

STAEDTER, T. *Digital pics read by computer*. 2006. Disponível em: <http://dsc.discovery.com/news/2006/11/09/images_tec.html>. Acesso em: 12 set 2012.

STOLLNITZ, E. J.; DEROSE, T. D. Wavelets for computer graphics: a primer. 2. *Computer Graphics and*, n. September, 1995. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=391497>.

STRICKER, M.; DIMAI, A. Spectral covariance and fuzzy regions for image indexing. *Machine vision and applications*, Springer, v. 10, n. 2, p. 66–73, 1997. ISSN 0932-8092.

SWAIN, M.; BALLARD, D. Color indexing. *International journal of computer vision*, Springer, v. 7, n. 1, p. 11–32, 1991. ISSN 0920-5691.

TAMURA, H.; MORI, S.; YAMAWAKI, T. Textural features corresponding to visual perception. *IEEE Transactions on Systems, Man and Cybernetics*, v. 8, p. 460–473, 1978.

TARR, M. J. Visual pattern recognition. *Encyclopedia of psychology*, p. 66–70, 2000. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.34.7178&rep=rep1&type=pdf>.

THOMAS, D. B.; HOWES, L.; LUK, W. A comparison of cpus, gpus, fpgas, and massively parallel processor arrays for random number generation. In: *Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays*. New York, NY, USA: ACM, 2009. (FPGA '09), p. 63–72. ISBN 978-1-60558-410-2. Disponível em: <http://doi.acm.org/10.1145/1508128.1508139>.

VOTANO, J. R.; PARHAM, M.; HALL, L. H. *Rapid System Prototyping with FPGAs*. [s.n.], 2004. ISBN 9780750678667. Disponível em: <http://onlinelibrary.wiley.com/doi/10.1002/cbdv.200490137/abstract>.

WANG, J.; LI, J.; WIEDERHOLD, G. SIMPLIcity: Semantics-sensitive integrated matching for picture libraries. *IEEE Transactions on pattern analysis and machine intelligence*, Published by the IEEE Computer Society, p. 947–963, 2001. ISSN 0162-8828.

WANG, Z.; CHI, Z.; FENG, D. Fuzzy integral for leaf image retrieval. In: IEEE. *Fuzzy Systems, 2002. FUZZ-IEEE'02. Proceedings of the 2002 IEEE International Conference on*. [S.l.], 2002. v. 1, p. 372–377. ISBN 0780372808.

WU, J. Content-Based Indexing of Multimedia Databases. *IEEE Transactions on Knowledge and Data Engineering*, IEEE Educational Activities Department, v. 9, n. 6, p. 989, 1997. ISSN 1041-4347.

YANG, Z.; KAMATA, S.; AHRARY, A. NIR: Content based image retrieval on cloud computing. In: IEEE. *Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on*. [S.l.], 2009. v. 3, p. 556–559.

YOUTUBE. 2012. Disponível em: <http://www.youtube.com/t/press_statistics>. Acesso em: 04 jun. 2012.

ZURITA, M. E. d. P. V. *Um Fluxo de Prototipagem Rápida em FPGA para Algoritmos de Processamento de Vídeo*. Tese (Doutorado) — Universidade Federal de Campina Grande, 2009.

Apêndice A

Resultados Experimentais do Sistema CBIR

Neste apêndice são demonstrados os resultados obtidos conforme o projeto de implementação do *Guaatupi*. Aborda-se a metodologia utilizada para executar os experimentos e os resultados da busca por imagem exemplo e rascunho.

Para analisar os resultados foram feitos testes a partir de imagens exemplo e rascunhos. Realizaram-se 20 consultas para cada tipo de busca. Nesse primeiro experimento, buscou-se identificar as imagens no sistema. No segundo experimento, para avaliar a precisão e revocação, cópias modificadas de uma imagem foram adicionadas.

A.1 Imagem Exemplo

Neste tipo de consulta o usuário envia ao sistema uma imagem semelhante, ou de baixa resolução que tenha sido digitalizada. O sistema então retorna todas as imagens que possuam semelhanças visuais.

A Figura A.1 ilustra todas as 20 imagens utilizadas para os experimentos. As imagens foram selecionadas aleatoriamente, sendo que todas as imagens possuem uma correspondente na base das imagens.

Nesse experimento, todas as imagens pesquisadas foram identificadas na primeira posição do resultado. A Tabela A.1 mostra os resultados obtidos.

Observa-se que os resultados obtidos a partir de uma imagem exemplo são bastante satis-

Figura A.1: Imagens utilizadas para experimento busca por exemplo.

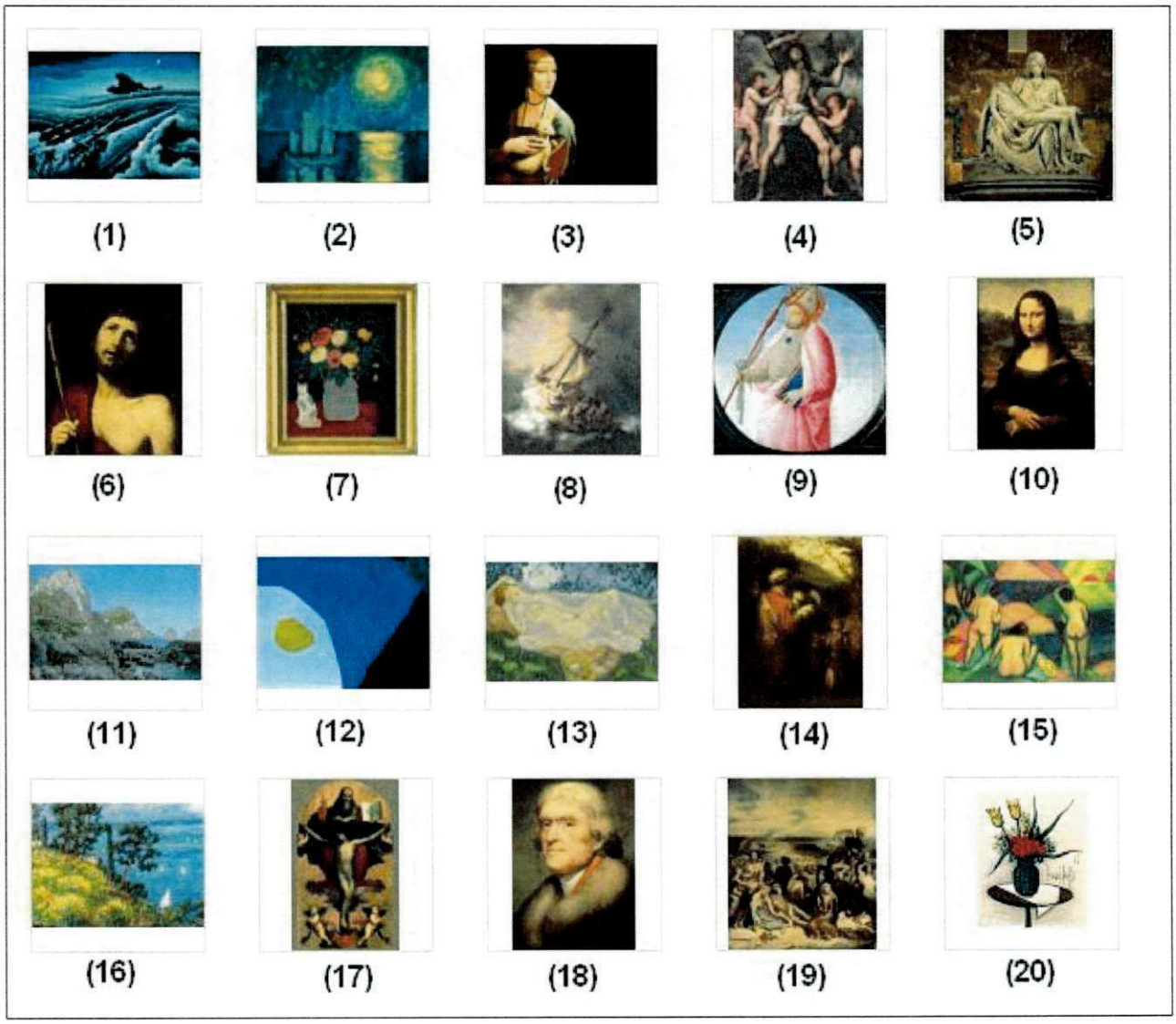


Tabela A.1: Experimento com busca de imagem exemplo.

Imagem	Pos. Recuperada
1	1,2,3
2	1
3	1,2
4	1,2,3
5	1,2,16
6	1
7	1
8	1
9	1,2,3,4,5,6,8
10	1,2,3
11	1,2,3,4,5,6,17
12	1
13	1
14	1
15	1-21
16	1,2
17	1
18	1
19	1,2,3,4
20	1

fatórios. Como as imagens foram indexadas automaticamente, nenhum controle foi realizado na aquisição de cópias duplicadas. Isso explica as várias posições encontradas no resultado da busca. Dentre as cópias dos resultados, observa-se a diferença em alguns aspectos de resolução e qualidade. Pouco esforço foi realizado pelo usuário para encontrar uma imagem.

A.2 Rascunho

Neste tipo de consulta, o usuário desenha o rascunho da imagem que deseja recuperar. Diversos fatores podem influenciar nessa consulta como: as posições dos objetos, a tonalidade das cores e a semelhança de fato com a imagem alvo.

Para os rascunhos, foram selecionadas aleatoriamente 20 imagens. Para desenhá-las, foram observadas as imagens originais pertencentes à base de imagens. A Figura A.2 ilustra os rascunhos utilizados nos experimentos. A Tabela A.2 mostra os resultados obtidos.

Figura A.2: Rascunhos utilizadas para experimento.

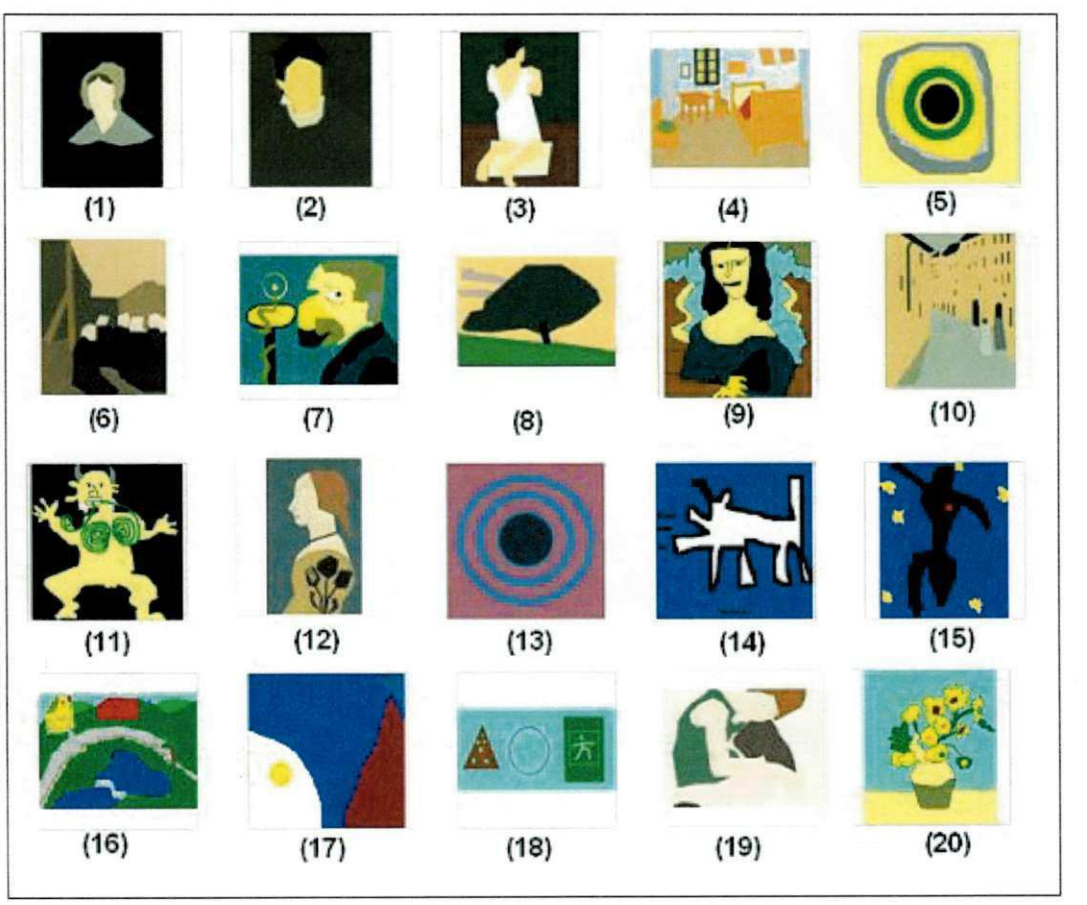


Tabela A.2: Experimento com busca de imagem pelo rascunho.

Rascunho	Pos. Recuperada
1	400
2	1120
3	5406
4	741
5	1215
6	847
7	23
8	251
9	Não encontrada
10	9952
11	Não encontrada
12	8290
13	15
14	80
15	13
16	868
17	Não encontrada
18	704
19	25
20	Não encontrada

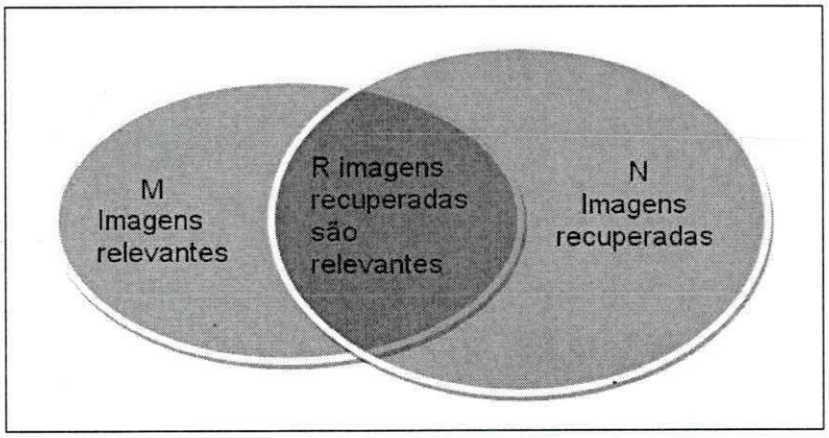
Os resultados obtidos no tipo de busca por rascunho são bastante relevantes. A maioria das imagens foi recuperada entre as 3.000 posições iniciais (13 imagens de 20), isso equivale a percorrer menos de 1% da base de imagens. Algumas imagens não foram encontradas, uma vez que o espaço de pesquisa foi limitado a 10.000 imagens.

A.3 Avaliação da eficiência

Conforme os experimentos anteriores, o sistema foi avaliado buscando uma imagem e analisando o resultado contendo ou não a imagem pesquisada. Muitos dos sistemas de recuperação de imagem por conteúdo consideram o quão realmente semelhante são as imagens recuperadas, sendo essa avaliação muito subjetiva. Para conseguir uma aproximação na avaliação desses sistemas, foi utilizado o método de precisão e revocação. Em uma determinada consulta por similaridade, assume-se:

- Existe um conjunto M de imagens relevantes.
- Quando a consulta é realizada, N imagens são recuperadas.
- Dentre as recuperadas, somente R são relevantes.

Figura A.3: Representação do conjunto de imagens Relevantes (M), imagens recuperadas (N), e das imagens recuperadas que são relevantes (R).



Tendo em mente a disposição desses conjuntos, conforme a Figura A.3, as medidas de revocação e precisão são definidas da seguinte forma:

- **Revocação** é a porção do conjunto de imagens recuperadas relevantes dentre todas as imagens relevantes.

$$Rev = R/M \tag{A.1}$$

•**Precisão** é a porção do conjunto de imagens recuperadas relevantes dentre todas as imagens recuperadas.

$$Pre = R/N \tag{A.2}$$

Para avaliação confiável dos resultados obtidos, estabeleceu-se um conjunto de imagens indexadas manualmente. Sendo indexadas 100 imagens, sendo 1 original com 9 cópias modificadas. Foi utilizada a ferramenta *GIMP* para modificar essas cópias. Para cada cópia, um tipo de transformação foi aplicado, como exemplo, filtro Gaussiano, rotação, imagem posterizada entre outras. A Figura A.4 ilustra uma imagem original com suas cópias modificadas.

Figura A.4: Exemplo de imagens indexadas para avaliação de precisão e revocação. Na parte superior, a imagem original seguida das cópias modificadas.

Foi definido o espaço do conjunto de imagens recuperadas (N) com diversos tamanhos (200, 100, 50, 25 e 20). Para cada imagem consultada, foram calculados os valores de

precisão e revocação. Na Tabela A.3, observa-se a média aritmética dos valores de precisão e revocação para os 5 espaços definidos.

Tabela A.3: Tabela com as médias dos valores de precisão e revocação.

Num. Imagens recuperadas	Revocação	Precisão
200	0.76	0.038
100	0.74	0.074
50	0.72	0.144
25	0.64	0.256
20	0.61	0.305

Conforme se observa no gráfico da Figura A.5, em um sistema ideal os valores de precisão e revocação devem ser iguais a 1. Contudo, na realidade esses valores vão em direção contrária. Quando a consulta é ampla, a revocação é alta mas precisão é baixa. Quando a consulta é restrita, a revocação diminui mas a precisão aumenta.

Portando, deve-se levar em consideração o sistema proposto, uma vez que possui resultado esperado para um sistema de recuperação a partir do conteúdo.

Figura A.5: Gráfico Precisão x Revocação.