Ministério da
Ciência e Tecnologia

INPE-00000-TDI/0000

IMPLEMENTAÇÃO DE FFT EM HARDWARE RECONFIGURÁVEL

Vitor Conrado Faria Gomes

Relatório Final de Projeto de Iniciação científica (PIBIC/CNPq/INPE)

Registro do documento original:
<http://urlib.net/xxx>

INPE
São José dos Campos
2009

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GB)
Serviço de Informação e Documentação (SID)
Caixa Postal 515 - CEP 12.245-970
São José dos Campos - SP - Brasil
Tel.:(012) 3945-6911/6923
Fax: (012) 3945-6919
E-mail: pubtc@sid.inpe.br

Ministério da
Ciência e Tecnologia

INPE-00000-TDI/0000

IMPLEMENTAÇÃO DE FFT EM HARDWARE RECONFIGURÁVEL

Vitor Conrado Faria Gomes

Relatório Final de Projeto de Iniciação científica (PIBIC/CNPq/INPE)

Registro do documento original:
<http://urlib.net/xxx>

INPE
São José dos Campos
2009

RESUMO

Este trabalho tem como objetivo implementar a Transformada Rápida de Fourier (FFT) em Hardware Reconfigurável (FPGA) com o propósito de aumentar o desempenho de aplicações numéricas. A FFT implementada em VHDL será integrada ao modelo DYNAMO: modelo meteorológico de água rasa unidimensional, representando um sistema de computação híbida. Para o desenvolvimento deste trabalho foram realizadas diversas atividades para subsidiar a implementação desta transforamada. Este relatório apresenta os conceitos envolvidos, o relato das atividades e os resultados obtidos até o momento.

SUMÁRIO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE TABELAS

Pág.

1 INTRODUÇÃO

A solução de aplicações numéricas científicas exige grande demanda de recursos computacionais, sendo que existem diversas estratégias para melhorar o desempenho deste tipo de aplicação. A utilização de dispositivos reconfiguráveis, em especial *Field-Programable Gate Arrays* (FPGAs), para a computação de aplicações é uma tendência atual para a computação de aplicações intensivas (CHAMBERLAIN et al., 2008; GOKHALE; GRAHAM, 2005; LIANG et al., 2003). A utilização de FPGAs em conjunto com aplicações em software define um recente paradigma conhecido como computação híbrida. Nesta abordagem, utiliza-se um ou mais dispositivos de computação para a execução de uma tarefa. Um exemplo de sistema híbrido é a utilização de processadores de propósito geral em conjunto com FPGAs para a execução de aplicações.

FPGAs são dispositivos lógicos programáveis compostos por blocos lógicos configuráveis que, diferentemente de circuitos dedicados, podem ser configurados diversas vezes. Sua utilização visa obter o desempenho de soluções implementadas em hardware e a flexibilidade de soluções baseadas em software (CHAMBERLAIN et al., 2008; GOKHALE; GRAHAM, 2005; SHIRAZI et al., 1995; WAIN; AL., 2004).

A Transformada de Fourier é uma transformada linear usada em diversas aplicações científicas. Esta transformação é usualmente encontrada como núcleo de aplicações que vão desde processamento de imagens até simulações atmosféricas. A Transformada Rápida de Fourier (FFT) é um algoritmo que computa a transformada de Fourier reduzindo sua complexidade de $O(N^2)$ para $O(NLogN)$.

Apesar da redução de complexidade da transformada de Fourier, a FFT continua sendo um algoritmo computacionalmente intensivo e, portanto, alvo de estudos que visam o aumento do desempenho de sua implementação (AGARWAL et al., 1994; HEMMERT; UNDERWOOD, 2005; BAHN et al., 2008; GUPTA; KUMAR, 1993; PALMER, 2005; VITE-FRIAS et al., 2005; HE; GUO, 2008; Dillon Engineering, Inc., ; FRIGO; JOHN-

SON, ; Xilinx Inc., ; ALI et al., 2007; CHAO et al., 2005; GONZALEZ-CONCEJERO et al., 2008). Seguindo esta tendência, este trabalho visa a implementação da Transformada Rápida de Fourier em Hardware Reconfigurável para o aumento do desempenho da computação de um sistema evolutivo implementado em software. Esta integração configura um sistema de computação híbrida.

Este relatório continua detalhando os objetivos e a fundamentação deste trabalho (capítulos 2 e 3), incluindo informações relevantes sobre dispositivos reconfiguráveis e sobre o sistema computacional híbrido utilizado. Nos capítulos seguintes, 4 e 5, apresenta-se respectivamente a metodologia e as atividades desenvolvidas. Finalizando, nos capítulos 6 e 7 são apresentados os resultados obtidos até o momento, assim como as conclusões.

2 OBJETIVOS

Este trabalho tem como objetivo implementar a Transformada Rápida de Fourier em Hardware Reconfigurável (FPGA) e integrar em um sistema evolutivo simples. Esta implementação visa obter ganho de desempenho na execução desta operação em sistemas híbridos.

Como objetivos específicos, tem-se:

- Implementar rotinas de FFT em software;
- Desenvolver a FFT em linguagem VHDL e realizar simulações;
- Implementar a rotina FFT em dispositivo de FPGA;
- Aplicar a implementação em sistemas evolutivos.

3 CONCEITOS

Este capítulo apresenta os conceitos utilizados para a realização deste trabalho. Inicialmente é apresentada a Transformada Rápida de Fourier e os algoritmos para sua computação sequencial e paralela. Na sequência apresenta-se os dispositivos reconfiguráveis, em especial FPGAs e sobre o fluxo de desenvolvimento para este tipo de recurso. Por fim, são apresentados sistemas híbridos de alto desempenho e

o sistema Cray XD1 que é utilizado neste trabalho. São apresentados a arquitetura, os modos de comunicação e a hierarquia de memória deste sistema.

3.1 Transformada Rápida de Fourier

A Transformada de Fourier é uma transformada linear usada em diversas aplicações científicas. Em sua formulação discreta, esta transformada é usualmente núcleo computacional de aplicações como processamento de sinais e solução de equações parciais. A Transformada Discreta de Fourier (DFT) de uma sequência de N números pode ser computada como:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \qquad k = 0, 1, ..., N-1 \tag{3.1}$$

onde $W_N = e^{-2\pi\sqrt{-1}/N}$ é um coeficiente trigonométrico conhecido como fator de giro. O algoritmo da Transformada Rápida de Fourier (FFT) (COOLEY; TUKEY, 1965) computa a DFT reduzindo sua complexidade de $O(N^2)$ para $O(NlogN)$. Existem várias formas de estruturar o algoritmo da FFT. Uma variante é o radix-2, o qual utiliza a abordagem dividir para conquistar, que opera sobre um vetor de N elementos, onde N é potência de 2. Esta operação básica é conhecida como "borboleta" e consiste de duas somas e uma multiplicação complexa. O algoritmo de radix-2 realiza cada operação sobre dois pontos, fornecendo a menor unidade computacional possível para a FFT, permitindo uma maior flexibilidade, em especial quanto a avaliação de espaço em dispositivos reconfiguráveis. A figura 3.1 ilusta o fluxo da computação da FFT para 8 pontos. A operação borboleta é representada por cada quadrado branco desta figura. Nesta ilustração é possível observar a ordem em que os pontos são combinados para o processamento de cada operação básica.

3.1.1 FFT paralela

Devido ao uso intensivo de processamento para a computação da Transformada Rápida de Fourier, diversas estratégias foram determinadas para permitir a aceleração da computação desta operação. Alguns esforços visam a implementação desta operação em hardware (HEMMERT; UNDERWOOD, 2005; VITE-FRIAS et al., 2005; HE; GUO, 2008; Dillon Engineering, Inc., ; 4DSP Inc., ; Xilinx Inc., ; CHAO et al., 2005; GONZALEZ–CONCEJERO et al., 2008), enquanto outros buscam o aumento de desempenho através de estratégias de paralelização em software (AGARWAL et al., 1994; BAHN et al., 2008;

GUPTA; KUMAR, 1993; PALMER, 2005; FRIGO; JOHNSON, ; ALI et al., 2007).

A paralelização da FFT pode ser realizada através do algoritmo *binary-exchange* que minimiza as comunicações entre os processos (GUPTA; KUMAR, 1993). Esta estrutura fornece uma complexidade ideal de $O(LogN)$ quando computada com N processos, isto pode ser observado na figura 3.1.

Cada passo da FFT opera em pontos que aumentam a distância de seus índices. No último passo, com o algoritimo radix-2, as borboletas operam nos pontos i e $(N/2)+i$, onde i varia de 0 até $(N/2) - 1$. No caso de particionar a FFT em duas tarefas paralelas (cinza escura e cinza claro na Fig. 3.1), o último passo da FFT não pode ser obtido sem que ocorra troca de dados entre os processos. Esta dependência de dados é encontrada mais cedo com o aumento da quantidade de processos paralelos que computam a FFT. Quando o custo de comunicação é alto, pode ser vantajoso atribuir a computação de (N/P) pontos para cada tarefa e calcular o restante da computação sequencialmente em um único processo.

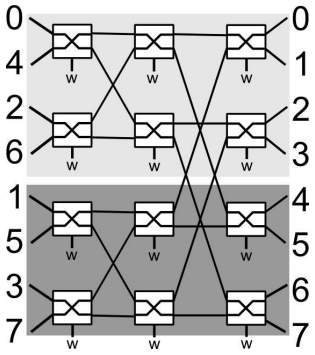

Figura 3.1 - Fluxo computacional para FFT de 8 pontos

3.2 Dispositivos reconfiguráveis

Dispositivos reconfiguráveis são recursos que podem ser programados para ter o comportamento de um circuito lógico em hardware. Diferentemente de circuitos integrados de aplicação específica (ASIC), podem ser reconfigurados diversas vezes para terem diferentes comportamentos lógicos.

Fazem parte desta categoria, dispositivos como *Programmable Array Logic* (PAL), *Generic Array Logic* (GAL), *Complex Programmable Logic Device* (CPLD) e *Field-*

Programmable Gate Array (FPGA). Destes dispositivos, o FPGA é o dispositivo que possui maior flexibilidade devido a abundância e o tamanho reduzido de suas unidades básicas, o que permite a configuração de sistemas complexos.

3.2.1 Field-Programmable Gate Array

Field-Programmable Gate Arrays (FPGAs) são dispositivos lógicos programáveis capazes de serem configurados para reproduzir o comportamento de um hardware. Estes dispositivos são formados por blocos lógicos programáveis que são conectados por interligações programáveis. Estes dois recursos permitem a criação de circuitos lógicos em FPGA, sendo limitados somente pela área e a memória disponíveis. O uso de FPGAs visa obter o desempenho de aplicações em dispositivos dedicados (ASIC) com a flexibilidade de aplicações em software. Sua flexibilidade é dada pela facilidade de configuração através de uma descrição de hardware escrita em VHDL ou Verilog. Essas linguagens permitem a descrição do comportamento de um circuito lógico e facilita a criação de novas aplicações em hardware devido ao nível de abstração que fornece ao programador.

3.2.2 Fluxo de desenvolvimento

Assim como o desenvolvimento de uma aplicação em software, a descrição de um hardware em linguagem VHDL ou Verilog possui uma sequência de desenvolvimento. Na figura 3.2 é possível observar as etapas envolvidas na implementação de uma aplicação para ser configurada em um FPGA. Inicialmente é descrito o sistema utilizando uma linguagem de descrição de hardware (no caso deste trabalho utilizou-se VHDL). Esta descrição determina o comportamento do sistema em relação aos sinais de entrada no módulos do dispositivo e determina os sinais de saída. Na sequência é utilizado um sintetizador, que transforma o código em alto nível em um esquema de elementos lógicos que representam a lógica descrita. O terceiro passo, é transformar estes circuitos lógicos em um sistema que se adapte a estrutura lógica já existente no FPGA, a qual definirá a configuração dos blocos lógicos e das conexões entre eles. O passo final é a geração do arquivo que contém as informações que devem ser passadas ao FPGA para que este dispositivo possa ser configurado. Este fluxo de desenvolvimento pode ser realizado através de ferramentas desenvolvidas pelos fabricantes de FPGAs como a Xilinx. Neste trabalho é utilizado o Xilinx Ise Foudation 10.1 para a realização das três últimas etapas deste fluxo de desenvolvimento.

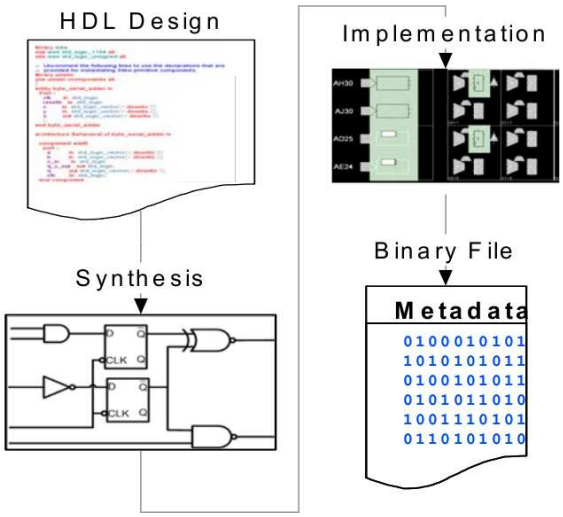

Figura 3.2 - Fluxo de desenvolvimento de aplicação para FPGA. Fonte: Adaptada de Cray Inc. (2005b)

3.3 Sistemas híbridos de alto desempenho

Nos últimos anos, fabricantes de sistemas de alto desempenho, como Cray, SGI e SRC, introduziram sistemas de computação híbrida como Cray XD1, XT3, XT4, XT5h, SGI RASC e SRC-6 MAP. Estes sistemas tem sido explorados em trabalhos de computação híbrida (BONDHUGULA et al., 2006; KINDRATENKO et al., 2007; KINDRATENKO; POINTER, 2006; ZHUO; PRASANNA, 2005; ZHUO; PRASANNA, 2008). Neste trabalho é utilizado o sistema XD1 da Cray, sendo assim, as próximas subseções explicaram as características desta arquitetura híbrida.

3.4 Cray XD1

O Cray XD1 é um sistema híbrido lançado em outubro de 2004. Trouxe no momento de seu lançamento algumas inovações tecnológicas. Entre elas está a rede de interconexão de alto desempenho RapidArray, otimizações no sistema Linux e a inclusão de FPGAs em seu chassis. Na figura 3.3 é possível ver um *rack* com diversos equipamentos XD1 e um equipamento em destaque.

3.4.1 Arquitetura

O Cray XD1 utilizado neste trabalho é composto por seis nós (*blades*), cada um contendo dois processadores de propósito geral AMD Opterom de 2.4GHz e um FPGA Xilinx Virtex II Pro. A arquitetura de um *blade* do Cray XD1 pode ser vista

Figura 3.3 - Cray XD1. Fonte: Cray Inc. (2005a)

na figura 3.4. É possível observar que o dispositivo reconfigurável tem acesso direto a quatro bancos de memória QDR II SRAM, que possuem 4MB cada. O RapidArray Processor permite que os processadores enviem dados para o FGPGA e que o FPGA leia dados da DRAM. No desenvolvimento de aplicações híbridas para este sistema, exitem duas questões chaves que devem ser observadas: a transferência de dados entre os dispositivos e o uso eficiente dos diferentes níveis de memória disponíveis no sistema.

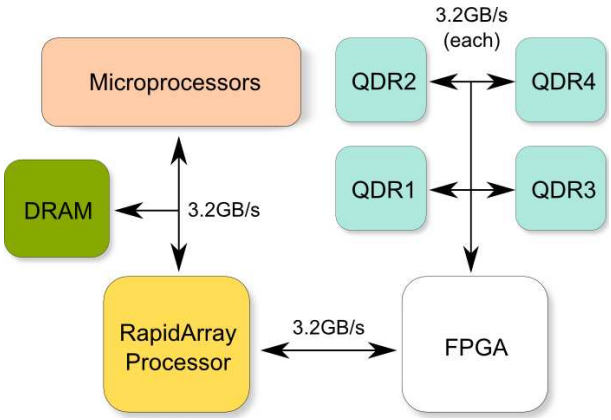

Figura 3.4 - Arquitetura do Blade do Cray XD1

3.4.2 Modos de comunicação

A Cray disponibiliza a API RapidArray Transport Core para a comunicação dos processadores com o FPGA. Esta API é composta por dois blocos denominados Fabric Request e User Request (Cray Inc., 2005e).

O bloco Fabric Request, que realiza a comunicação usando uma abordagem *push*, permite que o programa executado nos processadores envie dados para o FPGA. Esta abordagem mantém os processadores ocupados durante a transferência de dados. Para permitir esta comunicação, a Cray disponibiliza a biblioteca *enlib*, a qual abstrai ao programa o FPGA como um arquivo. A transferência de dados entre os processadores e o FPGA é realizada através de leituras e gravações pelo programa em C neste arquivo. Com a realização de uma leitura ou escrita, o FPGA recebe, através do RapidArray Transport Core, uma requisição que deve ser tratada pela aplicação do FPGA. Em caso de leitura, deverá ser retornado um valor ao RapidArray Transport Core para que ocorra o retorno da função chamada pelo programa em linguagem de alto nível. Somente é permitida a manipulação de um *quadword* (64 bits) por requisição utilizando o bloco Fabric Request (Cray Inc., 2005e).

A abordagem que utiliza o bloco User Request é conhecida como *pull* e, diferentemente da Fabric Request, mantém os processadores livres durante a transferência de dados entre o programa em C e o FPGA. Para isso, este bloco permite que o FPGA realize leituras e escritas em um espaço de memória compartilhado do programa. O endereço para a região de memória, que é compartilhada utilizando a *einlib*, é enviado ao FPGA através do bloco Fabric Request. Estando disponível o endereço, a aplicação descrita para o FPGA é capaz de fazer até 32 requisições sequenciais ao RapidArray Transport Core. Cada requisição pode solicitar até 8 posições contíguas da memória do programa. Os retornos das solicitações ao RapidArray Transporte Core não são necessariamente na ordem em que foram realizadas. O *core* garante somente a ordem das 8 posições contíguas de cada requisição (Cray Inc., 2005e). O sistema descrito para o FPGA deve ordenar os dados através do auxílio de *tags* disponibilizadas durante a requisição e o retorno. Outra solução é aguardar o retorno de uma solicitação antes de realizar outra.

3.4.3 Hierarquia de memória

A arquitetura híbrida do XD1 permite ao FPGA acesso a diferentes níveis de memória. A DRAM é o mais alto nível, com a maior quantidade de memória disponível e com latência de leitura não constante. Esta memória pode ser acessada usando o Bloco User Request interface. Abaixo deste nível estão os bancos de memória QDR II SRAM com 4MB cada. A latência de acesso a esses bancos é de 8 ciclos para a leitura. Além desses, a família de FPGAs Vitex Pro possuem bancos de memória internos que podem ser acessados em um único ciclo. A correta utilização destes

níveis de memória podem favorecer a obtenção de desempenho no desenvolvimento de aplicações para esta arquitetura.

4 METODOLOGIA

Para o desenvolvimento deste trabalho foi determinada uma metodologia a ser emprega na realização das atividades. Este metodologia visa o desenvolvimento incremental do trabalho, determinando atividades intermediárias que compõem este trabalho. Seguindo esta abordagem este projeto segue a metodologia:

- A FFT será implementada em software numa linguagem de alto nível;
- Testes preliminares com programação VHDL;
- Implementar FFT com teste de simulação em VHDL;
- Estudo de estratégias para emular operação de ponto flutuante em processador de ponto fixo;
- Avaliação de implementação de funções não lineares em FPGA;]
- Implementação da FFT em FPGA;
- Aplicação de técnicas de FFT em sistemas evolutivos.

5 ATIVIDADES DESENVOLVIDAS

Neste capítulo são apresentadas em ordem cronológia as atividades desenvolvidas neste trabalho.

5.1 Testes e avaliações iniciais

Para subsidiar a implementação da FFT em Hardware Reconfigurável foi necessário realizar algumas atividades prévias. Inicialmente, foi realizado um estudo teórico sobre a Transformada Rápida de Fourier e sua utilização em aplicações numéricas. Na sequência, implementou-se esta transformada em linguagem C, com o objetivo de familiarizar-se com esta operação.

A tarefa seguinte foi a realização de estudos sobre o ambiente computacional alvo das implementações deste projeto, o supercomputador Cray XD1 disponível no Laboratório Associado de Computação e Matemática Aplicada do INPE. Em paralelo, foram realizados estudos sobre a linguagem de descrição de *hardware* a ser utilizada na implementação da transformada: VHDL. Esta linguagem permite a descrição do comportamento lógico do *hardware*, entretanto, um conjunto de operações desta linguagem ainda não pode implementado para ser utilizado em FPGAs, somente podem ser simuladas em software. Sendo assim, foram estudadas técnicas de implementação usando VHDL de descrições sintetizáveis. Passadas as etapas iniciais do projeto e seguindo a metodologia deste trabalho, desenvolveu atividades intermediárias a fim de auxiliar a implementação da Transformada Rápida de Fourier no Cray XD1. Esta atividades, que tivem ciclos de desenvolvimento maiores, são descritas nas próximas subseções.

5.1.1 Comunicação

Conforme já discutido no ítem 3.4.2 o modo de comunicação entre os processadores de propósito geral e o FPGA é uma questão chave no desenvolvimento de aplicações para uma arquitetura híbrida. Além disso, é essencial que se conheça os métodos e modos de comunicação disponíveis no sistema utilizado. Neste sentido, durante esta etapa do trabalho, estudou-se os dois diferentes modos de comunicação existentes no Cray XD1 e desenvolveu-se aplicações de teste para ambos os modos. Para o desenvolvimento dessas aplicações foram utilizados os manuais (Cray Inc., 2005e; Cray Inc., 2005b; Cray Inc., 2005c) e exemplos (Cray Inc., 2005d) disponibilizados pela Cray. Com as duas implementações de comunicação em VHDL desenvolvidas, foram realizados testes de desempenho de ambos os modos de comunicação a fim de se obter as taxas de transferência de dados.

5.1.2 Ponto flutuante

Muitas aplicações candidatas às soluções de computação híbrida são implementadas inicialmente em software, usando Matlab, Fortran ou linguagem C, e utilizam variáveis e operações em ponto flutuante (Ligon III; AL., 1998; BELANOVIC; LEESER, 2002; GOKHALE; GRAHAM, 2005). É conveniente manter a mesma representação na migração de tais aplicações para FPGAs, pois frequentemente necessita-se representar números muito pequenos ou muito grandes, ou não se conhece a faixa de operação dos dados (UNDERWOOD, 2004). Entretanto, FPGAs não possuem suporte nativo

para operações em ponto flutuante, as quais necessitam ser implementadas assim como o restante da lógica da aplicação (WAIN; AL., 2004).

Visando suprir esta necessidade, companhias e grupos de pesquisa desenvolvem bibliotecas que implementam a representação e operações em ponto flutuante, geralmente baseadas no padrão IEEE 754. Estas bibliotecas, no entanto, possuem características e modos de funcionamento diferentes. Sendo assim, surge a necessidade de avaliar as diferentes implementações de bibliotecas de ponto flutuante disponíveis para determinar o desempenho destas implementações quanto a precisão das operações realizadas em FPGA em relação a implementação executada em CPU. Neste contexto, investigou-se implementações de bibliotecas de ponto flutuante sintetizáveis e foi criado um implementação teste com a biblioteca Vfloat (LEESER; AL.,). Para o teste foi implementado em VHDL uma aplicação que calcula o produto interno de dois vetores com elementos em ponto flutuante. Além disso, foi implementado uma aplicação em linguagem C para servir como referência para a comparação.

5.1.3 Estimativa de utilização de área no FPGA

Definida a biblioteca a ser utilizada e conhecidas as taxas de transferência do Cray XD1, iniciou-se a implementação de módulos para o desenvolvimento da FFT. Como já discutido no ítem 3.1, o núcleo básico da FFT é a operação "borboleta". Esta operação realiza uma multiplicação e duas somas complexas. Devido a área limitada para a acomodação da lógica do sistema, é necessário estabelecer a área ocupada por cada núcleo básico da FFT para que seja definida a quantidade de operações que poderão ser realizadas em paralelo dentro do FPGA. Seguindo este objetivo, estudos foram realizados para o estabelecimento da área ocupada por cada borboleta. Implementou-se duas versões da operação borboleta que diferem entre si pela reutilização de componentes básicos de operações em ponto flutuante. A primeira borboleta utiliza o mínimo necessário de componentes, possuindo uma maior latência e uma menor área. A segunda utiliza o máximo possível de componentes, possuindo uma menor latência e maior área. Ambas as borboletas utilizam representação em ponto flutuante com 64 bits. Além das áreas ocupadas pelas duas diferentes implementações de borboletas, este teste obteve também a área ocupado pelo projeto mínino a ser sintetizado para o FPGA e pelo projeto com um módulo de comunicação.

5.2 FFT em VHDL integrada com software

Após os estudos e avaliações realizados, foi possível realizar uma primeira implementação da FFT em VHDL para ser integrada com software. Esta primeira implementação visa obter o perfil de uma execução da FFT em hardware reconfigurável integrada com uma aplicação em software. Para isso, determinou-se que a aplicação executaria a computação da FFT (e sua inversa IFFT) para um vetor de 32 pontos em ponto flutuante representado com 32 bits. As subseções seguintes apresentam mais detalhes sobre esta implementação.

5.2.1 Implementação em VHDL

Foi implementada a FFT e a IFFT para 32 pontos em VHDL utilizando 3 núcleos básicos de borboleta. Estes núcleos operam em paralelo para a execução das operações que compõem esta transformada. Para a implementação foi utilizada uma máquina de estados que que controla as entradas e saídas dos módulos de borboleta. Esta implementação utiliza o bloco de comunicação Fabric Request do RapidArray Transporte Core, pois para poucos dados este bloco é o mais indicado, como já mostrado no ítem 5.1.1.

5.2.2 Biblioteca FFT Híbrida

Com a implementação da FFT e da IFFT em VHDL realizada, foi criada uma biblioteca em linguagem C para manipular a comunicação com o FPGA e para tornar transparente para a aplicação como é calculada a trasformada. Esta linguagem foi escolhida pois é a mesma da biblioteca disponibilizada pela Cray para a manipulação do FPGA pelos processadores. Alguns testes com esta biblioteca foram realizados com o objetivo de validar a implementação. Estes testes comparam os resultados da implementação para FPGA com a implementação para CPU.

5.2.3 Dynamo

A fim de integrar a implementação da FFT em FPGA em um sistema evolutivo mais próximo às aplicações do INPE, o modelo meteorológico Dynamo foi escolhido. Este é um modelo meteorológico de água rasa unidimensional implementado em Fortran. Este modelo constitui um sistema evolutivo que emprega condições de contorno periódicas para a simulação. Sistemas como este possuem características específicas que podem ser exploradas utilizando transformadas de Fourier. Para a adaptação

do modelo meteorológico, surgem duas questões a serem tratadas: a integração da biblioteca de FFT escrita em linguagem C com o modelo em Fortran e a adaptação de rotinas do modelo para a utilização da transformada de Fourier. Estas questões são tratadas nas duas subseções seguintes.

5.2.4 Integração de Fortran com C

Com o intuito de integar as transformadas implementadas em C como código Fortran do modelo Dynamo, estudou-se métodos para a invocação de funções em linguagem C por programas escritos em Fortran. Esta tarefa pode ser realizada através da compilação dos programas utilizando copiladores Intel e GNU, que permitem esta integração. Para isto basta a utilização de nomes especias nas funções implementadas e a utilização de um padrão único como parâmetro para os compiladores. Testes com chamadas de funções em C por programas em Fortran foram executados para confirmar o funcionamento desta técnica.

5.2.5 Adaptação do modelo

Para a integração das funções que calculam a FFT em FPGA com o modelo Dynamo, foi necessário identificar a subrotina do sistema que utilizava as condições de contorno a serem adaptadas. Nesta etapa foi, então, estudado o funcionamendo do modelo Dynamo através do seu código fonte. Além disso, diversos testes foram executados para verificar o perfeito funcionamento do sistema com diferentes argumentos de entrada.

A adaptação do modelo foi possível, pois em sistemas evolutivos onde são empregadas condições de contorno periódicas, as matrizes associadas a estes sistemas são matrizes circulantes. A decomposição de tais matrizes é dada por matrizes de Fourier. Esta decomposição permite a redução da complexidade da computação, permitindo um ganho de desempenho (VELHO; CLAEYSSEN, 1992). Matrizes circulantes são matrizes que apresentam a seguinte aparência

$$C = circ(c_1, c_2, ..., c_n) = \begin{bmatrix} c_1 & c_2 & \cdots & c_n \\ c_n & c_1 & \cdots & c_{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ c_2 & c_3 & \cdots & c_1 \end{bmatrix} \tag{5.1}$$

e pode ser decompostas por

$$C = F_n^* \Lambda^+ F_n; \tag{5.2}$$

onde F_n^* e F_n são matrizes de Fourier e Λ é a matriz diagonal

$$\Lambda = \sum_{k=0}^{N-1} c_{k+1}(\Omega_n)^n; \tag{5.3}$$

$$\Omega = diag[1, \omega, \omega^2, \cdots, \omega^{n-1}]; \tag{5.4}$$

Então, usando a fórmula 5.2 podemos calcular o produto de uma matriz circulante por um vetor (Cx) usando duas transformadas de Fourier e uma multiplicação de uma matriz diagonal (VELHO; CLAEYSSEN, 1992). Esta operação foi adaptada na subrotina Pois1d do modelo Dynamo que realiza a computação da equação de Poisson em um vetor de entrada. Para a validação da adaptação antes da implementação em linguagem C foram criados *scripts* para sciLab.

Validadas as alterações a serem realizadas na subrotina Pois1d, foi implementada uma função em linguagem C para a integração da implementação da FFT que utiliza FPGA com o modelo meteorológico Dynamo escrito em Fortran. Testes que avaliam o desempenho desta nova configuração foram executados para o levantamento do perfil deste tipo de execução no sistema híbrido.

5.3 FFT paralela em software

Considerando que o sistema utilizado nos testes possue 6 *blades*, totalizando 12 processadores e 6 FPGAs, a computação da FFT em paralelo pode auxiliar o ganho de desempenho juntamente com a utilização da computação da FFT em FPGA. Além disso, a implementação da FFT em VHDL até o momento considera sua execução totalmente em FPGA, sendo que a CPU permanece desocupada enquanto são realizados os cálculos no dispositivo reconfigurável. Esta abordagem pode significar desperdício de recursos computacionais. Neste sentido, estudou-se métodos para a paralelização da Transformada Rápida de Fourier para permitir uma futura utilização de todos os recursos disponíveis no Cray XD1. Para esta implementação, a

especificação de troca de mensagens *Message Passing Interface* (MPI) foi estudada, assim como algoritmos de paralelização da FFT. Foi então desenvolvida uma coleção de funções que permitem o cálculo da FFT em ambiente paralelo. Estas funções possibilitam o particionamento, o cálculo e a combinação dos dados calculados por cada processo. Testes foram executados em um aglomerado de *desktops* disponível no Laboratório de Sistemas de Computação da Universidade Federal de Santa Maria. Dados de desempenho foram coletados com o objetivo de estabelecer o perfil da granularidade deste tipo de computação paralela.

5.4 FFT intensivamente paralela para sistemas híbridos

Conhecidas as taxas de comunicação, a hierarquia de memória do sistema, o perfil de execução da FFT em hardware e em software e os algoritmos de paralelização da FFT, iniciou-se a descrição de uma FFT que utilizasse intensivamente os recursos disponíveis no Cray XD1. Esta implementação utiliza a CPU e o FPGA colaborando para a execução da transformada. Além disso, utiliza técnicas para aumentar o paralelismo de operações e para esconder os custos da movimentação de dados entre os dispositivos e entre os módulos do FPGA. Visando a análise da computação da FFT quanto a sua granularidade, esta implementação foi desenvolvida de maneira flexível, permitindo a computação de vetores de até 2^{18} pontos. A arquitetura desta implementação pode ser vista na figura 5.1.

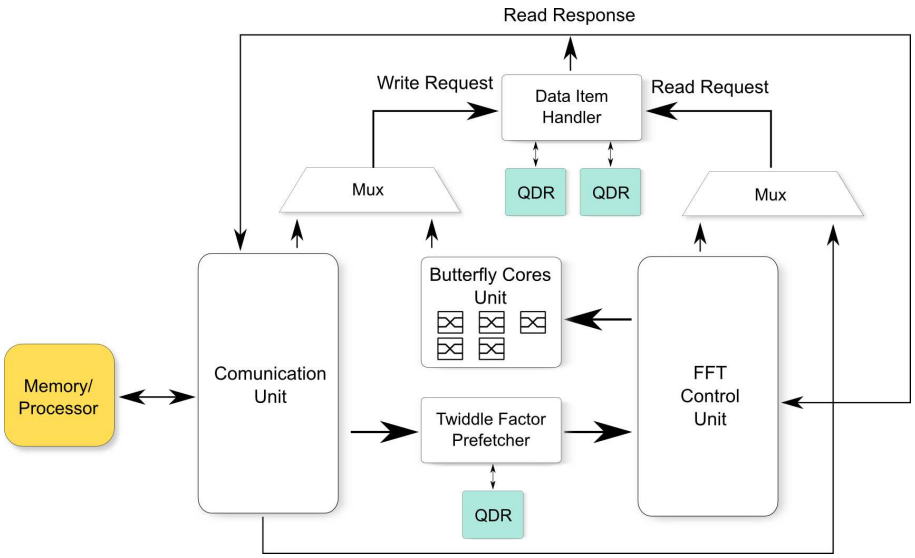

Figura 5.1 - Arquitetura da FFT Híbrida Flexível

5.4.1 Paralelização CPU-FPGA

O primeiro objetivo desta implementação é aproveitar o poder computacional da CPU enquanto o FPGA realiza a computação da FFT. Neste sentido divide-se a computação desta transformada em proporções que dependem do tempo de cálculo em cada um dos dispositivos e então são realizadas as computações. Ao fim da execução de cada parte da FFT, os resultados parciais vão para a memória do processador de propósito geral para que possa ser finalizada a computação. A movimentação intermediária de dados é evitada devido ao alto custo de comunicação.

5.4.2 Paralelização intra-FPGA

Outra frente tomada nesta implementação foi a paralelização de computações dentro do FPGA. Para isso, foram utilizados 5 núcleos borboleta para a computação de até 10 pontos simultaneamente. Além disso, a estruturação desta arquitetura, permite que ocorra computação durante a transferência de dados para o FPGA, reduzindo assim os custos relacionados à transferência dos dados. Outra técnica utilizada para o aumento do desempenho da computação desta transformada em FPGA é quanto ao acesso dos dados dos blocos de memória QDR. Foram implementadas duas entidades que tratam de reduzir a latência de leitura da QDR RAM. A primeira entidade realiza uma pré-busca dos fatores de giro, tornando disponível os dados antes que sejam solicitados. Isto é possível pois os dados estão armazenados em ordem de acesso na memória. A outra entidade, utiliza dois blocos de QDR para permitir o acesso paralelo a duas posições diferentes de memória. Este técnica aumenta a vazão de dados e evita que as borboletas fiquem ociosas sem dados para serem computados. Além dessas abordagens, a unidade de controle da FFT continua a busca e o cálculo de novos endereços enquanto a unidade de borboletas realiza suas operações. Com a modularização, que pode ser vista na figura 5.1, é possível especializar entidades no hardware para a execução de tarefas e ainda realizá-las facilmente em paralelo, sendo necessário somente alguns sinais de sincronismo.

6 RESULTADOS

6.1 Modos de comunicação

Com os testes realizados para a obtenção da taxa de transferência de dados entre CPU e FPGA foi escrito um artigo intitulado **"Avaliação de abordagens de**

comunicação com FPGA no supercomputador Cray XD1" que foi aceito na 9a. Escola Regional de Alto Desempenho - ERAD 2009. Os resultados deste trabalho podem ser visto na tabela 6.1. Nesta tabela, a primeira coluna apresenta a quantidade de dados transferida entre os dispositivos. Na duas colunas seguintes são mostrados os tempos de cada transferência para cada modo de comunicação. São mostrados também, nas colunas 4 e 5, o desvio padrão para cada um dos modos. Por fim, temos a taxa de transferência de dados para cada bloco do RapidArray Transporte Core. Com essas informações, podemos concluir que para a transferência de até 16KB o bloco Fabric Request (FR), é mais rápido que o User Request (UR). Para valores maiores, o bloco User Request é o mais indicada.

O artigo aceito no evento, bem como os testes implementados podem ser encontrados em `svn://paple.inf.ufsm.br/inpe-fpga/erad2009`.

Tabela 6.1 - Tempos de comunicação

	Tempo (μs)		σ		Taxa (MB/s)	
Dados	FR	UR	FR	UR	FR	UR
8B	0,9	1104,8	0,32	8,28	8,5	$< 0,1$
16B	1,6	1119,7	0,97	6,78	9,5	$< 0,1$
128B	11,7	1122,1	1,89	8,02	10,4	0,1
1KB	94,3	1130,2	12,96	5,47	10,4	0,9
16KB	1440,3	1284,6	79,53	9,19	10,9	12,2
128KB	10986,1	2421,0	959,73	17,62	11,4	51,6
1MB	83017,2	11726,5	2896,20	70,75	12,0	85,3
2MB	166725,0	22844,4	4940,44	71,91	12,0	87,6

6.2 Bibliotecas de ponto flutuante

Para os testes com bibliotecas de ponto flutuante foi escrito um artigo intitulado **"Avaliação de uma Biblioteca de Ponto Flutuante para FPGA no Supercomputador Cray XD1"** que foi aceito no 9o. Workshop em Sistemas Computacionais de Alto Desempenho (WSCAD-CTIC 2008). Na tabela 6.2 podem ser vistos os resultados obtidos com os testes com a biblioteca de ponto flutuante em VHDL Vfloat. Para testar uma maior faixa de aplicações que podem utilizar a biblioteca em análise, determinou-se 3 experimentos que diferem quanto a geração do vetor de entrada. Além disso, foi realizado testes com vetores de diferentes tamanhos, variando de 1 até 100.000.000 elementos. É possível observar que para até 10 pontos erro

relativo é nulo entre as implementações em software e em VHDL. Para os demais experimentos existe erro relativo e são menores que 10^{-4}.

Tabela 6.2 - Erro relativo no teste dos 3 experimentos

	Erro relativo δ		
Elementos	Experimento 1	Experimento 2	Experimento 3
1	0	0	0
10	0	0	0
100	$2,03 * 10^{-16}$	0	$1,87 * 10^{-16}$
1.000	$2,08 * 10^{-16}$	0	$1,68 * 10^{-16}$
10.000	$3,11 * 10^{-06}$	$3,29 * 10^{-06}$	$1,38 * 10^{-16}$
100.000	$4,87 * 10^{-06}$	$1,64 * 10^{-06}$	$7,95 * 10^{-06}$
1.000.000	$5,81 * 10^{-06}$	$1,53 * 10^{-05}$	$6,44 * 10^{-06}$
10.000.000	$6,84 * 10^{-06}$	$1,64 * 10^{-05}$	$6,14 * 10^{-06}$
100.000.000	$6,88 * 10^{-06}$	$1,64 * 10^{-05}$	$6,08 * 10^{-06}$

A fim de encontrar outra biblioteca de ponto flutuante que tivesse melhor diferença de precisão em relação a implementação em software, testou-se a biblioteca de ponto flutuante disponibiliada pela Xilinx. Esta biblioteca encontra-se integrada ao ambiente de desenvolvimento Ise Foudation e está disponível somente para dispositivos da Xilinx. Realizando o mesmo teste, trocando somente a biblioteca Vfloat pela a da Xilix, encontrou-se erro relativo nulo para todos os testes realizados. Estes dados auxiliaram na escolha da biblioteca da Xilinx como a biblioteca a ser utilizada na implementação da Transformada Rápida de Fourier em VHDL.

O artigo e os testes realizados para o teste das bibliotecas podem ser encontrados em `svn://paple.inf.ufsm.br/inpe-fpga/wscad2008`.

6.3 Estimativa de utilização de área no FPGA

A estimativa de utilização de área de diferentes implementações de borboletas e dos projetos mínimos e com o módulo de comunicação podem ser vistos na tabela 6.3.

Com os resultados obtidos pode-se verificar que a área ocupada por uma borboleta é relativamente grande, visto que ainda é necessário incluir a lógica para o cálculo da FFT no FPGA. Sendo assim, decidiu-se que seriam utilizados representação de 32 bits para ponto flutuante, visto que esta representação é suficiente para a aplicação em software onde será integrada a FFT.

Tabela 6.3 - Estimativa de Área

Módulo	Área %
Template Básico	11
Comunicação	2
Borboleta min	29
Borboleta máx	44

Um resumo com os resultados desta etapa foi apresentado na 23a. Jornada Acadêmica Integrada da Universidade Federal de Santa Maria. Este resumo foi selecionado por esta instituição para ser apresentado na 61a. Reunião Anual da Sociedade Brasileira para o Progresso da Ciência. Os resumos e os testes podem ser encontrados em `svn://paple.inf.ufsm.br/inpe-fpga/jai2008`.

6.4 FFT em VHDL integrada com software

A implementação da FFT em VHDL integrada com software nos forneceu um perfil desta execução híbrida no XD1. Os dados de tempo da execução do Dynamo original e adaptado com a FFT em VHDL podem ser vistos na tabela 6.4. Além de coletar os tempos de execução híbrida e em software, foram coletados os tempos de execução das funções adaptadas do Dynamo e os tempos relacionados a manipulação do FPGA. Verifica-se que o tempo da execução híbrida é maior que o tempo somente em software. Verificou-se também, que o tempo de envio e leitura dos dados do FPGA é maior que o tempo de processamento destes dados. Esta verificação nos indica a necessidade de implementar uma descrição para hardware que opera para uma quantidade maior de pontos para obter um melhor perfil desta transformada quando computada em FPGA, além de estudar estratégias que reduzam o custo de comunicação entre a CPU e o FPGA.

Os dados desta implementação e os códigos podem sem encontrados em `svn://paple.inf.ufsm.br/inpe-fpga/dynamo-hibrido`.

6.5 FFT paralela em software

A implementação da FFT paralela em software usando o algoritmo conhecido como *exchange-binary* resultou em uma biblioteca que possui funções para a manipulação deste tipo de operação em ambiente paralelo. Esta implementação utiliza o padrão MPI e foi testada utilizando a implementação MPICH. A biblioteca pode ser encon-

Tabela 6.4 - Dados da FFT em VHDL integrada com software

Operação	Tempo μ Software	Híbrido
Dynamo	557979	2570856
Pois1d	17,08	210,42
FFT	7,85	105,40
Abertura do FPGA	-	43
Configuração do FPGA	-	1672241
Envio FPGA	-	0,72
Leitura FPGA	-	1,78

trada no repositório svn://paple.inf.ufsm.br/inpe-fpga/pfft.

6.6 FFT intensivamente paralela para sistemas híbridos

A arquitetura que implementa a FFT visando a utilização combinada de CPU e FPGA, bem como a utilização de diversos níveis de paralelismo no FPGA resultou em um artigo enviado para o *21st International Symposium on Computer Architecture and High Performance Computing* entitulado **"A Parallel FPGA-based FFT Architecture for High-Performance Hybrid Computing"**. Este artigo ainda está sobre avaliação e pode ser encontrado, juntamente com a sua implementação, em svn://paple.inf.ufsm.br/inpe-fpga/sbac2009. Como resultado desta implementação tivemos a redução do tempo de computação da FFT quando utilizamos o FPGA combinado com a CPU em relação a computação somente em CPU. Na tabela 6.5 é possível observar os tempos e os *speedups* obtidos com esta implementação.

Tabela 6.5 - Hybrid versus CPU-only results

$log_2 N$	Input size	Time (ms) Hybrid	CPU	Speedup
10	1,024	0.66	0.19	0.29
12	4,096	1.72	0.79	0.46
14	16,384	5.66	4.33	0.77
16	65,536	23.29	20.59	0.88
17	131,072	49.77	47.59	0.96
18	262,144	150.63	191.25	1.27
19	524,288	310.48	461.59	1.49
20	1,048,576	813.57	979.56	1.20

7 CONCLUSÃO

Com as atividades já realizadas pôde-se concluir que implementações em sistemas híbridos necessitam de uma abordagem diferenciada em relação aos sistemas de computação tradicionais. Além disso, restrições dos dispositivos tornam necessário a avaliação de cada decisão do trabalho. Concluimos que é possível a melhora de desempenho do cálculo da Transformada Rápida de Fourier quando computada em conjunto com o processador de propósito geral.

REFERÊNCIAS BIBLIOGRÁFICAS

4DSP Inc. EEE-754 Floating Point FFT/IFFT IP core. Http://www.4dsp.com/fft.htm. 7

AGARWAL, R. C.; GUSTAVSON, F. G.; ZUBAIR, M. A high performance parallel algorithm for 1-d fft. In: **Supercomputing '94: Proceedings of the 1994 conference on Supercomputing**. Los Alamitos, CA, USA: IEEE Computer Society Press, 1994. p. 34–40. ISBN 0-8186-6605-6. 5, 6, 7, 8

ALI, A.; JOHNSSON, L.; SUBHLOK, J. Scheduling fft computation on smp and multicore systems. In: **ICS '07: Proceedings of the 21st annual international conference on Supercomputing**. New York, NY, USA: ACM, 2007. p. 293–301. ISBN 978-1-59593-768-1. 5, 6, 7, 8

BAHN, J. H.; YANG, J.; BAGHERZADEH, N. Parallel fft algorithms on network-on-chips. **Information Technology: New Generations, Third International Conference on**, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 1087–1093, 2008. 5, 6, 7, 8

BELANOVIC, P.; LEESER, M. A library of parameterized floating-point modules and their use. In: **FPL '02: Proceedings of the Reconfigurable Computing Is Going Mainstream, 12th International Conference on Field-Programmable Logic and Applications'**. London, UK: Springer-Verlag, 2002. p. 657–666. ISBN 3-540-44108-5. 14

BONDHUGULA, U.; DEVULAPALLI, A.; DINAN, J.; FERNANDO, J.; WYCKOFF, P.; STAHLBERG, E.; SADAYAPPAN, P. Hardware/software integration for fpga-based all-pairs shortest-paths. In: **FCCM '06: Proceedings of the 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines**. Washington, DC, USA: IEEE Computer Society, 2006. p. 152–164. ISBN 0-7695-2661-6. 10

CHAMBERLAIN, R. D.; LANCASTER, J. M.; CYTRON, R. K. Visions for application development on hybrid computing systems. **Parallel Comput.**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 34, n. 4-5, p. 201–216, 2008. ISSN 0167-8191. 5

CHAO, C.; QIN, Z.; YINGKE, X.; CHENGDE, H. Design of a high performance fft processor based on fpga. In: **ASP-DAC '05: Proceedings of the 2005 conference on Asia South Pacific design automation**. New York, NY, USA: ACM, 2005. p. 920–923. ISBN 0-7803-8737-6. 5, 6, 7

COOLEY, J. W.; TUKEY, J. W. An algorithm for the machine calculation of complex fourier series. **Mathematics of Computation**, v. 19, n. 90, p. 297–301, 1965. Disponível em: <http://dx.doi.org/10.2307/2003354>. 7

Cray Inc. **Cray XD1 Datasheet**. Mendota, MN, USA, 2005. 4, 11

_____. **Cray XD1 FPGA Development**. Mendota, MN, USA, 2005. 3–7 p. 4, 10, 14

_____. **Cray XD1 FPGA Programming**. Mendota, MN, USA, 2005. 3–7 p. 14

_____. **Cray XD1 Hello World FPGA**. Mendota, MN, USA, 2005. 3–7 p. 14

_____. **Design of Cray XD1 RapidArray Transport Core**. Mendota, MN, USA, 2005. 3–7 p. 11, 12, 14

Dillon Engineering, Inc. Fast Fourier Transform (FFT) IP Cores for FPGA and ASIC. Http://www.dilloneng.com/fft_ip. 5, 6, 7

FRIGO, M.; JOHNSON, S. G. Parallel FFTW. Http://www.fftw.org/parallel/parallel-fftw.html. 5, 6, 7, 8

GOKHALE, M.; GRAHAM, P. S. Reconfigurable computing: Accelerating computation with field-programmable gate arrays. In: **Reconfigurable Computing**. [S.l.: s.n.], 2005. ISBN 0-387-26106-0. 5, 14

GONZALEZ-CONCEJERO, C.; RODELLAR, V.; ALVAREZ-MARQUINA, A.; ICAYA, E. M. d.; GOMEZ-VILDA, P. An fft/ifft design versus altera and xilinx cores. In: **RECONFIG '08: Proceedings of the 2008 International Conference on Reconfigurable Computing and FPGAs**. Washington, DC, USA: IEEE Computer Society, 2008. p. 337–342. ISBN 978-0-7695-3474-9. 5, 6, 7

GUPTA, A.; KUMAR, V. The scalability of fft on parallel computers. **IEEE Trans. Parallel Distrib. Syst.**, IEEE Press, Piscataway, NJ, USA, v. 4, n. 8, p. 922–932, 1993. ISSN 1045-9219. 5, 6, 7, 8

HE, H.; GUO, H. The realization of fft algorithm based on fpga co-processor. **Intelligent Information Technology Applications, 2007 Workshop on**, IEEE Computer Society, Los Alamitos, CA, USA, v. 3, p. 239–243, 2008. 5, 6, 7

HEMMERT, K. S.; UNDERWOOD, K. D. An analysis of the double-precision floating-point fft on fpgas. **Field-Programmable Custom Computing Machines, Annual IEEE Symposium on**, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 171–180, 2005. 5, 6, 7

KINDRATENKO, V.; POINTER, D. A case study in porting a production scientific supercomputing application to a reconfigurable computer. In: **FCCM '06: Proceedings of the 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines**. Washington, DC, USA: IEEE Computer Society, 2006. p. 13–22. ISBN 0-7695-2661-6. 10

KINDRATENKO, V. V.; STEFFEN, C. P.; BRUNNER, R. J. Accelerating scientific applications with reconfigurable computing: Getting started. **Computing in Science and Engg.**, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 9, n. 5, p. 70–77, 2007. ISSN 1521-9615. 10

LEESER, M.; AL. et. VFLOAT: The Northeastern Variable precision FLOATing point library. Http://www.ece.neu.edu/groups/rpl/projects/floatingpoint, Acesso em 17/09/2008. 15

LIANG, J.; TESSIER, R.; MENCER, O. Floating point unit generation and evaluation for FPGAs. In: **FCCM '03: Proceedings of the 11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines**. [S.l.: s.n.], 2003. ISBN 0-7695-1979-2. 5

Ligon III, W. B.; AL. et. A re-evaluation of the practicality of floating-point operations on FPGAs. In: **FCCM '98: Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines'**. Washington, DC, USA: IEEE Computer Society, 1998. p. 206. ISBN 0-8186-8900-5. 14

PALMER, J. M. **The Hybrid Architecture Parallel Fast Fourier Transform (HAPFFT)**. Dissertação (Mestrado) — Brigham Young University, 2005. 5, 6, 7, 8

SHIRAZI, N.; WALTERS, A.; ATHANAS, P. Quantitative analysis of floating point arithmetic on FPGA based custom computing machines. In: **FCCM '95: Proceedings of the IEEE Symposium on FPGA's for Custom Computing Machines**. Washington, DC, USA: IEEE Computer Society, 1995. p. 155. ISBN 0-8186-7086-X. 5

UNDERWOOD, K. FPGAs vs. CPUs: trends in peak floating-point performance. In: **FPGA '04: Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays'**. New York, NY, USA: ACM, 2004. p. 171–180. ISBN 1-58113-829-6. 14

VELHO, H. F. de C.; CLAEYSSEN, J. C. R. Singular value decomposition in the numerical integration of an atmospheric model. In: **XIII Latin-American and Iberic Congress of the Computational Methods for Engineering**. [S.l.: s.n.], 1992. p. 344–353. 17, 18

VITE-FRIAS, J. A.; ROMERO-TRONCOSO, R. d. J.; ORDAZ-MORENO, A. Vhdl core for 1024-point radix-4 fft computation. In: **RECONFIG '05: Proceedings of the 2005 International Conference on Reconfigurable Computing and FPGAs (ReConFig'05) on Reconfigurable Computing and FPGAs**. Washington, DC, USA: IEEE Computer Society, 2005. p. 24. ISBN 0-7695-2456-7. 5, 6, 7

WAIN, R.; AL. et. An overview of FPGAs and FPGA programming; initial experiences at Daresbury. In: . Daresbury, Cheshire, UK: Council for the Central Laboratory of the Research Councils, 2004. p. 2–4. ISSN 1362-0207. 5, 15

Xilinx Inc. Fast Fourier Transform. Http://www.xilinx.com/products/ipcenter/FFT.htm. 5, 6, 7

ZHUO, L.; PRASANNA, V. K. High performance linear algebra operations on reconfigurable systems. In: **SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing**. Washington, DC, USA: IEEE Computer Society, 2005. p. 2. ISBN 1-59593-061-2. 10

_____. Scalable hybrid designs for linear algebra on reconfigurable computing systems. **IEEE Trans. Comput.**, IEEE Computer Society, Washington, DC, USA, v. 57, n. 12, p. 1661–1675, 2008. ISSN 0018-9340. 10