
Ministério da
Ciência e Tecnologia


INPE-00000-TDI/0000

# FFT PARALELA EM SISTEMA COMPUTACIONAL HÍBRIDO RECONFIGURÁVEL

Bruno Crestani Calegaro

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

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

INPE
São José dos Campos
20010

**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

# FFT PARALELA EM SISTEMA COMPUTACIONAL HÍBRIDO RECONFIGURÁVEL

Bruno Crestani Calegaro

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

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

INPE
São José dos Campos
20010

## RESUMO

Este trabalho tem como objetivo a continuidade do projeto de implementação de FFT em hardware reconfigurável, desenvolvido pelo bolsista Vitor Conrado Gomes. O objetivo específico é implementar uma Transformada Rápida de Fourier (FFT) em hardware reconfigurável,para acelerar a execução de um modelo meteorológico (DYNAMO).Para o desenvolvimento deste trabalho foram realizadas diversas atividades visando melhorar o desempenho da FFT implementada. 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 uso intensivo de processamento. Visando melhorar o desempenho de aplicações na área de computação científica, diversas estratégias têm sido desenvolvidas, sendo a computação paralela a mais difundida. Nesta abordagem, são utilizados diversos processadores de propósito geral para a computação de uma aplicação em paralelo. Contudo, a tendência atual é a utilização de dispositivos reconfiguráveis, em especial *Field-Programable Gate Arrays* (FPGAs), para a computação de aplicações intensivas (CHAMBERLAIN et al., 2008). A utilização de FPGAs ou mais dispositivos para a execução de uma tarefa define um recente paradigma conhecido como computação híbrida.

FPGAs são dispositivos lógicos programáveis que funcionam como circuitos dedicados. Sua vantagem é ter a flexibilidade de soluções baseadas em software com o desempenho de soluções implementadas em hardware (CHAMBERLAIN et al., 2008;

WAIN; AL., 2004). Elas permitem aumentar significativamente a densidade computacional e consomem menos energia que microprocessadores. Em sistemas híbridos reconfiguráveis de alto desempenho, o objetivo é acelerar a execução de tarefas críticas de uma aplicação através da configuração do FPGA como um coprocessador especializado.

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; GUPTA; KUMAR, 1993; PALMER, 2005; HE; GUO, 2008). Seguindo esta tendência, o trabalho implementou a Transformada Rápida de Fourier em Hardware Reconfigurável tendo como objetivo o aumento do desempenho da computação de um sistema evolutivo implementado em software(GOMES, 2009). Esta integração configura um sistema de computação híbrida.

Devido ao estado atual do projeto estar na fase de otimização, foi seguido um estudo de estratégias a fim de melhorar o desempenho da atual implementação. Esse estudo tem como alvo a comunicação entre CPU e FPGA, fator crucial no desempenho da aplicação. O objetivo principal é acelerar a execução da FFT implementada. O ambiente de testes utilizado é um Cray XD1 que incorpora FPGAs em sua arquitetura para permitir a aceleração de aplicações críticas.

Neste relatório são apresentados os conceitos e as tecnologias utilizados no desenvolvimento deste trabalho, incluindo informações relevantes sobre dispositivos reconfiguráveis e sobre o sistema computacional híbrido utilizado. Na sequência, apresenta-se as atividades desenvolvidas e os resultados obtidos até o momento, assim como as conclusões.

# 2 OBJETIVOS

Este trabalho tem como objetivo pesquisar estratégias de implementação para melhorar o desempenho da Transformada Rápida de Fourier em Hardware Reconfigurável (FPGA) implementada. Esta pesquisa visa obter ganho de desempenho na execução desta operação em sistemas híbridos para assim integrar em um sistema evolutivo simples.

Como objetivos, temos:

- Pesquisar estratégias de implementação para aumentar a performance;
- Implementar essas estratégias no projeto;
- Aplicar a nova 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 são mostrados os fundamentos de computação híbrida, 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; HE; GUO, 2008), enquanto outros buscam o aumento de desempenho através de estratégias de paralelização em software (AGARWAL et al., 1994; GUPTA; KUMAR, 1993; PALMER, 2005).

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 algoritmo 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.

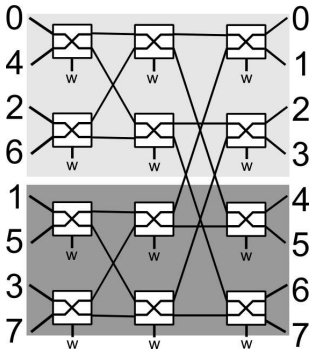


Figura 3.1 - Fluxo computacional para FFT de 8 pontos

## 3.2 Computação Híbrida Reconfigurável

De maneira formal, a Computação Híbrida, também conhecida como Computação Heterogênea, pode ser definida como a estratégia de utilizar vários tipos de componentes de processamento em um único fluxo de trabalho, permitindo que cada dispositivo execute tarefas diferentes. Esses dispositvos podem ser: GPUs, DSPs, FPGAs, etc, e cooperam com processadores de propósito geral na execução de uma aplicação. A Computação Híbrida Reconfigurável é uma subárea da Computação Híbrida que utiliza dispositivos reconfiguráveis (FPGAs) como coprocessadores especializados.

O conceito de computação híbrida, está se tornando cada vez mais importante no cenário de computação de alto desempenho. Em sistemas híbridos reconfiguráveis de alto desempenho, o objetivo é acelerar a execução de tarefas críticas através da configuração do FPGA como um coprocessador especializado. Aplicações para estes sistemas têm o potencial de obter um grande aumento de desempenho em relação às soluções baseadas somente em software. A grande vantagem desta abordagem é o paralelismo próprio de dispositivos reconfiguráveis em conjunto com o processamento em CPU.

### 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.

O uso de FPGAs tem algumas vantagens sobre sistemas paralelos de computadores, como no caso de clusters. Primeiramente, porque FPGAs consomem menos energia que os microprocessadores. Além disso, FPGAs permitem aumentar significativamente a densidade computacional e o desempenho de tarefas (CHAMBERLAIN et al., 2008). No entanto, nem todas aplicações podem ter um aumento de desempenho com o uso de hardware reconfigurável, em especial aquelas que possuem pouca oportunidade de paralelização.

### 3.2.2 Fluxo de desenvolvimento

O desenvolvimento de aplicações para FPGAs é feito através de linguagens de descrição de Hardware. Linguanges como VHDL e Verilog, permitem a descrição do comportamento lógico do sistema e o fluxo interno de dados. A conversão da 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. Primeiramente o processo de síntese transforma a lógica em alto nível e o código comportamental em portas lógicas interconectadas. Na sequência, o processo de mapeamento separa as portas lógicas em grupos para poderem ser melhor adaptadas aos recursos lógicos do FPGA. O roteamento vem na sequência, este processo indica em qual bloco logico cada grupo de portas lógicas vai ser configurado e determina as interconexões que irão transportar os sinais. 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 etapas deste fluxo de desenvolvimento.

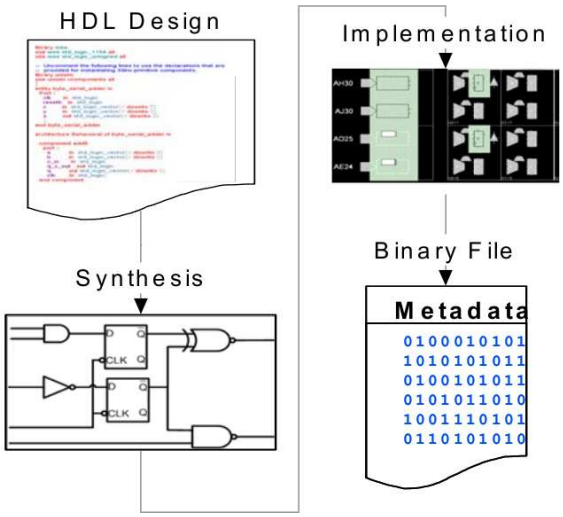


Figura 3.2 - Fluxo de desenvolvimento de aplicação para FPGA. Fonte: Adaptada de Cray Inc. (2005c)

## 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 (ZHUO; PRASANNA, 2005). 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 reconfigurável de alto desempenho 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(Cray Inc., 2005f). Na figura 3.3 é possível ver um *rack* com diversos equipamentos XD1 e um equipamento em destaque.

Cada sistema Cray XD1 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 na figura 3.5. É possível observar que o dispositivo reconfigurável tem acesso direto a quatro ban-


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

cos de memória QDR II SRAM, que possuem 4MB cada. O RapidArray Processor permite que os processadores enviem dados para o FPGA 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.

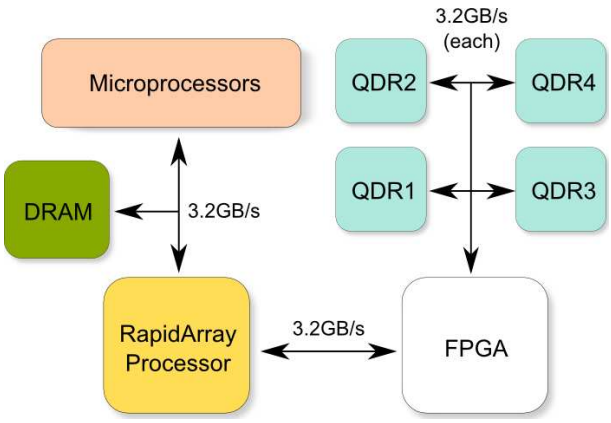


Figura 3.4 - Arquitetura do Blade do Cray XD1

### 3.4.1 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 na figura 3.5. 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 FPGA 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.

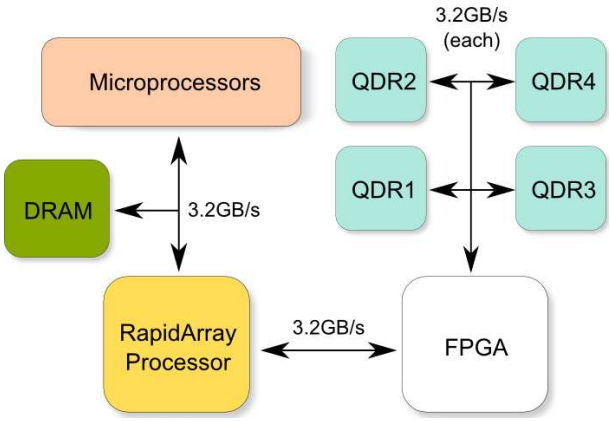


Figura 3.5 - Arquitetura do Blade do Cray XD1

### 3.4.2 Hierarquia de memória

A arquitetura híbrida do XD1 permite ao FPGA acesso a diferentes níveis de memória. Além dos quatro blocos de SRAM e da memória DRAM, o FPGA pode utilizar blocos de memória internos ao hardware reconfigurável.

O uso destes três níveis de memória deve ser planejado no desenvolvimento de uma aplicação, para evitar a redução de eficiência pelo gargalo no acesso à memória. Cada recurso disponível tem forma de acesso e quantidades distintas. 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 uma interface de comunicação disponibilizada pela Cray, o *RapidArray Transport Core*. Em um nível mais baixo 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, e o acesso é feito utilizando o QDR II SRAM Core, também disponibilizado pelo fabricante (Cray Inc., 2005g). Além desses, a família de FPGAs Vitex Pro possuem bancos de memória internos que podem ser acessados diretamente em um único ciclo.

A DRAM é o mais alto níve e normalmente é utilizada para compartilhar dados com o FPGA, sendo carregados para a QDR II SRAM para serem acessados durante a execução da aplicação. A memória interna do FPGA, disponível em menor quantidade, é utilizada para registradores e cache de dados. Cada problema necessitará de recursos e formas de acesso diferentes que devem ser otimizadas conforme a necessidade. A correta utilização destes níveis de memória podem favorecer a obtenção de desempenho no desenvolvimento de aplicações para esta arquitetura.

### 3.4.3 Modos de comunicação

A comunicação entre CPU e FPGA também é um elemento chave que precisa ser considerado durante o desenvolvimento de aplicações híbridas. 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., 2005h).

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. Normalmente somente é permitida a manipulação de um *quadword* (64 bits) por requisição utilizando o bloco Fabric Request (Cray Inc., 2005h).

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., 2005h). 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.

Cabe salientar ainda que existe um modo de manipular mais de um *quadword* em apenas uma requisição. Essa técnica é conhecica como *burst mode*, ou modo rajada, e pode ser usada tanto pelo Fabric Request quanto pelo User Request. Seu uso pode aumentar a transferência de dados entre a CPU e a FPGA, e gerar um ganho de desempenho quando usada corretamente.

# 4 ATIVIDADES DESENVOLVIDAS

Neste capítulo são apresentadas as atividades desenvolvidas neste trabalho.

## 4.1 Estudo do ambiente de desenvolvimento

Em primeiro lugar foi realizados 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. Ao mesmo tempo foram realizados estudos sobre a linguagem VHDL, que é uma linguagem de descrição de *hardware* utilizada na implementação do projeto. Essa linguagem baseia-se em criar uma descrição do comportamento lógico do *hardware*, contudo, alguns conjuntos de operações não podem ser implementados para ser utilizado em FPGAs, somente podem ser simulados em software. Por isso, foram estudadas as técnicas de implementação em VHDL de descrições sintetizáveis.

Através do estudo dos manuais da Cray(Cray Inc., 2005h; Cray Inc., 2005c; Cray Inc., 2005d) e exemplos (Cray Inc., 2005e; Cray Inc., 2005b) foram executados testes com a finalidade de aprender o funcionamento do sistema.

## 4.2 Estudo da FFT e implementação em software

Foi realizado um estudo teórico sobre a Transformada Rápida de Fourier e sua utilização em aplicações numéricas, bem como os algoritmos de computação dessa transformada(COOLEY; TUKEY, 1965; TAKAHASHI, 1999). Uma das formas de estudo foi implementar essa transformada em linguagem C, tendo como objetivo familiarizar-se com a tranformada.

### 4.2.1 Análise de Estratégias de Comunicação

Conforme já discutido no ítem 3.4.3 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. Para desenvolvimento de aplicações de teste foram utilizados os manuais (Cray Inc., 2005h; Cray Inc., 2005c; Cray Inc., 2005d) e exemplos (Cray Inc., 2005e; Cray Inc., 2005b) disponibilizados pela Cray. Uma parte dessa etapa já estava concluída (GOMES, 2009), e teve seu trabalho continuado tendo como meta a implementação do *modo rajada*. Primeiramente, pretendeu-se modificar o bloco Fabric Request para que esse atenda as solicitações do *modo rajada*. A seguir foram realizados testes de desempenho deste bloco tendo como comparação os resultados anteriores.

# 5 RESULTADOS

## 5.1 Modos de comunicação

o resultado dos testes realizados para a obtenção da taxa de transferência de dados entre CPU e FPGA com o *modo rajada* podem ser vistos na tabela 5.1. 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.

A análise dos dados do teste revelam nenhuma mudança significativa. Com isso se conclui que é necessário um melhor entendimento do *modo rajada* para que ele seja usado corretamente, pois alguns resultados mostram valores negativos o que representa que não houve altereção significante na comunicação. Outro fator que deve ser levado em conta, é que o bloco Fabric Request usa uma abordagem do tipo

Tabela 5.1 - Tempos de comunicação

| | Tempo ($\mu$s) | | SpeedUp (%) |
|---|---|---|---|
| Dados | Fabric Request | Burst Mode | |
| 8B | 0,8 | 0,7 | 12,5 |
| 16B | 1,3 | 1,2 | 7,7 |
| 128B | 10 | 9,9 | 1,89 |
| 1KB | 78,7 | 80,5 | −2,28 |
| 16KB | 1269 | 1269,2 | 0,1 |
| 128KB | 10242,6 | 10225,9 | 0,16 |
| 1MB | 81594,9 | 82064,6 | −0,5 |
| 2MB | 163155,1 | 164374,3 | −0,7 |

*pull*, o que mantém os processadores ocupados durante a transferência de dados, dessa forma, volta-se a atenção para o bloco User Request que teoricamente terá melhor aproveitamento desta técnica. Sendo assim, o trabalho está sendo continuado e novos testes serão realizados.

## 6 CONCLUSÃO

Com as atividades realizadas pode-se concluir que o uso de sistema híbridas como alternativa a computação de soluções científicas tende a crescer. Usar o paradigma de computação híbrida reconfigurável necessita de uma abordagem diferenciada em relação a sistemas de computação tradicionais. É importante ressaltar que o uso de diferentes dispositivos requer uma avaliação de qual tarefa é melhor de se executar. No trabalho é observado que usando uma FPGA como um coprocessador especializado em conjunto com o processador de propósito geral, é possível melhorar o desempenho do cálculo da Transformada Rápida de Fourier. Além disso, podem ser aplicadas diversas otimizações, tais como a especialização do modo de comunicação entre a CPU e a FPGA, que levariam a resultados ainda mais satisfatórios.

## REFERÊNCIAS BIBLIOGRÁFICAS

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, 7

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, 2008. ISSN 0167-8191. 4, 5, 9

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. 7, 15

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

_____. **Cray XD1 DMA FPGA Desing**. Mendota, MN, USA, 2005. 14, 15

_____. **Cray XD1 FPGA Development**. Mendota, MN, USA, 2005. 4, 10, 14, 15

_____. **Cray XD1 FPGA Programming**. Mendota, MN, USA, 2005. 14, 15

_____. **Cray XD1 Hello World FPGA**. Mendota, MN, USA, 2005. 14, 15

_____. **Cray XD1 System Overview**. Mendota, MN, USA, 2005. 10

_____. **Design of Cray XD1 QDR II SRAM Core**. Mendota, MN, USA, 2005. 12

_____. **Design of Cray XD1 RapidArray Transport Core**. Mendota, MN, USA, 2005. 13, 14, 15

GOMES, V. C. Transformada rápida de fourier paralela em sistema computacional híbrido reconfigurável. In: . [S.l.: s.n.], 2009. 5, 15

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, 1993. ISSN 1045-9219. 5, 7

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, 2008. 5, 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, 2005. 7

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

TAKAHASHI, Y. K. D. High-performance radix-2, 3 and 5 parallel 1-d complex fft algorithms for distributed-memory parallel computers. Computer Centre, University of Tokyo, Yayoi, Bunkyo-ku, Tokyo, Japan, 1999. 15

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. ISSN 1362-0207. 4, 5

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. ISBN 1-59593-061-2. 10