UFSC – UNIVERSIDADE FEDERAL DE SANTA CATARINA

Computação nas Nuvens e Computação de Alto Desempenho

Autor: Helber Maciel Guerra

Florianópolis – SC

2011/1

UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

CURSO DE SISTEMAS DE INFORMAÇÃO

Computação nas Nuvens e Computação de Alto Desempenho

Autor: Helber Maciel Guerra

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação

Florianópolis – SC

2011/1

Acadêmico: Helber Maciel Guerra

Computação nas Nuvens e Computação de Alto Desempenho

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador:

_____
Professor Mário Antônio Ribeiro Dantas, Dr.
Universidade Federal de Santa Catarina
mario@inf.ufsc.br

Banca Examinadora:

_____
Professor José Eduardo De Lucca, Me.
Universidade Federal de Santa Catarina
delucca@inf.ufsc.br

_____
Professora Lúcia Helena Martins Pacheco, Dr.
Universidade Federal de Santa Catarina
lucia@inf.ufsc.br

Agradecimentos

Dedico meus sinceros agradecimentos a:

Professor Mário Antônio Ribeiro Dantas, pela orientação e a motivação.

Minha esposa Tamara, pela paciência e compreensão do tempo dispendido para a realização deste trabalho.

A meu amigo Rodrigo, por me incentivar e ajudar na realização deste trabalho.

A meus amigos da DHW engenharia por acreditarem em meu trabalho e emprestarem os equipamentos para os testes e ajudarem a entender o real funcionamento do FPGA e seu ambiente.

Resumo

A grande demanda por poder computacional na área de computação nas nuvens, principalmente por potenciais clientes de grande porte, como a NASA, ávida por recursos específicos de cálculos como trigonometria, matrizes, vetores, ponto flutuante, entre outros. Estes clientes potenciais estão em processo de terceirização de suas infraestruturas para fornecedores de grande porte como Amazon e Rackspace, para centralizar os recursos e ter uma melhor utilização de suas infraestruturas, recursos energéticos e segurança.

Para ter uma melhora de desempenho do poder computacional é necessário uma aceleração de funções computacionais em hardware, para reduzir a quantidade de ciclos de CPU, e consequentemente o tempo necessário para se obter os resultados da computação.

Este trabalho tem como objetivo analisar a viabilidade do uso a aceleração em hardware para melhorar o desempenho de processamento em ambiente de computação nas nuvens utilizando FPGA. Pois este provê uma maneira de adaptar o hardware e moldar neste, a funcionalidade necessária para resolver cada um dos problemas específicos, o que não ocorre com soluções atualmente adotadas para aceleração em hardware, que são GPU e CELL.

Com a utilização de FPGA para resolver problemas especializados, tem-se uma melhoria significativa dentro da função específica, podendo em alguns casos ser 500 vezes [28] mais rápida que o calculo em processador convencional, mas perdendo este desempenho em funções básicas necessárias para realizar a operação, como é o caso de cópia ou carga dos dados para a memória, ou manipulação básica de barramento e rede.

Palavras chave: paralelização, grid, cluster, cloud, FPGA, GPU, CELL, HPC, TI Verde.

Abstract

The high demand for computing power in the area of cloud computing, especially for potential large customers such as NASA, eager to specific resources such as trigonometry, calculus, matrices, vectors, floating point, among others. These potential customers are in the process of outsourcing their IT infrastructures to large suppliers such as Amazon and Rackspace, to centralize the resources and have a better use of its infrastructure, energy resources and security.

To have a performance improvement of computing power is required an acceleration of computing functions in hardware to reduce the amount of CPU cycles, and hence the time needed to obtain the results of computation.

This study aims to examine the feasibility of using hardware acceleration to improve processing performance in an environment of cloud computing using FPGA. Because this provides a way to adapt the hardware and framing this, the functionality needed to solve each of the specific problems, which does not occur with solutions currently adopted for hardware acceleration, which are GPU and CELL.

With the use of FPGA to solve specialized problems, has been a significant improvement in the specific function, in some cases may be 500 times [28] faster than standard processors in the computation, but losing this performance in basic functions needed to perform the operation, such as copy or load data into memory, or basic manipulation on bus network.

Keywords: parallel, computer grids, computer cluster, cloud, green IT.

Sumário

Lista de Figuras

Lista de Siglas

AaaS - Aplication as a Service (Aplicação como Serviço) - Antes SaaS - Softtware como Serviço

API - Aplication Program Interface (Interface de Aplicação)

DaaS - Database as a Service (Banco de Dados como Serviço).

FOSS - Free and Open Source Software (Software Livre e de Código Fonte Aberto)

FPGA - Field Programmable Gate Array (Arranjo ou matriz de portas programável em campo).

GPU - Graphics Processing Unit (Unidade de Processamento Gráfico)

IaaS - Infra Estrutura como Serviço.

PaaS - Plataforma como Serviço.

JSON - Javascript Object Notation (Representação de Objeto em Linguagem Javascript)

MPI - Message Passing Interface (Interface de passagem de mensagens)

PVM - Parallel Virtual Machine (Maquina virtual paralela)

REST - (Representacional State Transfer)

SaaS - Storage as a Service (Armazenamento como Serviço).

SSI - Single System Image (Sistema de Imagem única)

URI - Uniform Resource Identifier é uma cadeia de caracteres compacta usada para identificar ou denominar um recurso na Internet

1 Introdução

A computação paralela permite a execução simultânea de processos de forma independente, sendo caracterizada por dois tipos de paralelismo básico, (i) paralelismo real, cuja unidades de processamento independentes realizam determinado processo e (ii) paralelismo virtual, onde dentro de um mesmo núcleo de processador, é emulado o paralelismo através do sistema operacional com compartilhamento de tempo de processamento. Para a troca de informações entre os processos e o controle de cada um dos processadores, é necessário um barramento de interconexão, onde é feito a troca de mensagens. O barramento de interconexão através da demanda crescente, ultrapassou os limites de uma maquina e são interconexões entre maquinas, utilizando redes de transmissão de dados locais e até a internet.

Dessa forma, devido à necessidade de flexibilização da expansão de recursos sob demanda, a tecnologia de computação paralela e distribuída teve que adaptar-se ao mercado. Uma das infraestruturas utilizadas na computação paralela e distribuída é a de Grid Computing. A computação em grade representa um significativo avanço na computação paralela e distribuída.

Segundo [1], todo o ferramental gerado para a utilização de computação em grade, junto com a técnicas de localização baseadas em URI e WEB 2.0 fomentou a criação do modelo de computação que hoje conhecemos chamamos de computação nas nuvens (*Cloud Computing*). Mas este modelo já era previsto segundo [2], onde é mostrada a importância do poder de processamento descentralizado, e sem localização física específica. Segundo [3] o uso de a infraestrutura de grades com a adição de uma camada de virtualização, para a abstração do recurso físico possibilitou a formação do modelo de computação nas nuvens, e neste mesmo artigo foi feita uma extensa análise comparativa entre os dois modelos.

Dados publicados em periódicos da área de TI demonstram o significativo interesse pelo assunto, como o artigo publicado [4] pela revista eletrônica TI-Inside no dia 17 de agosto de 2010, citando pesquisas da Frost&Sullivan pelo analista de mercado Fernando Belfort. Segundo Fernando Belfort até o ano de 2012 computação nas nuvens (cloud computing) será adotada pela maioria das empresas brasileiras. A pesquisa também aponta um crescimento no interesse dos CIOs brasileiros dos atuais 50% para 85% até 2012 no assunto. Sendo que o modelo com maior índice de uso, aproximadamente 70% é o de nuvens privadas, restando os outros 30% para as nuvens públicas. Este paradigma é adotado devido redução de custos, flexibilidade operacional e devido a continuidade dos negócios. Contudo, ainda existem obstáculos a serem vencidos, tais como a falta de conhecimento e definição acerca do assunto e fundamentalmente a cultura estanque que o ambiente de TI está balizada.

Alguns analistas e fornecedores tentam definir computação nas nuvens como servidores virtuais disponíveis na internet em um modelo *outsourcing.* Servidores virtuais resolvem uma parte do que é cloud computing por facilitar a agregação de recursos (memória, cpu e armazenamento), geralmente em um modelo IaaS, mas não define o que realmente é computação nas nuvens, tentando assim se beneficiar do termo que está em evidencia.

O presente trabalho tem como objetivo analisar a viabilidade da utilização de FPGA como componente de aceleração de serviço em computação nas nuvens, e colocar uma melhor definição do que vem a ser computação nas nuvens.

1.1 Problema de pesquisa:

Com a popularização do uso de computação nas nuvens, a necessidade de cálculos para tratamento de problemas recorrentes e muitas vezes especializados precisarão de mecanismos que ofereçam, a um baixo custo, resposta ágil.

Atualmente a utilização de arquiteturas de processadores GPU e CELL atendem ao requisito de aumento da rapidez em cálculos vetoriais, alguns cálculos de matrizes e matemática de ponto flutuante por exemplo, mas não oferecem a flexibilidade necessária para atender diferentes realidades de negócio que podem ser flexibilização do modelo de hardware, para realizar cálculos especializados, com o desempenho fornecido pelo hardware, a um custo atrativo para o volume de chips especializados nos cálculos desejados, tratando os volumes de dados necessários apenas para o problema.

A resolução de problemas específicos recorrentes (resolução de matrizes, cálculo de vetores, conversão de mídias, entre outros), bem como a utilização destes componentes para outras funcionalidades alem daquelas para os quais foram projetadas é uma grande limitação principalmente das arquiteturas baseadas em GPU.

Até o presente momento existem estudos da utilização de FPGA em alguns grids como "*Uma abordagem de alto desempenho para multiplicação de matrizes densas em sistemas reconfiguráveis" [5].*

1.2 Solução proposta:

Utilizar FPGA, para aceleração de funcionalidades específicas dentro do ambiente de computação nas nuvens, para obter a flexibilidade de moldar este chip através de sua reprogramação dinâmica e a velocidade de processamento do hardware.

1.3 Objetivo Geral:

Analisar a possibilidade de utilizar aceleração em *hardware* reconfigurável para ser utilizado como um serviço em computação nas nuvens.

1.4 Objetivo Específico:

Analisar a possibilidade da utilização da aceleração em hardware para computação

nas nuvens usando FPGA, que pode ser moldada à necessidade de funcionalidade específica a ser acelerada, e após sua utilização possa ser reutilizada para outro consumidor de serviços, com outra necessidade de hardware diferente do anterior.

1.5 Motivação:

É evidente os benefícios de redução de custos obtidos pela utilização de computação nas nuvens, como economia de energia, melhor aproveitamento de recursos computacionais. As funções recorrentes necessárias para resolver problemas está mais evidente pois entidades que consomem grande volume de processamento para cálculos científicos, como a NASA, que está terceirizando seu ambiente de computação nas nuvens para o provedor Hackspace [6].

2 Ambientes Computacional Paralelos e Distribuídos

2.1 Computação em Cluster (Cluster Computing)

No glossário de computação paralela [7] computação em cluster consiste em um aglomerado de computadores distintos, interconectados através de uma rede de interconexão (geralmente uma rede local Ethernet) e visíveis como sendo um único computador paralelo, trazendo benefícios como alta disponibilidade, redundância. Os recursos compartilhados são recursos de mais baixo nível (CPU, memória, armazenamento). Segundo Dantas [8] “visualizavam que as aplicações pudessem se mover de um computador para o outro visando à utilização dos ciclos ociosos das máquinas que formavam a rede ”

Os computadores pertencentes ao cluster não são necessariamente preparados especificamente para o cluster, podendo ser computadores de uso geral (*off-the-shelf*) que se retirados do cluster podem ser utilizados isoladamente.

Alguns exemplos de cluster:

- SSI (*Single System Image*) onde os recursos compartilhados são de mais baixo nível, como cpu, memória e armazenamento. Exemplos deste tipo de cluster é o MOSIX (*Multicomputer Operating System for UnIX*) e OpenSSI [9] (*Open Single System Image*) que tem como característica poder rodar aplicações que não são preparadas para rodar em cluster sem a necessidade de modificação do código desta. A Figura [1] mostra um modelo simplificado em que os processos podem acessar a memória e armazenamento compartilhados, gerenciados pelo SO (Sistema operacional). Como este sistema é de arquitetura uniforme, o sistema operacional das maquinas podem fazer migração de processos entre os nós do cluster.

- Cluster baseado em troca de mensagens, que utilizam MPI e PVM para as troca de mensagens entre as maquinas. Um exemplo deste tipo de cluster é o Beowulf Cluster [10] que foi projetado pela NASA para processamento para processamento de informações espaciais.

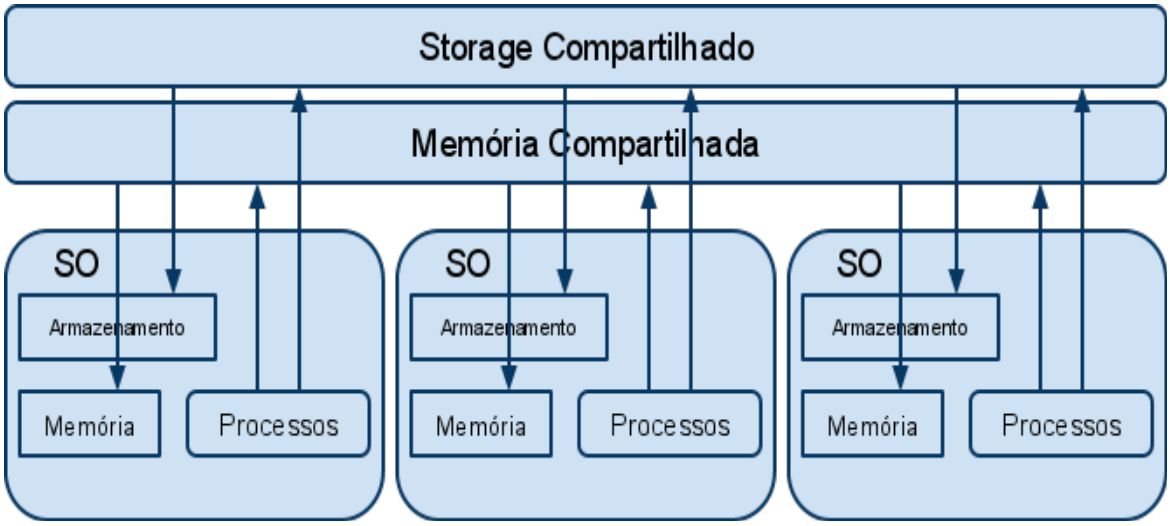


Figura 1: Modelo de Memória e Armazenamento Compartilhado SSI

2.2 Computação em Grade (*Grid Computing*)

Segundo Skillicorn [11] Computação em Grade (*Grid Computing*) são plataformas computacionais geograficamente distribuídas e acessível para seu usuário por uma interface única. Estas plataformas fornecem uma abstração de recursos computacionais físicos, através de uma camada de abstração, tais como armazenamento, memória e processamento de dados, permitindo que os dados sejam replicados e armazenados com redundância tornando assim, tolerantes à falhas. Segundo [8] algumas propriedades satisfeitas para definir um Grid:

- Estes devem ser grandes, tanto em número de recursos potenciais, como distâncias geográficas entre eles. Como os grids são distribuídos os atrasos na movimentação de dados deve ser considerados no desenvolvimento das aplicações que vão fazer uso destes. São dinâmicos tanto em recursos como o tempo de vida das aplicações;

- São heterogêneos;
- Vão além das fronteiras das organizações;

Para Berstis [12] a grande maioria das organizações possui recursos computacionais subutilizados. Este valor é estimado em 95% de tempo ocioso. Neste contexto, a computação em grade permite que um framework explore este recurso não utilizado de forma a distribuir e melhorar a eficiência de uso destes recursos.

O uso mais simples da computação em grade é a execução de lotes *(batch)* de trabalhos. Estes trabalhos são alocados dinamicamente entre os nós do grid, e estes podem executar outras tarefas em paralelo, ou executar as tarefas submetidas pelo grid quando estão aguardando pelo utilização *(idle state*) como observado na figura [2] abaixo, os nós (SO) podem ser de diferentes arquiteturas e as diferenças são abstraídas pelo *Middleware,* fazendo a interface entre o recurso e o grid.

Segundo Ian Foster [13] recursos não estão sujeitos a controle centralizado do grid, uso de padrões e protocolos de propósito geral e disponibilização de recursos, mesmo estes sendo não triviais, nos deixa a par da importância da tolerância a falha dos nós, pois está além do controle do grid. A figura [2] ilustra o grid BOINC [14], que é um *middleware* que utiliza computadores comuns de voluntários espelhados pela internet. Nos nós SO os recursos só estão disponíveis se estes estiverem ligados, conectados à internet e o usuário deste não esteja fazendo uso intensivo.

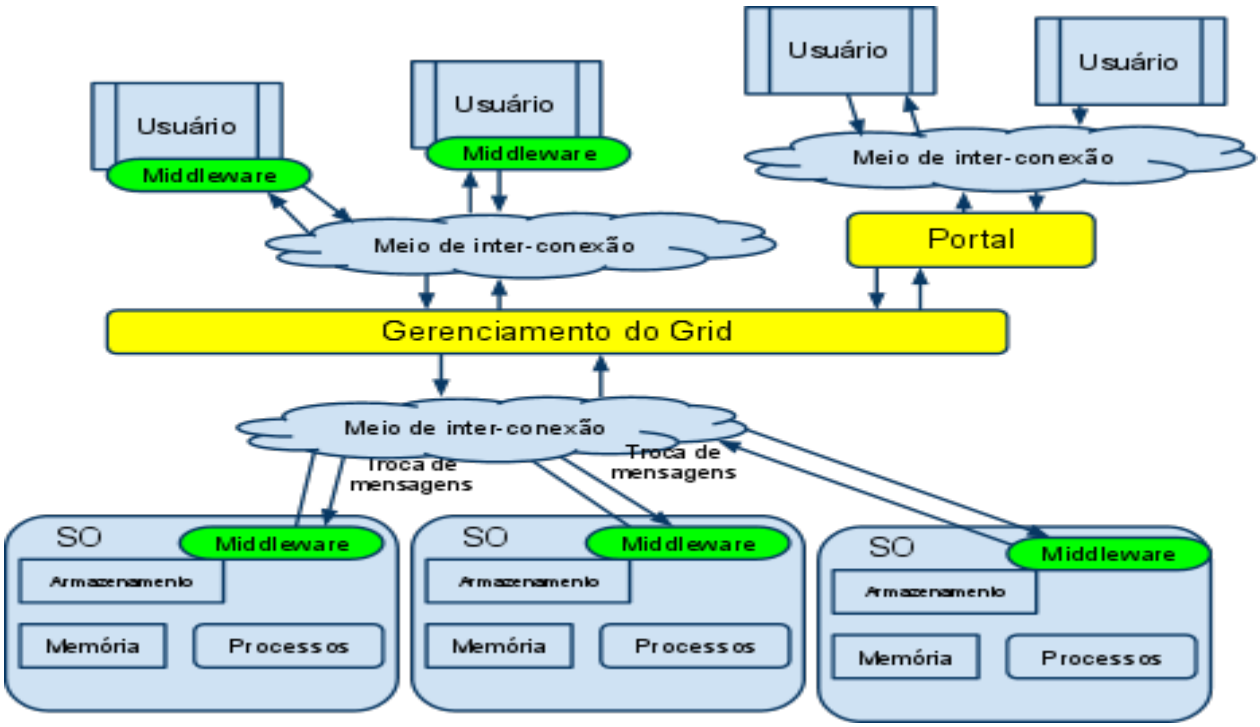


Figura 2: Arquitetura Simplificada de um Grid genérico.

Os grids podem ser interconectados formando um grid maior, mas para isto são necessários protocolos de intercomunicação. Segundo Foster [13], no capítulo “The Grid: The Need for InterGrid Protocols” através dos protocolos padronizados e abertos possibilita a integração, principalmente com o uso do projeto *open source* “Globus Toolkit” e OGSA [15] (*Open Grid Services Architecture*).

2.3 Computação nas Nuvens *(Cloud Computing)*

2.3.1 Definição:

Segundo [16] ainda não existe um consenso geral a respeito da definição sobre o que é computação nas nuvens ou Cloud Computing. Contudo é amplamente debatido que é o resultado da junção de tecnologias como *grid computing, utility computing*, SOA, Web 2.0 e outras tecnologias. Continuando, ainda segundo [16], o ponto central da computação nas nuvens está na capacidade de modificar (alocar ou liberar) o recurso computacional dinamicamente sem complexidade.

Segundo [17] página 10 Cloud Computing cunhado a partir do clichê Internet mais(+) Computação (Figura 3) e segundo Ben Pring, analista sênior do grupo Gartner, é como o termo Web 2.0, ninguém tem uma definição melhor.

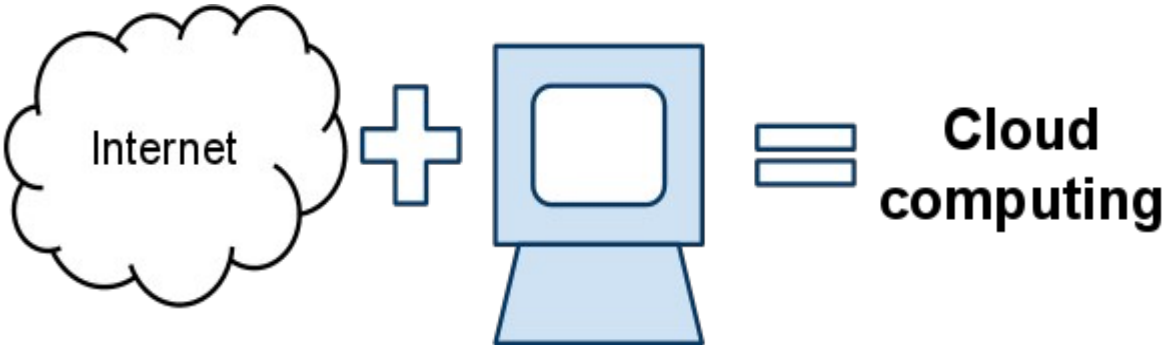


Figura 3: Cloud Computing

Segundo Scheier [17] (paginas 12-14, The 9 myths of cloud computing), unindo com Knorr e Gruman [17] (paginas 10-11, What cloud computing really is) em computação nas nuvens existem conceitos que devem ser levados em consideração, tais como:

- Computação nas nuvens não é virtualização, virtualização abstrai um computador físico dentro de uma aplicação que emula um computador real;
- Não existe caixinha pronta de computação nas nuvens, o cliente de nuvens deve

preparar sua aplicação para funcionar utilizando as APIs de seu fornecedor;

- Mesmo utilizando as mesmas plataformas em uma nuvem privada e uma nuvem publica, a integração e migração pode não ser triviais e sem problemas;
- A segurança não pode ser garantida apenas pelo fornecedor do nuvens, isto é verdade não só para as nuvens, como para segurança de qualquer tipo de aplicação;
- Utilizar serviços de computação nas nuvens requer o entendimento dos complexos recursos envolvidos para gerenciá-los corretamente;
- Computação nas nuvens requer APIs, pois entre as várias camadas e independência de localização, as aplicações diferentes precisam se comunicar baseados em URI (*Uniform Request Interface*) igual(=) URL;
- Armazenamento de arquivos, abstração de sistemas de arquivos;
- Monitoramento do estado das aplicações e processos, SLA (acordos de nível de serviço)
- Serviços - Aplicações, sistemas operacionais;

2.3.2 Complementação Adicional à Definição:

Adicionalmente o presente autor define computação nas nuvens também como:

- Sistema computacional altamente distribuído e abstraído de sua localização.
- Acessível através de APIs e URI (*Uniform Resource Information*)
- Ideia de recurso infinito (Escalabilidade sob demanda)
- Modelo de tarifação (Recursos como serviços)

3 A Computação nas Nuvens

3.1 Serviços

3.1.1 SaaS Armazenamento como Serviço:

Como visto no capítulo II o grid utiliza um middleware para abstrair os recursos computacionais tornando disponíveis para plena utilização ou sujeito a quotas, no caso de SaaS, pode ser utilizado os mesmos recursos do grid, mas o cliente utiliza uma fração deste recurso, conforme ele queira, e por um tempo definido. Um exemplo de sistema de arquivos distribuído utilizado pela Amazon S3 é o HDFS (Haddop Distributed File System), do projeto Apache Hadoop [18] que além de prover um sistema de arquivos redundante, tolerante à falhas e com balanceamento de acesso, fornece suporte à MapReduce (Mapeamento e Redução) que faz a divisão de dados e a posterior união destes por redução Figura 4, onde é realizado o mapeamento em uma lista de e-mails, que pode ser distribuídos para realizar alguma tarefa e unidos posteriormente por redução.

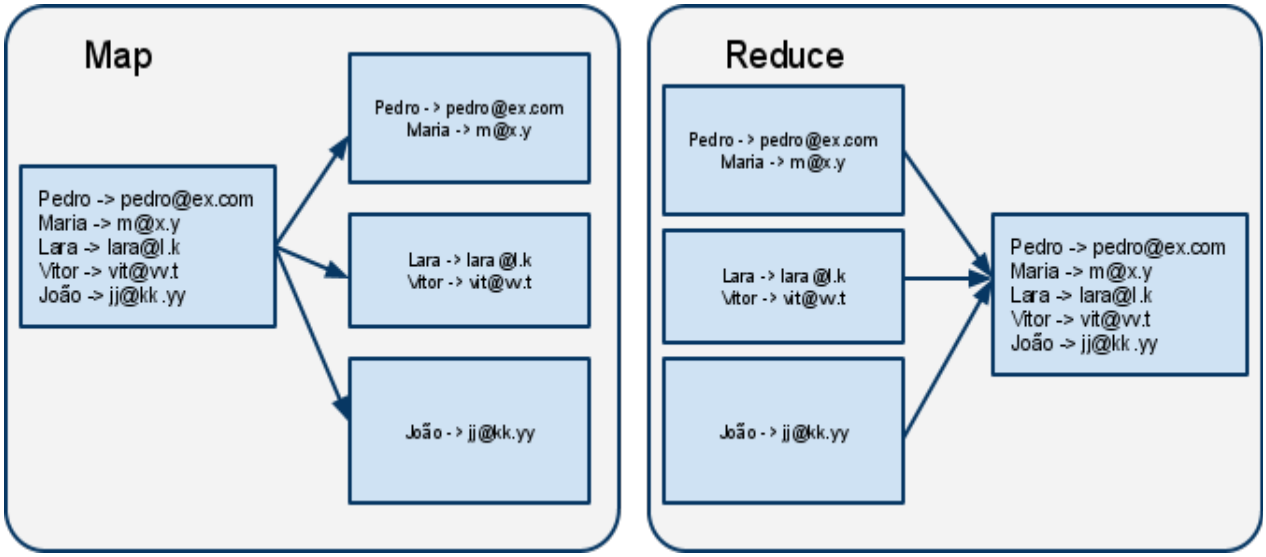


Figura 4: MapReduce

Alguns fornecedores de SaaS disponibilizam armazenamento para arquivos, tendo como vantagens garantia de alta disponibilidade, segurança de acesso e ubiquidade. Neste contexto, um arquivo armazenado em um serviço de cloud é acessíveis por interfaces de aplicação que são disponibilizados pelo fornecedor, como uma interface web, aplicativos de sincronização instalável, APIs associados a serviços e URIs, ou pontos de montagem remotos (WebDav, sshfs, etc). Uma desvantagem deste tipo de serviço é o atraso para a manipulação destes arquivos, da mesma maneira que em grids. Alguns serviços adicionais geralmente são utilizados em conjunto com storage cloud, como a possibilidade de compartilhar, fazer uso de controles de acesso, ou utilizar junto com outros serviços. Exemplos de cloud storage: Amazon S3, Memopal, Rackspace File. Exemplos de cloud storage aplicativos ou integrados ao sistema operacional: Ubuntu One, CloudApp para MACOs, DropBox.

3.1.2 DaaS - Banco de Dados como Serviço:

3.1.2.1 Banco de Dados Relacional (RDBMS):

Bancos de dados relacionais orientados à tabelas e geralmente transacionais. São bancos de dados de marcas conhecidas, como Oracle, MySQL, PostgreSQL, SQLServer, etc. No contexto de cloud, deve ser levado em consideração o *overhead* de operações sobre volumes de dados, sendo que estes bancos podem ter atrasos na transferência dos dados. A criação de manutenção destes servidores fica a cargo do fornecedor do serviço, bem como a infraestrutura para garantir o desempenho necessário pelo cliente de cloud.

3.1.2.2 Banco de Dados Não Relacional (NoSQL):

Estes bancos de dados conhecidos como NoSQL [19] (Not Only SQL) estão em

evidencia com o uso de cloud computing, devido a algumas facilidades na forma de implementar uma arquitetura facilmente escalável e distribuído. Na sua grande maioria os banco de dados não relacionais são chave valor, ou orientados à documentação. E dos orientados à documentação os documentos armazenados nestes bancos são documentos estruturados no formato JSON [21] Figura 5.

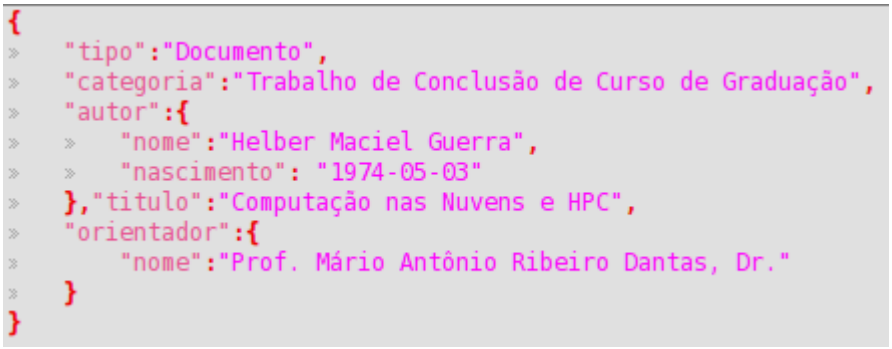
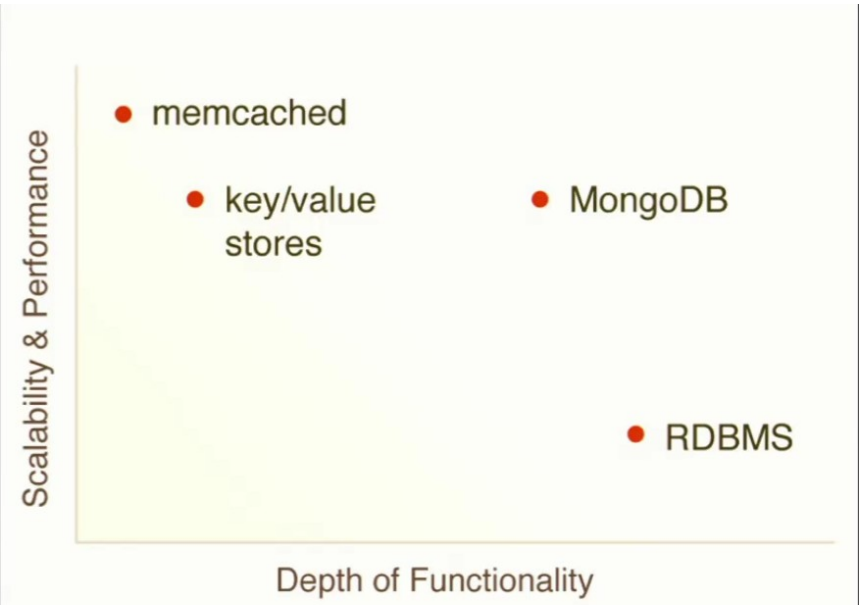


Figura 5: Exemplo de documento em formato JSON

MongoDB [20]: Ele é um FOSS e seu nome vem de “humongous” (Gigantesco). É um banco de dados orientado à documentação no formato BSON (Binary JSON), livre de esquema, com suporte à agregação, MapReduce (Figura 4), GridFS, Shard, replicação e índices geo-espaciais. Uma desvantagem deste tipo de banco de dados, a mesma de RMDBS em cluster, é que devemos ponderar a importância da replicação dos dados entre os nós, com a performance de armazenamento e recuperação da informação. Ao garantir a replicação de dados entre os nós, perdemos em performance, pois é necessário utilizar mecanismos que garantam que todos os nós de replica estejam com os dados consistentes.



Fonte: Palestra de Michael Dirolf Rubyconf 2009

Figura 6: Comparação de Banco de Dados

Memcached [21]: Banco de dados volátil, baseado em chave <=> valor, que é basicamente um esquema de memória distribuída entre os nós como visto em cluster (capítulo II 1).

CouchDB [22]: Banco de dados orientado à documentação com API baseada em HTTP/JSON, com modelo de acesso via REST [23], é um conjunto de princípios arquiteturais que quando aplicadas como um todo enfatiza a escalabilidade da interação entre componentes para reduzir a latência de interação, garantir segurança e encapsular sistemas legados,

Google BigTable [24]: Banco de dados chave <=> estrutura de dados, altamente distribuído, com estrutura de dados variável.

Cassandra [25]: Banco de dados inspirado no google BigTable, ele se utiliza de chave <=> colunas, altamente distribuído com agregação de nós e tolerância a falhas baseada em P2P.

3.1.3 AaaS - Aplicação como Serviço:

3.1.3.1 Aplicação:

No modelo de aplicação como serviço, o fornecedor disponibiliza a aplicação completa para o cliente e realiza a tarifação com numero de usuários, volume de transações, tempo pre definido entre outros.

Alguns fornecedores: Salesforce.com, Google Apps, NetSuite, Microsoft Online Services

3.1.3.2 Sistema Operacional:

O recurso computacional é executado nos servidores, e o usuário acessa este sistema operacional através de uma interface definida pelo provedor do serviço. Este sistema operacional pode ser completamente executado dentro do ambiente de nuvens ou parcial. E sua interface de acesso pode ser desde paginas web 2.0, interface gráfica de troca de mensagens, terminal remoto, entre outros.

No caso de execução mista, parte do sistema operacional é executada dentro de um dispositivo dedicado, e parte da execução é executada no ambiente de nuvens. Os dispositivos dedicados geralmente são baseados em um sistema operacional baseado em Linux, como por exemplo o google-chromeOS e o HP-webOS, e os dados e aplicativos ficam armazenados no ambiente de nuvem. Enquanto o sistema operacional local é responsável pelo gerenciamento do hardware as aplicações podem ser mistas (parte executada no dispositivo, parte na nuvem).

Podemos citar alguns exemplos de sistemas operacionais como: EyeOs, Google ChromeOs, HP-webOS, Windows RTP.

3.1.4 PaaS - Plataforma como Serviço:

Plataforma de desenvolvimento e execução que se utiliza da infraestrutura, componentes de software e ambiente de desenvolvimento do fornecedor, ambientes de execução das aplicações. Consiste em um ambiente de desenvolvimento integrado para projeto, desenvolvimento, testes e instalação para aplicações customizadas.

Ex.: Google App Engines, Microsoft Azure, Amazon.

3.1.5 InaaS - Informação como Serviço:

Antes conhecido como SaaS, mas a informação como serviço libera interfaces para consultas a determinados tipos de informação que podem ser previamente processados pelo fornecedor da informação, por exemplo cotação da bolsa de valores ou previsão de aumento de vendas.

3.1.6 IaaS - Infra Estrutura como Serviço:

É a disponibilização de recursos de infraestrutura sob demanda, que geralmente é um suporte à maquinas virtuais, para facilitar o gerenciamento, manutenção e customização pelo cliente de nuvens. O sistema operacional do cliente é enviado para o sistema de armazenagem do provedor de serviços, e o cliente acessa uma interface de gerenciamento para decidir sobre a alocação de recursos são disponibilizados para cada uma de suas instancias de maquinais virtuais que são executadas. O provedor de serviços de IaaS, geralmente disponibiliza maquinas virtuais pré instaladas e configuradas chamadas de *appliances* virtuais, prontas para ser executadas dentro do ambiente.

Ex.: Amazon EC2 (Amazon Elastic Cloud Computing)

4 Computação nas Nuvens e Computação de Alto Desempenho

4.1 FPGA, GPU, CELL

4.1.1 CELL (CELL/B.E.)

Segundo [26] [22 paginas 209 e 210] Cell Broadband Engine (CELL/B.E), é um multiprocessador heterogêneo com 9 cores (núcleos) projetado inicialmente para ser utilizado no console Playstation 3 (PS3), pela Sony, Toshiba e IBM, hoje bastante utilizado em multimídia e HPC. O CELL/B.E. possui 9 núcleos, sendo 1 Power Processing Element (PPE) , agindo como processador principal, e 8 Synergistic Processing Elements (SPEs) , agindo como coprocessadores. Para interconectar estes elementos existe um barramento EBI (High bandwidth Element Interconnection Bus). Este elemento conecta o processador principal com os outros processadores, com uma topologia em anel, tendo este uma velocidade de interconexão que chega ao pico de 204,8 Gflops em precisão simples ou seja 204,8 GB/s. O PPU (Power processor Unit) com precisão de 64 bits, e 2 threads e VMX/AltiVec unit, dentro do núcleo.

4.1.2 GPU

O GPU é a sigla para Graphical Processor Unit (Unidade de processamento gráfico), que tem como principal função e de executar instruções de processamento gráfico, e gerar a interface com o usuário em computadores convencionais. Devido à grande quantidade de instruções, velocidade de execução e numero de unidades processadoras disponíveis nas placas graficas, os fornecedores de placas graficas disponibilizaram meios de acessar estes recursos, para outros fins diferentes do primário. As interfaces de acesso aos recursos das GPUs geralmente são fornecidas pelos fabricantes de placas por meio de SDKs. Os fornecedores mais famosos de placas gráficas são NVIDIA e AMD,

com os SDKs CUDA e AMD Developer Kit respectivamente.

Para facilitar a implementação por parte dos usuários de serviços de GPU, foi criado um projeto chamado OPENCL [27], que abstrai as funcionalidades específicas de cada fornecedor trazendo um ambiente de desenvolvimento com funções padronizadas de acesso aos recursos.

Segundo [26] [22 pagina 210] os elementos de processamento dentro das GPUs compartilham uma memória de registradores onde os dados a ser processados são acessados por cada um dos kernels de execução, e estes são executados dentro de cada um dos chips de processamento gráfico. O ambiente de desenvolvimento é baseado em C/C++ e é então compilado nestes kernels, que são carregados dentro da GPU. Além dos registradores compartilhados, as placas graficas possuem uma memória compartilhada, onde podem ser feita as trocas de informações entre os kernels, e acessíveis pelo sistema operacional de execução para carregar informações para a placa grafica e ler os resultados.

4.1.3 FPGA

Para desenvolver um projeto de um circuito digital para um CI (circuito integrado) são necessários semanas ou mesmo meses de esforço das equipes de projeto. Ocorre que esse esforço possui custo de mão de obra e de ferramentas de desenvolvimento. Dessa forma, qualquer projeto de CI (circuito integrado) que possua algum risco ao investidor como erros na especificação ou em qualquer etapa de síntese, não se justificam. Nesse caso, quando requisitos de tamanho, consumo de energia (*low-power*) entre outros, não são fatores determinantes, e quando existe a necessidade de desenvolvimento rápido, justifica então o uso de tecnologias de CI programáveis (VAHID, 2008), como os PLDs e FPGAS.

Os dispositivos lógicos programáveis (PLDs – Programmable Logic Devices) são

constituídos de arranjos de células lógicas com interconexões programáveis eletricamente. Nesse contexto o projetista fica encarregado de configurar as interconexões das células, dispensando a participação do fabricante. Os PLDs podem ter diferentes níveis de densidade lógica e segundo [28], os primeiros dispositivos de baixa densidade desenvolvidos para programar circuitos lógicos foram os PLAs (*Programmable Logic AND*), os quais são constituídos pelo arranjo de dois tipos de portas lógicas programáveis: a porta lógica AND e a porta lógica OR. Um segundo tipo são os PALs (*Programmable Array Logic*), diferentemente dos PLAs não possuem a porta lógica OR programável, mas sim fixa. Os CPLDs (*Complex Programmable Logic Devices*) possuem de baixa a média densidade lógica e utilizam uma estrutura de interconexão contínua nas quais os atrasos são não-cumulativos. Dessa forma a sincronização (timing) é fixa e previsível e a compilação do projeto é rápida. Ainda assim existem os FPGAs que possuem uma densidade lógica de média a alta, estruturado com interconexão segmentada, cujo resultados são atrasos acumulativos, timing variável no roteamento e a não previsibilidade e o tempo de compilação mais lento.

Segundo Vahid (2008) a tecnologia mais consolidada de IC programável é o FPGA ou Arranjo ou matriz de portas programável em campo. Constituída basicamente de blocos de entrada e saída (I/O blocks), tabelas de consulta (tabelas lookup ou lookup tables) e matrizes de chaveamento (switch matrices). Para Varid (2008) FPGAs fundamentalmente implementam lógica combinacional utilizando memórias, ou seja, utilizando as tabelas lookup. Para implementar circuitos sequências, em cada saída da tabela lookup possui um flip-flop. Assim a cada ciclo de clock, o flip-flop é carregado com o valor correspondente da tabela lookup. Utilizando as matrizes de chaveamento é possível executar a programação das conexões entre as tabelas de lookup, de forma a customizar as conexões entre as memórias. Finalmente, existem os blocos de entradas e saídas responsáveis pelos interfaceamento entre as saídas das tabelas lookup e as

entradas e saídas do FPGA. Constituído de *buffers* e de uma circuitaria de chaveamento, que funcionam como pinos bidirecionais entrada e saída do FPGA.

Historicamente a pioneira na criação da tecnologia de dispositivos lógicos programáveis do tipo FPGA foi a XILINX, em 1984. Em seguida, outra empresa, a ALTERA foi pioneira na criação e desenvolvimento de dispositivos do tipo CPLD – *Complex Programmable Logic Device* ou Dispositivo Lógico Programável Complexo em 1983, suprindo a demanda por dispositivos não voláteis para a indústria de semicondutores.

Comercialmente os FPGAs possuem um custo maior que os ICs produzidos em escala, contudo são em uma boa opção quando a necessidade é de um projeto customizado, com poucas unidades, baixo Tempo para Mercado (*time-to-market)*, rápido Tempo de prototipação (*time-to-prototype)* e exatidão, entre outros fatores.

5 Utilização de FPGA em Ambiente cloud (Proposta)

Para atender ao conceito de escalabilidade e independência dos elementos foi adotado a proposta de criação de um SOC (System On Chip), onde todos os elementos computacionais necessários para a execução do sistema estão presentes dentro de um único chip. Foi criado dentro do chip FPGA um processador chamado NIOS-II fornecido pelo fabricante de FPGAs Altera, bem como um barramento chamado Avalon e memória externa ao chip, dando assim suporte a execução de um sistema operacional dentro do próprio chip. Desta maneira o sistema operacional provê um ambiente de alto nível, com suporte à rede, execução de programas locais, e controle de processos.

Uma possível utilização deste ambiente criado é a ideia de reprogramar dinamicamente o FPGA buscando uma nova imagem de sistema operacional com as funções de aceleração embutidas dentro desta, através de um sistema de arquivos em rede como NFS ou CODA. O Cliente de cloud acessaria as APIs de programação e

instruções do FPGA e os núcleos do FPGA realizariam os cálculos. Conforme a quantidade de recurso necessário, o cliente solicitaria uma quantidade de nós de processamento desejado (desta forma o FPGA estaria inserido no cloud).

Foi criado uma imagem para testes do ambiente montando os recursos básicos de um computador, através do sistema operacional colaborativo NIOS-II, que é baseado em Linux como sistema operacional e as ferramentas busybox e biblioteca uClibc.

O hardware utilizado para o experimento (Figura 7) é uma placa educacional DE2 [29] que possui um FPGA Cyclone II EP2C35F672C6 [30]. Este modelo possui 33.216 elementos lógicos.

Figura 7: Placa DE2

5.1 Implementação:

A implementação é composta de quatro etapas, a primeira é a preparação do ambiente de desenvolvimento, a segunda é a criação da imagem de sistema de hardware, a terceira etapa é a criação da imagem de sistema operacional para ser executado sobre o hardware criado, e a quarta etapa é o teste do sistema operacional rodando sobre o hardware com interfaces de acesso ao ambiente.

5.1.1 Preparação do Ambiente de desenvolvimento:

Esta implementação foi realizada em um computador com arquitetura 64 bits e um sistema operacional Linux Fedora 14 de 64 bits, mas nada impede que seja utilizado uma arquitetura e sistema operacional de 32 bits.

As ferramentas utilizadas para se realizar o projeto do hardware são fornecidos pela Altera, e podem ser baixadas do site http://www.altera.com/. São elas Quartus II web edition versão 10.1 sp1, Nios II EDS 10.1 sp1, que foram instaladas no diretório “/mnt/projetos/altera/10.1”. Após a instalação é necessário a configuração de variáveis de ambiente para definir os caminhos para aplicativos executáveis e outras variáveis utilizadas pelos programas. Para isto foi criado o script shell env_nios com todas as definições e estes podem ser carregados com o comando source (source env_nios) Texto 1.

```
#!/bin/bash
## Altera
AROOT="/mnt/projetos/altera/10.1"
export AROOT
QUARTUS_ROOTDIR="${AROOT}/quartus"
export QUARTUS_ROOTDIR
QUARTUS="${AROOT}/quartus"
export QUARTUS
NIOS2EDS="${AROOT}/nios2eds"
export NIOS2EDS
MODELSIM_ROOT="${AROOT}/modelsim_ase"
export MODELSIM_ROOT
SOPC_KIT_NIOS2="${AROOT}/nios2eds"
export SOPC_KIT_NIOS2
# Caminhos de executáveis
PATH="${QUARTUS_ROOTDIR}/bin:${QUARTUS_ROOTDIR}/linux64:${QUARTUS_ROOTDIR}/sopc_builder/bin:${NIOS2EDS}/bin:${MODELSIM_ROOT}/bin:${NIOS2EDS}/bin/nios2-gnutools/H-i686-pc-linux-gnu/bin:/usr/bin:${PATH}"
## Make
PATH="/usr/local/bin:${PATH}"
## GCC
PATH="${PATH}:/mnt/projetos/altera/gcc/bin"
## LLVM-GCC (legup)
#PATH="/mnt/projetos/altera/legup/llvm-gcc4.2-2.7-x86_64-linux/bin:${PATH}"
PATH="/mnt/projetos/altera/legup/llvm-gcc-4.2-2.7-i686-linux/bin:${PATH}"
## LEGUP
PATH="/mnt/projetos/altera/legup/legup-1.0/llvm/Release/bin:${PATH}"
export PATH
# Cross compilação
CROSS_COMPILE="nios2-linux-gnu-"
export CROSS_COMPILE
```

Texto 1: Variáveis de Ambiente

Como as interações com o hardware FPGA no período de desenvolvimento é realizado através de um cabo JTAG é necessário a configuração da aplicação de manipulação de dispositivo usb-Blaster, e para isto utilizamos o mecanismo udev do sistema linux, criando o arquivo “/etc/udev/rules.d/80-usbblaster.rules” contendo apenas a linha de detecção Texto 2 :

```
SUBSYSTEM=="usb", ATTR{idVendor}=="09fb", ATTR{idProduct}=="6001",MODE="0666", PROGRAM="/bin/sh -c 'K=%k; K=$${K#usbdev};
printf /proc/bus/usb/%%03i/%%03i $${K%%%%.*} $${K#*.}'", RUN+="/bin/chmod 0666 %c"
```

Texto 2: Arquivo UDEV

Para o acesso ao dispositivo jtag o aplicativo “/mnt/projetos/altera/10.1/quartus/bin/jtagd” deve estar em execução para intermediar os aplicativos e os dispositivos jtag que possam estar disponíveis no sistema.

5.1.2 Criação da Imagem de Hardware:

Para a criação da imagem de hardware a ser carregada no FPGA, foi utilizado como base o sistema de referencia DE2_NET fornecido pela Altera, que já possui um core IP

Nios II, com o mapeamento de memória externa já feito, bem como o barramento Avalon.

A manipulação do sistema foi feito utilizando o aplicativo Quartus e as referencias de endereçamento para a placa de desenvolvimento DE2 foi consultando o manual desta placa.

Para a definição dos macro componentes utilizamos a ferramenta SOPC Builder com a qual podemos ter uma visão de altíssimo nível do hardware com as criações de entradas de core IPs, endereçamento de memória e ligações de ativação e interrupções Figura 8.

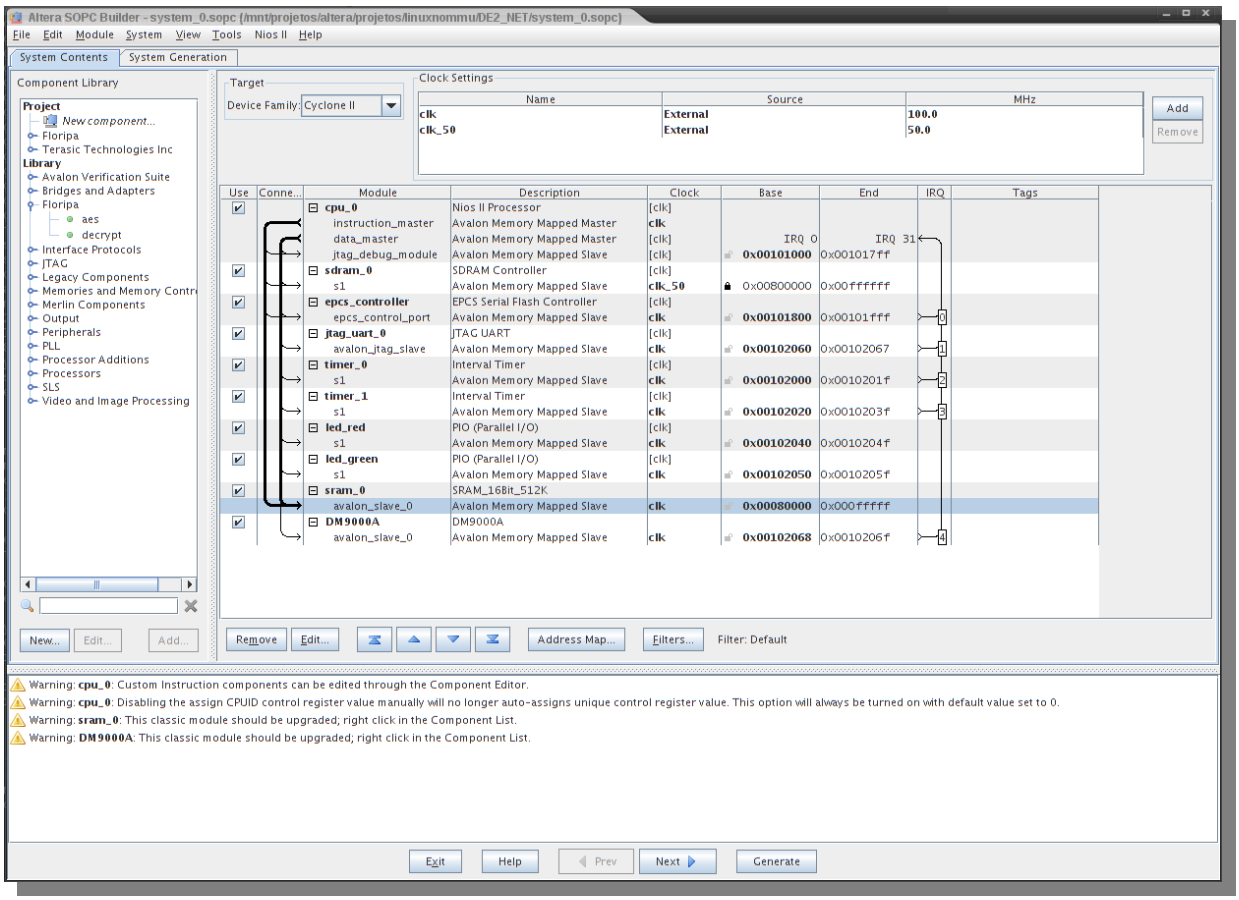


Figura 8: SOC com SOPC Builder

Para facilitar e interligação de componentes do sistema foi utilizado a aplicação RTL Viewer, que tem como objetivo fornecer uma visão no modelo caixa cinza e caixa preta de todo o sistema eletrônico do projeto. No caso deste projeto temos na Figura 9 uma visão de todo o circuito eletrônico.

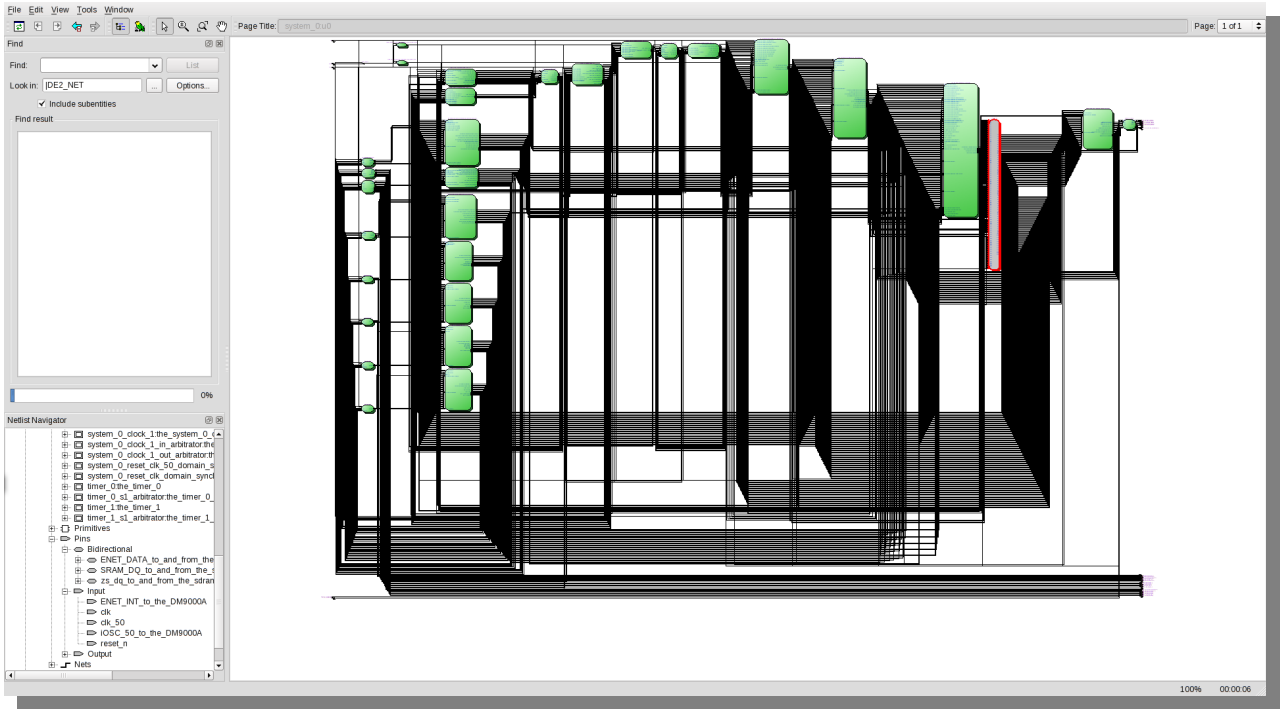


Figura 9: RTL do SOC

Na figura 10 a visão de um dos elementos (um dos componentes do sistema de rede DM9000) em ampliação.

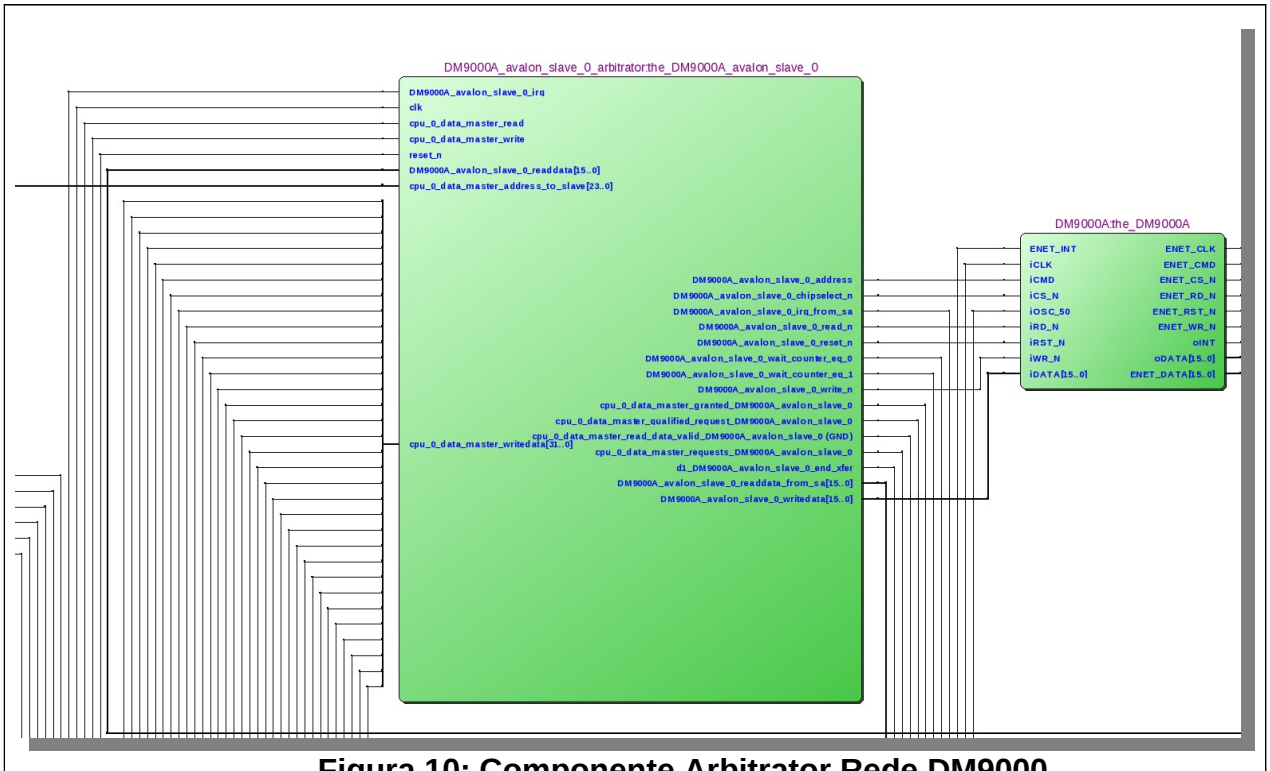


Figura 10: Componente Arbitrator Rede DM9000

E finalmente na figura 11 uma visão interna dos elementos lógicos presentes dentro do componente da rede DM9000.

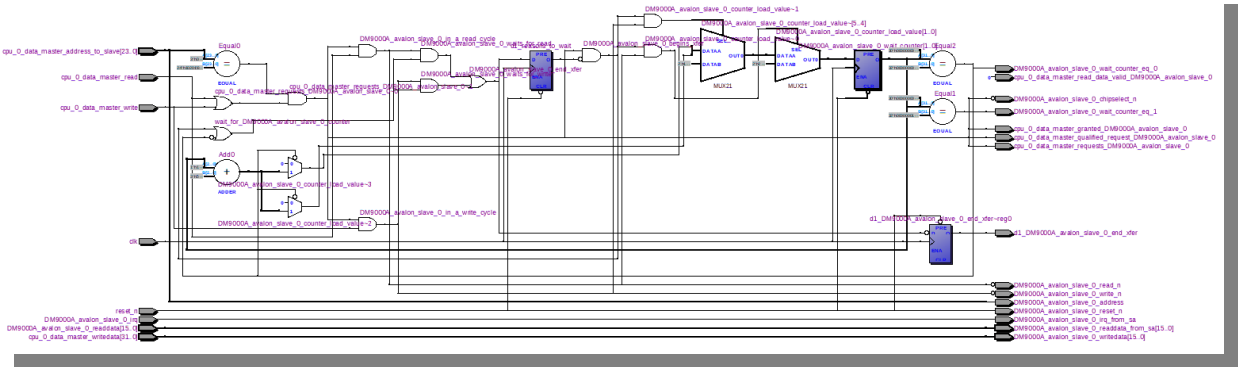


Ilustração 11: Elementos Lógicos do Arbitrator DM9000

A compilação do hardware gera o arquivo com a imagem (.sof) a ser configurada dentro do FPGA através do comando nios2-configure-sof <arquivo.sof>.

5.1.3 Criação da Imagem do sistema operacional:

O sistema operacional utilizadas é o linux sem suporte à mmu baseado na distribuição embarcada uClinux-dist, com kernel linux com suporte ao processador nios.

A versão mais comum desta distribuição é a que foi distribuída em formato de código fonte em um arquivo compactado tar.gz, criado no dia 29/09/2009 pela equipe de desenvolvimento nios2-linux, mas para este projeto foi utilizado as versões de desenvolvimento direto dos repositórios de controle de versão git do projeto acessíveis em http://sopc.et.ntust.edu.tw/git/. Os principais projetos clonados dos repositórios e utilizados são:

- linux-2.6: Código fonte do kernel Linux com modificações Nios2. A versão utilizada foi a v2.6.30.
- uClinux-dist: A *uClinux userspace libraries* e aplicativos.
- binutils, gcc3, elf2flt, insight: *gnu tools source*.

- uClibc: *userspace libraries*, versão menor que a glibc.
- u-boot: boot loader e monitor

Para iniciar o processo de configuração e compilação foi selecionado o *branch* test-nios2 do repositório uClinux-dist que é o ramo sem suporte à mmu.

5.1.3.1 Gerando a imagem:

Como os arquivos makefile de configuração de compilação da distribuição são baseados em uma versão antiga de definição de variáveis, é necessário utilizar uma versão mais antiga do programa GNU Make: a versão 3.81 ou anterior.

Para gerar uma imagem de linux sem MMU são necessários os seguintes comandos:

1. cd uClinux-dist
2. make menuconfig, após realizar este comando uma tela idêntica a tela abaixo irá aparecer, nesta tela a configuração do seu kernel deve ser feita, para gerar nossa primeira imagem vamos manter tudo padrão:

```
Vendor/Product Selection --->           # select
        --- Select the Vendor you wish to target
        Vendor (Altera) --->            # should have default to Altera
        --- Select the Product you wish to target
        Altera Products (nios2)  --->   # should have defaulted to nios2

Kernel/Library/Defaults Selection --->  # select
        --- Kernel is linux-2.6.x
    Libc Version (None) --->            # should default to None - very important.
        [*] Default all settings (lose changes) # select
        [ ] Customize Kernel Settings
        [ ] Customize Vendor/User Settings
        [ ] Update Default Vendor Settings
```

Após esta configuração estar realizada, <exit> <exit> <yes>.

É necessário referenciar seu projeto de hardware para que a “colagem” sistema operacional/hardware seja realizada.

Para isto basta realizar o seguinte comando:

```
make vendor_hwselect SYSPTF=/mnt/projetos/altera/projetos/linuxnommu/DE2_NET/system_0.ptf
```

Para compilar basta executar o comando:

```
make
```

Com isto a imagem será gerada no diretório imagens, o padrão da imagem é images/zImage

5.1.4 Sistema operacional executando sobre SOC:

Tendo em mãos as imagens de hardware e software gerados nas etapas anteriores vamos iniciar os testes de carga e ambiente necessários para o sistema ser capaz de interagir no ambiente de cloud computing ou hpc.

A imagem de hardware deve ser configurada no FPGA com o comando: “nios2-configure-sof DE2_NET.sof”. Todas as customizações e adaptações de hardware entram em um estado de pronto e executando logo após esta execução.

Com o hardware pronto devemos carregar o sistema operacional gerado na etapa anterior através do comando “nios2-download -g images/zImage” que pelo parametro -g indica ao programa que deve iniciar a execução imediatamente ao fim do download (envio para a placa). E iniciamos o monitoramento do console principal do sistema operacional em execução pelo cabo usb-Blaster, usando o comando “nios2-terminal”.

Para a interação entre o sistema operacional linux executando dentro do chip FPGA, foi adicionado as configurações de rede, servidor http, servidor telnet e montagem de sistema de arquivos em rede NFS. A configuração de rede e montagem NSF foi selecionada através do suporte do kernel à NFS, seleção de aplicativos na distribuição uClinux-dist, como o portmap, mount com suporte à NFS, e servidor boa-http. Para a inicialização do sistema foi editado o arquivo uClinux-dist/vendors/Altera/nios2/rc ficando

como o Texto 3:

```
##
hostname floripa-nios
# montagem basica
mount -t proc proc /proc -o noexec,nosuid,nodev
mount -t sysfs sysfs /sys -o noexec,nosuid,nodev
mount -t devpts devpts /dev/pts -o noexec,nosuid
#mount -t usbfs none /proc/bus/usb
mkdir /var/tmp
mkdir /var/log
mkdir /var/run
mkdir /var/lock
mkdir /var/empty
# Config de rede
ifconfig lo 127.0.0.1
route add -net 127.0.0.0 netmask 255.0.0.0 lo
ifconfig eth0 hw ether 00:07:ed:00:00:01
ifconfig eth0 10.0.0.25 netmask 255.255.255.0
route add default 10.0.0.1
#dhcpcd -p -a eth0 &
## montagem de nfs
#portmap &
mkdir /nfs
mount -t nfs -n -o nolock,rsize=1024,wsize=1024 10.0.0.20:/mnt/projetos/altera /nfs
cat /etc/motd
```

Texto 3: Arquivo de inicialização rc

O acesso ao nível mais baixo do FPGA é demonstrado através de um programa cgi rodando dentro do servidor boa. O trecho de código do Texto 4 demonstra como utilizar o hardware:

```
#include <stdio.h>
#include "/mnt/projetos/altera/apptest/DE2_NET.h"
#include "cgivars.h"
#include "htmllib.h"
#include <stdlib.h>
#include <string.h>
#define led_red (char*) LED_RED_BASE
#define led_green (char*) LED_GREEN_BASE

void led(const char * nome, const char * ligado)
{
#if DEBUG
   printf("<li> led(%s,%s)\n",nome,ligado);
#endif
   if (strcmp(nome,"red") == 0)
   {
      if (strcmp(ligado,"on") == 0)
      {
         *led_red=0x01;
      }
      else
      {
         *led_red=0;
      }
   }
   if (strcmp(nome,"green") == 0)
   {
      if (strcmp(ligado,"on") == 0)
      {
         *led_green=0x01;
      }
      else
      {
         *led_green=0;
      }
   }
}
```

Texto 4: Fragmento de Código com Acesso ao Hardware

As definições dos endereços são exportados no arquivo .h usando o comando “sopc-create-header-files --single /mnt/projetos/altera/apptest/DE2_NET.h” e analizando este arquivo podemos observar que são as mesmas referencias de endereços de memórias definidas no SOPC Builder na Figura 8. As posições de endereço base, IRQ e outras definições podem ser observadas no fragmento de código retirado de DE2_NET.h Texto 5:

```
/*
 * Macros for device 'led_green', class 'altera_avalon_pio'
 * The macros are prefixed with 'LED_GREEN_'.
 * The prefix is the slave descriptor.
 */
#define LED_GREEN_COMPONENT_TYPE altera_avalon_pio
#define LED_GREEN_COMPONENT_NAME led_green
#define LED_GREEN_BASE 0x102050
#define LED_GREEN_SPAN 16
#define LED_GREEN_END 0x10205f
#define LED_GREEN_DO_TEST_BENCH_WIRING 0
#define LED_GREEN_DRIVEN_SIM_VALUE 0x0
#define LED_GREEN_HAS_TRI 0
#define LED_GREEN_HAS_OUT 1
#define LED_GREEN_HAS_IN 0
#define LED_GREEN_CAPTURE 0
#define LED_GREEN_BIT_CLEARING_EDGE_REGISTER 0
#define LED_GREEN_BIT_MODIFYING_OUTPUT_REGISTER 0
#define LED_GREEN_DATA_WIDTH 9
#define LED_GREEN_RESET_VALUE 0x0
#define LED_GREEN_EDGE_TYPE "NONE"
#define LED_GREEN_IRQ_TYPE "NONE"
#define LED_GREEN_FREQ 100000000
```

Texto 5: Fragmento DE2_NET.h - LED verde

5.2 Resultado da execução do sistema dentro do chip FPGA:

```
  [helber@note images]$
  Uncompressing Linux... Ok, booting the kernel.
  [    0.000000] Linux version 2.6.30-floripa-00395-gd978694 (helber@note.helbermg.dyndns.org) (gcc version
3.4.6) #111 PREEMPT Mon May 2 02:01:52 BRT 2011
  [    0.000000] uClinux/Nios II
  [    0.000000] Built 1 zonelists in Zone order, mobility grouping off.  Total pages: 2032
  [    0.000000] Kernel command line:
  [    0.000000] NR_IRQS:32
  [    0.000000] PID hash table entries: 32 (order: 5, 128 bytes)
  [    0.000000] Dentry cache hash table entries: 1024 (order: 0, 4096 bytes)
  [    0.000000] Inode-cache hash table entries: 1024 (order: 0, 4096 bytes)
  [    0.010000] Memory available: 5496k/2414k RAM, 0k/0k ROM (1654k kernel code, 759k data)
  [    0.050000] Calibrating delay loop... 44.95 BogoMIPS (lpj=224768)
  [    0.240000] Mount-cache hash table entries: 512
  [    0.390000] net_namespace: 264 bytes
  [    0.420000] NET: Registered protocol family 16
  [    0.430000] init_BSP(): registering device resources
  [    0.560000] bio: create slab <bio-0> at 0
  [    0.680000] NET: Registered protocol family 2
  [    0.690000] IP route cache hash table entries: 1024 (order: 0, 4096 bytes)
  [    0.720000] TCP established hash table entries: 512 (order: 0, 4096 bytes)
  [    0.720000] TCP bind hash table entries: 512 (order: -1, 2048 bytes)
  [    0.720000] TCP: Hash tables configured (established 512 bind 512)
  [    0.720000] TCP reno registered
  [    0.740000] NET: Registered protocol family 1
  [    6.270000] io scheduler noop registered
  [    6.270000] io scheduler deadline registered (default)
  [    6.400000] ttyJ0 at MMIO 0x102060 (irq = 1) is a Altera JTAG UART
  [    6.400000] console [ttyJ0] enabled
  [    6.420000] dm9000 Ethernet Driver, V1.31
  [    6.550000] dm9000 dm9000.0: eth%d: Invalid ethernet MAC address. Please set using ifconfig
  [    6.570000] eth0 (dm9000): not using net_device_ops yet
  [    6.590000] eth0: dm9000a at 80102068,8010206c IRQ 4 MAC: 00:00:00:00:00:00 (chip)
  [    6.600000] TCP cubic registered
  [    6.600000] NET: Registered protocol family 17
  [    6.630000] RPC: Registered udp transport module.
  [    6.630000] RPC: Registered tcp transport module.
  [    6.670000] Freeing unused kernel memory: 532k freed (0x9d6000 - 0xa5a000)
  Shell invoked to run file: /etc/rc
  Command: hostname floripa-nios
  Command: mount -t proc proc /proc -o noexec,nosuid,nodev
  Command: mount -t sysfs sysfs /sys -o noexec,nosuid,nodev
  Command: mount -t devpts devpts /dev/pts -o noexec,nosuid
  Command: mkdir /var/tmp
  Command: mkdir /var/log
  Command: mkdir /var/run
  Command: mkdir /var/lock
  Command: mkdir /var/empty
  Command: ifconfig lo 127.0.0.1
  Command: route add -net 127.0.0.0 netmask 255.0.0.0 lo
  Command: ifconfig eth0 hw ether 00:07:ed:00:00:01
  Command: ifconfig eth0 10.0.0.25 netmask 255.255.255.0
  [   12.660000] eth0: link down
  Command: route add default 10.0.0.1
  [   13.150000] eth0: link up, 100Mbps, full-duplex, lpa 0x41E1
  route: SIOCADDRT: No such device
  Command: mkdir /nfs
  Command: mount -t nfs -n -o nolock,rsize=1024,wsize=1024 10.0.0.20:/mnt/projetos/altera /nfs
  Command: cat /etc/motd
  Welcome to
           /  __| ||_|
      _    _| |  | | _ ____  _   _  _  _
     | | | | |  | || |  _ \| | |\ \/ /
     | |_| | |__| || | | | | |_| |/    \
     |  ___\____|_||_|_| |_|\____|\_/\_/
     | |
     |_|
  For further information check:
  http://www.uclinux.org/
  Sistema configurado para o TCC UFSC
  ---------- Helber Maciel Guerra ------------
  ------------ DHW - Engenharia --------------
  Sash command shell (version 1.1.1)
  />
```

6 Conclusões e Trabalhos Futuros

6.1 Conclusões

A conversão de funções específicas de uma linguagem de alto nível como C, para Verilog ou SystemC, realizadas com o uso dos compiladores llvm e altera C2verilog, auxiliam em muito a construção da função em hardware, bem como as funções de acesso à execução (funções wrapper). Como as funções podem ser alocadas diversas vezes dentro de um mesmo chip FPGA, e estas são executadas simultaneamente dentro de um mesmo ciclo de relógio (*clock)*, isto torna fornece um paralelismo real dificilmente alcançado em processamento convencional, podendo ser de 10 a 500 vezes mais rápida que a execução em *software* dentro de um processador convencional.

A utilização das funções aceleradas requer funcionalidades auxiliares de carga e transferência de dados, bem como outras funções auxiliares, que consomem uma grande quantidades de ciclo de processadores, podendo ser em muitos casos mais lento que o processador convencional, para uma mesma frequência de operação. Isto se agrava, comparando o processador convencional operando a uma frequência de 2000 Mhz com um FPGA com clock de operação de 50 Mhz. Neste caso o resultado chega e ser muito pior que o processamento multi-core.

6.2 Trabalhos Futuros

Este trabalho foi executado utilizando um processador NIOSII sem MMU, e a carga do sistema feita manualmente. Como trabalho futuro implementar a CPU NIOSII com MMU e a reprogramação dinâmica do sistema.

Para uma maior agilidade na troca de função em hardware, implementar a reprogramação parcial do chip FPGA, com o sistema em execução, que com a tecnologia

atual é muito difícil de ser realizado.

Referência Bibliográfica

Bibliografia

1: , Cloud Computing and Grid Computing 360-Degree Compared,

2: J. Martin, There's Gold in them thar Networks! or Searching for, 1991

3: , A Break in the Clouds: Towards a Cloud Definition, 2009

4: Fernando Belfort, Cloud computing estará na maioria das empresas até 2012, diz estudo, 2010

5: Viviane Souza, Victor Medeiros, Derci Lima e Manoel Lima, Uma abordagem de alto desempenho para multiplicação de matrizes densas em sistemas reconfiguráveis, 2009

6: , NASA - Openstack,

7: , Parallel Computing, 2005

8: Dantas, Mario, Computação Distribuída de Alto Desempenho: Redes, Clusters e GridsComputacionais, 2005

9: , OpenSSI. Novembro 2007, 2007

10: , BEOWULF Cluster. Novembro 2009, 2009

11: DB Skillicorn, Motivating Computational Grids - Proceedings of the Second IEEE/ACM International Symposium on Cluster Computing and the Grid, 2002

12: Berstis, Viktors, Fundamentals of Grid ComputingFundamentals of Grid Computing, 2002

13: Foster, Ian, What is the Grid? A Three Point Checklist, 2002

14: , BOINC (Berkeley Open Infrastructure for Network Computing),

15: , OGSA (Open Grid Services Architecture),

16: , Open Cloud Manifesto, 2009

17: , Cloud Computing Deep Dive Report, 2009

18: , HDFS - Hadoop Filesystem,

19: Leavitt, N, Will NoSQL Databases Live Up to Their Promise?, 2010

20: , MongoDB,

21: , Memcached,

22: , CouchDB,

23: OLIVEIRA, Leonardo Eloy, Estado da arte de banco de dados orientados a documento, 2009

24: , Bigtable: A Distributed Storage System for Structured Data, 2006

25: , Cassandra,

26: Amesfoort , Alexander S. Van, Evaluating Multi-Core Platforms for HPC Data-Intensive Kernels, 2009

27: , OPENCL - Site,

28: S. Brown and J. Rose, FPGA and CPLD Architectures: A Tutorial - Design and Test of Computers, 1996

Anexos