Conteúdo

1 Motivação **1**

2 Estado-da-arte **2**

2.1 Arquiteturas Reconfiguráveis 2

2.2 Exemplos de Arquiteturas Reconfiguráveis 4

2.2.1 PRISM - Processor Reconfiguration Through Instruction-Set Metamorphosis 4

2.2.2 PRISM-I 5

2.2.3 SPLASH 5

2.2.4 PAM - Programmable Active Memories 6

2.2.5 DISC - Dynamic Intruction Set Computer 7

2.2.6 TRANSMOGRIFIER - 2 7

2.3 Field Programable Gate Arrays - FPGAs 9

2.4 Blocos Lógicos 11

2.5 Interconexões Programáveis 12

2.6 Blocos de entrada/saída 12

2.7 *Cores* 13

2.7.1 Tipos de *cores* disponíveis 13

2.7.2 Metodologia para projetos baseados em *cores* 14

3 Objetivos **15**

3.1 Objetivos Gerais 15

3.2 Ojetivos Específicos 15

4 Cronograma e recursos necessários **16**

4.1 Cronograma 16

4.2 Recursos necessários 16

Bibliografia **16**

Capítulo 1

Motivação

Implementar um módulo de *hardware* (*core*) que minimize as limitações na transferência de dados na interação *hardware/software* em arquiteturas reconfiguráveis. Os *cores* atualmente existentes são protegidos pelas leis da Propriedade Intelectual e só podem ser adquiridos com um alto investimento. Mesmo assim, o *core* é vendido em forma de "caixa preta", não permitindo acesso ao seu código fonte. Projetos baseados em *core* apresentam benefícios como: redução do tempo de projeto, redução de riscos de projeto, impõe maior desempenho em projetos com alto nível de integração, flexibilidade, além da redução do *time-to-market*.

Capítulo 2

Estado-da-arte

Este trabalho insere-se nas seguintes áreas:

- Arquiteturas reconfiguráveis;
- *Field Programable Gate Arrays* - FPGAs;
- *Cores*.

2.1 Arquiteturas Reconfiguráveis

Arquiteturas reconfiguráveis representam uma área de pesquisa recente, embora Gerald Estrin, da Universidade da Califórnia, na década de 60, já houvesse proposto esta abordagem e implementado suas idéias [8]. Um forte impulso à área ocorreu na década de 80, com o advento dos dispositivos *Field Programable Gate Arrays* - FPGAs. Desde então, estes dispositivos tem sido amplamente utilizados em arquiteturas reconfiguráveis que também encontradas na literatura técnica com denominações tais como: *hardware* reconfigurável, *hardware* programável, sistema computacional reprogramável, plataforma reconfigurável e arquitetura computacional configurável. A primeira arquitetura reconfigurável foi proposta, projetada e implementada por Gerald Estrin, nos primórdios de 1960. Estrin propôs um "computador com estrutura fixa e variável", no qual o *hardware*

era dedicado tanto a uma (inflexível) abstração de um processador programável quanto a um (flexível) componente que implementava a lógica digital. Esta arquitetura básica, que dá suporte a *hardware* programável e *software*, é o núcleo de muitos sistemas computacionais configuráveis subsequentes. Infelizmente, os conceitos arquiteturais de Estrin estavam bem à frente da tecnologia à disposição para a sua época, o que só lhe permitiu uma prototipação parcial de sua idéia. Muitos dos conceitos aplicados atualmente em computação configurável tem como base o trabalho de Estrin.

Ao se projetar uma arquitetura reconfigurável, 3 parâmetros devem ser considerados:

- Granularidade do *hardware* programável:

  O desenvolvimento de arquiteturas reconfiguráves utiliza, normalmente, ferramentas de *Computer Aidded Design* - CAD (Projeto com Auxílio de Computador), que foram desenvolvidas para o projeto de *Application-Specific Integrated Circuits* - ASICs (Circuitos Integrados de Aplicação Específica). Para aumentar o nível de abstração, vários sistemas computacionais configuráveis em desenvolvimento limitam o *hardware* programável à interconexão, e no lugar de portas lógicas e *flip-flops*, utilizam componentes como *Arithmetic Logic Units* - ALUs (Unidades Lógicas e Aritméticas) ou multiplicadores [8].

- Proximidade do *hardware* programável em relação ao microprocessador:

  A primeira geração de sistemas configuráveis tipicamente utilizavam barramentos periféricos, com o Sparc Sbus, a fim de prover uma estrutura de coprocessador. Porém, alguns pesquisadores propõem que o *hardware* programável deve ficar o mais próximo possível do processador e, se possível, até nas linhas de dados alimentadas pelos registradores do processador [8]. Trabalhos como o apresentado em [9], propõem a implementação do processador-FPGA-DRAM em um mesmo circuito integrado afim de minimizar os tempos de comunicação entre *hardware*/*software*.

- Capacidade de reconfiguração:

  Este parâmetro envolve restrições tais como: relação entre *hardware* programável

e capacidade de memória, largura de banda de comunicação do processador, quantidade de *hardware* programável requerido e forma de reconfiguração (estática ou dinâmica).

2.2 Exemplos de Arquiteturas Reconfiguráveis

Ao final do anos 80 e durante os anos 90, várias arquiteturas reconfiguráveis foram propostas e desenvolvidas. Entre os exemplos mais significativos, apresentados na seqência deste Capítulo, estão:

- PRISM (PRISM-I)[1];
- SPLASH [4];
- PAM (DECPeRLe-0 e DECPeRLe-1) [13];
- DISC [14];
- TRANSMOGRIFIER-2 [5];

2.2.1 PRISM - Processor Reconfiguration Through Instruction-Set Metamorphosis

Um compilador de configuração é aquele compilador responsável por receber um programa como entrada e produzir tanto uma imagem de *hardware* como uma imagem de *software* como saída. A imagem de *hardware* consiste de especificações para programar uma plataforma reconfigurável. A imagem de *software*, similar em função a um compilador convencional, consiste em um código de máquina pronto para execução em um processador [1]. Esta arquitetura assume que a plataforma do processador consiste em recurso reconfigurável intimamente integrado a um núcleo processador central. O núcleo processador executa a imagem de *software* e coordena a execução de operações sintetizadas. O dispositivo reconfigurável sempre tem um retardo de propagação muito menor, bem como uma densidade de portas lógicas menor que um ASIC compatível.

2.2.2 PRISM-I

O PRISM-I foi desenvolvido para demonstrar a viabilidade de se incorporar uma parte adaptativa em máquinas de propósito geral. Seu compilador recebe em sua entrada um programa em linguagem C, produzindo uma imagem de *hardware* e uma imagem de *software* ao longo de seu fluxo de dados intermediário. O primeiro passo no processo de compilação é separar as porções do código fonte em duas partes: as que são candidatas a serem implementadas em *hardware* e as que permanecerão em *software*. Logo após, o PRISM-1 transforma os candidatos a *hardware* em descrições físicas para serem implementadas na plataforma reconfigurável, criando assim uma imagem de *hardware*. Tendo sido gerada com sucesso a imagem de *hardware*, o restante do programa fonte que não é sintetizável em *hardware*, produz um novo código C. A especificação resultante é então compilada produzindo, neste passo final, um arquivo imagem de *software*. Quando a imagem de *software* é executada, a imagem de *hardware* é carregada automaticamente nas FPGAs apropriadas [1].

2.2.3 SPLASH

O SPLASH, desenvolvido pelo *Supercomputing Research Center* - SRC em 1988, é um arranjo lógico linear composto por duas placas conectadas em dois barramentos de uma estação de trabalho Sun. Uma das placas abriga o SPLASH que se comunica com a Sun através de uma interface VME. Uma interface VSB é empregada para conectar os blocos de memória e o arranjo linear de 32 estágios. A outra placa é composta de memórias que se comunicam com os barramentos VME e VSB, configurando um estágio de memória. O barramento VME controla o SPLASH através de um *host computer* (computador hospedeiro) Sun. Um FPGA Xilinx XC3090 é programado via VME toda vez que o SPLASH é inicializado para então controlar a interface. Esta técnica permite que a interface seja facilmente modificada caso necessário. Com o envio de comandos através da interfaceé possível programar o SPLASH, ler o conteúdo dos registradores, bem como controlar sua operação. A principalárea de aplicação do SPLASH tem sido

a Biologia Computacional, apresentando um excelente desempenho em tarefas como a comparação e emparelhamento de seqüências de DNA. Uma segunda versão do SPLASH, o SPLASH 2, foi desenvolvida a partir de 1992 com FPGAs de maior capacidade, as XILINX XC 4010 (10.000 portas lógicas) [2].

2.2.4 PAM - Programmable Active Memories

O projeto PAM, foi desenvolvido pelo *Digital Equipment Corporation's Paris Research Laboratory* - DEC-PRL, em conjunto com o *Institut National de Recherche en Informatique et en Automatique* - INRIA, com o propósito de implementar uma máquina virtual que pode ser reconfigurada dinamicamente. O PAM é conectado, através de ligações de entradas e saídas a um hospedeiro (host computer). A função do host é descarregar o arquivo de configuração binário para as FPGAs que compõe o PAM, que após a configuração comporta-se como um ASIC. Ele pode operar em modo *stand-alone*, conectado a um dispositivo externo através das conexões de entrada e saída. Pode operar também como um coprocessador sob o comando de um host, a fim de melhorar o desempenho em operaões computacionais mais complexas. Se estiver conectado a um *host* e a dispositivos externos concomitantemente, permite a execução de tarefas como por exemplo: processador de áudio ou vídeo. Isto justifica o seu nome, já que pode estar ligado a um barramento de alta velocidade de um computador, como qualquer módulo de memória *Random Acces Memory* - RAM. O *host* pode escrever e ler dados do PAM. A diferença entre o PAM e as RAMs é a de que o PAM tem a capacidade de processar dados entre as rotinas de escrita e leitura, caracterizando-o como uma memória ativa. O primeiro protótipo do PAM, denominado DECPeRLe-0 e construído em 1987 pelo INRIA, era composto de uma matriz de 25 FPGAs Xilinx XC3020. Cinco anos após o DEC-PRL implementa o DECPeRLe-1 e distribui cópias para vários centros de pesquisa científica do mundo.

2.2.5 DISC - Dynamic Intruction Set Computer

O DISC (Computador com Conjunto Dinâmico de Instruções) implementa cada instrução do seu conjunto de instruções em um módulo de circuito reconfigurável independente. As instruções são implementadas individualmente em *hardware* reconfigurável na medida em que o programa de aplicação necessita, semelhante a um sistema de paginação em memória virtual. As limitações do *hardware* disponível para implementar as instruções são eliminadas com a retirada dos módulos, em tempo real, de instruções que não estão em uso. Uma aplicação rodando no DISC possui código fonte, indicador ordenador de instruções e uma biblioteca com módulos de circuitos com instruções de aplicações específicas [14]. Para que a paginação dinâmica de instruções seja implementada, o DISC utiliza uma técnica de reconfiguração parcial de FPGAs. A reconfiguração parcial permite que uma parte da FPGA seja modificada enquanto a lógica remanescente opera normalmente. No projeto do DISC foi empregado o FPGA da *National Semiconductor Configurable Logic Array* - CLAy, devido a sua característica de reconfiguração dinâmica. O processador DISC foi implementado em uma placa de circuito impresso confeccionada exclusivamente para o projeto. A placa inclui circuitos de interface de barramento, 2 FPGAs CLAy31 e memória. Um controlador de configuração é implementado no primeiro FPGA a fim de monitorar o processo de execução e requisição de instruções do host. No segundo FPGA fica o DISC propriamente, e o programa de aplicação armazenado em memória acoplada. A placa opera no sistema operacional UNIX controlado pelo host. Um exemplo de aplicação do DISC foi a implementação do afinamento (sharpening) de imagens para reconhecimento de objetos, e para tanto foi criada uma biblioteca de instruções. Nesta aplicação, o DISC alcançou um ganho de até 10 vezes em relação à mesma implantação em *software* em um PC 486 à 66 MHz [2].

2.2.6 TRANSMOGRIFIER - 2

O Transmogrifier-2 (ou TM-2) é um sistema de prototipação rápida de alta capacidade, com altas taxa de clock , suficientemente flexível para nele implementar uma

ampla variedade de sistemas computacionais [5]. Antes do TM-2, o Transmogriphier-1 (TM-1) já servia para prototipação rápida de sistemas computacionais. O TM-1 é composto por 40000 portas lógicas em FPGAs e 128 KB de memória RAM, sendo capaz de implementar pequenos sistemas computacionais. Todos os projetos implementados no TM-1, com sucesso ou não, contribuíram para os objetivos alcançados com o TM-2. Características do TM-2:

- Modularidade e Escalabilidade:
  Entende-se como escalabilidade a capacidade de reconfiguração da arquitetura em diferentes tamanhos de blocos de sistemas computacionais. Já o termo modularidade refere-se à maneira como a qual estes blocos são construídos e a partir de um nmero de módulos idênticos. Cada módulo corresponde a um FPGA agregado a recursos de roteamento que provê sua ligação ao resto do sistema, bem como a outro *hardware* associado, tais como memórias.

- Capacidade de 1.000.000 de portas lógicas:
  O Transmogrifier-2 pode ser reconfigurado para empregar até 1x106 portas lógicas mais memória RAM utilizando FPGAs. O TM-2 utiliza 32 FPGAs 10K50 da Altera, o que excede 1x106 de portas lógicas.

- Alta velocidade:
  O TM-2 consegue trabalhar a uma taxa de clock considerada alta (10 MHz). Levando-se em consideração a arquitetura do TM-2 no momento da implementação de uma nova configuração, taxas mais altas podem ser alcançadas.

- *Clock* de alta qualidade programável pelo usuário:
  Esta qualidade do TM-2 permite habilitações de escrita assíncrona em memórias RAM, as quais devem ser precisamente controladas porém não configuradas a cada ciclo.

- RAM e largura de banda alta:
  Possui alta velocidade de acesso e uma grande largura de banda com as RAMs. O

TM-2 é composto de memórias SRAM e DRAM, bem como uma grande capacidade de I/O (*input/output*).

- Facilidade de uso:
  Através de ferramentas CAD é possível projetar em alto nível e compilar um programa com facilidade, gerando arquivos binários de configuração com um simples comando. Pode inclusive, ser programado via rede de computadores.

- Tempo de programação:
  O tempo de *download* de um arquivo de *bits* para configuração do TM-2 fica próximo a 10 ms.

- Confiabilidade:
  O TM-2 é um sistema robusto, física e eletricamente, sendo capaz de suportar sobrecorrentes que poderiam danificar os componentes e causar falhas.

- Manufatura:
  Construído sobre circuito impresso de tecnologia e padrões comerciais e circuitos integrados de prateleira (*of the shelf*).

Computação gráfica, criptografia e mapeamento de texturas estão entre as aplicaões do TM-2. Um versão avançada do Transmogrifier-2 é constituída de FPGAs Altera 10K100 e 16 placas de circuito impressos, alcançando um tempo de reconfiguração de apenas dezenas de milissegundos [5].

2.3 Field Programable Gate Arrays - FPGAs

O *Field Programable Gate Array* - FPGA, circuito programável lançado no mercado pela Xilinx [15] no início da década de 80, tem revolucionado a forma de projetar circuitos digitais. O FPGA é um circuito que pode ser configurado como diferentes

Application Specific Integrated Circuit - ASICs. Esta tecnologia nos permite projetar, testar e corrigir circuitos dedicados com um custo baixo de prototipação. Os projetistas de computadores enfrentam sempre o desafio de encontrar o balanço correto entre velocidade e generalidade de processamento do seu *hardware*. É possível desenvolver um *chip* que realiza muitas funções diferentes, porém com sacrifício de desempenho, ou *chips* dedicados à aplicações específicas que tem apenas um conjunto limitado de tarefas, estes com uma velocidade de operação muitas vezes superior aos *chips* genéricos. Circuitos ASICs, têm como características ocupação mínima de área de silício, custo elevado de fabricação, rapidez e um menor consumo de potência comparados com processadores programáveis. Um fator importante na escolha entre versatilidade e velocidade é o custo. Um ASIC executa a função para qual foi concebido de uma forma otimizada, porém uma vez desenvolvido o *chip*, alterações na funcionalidade do circuito integrado tornam-se impossíveis. Logo, todo esforço dispendido no seu projeto e implementação deve ser amortizado em um número elevado de unidades. Os FPGAs possuem uma estrutura lógica interna que pode ser modificada toda vez que se fizer necessário [12]. Um FPGA é programado com o uso de chaves eletrônicas programáveis. As propriedades destas chaves, tais como tamanho, resistência de contato e capacitâncias definem os compromissos de desempenho da arquitetura interna do FPGA. Existem também diferentes tecnologias de programação de chaves eletrônicas que são: *Static RAM* - SRAM, anti-fusível, porta flutuante. No caso dos FPGAs Xilinx [15], Plessey [10], Algotronix [7], Concurrent Logic [6] e Toshiba [3], a tecnologia de programação utiliza SRAM. Sendo as SRAM voláteis, o FPGA deve receber seu arquivo de configuração toda vez que o circuito for energizado. Isto requer memória externa e permanente para armazenar os bits de configuração, podendo ser empregadas *Programmable Read-Only Memories* - PROMs, *Erasable* PROM - EPROM , *Electrically* EPROM - EEPROM , ou discos magnéticos. A maior desvantagem das SRAM é a área ocupada. São necessários pelo menos 5 transistores para implementar um célula SRAM e pelo menos mais um transistor para servir de chave programável. Apesar da desvantagem de maior área ocupada e necessidade de memória externa para configuração, esta é a tecnologia que domina hoje o mercado de FPGAs, pois permite reconfiguração de arquiteturas,

tanto estática quanto dinamicamente. Sua estrutura é similar aos *Mask-Programmable Gate Arrays* - MPGAs, consistindo de um arranjo bi-dimensional de blocos lógicos que podem ser interconectados para executarem diferentes tarefas. A maior diferença entre os FPGAs e MPGAs, é que o MPGA tem suas interconexões produzidas através de um padrão de condutores definido pelo projetista, (necessitando ser fabricado em uma *foundry* (fundiçã de siício), enquanto que um FPGA é programado via chaves eletricamente programáveis,que comandam a conexão entre segmentos condutores pré-definidos, tais como nos *Programmable Logic Devices* - PLDs. Os FPGAs podem alcançar níveis de integração muito mais elevados que os PLDs, embora possuam o roteamento de sua arquitetura e a implementação lógica mais complexos [11]. A arquitetura de um FPGA está dividida em 3 partes: blocos lógicos, interconexões programáveis e blocos de entrada/saída.

2.4 Blocos Lógicos

Um bloco lógico de um FPGA podem potencialmente ser tão simples como um transistor ou tão complexo quanto um microprocessador. É tipicamente capaz de implementar várias funções lógicas combinacionais ou sequenciais. Os FPGAs comerciais atuais empregam blocos lógicos que são baseados em um ou mais dos seguintes componentes:

- Par de transistores;
- Portas lógicas tais como NãO-E ou OU-Exclusivo, ambas com duas entradas;
- Multiplexadores;
- *Look-up Tables* - LUTs, tecnologia utilizada, sobretudo, pela empresa Xilinx;
- Estruturas E/OU com alto *fan-in* (número de entradas por porta lógica), tecnologia utilizada prioritariamente pela empresa Altera.

Estes blocos também são chamados de *Configurable Logic Blocks* - CLBs (blocos lógicos configuráveis). Cada CLB pode implementar funções lógicas de **n** variáveis

(onde **n** é o número de entradas), ou como memória de simples ou duplo acesso (ex.: famílias XC4000 Xilinx).

2.5 Interconexões Programáveis

Existem 4 tipos de interconexões programáveis atualmente:

1. SRAM - na qual a chave é um transistor de passagem controlado por um bit de RAM estática;
2. Antifusível - quando eletricamente programado forma caminho com baixa resistência;
3. EPROM - quando a chave é um transistor com porta flutuante, que pode ser desligado ao se injetar uma carga em sua porta.
4. FLASH - quando a chave é baseada em memória Flash.

Em todos os casos, as chaves programáveis ocupam grande área e apresentam resistência e capacitância parasitas maiores comparados a um MPGA. Devido aos resultados e desempenhos alcançados pelos FPGAs atuais, sua densidade é de uma ordem de grandeza maior em relação aos MPGAs fabricados com a mesma tecnologia [11].

2.6 Blocos de entrada/saída

Para conectar o dispositivo configurável com o mundo externo, os FPGAs possuem componentes de entrada/saída chamados I/O Blocks, formados por estruturas bidirecionais que incluem *buffer*, *flip-flop* de entrada, *buffer tri-state* e *flip-flop* de saída.

A flexibilidade dos FPGAs facilita a reconfiguração de *hardware*. Ferramentas de *Computer Aidded Design* - CAD já disponíveis, nos permitem implementar, testar, simular e corrigir projetos eletrônicos computacionais com grande facilidade sem que se tenha gastos elevados no como no desenvolvimento de ASICs (não implementados em

FPGAs). É possível simular o funcionamento do projeto antes de produzi-lo, evitando assim desperdícios de recursos materiais e temporais.

2.7 *Cores*

Core é um bloco funcional complexo, pré-definido e pré-testado que é integrado à lógica do projeto eletrônico digital. O avanço nas tecnologias de fabricação de circuitos integrados com dimensões na ordem de sub-micron, introduziu o conceito de *System-Level Integration* - SLI (Integração em Nível de Sistema), também conhecido como *Systems on a Chip* - SOC (sistema digital em um único chip). Esta abordagem viabiliza o emprego de *cores* gerando uma redução no tempo de desenvolvimento de projetos, menor custo de produção e menor *time-to-market*.

2.7.1 Tipos de *cores* disponíveis

Cores podem ser classificados em 3 categorias: hard *cores*, firm *cores* e soft *cores*.

- Hard *cores*:
  São aqueles otimizados para uma tecnologia específica e não podem ser modificados pelo projetista do sistema digital. Estes *cores* tem um *layout* e *floorplan* pré-definidos incluídos na arquitetura do projeto. Possuem a vantagem de garantir o tempo de propagação no circuito. Sua desvantagem é a de não permitir qualquer tipo de modificação ou personalização

- Firm *cores*:
  São um misto de código fonte com o *netlist* (lista de conexões) gerado para a tecnologia empregada (que muda de fabricante para fabricante). Neste tipo de *core* o código fonte é visível ao projetista, permitindo que parte da sua estrutura seja adaptada à necessidade do projeto. Entretanto, como o netlist é específico para

uma dada tecnologia, existe a dificuldade do uso de componentes fornecidos por fabricantes diferentes.

- Soft *cores*:
  São baseados em *Hardware Description Language* - HDL (Linguagem de Descrição de *Hardware*), oferecendo flexibilidade e independência de tecnologia. O *core* pode ser modificado (adaptado) e empregado com tecnologias de fabricantes diferentes. Contudo, os *soft cores* apresentam a desvantagem da não garantia de se alcançar os critérios temporais (*timing*) do circuito. Neste caso o *core* precisa ser sintetizado o circuito roteado para cada diferente utilização.

2.7.2 Metodologia para projetos baseados em *cores*

Para que se obtenha sucesso em um projeto baseado em *core* faz-se necessário passar por 3 estágios fundamentais:

1. Projeto: Consiste da síntese do nível mais alto de abstração do projeto e do projeto da aplicação do usuário com o *firm core*;

2. Implementação: Durante este estágio, tarefas de transição são realizadas. Muitas etapas intermediárias resultam em partes de projeto não visíveis ao projetista, resultando em um bloco único formado pelo *firm core* e a aplicação.

3. Verificação: É o estágio final do fluxo de projeto. Este estágio consiste em duas tarefas principais: verificação e simulação. Na verificação a análise estática de tempo verifica se as metas do projeto foram alcançadas. Na simulação o timing do sistema e sua funcionalidade são checados.

Capítulo 3

Objetivos

3.1 Objetivos Gerais

Implementar um módulo de *hardware* (*firm core*) que minimize o gargalo da transferência de dados na interação *hardware/software*. Este *hardware*, implementado em FPGA, será constituído de uma iterface para barramento *Peripheral Component Interconnect* - PCI.

3.2 Ojetivos Específicos

- Estudar o barramento PCI 33 MHz, 32 bits;
- Desenvolver *drivers* em *software* para acesso à barramento PCI;
- Definir a especificação de um *core* PCI;
- Implementação e simulação funcional em *Very High Speed Integrated Circuits Hardware Description Language* - VHDL, deste *core*;
- Estudo dos sinais PCI utilizando analisador lógico de sinais digital;
- Implementar o módulo de *hardware* em placa de prototipação, desenvolvendo uma aplicação *back-end* simples.

Capítulo 4

Cronograma e recursos necessários

4.1 Cronograma

A Tabela 1 apresenta o cronograma de todas as tarefas a serem cumpridas.

4.2 Recursos necessários

Este trabalho contará com recursos existentes no Grupo de Apoio ao Projeto de *Hardware* - GAPH, da Faculdade de Informática - FACIN. *Software*s de síntese (lógica e física) de circuitos digitais como Active VHDL e Foundation 2.1i, serão amplamente empregados. O analisador lógico HP1663 e a placa de prototipação Virtex-V300, serão utilizados na análise de sinais PCI e na implementação do *core* bem como da aplicação *back-end*. Há também, uma dependência de recursos a serem aplicados na aquisição de placas extensoras para placas padrão PCI, as quais permitem acesso a todos os sinais do barramento.

Bibliografia

[1] Athanas, P. M., and Silverman, H. F. Processor reconfiguration through instruction-set metamorphosis. In *IEEE Computer*. IEEE, Mar 1993, pp. 11–18.

[2] dos Santos Adário, A. M. Arquiteturas reconfiguráveis. T.I 650 - CPGCC-UFRGS.

[3] et al., H. M. A large scale fpga with 10k core cells with cmos 0.8um 3-layered metal process. In *Custom Integrated Circuits Conference CICC'99* (May 1991), pp. 6.4.1–6.4.4.

[4] Gokhale, M., Holmes, B., Kopser, A., Kunze, D., Lopresti, D., Lucas, S., Minnich, R., and Olsen, P. Splash: A reconfigurable linear logic array. ftp:ftp.super.org/pub/fpga/splash-1/splash-1.ps, May 1997.

[5] Lewis, D. M., Galloway, D. R., van Ierssel, M., Rose, J., and Chow, P. The transmogrifier-2: A 1 million gate rapid-prototyping system. In *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 6. IEEE, Jun 1998, pp. 188–198.

[6] Logic, C. *CFA 6006 Field Programable Gate Array Data Sheet.* Sunnyvale, CA, 1991.

[7] Ltd, A. *CAL 1024 Data Sheet.* Edinburg, Scotland, 1989.

[8] Mangione-Smith, W. H., Hutchings, B., Andrews, D., DeHon, A., Ebeling, C., R.Hartenstein, Mencer, O., Morris, J., Palem, K., Prasanna,

V. K., and Spaanemburg, H. A. E. Seeking solutions in reconfigurable computing. In *IEEE Computer*. IEEE, Dec 1997, pp. 38–43.

[9] Perissakis, S., Joo, Y., Ahn, J., Dehon, A., and Wawrzynek, J. Embedded dram for a reconfigurable array. In *Proceedings of the 1999 Symposium on VLSI Circuits - VLSI'99* (Jun 1999).

[10] Plessley Semiconductors. *ERA 60100 Preliminary Data Sheet*. England, 1989.

[11] Rose, J., Gamal, A. E., and Sangiovanni-Vincentelli, A. Architecture of field-programmable gate arrays. In *Proceedings of the IEEE* (Jul 1993), vol. 81, IEEE, pp. 1013–1029.

[12] Villasenor, J., and Mangione-Smith, W. H. Configurable computing. In *Scientific American*. Scientific American, Jun 1997, pp. 54–59.

[13] Vuillemin, J., Bertin, P., Roncin, D., Shand, M., Touati, H., and Boucard, P. Programmable active memories: Reconfigurable systems come of age. In *IEEE Transactions on VLSI Systems*. IEEE, 1995, pp. 1–14.

[14] Wirthlin, M. J., and Hutchings, B. L. Disc: The dynamic instruction set computer. In *Proceedings of the SPIE 2607* (1995), J. Schewel, Ed., pp. 92–103.

[15] Xilinx. *The Real PCI*. Xilinx, CA, March 1999.