Pontifícia Universidade Católica do Rio Grande do Sul
Faculdade de Informática
Pós-Graduação em Ciência da Computação

Implementação de Sistemas Digitais Reconfiguráveis Parcial, Remota e Dinamicamente

Daniel Gomes Mesquita

Orientador: Professor Fernando Gehm Moraes

Plano de Estudos e Pesquisa

Porto Alegre, Novembro de 2000

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

1.1 Considerações Iniciais

Muitas aplicações emergentes em comunicação, computação e eletrônica necessitam que suas funcionalidades permaneçam flexíveis mesmo depois de o sistema ter sido manufaturado [HAD95]. Tal flexibilidade é fundamental, uma vez que requisitos dos usuários, características dos sistemas, padrões e protocolos podem mudar durante a vida do produto. Essa flexibilidade também pode prover novas abordagens de implementação voltadas para ganhos de desempenho, redução dos custos do sistema ou redução do consumo geral de energia.

A flexibilidade funcional pode ser conseguida através de atualizações de software, mas desta forma a mudança é limitada somente à parte programável dos sistemas. Desenvolvimentos recentes na tecnologia de matrizes de portas programáveis no campo (*Field-Programmable Gate Array*s, ou FPGAs) têm introduzido suporte para modificações rápidas em tempo de execução do hardware do sistema [XIL00]. Essas modificações referem-se a mudanças em circuitos digitais via reconfiguração sem a interrupção da operação do circuito. A implementação de sistemas que demandam flexibilidade, alto desempenho, alta taxa de transferência de dados e eficiência no consumo de energia são possibilitadas por essas tecnologias. Isto inclui aplicações de televisão digital, comunicações sem fio reconfiguráveis, sistemas de computação de alto desempenho, processamento de imagens em tempo real e sinais digitais adaptáveis, produtos para consumo atualizáveis remotamente, entre outros.

Além das características citadas acima, a reconfigurabilidade também contribui para a economia de recursos: quando uma dada tarefa pode ser realizada em várias fases, uma diferente configuração pode ser carregada para cada fase seqüencialmente [VIL97]. Desta forma o tamanho do sistema pode ser menor, o que implica na redução de preço. Reconfigurabilidade também faz do desenvolvimento e teste de hardware tarefas mais rápidas e mais baratas. E há ainda o uso de reconfigurabilidade como tecnologia para construção de sistemas tolerantes a falhas: tais sistemas podem realizar auto-checagem e reconfigurar a si mesmos, substituindo elementos defeituosos por elementos reserva disponíveis.

1.2 Motivação

A despeito do alto desempenho demonstrado pelos sistemas de computação reconfigurável, esses sistemas apresentam algumas limitações comuns:

- *Baixa largura de banda e alta latência de interface:* Como os sistemas reconfiguráveis são comumente anexados a um computador hospedeiro como um periférico através de um barramento com banda limitada e alta latência, ocorre uma alta sobrecarga de comunicação entre o sistema reconfigurável e o processador do hospedeiro. Isto limita as acelerações possíveis e evita uma cooperação mais próxima entre sistemas fixos e reconfiguráveis. A exceção a esta desvantagem são as arquiteturas compostas de processador, memória e lógica reconfigurável em um só circuito integrado; como por exemplo as arquiteturas do FIPSOC [SID99] e Trumpet [PER99] .

- *Alta sobrecarga de reconfiguração*: Uma vez que os custos de reconfiguração são altos em quase todas tecnologias reconfiguráveis disponíveis, o tempo de reconfiguração deve ser amortizado sobre uma grande

quantidade de processamento para justificar o sistema computacional [WIR98]. Freqüentemente isto significa que uma única configuração deve ser mantida durante toda uma dada aplicação, mesmo quando porções diferentes da aplicação devam ser aceleradas com diferentes lógicas especializadas. Sistemas que utilizam o paradigma de reutilização seqüencial já não possuem esse problema, tais como o DPGA [DEH94].

Apesar da tecnologia FPGA reconfigurável ter sido comercialmente disponibilizada a mais de uma década [ALG00], o número de ferramentas capazes de suportar projeto de sistemas reconfiguráveis é ainda muito limitado. Muitas das ferramentas existentes são baseadas no fluxo de projeto de FPGAs convencionais, e requerem altos níveis de conhecimento e improvisação, no sentido de produzir um sistema reconfigurável funcional. Alguns fatores como ferramentas de projeto de sistemas reconfiguráveis relativamente imaturas e requisitos de projetos conflitantes (devido ao grande número de tecnologias reconfiguráveis) tornam o projeto destes sistemas reconfiguráveis uma tarefa desafiadora.

Uma das áreas para a qual inexiste ferramenta eficiente é reconfiguração parcial. Um exemplo deste tipo de reconfiguração foi descrito em [XIL00a], e utiliza um *script* escrito em Perl para a escrita de um arquivo de configuração parcial. Tal forma de reconfiguração é limitada a esta aplicação e possui interface pouco amigável. Além da falta de ferramentas de *CAD* adequadas, para a reconfiguração parcial, bem como para a reconfiguração dinâmica, ainda não existem aplicações comerciais que justifiquem o esforço para sua utilização.

Contudo, a possibilidade de usar hardware reconfigurável da mesma forma que o microprocessador utiliza a memória virtual conduz ao conceito de hardware sob demanda, que pode ter aplicações interessantes em processamento de imagens [VIL97] e em telecomunicações.

2 Referencial Teórico

2.1 Definições

Por ser uma área relativamente nova, a computação reconfigurável introduz alguns neologismos, e altera o significado de algumas expressões. A seguir são apresentados conceitos, palavras e expressões relevantes ao entendimento deste trabalho.

FPGA: *Fiel Programmable Gate Array* - dispositivo que consiste em uma matriz de blocos lógicos cercada de blocos de entrada e saída, e conectada por fios programáveis de interconexão.

Grão-grosso: Os FPGAs de granularidade grossa possuem como grão unidades lógicas e aritméticas (ULAs), pequenos microprocessadores e memórias. Como exemplos desse tipo de arquitetura podem ser citadas as máquinas RAW [WAI97] e a arquitetura GARP [CAL00].

Grão-médio: Os FPAGAs que tem granularidade média consistem em blocos lógicos bastante grandes, freqüentemente contendo duas ou mais tabelas de busca (*look-up tables* ou *LUTs*) e dois ou mais *flip-flops*. A maioria das arquiteturas de FPGAs implementa a lógica em *LUTs* de quatro entradas. Como exemplos de dispositivos que atualmente possuem granularidade média podem ser citados as famílias Spartan e Virtex, da Xilinx; Flex e Apex, da Altera; e AT40K, da Atmel.

Grão-fino: Nos dispositivos com granularidade fina há um grande número de blocos lógicos simples. Os blocos lógicos normalmente contêm uma função lógica de duas entradas ou um multiplexador 4 para 1 e um *flip-flop*. As famílias SPGA (Actel) e AT6000 (Atmel) são exemplos atuais de dispositivos de granularidade fina.

Reconfiguração total: É a forma de configuração onde o dispositivo reconfigurável é inteiramente alterado .

Reconfiguração parcial: É a forma de configuração que permite que somente uma porção do sistema reconfigurável seja reconfigurada. Uma reconfiguração parcial pode ser *não-disruptiva* - onde as porções do sistema que não estão sendo reconfiguradas permanecem completamente funcionais durante o ciclo de reconfiguração; ou *disruptiva* - onde a reconfiguração parcial afeta outras partes do sistema, tipicamente necessitando de uma parada no relógio. Reconfiguração parcial não-disruptiva é freqüentemente abreviada para *reconfiguração parcial*.

Reconfiguração dinâmica: Também chamada de *run-time reconfiguration* (*RTR*), *on-the-fly reconfiguration* ou *in-circuit reconfiguration*. Todas essas expressões podem ser traduzidas também como reconfiguração em tempo de execução. Reconfiguração dinâmica é outra forma de expressar a reconfiguração parcial não-disruptiva. Não há necessidade de reiniciar o circuito ou remover elementos reconfiguráveis para programação.

DPGA: *Dinamically Programmable Gate Array*. É uma terminologia proposta em [DEH94], que denota um hardware que pode ser programado em execução, durante a operação do sistema, em função de um conjunto de arquivos de configuração pré-carregados. Pode suportar reconfiguração parcial ou total.

Cache Logic: É um termo usado para indicar hardware reconfigurável dinamicamente através de chaveamento de contexto, como o DPGA. É uma marca registrada da empresa ATMEL [ATM00].

2.2 Estado da Arte

Por tratar-se de uma área bastante nova, há poucos trabalhos relacionados com reconfiguração dinâmica. Contudo, algumas idéias interessantes foram lançadas, e servirão como inspiração para o desenvolvimento deste trabalho. A seguir serão citados os trabalhos mais importantes a respeito de reconfiguração dinâmica. Logo após será descrita a tecnologia que possibilia a implementação de sistemas digitais reconfiguráveis parcial, remota e dinamicamente.

2.2.1 Trabalhos Relacionados

DPGA

A arquitetura da matriz de portas dinamicamente programável (*Dinamically Programmable Gate Array - DPGA*) é particularmente bem adaptada para computação reconfigurável. Diferentemente de FPGAs normais, onde a função de cada elemento da matriz é determinada por seqüências relativamente lentas de reconfiguração, os elementos da matriz DPGA podem chavear rapidamente entre várias configurações pré-programadas. A reconfiguração rápida permite aos elementos do DPGA pré-carregarem múltiplas personalizações para a matriz, e chavear entre configurações em um ciclo de relógio [DEH94].

Semelhantemente aos FPGAs, DPGAs são compostos de uma cadeia de elementos computacionalmente simples. Cada elemento pode desempenhar uma função lógica simples em vários bits de entrada produzindo um ou mais bits de saída. Muitos FPGAs modernos são modelados como tabelas de busca (*lookup tables - LUTs*) programáveis. A LUT constitui a configuração de cada elemento da matriz, conforme pode ser denotado da Figura 2.1. Entre os elementos da matriz há uma rede de interconexão programável que permite aos elementos serem ligados conforme a aplicação necessita. A interconexão em FPGAs é tipicamente configurada pela programação de portas de passagens e multiplexadores. Cada elemento do DPGA usa uma segunda LUT para armazenar diferentes contextos em uma configuração local. Uma mensagem de configuração global informa a cada elemento do DPGA qual a função que irá desempenhar no próximo ciclo do relógio. A interconexão configurável em um DPGA tem uma tabela de configurações carregadas, e seleciona entre as configurações baseada no corrente identificador de contexto.

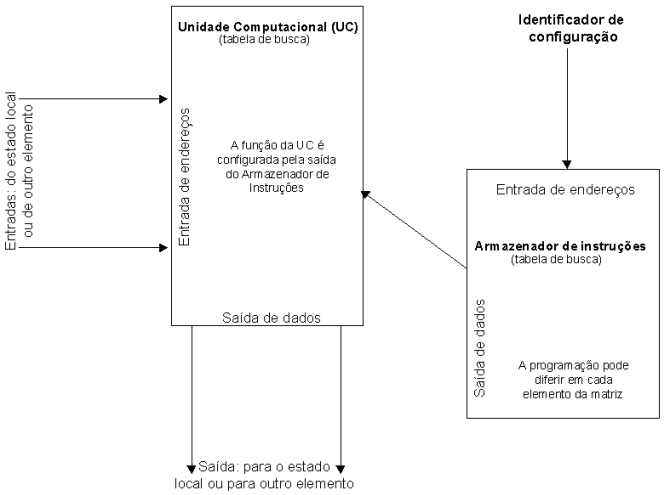

Figura 2.1: Elemento da matriz DPGA

Ao custo de um tamanho maior de elementos e interconexão, o DPGA serve como um FPGA de múltiplos-contextos. Com o espaço dos contextos pré-carregados, o DPGA pode chavear "personalidades" completamente diferentes de um ciclo de relógio para outro. A tabela de configuração do DPGA efetivamente atua como um cache de configurações de elementos da matriz. Fornecendo a cada elemento uma pequena cache de configuração estreitamente acoplada a ele, consegue-se efetivamente uma taxa de reconfiguração alta.

Os múltiplos-contextos carregados permitem a utilização mais eficiente da matriz de elementos. Em aplicações mais pesadas, a rápida reconfiguração permite a uma única matriz DPGA ser carregada com múltiplas configurações simultaneamente. O DPGA pode chavear entre configurações para acelerar diferentes partes de uma aplicação.

FIPSOC

FIPSOC é um dispositivo do tipo *Sistem-on-Chip (SoC)* desenvolvido pela empresa SIDSA para desenvolvimento rápido de aplicações analógicas e digitais integradas. Cada membro da família FIPSOC possui um microcontrolador 8051 embutido, um FPGA, e um conjunto de células analógicas otimizadas para aquisição de sinais e conversões A/D e D/A, como pode ser visto na Figura 2.2. A SIDSA disponibiliza várias ferramentas de CAD integradas, que possibilitam ao usuário especificar, emular, e mapear todo o projeto em apenas um circuito integrado. A lógica programável do FIPSOC é uma matriz de macro células digitais (*Digital Macro Cells - DMCs*) que podem ser configuradas para funções específicas. As DMCs são de granularidade fina, baseadas em células de RAM estática programáveis, que incluem LUTs de 4 entradas e 4 *flip-flops* para cada uma. O subsistema analógico consiste de blocos analógicos configuráveis (*Configurable Analog Blocks - CABs*) de granularidade grossa. Os CABs permitem configurar diferentes funções analógicas, tais como amplificação diferencial e conversão de dados. A parte digital das células analógicas pode ser independentemente controlada pelo microprocessador ou pelo FPGA [SID99].

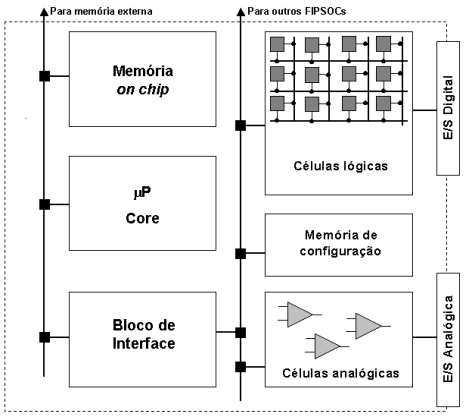

Figura 2.2: Estrutura do FIPSOC

A configuração do circuito integrado é armazenada nos bits de RAM estática. Pode ser feito o carregamento de uma nova configuração para a memória de recuperação enquanto a célula está em operação. Não há a necessidade de parar o circuito integrado para reconfigurá-lo: duas configurações extras podem ser trocadas em tempo-real.

Por exemplo, um contexto de recuperação pode sempre ser usado para configuração enquanto outro realiza uma computação de propósito geral. Esta reconfiguração dinâmica, parcial ou total, pode também ser colocada em funcionamento pelo próprio hardware reconfigurável, sem a intervenção do microprocessador. A reconfiguração dinâmica pode ser aplicada sobre uma única célula programável, sobre um conjunto selecionado delas, ou sobre toda a lógica reconfigurável.

Spyder

SPYDER é um anagrama de *REconfigurable Processor Development SYstem*. Este sistema é um coprocessador reconfigurável que foi criado com o objetivo de auto-adaptar-se a uma dada aplicação de uma forma transparente ao usuário [SAN99].

Um processador consiste de uma unidade de controle e uma unidade de processamento. SPYDER atua como uma unidade de processamento, que consiste de três FPGAs conectados a um banco de registradores. Cada FPGA mantém um acesso independente aos registradores no sentido de permitir processamento paralelo de dados e, por conseguinte, a implementação de arquiteturas superescalares. A Figura 2.3 exibe a arquitetura SPYDER.

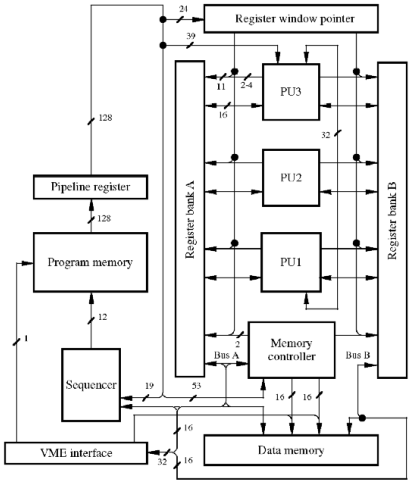

Figura 2.3: Arquitetura Spyder

Dada a complexidade do problema que essa arquitetura propos-se a resolver, foi implementada uma solução intermediária: o usuário determina os operadores e os descreve em uma linguagem de alto nível (C++). O compilador então gera a configuração correspondente aos FPGAs. Finalmente a aplicação é escrita pelo usuário através do uso do código correspondente e determina as operações de tal forma que consiga o máximo grau de paralelismo.

Como um coprocessador comum, SPYDER é conectado a um computador hospedeiro, que manipula entradas e saídas, e executa os programas de desenvolvimento. Sua arquitetura segue o modelo de von Neumman [RAD98].

Uma das grandes vantagens desta abordagem é a idéia de colocar unidades reconfiguráveis em uma unidade de processamento von Neumman. Isto combina as vantagens de um processador von Neumman em aplicações de propósito geral com a alta eficiência de arquiteturas reconfiguráveis em aplicações de domínio específico.

Contudo, mais uma vez, o problema desta arquitetura é a falta de uma ferramenta capaz de esconder detalhes de hardware do usuário.

Trumpet

Trumpet é um circuito integrado de teste que foi projetado para uso no estudo das vantagens envolvidas no projeto de matrizes reconfiguráveis acopladas a bancos de memória DRAM de alta capacidade [PER99]. O ponto alto desta arquitetura consiste em uma rede de "páginas computacionais" (submatrizes configuráveis) e páginas de memória (*Configurable Memory Blocs - CMBs*). Páginas computacionais são baseadas em LUTs de 5 entradas. Páginas de memória e computacionais são interconectadas por uma rede (em forma de árvore) que se estende desde cada submatriz (folhas), alcançando os blocos lógicos individualmente, até uma conexão com um microprocessador (raiz). A Figura 2.4 ilustra essa arquitetura.

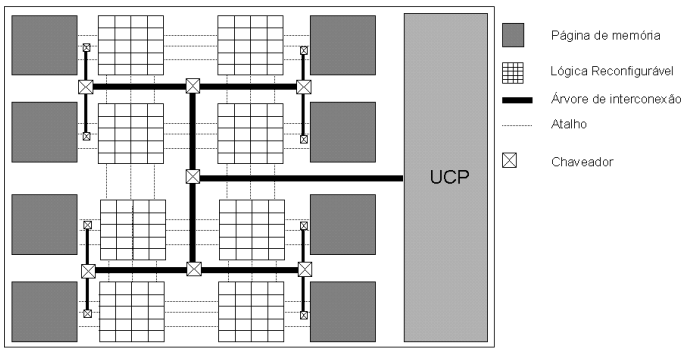

Figura 2.4: Arquitetura Trumpet

Cada submatriz, CMB e chaveador requer um ou mais conjuntos de bits de configuração, dependentes da aplicação. Tais conjuntos de bits podem ser pré-carregados na DRAM, e carregados sob demanda em seus respectivos destinos. Submatrizes e CMBs são arranjados em pares para propósitos de reconfiguração, enquanto a configuração de cada chaveador pode ser assinalada ao CMB mais próximo. Desta forma é possível conseguir reconfiguração total ou parcial no tempo em que se leva para configurar um subarray, um CMB e um ou dois chaveadores. Além disso, conjuntos de bits de estados para CMBs e submatrizes podem ser carregados para inicialização, diagnóstico, ou chaveamento de contexto.

Trumpet é importante por ser uma das primeiras abordagens a valorizar a união de processador, memória e lógica reconfigurável. Grandes bancos de memória tornam possível armazenar várias configurações num mesmo circuito integrado, possibilitando uma rápida reconfiguração em tempo de execução. Ademais, núcleos de aplicações podem beneficiar-se da largura de banda para acessar dados. Também vale ressaltar que a complexidade de desenvolvimento para DRAM é ocultada do programador, através da utilização de uma interface semelhante a SRAM.

Apesar de não haver conprovação de que Trumpet represente o equilíbrio na utilização de recursos de memória e processamento, ainda assim é um passo importante nessa direção.

2.2.2 Tecnologias que habilitam sistemas digitais reconfiguráveis parcial, remota e dinamicamente

A reconfiguração parcial foi implementada primordialmente pelas empresas National, Algotronix e Xilinx; que produziram as famílias de FPGAS Clay [NAT98], Cal1024 [ALG89] e XC6200 [XIL99], respectivamente. Tais FPGAs não lograram grande sucesso comercial principalmente pelo fato de não terem sido produzidas ferramentas eficientes de projeto, de roteamento e de posicionamento.

Outro fabricante de FPGAs, a Altera, alega que a partir da família APEX permitiu reconfiguração parcial, contudo isto ocorre de forma muito limitada. A reconfiguração parcial dessa família dá-se através do projeto de lógica em RAM, criando uma LUT onde podem ser implementadas funções com 7 entradas e 16 saídas. Depois dessa lógica ser implementada no bloco de RAM o sistema pode reescrevê-la em qualquer tempo, mudando a configuração de parte do sistema. A grande limitação desta abordagem é que em algum lugar do circuito deve-se armazenar todas as configurações possíveis que irão modificar a RAM, isto porque não há como fazer a carga externa de novas configurações.

Duas empresas - Atmel e Xilinx - comercializam famílias FPGAs que suportam reconfiguração parcial. A seguir será feita uma breve descrição da família AT40k da Atmel e das características da família Virtex, da Xilinx.

Atmel

Os FPGAs da família AT40K foram especialmente projetados para suportarem *Cache Logic,* que é uma técnica para construir sistemas e lógica adaptáveis. Num sistema de *Cache Logic* somente as porções da aplicação que estão ativas em um dado momento realmente estão implementadas no FPGA, enquanto funções inativas são armazenadas externamente numa memória de configuração barata. Se novas funções se fazem necessárias, as antigas são sobrescritas, como mostrado no diagrama contido na Figura 2.5. Este procedimento aproveita-se da latência funcional inerente a muitas aplicações - em qualquer tempo dado, somente uma pequena proporção da lógica está de fato ativa.

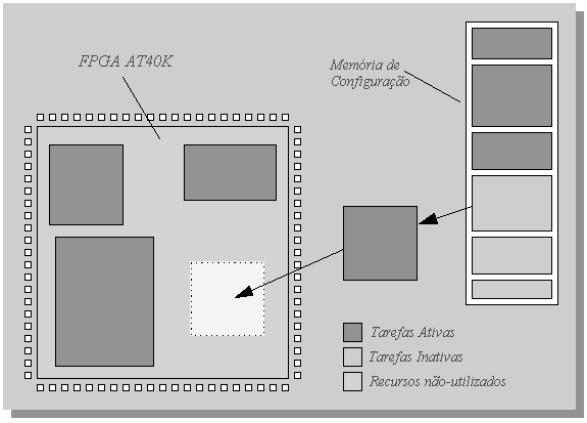

Figura 2.5: Diagrama da *Cache Logic*

A série AT40K implementa *Cache Logic*, pois pode ser parcialmente reconfigurável, sem interromper a operação da lógica remanescente no dispositivo. Isto permite que funções sejam substituídas em tempo de execução no FPGA, enquanto o sistema continua a operar [ATM00].

Xilinx

A arquitetura Virtex suporta reconfiguração parcial [XIL99a]. Cada dispositivo Virtex contém blocos lógicos configuráveis (*Configurable Logic Blocks - CLBs*), blocos de entrada/saída (*Input/Output Blocks - IOBs*), blocos de RAM, recursos de relógio, roteamento programável e configuração do circuito elétrico. Essas funcionalidades lógicas são configuradas através do arquivo de configuração. Arquivos de configuração contém uma mescla de comandos e dados. Eles podem ser lidos e escritos através de uma das interfaces de configuração da Virtex.

A memória de configuração da Virtex pode ser vista como uma matriz retangular de bits. Estes bits são agrupados em quadros verticais com um bit de largura, e se estendem do topo à base da matriz. Um quadro é a unidade atômica de configuração: é a menor porção de memória de configuração que pode ser lida ou nela escrita.2.6.

Quadros são lidos e escritos seqüencialmente, com endereços crescentes para cada operação. Múltiplos quadros consecutivos podem ser lidos ou escritos com um único comando de configuração. A menor quantidade de informações que pode ser lida ou escrita com um único comando é um único quadro. A matriz de CLBs inteira, mais o bloco de interconexão de SelectRAM podem ser lidos ou escritos com apenas um comando. Cada bloco de conteúdo de SelectRAM deve ser lido ou escrito separadamente.

Como os membros da família Virtex possuem uma estrutura em colunas, onde os quadros podem ser lidos ou escritos individualmente, é possível reconfigurar parcialmente esses dispositivos através da modificação desses quadros no arquivo de configuração.

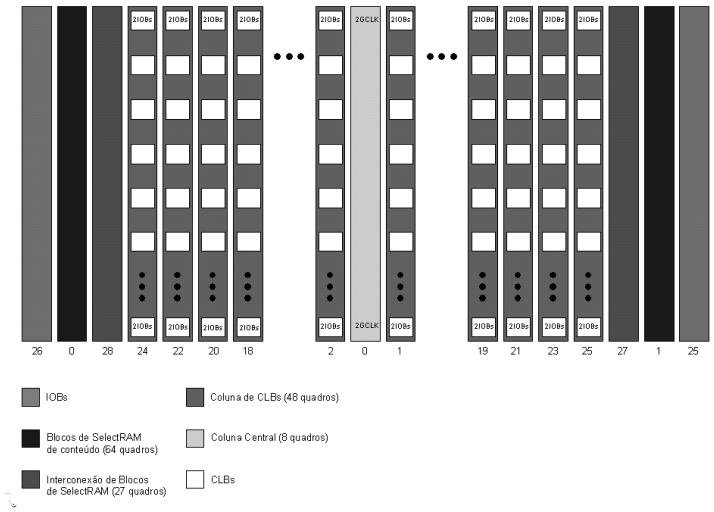

Figura 2.6: Exemplo de colunas de configuração para o FPGA XCV50

JBITS

JBits é um conjunto de classes Java que fornecem uma API (Application Program Interface) que permite manipular o arquivo de configuração da família de FPGAs Virtex da Xilinx. Esta interface opera tanto em arquivos de configuração gerados pelas ferramentas de projeto da Xilinx quanto em arquivos de configuração lidos do hardware [XIL00b].

O modelo de programação utilizado pelo Jbits é um a matriz bidimensional de blocos lógicos configuráveis (*Configurable Logic Blocks* - CLBs). Cada CLB é referenciada por uma linha e uma coluna. Assim, todos os recursos configuráveis na CLB selecionada podem ser configurados ou analisados. Além disso, o controle de todo o roteamento adjacente à CLB selecionada torna-se disponível. Devido ao código ser escrito em Java, o tempo de compilação é bastante rápido, e pelo controle ser ao nível de CLB, os arquivos de configuração podem ser modificados ou gerados rapidamente.

Esta API tem sido utilizada para construir circuitos completos e para modificar circuitos existentes. O suporte à orientação à objetos da linguagem Java permite que *cores* parametrizáveis sejam implementados. Esta API pode ser utilizada como base para a construção de outras ferramentas. Isto inclui ferramentas de projeto tradicionais para executar tarefas como posicionamento e roteamento do circuito, bem como ferramentas de aplicação específica, como por exemplo um configurador de *cores*.

É necessário que o projetista tenha conhecimento do seu circuito e dos detalhes de configuração do dispositivo, pois, caso contrário, o JBits pode gerar dados que danifiquem o dispositivo. O JBits fornece uma abordagem de linguagem de alto nível para desenvolvimento de sistemas reconfiguráveis incluindo reconfiguração em tempo de execução.

A característica mais importante do JBits para este trabalho é o seu uso no desenvolvimento de aplicações Java RTR. Neste fluxo, os circuitos podem ser configurados durante a execução através de uma aplicação Java que se comunica com a placa contendo o dispositivo Virtex. Isto é possível através do uso do JBits para especificar o projeto e da API do XHWIF para carregar o projeto dentro da mesma aplicação Java.

3 Objetivos

3.1 Objetivo Geral

Este trabalho visa primordialmente o desenvolvimento de uma ferramenta de apoio à implementação de sistemas digitais reconfiguráveis parcial, remota e dinamicamente.

3.2 Objetivos Específicos

3.2.1 Estudo minucioso do endereçamento dos componentes internos à arquitetura Virtex

Atualmente a reconfiguração total de FPGAs com o auxílio de ferramentas automatizadas de posicionamento e roteamento tornam o cálculo de endereçamento dos componentes internos à arquitetura do FPGA transparente ao projetista de hardware.

Contudo, como tais ferramentas não foram escritas para permitir reconfiguração parcial, para realizá-la é necessária a concepção de tal ferramenta. Para tanto é primordial um estudo minucioso do endereçamento dos componentes internos à arquitetura Virtex. Este estudo é, portanto, o primeiro objetivo específico deste trabalho.

3.2.2 Programação do acesso a estes componentes (*slices*, *luts*, blocos de RAM) em uma *web-tool*

Após o estudo minucioso, desenvolver uma ferramenta com o objetivo de inspecionar remotamente valores em arquivos de configuração será a próxima tarefa. Essa será a primeira atividade eminentemente prática do trabalho.

3.2.3 Extender a *web-tool* citada no item 2 para funcionamento com *ReadBack*

Readback é um processo-padrão de leitura dos dados do arquivo de configuração localizado na memória interna do FPGA. A diferença para o objetivo anterior é que o uso do *ReadBack* consiste em ler diretamente do FPGA seu conteúdo, ao invés de isto ocorrer através do arquivo de configuração.

3.2.4 Modificação e geração do arquivo de configuração de forma total e, em seguida, parcialmente

Baseado no exemplo proposto em [XIL00a], modificar parcialmente arquivo de configuração é a quarta tarefa deste trabalho. Note-se que apesar de ser uma modificação parcial, a geração do arquivo de configuração ainda proporcionaria uma reconfiguração total. A tarefa seguinte é gerar um arquivo de configuração parcial.

3.2.5 Desenvolvimento de uma *web-tool* para reconfiguração parcial, remota e dinâmica de dispositivos Virtex

Com base no conhecimento da arquitetura interna e modo de endereçamento de um dispositivo da família Virtex, e dominando a tecnologia para geração de arquivos de configuração parcial, tem-se como meta o desenvolvimento de uma ferramentea que permita reconfiguração parcial e dinâmica, através da Internet.

3.2.6 Validação da ferramenta

Utilizando módulos de hardware (*cores*) produzidos em pesquisa que ocorre paralelamente no GAPH, validar, na forma de implementação remota dos exemplos, a ferramenta citada no item acima. Este objetivo será trabalhado cooperativamente com o mestrando José Carlos Sant'Anna Palma, membro do GAPH que pesquisa metodologias para desenvolvimento de *cores*.

3.2.7 Estudo de arquiteturas-padrão para permitir reconfigração dinâmica em FPGAs

Não foi encontrado na literatura referência sobre proposta de padronização de arquiteturas para que dispositivos possam receber *cores* sem que haja maior necessidade de modificações nesses dispositivos. Pretende-se verificar se de fato não há uma arquitetura-padrão já proposta, e, caso isso seja verdadeiro, estudar a viabilidade da criação de tal arquitetura.

3.2.8 Definição, proposta e experimentos de uma arquitetura para *hard/soft cores* em dispositivos reconfiguráveis parcialmente

O penúltimo objetivo deste trabalho é a especificação e possivelmente o desenvolvimento de uma arquitetura que permita o *download* de *cores* no dispositivo reconfiguirável, de forma análoga a uma arquitetura PCI, onde se anexa uma placa ao sistema sem ter que alterar este.

3.2.9 Redação

A redação de artigos científicos, bom como da dissertação, são os últimos objetivos deste trabalho.

4 Cronograma

A tabela 4.1 mostra a distribuição dos objetivos citados na Seção 3 no decorrer dos meses de março a dezembro de 2001.

	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez
1	X	X								
2		X	X	X						
3				X	X					
4				X	X					
5					X	X	X			
6						X	X	X	X	
7					X	X				
8					X	X	X	X		
9				X				X	X	X

Tabela 4.1: Cronograma para 2001

Legenda

1. Estudo minucioso do endereçamento dos componentes internos à arquitetura Virtex
2. Programação do acesso a estes componentes (*slices*, *luts*, blocos de RAM) em uma *web-tool*
3. Extender a *web-tool* citada no item 2 para funcionamento com *ReadBack*
4. Modificação e geração do arquivo de configuração total e, em seguida, parcialmente
5. Desenvolvimento de uma *web-tool* para reconfiguração parcial, remota e dinâmica de dispositivos Virtex
6. Validação da ferramenta
7. Estudo de arquiteturas-padrão para permitir reconfigração dinâmica em FPGAs
8. Definição, proposta e experimentos de uma arquitetura para *hard/soft cores* em dispositivos reconfiguráveis parcialmente
9. Redação de artigos científicos e da dissertação

5 Considerações Finais

Este trabalho explora novas tecnologias em computação reconfigurável, e traz uma abordagem inédita quando propõe uma arquitetura padrão para sistemas reconfiguráveis parcialmente, além de pretender desenvolver uma ferramenta que permita implementação de sistemas digitais reconfiguráveis parcial, remota e dinamicamente.

Trata-se de um trabalho eminentemente prático, onde as maiores dificuldades concentrar-se-ão na escassez de bibliografia e na imaturidade das tecnologias que habilitam reconfiguração parcial.

Sob o ponto de vista de pesquisa e contribuição científica, dado o ineditismo e o grau de complexidade, acredita-se que este trabalho seja o ponto de partida para o desenvolvimento de novas ferramentas e métodos para o projeto de circuitos digitais, sejam eles completos, ou modulares.

Referências Bibliográficas

[ALG00] ALGOTRONIX, Equipe de documentação da. **Company profile**. Disponível por WWW em http://www.algotronix.com/profile.htm (2000).

[ALG89] ALGOTRONIX, Equipe de documentação. **Cal1024 datasheet**. Disponível por WWW em http://dec.bournemouth.ac.uk/drhw_lib/archive/cal1024.ps.gz (1989).

[ATM00] ATMEL, Equipe de documentação da. **At40k series configuration**. Disponível por WWW em http://www.atmel.com/atmel/postscript/doc1009.ps.zip (2000).

[CAL00] CALLAHAN, Timothy J.; HAUSER, John R.; WAWRZINEK, John. Introduction to the special issue on computational linguistics using large corpora. **Computer Magazine**, n.4, p.62–69, 2000.

[DEH94] DEHON, André. **Dpga-coupled microprocessors**: commodity ics for the early 21st century. Disponível por WWW em http://www.ai.mit.edu/projects/transit/tn100/tn100.html (Janeiro 1994).

[HAD95] HADLEY, John D.; HUTCHINGS, Brad L. Design methodologies for partially reconfigured systems. In: IEEE WORKSHOP ON FPGAS FOR CUSTOM COMPUTING MACHINES 1995, 1995, California, USA. **Anais...** 1995. p.78–84.

[NAT98] NATIONAL, Equipe de documentação da. **Navigator**. Disponível por WWW em http://www.national.com/appinfo/milaero/files/f61.pdf (Julho 1998).

[PER99] PERISSAKIS, Stylianos; JOO, Yangsung; AHN, Jinhong; DEHON, Adré; WAWRZYNEK, John. Embedded dram for a reconfigurable array. In: VLSI '99, 1999, USA. **Anais...** 1999.

[RAD98] RADUNOVIC, Bozidar; MILUTINOVIC, Veljko. A survey of reconfigurable computing architectures. In: FPL98, 1998, Tallin, Estonia. **Anais...** 1998.

[SAN99] SANCHEZ, Eduardo; AL. et. Static and dinamic configurable systems. Nova Iorque, EUA: [s.n.], 1999. p.556–564.

[SID99] SIDSA, Equipe de documentação da. **Fipsoc mixed signal system-on-chip**. Disponível por WWW em http://www.sidsa.com/FIPSOC/fipsoc1.pdf (Setembro 1999).

[VIL97] VILLASENOR, John; MANGIONE-SMITH, William H. Configurable computing. **Scientific American**, v.19, p.54–58, 1997.

[WAI97] WAINGOLD, Elliot; TAYLOR, Michael; SRIKRISHNA, Devabhaktuni; SARKAR, Vivek; LEE, Walter; LEE, Victor; KIM, Jang; FRANK, Matthew; FINCH, Peter; BARUA, Rajeev; BABB, Jonathan; AMARASINGHE, Saman; AGARWAL, Anant. Baring it all to software: raw machines. **IEEE Computer**, New York, USA, p.86–93, 1997.

[WIR98] WIRTHLIN, Michael J.; HUTCHINGS, Brad L. Improving computational density using run-time circuit reconfiguration. Nova Iorque, EUA: [s.n.], 1998. p.247–256.

[XIL00] XILINX, Equipe de documentação da. **Frequently asqued questions about partial reconfiguration**. Disponível por WWW em http://www.xilinx.com/xilinxonline/partreconfaq.htm (2000).

[XIL00a] XILINX, Equipe de documentação da. **Xapp153**: status and control semaphore registers using partial reconfiguration. Disponível por WWW em http://www.xilinx.com/xapp/xapp153.pdf (Novembro 2000).

[XIL00b] XILINX, Equipe de documentação da. **The jbits 2.4 sdk for virtex**. Disponível por FTP em ftp://customer:xilinx@ftp.xilinx.com/download/JBits2_4.exe (Novembro 2000).

[XIL99] XILINX, Equipe de documentação. **Xc6200 datasheet**. Disponível por WWW em http://dec.bournemouth.ac.uk/drhw_lib/archive/xc6200.pdf (Junho 1999).

[XIL99a] XILINX, Equipe de documentação. **Satus and control semaphore registers using partial reconfiguration**. Disponível por WWW em http://www.xilinx.com/xapp/xapp153.pdf (Junho 1999).