

RECONFIGURAÇÃO PARCIAL E REMOTA DE CORES FPGAS

Daniel Mesquita ¹
Fernando Gehm Moraes ²
José Carlos Sant'Anna Palma ³
Leandro Heleno Möller ⁴
Ney Laert Vilar Calazans ⁵

Pontifícia Universidade Católica do Rio Grande do Sul - PUCRS
Faculdade de Informática - FACIN
Grupo de Apoio ao Projeto de Hardware - GAPH
Av. Ipiranga, 6681 - Prédio 30 / BLOCO 4 Telefone: +55 51 320-3611 - Fax: +55 51 320-3621
CEP 90619-900 - Porto Alegre - RS - BRASIL

ABSTRACT

In application areas such as multimedia, telecommunications and embedded systems in general, high performance is an essential issue. The strategy of using a conventional microprocessor executing the tasks of such systems in software usually may not correspond to the performance expectations or requirements. Combining conventional microprocessors with (re) configurable hardware devices such as RAM-based FPGAs it is possible to obtain an economically interesting product and yet efficient for a more vast range of applications. However, not always the ideal hardware for fulfilling the performance requirements fits in the configurable device due to size, power or speed restrictions of the device or its technology. The problem can be avoided, for example, by employing a scheme for dynamically reconfiguring the FPGA(s). This work addresses partial and/or remote reconfiguration of FPGAs, but has as main objective to provide a base for research and development in the field of dynamically, partially and remotely reconfigurable systems. It presents an analysis of current devices and tools that enable partial and remote reconfiguration of FPGAs. Next, it shows the results of applying a partial reconfiguration process achieved by using software developed by the Hardware Design Support Group (GAPH) at the authors' Institution.

RESUMO⁶

Em áreas como multimídia, telecomunicações e sistemas embarcados, processamento de alto desempenho é essencial. Um microprocessador convencional executando tarefas de tais sistemas em software costuma não corresponder às expectativas de desempenho desejadas. Combinando microprocessadores convencionais com dispositivos de hardware reconfiguráveis tais como FPGAs pode ser possível obter um produto economicamente viável e ainda assim eficiente para uma gama mais vasta de aplicações. Contudo, nem sempre o hardware ideal pode ser implementado no dispositivo reconfigurável, por exigir mais recursos do que os disponíveis neste. O problema pode ser contornado, por exemplo, através da reconfiguração dinâmica do dispositivo. Este trabalho contempla apenas a reconfiguração parcial e remota de FPGAs, mas tem como finalidade lançar as bases para pesquisa e desenvolvimento na área de sistemas reconfiguráveis parcial, remota e dinamicamente. Será apresentada aqui uma análise de ferramentas e dispositivos que tornam possível a reconfiguração parcial. Também serão mostrados os resultados da aplicação de um processo de reconfiguração parcial realizada através de software desenvolvido pelo Grupo de Apoio ao Projeto de Hardware (GAPH) da Instituição dos autores.

¹ Mestrando em Ciência da Computação (PPGCC – FACIN – PUCRS), Bacharel em Ciência da Computação pela Universidade de Cruz Alta (UNICRUZ, 1999). E-mail: dmesquita@inf.pucrs.br
² Doutor em Informática, opção Microeletrônica (LIRMM, França, 1994), Engenheiro Eletrônico (UFRGS, 1987), Professor Adjunto da Faculdade de Informática/PUCRS. E-mail: moraes@inf.pucrs.br
³ Mestrando em Ciência da Computação (PPGCC – FACIN – PUCRS), Bacharel em Informática pela PUCRS (Campus II – Uruguiana, 1999). E-mail: jpalma@inf.pucrs.br
⁴ Aluno do Bacharelado em Informática da PUCRS E-mail: moller@inf.pucrs.br
⁵ Doutor em Ciências Aplicadas, opção Microeletrônica (UCL, Bélgica, 1993), Engenheiro Eletrônico (UFRGS, 1985), Professor Titular da Faculdade de Informática/PUCRS. E-mail: calazans@inf.pucrs.br
⁶ Este trabalho tem o suporte parcial do CNPq e da FAPERGS.

1. Introdução

Muitas aplicações emergentes em telecomunicações, multimídia e processamento de imagens necessitam que suas funcionalidades permaneçam flexíveis mesmo depois de o sistema ter sido manufaturado [1]. Tal flexibilidade é fundamental, uma vez que os requisitos das aplicações, as características dos sistemas e os padrões e protocolos de comunicação podem mudar durante a vida do produto. Essa flexibilidade propicia novas abordagens de projeto voltadas para ganhos de desempenho, redução dos custos do sistema e/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 sistemas digitais reconfiguráveis tais como os *Field-Programmable Gate Arrays* (FPGAs) baseados em RAM têm introduzido suporte para modificações rápidas em tempo de execução do hardware do sistema [2]. Essas modificações pode hoje ser realizadas via reconfiguração durante a operação do circuito. A implementação de sistemas que exigem 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, comunicação sem fio reconfigurável, sistemas de computação de alto desempenho, processamento de imagens em tempo real e sinais digitais adaptáveis, produtos de 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 configuração diferente pode ser carregada para cada fase, de forma seqüencial [3]. Assim, o tamanho do hardware reconfigurável, e do sistema, pode ser menor, o que pode implicar a redução do preço final. A reconfigurabilidade também faz do desenvolvimento e teste de hardware tarefas mais rápidas e mais baratas.

Neste contexto, é importante avaliar a disponibilidade de dispositivos e ferramentas que permitem a implementação de sistemas digitais reconfiguráveis parcialmente. Este trabalho objetiva discutir o tema da reconfiguração parcial e remota, suas vantagens e problemas. Também é objetivo deste demonstrar a viabilidade do uso de dispositivos específicos para implementar reconfiguração parcial e remota. A reconfiguração dinâmica, conforme definição encontrada na Seção 1.1) excede ao escopo deste trabalho, que é contudo a base para o desenvolvimento de ferramentas que permitam habilitar esta capacidade.

Este trabalho está organizado da seguinte forma: a Seção 1 contextualiza reconfiguração parcial, e fornece algumas definições relevantes. A Seção 2 informa a respeito de dispositivos que permitem reconfiguração parcial e que estão (ou estiveram) disponíveis comercialmente. A Seção 3 apresenta uma breve revisão do estado-da-arte em ferramentas para reconfiguração parcial. A Seção 4 descreve o status atual do projeto de um reconfigurador parcial e remoto de *cores*. Na Seção 5 há uma proposta de interface implementável em lógica configurável que permita inclusão e remoção de *cores* sem maiores alterações no projeto do sistema. Finalmente, a Seção 6 apresenta algumas conclusões deste trabalho, e aborda possíveis trabalhos futuros.

1.1. Definições

Considerando-se que a área de sistemas digitais reconfiguráveis é muito recente, faz-se necessário unificar a terminologia. Para isto propõe-se a seguir uma nomenclatura para os conceitos mais relevantes:

- **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 digital seja reconfigurada. Uma reconfiguração parcial pode ser não-disruptiva, quando as porções do sistema que não estão sendo reconfiguradas permanecem completamente funcionais durante o ciclo de reconfiguração; ou disruptiva, quando a reconfiguração parcial afeta outras partes do sistema, tipicamente necessitando de uma parada no sistema inteiro. Reconfiguração parcial não-disruptiva é freqüentemente abreviada para reconfiguração parcial apenas.
- **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. O termo implica não haver necessidade de reiniciar o circuito ou remover elementos durante a reconfiguração.
- **Chaveamento de contexto:** É a capacidade de um dispositivo ou sistema de ser configurado em tempo de execução, durante a operação do sistema, em função de um conjunto de arquivos de configuração pré-carregados em uma memória de controle interna ao dispositivo. Pode estar associado a procedimentos de reconfiguração parcial ou total.

Além desses conceitos, cabe ainda definir o que seja um *core*. O termo *core* significa “módulos pré-projetados e pré-validados de hardware”.

1.2. Aplicações em reconfiguração parcial remota

O campo da computação reconfigurável avançou significativamente na década de 90, utilizando FPGAs como a base para sistemas reprogramáveis de alto desempenho [4]. Muitos desses sistemas alcançaram altos níveis de desempenho e demonstraram sua aplicabilidade à resolução de uma larga variedade de problemas. Contudo, apesar dos autores desses sistemas os classificarem como reconfiguráveis, a maioria destes é tipicamente configurados uma vez antes de iniciarem a execução da aplicação. Sistemas que utilizam reconfiguração parcial são distintos destes por permitirem especialização da lógica e do roteamento em tempo de execução, conforme pode se visto na Figura 1. A Figura ilustra diferentes estados do sistema no decorrer do tempo. A cada reconfiguração do sistema, novos módulos funcionais podem ser inseridos ou substituídos.

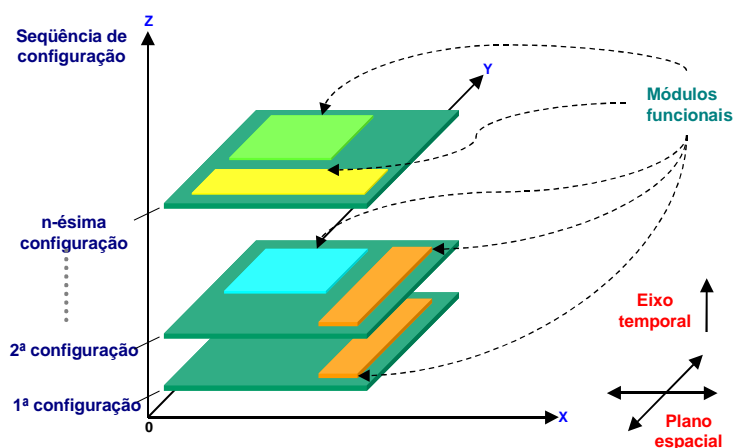


Figura 1 – Gráfico indicativo de RTR.

As partes de uma aplicação que podem ser aceleradas através do uso de hardware reconfigurável freqüentemente são muito numerosas ou complexas para serem carregadas simultaneamente no FPGA disponível. Para esses casos, é interessante que seja possível intercambiar diferentes módulos no mesmo dispositivo.

Reconfiguração parcial é semelhante ao conceito de memória virtual, e pode-se até utilizar o termo “hardware virtual” em alguns casos. Conforme essa idéia, o hardware físico é muito menor do que o somatório dos recursos requeridos para cada uma das configurações. Então, ao invés de reduzir o número de configurações que são mapeadas no dispositivo, apenas ocorre uma troca entre o hardware necessário e o hardware implementado fisicamente.

Na área de telecomunicações pode ser citado um exemplo de reconfiguração parcial não-disruptiva. Em [5] é descrito um projeto para comunicação de dados. A aplicação exige um dispositivo que suporte três diferentes protocolos de rede, e que tenham todos uma interface comum. O dispositivo deve colocado numa central de comunicações, e diferentes clientes podem adquirir canais no sistema escolhendo o protocolo que queiram executar. Como cada canal pode pertencer a um cliente diferente, executando protocolos diferentes, o sistema não pode ser interrompido por causa de um cliente para que seu canal seja mudado. A solução proposta utiliza o FPGA ATMEL AT40K [11], que permite reconfiguração parcial através de uma tecnologia conhecida como *Cache Logic*¹. A lógica de interface foi disposta na base do FPGA, enquanto a área restante foi igualmente dividida entre os três canais. Um arquivo de configuração é gerado para cada canal. No sentido de poder programar os três protocolos em qualquer ordem, também é necessário um arquivo de configuração vazio para cada porção do dispositivo. Assim é possível programar o dispositivo sempre a partir de um estado conhecido.

O interesse em realizar essa reconfiguração parcial remotamente dá-se principalmente pela economia de recursos necessários para modificação do hardware em locais de acesso dificultado pela distância. Por exemplo, um robô enviado ao espaço que contenha um hardware programável pode necessitar de reparos ou atualização do dispositivo reconfigurável. Em não havendo possibilidade de reconfiguração remota, a única forma de efetuar essas modificações seria através do envio de uma missão espacial. Obviamente, o custo de uma reconfiguração remota é irrisório diante daquela hipótese.

1 Cache Logic é a tecnologia utilizada para implementação de chaveamento de contexto em dispositivos da família AT40K da Atmel.

Apesar da aparentemente vasta aplicabilidade para sistemas baseados em reconfiguração parcial, existem poucos dispositivos e sistemas que de fato implementam esta característica. Isto se dá principalmente pela quase inexistência de ferramentas capazes de dar suporte ao projeto de sistemas reconfiguráveis. Um segundo motivo é a incerteza do futuro, a longo prazo, desta técnica. Muitas das ferramentas existentes são baseadas no fluxo de projeto de FPGAs convencionais, e exigem altos níveis de conhecimento e improvisação, no sentido de produzir um sistema reconfigurável funcional. Além disto, há o problema da sobrecarga implicada pelas tarefas de reconfiguração: uma vez que os custos de reconfiguração, em termos de tempo, são relativamente altos em quase todas as tecnologias reconfiguráveis disponíveis, o tempo de reconfiguração deve ser amortizado sobre uma grande quantidade de processamento para justificar o sistema computacional [6]. Por exemplo, um dispositivo Virtex XCV300, que possui aproximadamente 1,5 Mbits de configuração, a uma taxa de 100MHz toma 15ms para ser completamente configurado. Frequentemente, 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 chaveamento de contexto interno ao dispositivo reconfigurável tais como o DPGA [7] não possuem esse problema, mas implicam um aumento significativo da área do dispositivo destinada ao controle da reconfiguração e o armazenamento de contextos.

2. Dispositivos que permitem reconfiguração parcial

A reconfiguração parcial foi implementada primordialmente pelas empresas National, Algotronix e Xilinx, que produziram as famílias de FPGAs Clay [8], Cal1024 [9] e XC6200 [10], 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 permite 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 tabela verdade 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 possuem suporte explícito a reconfiguração parcial. Os FPGAs da família AT40K (Atmel) foram especialmente projetados para dar suporte à tecnologia *Cache Logic*. 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. Se novas funções se fazem necessárias, as antigas são sobrescritas. A implementação de *Cache Logic* é conceitualmente semelhante à de memórias cache [11]. Na *Cache Logic*, somente uma pequena porção do circuito - aquelas funções que são nela carregadas - estão ativas no sistema num dado momento, enquanto as funções não utilizadas permanecem na memória externa. É possível compilar variações de um projeto em tempo real. À medida que novas funções são necessárias, elas podem ser carregadas na *Cache Logic*, substituindo ou complementando a lógica já presente.

A seguir será feita uma descrição um pouco mais detalhada das características da família Virtex, da Xilinx, que foi escolhida para este trabalho pesquisa, dada sua disponibilidade, e por esta possuir uma estrutura de blocos lógicos homogêneos, dispostos em colunas. Esta característica é propícia para reconfiguração parcial porque facilita a desfragmentação e relocação da lógica implementada [12].

Xilinx Virtex

Cada dispositivo Virtex [13] contém blocos lógicos configuráveis (*Configurable Logic Blocks* - CLBs), blocos de entrada/saída (*Input/Output Blocks* - IOBs), blocos de RAM e recursos de relógio. Todos estes módulos são configuráveis através de um arquivo binário de configuração que adapta o roteamento programável e os bits de memória de configuração do dispositivo para que o FPGA implemente a funcionalidade do projeto. O processo de projeto tem como produto final este arquivo binário. Arquivos de configuração contêm 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, conforme ilustrado na Figura 2. Um quadro é a unidade atômica de configuração: é a menor porção de memória de configuração que pode ser lida ou escrita.

Quadros são lidos e escritos sequencialmente, 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 e o bloco de interconexão de *SelectRAM*² podem ser lidos ou escritos com apenas um comando. Cada bloco de conteúdo de *SelectRAM* deve ser lido ou escrito separadamente.

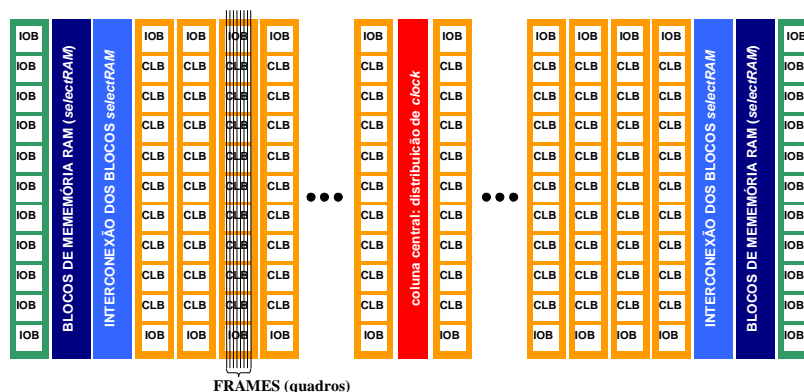


Figura 2 – Organização interna do dispositivo FPGA Virtex. Os elementos internos são organizados em colunas, sendo cada coluna interseccionada por um conjunto vertical de quadros. Por exemplo, as colunas de blocos lógicos (CLB) são interseccionadas por 48 quadros.

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 destes quadros no arquivo de configuração.

Ressalta-se que a unidade atômica de reconfiguração é o quadro, e um quadro intersecciona tantas CLBs quanto for o número de linhas de CLBs do dispositivo. Portanto, toda vez que se fizer uma mudança em uma CLB, essa mudança afetará todas as CLBs de sua respectiva coluna. Isto pode acarretar a perda de informações quando ocorre uma reconfiguração parcial. Para evitar este problema, é imprescindível que se utilize a técnica RMW (*Read-Modify Write*), onde primeiro é realizada a leitura da seqüência de quadros que contém a (s) CLB(s) que serão modificadas (através de *ReadBack*³), logo após é realizada a modificação dos elementos desejados, e somente então o arquivo de configuração parcial é descarregado no dispositivo.

3. Ferramentas para desenvolvimento de hardware parcialmente reconfigurável

A. Java Run-Time Reconfiguration e JBits

JBits é um conjunto de classes Java que fornecem uma API (*Application Programming Interface*) para manipular o arquivo de configuração da família de FPGAs Virtex. Esta interface opera tanto em arquivos de configuração gerados pelas ferramentas de projeto da Xilinx quanto em arquivos de configuração lidos do hardware [14]. O modelo de configuração utilizado pelo programa JBits é uma matriz bidimensional de CLBs. Cada CLB é referenciado por uma linha e uma coluna. Assim, todos os recursos configuráveis no CLB selecionado podem ser configurados ou analisados. Além disso, o controle de todo o roteamento adjacente ao CLB selecionado 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 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*. O suporte à orientação à objetos da linguagem Java permite que *cores* parametrizáveis sejam implementados. Contudo, 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.

² *SelectRAM* - blocos de memória RAM internas ao dispositivo programável.

³ *ReadBack* - ação de ler os dados de um FPGA em um dado momento de execução.

A característica mais importante do JBits 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. No sentido de obter maior vantagem do suporte à reconfiguração parcial, a API JBits foi estendida com a API JRTR. Esta interface provê um modelo de cache onde as mudanças dos dados de configuração são ajustadas, e somente os dados realmente necessários são escritos no dispositivo, ou lidos dele.

A Figura 3 mostra o diagrama de blocos para o sistema JRTR. A aplicação do usuário comunica-se com 4 módulos, responsáveis pelas seguintes tarefas:

- geração de comandos para *Readback*;
- geração e análise do arquivo de configuração: mantém a *cache* de configurações atualizada em relação às modificações efetuadas no arquivo de configuração;
- execução dos procedimentos JBits: leitura e escrita dos arquivos de configuração em disco, ou de outro dispositivo externo, além dos procedimentos de manipulação do arquivo de configuração;
- comunicação com o hardware (servidor): carrega o arquivo de configuração no FPGA e executa os comandos de *Readback* enviados pelo módulo de geração de comandos de *Readback*.

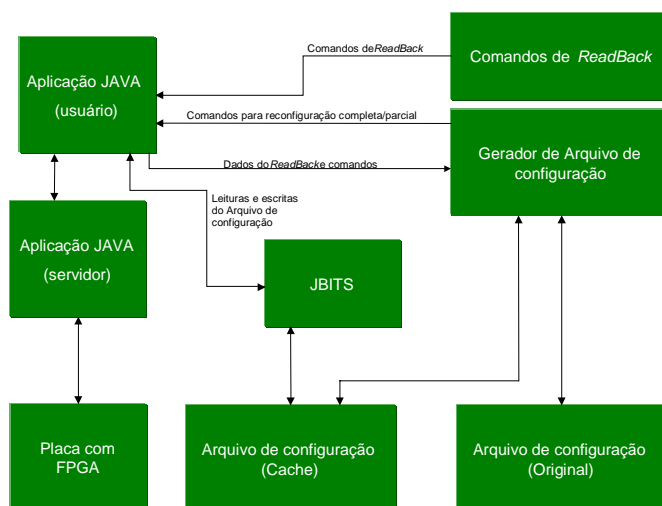


Figura 3 - Diagrama de blocos do JRTR.

A API JRTR provê controle simples, mas completo, da cache de configurações. O usuário pode produzir configurações parciais em qualquer tempo, e então carregá-las no hardware.

B. JHDL

Segundo Bellows e Hutchings [15], durante o desenvolvimento de aplicações para arquiteturas configuráveis (*Configurable Computing Machines - CCMs*), os projetistas destas devem realizar duas tarefas fundamentais. Primeiro, projetar o circuito que implementa a funcionalidade necessária para a aplicação. Isto é tipicamente feito usando ferramentas comerciais de CAD, tais como síntese VHDL, em conjunto com ferramentas *back-end* obtidas junto a fabricantes de FPGAs. Segundo, escrever um programa para supervisionar a operação da aplicação. Nos casos de aplicações em reconfiguração parcial de maior complexidade, este programa de controle pode ser igualmente complexo, carregando uma variedade de configurações e dados, sob demanda, conforme a necessidade da aplicação. Atualmente, o programa de controle e a descrição do circuito devem ser desenvolvidos simultânea e independentemente. O projetista é responsável por garantir que esses dois códigos irão cooperar corretamente, tipicamente através de ciclos de carga, execução e compilação repetidos. Esta divisão entre descrição do circuito e programa de controle é na realidade a divisão da aplicação entre partes estática e dinâmica: a estática representada pela biblioteca de circuitos, e a dinâmica constituída pelo programa de controle, que escolhe configurações de hardware de uma biblioteca, configura o dispositivo e executa a aplicação.

Contudo, devido às vertiginosas mudanças que ocorrem no campo da computação reconfigurável, tratar as partes estática e dinâmica da aplicação é inadequado e limitante. O que é necessário é uma descrição única e integrada que permita ao projetista expressar naturalmente as partes dinâmica e estática da aplicação simultânea e conjuntamente.

Nesse sentido Bellows e Hutchings [15] propuseram uma abordagem de projeto e uma ferramenta de CAD centradas na criação de uma descrição integrada. Seu projeto foi desenvolvido baseado nos seguintes requisitos:

1. A ferramenta deve usar uma linguagem de programação existente, sem extensões. Isto possibilita que um grande número de projetistas possam usar a ferramenta.
2. O paradigma de controle da CCM deve ser o de CCM independente. Detalhes de controle da CCM devem ser elevados a um nível mais alto de abstração, com o objetivo de facilitar o processo de redirecionamento das aplicações para vários dispositivos alvo.
3. O método de descrição deve dar suporte a reconfiguração em tempo de execução, total e parcial.
4. A descrição integrada deve servir para simulação e execução, sem modificações. O sistema JHDL é implementado como um conjunto de bibliotecas de classes Java, com funcionalidade dividida em duas áreas básicas: simulação do circuito e suporte à execução da CCM. As classes referentes ao suporte da execução provêm acesso transparente às funções de controle da CCM via mecanismos de construção/destruição.

Projetistas desenvolvem circuitos em JHDL selecionando um conjunto de elementos síncronos ou combinacionais, e ligando-os de modo a formar um circuito síncrono arbitrário. Existem três classes diferentes que podem ser utilizadas para implementar um circuito: *CL* (combinacional), *Synchronous* (síncrono) e *Structural* (interconexão entre elementos síncronos e combinatórios). No uso de cada classe, o projetista define uma nova classe que herda características da classe apropriada e implementa a funcionalidade desejada no construtor e em outros métodos. Circuitos individuais são interconectados instanciando objetos *Wire* e passando esses objetos como argumentos para os construtores. Maiores informações a respeito do status atual do projeto JHDL podem ser obtidas no endereço <http://www.jhdl.org/>

C. Dynasty

A ferramenta proposta Vasilko em [16] implementa uma biblioteca para projeto de sistemas RTR. A implementação é baseada na abstração de RTR, no sentido da utilização de uma planta-baixa temporal, que permite manipulação do projeto nas dimensões espacial e temporal.

Projeto de planta-baixa (*floorplanning*) no nível de disposição (*layout*) é uma técnica comum em fluxos de projetos para FPGAs/ASICs, a qual é utilizada para definir ou modificar posições espaciais de módulos de projeto, visando incrementar o desempenho ou a eficiência na implementação do projeto. Para fins de diferenciação, este tipo de projeto de planta-baixa será doravante denominado planta-baixa espacial.

Uma planta-baixa espacial de um projeto estático (que não sofrerá reconfiguração) permanece inalterada durante todo o tempo de vida do projeto. Em sistemas RTR, contudo, a presença e a posição espacial de cada módulo de projeto reconfigurado podem mudar com o tempo. Em cada instante durante o tempo de execução, o projeto de planta-baixa espacial é determinado pela sua coordenada num eixo de projeto temporal.

Um eixo definido pela coordenada temporal corresponde ao projeto de planta-baixa temporal para uma configuração de projeto.

Os autores desta ferramenta chamaram o processo de transformação do projeto de alto-nível na forma comportamental para sua implementação em projetos de planta-baixa espacial posicionados num eixo temporal de projeto de planta-baixa temporal. Este processo combina dois problemas de fluxo de projeto em uma fase: (i) particionamento e seqüenciamento de módulos de projeto em configurações de projeto (também chamadas de particionamento temporal), e (ii) posicionamento temporal de módulos de projeto e conexão com a área reconfigurável de cada configuração.

Com o intuito de permitir a exploração de projeto espacial das porções RTR do projeto, o projeto de planta-baixa temporal precisa resolver outros problemas relacionados: (iii) escalonamento (particionamento do passo de controle), (iv) alocação de unidades funcionais, e (v) alocação de registradores.

O projeto de planta-baixa temporal fornece ao projetista uma melhor abstração de projeto de espaço para sistemas RTR. O fluxo de projeto convencional não provê visualização do espaço, além de tornar sua exploração uma tarefa difícil. Por outro lado, projeto de planta-baixa temporal é bem ajustado para visualização de projeto para dimensões espacial ou temporal, em 2D ou 3D.

4. Uma Proposta de Reconfigurador parcial

Esta Seção apresenta o reconfigurador de *cores* desenvolvido, o qual possibilita a reconfiguração parcial e remota de *cores* em dispositivos Virtex. Para tanto estudaram-se duas possibilidades. A primeira foi a utilização de classes do

JBits. Conforme visto, JBits facilita tarefas como leitura, escrita e localização de elementos em um arquivo de configuração. Contudo, por ser uma ferramenta com código fonte fechado, não permite a exploração, investigação e melhoria de suas funcionalidades. A segunda abordagem foi a criação de um programa que tivesse funcionalidades semelhantes ao JBits, mas cujo código-fonte será disponibilizado para o meio acadêmico. O segundo enfoque permitiu ainda compreender a estrutura interna dos dispositivos da família Virtex, o que contribuirá para o desenvolvimento de novas ferramentas para reconfiguração parcial e remota, e futuramente dinâmica.

O objetivo deste reconfigurador é alterar um arquivo binário de configuração de um dispositivo da família Virtex utilizando as classes do JBits. A Figura 4 mostra um diagrama em alto nível do projeto. Apesar de ser um aplicação extremamente simples, permite demonstrar as possibilidades advindas da reconfiguração. Suponha que o *core* seja um circuito para telecomunicação, com uma dada codificação de parâmetros armazenada em memória (LUT *SelectRAM*). Pela simples alteração destes bits de configuração pode-se alterar toda a funcionalidade do circuito. O passo seguinte à esta aplicação de demonstração será de alterar não mais LUT RAMs, mas sim substituir dinamicamente *cores*, conforme será apresentado na Seção 5.

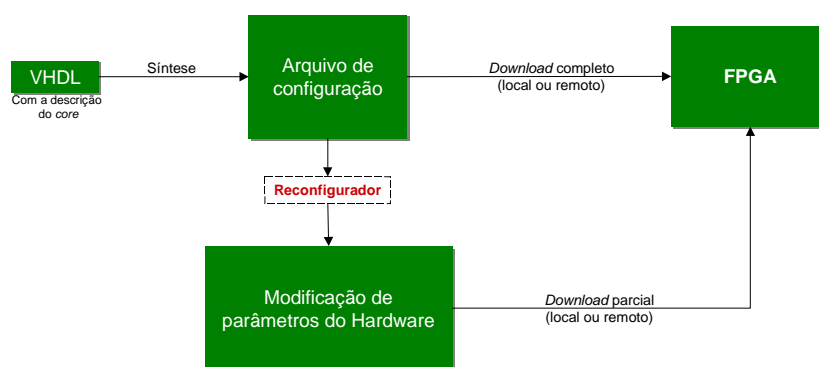


Figura 4 – Diagrama em alto nível do reconfigurador parcial.

Uma aplicação de demonstração foi escrita em VHDL. A finalidade desta aplicação é ler os bits armazenados na memória (LUT *SelectRAM*), acendendo ou apagando *leds* da placa de prototipação de acordo com a informação lida. Para que a LUT *SelectRAM* pudesse ser alterada, restringiu-se sua localização no dispositivo através do arquivo de restrições do usuário.

A descrição foi então sintetizada para o dispositivo alvo e o arquivo binário de configuração foi gerado. Logo após, o arquivo de configuração foi carregado no FPGA, obtendo-se nos *leds* os valores padrão.

O próximo passo foi utilizar as classes do JBits para alterar os bits da memória LUT *SelectRAM*, de maneira a acender somente os *leds* desejados. Para isto, criou-se uma aplicação Java que recebe como parâmetros o arquivo binário a ser alterado, e a posição da CLB a ser modificada, e o nome do arquivo a ser gravado com as alterações.

A aplicação abre o arquivo binário e permite que o usuário visualize e altere a configuração atual de determinada CLB. Ela também tem um método que procura e mostra todas as CLBs com configurações diferentes da configuração inicial.

A última função implementada na ferramenta de reconfiguração foi o download através da própria ferramenta. Para isso utilizou-se métodos nativos de Java para a chamada de um código feito em C. Este código comunica-se com a plataforma de prototipação através de comandos à interface de programação JTAG. Com a utilização de *Sockets* em Java é realizado o *download* a partir de uma conexão TCP/IP em uma máquina servidora que estará sempre aceitando conexões de um cliente.

Esta ferramenta atualmente encontra-se operacional para as funcionalidades reconfiguração total e remota. Em paralelo, está sendo feita a geração de arquivos para configuração parcial. A partir de mudanças no arquivo de configuração inicial, é feita uma comparação com o novo arquivo de configuração que contém as últimas modificações, gerando o arquivo de configuração parcial. Para que o download do arquivo de configuração parcial seja efetuado com sucesso é ainda necessário o cálculo de CRC deste.

5. Proposta de interface para conexão de cores em dispositivos Virtex

Como visto, é possível alterar parcialmente a funcionalidade do hardware sem interromper o funcionamento do sistema. Para que este tipo de aplicação seja de fato uma realidade em projetos de hardware é necessário definir uma estratégia de reconfiguração parcial dinâmica para os dispositivos FPGA. A idéia é que haja no FPGA uma estrutura análoga a uma interface PCI em um PC, onde conectam-se dispositivos sem modificação do sistema.

Até o presente momento não foi encontrada na literatura referência sobre uma proposta de interface intra-FPGA para que dispositivos reconfiguráveis possam receber *cores* de forma modular. Essa modularidade consiste em conectar e remover *cores* sem que haja necessidade de maiores alterações na lógica pré-existente no FPGA. Há técnica semelhante, voltada para o projeto de *cores* para ASICs. Como exemplo podem ser citados o *CoreConnect* da IBM [16] e o *MessageBuffer* [18].

Um ponto de partida para a definição desta interface de comunicação entre diversos *cores* é estabelecer uma porção de hardware fixo (estático) no FPGA responsável pela comunicação com o mundo externo e um barramento para conexão dos *cores*, provido de *pinos virtuais*. A Figura 5 ilustra esta proposta.

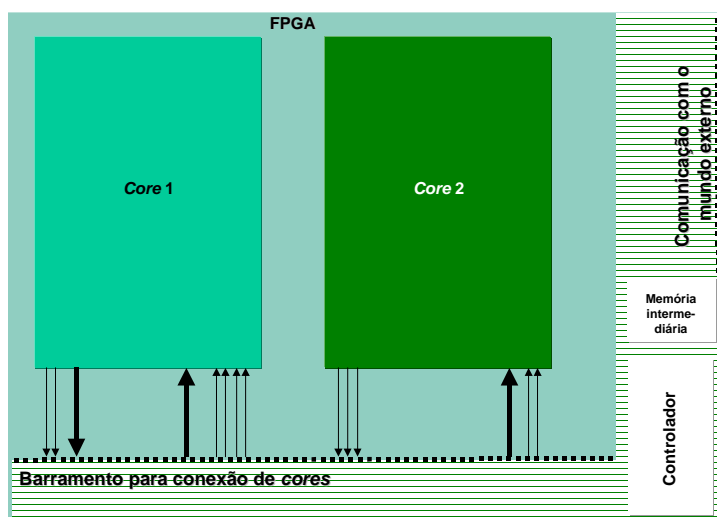


Figura 5 - Interface para conexão de cores.

Este barramento para conexão entre *cores*, denominado *interface*, seria também um módulo de hardware, porém estático. O FPGA seria inicialmente carregado apenas com o controlador e esta *interface*. O controlador comunica-se externamente com os pinos de entrada/saída do FPGA, e internamente com os sinais de entrada/saída do(s) *core(s)*, através de *pinos virtuais*. Isto significa que os módulos de hardware a serem conectados, *cores*, somente comunicar-se-ão com o mundo externo através desta *interface*. Os *cores* serão carregados em tempo de execução, possuindo conexões aos sinais da interface. O controlador da interface será o árbitro deste barramento, gerenciando o aceite ou a rejeição de conexões, bem como o controle de conflitos relativos ao acesso aos pinos de entrada/saída do FPGA.

Em [3] é apresentado um sistema de reconfiguração utilizando como estudo-de-caso um sistema para processamento de imagens. Este sistema sofre quatro reconfigurações para cada conjunto de dados (imagem) a ser processado. O FPGA primeiramente armazena o sinal de vídeo em uma memória, aplicando em seguida duas transformações diferentes sobre as imagens, e finalmente torna-se um modem e transmite o resultado do processamento. Este exemplo poderá ser implementado utilizando a proposta de interface de conexão de *cores*, sendo cada uma das quatro configurações um *core* comunicando-se com a *interface*.

Julga-se essencial o desenvolvimento desta interface de comunicação de *cores* para que seja possível a criação de ferramentas de projeto para reconfiguração dinâmica de sistemas digitais.

6. Conclusões e trabalhos futuros

Este trabalho revisou o estado da arte em ferramentas para reconfiguração de hardware. Através do desenvolvimento de uma ferramenta de reconfiguração mostrou-se a viabilidade da utilização desta técnica em

projeto de sistemas digitais. A reconfiguração, total ou parcial, executada remotamente abre novas perspectivas para o projeto de sistemas digitais, pois passa a permitir praticamente a mesma flexibilidade no hardware quanto a que existe no software. Trata-se agora de reconfigurar módulos completos de hardware, não simples bits de memória. A configuração de módulos completos permitirá a inserção e remoção de diferentes blocos funcionais, da mesma forma que um sistema de gerenciamento de memória virtual opera com partes de um determinado programa.

O principal trabalho que deve ser desenvolvido para viabilizar a reconfiguração parcial e dinâmica de cores em FPGAs, na opinião dos autores deste artigo, é o aprofundamento da pesquisa a respeito da interface para conexão de *cores*. A seqüência natural do projeto, em um nível mais alto de abstração, é a definição de um estudo-de-caso sobre reconfiguração dinâmica, e a criação de mecanismos para sua implementação.

Agradecimentos

O autor Fernando Gehm Moraes agradece o suporte do CNPq (projeto integrado número 522939/96-1) e da FAPERGS (projeto número 96/50369-5). O autor Ney Laert Vilar Calazans agradece o suporte do CNPq (projeto integrado número 520091/96-5) e da FAPERGS (projeto número 99/1555-3).

Referências

- [1] HADLEY, John D.; HUTCHINGS, Brad L. **Design methodologies for partially reconfigured systems**. In: IEEE Workshop on Fpgas For Custom Computing Machines, pp.78-84, California, Estados Unidos, 1995.
- [2] XILINX. **Virtex series configuration architecture user guide**. Disponível em <http://www.xilinx.com/xapp/xapp151.pdf> Setembro, 2000.
- [3] J. Villasenor and W. H. Mangione-Smith. **Configurable Computing**. Scientific American, pp. 54-59, June 1997.
- [4] GUCCIONE, Steve. **List of FPGA-based computing machine**. Disponível em http://www.io.com/~guccione/HW_list.html Agosto, 2000.
- [5] LO, Hing Kai. **Info about reconfigurability**. Comunicação pessoal via e-mail, 2000.
- [6] WIRTHLIN, Michael J.; HUTCHINGS, Brad L. **Improving computational density using run-time circuit reconfiguration**. In: IEEE Transactions on VLSI Systems, pp.247-256, Nova Iorque, EUA, 1998.
- [7] A. Dehon, **DPGA – Coupled Microprocessors: Commodity Ics for the Early 21st Centur**. IEEE Workshop on FPGA's Custom Computing Machines – FCCM '93, D. A . Buell and K. L. Pocek, Napa, CA, pp. 202-211, April, 1993.
- [8] NATIONAL. **Navigator**. Disponível por WWW em <http://www.national.com/appinfo/milaero/files/f61.pdf>, Julho, 1998.
- [9] ALGOTRONIX. **Cal1024 datasheet**. Disponível em http://dec.bournemouth.ac.uk/drhwl_lib/archive/cal1024.ps.gz , 1989.
- [10] XILINX, **XC6200 datasheet**. Disponível por WWW em http://dec.bournemouth.ac.uk/drhwl_lib/archive/xc6200.pdf. Junho, 1999.
- [11] ATMEL. **Implementing Cache Logic with FPGAS**. Disponível em <http://www.atmel.com/atmel/postscript/doc0461.ps.zip>. 2000.
- [12] COMPTON, Katherine. **Programming architectures for run-time reconfigurable systems**. Dissertação de Mestrado. Washington State University, Estados Unidos. Disponível em <http://www.ee.washington.edu/faculty/hauck/publications-/KatiThesis.pdf>, 1999.
- [13] Xilinx Inc. **Virtex 2.5V Field Programmable Gate Arrays (DS003)**. Virtex Series datasheet, 2000.
- [14] MCMILLAN, Scott; GUCCIONE, Steven A. **Partial Run-Time Reconfiguration Using JRTR**. In: Proceedings Of Field-Programmable Logic And Applications, Glasgow, Escócia, 1999.
- [15] BELLOWS, Peter; HUTCHINGS, Brad. **JHDL-an hdl for reconfigurable systems**. In: Proceedings Of The IEEE Symposium on Field-Programmable Custom Computing Machines, pp.124_133, 1998.
- [16] VASILKO, Milan. **Dynasty: a temporal floorplanning based cad framework for dynamically reconfigurable logic systems**. In: Proceedings of Field-Programmable Logic and Applications, pp.124_133, Glasgow, Escócia. 1999.
- [17] IBM. **The CoreConnect™ Bus Architecture**. Disponível em http://www.chips.ibm.com/products/coreconnect/docs/crcon_wp.pdf, 2000.
- [18] NOTBAUER, J., ALBRECHT, T., NIEDRIST, G.,ROHRINGER, S. **Verification and Management of a multimillion-gate embedded core design**. IEEE/ACM Design Automation Conference, New Orleans, Louisiana, USA, 1999.