

Reconfiguração Parcial e Remota de Dispositivos FPGA da Família Virtex

Daniel Mesquita, Fernando Moraes, Leandro Moller, Ney Calazans

Faculdade de Informática – PUCRS
Av. Ipiranga, 6681 – Porto Alegre – CEP: 90619-900
{dmesquita, moraes, moller, calazans}@inf.pucrs.br

Abstract: *This work addresses partial and remote reconfiguration of FPGAs, having as main objective to provide the basis for research and development in the field of dynamically, partially and remotely reconfigurable systems. The main idea is to bring to hardware mechanisms similar to those used in virtual memory, i.e., hardware virtualization. It presents an analysis of FPGA device (VIRTEX) and tools (Jbits) that enable partial and remote reconfiguration of FPGAs. Using in-house developed software, results of applying a complete and remote reconfiguration process are presented. Extensions of the tools to deal with partial reconfiguration are under development.*

Keywords: *reconfiguration, partial reconfiguration, dynamic reconfiguration, FPGA, prototyping.*

1. Introdução

Muitas aplicações emergentes em comunicação, computação e eletrônica necessitam que suas funcionalidades permaneçam flexíveis mesmo depois de o sistema ter sido manufaturado [2]. 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 FPGAs têm introduzido suporte para modificações rápidas em tempo de execução do hardware do sistema [7]. 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 sequencialmente [6]. 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-verificação e reconfigurar a si mesmos, substituindo elementos defeituosos por elementos reserva disponíveis.

Este artigo está organizado da seguinte forma. A Seção 2 descreve de forma sucinta a arquitetura dos dispositivos VIRTEX, salientando as características que são importantes para este trabalho. A Seção 3 apresenta inicialmente as classes JBITS, que são utilizadas como base para a construção da ferramenta de reconfiguração remota. A Seção 4 apresenta um estudo de caso utilizando a ferramenta desenvolvida. Finalmente, na Seção 5, as conclusões e sugestões de trabalhos futuros são apresentadas.

2. Arquitetura Virtex

Os FPGAs da família Virtex contém blocos lógicos configuráveis (do inglês *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. Cada CLB possui recursos para o roteamento local e conexão com a matriz de roteamento geral (GRM). Um anel de roteamento periférico, denominado de *VersaRing* permite um roteamento adicional com os blocos de entrada e saída (IOBs). Esta arquitetura apresenta blocos de memória RAM dedicados (BRAMs) de 4096 bits cada um, e conta com 4 a 8 blocos dedicados, que implementam as funções de DLLs para o controle, distribuição e compensação de atrasos do relógio.

A funcionalidade deste dispositivo é determinada através do arquivo de configuração, denominado *bitstream*. 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.

Os dispositivos VIRTEX têm a arquitetura interna organizada em colunas, podendo estas serem lidas ou escritas individualmente. Logo, é possível reconfigurar parcialmente esses dispositivos através da modificação destas colunas no arquivo de configuração.

2.1 Arquitetura interna de um FPGA Virtex

A regularidade na distribuição dos elementos computacionais que compõem a arquitetura interna da família Virtex permite ações de relocação e desfragmentação que são importantes para reconfiguração parcial [1]. Na Figura 1 pode ser visto que CLBs, IOBs, *BlockRAMs*, recursos de relógio são distribuídos pelo dispositivo de forma regular, em colunas de elementos configuráveis.

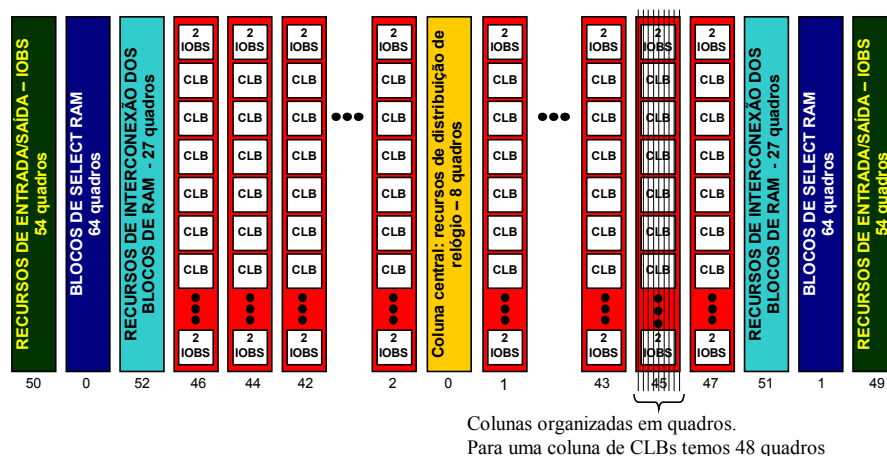


Figura 1 - Disposição em colunas dos elementos do FPGA Virtex XCV300.

Ainda na Figura 1 podem ser vistos os quadros, sendo o quadro a unidade mínima de reconfiguração. Cada CLB é composta por 48 quadros sucessivos. Como a disposição das CLBs é em colunas, a modificação de uma CLB implica na alteração de todas as CLBs da coluna a que pertence, logo não é possível reconfigurar uma CLB individualmente. Note-se

ainda na mesma Figura que as colunas são numeradas a partir do 0 (zero) – atribuído à coluna central. As demais colunas são numeradas em ordem crescente, com valores pares à esquerda da coluna central, e ímpares à sua direita.

Cada CLB possui dois conjuntos de funções idênticos, denominados *slices*. A Figura 2 exibe a estrutura interna de uma CLB, onde cada *slice* contém duas tabelas de busca (*LookUp Tables*, ou LUTs) denominadas F e G, dois flip-flops, um buffer tri-state, circuito de propagação rápida de vai-um, além de recursos de roteamento. Cada LUT é composta por 16 bits de configuração, podendo ser configurada como qualquer função booleana de até 4 variáveis, ou memória de porta simples ou porta dupla (neste caso a LUT é denominada LUTRAM).

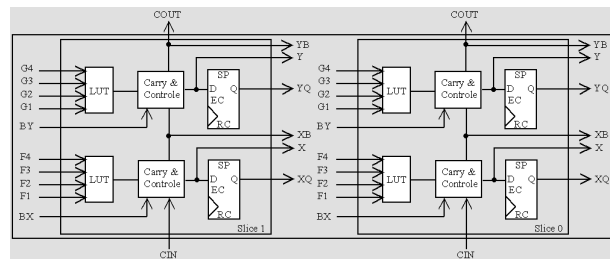


Figura 2 - Estrutura interna de um CLB do FPGA Virtex XCV 300.

Os bits da memória de configuração são agrupados em quadros verticais com um bit de largura, que se estendem do topo à base da matriz. 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, permitindo assim reconfiguração parcial. Cada CLB é cortado por 48 quadros, utilizando 18 bits de cada um. Desta forma, um CLB é completamente configurado por 864 bits.

Nos arquivos de configuração dos FPGAs da família Virtex as informações são gravadas em palavras de 32 bits. Por exemplo, o dispositivo XCV300 possui 48 linhas de CLBs, o que corresponde a 20 palavras de 32 bits. Como deve haver uma palavra vazia ao final de cada quadro, tem-se então 21 palavras de 32 bits por quadro.

Calculando-se o total de quadros, pode-se chegar ao número de bits necessários à configuração de um FPGA Virtex. Este total é dado pela Equação 1:

$$T_{quadros} = (IOB + RAM + RAMConnect) * 2 + (Chip_{cols} * 48) + relógio + 3 \quad (1)$$

Onde:

- IOB é o número de quadros por coluna de IOBs (54);
- RAM é o número de quadros por coluna de RAM (64);
- RAMConnect é o número de quadros por coluna de RAMConnect (27);
- A constante 2 significa que há dois conjuntos de IOB+RAM+RAMConnect (um à esquerda e outro à direita do FPGA);
- $CHIP_{cols}$ é o número de colunas de CLBs do dispositivo (48 para o dispositivo XCV300);
- Relógio é a coluna central da Virtex, que possui 8 quadros;
- A constante 3 significa que há um quadro de preenchimento depois de cada coluna de RAM, e outro quadro para complementar as colunas de CLBs.

Substituindo as constantes na Equação 1, tem-se que o total de quadros do FPGA XCV300 é de 2605. Como cada quadro possui 21 palavras de 32 bits, o XCV300 possui 1.750.560 bits de configuração, mais palavras de comando e sincronização.

2.2 Arquivo de configuração e endereçamento dos elementos internos

O *bitstream* é composto por um fluxo de palavras que segue o protocolo de configuração determinado pelo fabricante para o dispositivo [9]. Este protocolo de configuração é composto por uma coleção de registradores de 32 bits. A lógica de configuração é controlada e acessada através desses registradores.

A identificação dos registradores de comandos e do formato da palavra de dados para cada um deles é necessária para que seja possível a modificação dos parâmetros para uma reconfiguração parcial.

Para que seja factível uma reconfiguração parcial, além do domínio da escrita nos registradores de um FPGA, faz-se necessária a possibilidade de localizar determinados bits no arquivo de configuração. A tarefa a princípio é árdua, pois que somadas todas as palavras de um *bitstream* e multiplicadas por sua largura em bits (32), tem-se que localizar um bit dentre 1.751.808 bits!

Analisando as equações apresentadas em [9], é possível estabelecer um roteiro para localização de um determinado bit de uma CLB no arquivo de configuração. Como consequência disto, é possível a leitura de um conjunto de elementos. Por exemplo, dada uma aplicação qualquer, pode ser necessária a modificação do bit 14 de uma F-LUT. A Figura 3 mostra que esse bit está dentro da fatia 0 (S0) da CLB situada intersecção entre a primeira linha (R1) e a primeira coluna (C1) de um FPGA XCV100. O posicionamento de uma CLB em coordenadas pré-definidas pode ser obtido através da edição do arquivo de restrições de usuário (UCF, do inglês *User Constraints File*). As coordenadas de uma CLB são dadas, pois, por Linha (*Row*), Coluna (*Column*) e Fatia (*Slice*), no seguinte formato: R1C1.S0.

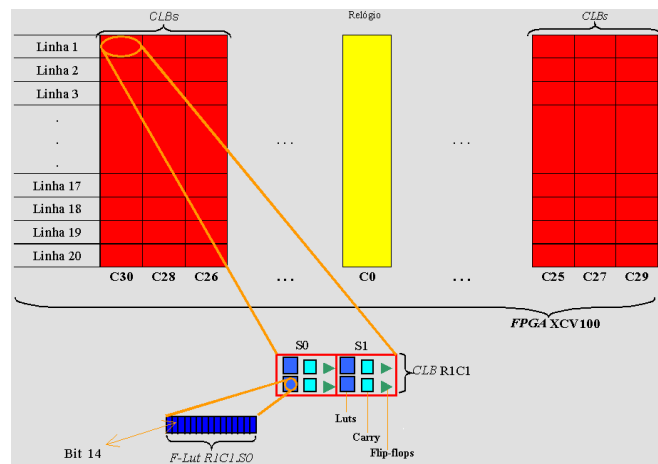


Figura 3 - Localizando o bit 14 na CLB dada por R1C1.S0 de um FPGA XCV100.

A localização de um dado elemento, por exemplo um bit dentro de uma LUT, é feito através dos seguintes passos:

- localização da coluna onde se encontra a CLB;
- localização do quadro onde se encontra o bit desejado, em função do slice e do número do bit;
- localização do bit no quadro, em função da linha (row), e se é LUT G ou LUT F;
- localização de qual palavra no quadro o bit pertence;
- localização de qual bit na palavra corresponde ao bit que se deseja manipular.

Depois de localizado o bit desejado, pode-se proceder a reconfiguração de forma total ou parcial. Reconfigurar o FPGA totalmente é tarefa trivial, uma vez que, após localizar o bit (ou os bits) que se deseja(m) mudar, se faz necessário apenas o recálculo de CRC para

geração do novo *bitstream*, sem qualquer modificação no protocolo de configuração.

Contudo, para uma reconfiguração parcial, o fluxo de configuração para geração do *bitstream* é diferente, e não está completamente descrito na documentação do fabricante. Porém, depois de dominado o novo protocolo a tarefa torna-se igualmente simples. Atualmente o GAPH está desenvolvendo uma ferramenta que gere automaticamente o *bitstream* parcial.

3. Reconfiguração de dispositivos FPGA

3.1 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 [4].

O modelo de programação utilizado pelo JBits também usa a abstração de uma matriz de CLBs. Assim, todos os recursos configuráveis na CLB selecionada podem ser configurados ou analisados. Além disso, o controle de todo o roteamento adjacente a CLB selecionada torna-se disponível.

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*¹. O suporte à orientação a objetos da linguagem Java permite que *cores* parametrizáveis sejam implementados.

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 o danifiquem. 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 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 interface do JBits existente é ainda utilizada para ler e escrever arquivos de configurações do disco, ou de outro dispositivo externo. Mas o “Gerador de Arquivos de configuração” do JRTR é utilizado para analisar o arquivo de configuração, e para manter a imagem dos dados e as informações de acesso.

3.2 Ferramenta para reconfiguração remota e completa

Todo o trabalho de reconfiguração é feito sobre o *bitstream*, ou seja, com *hard cores* e não *soft cores*. A primeira ferramenta desenvolvida para reconfiguração de dispositivos FPGA teve três objetivos, plenamente alcançados:

- Permitir a **alteração** de valores armazenados em memória interna do FPGA (LUTRAM). A alteração de valores armazenados em LUTRAM permite modificar o comportamento do

¹ Representação eletrônica de um componente em nível de processamento; módulos pré-projetados e pré-validados de hardware.

circuito pela alteração dos parâmetros de configuração do circuito.

- Que esta alteração pudesse ser feita de forma **remota**. Se a configuração do FPGA pode ser realizada remotamente, isto pode significar uma grande redução de custos para o fabricante do dispositivo. Por exemplo, um determinado fabricante detecta um erro em seu circuito. Se reconfiguração remota é utilizada, ele pode atualizar o hardware de todos os seus clientes a distância, sem que estes percebam que o sistema foi modificado.
- "**Esconder**" do usuário final a arquitetura do sistema. Devido à complexidade na manipulação de *bitstreams* uma camada de software é acrescentada, de forma que a aplicação final (na forma de uma *Applet*) mostre ao usuário apenas os parâmetros que devem ser alterados. A Figura 4 ilustra em (a) os comandos para gerar a aplicação de configuração de uma determinada aplicação, e em (b) a aplicação final vista pelo usuário. Na aplicação de reconfiguração tem-se basicamente os parâmetros a inserir, o formato dos parâmetros e a leitura e *download* do *bitstream*.

```
<html>
<head><title>DropInsert
</title></head>
<body bgcolor="#00003B">
<APPLET code="BITGenerico.class" width=400 height=600>
<PARAM name="nrosinais" value="18">
<PARAM name="caminho" value="moller.bit">
<PARAM name="ip" value="200.17.93.83">
<PARAM name="porta" value="5000">
<PARAM name="I[1]" value="f2048,bin,1,1,F,0,0,0">
<PARAM name="I[2]" value="rst,bin,1,1,F,0,1,1">
<PARAM name="I[3]" value="f100m,bin,1,1,F,0,2,2">
<PARAM name="I[4]" value="code,bin,1,1,F,0,3,3">
<PARAM name="I[5]" value="CRCControl,bin,1,1,F,0,4,4">
<PARAM name="I[6]" value="insertServ,hex,1,1,F,0,0,3">
<PARAM name="I[7]" value="servicoIn,hex,1,1,F,0,4,7">
<PARAM name="I[8]" value="n64,hex,1,1,F,0,8,11">
<PARAM name="I[9]" value="ss,hex,1,1,F,0,12,15">
<PARAM name="I[10]" value="dataInsert,hex,1,1,F,0,0,15">
<PARAM name="I[11]" value="dataDrop,hex,1,1,F,0,0,15">
<PARAM name="I[12]" value="fn64,hex,1,1,F,0,0,15">
<PARAM name="I[13]" value="synchronized,hex,1,1,F,0,0,15">
<PARAM name="I[14]" value="OutDecodPlus,hex,1,1,F,0,0,15">
<PARAM name="I[15]" value="OutDecodMinus,hex,1,1,F,0,0,15">
<PARAM name="I[16]" value="servicoOut,hex,1,1,F,0,0,15">
<PARAM name="I[17]" value="servicoClk4khz,hex,1,1,F,0,0,15">
<PARAM name="I[18]" value="slotPattern,hex,1,1,F,0,0,15">
</APPLET>
</body>
</html>
```

(a)



(b)

Figura 4 - (a) Comandos para geração da *Applet* e (b) aplicação de reconfiguração.

A ferramenta para reconfiguração remota e completa de FPGAs foi desenvolvida em Java. Ela permite acessar remotamente os *bitstreams* e visualizar suas configurações. Possibilita modificação de LUTs, procura de uma LUT com uma configuração específica e visualização da listagem de todas as LUTs com configurações diferentes da configuração padrão. Após modificar o *bitstream* é possível salvá-lo e baixá-lo remotamente para a placa sem a necessidade de nenhum software extra. Essa ferramenta foi implementada seguindo o paradigma cliente-servidor.

O lado servidor da aplicação é responsável por atender os clientes e executar todas as funções citadas anteriormente e passar via *sockets* as respostas para os clientes. Também é

responsabilidade do servidor salvar um arquivo de configuração e realizar a chamada ao programa JTAGProgrammer, que realiza o download da configuração para o dispositivo alvo.

Os requisitos para a instalação do servidor é ter o conjunto de classes JBits na máquina, um servidor de páginas para disponibilizar o *Applet* (Apache), o JTAGProgrammer e o JDK 1.2 (ou versão superior) para rodar a aplicação em Java.

Existem duas versões de clientes: uma aplicação Java e outra na forma de *Applet*. A vantagem da *Applet* é que não necessita ter na máquina cliente o JDK instalado; basta um navegador de internet que provenha uma máquina virtual JAVA. Em contrapartida, em função das restrições para acesso a recursos locais do cliente pelo *Applet* [3], não é possível transmitir via esta ferramenta um arquivo de configuração do cliente para o servidor. Portanto o *Applet* possibilita que remotamente se façam modificações em *bitstreams* previamente colocados na máquina servidora.

Através da versão aplicação é possível transmitir um arquivo ao servidor, e realizar todas funcionalidades possíveis da ferramenta. Contudo, há necessidade de ter o JDK instalado na máquina do cliente, além de a aplicação ter de ser disparada via linha de comando (DOS).

A Tabela1 ilustra os principais métodos disponíveis no Jbits para manipular um *bitstream*.

Tabela 1 - Comandos Jbits.

Comando	Descrição
Jbits jbits	Criar instância de JBits para posterior chamadas dos métodos JBits
jbits.read("entrada.bit")	abrir um <i>bitstream</i>
jbits.get(row, col, LUT.SLICE0_F)	capturar o valor de uma LUT
jbits.set(row, col, LUT.SLICE0_F, val)	gravar um novo valor em uma LUT
jbits.write("saida.bit")	salvar um <i>bitstream</i>

Uma observação importante é quanto ao endereçamento das CLBs por parte das classes JBits e da documentação da Xilinx. Por exemplo, no FPGA XCV300 as CLBs são numeradas de (1,1) a (32,48) conforme [9]. No, entretanto, JBits as CLBs são numeradas de (31,0) a (0,47).

A documentação do JBits menciona a possibilidade e indica classes e métodos para geração de um *bitstream* parcial. Contudo, experimentos realizados mostraram que a estrutura do arquivo *bitstream* parcial gerado estava incorreto em relação ao protocolo padrão do fabricante, o que impediu de continuar a utilizar JBits para a geração de arquivos parciais.

3.3 Ferramenta para reconfiguração parcial

A reconfiguração completa é o primeiro passo para a "virtualização" do hardware. Este termo significa que módulos de hardware podem ser alternadamente inseridos e removidos de um FPGA em operação, exatamente como ocorre com sistemas de memória virtual. Para isto é necessário que o dispositivo FPGA permita reconfiguração parcial e dinâmica.

Encontra-se em fase de desenvolvimento no âmbito do grupo de pesquisa GAPH (PUCRS) uma ferramenta escrita em JAVA, sem a utilização das classes JBits, que permite reconfiguração parcial de FPGAs da família Virtex.

Baseada em um estudo minucioso da arquitetura interna de FPGAs da família Virtex, e de seu arquivo de configuração, essa ferramenta realiza as mesmas operações citadas na

Subseção 3.2, além de gerar um *bitstream* parcial. Esse *bitstream* parcial foi pré-validado através de sua inserção em um arquivo de configuração completo, onde substituiu parte da lógica do *bitstream* original (completo).

As etapas de análise do formato do *bitstream* [9][8], estudo do protocolo de *download*, geração do *bitstream* parcial estão concluídas. O trabalho que segue a este é a validação do *bitstream* parcial na placa de prototipação. Esta fase ainda não foi realizada devido à ausência de ferramenta de *download* que permita a configuração parcial sem reinicialização do dispositivo programável.

4. Conclusão e trabalhos futuros

Este trabalho mostrou como reconfiguração pode ser utilizada em dispositivos FPGA. A reconfiguração permite a redução de custos no momento da manutenção/atualização do hardware e a redução de custos pela utilização de dispositivos menores através da virtualização do hardware. Para que seja possível a utilização de reconfiguração é necessário preencher dois requisitos: existência de hardware que suporte reconfiguração e ferramentas de projeto para reconfiguração. Apresentou-se um dispositivo que permite reconfiguração, FPGAs VIRTEX, e desenvolveu-se um conjunto de ferramentas para reconfiguração. Um reconfigurador remoto e completo está operacional, enquanto que a reconfiguração parcial está em desenvolvimento.

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 de uma interface para conexão de *cores* [5].

5. Bibliografia

- [1] 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.
- [2] 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.
- [3] MATAYOSHI, Cláudio; RUGGIERO, Wilson V. **Modelo de segurança da linguagem JAVA**. Anais do XVI Simpósio Brasileiro de Redes de Computadores, 1998.
- [4] MCMILLAN, Scott; GUCCIONE, Steven A. **Partial Run-Time Reconfiguration Using JRTR**. In: Proceedings Of Field-Programmable Logic And Applications, Glasgow, Escócia, 1999.
- [5] MESQUITA, Daniel G.; MORAES, Fernando G. **Implementação de sistemas digitais reconfiguráveis parcial, remota e dinamicamente**. Disponível por WWW em <http://www.inf.pucrs.br/~dmesquita/interest/itens/pep.ppt>. Novembro de 2000.
- [6] VILLASENOR, J.; MANGIONE-SMITH, W. H. **Configurable Computing**. Scientific American, pp. 54-59, June 1997.
- [7] XILINX Inc. **Virtex 2.5V Field Programmable Gate Arrays (DS003)**. Virtex Series datasheet, 2000.
- [8] XILINX Inc. **Virtex FPGA Series Configuration and Readback**. Disponível por WWW em <http://www.xilinx.com/xapp/xapp138.pdf>, Setembro de 2000.
- [9] XILINX Inc. **Virtex series configuration architecture user guide**. Disponível por WWW em <http://www.xilinx.com/xapp/xapp151.pdf>, Setembro de 2000.