

# **CONTROLE DE CONFIGURAÇÕES EM SISTEMAS DINÂMICA E PARCIALMENTE RECONFIGURÁVEIS**

*Ewerson L. S. Carvalho, Frederico B. Möller, Fernando G. Moraes, Ney L. V. Calazans,*

Pontifícia Universidade Católica do Rio Grande do Sul – PUCRS  
Faculdade de Informática – FACIN  
Av. Ipiranga, 6681 - Prédio 30 / Bloco 4 Telefone: +55 51 3320-3611 - Fax: +55 51 3320-3621  
90619-900 - Porto Alegre - RS - BRASIL

{ecarvalho,frebm,moraes,calazans}@inf.pucrs.br

## **SUMMARY**

Reconfigurable systems where the hardware can be changed during execution time have the potential to provide hardware with the same flexibility as software. These may, at the same time, achieve better performance and smaller size. However, there is a clear lack of support devices, tools and design flows adequate for such systems. One of the main problems is the unavailability of efficient methods to control the hardware reconfiguration process. The main contribution of this work is the proposal and construction of a hardware reconfiguration controller totally built in hardware. This is different from previous approaches, where software implementations dominate. An important characteristic of the implemented controller is that it is part of the system hardware, making the latter capable of self reconfiguration without resource to external controlling devices. Also included is a brief review of the state of the art in hardware reconfigurability, putting emphasis in dynamically and partially reconfigurable systems and devices. The PADREH framework for the design and management of reconfigurable systems is proposed.

## **RESUMO**

Sistemas reconfiguráveis onde o hardware pode ser alterado em tempo de execução possuem o potencial para flexibilizar hardware de forma similar à flexibilidade provida pelo uso de software. Eles podem apresentar a vantagem adicional de poderem simultaneamente alcançar melhor desempenho e menor tamanho. Contudo, existem carências em dispositivos de suporte, ferramentas e fluxos de projeto para tais sistemas. Uma das principais carências são métodos eficientes de controle do processo de reconfiguração do hardware. A principal contribuição deste trabalho é a proposta e construção de um controlador de configurações de hardware implementado totalmente em hardware, em contraposição a propostas da literatura, realizadas predominantemente em software. Uma característica importante do controlador implementado é que este é parte do hardware do sistema, tornando o mesmo capaz de se autoreconfigurar, sem recurso a dispositivos de controle externos. Ainda no presente trabalho, apresenta-se um resumo do estado da arte em sistemas reconfiguráveis, com ênfase em sistemas dinâmica e parcialmente reconfiguráveis. Propõe-se o arcabouço PADREH para projeto e gerenciamento destes sistemas.

# CONTROLE DE CONFIGURAÇÕES EM SISTEMAS DINÂMICA E PARCIALMENTE RECONFIGURÁVEIS

*Ewerson L. S. Carvalho, Frederico B. Möller, Fernando G. Moraes, Ney L. V. Calazans,*

Pontifícia Universidade Católica do Rio Grande do Sul – PUCRS  
Faculdade de Informática – FACIN  
Av. Ipiranga, 6681 - Prédio 30 / Bloco 4 Telefone: +55 51 3320-3611 - Fax: +55 51 3320-3621  
90619-900 - Porto Alegre - RS - BRASIL

{ecarvalho,frebm,moraes,calazans}@inf.pucrs.br

## RESUMO

Sistemas reconfiguráveis onde o hardware pode ser alterado em tempo de execução possuem o potencial para flexibilizar hardware de forma similar à flexibilidade provida pelo uso de software. Eles podem apresentar a vantagem adicional de poderem simultaneamente alcançar melhor desempenho e menor tamanho. Contudo, existem carências em dispositivos de suporte, ferramentas e fluxos de projeto para tais sistemas. Uma das principais carências são métodos eficientes de controle do processo de reconfiguração do hardware. A principal contribuição deste trabalho é a proposta e construção de um controlador de configurações de hardware implementado totalmente em hardware, em contraposição a propostas da literatura, realizadas predominantemente em software. Uma característica importante do controlador implementado é que este é parte do hardware do sistema, tornando o mesmo capaz de se autoreconfigurar, sem recurso a dispositivos de controle externos. Ainda no presente trabalho, apresenta-se um resumo do estado da arte em sistemas reconfiguráveis, com ênfase em sistemas dinâmica e parcialmente reconfiguráveis. Propõe-se o arcabouço PADREH para projeto e gerenciamento destes sistemas.

## 1. INTRODUÇÃO

Hoje, pode-se notar o crescimento acentuado do interesse em computação configurável, como evidenciado por Zhang e Ng [1]. A flexibilidade proporcionada pela computação reconfigurável é um fator relevante que pode aumentar a janela de tempo na qual o produto permanece no mercado (tempo de vida do produto). Como no caso dos sistemas de software, que recebem atualizações constantes, o hardware implementado em dispositivos reconfiguráveis também pode usufruir desta estratégia para se manter útil por mais tempo.

O ciclo de desenvolvimento de sistemas vem diminuindo em duração por pressões de mercado. O uso da tecnologia configurável e o reuso de núcleos de propriedade intelectual (IP-cores [2]) tornam o projeto de SoCs (*System-on-Chip*) mais rápido [3], adequando-o às restrições do mercado. O tempo para o produto chegar ao mercado pode ser reduzido de maneira significativa.

Um dos fatores mais atraentes que podem tornar o uso de computação reconfigurável uma alternativa interessante é a possibilidade de implementar, em uma área reduzida, um sistema conceitualmente maior que esta, implementando *hardware virtual* [4]. O uso de técnicas de reconfiguração em tempo de execução (*Run-Time Reconfiguration* ou RTR) pode resultar em uma economia de recursos e um melhor aproveitamento da área, já que partes do sistema não utilizadas num determinado instante podem ser “removidas” do hardware físico para dar lugar a uma outra parte do sistema necessária no momento.

RTR carece de suporte como evidenciado por Zhang e Ng [1]. Neste contexto, o suporte refere-se principalmente a ferramentas habilitadoras de uso da tecnologia e infraestrutura de sistemas dinamicamente reconfiguráveis (SDRs). Na Tabela 1 apresenta-se um resumo das carências de suporte a SDRs.

Suporte	Ferramentas	- Projeto de SDRs - Verificação de SDRs
	Infra-estrutura	- Dispositivos habilitadores de SDRs - Núcleos para controlar a operação de SDRs - Interfaces padronizadas entre núcleos

**Tabela 1-** Carência de suporte a RTR

A implementação de SDRs pressupõem uma infraestrutura composta por núcleos específicos para controle e operação de SDRs, como citado. Dentre estes núcleos destaca-se aquele responsável por gerenciar o processo de configuração, ou seja, controlar qual configuração deve ser inserida no hardware físico, e qual deve ser “removida”. Esse núcleo realiza tarefas similares às

executadas pela parte de um sistema operacional, responsável por gerenciar memória virtual e carregar processos para executar no processador, seguindo os comandos passados por um escalonador de processos. O presente trabalho objetiva atacar uma das carências relacionadas à infra-estrutura de reconfiguração para SDRs.

Este documento encontra-se organizado da seguinte forma. Na Seção 2 propõe-se um arcabouço para projeto e implementação de SDRs. A seguir, na Seção 3, trata-se o estado da arte em reconfiguração dinâmica e parcial. O estado da arte em controladores de configurações é apresentado Seção 4. A partir daí, propõe-se um modelo de controlador de configurações, na Seção 5, seguida da apresentação do estudo de caso utilizado para validar este modelo, na Seção 6. Os resultados obtidos são apresentados na Seção 7. As conclusões obtidas, por sua vez, apresentam-se na Seção 8.

## 2. PADREH - ARCABOUÇO PARA SDR

O presente trabalho é parte de um ambiente para desenvolvimento e implementação de SDRs denominado *Partial and Dynamic Reconfiguration of Hardware* (PADREH), cujo objetivo é permitir aos desenvolvedores gerarem um sistema complexo, conceitualmente maior que o dispositivo alvo com garantia de que o mesmo se comportará da maneira adequada. Pretende-se com este ambiente obter vantagens com a utilização de técnicas de reconfiguração dinâmica e parcial de hardware. O sistema PADREH divide-se em três partes:

- *Módulo de captura e validação funcional de projeto*, responsável pelas descrição e validação do sistema no nível transacional (TLM, *Transaction Level Modeling*) e tradução para o nível de transferência entre registradores (RTL, *Register Transfer Level*);
- *Módulo de particionamento e escalonamento*, que gera arquivos que descrevem o comportamento do sistema na forma de uma descrição em HDL para o módulo de *síntese física e infra-estrutura de reconfiguração*;
- *Módulo de síntese física e infra-estrutura de reconfiguração*, que gera configurações totais e/ou parciais de acordo com o particionamento, adiciona o módulo controlador parametrizado conforme as características do sistema, e gera a implementação física de uma rede de interconexão entre os núcleos.

O submódulo de *controle reconfiguração*, proposto nesse trabalho, faz parte do módulo de síntese física e infra-estrutura de reconfiguração como pode ser visto na Figura 1, onde se pode perceber que os dois primeiros

módulos fazem parte da fase de projeto do SDR, enquanto que o último, além das parametrizações realizadas na fase de projeto, atua também na execução do mesmo.

O desenvolvimento do arcabouço proposto implica abordar temas tais como o particionamento de hardware em partes fixas e reconfiguráveis, o escalonamento de configurações para execução, a forma de comunicação entre configurações, a geração de arquivos de configuração parciais e totais, a configuração de dispositivos parcialmente reconfiguráveis, e o gerenciamento da operação de configurações.

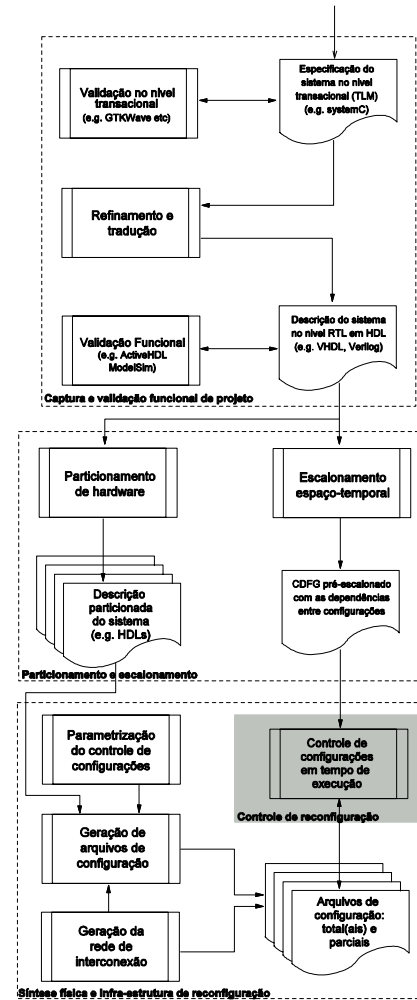


Figura 1- Fluxo de desenvolvimento e implementação de SDRs de acordo com o arcabouço PADREH proposto.

## 3. RECONFIGURAÇÃO DINÂMICA E PARCIAL

Nos últimos anos, diversos trabalhos de pesquisa em reconfiguração dinâmica e parcial têm sido desenvolvidos, tais como os descritos em [5][6]. Estes apresentam soluções para problemas em diferentes áreas valendo-se

dos benefícios da computação reconfigurável. Um conjunto distinto de trabalhos propõe métodos e ferramentas de suporte à reconfiguração parcial. Exemplos destes últimos são as propostas de controladores do processo de configuração.

Em virtude do fato de existirem bons trabalhos de revisão do estado da arte em ferramentas para reconfiguração dinâmica e parcial [1][11], sobretudo no que tange o suporte de software, no presente documento não é apresentada uma revisão mais ampla de SDRs. Revisa-se nessa Seção algumas ferramentas para geração de configurações parciais, além de métodos para interconexão de núcleos, ambos fatores importantes na realização da fase de síntese física e infra-estrutura de reconfiguração do arcabouço PADREH.

### 3.1. Tecnologias habilitadoras

Ainda hoje, são poucas as famílias de dispositivos semicondutores comerciais que habilitam reconfiguração dinâmica e parcial. A Atmel fabrica duas famílias que possibilitam reconfiguração dinâmica e parcial denominadas AT6000 e AT40k. Estas implementam a técnica denominada *Cache Logic*, que permite reconfiguração dinâmica e parcial.

A Xilinx comercializa pelo menos três famílias de dispositivos que suportam reconfiguração dinâmica e parcial. São elas: Virtex, VirtexII e VirtexII-Pro. Para configurar um dispositivo dessas famílias, deve-se preencher uma memória de configuração que determina o comportamento dos elementos componentes do dispositivo. Tal memória pode ser vista como uma matriz bidimensional de bits. Estes bits são agrupados em quadros verticais com um bit de largura, e se estendem verticalmente na forma de colunas que atravessam todo o dispositivo. Um *quadro* é a menor porção de memória de configuração que pode ser lida ou escrita. Cada coluna de recursos é dividida em um certo número de quadros.

### 3.2. Manipulação de arquivos de configuração

A geração de arquivos de configuração parciais apresenta-se como uma tarefa crucial para o desenvolvimento de SDRs. Os fabricantes fornecem junto com seus dispositivos, ambientes de CAD para desenvolvimento de sistemas que contém, muitas vezes, centenas de ferramentas. Dentre estas, algumas se dedicam à geração de arquivos de configuração parciais. O fluxo de geração de arquivos parciais com as ferramentas fornecidas pelos fabricantes deve ser realizado, ainda hoje, de forma manual. Por isso, trata-se de um processo demorado, complexo e passível de erros. Necessita-se, portanto, de ferramentas mais automatizadas para o projeto de sistemas

reconfiguráveis.

Algumas ferramentas que tem por objetivo auxiliar projetistas na tarefa de gerar arquivos de configuração parciais citadas na literatura são: PARBIT (PARTial BITfile Transformer)[12], JBits[13], além de outras desenvolvidas no Grupo de Apoio ao Projeto de Hardware (GAPH) [14].

Dyer et. al. [15] apresentam três alternativas para a geração de arquivos de configuração parciais. A primeira delas emprega fluxos realizados usando apenas ferramentas fornecidas por vendedores de dispositivos reconfiguráveis. Estas são restritas, permitem manipulações incrementais, ou seja, de pequeno porte. Como mencionado anteriormente, essas ferramentas, ainda hoje, manuais e enfrentam problemas relativos à limitação do roteamento. A segunda alternativa evidenciada por Dyer et. al. é usar a API JBits para gerar arquivos parciais. Segundo o mesmo autor, esta alternativa permite um alto grau maior de abstração, mas não fornece opções de síntese avançadas que permitam realizar otimizações relativas à potência dissipada e temporização, por exemplo. A terceira, segundo os autores, a melhor delas, reflete a idéia de combinar as duas anteriores, aproveitando as melhores características de cada. Esse fluxo misto propõe a utilização das ferramentas fornecidas pelo fabricante para realizar as tarefas de síntese e o JBits para gerar os arquivos parciais.

### 3.3. Métodos de RTR dirigidos a núcleos IP

Palma [16] apresenta uma proposta de métodos para desenvolvimento e comunicação de núcleos de hardware no contexto de reconfiguração dinâmica e parcial. O fluxo de desenvolvimento proposto sofre com o problema de falta de suporte para restringir o roteamento de arquivos de configuração parciais. A maior contribuição do trabalho de Palma é a proposta de um barramento para interconexão de núcleos de hardware dinâmica e parcialmente reconfiguráveis. O barramento proposto por Palma é composto por um árbitro que controla o acesso ao barramento.

Uma outra proposta para conexão de núcleos sugerida pela Xilinx é encontrada em [5]. Nessa referência são evidenciados dois fluxos para realização de configuração parcial: o primeiro deles representa reconfigurações onde são reconfigurados apenas alguns bits, reconfiguração parcial do tipo *incremental*. O segundo fluxo representa a reconfiguração de toda uma parte da lógica, reconfiguração parcial do tipo *modular*. Para a comunicação de núcleos utiliza-se uma estrutura denominada *Bus-Macro*, um módulo composto por quatro fios para comunicação, e outros quatro para controle de

fluxo. As *Bus-Macros* devem ser localizadas no limite entre o módulo fixo e o reconfigurável ou entre dois módulos reconfiguráveis de maneira a possibilitar a comunicação entre os mesmos.

#### 4. CONTROLADORES DE CONFIGURAÇÕES

O hardware, implementado sobre lógica reconfigurável, necessita de entidades que executem operações semelhantes as de um sistema operacional para que sua utilização ocorra de forma adequada. Da mesma forma que existe a necessidade de um sistema operacional para gerenciar a operação dos processos em uma UCP, em SDRs, deve existir uma entidade que realize tarefas semelhantes a estas, ou seja, controle a configuração de núcleos no dispositivo reconfigurável. A entidade que controla a configuração dinâmica de núcleos no sistema recebe o nome de *Controlador de Configurações*, mas pode ser chamado também *Gerente de Configurações*.

##### 4.1. Modelos propostos na literatura

Shirazi et. al. [7] apresentam um modelo genérico de controlador de configurações, composto por três partes principais: um *carregador de configurações*, um *monitor* e um *armazenamento de configurações*. O *monitor* mantém informações sobre o estado da configuração, tais como o tipo, localização no FPGA. Ainda verifica condições do usuário que ativam reconfigurações. Se uma condição é satisfeita e a configuração necessária não se encontra disponível no dispositivo, então o *monitor* notifica o *carregador* para configurá-la. O *carregador* é responsável por transportar configurações para dentro do dispositivo. Quando recebe uma requisição do *monitor*, o *carregador* obtém a localização da configuração requisitada no *armazenamento de configurações* e então inicia o processo de configuração.

Burns et.al. [8] descrevem a estrutura de um SDR, denominado RAGE (*Reconfigurable Architecture Group*), composto por 4 módulos. O *controlador de hardware virtual* é a principal interface entre a aplicação e o sistema em execução que recebe requisições para configurar uma configuração ou liberar determinada área ocupada por uma configuração. O *controlador de transformações* é responsável por relocar configurações. O *controlador de configurações* provê uma interface para o controlador de hardware virtual independente do dispositivo. O *gerenciador de dispositivo* fornece um conjunto de instruções que possibilitam manipular o dispositivo.

Segundo Lysaght [9], uma unidade de controle de configurações para SDRs deve realizar três operações principais: (i) identificar condições de reconfiguração, (ii) recuperar dados de configuração e (iii) configurar o

dispositivo.

No sistema proposto em [9], um *controlador central* comanda a ativação e a desativação de núcleos de acordo com as informações obtidas do *escalador*. Técnicas de preempção são abordadas nesse modelo e gerenciadas por uma *interface de preempção*. Um *monitor de condições de reconfiguração* é responsável por detectar e armazenar a satisfação de condições de reconfiguração para que estas sejam processadas pelo *controlador central*. O estado de cada núcleo é armazenado em *registradores de estado*. Um *decodificador de dados de configuração* inclui os circuitos definidos pelo usuário para converter os dados de configuração codificados em dados válidos, de acordo com as estratégias de compactação de dados ou encriptação adotadas. Um *transformador de layout* é usado para obter-se a máxima utilização do dispositivo configurável. Isso é obtido atrasando a decisão do posicionamento final do núcleo no dispositivo.

##### 4.2. Implementações encontradas na literatura

Curd [10] descreve a implementação de um controlador de configurações para plataformas VirtexII-Pro, específico para configurar transceptores de banda larga incorporados a estes dispositivos, denominados *RocketIO*. Este realiza apenas reconfiguração incremental, onde apenas alguns bits são alterados. O software implementado em um processador *PPC405* controla o processo de configuração, lendo dados para configuração de uma memória BRAM e enviando-os para a *ICAP*, uma interface interna de configuração do dispositivo. A escrita/leitura de dados no/do *ICAP* é controlada pelo gerenciador de *DMA*.

Blodget et. al. [17] descrevem uma proposta semelhante a apresentada em [10], já que a lógica de controle é implementada em software. O controlador executando no processador *MicroBlaze* (IP Core fornecido pela Xilinx) requisita um quadro de configuração. A partir daí, a lógica de controle realiza uma operação de readback através da *ICAP* e carrega os dados lidos em uma BRAM. Após o readback, o programa modifica estes dados e os envia à *ICAP* para reconfigurar dispositivo com a lógica modificada.

As duas implementações representam controladores implementados em software, que executando em um processador, controla o processo de reconfiguração.

##### 4.3. Considerações em controladores de configurações

Os modelos e implementações estudadas até o presente momento diferem em sua complexidade. Contudo, todos possuem blocos (ou conjuntos de blocos) que desempenham um mesmo conjunto de tarefas.

Características	Modelos			Implementações		Proposta
	Shirazi et.al.	Burns et.al	Lysaght et.al	Curd	Blodget et.al	RSCM
Hardware/software	Não aplicável	Não aplicável	Não aplicável	Software	Software	Hardware
Dispositivo alvo	XC6200	XC6200	XC6200	VirtexII-Pro	VirtexII	VirtexII
Escalonamento	Estático	Estático	Dinâmico	Nenhum	Nenhum	Estático
Preempção	Não	Não	Sim	Não	Não	Não
Relocação	Não	Sim	Sim	Não	Não	Não
Armazenamento de configurações	Sim	Sim	Sim	Sim	Não	Sim
Decodificação de configurações	Não	Não	Sim	Não	Não	Não
Localização do controlador	Não aplicável	Não aplicável	Não aplicável	Interno ao FPGA	Interno ao FPGA	Interno ao FPGA
Ano da publicação	1998	1997	1999	2003	2003	2003

**Tabela 2-** Resumo dos controladores de configurações estudados

Em todos os casos existe:

- armazenamento de configurações, com exceção da implementação de Blodget [17];
- monitor de condições de reconfiguração opcional;
- armazenamento do estado do sistema;
- carregador de configurações;
- relocador de módulos opcional.

Um quadro comparativo entre as propostas e implementações de controladores de configurações encontrados na literatura é apresentado na Tabela 2. O modelo mais sofisticado prevê o uso de preempção além da possibilidade de manipular dados compactados ou encriptados. Ele ainda possui um escalonador, que nos outros modelos não faz parte do sistema e apenas fornece entradas para o controlador de configurações. As três primeiras propostas apresentadas foram desenvolvidas visando, sobretudo uma família de FPGAs descontinuada, a XC6200 da Xilinx. Os dois últimos modelos, no entanto, foram desenvolvidos para arquiteturas atualmente disponíveis. O primeiro deles para dispositivos VirtexII-Pro da Xilinx e o segundo para dispositivos da família VirtexII, do mesmo fabricante.

Nenhuma das implementações apresentadas segue algum dos modelos propostos. No presente trabalho será apresentado um sistema de controle de configuração que possui a maioria dos submódulos propostos nos modelos apresentados anteriormente. As implementações de Curt e Blodget et. al. refletem sistemas controladores de configurações implementados parcial ou totalmente em software. A abordagem aqui adotada será implementar o sistema de controle de configurações em hardware. Estuda-se tal abordagem com o objetivo de obter informações relevantes para futuramente elaborar um comparativo entre o modelo aqui proposto, em hardware, e modelos desenvolvidos em software. Até o presente instante é impossível estimar de maneira adequada qual das abordagens é a melhor porque nenhum sistema dessa natureza foi implementado inteiramente em hardware.

#### 4.4. Gerenciamento de recursos em SDRs

##### 4.4.1. Particionamento de hardware

Particionar um sistema consiste em separá-lo *partes* menores mantendo inalterado o seu comportamento. Dessa forma, o fato do sistema estar particionado é transparente a entidades externas. O principal objetivo do particionamento é reduzir a complexidade do projeto de um sistema, onde cada parte pode ser tratada independentemente. Modernamente, no âmbito de sistemas configuráveis, outros objetivos adquirem maior importância, tal como solucionar o problema relacionado às restrições físicas do dispositivo configurável, que pode não disponibilizar recursos suficientes para implementar completamente o sistema.

Segundo [1], o particionamento de sistemas dinâmica e parcialmente reconfiguráveis pode acontecer em dois domínios. No *domínio espacial* o sistema é dividido em recursos estáticos de hardware, enquanto que no domínio temporal, o sistema é dividido dentro de segmentos de tempo exclusivos. O particionamento espacial é a abordagem clássica em projetos de hardware em geral. O particionamento temporal explora a possibilidade de dividir o sistema no tempo, onde apenas parte deste é configurado a cada instante, enquanto outras partes são configuradas quando necessárias de acordo com a necessidade da aplicação.

##### 4.4.2. Políticas de escalonamento de bitstreams parciais

O conceito escalonamento aplica-se frequentemente a assuntos relacionados a software, especificamente em sistemas operacionais. Para escopo de SDRs, Vasilko [19] define o problema de escalonamento como:

*“Dado um conjunto de configurações e a área total do dispositivo reconfigurável, encontrar o melhor escalonamento que obedeça a restrições de precedência*

(ou prioridade) e que o somatório das áreas de cada uma das configurações fisicamente presentes no dispositivo não seja maior que a área total”.

O escalonador pode aplicar técnicas de preempção (interrompção da execução de uma configuração), dependendo da política adotada. Seu emprego, no entanto, exige a presença de entidades responsáveis por gerenciar o estado atual de execução da configuração a ser preemptada. Esta entidade responsabiliza-se por armazenar o estado da configuração para que este seja “retomado” quando a configuração retornar ao dispositivo.

O escalonamento de configurações pode ser dinâmico ou estático. Um sistema pode ter seu cronograma de escalonamento definido em tempo de projeto, sendo portanto fixo. Este modelo recebe o nome de escalonamento *estático*. De outra forma, o escalonamento é dito *dinâmico*. Este último modelo indica uma nova área de pesquisas ainda inexplorada.

Em [18], Walder e Platzner apresentam quatro políticas de escalonamento: FCFS (*First Come First Serve*); SJF (*Shortest Job First*); SRPT (*Shortest Remaining Processing Time*); e EDF (*Earliest Deadline First*). As políticas sem preempção (FCFS e SJF) podem causar postergação indefinida (*starvation*).

#### 4.4.3. Relocação de bitstreams parciais

Em [20], Compton et. al. definem *relocação* como a habilidade de determinar em tempo de execução o posicionamento de uma configuração no dispositivo configurável. Sua aplicação exige métodos para modificar/adequar uma dada configuração, projetada para uma localização específica no dispositivo, de forma que a mesma possa ser inserida/configurada em uma outra localização.

Através de relocação, pode-se reduzir substancialmente o tamanho da memória necessária para armazenar as configurações, pois não será mais necessário armazenar uma instâncias do bitstream para cada área reconfigurável.

Segundo [20], uma *colisão* ocorre quando uma configuração a ser realizada foi projetada para ocupar uma localização que sobrepõe uma outra já residente no dispositivo. Quando esse evento ocorre, apenas uma das configurações pode estar residente no dispositivo num dado instante. Relocação resolve o problema de colisão. Quando duas configurações colidem, pelo menos uma delas deve ser relocada.

## 5. O CONTROLADOR DE CONFIGURAÇÕES

## RSCM

A proposta do presente trabalho é propor e desenvolver um controlador de configurações, que recebe o nome de *Reconfigurable System Configuration Manager* (RSCM). Esta proposta limita-se ao desenvolvimento de uma implementação puramente em hardware, onde o dispositivo configurável possui internamente toda, ou a maior parte da, lógica de controle de configurações. Esta limitação é imposta pelo tempo reduzido para que se possa implementar a solução também em software, soluções mistas e realizar a comparação de compromissos.

Parte da infra-estrutura de reconfiguração parcial, apresentada na Seção 2, é suposta como funcional. Esta parte corresponde tanto ao suporte para desenvolvimento de configurações parciais quanto à padronização da interface de comunicação entre diferentes módulos. Assume-se também que a arquitetura alvo compõe-se de dispositivos da família VirtexII da Xilinx, devido a sua disponibilidade no ambiente do trabalho proposto e porque habilitam sua reconfiguração dinâmica e parcial.

A estrutura do sistema proposto pode ser vista na Figura 2. O sistema RSCM proposto é composto por seis módulos. A função da *Memória de Configurações* (MC) é armazenar todas as configurações que são usadas no sistema. Normalmente é necessária muita memória para armazenamento, a qual pode ser interna ou externa ao dispositivo. Pode-se pensar em uma pequena memória interna ao dispositivo com as configurações mais usadas, uma cache de configurações, mas esta estratégia não será investigada em mais detalhes aqui.

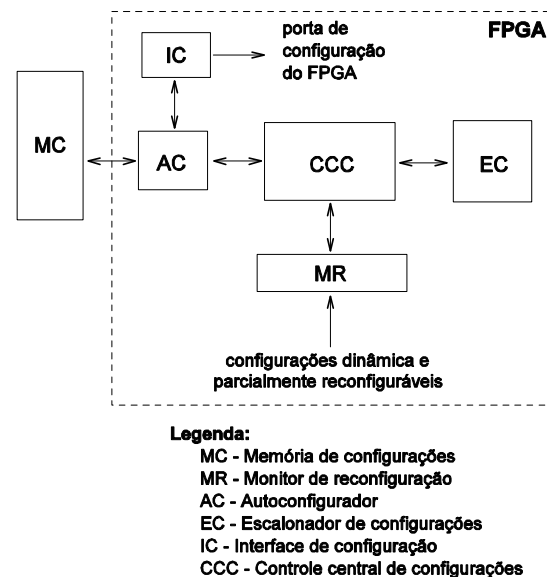


Figura 2 -Estrutura do modelo RSCM proposto e implementado

O *Autoconfigurador* (AC) é o módulo responsável por

controlar a tarefa de configuração propriamente dita. Ele possui interface com a MC, para onde envia sinais de controle requisitando dados para configurar o dispositivo. A interface com o módulo de *Interface de Configuração* permite o envio dos dados de configuração. A interface com o *Controle Central de Configurações* permite receber requisições de disparo do processo de configuração e enviar sinais de status. O módulo *Autoconfigurador* pode conter, além da lógica para configurar o dispositivo outra para gerenciar a relocação de configurações.

A *Interface de Configuração* (IC) responsabiliza-se por receber os dados de configuração provenientes de AC e enviá-los para a porta de configuração do dispositivo. Através da porta de configuração do dispositivo o FPGA recebe os dados de configuração configura-se de acordo com as informações contidas nesses dados. A máquina de configuração do dispositivo é responsável por interpretar os dados de configuração e personalizar o dispositivo. Pode-se pensar em um arquivo de configuração como sendo um programa que informa quais as tarefas a serem realizadas pela máquina de configuração para que o dispositivo se comporte de determinada maneira.

O *Controle Central de Configurações* (CCC) é responsável por gerenciar todo o fluxo de controle entre os módulos do sistema RSCM internos ao dispositivo. Ele ainda aplica o cronograma que recebe do módulo *Escalaonamento de Configurações*. O CCC recebe requisições do módulo MR e requisita os serviços do *Escalaonamento de Configurações* e do AC.

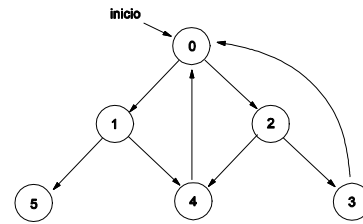
O *Escalaonador de Configurações* (EC) é módulo responsável por interpretar o cronograma de execução das configurações. Tal módulo recebe do CCC requisições de serviço. O escalonador possui uma estrutura de dados que contém informações sobre as dependências entre as configurações, denominada *Tabela de Dependência e Descritores* (TDD), como aquela apresentada na Tabela 3, onde *Config* indica o identificador da configuração, *Npredec* indica o número de predecessores da configuração, e *Suc N* é o sucessor *N* da configuração. A Tabela 3 representa o grafo de dependência apresentado na Figura 3.

Config.	Npredec	Suc 1	Suc 2
0	2	1	2
1	1	5	4
2	1	4	3
3	1	0	-
4	1	0	-
5	1	-	-

**Tabela 3** - Estrutura da TDD do EC do sistema RSCM.

Além de simular o controlador RSCM, também prototipou-se o sistema em FPGA. Durante essa fase,

pode-se perceber que o sistema portou-se adequadamente e apresentou resultados idênticos aos obtidos através da simulação. Em virtude disso, as formas de ondas relativas à prototipação não serão aqui apresentadas.



**Figura 3** - Grafo utilizado para criar a TDD da Tabela 3.

Todo o sistema RSCM foi descrito em linguagem de descrição de hardware VHDL. Todos os fontes são disponibilizados em <http://www.inf.pucrs.br/~gaph>.

## 6. ESTUDO DE CASO: R8R

Apresenta aqui o estudo de caso elaborado que ao mesmo tempo valida o sistema RSCM e enfatizando a importância do emprego de técnicas RTR em sistemas programáveis. O sistema que compõe esse estudo denomina-se R8R (i.e. R8 Reconfigurável), sendo um processador dinâmica e parcialmente reconfigurável. Ele é composto pelo processador R8 e pelo controlador de configurações RSCM.

### 6.1. O processador R8

O processador R8 [21] faz parte de uma família de processadores, concebida com a finalidade de dar suporte ao ensino de conceitos de arquitetura e organização de computadores em nível de graduação e pós-graduação. Este processador é uma máquina Von Neuman, com memória de dados e instruções conjunta. Ele possui uma arquitetura load-store, onde as operações lógico/aritméticas são executadas entre registradores, e as operações de acesso à memória só executam ou uma leitura ou uma escrita de uma posição de memória. Devido à característica load/store, o processador deve ter um conjunto de registradores de trabalho, para reduzir o número de acessos à memória. O processador R8 é praticamente uma máquina RISC, faltando, contudo algumas características que existem em qualquer máquina RISC, tal como pipelines.

### 6.2. O projeto R8R

O projeto R8R emprega conceitos de processadores que contém um conjunto de instruções dinâmico, denominados



*Dynamic Instruction Set Computers* ou DISCs, tratados por Wirthlin e Hutchings, em [23]. Tal projeto propõe o desenvolvimento de um processador dinâmica e parcialmente reconfigurável, baseado na estrutura do processador R8 e em uma versão simplificada do controlador de configurações RSCM proposto. O principal objetivo desse sistema é possibilitar a implementação de um processador com um conjunto de instruções variável e expansível. Além das instruções padronizadas do processador R8, o R8R agrega outras instruções destinadas a controlar a execução de coprocessadores reconfiguráveis. Esses coprocessadores são configurados no sistema de acordo com a demanda definida pelo software do R8R, e podem executar as mais diferentes tarefas, tais como operações aritméticas complexas e entrada/saída de dados em altas taxas.

Na Figura 4 apresenta-se um diagrama de blocos do processador reconfigurável R8R. O sistema compõe-se do processador R8R, da memória de programa e dados, do controlador de configurações RSCM e dos coprocessadores dinâmica e parcialmente reconfiguráveis.

Durante a execução do sistema, o processador R8R seleciona determinado coprocessador para operar. Esta seleção é enviada ao controlador RSCM. Este, de posse de informações da alocação das áreas reconfiguráveis, verifica a necessidade de configurar tal processador ou não. Após tomar as devidas providências, RSCM notifica R8R que o processador está disponível. A R8R, agora pode solicitar serviços do coprocessador.

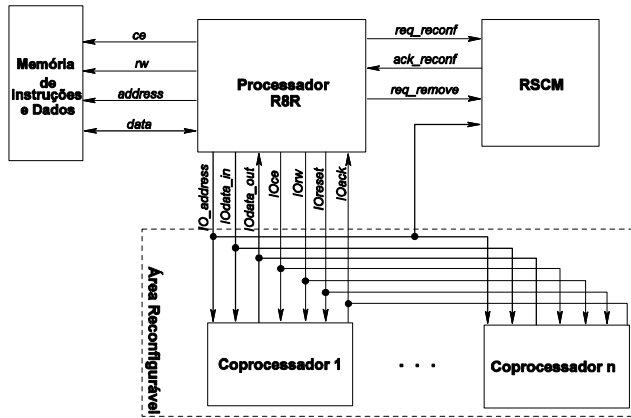


Figura 4 - Estrutura do processador reconfigurável R8R [22].

Além de desenvolver os módulos apresentados na Figura 4 foram desenvolvidos cinco coprocessadores, listados abaixo.

- **sqrt**: calcula a raiz quadrada de um valor de 32bits e apresenta uma resposta de 16bits;
- **multi**: executa a multiplicação de dois valores de 16

bits, gerando uma resposta em 32 bits;

- **div**: executa a divisão entre dois valores de 16 bits, gerando uma resposta em 32 bits.

## 7. RESULTADOS OBTIDOS

Arquivos de configuração com 460 palavras de dados foram reconfigurados no sistema com tempos na ordem de  $\mu s$ . A partir daí, foram obtidos os tempos de inicialização da reconfiguração e o tempo para enviar cada palavra à interface ICAP. O tempo de inicialização corresponde ao intervalo utilizado para buscar a primeira palavra da memória, já que as demais leituras são escondidas por sua sobreposição à tarefa de reconfiguração, tal qual um pipeline. O tempo de inicialização obtido foi  $884ns$ . O tempo para enviar cada palavra obtida foi de  $748ns$ . Substituindo estes valores na Equação 1 do tempo de reconfiguração, obtém-se os valores correspondentes, específicos para o sistema RSCM em questão, conforme descrito pela Equação 2.

$$T_{reconfiguração} = T_{inicialização} + (N_{palavras} \times T_{palavra}) \quad (1)$$

$$T_{reconfiguração} = 884ns + (N_{palavras} \times 748ns) \quad (2)$$

Este mesmo procedimento foi realizado com informações obtidas através de uma reconfiguração utilizando o software Impact, no modo Paralelo. Observou-se que o tempo gasto para configurar cada uma das palavras do bitstream é  $8,44\mu s$ . O tempo de inicialização é de aproximadamente  $160ms$ . Dessa forma, pôde-se definir uma equação específica para os sistemas configurados através do software Impact, a Equação 3.

$$T_{reconfiguração} = 160ms + (N_{palavras} \times 8,44\mu s) \quad (3)$$

A partir daí, foi criado o gráfico apresentado na Figura 5. Este compara os tempos de reconfiguração, utilizando os dois métodos citados, de acordo com o tamanho do arquivo de configuração. Um arquivo de configuração total para dispositivos da família VirtexII da Xilinx possui 127.581 palavras de 32 bits.

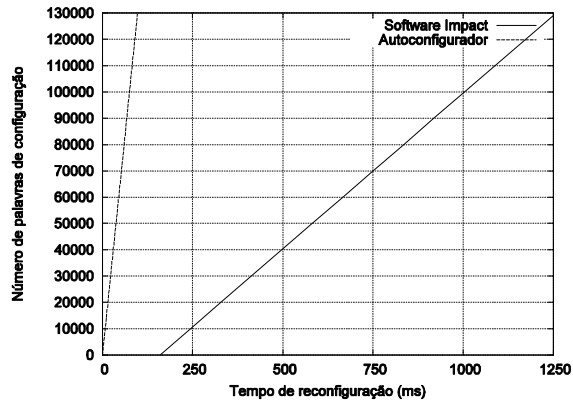


Figura 5 - Relação tamanho do arquivo de configuração versus tempo de reconfiguração

No gráfico da Figura 5 nota-se que o tempo de reconfiguração utilizando o Autoconfigurador do sistema RSCM é sempre significativamente menor que aquele obtido utilizando o software Impact.

Para cada um dos coprocessadores desenvolvidos em hardware foi desenvolvida uma versão em software. Executando programas que utilizam esses recursos por determinado tempo, pôde-se obter um comparativo entre o desempenho global do sistema quando se utiliza apenas rotinas desenvolvidas em software e, quando utiliza-se coprocessadores implementados em hardware.

Na Figura 6 apresentam-se os resultados obtidos a partir da execução das rotinas implementadas em software e da utilização dos coprocessadores. De acordo com medições, o tempo de reconfiguração de cada coprocessador é de aproximadamente 10ms.

Observou-se que programas que realizam mais que 750 operações de multiplicação possuem um melhor desempenho quando o coprocessador *multi* é usado, em vez das rotinas implementadas em software. Programas que realizam mais de 260 operações de divisão possuem um melhor desempenho quando utilizam o processador *div* em vez da rotina implementada em software. Apenas 200 operações de extração da raiz quadrada já validam o uso do coprocessador *sqrt* em contraproposta à implementação em software.

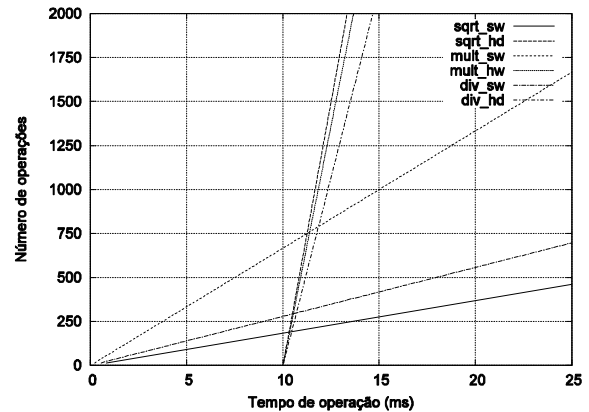


Figura 6 – Tempos de execução dos coprocessadores comparados às suas versões em software

## 8. CONCLUSÕES

Neste trabalho, pôde-se compreender o cenário atual do desenvolvimento de sistema dinâmica e parcialmente reconfiguráveis. Ficou evidenciada a falta de ferramentas para o projeto e suporte à implementação de sistemas desta natureza. Embora diversos estudos sejam realizados enfatizando RTR, os fabricantes ainda não disponibilizam dispositivos que habilitem a sua reconfiguração dinâmica e parcial de uma forma mais facilmente integrável ao fluxo de projeto tradicional de sistemas computacionais.

A necessidade de um subsistema para suporte a operação de SDRs é evidente. RSCM é uma proposta que visa suprir a necessidade de um controlador de configurações implementado em hardware, ocasionando baixa sobrecarga de área do dispositivo reconfigurável. O RSCM é utilizado para controlar a reconfiguração de núcleos IP arbitrariamente complexos.

## 9. REFERÊNCIAS

- [1] X. Zhang and K. W Ng. A review of high-level synthesis for dynamically reconfigurable FPGAs. *Microprocessors and Microsystems*, v.24, p.199-211. 2000.
- [2] R. A. Bergamaschi and W. R. Lee. Designing system-on-chip using cores. In *37th Design Automation Conference (DAC'00)*, p.420-425. USA. 2000.
- [3] G. Martin and H. Chang. Tutorial – System on Chip Design. In *9th International Symposium on Integrated Circuits, Devices & Systems*. Singapore. 2001.
- [4] A. DeHon, Comparing Computing Machines. In *Configurable Computing: Technology and Applications*. v.3526, p.124-133, 1998.
- [5] D. Lim and M. Peattie. Two Flows for Partial Reconfiguration: Module Based or Small Bit Manipulations.

Xilinx, 2002.

- [6] E. Sanchez, M. Sipper, J-O. Haenni, J-L. Beuchat, A. Stauffer and A. Perez-Urbe. Static and Dnamic Configurable Systems. *IEEE Transactions on Computers*. v.48, n.6, p.556-564. 1999.
- [7] N. Shirazi, W. Luk and P.Y.K. Cheung. Run-Time Management of Dynamically Reconfigurable Designs. In *8th Field-Programmable Logic and Applications (FPL'98)*, v.1482. Estonia. 1998.
- [8] J. Burns, A. Donlin, J. Hogg, S. Singh and M.Wit. A Dynamic Reconfiguration Run-Time System. In *5th Field-Programmable Custom Computing Machines (FCCM'97)*, p.66-75. USA. 1997.
- [9] D. Robinson and P. Lysaght. Modelling and Synthesis of Configuration Controllers for Dynamically Reconfigurable Logic Systems using the DCS CAD Framework. In *9th Field-Programmable Logic and Applications (FPL'99)*, v. 1673. Glasgow, UK. 1999.
- [10] D.R. Curd. Partial Reconfiguration of RocketIO Pre-emphasis and Differential Swing Control Attributes. Xilinx, 2003.
- [11] D. Mesquita. Contribuições para Reconfiguração Parcial, Remota e Dinâmica de FPGAs. 2002. Dissertação de Mestrado - Pontificia Universidade Católica do Rio Grande do Sul - PPGCC - FACIN - Porto Alegre, RS, Brasil.
- [12] E.L. Horta, J.W. Lockwood and S.T. Kofuji. Using PARBIT to Implement Partial Run-Time Reconfigurable Systems. In *12th Field Programmable Logic and Application (FPL'02)*. v.2438. France. 2002.
- [13] S.A. Guccione, D. Levi and P. Sundararajan. JBits: Java based interface for reconfigurable computing. In *2nd Military and Aerospace Applications of Programmable Devices and Technologies Conference (MAPLD')*. 1999.
- [14] L.H. Möller. Ferramentas de Reconfiguração Parcial, Remota e Dinâmica de FPGAs Virtex. 2003. Relatório Técnico (RT035) - Pontificia Universidade Católica do Rio Grande do Sul - PPGCC - FACIN. Porto Alegre, RS, Brasil.
- [15] M. Dyer, C. Plessl and M. Platzner. Partially Reconfigurable Cores for Xilinx Virtex. In *12th Field Programmable Logic and Application (FPL'02)*, v.2438. France. 2002
- [16] J.C. Palma, A.V. Mello, L.H. Möller, F.G. Moraes and N.L.V. Calazans. Core Communication Interface for FPGAs. In *15th Symposium on Integrated Circuits and Systems Design (SBCCI'02)*. Brazil. 2002.
- [17] B. Blodget, S. McMillan and P. Lysaght. A Lightweight Approach for Embedded Reconfiguration of FPGAs. In *6th Design, Automation and Test in Europe Conference and Exhibition (DATE'03)*, p.399-400. Germany. 2003.
- [18] C. Steiger, H. Walder and M. Platzner. Heuristics for Online Scheduling Real-time Tasks to Partially Reconfigurable Devices. In *13rd International Conference on Field Programmable Logic and Application (FPL'03)*. Portugal. 2003.
- [19] M. Vasilko and D. Ait-Boudaoud. Scheduling for Dynamically Reconfigurable FPGAs. In *7th International Workshop on Logic and Architecture Synthesis (IFIP'95)*, p.328-336. France. 1995.
- [20] K. Compton, J. Cooley, S. Knol and S. Hauck. Configuration Relocation and Defragmentation for Reconfigurable Computing. In *8th Field-Programmable Custom Computing Machines (FCCM'00)*, p.279-280. USA. 2000.
- [21] F.G. Moraes and N.L.V. Calazans. R8 Processor Architecture and Organization Soecification and Desine Guidelines. 2003. disp. em [http://www.inf.pucrs.br/~gaph/Projects/R8/public/R8\\_arq\\_spec\\_eng.pdf](http://www.inf.pucrs.br/~gaph/Projects/R8/public/R8_arq_spec_eng.pdf)
- [22] N.L.V. Calazans and F.G. Moraes. Integrating the Teaching of Computer Organization and Architecture with Digital Hardware Design Early in Undergraduate Courses. *IEEE Transactions on Education*, v.44, n.2, p.109-119. 2001.
- [23] M. J. Wirthlin and B. L. Hutchings. A Dynamic Instruction Set Computer. In *th Field-Programmable Custom Computing Machines (FCCM'95)*. CA. 1995.