

Tendências em Reconfiguração dinâmica de FPGAs

Fernando Moraes¹, Daniel Mesquita²

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

Resumo: Esta palestra compreende quatro partes: (1) a primeira parte introduz conceitos relacionados à reconfiguração parcial, assim como justifica seu uso; (2) a segunda parte apresenta o estado-da-arte em sistemas reconfiguráveis, classificando-os temporariamente; (3) a terceira parte da palestra analisará as duas únicas famílias de dispositivos configuráveis (disponíveis comercialmente) que habilitam reconfigurabilidade dinâmica e parcial; (4) a última parte da palestra mostrará as tendências observadas na literatura quanto à reconfiguração parcial e dinâmica.

1. Introdução

Muitas aplicações emergentes em telecomunicações, multimídia e sistemas embarcados necessitam que suas funcionalidades permaneçam flexíveis mesmo depois de o sistema ter sido manufaturado [12]. 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 [22]. Essas modificações referem-se a mudanças em circuitos digitais via reconfiguração sem a interrupção da operação do circuito (reconfiguração dinâmica). 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,

¹ **Fernando Gehm Moraes** é professor adjunto pela Faculdade de Informática, PUCRS, desde 1996. Engenheiro eletrônico pela UFRGS em 1987, Mestre em Ciência da Computação pelo CPGCC-UFRGS em 1990 e Doutor em Microeletrônica pela Universidade de Montpellier II (França) em 1994. Autor de dezenas de artigos em conferências relacionadas a microeletrônica. Seus interesses de pesquisa incluem prototipação rápida de sistemas digitais, projeto conjunto/hardware software, desenvolvimento de ferramenta de CAD para projeto de sistemas digitais e síntese física de circuitos integrados. Currículo disponível em: <http://www.inf.pucrs.br/~moraes/cvlatte.htm>.

² **Daniel Gomes Mesquita** é mestrando em Ciência da Computação pela PUCRS, ênfase em Sistemas Digitais e Arquitetura de Computadores. Bacharel em Ciência da Computação pela Universidade de Cruz Alta, RS, em 1999. Realiza pesquisas a respeito de sistemas digitais reconfiguráveis desde 2000, já tendo publicado artigo em evento internacional a esse respeito. Como membro do Grupo de Apoio ao Projeto de Hardware da PUCRS, participa de projeto relacionado com telecomunicações, numa parceria com a indústria. Currículo disponível em : <http://www.inf.pucrs.br/~dmesquita/pessoal/itens/curricu2.htm>

comunicações sem fio reconfiguráveis, sistemas de computação de alto desempenho, processamento de imagens em tempo real e produtos para consumo atualizáveis remotamente, entre outros.

Além das características citadas acima, a reconfiguração dinâmica 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 [20]. 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 da reconfiguração dinâmica 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.

1.1 Definições

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

- **FPGA:** Dispositivo que consiste em uma matriz de blocos lógicos programáveis cercada de blocos de entrada e saída, e conectada por fios programáveis de interconexão também programável.
- **Granularidade:** Característica do dispositivo ou sistema relacionada com o grão; sendo que entende-se por grão a menor unidade programável da qual é composta um dispositivo (FPGA) ou um SDR.
- **Grão-grosso:** Os FPGAs de grão grosso possuem como grão unidades lógicas e aritméticas (ULAs), pequenos microprocessadores e memórias. Como exemplos desse tipo de arquitetura podem ser citadas as máquinas RAW [21] e a arquitetura GARP [6].
- **Grão-médio:** Os FPGAs que tem grão médio consistem em blocos lógicos bastante grandes, freqüentemente contendo duas ou mais tabelas de busca (*look-up tables* ou LUTs) e dois ou mais *flip-flops*. A maioria das arquiteturas de FPGAs implementa a lógica em LUTs de quatro entradas. Como exemplos de dispositivos que atualmente possuem grão médio podem ser citados as famílias Spartan e Virtex, da Xilinx; Flex e Apex, da Altera; e AT40K, da Atmel.
- **Grão-fino:** Nos dispositivos com grão fino há um grande número de blocos lógicos simples. Os blocos lógicos normalmente contêm uma função lógica de duas entradas ou um multiplexador 4 para 1 e um *flip-flop*. As famílias SPGA (Actel) e AT6000 (Atmel) são exemplos atuais de dispositivos de grão fino.
- **Reconfiguração total:** É a forma de configuração onde o dispositivo reconfigurável é inteiramente alterado.
- **Reconfiguração parcial:** É a forma de configuração que permite que somente uma porção do sistema reconfigurável seja reconfigurada. Uma reconfiguração parcial pode ser não-disruptiva - onde as porções do sistema que não estão sendo reconfiguradas permanecem completamente funcionais durante o ciclo de reconfiguração; ou disruptiva - onde a reconfiguração parcial afeta outras partes do sistema, tipicamente necessitando de uma parada no sistema inteiro. Reconfiguração parcial não-disruptiva é freqüentemente abreviada para reconfiguração parcial.
- **Reconfiguração dinâmica:** Também chamada de *run-time reconfiguration* (RTR), *on-the-fly reconfiguration* ou *in-circuit reconfiguration*. Todas essas expressões podem ser

traduzidas também como reconfiguração em tempo de execução. Reconfiguração dinâmica é outra forma de expressar a reconfiguração parcial não-disruptiva. Não há necessidade de reiniciar o circuito ou remover elementos reconfiguráveis para programação.

- DPGA: *Dinamically Programmable Gate Array*. É uma terminologia proposta em [7], que denota um hardware que pode ser programado em execução, durante a operação do sistema, em função de um conjunto de arquivos de configuração pré-carregados. Pode suportar reconfiguração parcial ou total.
- *Cache-Logic*: É um termo usado para indicar hardware reconfigurável dinamicamente através de chaveamento de contexto, como o DPGA. É uma marca registrada da empresa ATMEL [4].
- Core: Representação eletrônica de um componente em nível de processamento, como um microprocessador, ou um periférico [19]. Também pode ser tratado como módulos pré-projetados e pré-validados de hardware.

2. Estado-da-arte

As arquiteturas reconfiguráveis podem ser analisadas temporalmente, em função dos problemas a que se dispuseram resolver. A partir do amadurecimento da tecnologia habilitadora para esses sistemas (FPGAs), alguns centros de pesquisa criaram as primeiras arquiteturas reconfiguráveis, com o intuito principal de aumentar o desempenho de algoritmos que até então eram executados em software. Dentro desta primeira geração estão projetos como DECPeRLe [5], PRISM [3] e Splash [10].

Alguns sistemas mais modernos ainda utilizam essa abordagem, como o Transmogripher-2 [14], o RPM-2 [9] e o SPYDER [17]. Tais sistemas podem ser vistos como a primeira geração dos sistemas digitais reconfiguráveis, conforme a Figura 1.

Já neste primeiro momento verificou-se a eficiência da utilização de FPGAs no domínio de aplicação em questão, tanto em termos de desempenho com relação a abordagens em software quanto no que tange ao critério econômico, quando comparada a soluções ASIC. Contudo também alguns problemas foram levantados. Em geral esses sistemas possuíam um gargalo de comunicação entre microprocessadores e FPGAs e apresentavam um tempo de reconfiguração muito alto, além de poderem ser reconfigurados apenas totalmente. Essa última desvantagem significa que o sistema precisava necessariamente ser parado para que pudesse ser reconfigurado.

Em função disso novas propostas de arquiteturas surgiram. O problema de comunicação entre microprocessadores e FPGAs contou com o avanço da tecnologia habilitadora para ser resolvido, formando uma segunda geração de SDR (Figura 1). Com o aumento do número de transistores por circuito integrado (CI) tornou-se possível o desenvolvimento de um sistema composto por microprocessador, FPGA(s) e memória em um único CI (*System-on-Chip*, ou simplesmente SoC). Foram desenvolvidos SoCs de granularidade baixa (FIPSoC [18], TRUMPET [16]) e com granularidade média (Garp [6], RAW [21]).

Outro avanço tecnológico ocorrido foi a possibilidade de reconfiguração dinâmica. Isto permitiu que as arquiteturas em questão pudessem ser reconfiguradas sem que precisassem parar totalmente de desempenhar suas funções. Essa reconfiguração dinâmica pode ser realizada por chaveamento de contexto, como o que ocorre com os sistemas derivados do DPGA. Este também é o caso do DISC, FireFly, FIPSoC e Garp. Tais sistemas encontram-se na segunda geração de SDRs, conforme a Figura 1. Splash2 apresenta uma

reconfiguração alternativa, na sua rede de interconexão externa [13] .

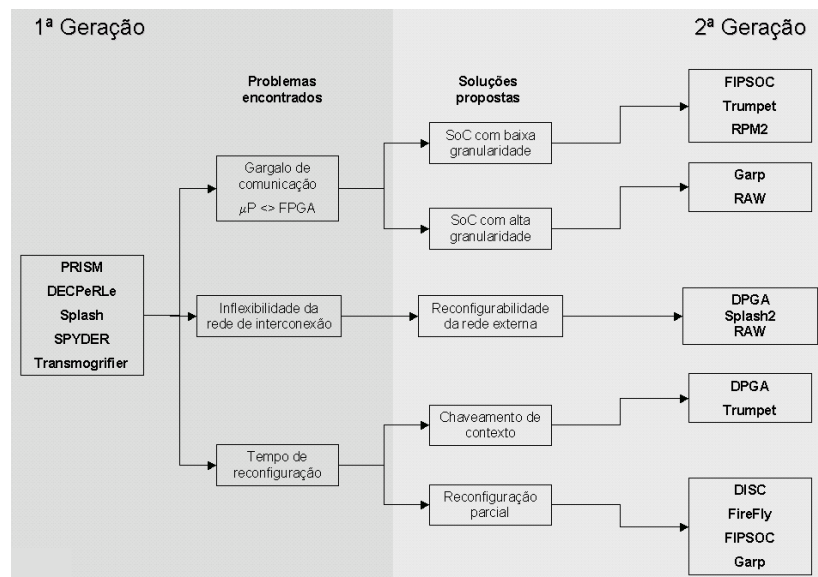


Figura 1 - Evolução das arquiteturas reconfiguráveis.

Além das abordagens derivadas de avanços tecnológicos e resolução de antigos problemas, arquiteturas reconfiguráveis tem sido utilizadas para implementação de algoritmos genéticos. FireFly é um exemplo de "hardware evolutivo" criado a partir desse enfoque. Demonstra uma utilização exótica de reconfiguração parcial e dinâmica.

3. Hardware que permite reconfiguração dinâmica

Os primeiros dispositivos que suportaram reconfiguração parcial foram criados pelas empresas National, Algotronix e Xilinx. O resultado dessa criação foram famílias de FPGAs Clay [15], Cal1024 [1] e XC6200 [23], respectivamente. Tais FPGAs não lograram grande sucesso comercial principalmente pelo fato de não terem sido produzidas ferramentas eficientes de projeto, de roteamento e de posicionamento.

Outro fabricante de FPGAs, a Altera, alega que a partir da família APEX permitiu reconfiguração parcial [2]. Contudo isto ocorre de forma muito limitada. A reconfiguração parcial dessa família dá-se através do projeto de lógica em RAM, criando uma LUT onde podem ser implementadas funções com 7 entradas e 16 saídas. Depois dessa lógica ser implementada no bloco de RAM o sistema pode reescrevê-la em qualquer tempo, mudando a configuração de parte do sistema. A grande limitação desta abordagem é que em algum lugar do circuito deve-se armazenar todas as configurações possíveis que irão modificar a RAM, isto porque não há como fazer a carga externa de novas configurações.

Duas empresas –Atmel e Xilinx – comercializam famílias FPGAs que permitem reconfiguração parcial. Nas próximas Subseções são feitas breves descrições da família AT40k da Atmel e das características da família Virtex, da Xilinx.

3.1 Atmel

Os FPGAs da família AT40K foram especialmente projetados para suportarem *Cache Logic*, que é uma técnica para construir sistemas e lógica adaptáveis [4]. Num sistema de *Cache Logic* somente as porções da aplicação que estão ativas em um dado momento realmente estão implementadas no FPGA, enquanto funções inativas são armazenadas externamente numa memória de configuração barata. Se novas funções se fazem necessárias, as antigas são sobrescritas, como mostrado no diagrama contido na Figura 2. Este procedimento aproveita-se

da latência funcional inerente a muitas aplicações – em qualquer tempo dado, somente uma pequena proporção da lógica está de fato ativa.

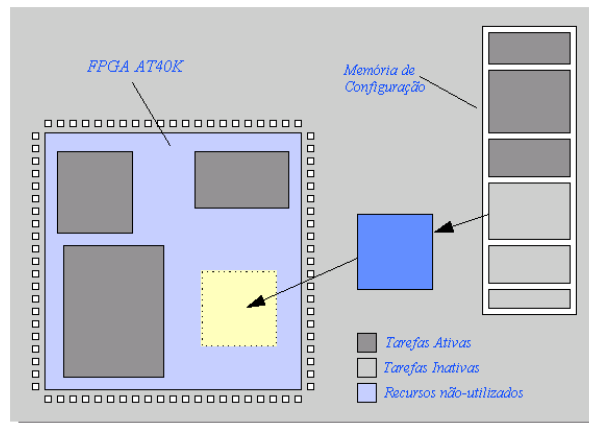


Figura 2 - Diagrama da Cache Logic, onde cores armazenados em memória configuram o FPGA em tempos diferentes.

A série AT40K implementa Cache Logic, à medida que pode ser parcialmente reconfigurável, sem interromper a operação da lógica remanescente no dispositivo. Isto permite que funções sejam substituídas em tempo de execução no FPGA, enquanto o sistema continua a operar [4].

3.2 Xilinx

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 [22].

A Figura 3 exibe uma abstração da arquitetura interna de um FPGA Virtex, onde podem ser vistos os elementos citados acima.

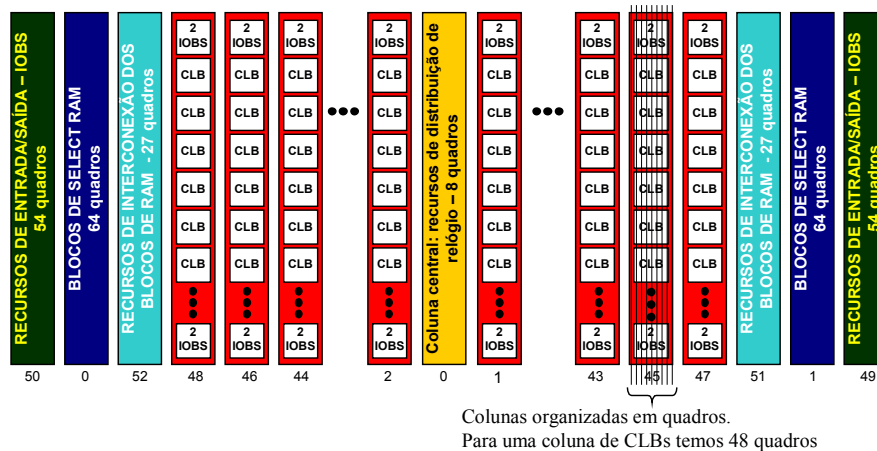


Figura 3 - Disposição em colunas dos elementos do FPGA Virtex XCV300.

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 (Figura 3), podendo estas serem lidas ou escritas individualmente. Logo, é possível reconfigurar parcialmente esses dispositivos através da modificação destas colunas no arquivo de configuração.

4. Tendências em reconfiguração dinâmica

O campo da computação reconfigurável avançou amplamente na década passada, utilizando FPGAs como a base para sistemas reprogramáveis de alto desempenho [11]. Muitos desses sistemas alcançaram altos níveis de performance e demonstraram sua aplicabilidade à resolução de uma larga variedade de problemas. Contudo, apesar dos autores desses sistemas o classificarem como reconfiguráveis, eles são tipicamente configurados uma vez antes de iniciarem a execução da aplicação.

Sistemas RTR são distintos de sistemas digitais estaticamente configuráveis por permitirem especialização da lógica e/ou do roteamento em tempo de execução. A Figura 4 ilustra o que seja RTR através de um gráfico com três eixos. Os eixos x e y indicam o plano espacial, no qual ocorre o roteamento e o posicionamento da lógica a ser implementada em um FPGA. Sistemas digitais baseados em FPGAs, mas que não são dinamicamente reconfiguráveis podem ser vistos apenas neste plano. Contudo, para SDRs, há a necessidade de uma análise temporal das reconfigurações. Em vista disso foi introduzido o eixo z, que indica as diferentes implementações de módulos funcionais no dispositivo, em relação ao tempo que ocorreram.

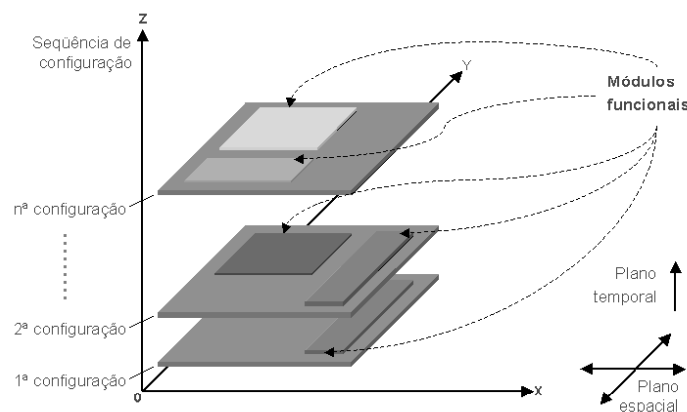


Figura 4 - Gráfico indicativo de reconfiguração parcial não-disruptiva.

As áreas de um programa que podem ser aceleradas através do uso de hardware reconfigurável frequentemente são muito numerosas ou complexas para serem carregadas simultaneamente no FPGA disponível. Para esses casos, é interessante que seja possível trocar entre diferentes configurações no mesmo dispositivo.

RTR é semelhante ao conceito de memória virtual. Conforme essa idéia, o hardware físico pode ser muito menor que do que o somatório dos recursos requeridos para cada uma das configurações. Então, ao invés de reduzir o número de configurações que são mapeadas, apenas ocorre uma troca entre o hardware necessário e o hardware implementado fisicamente [8].

Apesar da aparente vasta aplicabilidade para sistemas RTR, existem poucos sistemas

que de fato implementam esta característica. Isto se dá por dois fatores principais:

1. Falta de software para projeto, depuração e teste;
2. Falta de hardware (comercialmente disponível) especialmente projetado para permitir RTR .

Apesar de os FPGAs atuais possuírem características de hardware interessantes, como dispositivos com mais de um milhão de portas lógicas equivalentes, altas taxas de relógio, reconfiguração rápida e grande largura de banda, seu uso para reconfiguração dinâmica ainda é limitado. Essa limitada aplicabilidade deve-se a exígua oferta de ferramentas para automação de projeto e implementação, e também, pela ausência de um modelo computacional que abstraia os recursos fixos dos dispositivos da descrição do hardware a ser projetado [7].

A Figura 5 ilustra três situações relativas a uma determinada aplicação que se pretende seja implementada em um FPGA. No primeiro caso, o FPGA oferece mais recursos do que seria necessário para esta aplicação. Portanto é possível realizar a configuração do dispositivo. No caso seguinte, o download do arquivo de configuração ainda é possível, mas o FPGA ficaria sem nenhum recurso adicional disponível. Em contrapartida, no último caso, para um fluxo de projeto e implementação convencionais, seria impossível implementar a aplicação, já que ela demandaria mais recursos do que os existentes no dispositivo de destino.

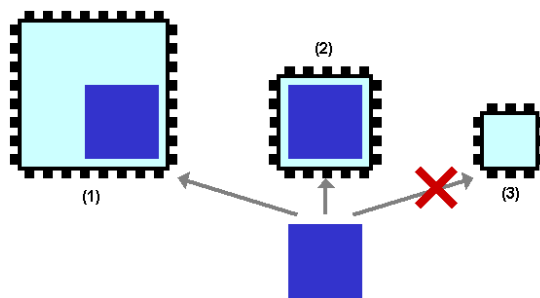


Figura 5 - Lógica projetada é dependente do dispositivo.

Portanto, a limitação de recursos de hardware tende a prejudicar a aplicabilidade desses sistemas. Em tais modelos de projeto de hardware, a escolha de um algoritmo para aplicação está restrita ao tamanho do hardware disponível. Além disso, uma estrutura de computação tem que ser fixa em tempo de execução, sem a possibilidade de alocação dinâmica de recursos.

Além disso, programas desenvolvidos para um dispositivo em particular (ou conjunto de dispositivos) possuem compatibilidade de código-fonte bastante limitada, e nenhuma compatibilidade do arquivo de configuração entre dispositivos de diferentes fabricantes. Organizar um programa para uma nova geração de dispositivos, ou para uma menor e mais barata, ou com consumo mais baixo, tipicamente requer esforço humano substancial.

Como solução a estes problemas, é apresentada a computação organizada em fluxos para execução reconfigurável (*Stream Computations for Reconfigurable Execution* - SCORE) [8]. O modelo computacional SCORE tenta resolver a limitação de recursos físicos através virtualização de recursos computacionais, de comunicação e de memória do hardware reconfigurável. Configurações do FPGA são particionadas em páginas de tamanhos fixos que se comunicam: em analogia à memória virtual, o hardware é carregado (*paged in*) sob demanda.

A comunicação por fluxo entre as páginas que não estão simultaneamente no dispositivo pode ser transparentemente buferizada através da memória. O fluxo é uma ligação

unidirecional página-à-página (Figura 6). Este esquema permite uma aplicação particionada explorar mais as páginas disponíveis fisicamente, sem necessidade de recompilação.

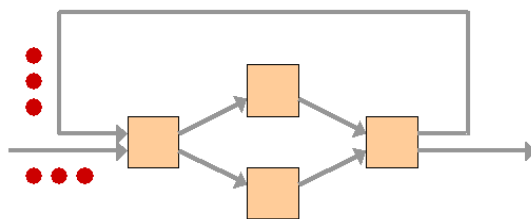


Figura 6 - Ligação unidirecional página-à-página.

Em função dessa abordagem, SCORE permite que as reconfigurações sejam menos frequentes, o que acarreta a amortização do custo de reconfiguração sobre um grande conjunto de dados.

Com um projeto adequado de hardware, este esquema permite compatibilidade binária e escalabilidade entre diferentes famílias de dispositivos através da compatibilidade das páginas.

Para os programas beneficiarem-se dos recursos físicos adicionais (páginas), o modelo de programação é uma abstração natural da comunicação que ocorre espacialmente entre blocos de hardware. Isto é, o grafo do fluxo de dados captura os blocos de computação (operadores) e a comunicação (fluxo) entre eles. Uma vez capturado, é possível explorar técnicas para mapeamento desses grafos em hardware de tamanho arbitrário. Além do mais, a composição em tempo de execução dos grafos é suportada, possibilitando uma estrutura de aplicação orientada a dados, alocação dinâmica de recursos e integração entre módulos de hardware (cores) desenvolvidos ou compilados separadamente.

5. Bibliografia

- [1] ALGOTRONIX, Equipe de documentação. **Call1024 datasete**. Disponível por WWW em http://dec.bournemouth.ac.uk/dhrhw_lib/archive/cal1024.ps.gz (1989).
- [2] ALTERA, Equipe de documentação. **Configuration (In-Circuit Reconfiguration)**. Disponível por WWW em <http://www.altera.com/support/devices/programming/sup-configuration.html> (2001).
- [3] ATHANAS, Peter M.; SILVERMAN, Harvey F. **Processor reconfiguration through instruction-set metamorphosis**. IEEE Computer, p.11_18, 1993.
- [4] ATMEL Equipe de documentação da. **Implementing Cache Logic with FPGAS**. Disponível em <http://www.atmel.com/atmel/postscript/doc0461.ps.zip>. 2000.
- [5] BERTIN, Patrice; RONCIN, Didier; VUILLEMIN, Jean. **Introduction to pro-grammable active memories**. [S.l.: s.n.], 1989. Digital Equipment Corporation, Paris Research Lab (PRL Report 3).
- [6] CALLAHAN, Timothy J.; HAUSER, John R.; WAWRZINEK, John. **The Garp architecture and c compiler**. Computer Magazine, n.4, p.62_69, 2000.
- [7] DEHON, A. **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] DEHON, André; CASPI, Eylon; CHU, Michael; HUANG, Randy; YEH, Joseph; MARKOVSKY, Yury; WAWRZYNEK, John. **Stream Computations Organized for Reconfigurable Execution (SCORE): Introduction and Tutorial**. FPL’2000 - Proceedings of Field-Programmable Logic and Applications, 2000.
- [9] DUBOIS, Michael; JEONG, Jaeheon; SONG, Yong Ho; MOGA, Adrian. **Rapid hardware prototyping on RPM 2**. IEEE Design and Test of Computers, n.3, p.112_118, 1998.
- [10] GOKHALE, Maya; et. Al. **Splash, a reconfigurable linear logic array**. International Conference on Parallel Processing, v.9, p.219_314, 1990.
- [11] GUCCIONE, Steve. **List of FPGA-based computing machine**. Disponível em http://www.io.com/~guccione/HW_list.html Agosto, 2000.

- [12] 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.
- [13] J. M. Arnold; et al. **The Splash 2**. In: Symposium on parallel algorithms and architectures, 1992. Anais. . 1992. p.316_324.
- [14] LEWIS, David M.; GALLOWAY, David R.; IERSSEL, Marcus van; ROSE, Johnathan; CHOW, Paul. **The Transmogrifier-2: a 1 million gate rapid prototyping system**. Nova Iorque, EUA: [s.n.], 1998. IEEE Transactions on VLSI systems. p.188_198.
- [15] NATIONAL, Equipe de documentação da. **Navigator**. Disponível por WWW em <http://www.national.com/appinfo/milaero/files/f61.pdf> (Julho 1998).
- [16] PERISSAKIS, Stylianos; JOO, Yangsung; AHN, Jinhong; DEHON, Adré; WAWRZYNEK, John. **Embedded dram for a reconfigurable array**. In: VLSI '99, 1999, USA. Anais. . . 1999.
- [17] SANCHEZ, Eduardo; AL. et. **Static and dinamic configurable systems**. Nova Iorque, EUA: IEEE Transactions on Computers, 1999. p.556_564.
- [18] SIDSA, Equipe de documentação da. **Fipsoc mixed signal system-on-chip**. Disponível por WWW em <http://www.sidsa.com/FIPSOC/fipsoc1.pdf> (Setembro 1999).
- [19] VAHID, Frank; GIVARGIS, Tony. **Platform Tuning for Embedded Systems Design**. IEEE Computer, California, p.112_114, 2001.
- [20] VILLASENOR, J.; MANGIONE-SMITH, W. H. **Configurable Computing**. Scientific American, pp. 54-59, June 1997.
- [21] WAINGOLD, Elliot; TAYLOR, Michael; SRIKRISHNA, Devabhaktuni; SARKAR, Vivek; LEE, Walter; LEE, Victor; KIM, Jang; FRANK, Matthew; FINCH, Peter; BARUA, Rajeev; BABB, Jonathan; AMARASINGHE, Saman; AGARWAL, Anant. **Baring it all to software: raw machines**. IEEE Computer, New York, USA, p.86_93, 1997.
- [22] XILINX Inc. Virtex 2.5V Field Programmable Gate Arrays (DS003). Virtex Series datasheet, 2000.
- [23] XILINX, Equipe de documentação. **XC6200 datasheet**. Disponível por WWW em http://dec.bournemouth.ac.uk/drhwh_lib/archive/xc6200.pdf (Junho 1999).