Programação de Processadores Multi-Core: Uma Experiência Educacional Utilizando Plataformas Didáticas Embarcadas em FPGA

Ivan Saraiva Silva, Laysson O. Luz, Ramon Nepomuceno, Jonatas C. dos Santos
Departamento de Computação - DC
Universidade Federal do Piauí - UFPI
Teresina, Barsil
(ivan, layssonluz, ramonnepomuceno, jonatasferreira) @ufpi.edu.br

***Abstract*—A popularização das arquiteturas multi-core impôs a necessidade de inserir modificações nos métodos de ensino da disciplina arquitetura de computadores. A disponibilidade de modelos arquiteturais simuláveis, descritos tanto com linguagens de alto nível quanto com linguagens de descrição de hardware, eleva o nível de eficácia do processo de ensino. Este artigo descreve experiências de ensino realizadas na Universidade Federal do Piauí, tendo como base o desenvolvimento e uso de modelos arquiteturais simuláveis. Com estas experiências mostrou-se que a taxa de sucesso na realização de tarefas associadas a programação para multi-core subiu de 63% para 87% dos alunos envolvidos.**

Keywords—multi-core; ensino; programação paralela; plataformas simuláveis;

I. INTRODUÇÃO

O ensino de arquitetura de computadores tradicionalmente envolve o desenvolvimento de modelos simuláveis de microprocessadores ou o uso de simuladores arquiteturais. Tais modelos são desenvolvidos tanto com o auxílio de linguagens de alto nível, como Java e C, quanto com o auxílio de linguagens de descrição de hardware, como VHDL e SystemC.

As atividades de desenvolvimento e uso dos modelos simuláveis, via de regra, têm como objetivo possibilitar que os alunos ponham em prática os conceitos vistos em sala de aula. Os conceitos abordados nos experimentos práticos são tanto conceitos arquiteturais, quanto conceitos associados ao desenvolvimento de software (aplicações) ou mesmo compiladores e sistemas operacionais.

Com o advento dos processadores multi-core, mudanças consideráveis na forma de ensinar arquitetura de computadores e as demais disciplinas da área, tanto para alunos de graduação quanto para alunos de pós-graduação, se fazem necessárias.

Nos últimos anos, é notável o interesse por adaptações e mudanças curriculares [1], novos cursos [2], novas metodologias de ensino [3] e novas ferramentas [4], que permitam introduzir de maneira satisfatória uma vasta gama de conceitos que sofreram impacto da nova realidade introduzida pelos processadores multi-core.

Do ponto de vista educacional puro e simples, diversas arquiteturas, recursos e ferramentas de software, muitas vezes proprietárias, estão à disposição de professores e instituição de ensino. Entretanto, do ponto de vista da educação para desenvolvimento e domínio tecnológico, muito esforço pessoal e localizado ainda é necessário.

Este artigo apresenta de forma resumida três anos de experiências educacionais realizadas na Universidade Federal do Piauí (UFPI), com foco em educação e desenvolvimento tecnológico na área de projeto e programação de multi-cores.

II. TRABALHOS RELACIONADOS

Artigos recentes [1-3] apresentam a preocupação com bons currículos e boas práticas de ensino relacionadas à programação paralela de multi-core.

Unanimemente, discute-se a dificuldade de abordar o tema programação paralela sem o uso de recursos adequados [3-5]. Por outro lado, com a agora incontornável presença dos multi-cores, esta disciplina cresce em importância

Udugama [4] propõe MCSEP, um processador multi-core configurável, constituído de 16 núcleos SEP (*Students' Experimental Processor*) interconectados por uma NoC-2D. Os núcleos SEP podem ser configurados para executar uma entre seis ISAs (*Instruction Set Architectures*) distintas. Alem disto, MCSEP pode ser configurado para operar segundo o modelo de operação SIMD (*Single Instruction Multiple Data*) ou MIMD (*Multiple Instruction Multiple Data*). A principal deficiência dessa proposta está na indisponibilidade de métodos e ferramentas adequadas de programação. A ausência destes recursos provavelmente faz da programação de MCSEP uma tarefa árdua e desestimulante. Os experimentos apresentados neste artigo mostram a vantagem da utilização de métodos e ferramentas consolidadas.

III. EXPERIMENTOS DE DESENVOLVIMENTO COM GRADUANDOS

Até o início de 2010, não havia no Curso de Ciência da Computação da UFPI experiência acumulada quanto ao desenvolvimento de modelos simuláveis de

microprocessadores ou simuladores arquiteturais. Neste ano, iniciou-se a formação sistemática dos alunos para esta competência. As linguagens adotadas foram VHDL e SystemC e as disciplinas que abordaram estes temas foram Circuitos Digitais, Arquitetura de Computadores e Tópicos em Arquitetura de Computadores.

Ao final do ano de 2010, dispunham-se de quatro modelos de microprocessadores descritos em VHDL:

- **uDIP** – *UFPI's Didactic Processor*: Microprocessador multiciclo didático de 16 bits, tal como apresentado por Hamblen [6];
- **MIPS Multiciclo**: Microprocessador MIPS completo;
- **RISCO**: Microprocessador RISC proposto em [7];

Com estes modelos de microprocessadores e com a experiência de codificação em VHDL, já no início de 2011, foi possível iniciar o desenvolvimento de modelos de plataformas multi-core. As plataformas e ferramentas de programação desenvolvidas desde então são:

A. Plataforma uVMP e uVMP-A&A

uVMP é a sigla de *UFPI's Virtualizable Multi-Core Platform* [8] e uVMP-A&A é a sigla para *UFPI's Virtualizable Multi-Core Platform – Assembly and Assembler*. A plataforma uVMP é dotada de 9 núcleos de processamento (microprocessadores uDIP), sendo um núcleo mestre e oito núcleos escravos (ver figura 1). Duas malhas de interconexão do tipo *crossbar* possibilitam o agrupamento de subconjuntos de núcleos escravos, criando *clusters* independentes. A hierarquia de memória pode ser configurada de modo a atribuir espaço de endereçamento específico a cada *cluster*. O espaço de endereçamento pode ainda ser organizado para acesso compartilhado ou distribuído entre os núcleos do *cluster*.

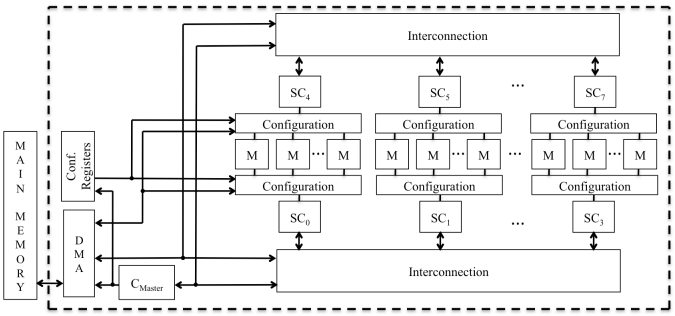

Fig. 1. Visão esquemática de uVMP

Os processadores são dotados de duas instruções de comunicação e sincronização do tipo *Send*/*Receive*. Tais instruções foram didaticamente utilizadas para possibilitar programação paralela.

Com o auxilio de uVMP-A&*A*, *benchmarks* foram desenvolvidos e executados na plataforma uVMP prototipada em FPGA. Os *benchmarks* e os resultados serão apresentados na próxima seção.

B. Plataforma MaRISCO

MaRISCO é um acrônimo para "Mar de RISCOs". A plataforma MaRISCO foi concebida para integrar múltiplos processadores RISCO [7] por intermédio de um subsistema de interconexão. Uma instância dotada de cinco núcleos de processamento, uma malha *corssbar* e memória distribuída foi implementada em VHDL [9], conforme mostra a visão esquemática apresentada na figura 2.

Ao conjunto de instrução dos microprocessadores RISCO, foram acrescidas as duas instruções de comunicação e sincronização do tipo *Send*/*Receive*, já presentes nos microprocessadores de uVMP.

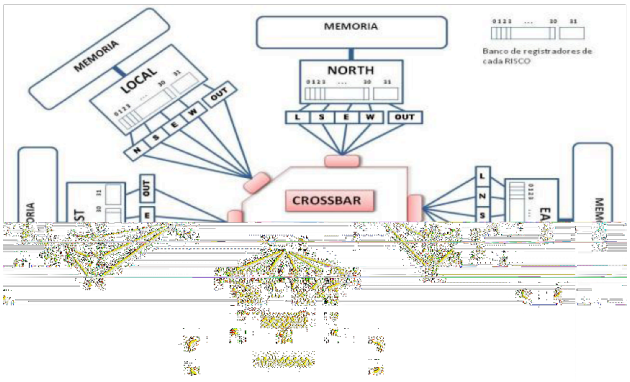

Fig. 2. Visão esquemática de MaRISCO

Para o uso da plataforma MaRISCO, dispunha-se de um compilador LLVM [10] capaz de gerar código objeto para RISCO a partir da linguagem C. Além do compilador, entretanto, não se dispunha de nenhuma ferramenta ou modelo específico de programação paralela. Para esta plataforma, o compilador e as instruções tipo *Send*/*Receive* foram utilizadas para o desenvolvimento de funções de comunicação por troca de mensagem. As funções de comunicação foram então didaticamente utilizadas para possibilitar programação paralela na plataforma MaRISCO.

C. Plataforma MIPNoSys

A partir da experiência de uso da arquitetura MaRISCO, ficou clara a necessidade de oferecer aos alunos, recursos e ferramentas de programação de uso mais consolidado. Com base na estrutura da plataforma MaRISCO, foi desenvolvida a plataforma MIPNoSys, esquematicamente representada na figura 3.

Em MIPNoSys os núcleos de processamento são microprocessadores MIPS, aos quais foi acrescido o subconjunto de instruções de comunicação e sincronização da plataforma IPNoSys [11].

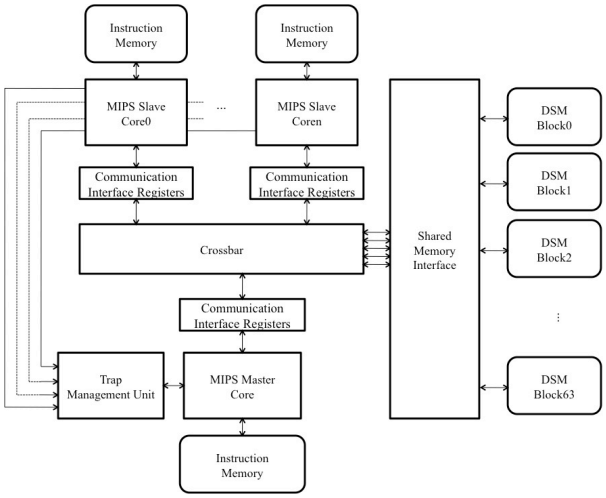

Fig. 3. Visão esquemática de MIPNoSys

A plataforma IPNoSys utiliza um modelo de programação de paralelismo explícito, onde a relação entre as unidades de execução (pacotes no caso de IPNoSys, processos em MIPNoSys) e explicitamente indicada. A figura 4 mostra a modelagem de uma aplicação dotada de seis processos. As arestas *Exec*, *Sync* e *Synexec* são instruções de sincronização, e a aresta *Send* é uma instrução de comunicação.

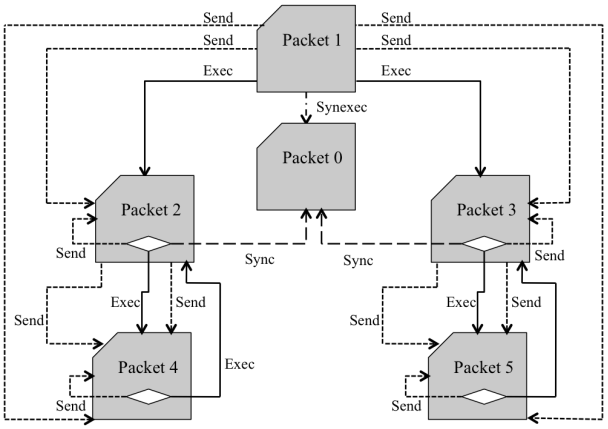

Fig. 4. Modelo de Programação IPNoSys

Nesta plataforma, a programação é feita com linguagem C convencional e a compilação é realizada com o *GNU GCC Cross Compiler*. Com o uso do compilador e de uma metodologia adequada de desenvolvimento, muitos experimentos de programação paralela puderam ser executados.

IV. Experimentos Educacionais com Plataformas Embarcada

Como já mencionado, os experimentos, tanto de desenvolvimento das plataformas, quanto de uso destas como recurso didático, foram realizados nas disciplinas Circuitos Digitais, Arquitetura de Computadores e Tópicos em Arquitetura de Computadores. As duas primeiras, por serem disciplinas regulares e obrigatórias da graduação, tiveram seus programas adaptados de modo a atender também aos seguintes objetivos:

- **Circuitos Digitais**: Domínio das linguagens de descrição de hardware e prototipagem em FPGA;
- **Arquitetura de Computadores**: Desenvolvimento de microprocessadores e plataformas multi-core;

A última disciplina, por ser eletiva e de ementa aberta, teve sua ementa e programa elaborados de modo a cobrir os seguintes tópicos:

- Conceitos de processos e *threads* (revisão); Programação paralela com memória compartilhada; Programação paralela com troca de mensagens: Biblioteca MPI; Mecanismos de comunicação e sincronização e *Software pipelining*; Experimentos de programação em plataformas multi-core prototipadas em FPGA.

Os experimentos de uso das plataformas, descritos a seguir, foram realizados ao longo de três semestres, em 2013 e no primeiro semestre de 2014.

A. Uso Didático da Plataforma uVMP

Nos experimentos de uso de uVMP os alunos utilizaram a ferramenta uVMP-A&A para o desenvolvimento de dois *benchmarks* específicos e de paralelização relativamente simples: Multiplicação de Matrizes e *Merge Sort*. Como a arquitetura permite a configuração da organizado da memória para acesso compartilhado ou distribuído, os experimentos realizados foram os apresentados na tabela 1.

TABELA I. Aplicações

Aplicação	Modelo de Memória	# Núcleos
Matriz	Memória compartilhada	4
Matriz	Memória distribuída	4
Merge Sort	Memória compartilhada	2
Merge Sort	Memória distribuída	2
Matriz/Merge	Compart. / Distrib.	6

Sem a disponibilidade de uso de linguagens de alto nível, a programação com uVMP-A&A agregou muitas dificuldades, tanto relativas ao desenvolvimento quanto à depuração. Em três semanas de desenvolvimento (quatro horas por semana) a taxa de sucesso (implementação e análise dos resultados) foi de 63% dos alunos envolvidos.

B. Uso Didático da Plataforma MaRISCO

A plataforma MaRISCO usa o modelo de memória distribuída. Sua programação é feita com o compilador apresentado em [10] e com funções de comunicação por troca de mensagem. Aos alunos foi solicitado o desenvolvimento de uma biblioteca básica de comunicação por troca de mensagem,

explorando as instruções de comunicação e sincronização (*Send*/*Receive*).

As aplicações solicitadas foram a transformada discreta do cosseno em duas dimensões (DCT-2D) e o cálculo de vetores de movimento para dois quadros de imagem de tamanho Sub-QCIF, usando o algoritmo de busca exaustiva.

A implementação da biblioteca de comunicação agregou certa complexidade ao trabalho, pois o compilador não tinha suporte para as instruções *Send* e *Receive* e estas tiveram que ser inseridas manualmente. A disponibilidade do compilador, entretanto, permitiu que a taxa de sucesso (implementação e análise correta) subisse para 78% dos alunos envolvidos, em três semanas de esforço de desenvolvimento (quatro horas por semana).

C. Uso Didático da Plataforma MIPNoSys

A plataforma MIPNoSys foi concebida de forma a incorporar o conjunto das lições aprendidas com os experimentos anteriores. Uma das mais importantes foi a utilização, tanto quanto possível, de ferramentas já consolidadas. Assim, por exemplo, ficou clara a necessidade de manter o uso da linguagem C e incorporar o uso de um compilador mais estável, com mais tempo de desenvolvimento.

Com base nestes conceitos, a plataforma MIPNoSys adotou microprocessadores MIPS como núcleos de processamento. Uma implementação pipeline do MIPS, incluindo tratamento de todos os casos de conflito, foi desenvolvida em VHDL. Esta implementação mostrou-se, como esperado, aproximadamente 3 vezes mais veloz, em tempo de execução, que o microprocessador MIPS multiciclo [12].

Na plataforma MIPNoSys, todos os núcleos têm memória privada de instrução e acessam de forma compartilhada uma memória de dados, implementada na forma de memória multi-bancos. A implementação como memória multi-bancos possibilita acesso concorrente, economia de energia e baixa dissipação de potência.

As aplicações solicitadas foram a comparação de sequências genéticas segundo o algoritmo de *Smith e Waterman* e o cálculo de vetores de movimento para dois quadros de imagem de tamanho Sub-QCIF, usando o algoritmo *Diamond Search*. Ambas as aplicações são de implementação paralela relativamente mais complexa que as anteriormente solicitadas.

Apesar da maior complexidade das aplicações solicitadas, a taxa de sucesso (implementação e análise dos resultados) subiu para 87% dos alunos envolvidos, em três semanas de desenvolvimento (quatro horas por semana).

V. Conclusões e Trabalhos Futuros

Este artigo apresentou três experiências didáticas de uso de plataformas multi-core embarcadas em FPGA. Nas três experiências, os alunos foram convidados a desenvolver e executar aplicações paralelas para plataformas multi-core em um período de três semanas. Com a melhoria da metodologia de desenvolvimento das aplicações, observou-se que o percentual de alunos que obtiveram sucesso no desenvolvimento das tarefas, considerando o mesmo período de desenvolvimento, aumentou de 63% para 87%.

Nos três experimentos, não foi utilizado o mesmo grupo de alunos, uma vez que as disciplinas foram ministradas em três semestres distintos. Avalia-se, entretanto que as habilidades e competências dos alunos dos três grupos são equivalentes, pois: (i) Os alunos haviam cursado as mesmas disciplinas pré-requisitos, ao ingressarem em Tópicos em Arquitetura de Computadores; (2) A formação ocorreu com o mesmo grupo de professores, com apenas três semestre de intervalo entre a primeira turma e a última; e (3) A análise das médias dos coeficientes de rendimento acadêmico dos três grupos mostrou pequena discrepância.

Referências

[1] Zhongwen Li; Wuling Lv, "Research on Curriculum Design of “Real-time Analysis and Design” Based on Multi-core Platform,"*Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for*, vol., no., pp.2572,2576, 18-21 Nov. 2008

[2] Jianhua Li; Weibin Guo; Hong Zheng, "An Undergraduate Parallel and Distributed Computing Course in Multi-Core Era,"*Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for* , vol., no., pp.2412,2416, 18-21 Nov. 2008

[3] Manogaran, E., "ACt-PBL: An Adaptive Approach to Teach Multi-core Computing in University Education," *Technology for Education (T4E), 2013 IEEE Fifth International Conference on*, vol., no., pp.19,23, 18-20 Dec. 2013

[4] Udugama, L.S.K.; Geeganage, J.; Kuruppuarachchi, W.V., "A configurable multi-core processor for teaching parallel processing,"*Industrial and Information Systems (ICIIS), 2013 8th IEEE International Conference on*, vol., no., pp.326,331, 17-20 Dec. 2013

[5] Jianfeng Yang; Yinbo Xie; Qing Geng; Jolly Wang; Bao, N., "Using cP^{2}BL in Teaching Multi-Core Related Contents," *Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for* , vol., no., pp.2449,2453, 18-21 Nov. 2008

[6] Hamblen, J. O.; Hall, T. S.; Furman, M. D.; Rapid Prototyping od Digital Systems – Quartuss II Edition. Springer. 2006.

[7] A. Junqueira and A. Suzim. Microprocessador RISC CMOS de 32 bits. Master's thesis, Universidade Federal do Rio Grande do Sul, 1993

[8] Silva, I. S.; Nepomuceno, R.; Mafuta, T.; Carvalho, E.S., "uVMP: Virtualizable multi-core platform,"*Informatica (CLEI), 2012 XXXVIII Conferencia Latinoamericana En*, vol., no., pp.1,6, 1-5 Oct. 2012

[9] Luz, L. O.; Silva, I. S.; Soares, T. R. B. S.; MaRISCO - A Multi-Core Platform. In: SFORUM-Student Forum on Microeletronics, 2012, Brasília. Anais - SBCCI Symposium on Itegrated Circuits and Systems Design, 2012.

[10] Vilela, G.; Correa, E.; Kreutz, M., "A LLVM Based Development Environment for Embedded Systems Software Targeting the RISCO Processor," *Computing System Engineering (SBESC), 2012 Brazilian Symposium on*, vol., no., pp.77,82, 5-7 Nov. 2012.

[11] Fernandes, S. R.; Oliveira, B. C.; Costa, M.; Silva, I. S., "Processing while routing: a network-on-chipbased parallel system,"*Computers & Digital Techniques, IET*, vol.3, no.5, pp.525,538, September 2009.

[12] Silva, F. C.; Silva, I. S.; Luz, L. O.; Nepomuceno, R., “Designing a Complete Pipelined Datapath to MIPS ISA: Learning in Pratice”. In: SFORUM-Student Forum on Microeletronics, 2014, Aracaju. Anais - SBCCI Symposium on Itegrated Circuits and Systems Design, 2014. Accepted Paper..