SÍNTESE E INTEGRAÇÃO DE UM AGREGADO RECONFIGURÁVEL APLICADO A UM ALGORITMO GENÉTICO DISTRIBUÍDO

Alexandra Aguiar, Fabiano Kist, Márcio E. Kreutz, João Carlos Furtado, Rafael R. Santos, Tatiana G. S. Santos

Universidade de Santa Cruz do Sul, RS, Brasil

{aaguiar, fabianokist}@mx2.unisc.br, {kreutz, jcfurtado, rsantos, tatianas}@unisc.br

ABSTRACT

Sistemas em grade e agregados de computadores têm sido amplamente empregados como alternativa para aplicações científicas que necessitam de processamento de alto desempenho. Entretanto, é comum que essas aplicações utilizem apenas uma parte do conjunto de instruções de um processador de propósito geral, comumente encontrado nos nodos dos agregados e grades. Como alternativa este trabalho propõe o fluxo de concepção de um agregado de computadores reconfiguráveis, onde um processador customizado é integrado a uma pilha de comunicação para reduzir a latência da rede. Para validar essa idéia, um algoritmo genético distribuído foi mapeado para um núcleo FemtoJava e integrado a uma pilha de comunicação. Resultados demonstram que a alternativa é atrativa, sobretudo, no que diz respeito à área e potência, visto que um nodo ocupa cerca de 30% do FPGA Xilinx VP30 da família Virtex II-Pro e consome apenas 1,1 W. Além disso, quando comparado aos sistemas tradicionais o agregado reconfigurável teve um ganho de desempenho de até 70 % para a aplicação utilizada.

1. INTRODUÇÃO

Agregados e grades tornaram-se uma alternativa bastante atraente para a execução de aplicações que demandam um alto poder computacional, especialmente, pelo custo x benefício apresentado. Normalmente, sistemas desse tipo respondem à demanda de processamento de grande parte das aplicações distribuídas.

Entretanto, o modelo de memória usado em sistemas assim requer que troca de mensagens entre aplicações seja efetuada através de um *pool* de nodos em rede [1]. E mesmo com conexões atuando com alta largura de banda a um custo relativamente baixo, muitas aplicações não se adequam a esse modelo.

Além disso, a maioria dessas aplicações também necessita de recursos específicos em nível de processador, que não são encontrados em um GPP. Assim, uma alternativa natural pode ser a combinação de um processador reconfigurável customizado e um cartão de rede de alta velocidade, dedicados às aplicações que requerem recursos específicos e comunicação eficiente.

Assim, este trabalho apresenta o conceito de um nodo reconfigurável, onde o processador é customizado de acordo com as demandas de uma aplicação específica e integrado com uma pilha de comunicação. Esse modelo reduz a pressão sobre o processador, visto que mensagens são processadas somente e diretamente pela pilha de comunicação e entregues ao processador a uma velocidade proporcional a do processador em si e vice-versa.

Para validar esse conceito, um protótipo foi desenvolvido e funcionalmente verificado. Além disso, o protótipo foi sintetizado e os resultados de síntese e integração foram analisados.

Este trabalho está organizado do seguinte modo: a Seção 2 mostra os trabalhos correlatos, enquanto que a Seção 3 apresenta os detalhes na concepção de um agregado reconfigurável. A Seção 4, por sua vez, descreve um estudo de caso com a implementação de um algoritmo genético distribuído. A Seção 5 mostra como as operações de síntese e validação foram efetuadas ao longo desta pesquisa, enquanto que a Seção 6 apresenta os resultados alcançados. Finalmente, a Seção 7 apresenta as conclusões e trabalhos futuros.

2. TRABALHOS CORRELATOS

Alguns trabalhos previamente desenvolvidos estudam o uso de dispositivos reconfiguráveis em sistemas de agregados, já que esses são dispositivos que têm se tornado cada vez mais populares ao longo dos anos.

Yeh et. al [2] propôs o uso de FPGAs para a concepção de um *switch*. Jones et. al [3] incluiu componentes reconfiguráveis em cada nodo de um agregado.

Outras implementações, como Dandalis [4] propuseram o uso de cartões de rede reconfiguráveis para implementar protocolos específicos, tais como o IPsec. Sass [5] propôs o uso de um cartão de rede inteligente (INIC) capaz de processar mensagens e injetá-las direto na rede, aliviando a pressão no processador e permitindo a exploração de banda e das

latências de redes de alta velocidade modernas. Underwood [6], por sua vez, apresentou uma análise de custos de um agregado adaptável, baseado no projeto INIC.

Mais recentemente, Jacob [7] propôs um *framework* chamado CARMA, destinado ao gerenciamento de diferentes esquemas de configuração para agregados reconfiguráveis. Willians [8] apresentou uma arquitetura de um agregado reconfigurável *on-chip* que suporta bibliotecas para desenvolvimento de sistemas reconfiguráveis multi-core utilizando o padrão MPI (Message Passing Interface).

3. FLUXO DE DESENVOLVIMENTO DE UM NODO RECONFIGURÁVEL

O agregado reconfigurável apresentado neste trabalho é diferente dos demais apresentados em pelo menos dois aspectos.

Primeiro, a abordagem proposta por este trabalho é customizar o processador em si, em função de uma dada aplicação. Isso é realizado através de um fluxo automático que gera um microprocessador a partir de uma aplicação Java [9]. Assim, o microprocessador é dedicado para uma determinada aplicação, permitindo que as otimizações necessárias para essa aplicação sejam efetuadas. Além disso, apenas os recursos necessários para a referida aplicação são sintetizados, ocasionando um processador menor e mais eficaz.

O segundo aspecto que diferencia esta pesquisa das demais reside no fato da integração desse processador com uma pilha de comunicação TCP/IP. Essa pilha implementa *buffers* de transmissão/recepção que são acessadas na mesma freqüência em que o processador trabalha. Assim, o processador consegue explorar mais eficientemente a comunicação de alta velocidade oferecida pela rede.

Além disso, o fluxo de desenvolvimento proposto permite que o programador concentre-se na aplicação, ou seja, no algoritmo propriamente dito. Assim, o uso desse fluxo automático permite um rápido desenvolvimento em um alto nível de abstração, não encontrado em projetos anteriores.

O resultado é um dispositivo fortemente acoplado que integra o processador e a comunicação em um único FPGA. É importante salientar que o objetivo não é propor um dispositivo para substituir nodos de agregados convencionais (GPP) ou um nodo reconfigurável que seja complementar a um PC hospedeiro. Abordagens que funcionam assim foram estudadas anteriormente por [3, 4, 5, 6].

O objetivo deste trabalho é permitir um rápido desenvolvimento de aplicações distribuídas que requerem processadores dedicados de modo a reduzir área, consumo de potência e aumentar o desempenho. É igualmente importante destacar que uma vez que a aplicação é particionada cada nodo pode ter um processador diferente, dedicado e otimizado para uma determinada parte da aplicação.

4. ESTUDO DE CASO: UM ALGORITMO GENÉTICO DISTRIBUÍDO

Uma aplicação real foi utilizada para validar a idéia de fluxo de desenvolvimento proposta neste trabalho. Essa aplicação, amplamente utilizada na indústria farmacêutica, faz uso de técnicas espectrográficas para dosar e caracterizar anti-hipertensivos. A utilização dessas técnicas espectrográficas resulta em uma grande quantidade de variáveis que devem ser ajustadas para o fármaco [10].

Assim, uma solução para a busca do subconjunto de variáveis que melhor se ajuste seria obter o resultado gerado pela análise combinatória de todas as variáveis disponíveis. Isso, porém, torna-se inviável pela alta demanda de tempo e custo computacionais [10].

Como conseqüência disto, diversos métodos de otimização combinatória são empregados, buscando uma forma mais rápida para a obtenção do conjunto de variáveis. Uma técnica que se mostrou bastante efetiva, diz respeito à utilização de Algoritmos Genéticos (AGs).

O AG, inicialmente descrito em uma pesquisa desenvolvida por Holland em [11], é um algoritmo evolutivo baseado na teoria evolucionista de Darwin. Seu objetivo é melhorar uma população de indivíduos na medida em que as iterações passam, através de procedimentos como análise, seleção, cruzamento e mutação.

Assim como em outros casos mais gerais, uma alternativa para o aumento de desempenho de algoritmos genéticos pode ser a aplicação de técnicas de paralelização e distribuição. Isso é necessário, sobretudo, pela ineficiência atingida na execução dessas aplicações em computadores de propósito geral disponíveis no mercado. Vários trabalhos já propuseram diversas formas de paralelização de AGs, dentre os quais destaca-se o Algoritmo Genético Paralelo de granularidade grossa [12].

No AGP de granularidade grossa, também conhecido como modelo da ilha, cada máquina mantém sua própria subpopulação, realizando todas as tarefas típicas de um algoritmo genético (análise, cruzamento e mutação). Isso significa que a população total de uma versão seqüencial do algoritmo poderia ser dividida entre subpopulações menores nos diversos nodos (ou ilhas) utilizados na distribuição [12].

Os nodos participantes da paralelização desse modelo trabalham em conjunto trocando porções das suas subpopulações em um processo chamado de migração. O modelo da ilha está ilustrado na Figura 1.

É possível observar na Figura que cada nodo do agregado traz consigo uma ilha específica do modelo. Cada uma dessas ilhas possui sua própria população e realiza as tarefas típicas de um AG: análise, cruzamento e mutação. Os indivíduos de cada subpopulação são inicialmente gerados sem quaisquer relações com as populações criadas nas outras ilhas. De acordo com a freqüência de migração determinada no início da simulação, elas comunicam-se umas com as outras,

enviando uma parcela de sua subpopulação. Adicionalmente, na fase de migração, mensagens são trocadas entre as ilhas para passar os indivíduos de uma determinada subpopulação para outra.

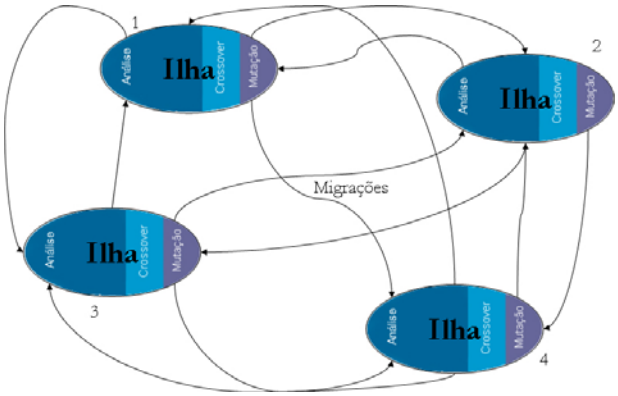

Figura 1: Paralelização do Algoritmo Genético de acordo com o modelo da ilha.

Dessa forma, o algoritmo que implementa o modelo da ilha é dividido nos seguintes procedimentos:

- Análise – Avalia os indivíduos da população utilizando fórmulas matemáticas;
- Cruzamento – Realiza cruzamento entre os indivíduos;
- Mutação – Realiza mutação na população atual, gerando a nova população;
- Comunicação entre Ilhas – Cada nodo envia e recebe dos demais um determinado número de indivíduos. Após completar o procedimento é necessário integrar os novos indivíduos à sua própria população.

No contexto do presente trabalho, cada nodo reconfigurável implementa uma ilha do AGP de granularidade grossa. No caso implementado, todas as ilhas são similares entre si.

5. SIMULAÇÃO, SÍNTESE E VALIDAÇÃO

A prototipação do sistema foi realizada utilizando como alvo a placa Xilinx XUP – V2P, que possui o FPGA XC2VP30 da família Virtex II pro e foi projetada pela Digilent Inc. Além do poderoso FPGA (que possui, inclusive, dois processadores PowePCs 405 *hardwired*), essa placa possui diversas interfaces de E/S, tais como RS-232, PS/2, LEDs, saída VGA, dentre outras.

Após a geração do núcleo FemtoJava para implementar cada [13] ilha em um nodo, simulações foram executadas para verificar a funcionalidade. Nessa etapa, o software ModelSim da Mentor Graphics foi utilizado. Os dados de referência para verificação foram extraídos do mesmo algoritmo modelado em Java e executado em um agregado convencional.

Os *buffers* de transmissão/recepção para a sincronização do núcleo e pilha foram implementados em VHDL, sendo integrados ao núcleo FemtoJava em seguida. Novas simulações foram efetuadas para verificar a funcionalidade dos dois blocos integrados.

Após as simulações, o processo de síntese/depuração foi iniciado através da utilização da ferramenta ISE Foundation da Xilinx. O FPGA alvo escolhido foi o XC2VP30 da família Virtex II pro, disponível na placa de desenvolvimento utilizada no projeto.

Para a implementação da pilha TCP/IP, utilizou-se uma biblioteca escrita em C e executada em um dos processadores PowerPC disponíveis na placa de prototipação. A biblioteca, chamada *Light Weight IP* (LWIP) [14], implementa uma pilha TCP/IP 10/100 para processadores embarcados disponibilizando duas interfaces principais para programação: a *Raw API* e a *Sockets* API. Neste trabalho foi utilizada a interface *Sockets API* que permitiu uma comunicação baseada em *Sockets BSD*. A verificação da parte de comunicação pode ser realizada interligando o nodo reconfigurável a um PC hospedeiro que executa uma aplicação através dessa API de comunicação.

A biblioteca LWIP, disponibilizada pela ferramenta EDK Platform Studio da Xilinx, utiliza a interface Ethernet existente na placa e foi ligada ao FPGA através do barramento PLB, disponível através da arquitetura *CoreConnect* da Xilinx e responsável pela interligação de E/S – FPGA – PowerPCs. A ferramenta EDK Platform Studio também foi utilizada para integrar os blocos, visto que a mesma provê todo o ambiente necessário para o desenvolvimento de projetos que utilizam a arquitetura *CoreConnect*.

A partir daí, o *bitstream* produzido pela ferramenta EDK Platform Studio foi carregado no FPGA alvo através de uma interface de programação USB2 (JTAG).

A verificação da funcionalidade do nodo (integração da pilha de comunicação, *buffers* e núcleo) foi efetuada através do monitoramento de resultados enviados pelo mesmo para a porta RS-232. Com isso, foi possível observar os resultados obtidos através de um terminal aberto em um PC hospedeiro. Esses resultados foram comparados com os dados obtidos em simulação e através da versão em software previamente desenvolvida na linguagem Java.

A próxima Seção apresenta os principais resultados de síntese obtidos na pesquisa.

6. RESULTADOS

Como já discutido, cada nodo foi sintetizado para o FPGA XC2VP30, presente na placa XUP – V2P. A seguir, o sumário e análise dos resultados mais relevantes atingidos pelo processo de síntese.

Tabela 1: Dados de síntese para o FPGA alvo

Resultado de Síntese para XC2VP30	
Freqüência Máxima	100,59 MHz
Consumo de Potência	1,1 W
Área Ocupada (LUTs)	8.372 (30%)

É possível observar que a utilização do FPGA permaneceu em torno de 30% das LUTs disponíveis, mostrando que há espaço suficiente para a integração de

otimizações e outras funcionalidades, sobretudo, em relação a aplicação alvo. Essas otimizações, entretanto, ainda serão estudadas no futuro e não fazem parte dessa pesquisa.

A freqüência máxima de operação obtida, por sua vez, foi cerca de 100 MHz. Essa freqüência é bastante satisfatória considerando a tecnologia FPGA utilizada. Além disso, considerando que a aplicação está sincronizada com os *buffers* de envio/recebimento e pilha, essa freqüência é suficiente para manter o processamento eficaz.

Adicionalmente, é importante destacar o baixo consumo de potência de cada nodo, cerca de 1,1 W. Esse resultado demonstra o quanto a potência pode ser reduzida ao utilizar circuitos específicos para uma determinada aplicação, tendo em vista que a média de consumo de potência do GPPs utilizado no agregado convencional – Pentium 4 – é de 90W. Já os resultados mais relevantes em relação às arquiteturas tradicionais estão exibidas a seguir.

Tabela 2: Agregados Reconfiguráveis x Tradicionais

Resultado das comparações entre agregados			
	Reconfig.	Tradicional	Diferença (%)
Média de tempo de execução (s)	14,16	22,53	37,15
Menor tempo de execução(s)	5,36	19,5	72,51

Através da análise da tabela é possível observar o quão vantajosa a utilização do agregado reconfigurável pode ser, uma vez que de fato, latências existentes em hardware e software impactam negativamente no desempenho da execução de uma determinada aplicação.

7. CONCLUSÕES

Este trabalho propõe uma nova alternativa para distribuição eficiente de aplicações embarcadas através de um agregado reconfigurável. De acordo com o fluxo apresentado, os núcleos dos nodos devem ser desenvolvidos na linguagem Java, sendo que o VHDL sintetizável é gerado automaticamente através da ferramenta SASHIMI. Após a criação do núcleo, o mesmo deve ser integrado com os *buffers* de transmissão e recebimento, bem como com a pilha de comunicação.

A aplicação implementada foi um algoritmo genético aplicado à análise espectrográfica, um problema bastante conhecido na indústria química e farmacêutica. Os nodos foram então completamente implementados e integrados. O processo de verificação da comunicação de dois nodos foi completo e obteve-se um aumento de desempenho de até 70% quando comparado às arquiteturas tradicionais.

O nodo reconfigurável usando a placa XUP-V2P alcançou quase 100 MHz de freqüência e consome apenas 30% do dispositivo, um FPGA XC2VP30 Virtex II Pro da Xilinx. Além da alta freqüência, ainda existe bastante espaço para otimizações, visto que o núcleo foi completamente posicionado em apenas 30% das LUTs disponíveis no FPGA. Algumas otimizações manuais podem ser facilmente realizadas, já que um fluxo automático foi utilizado para gerar o núcleo FemtoJava. Adicionalmente, novas alternativas para comunicação entre os nodos serão estudadas e analisadas. Além disso, novas aplicações serão desenvolvidas de modo a verificar a eficiência dessa abordagem.

8. REFERÊNCIAS

[1] Tanenbaum, A. *Distributed Systems: Principles and Paradigms*. 2002.

[2] Yeh, C. –C; Wu, C. –H.; Juang, J. –Y. “Design and Implementation of a Multicomputer Interconnection Network Using FPGAs”. Proc. of the IEEE Symposium on Field-Programmable Custom Computing Machines. Napa Valley, California, 1995.

[3] Jones, M.; Scharf, L.; Scott, J.; Twaddle, C.; Yaconis, M.; Yao, K.; Athanas, P. “ Implementing an API for Distributed Adaptative Computing Systems”. Proc. of the IEEE Symposium on Field-Programmable Custom Computing Machines, 2000.

[4] Dandalis, A.; Prasanna, V.; Rolim, J. “An Adaptative Cryptographic Engine for IPSec Architectures”. Proc. of the IEEE Symposium on Field-Programmable Custom Computing Machines, 2000.

[5] Sass, R.; Underwood, K.; Lingon, W. “Design of Adaptable Computing Cluster”. Proc. of the Military and Aerospace Programmable Logic Device International Conferences, 2001.

[6] Underwood, K.; Sass, R.; Ligon, W. “Cost Effectivesess of an Adaptable Computing Cluster”. Proc. of the ACM/IEEE Supercomputing, 2001.

[7] Jacob, A.; Troxel, I.; George, A. “Distributed Configuration Management for Reconfigurable Cluster Computing”. Technical Report, HCS Research Lab/ University of Florida, 2004.

[8] Willians, J.; Syed, I.; Wu, J.; Bergmann N. “A Reconfigurable Cluster-on-Chip Architecture with MPI Communication Layer. Proc. of the IEEE Symposium on Field-Programmable Custom Computing Machines, 2006.

[9] Ito, S.; Carro, L.; Jacobi, R. “Sashimi and FemtoJava: Making Java Work for Microcontroller Applications”. IEEE Design & Test of Computers, 2001.

[10] Ferrão, M.; “Algoritmo Genético Empregado na Otimização de HCA de Espectros por Reflexão Difusa no Infravermelho de Medicamentos Anti-Inflamatórios. “Revista Tecnológica”. v. 8, 2004.

[11] Holland, J. “Adaptation in Natural and Artificial Systems”. University of Michigan Press, 1975.

[12] Alba, E.; Troya, J. “A Useful Review on Coarse Grain Parallel Genetic Algorithms”. Universidad de Málaga, Espanha, 1997.

[13] Xilinx. “Virtex-II Pro Datasheet”. Disponível por www em:<http://www.xilinx.com/publications/products/v2pro/ds_pdf/ds083.htm>. Acesso em 17 de novembro, 2006.

[14] LWIP. “A Lightweight TCP/IP Stack”. Disponível por www em: < http://www.sics.se/~adam/lwip/ >. Acesso em 4 de novembro de 2006.