PublICa III (2007) 09 - 17

Implementação de uma plataforma MP-SoC baseada em NoC com solução de diretório para manutenção da coerência de cache

Gustavo Girão[1], Bruno Cruz de Oliveira[1], Ivan Saraiva Silva[2]

[1]Bolsista FINEP/SBTVD, [2]Professor Orientador, Departamento de Informática e Matemática Aplicada, Laboratório Natalnet, Universidade Federal do Rio Grande do Norte

---

Resumo

Com o aumento da complexidade de aplicações, principalmente as relacionadas à multimídia, surge a necessidade de um suporte de *hardware* computacionalmente mais poderoso. A solução adotada para suprir esta necessidade é o uso de paralelismo. Neste contexto, surgem os sistemas inteiros multi-processados em *chip* único. Neste artigo, é apresentada uma plataforma multiprocessada, utilizando-se uma rede em *chip* (NoC, do inglês *Network-on-chip*) para comunicação entre componentes, e adotando a solução de diretório, como mecanismo de coerência de cache. São apresentados resultados que mostram que esta solução não se caracteriza como um grande *overhead* em termos de tempo de resposta, bem como carga inserida na rede.

Palavras-chave: MP-SoC, Network-on-chip, Coerência de Cache, Diretório.

Abstract

With the increasing of applications complexity**,** specially the ones related to multimedia, a computationally powerful hardware support becomes necessary. The solution adopted to fulfill this need is parallelism. In this context, full multiprocess systems in a single chip arise. This paper presents a multiprocessor platform that uses a Network-on-chip as communication components, and adopting the directory solution, as a cache coherence mechanism. The results shows that this solution does not present itself as a great overhead in terms of response time, as well the load inserted on the network.

Keywords: MP-SoC, Network-on-chip, Cache Coherence, Directory.

Introdução

Nos últimos anos, a capacidade de processamento dos microprocessadores tem evoluído em várias ordens de grandeza, entretanto, a complexidade dos sistemas tem aumentado de forma igual ou superior, como por exemplo, o processamento de vídeo de alta definição em tempo-real, que requer 3 bilhões de operações por segundo e geração sintética de vídeo que pode chegar a 1 trilhão de operações por segundo (BERTOZZI et al., 2005). Em aplicações como estas, e considerando que o desempenho dos processadores atuais se aproxima do limite físico, percebe-se que um único processador não atende a necessidade computacional de tais aplicações. Baseado nisso e na crescente capacidade de integração, surgiram os sistemas multiprocessados em chip único (MP-SoCs, do inglês *Multi-Processor System-on-Chip*) (LOGHI et al., 2004).

O MP-SoC é um sistema inteiro integrado em *chip* único. Os MP-SoC, além de múltiplos processadores, integram co-processadores, processadores digitais de sinais (DSP), blocos de memória, blocos de entrada/saída e dispositivos que gerenciam a comunicação interna ao chip (JERRAYA e WOLF, 2004).

Nestes sistemas complexos, a comunicação entre os módulos é crítica, por isso tem sido alvo de diversas investigações arquiteturais (BERTOZZI et al., 2006). A mais simples das soluções, tanto no que se refere à implementação quanto no que se refere ao custo, é o uso de barramentos. O problema dos barramentos se encontra na baixa escalabilidade, ou seja, na degradação do desempenho à medida que o número de dispositivos aumenta.

Uma solução mais atual e que é objeto de pesquisas mais recentes, é o uso de uma rede em *chip* (NoC, do inglês *Network-on-Chip*) como elemento de interconexão. Essa solução apresenta maior flexibilidade devido ao fato de poder ser configurável no que diz respeito às topologias de conexão entre os componentes e também por apresentar uma maior escalabilidade (ZEFERINO et al., 2002).

Nesta perspectiva, este trabalho apresenta a implementação de uma plataforma MP-SoC com suporte à manutenção da coerência da hierarquia de memória. A plataforma, batizada de STORM (*MP-**S**oC Direc**t**ory Based Platf**orm***) foi implementada com a linguagem *SystemC* de descrição de hardware.

Plataforma STORM

O projeto baseado em plataforma é um conceito que vem sendo amplamente discutido, tanto no meio acadêmico quanto no contexto industrial. Apesar de este conceito existir há alguns anos, várias definições foram dadas, tornado sua interpretação confusa, entretanto, Sangiovanni-Vincentelli e Martin (2001) definem plataforma como uma abstração que cobre vários possíveis refinamentos de baixo-nível.

O conceito de plataforma está intimamente ligado ao conceito de reuso de componentes. A reusabilidade que caracteriza este modelo de projeto se dá pelo fato do projeto do sistema ser baseado em um conjunto de componentes pré-definidos que se comunica através de um mecanismo de interconexão. Essa característica implica um sistema em um sistema integrado "flexível" e customizável para alguma aplicação em particular (SANGIOVANNI-VINCENTELLI et al., 2004).

Visão geral

STORM é uma plataforma multiprocessada em *chip* único, criada para suportar até 256 módulos. Seu projeto visou à descrição de um sistema real, executando código binário gerado a partir de uma linguagem de alto nível (linguagem C), capaz de prover um conjunto de resultados especialmente úteis na exploração de espaço de projeto. Em particular, estuda-se o desempenho do processador e *overhead* causado pela hierarquia de memória e pela rede de interconexão.

Atualmente, STORM suporta a integração de um processador (*SPARC V8*), um módulo de *Cache*, um módulo de *Memória* e um módulo *Diretório*, cujo objetivo é realizar a manutenção da coerência das caches. Uma vez que se trata de uma plataforma, STORM não possui uma arquitetura fixa. Os módulos suportados por ela podem ser dispostos em qualquer lugar da NoC. Uma configuração genérica da plataforma pode ser vista na figura 1a. Outros módulos *SystemC* podem ser conectados à STORM, contanto que sigam o protocolo de comunicação padrão da NoC.

STORM utiliza o modelo de memória compartilhada, sendo o espaço de endereçamento formado pela soma dos espaços de todos os módulos de memória instanciados. Desta forma, os processos se comunicam através de variáveis compartilhadas, que são protegidas via *mutexes*. Os *mutexes* foram implementados através de uma biblioteca em C, garantindo a atomicidade dos acessos.

Coerência de cache

Atualmente STORM usa o protocolo baseado em diretório para manter a coerência entre todas as caches do sistema. O problema da coerência de cache pode ser facilmente resolvido em um sistema baseado em barramento com o uso do protocolo *snoop* (TANENBAUM, 1999). A implementação do protocolo *snoop* em uma NoC não é impossível, mas representaria um grande *overhead* de comunicação devido à necessidade de realizar o envio de mensagens em *broadcast* para cada acesso de uma cache à memória. Assim, uma escolha viável é o uso do protocolo de diretório, o qual centraliza toda informação que diz respeito ao acesso à memória (HWANG, 1993).

Cada módulo de memória na STORM tem um módulo Diretório que gerencia sua comunicação com a NoC e mantém a coerência de cache. O Diretório tem informação sobre o *status* de todos os blocos na memória a que está associado (bloco limpo ou bloco sujo) e também a informação sobre que caches têm cópias de quais blocos.

STORM utiliza Diretório Dir_n NB, o que significa que são necessários n+1 *bits* por bloco na memória. O valor *n* diz respeito ao número de caches na plataforma (HWANG, 1993). Isto garante que cada processador possa ou não manter cópia de um mesmo bloco, se necessário.

Para armazenar a informação sobre o *status* do bloco, o diretório utiliza duas tabelas: a *Status TAble* (STA) e a *Processor TAble* (PTA). A STA contém informação que diz respeito a todos os blocos de um determinado módulo de memória, como descrito acima. A PTA contém o endereço de NoC de todos os processadores do sistema. É importante perceber que a STA não tem informação alguma sobre em que lugar na NoC estão as caches com as cópias dos blocos. Por isso, o Diretório tem uma tabela com o endereço de NoC de cada uma das caches, a PTA. A PTA e STA foram implementadas como memórias dedicadas do módulo Diretório, logo, não é necessário nenhum acesso à memória do sistema.

O módulo de Cache tem um gerenciador de comunicação das caches (CaCoMa, do inglês *Cache Communication Manager*), o qual realiza a comunicação entre a cache e o roteador da NoC. O CaCoMa tem uma tabela de tradução do endereço virtual para o endereço de NoC (ATA – *Address TAble*), de forma que o CaCoMa pode calcular o módulo de memória ao qual deve ser feito o acesso. Tanto a tabela PTA do Diretório, quanto a tabela ATA do CaCoMa são montadas durante o processo de *boot*. A STA varia de acordo com o comportamento do sistema enquanto que a PTA é estática. A *Adress Table* também é implementada como uma memória dedicada do CaCoMa.

Interconexão

Os barramentos são mecanismos de interconexão de baixa escalabilidade, o que inviabiliza seu uso em MP-SoCs contendo centenas de processadores. O surgimento de NoCs, como elementos de interconexão em MP-SoCs (BENINI e DE MICHELI, 2002), cuja característica maior é a grande escalabilidade, permite um rápido dimensionamento do sistema.

A NoC utiliza características existentes nas redes de computadores para atender aos propósitos dos sistemas paralelos. Por esta razão, o principal componente de uma NoC é o roteador que garante o fluxo de dados entre os sucessivos nós da rede.

STORM utiliza uma NoC para a interconexão dos módulos. O modelo de NoC implementado foi a NoCX4 que utiliza uma topologia de grelha. A NoCX4 utiliza chaveamento VCT, controle de fluxo baseado em créditos, arbitragem *Round-Robin* ou FCFS (*First-Come First-Served*) e armazenamento FIFO (*First-In First-Out*) de tamanho configurável, além de utilizar roteamento dimensional XY determinístico.

Simulações

As aplicações implementadas e simuladas foram um algoritmo de ordenação *mergesort*, um algoritmo para obtenção de um vetor para estimação de movimento e um codificador JPEG. Estas aplicações foram implementadas de forma paralela para que pudessem ser utilizadas na plataforma em uma instância com vários processadores.

A instância ilustrada na figura 1b foi utilizada na simulação com um único diretório para as aplicações de *mergesort* e estimação de movimento, enquanto que a figura 1e, ilustra aquela utilizada para a simulação da aplicação do codificador JPEG.

Foram realizadas simulações com diferentes instâncias da plataforma para que pudesse ser analisado o impacto do uso de mais de um diretório. As figuras 1c, 1d, e 1f, ilustram as demais instâncias utilizadas.

Resultados

Os resultados obtidos foram relativos ao número de ciclos por instrução (CPI), tempo médio de resposta das operações de leitura e escrita e carga total, em *bytes*, que trafega na NoC. As simulações foram realizadas com caches cujo tamanho variava entre 32 *bytes* e 64*Kbytes*.

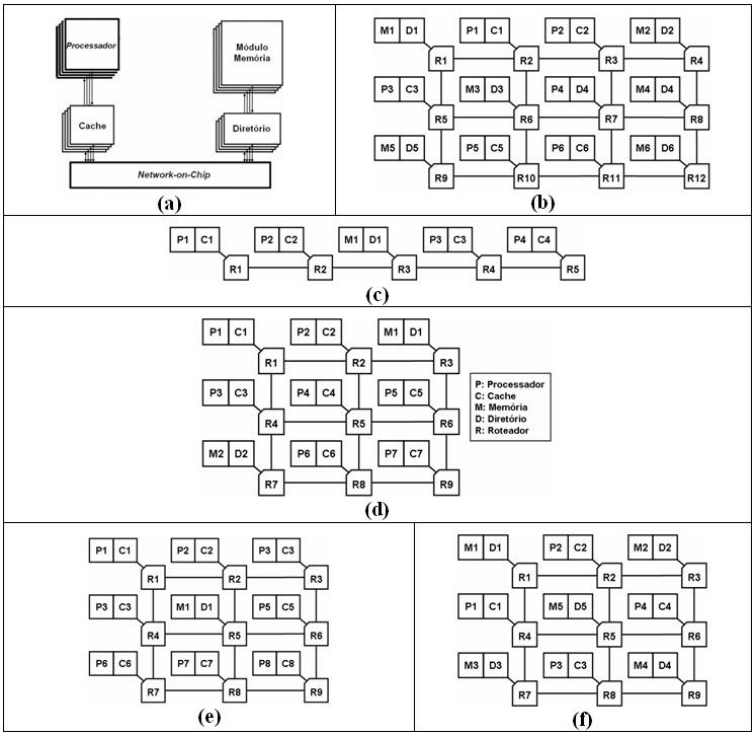

Figura 1 – Instâncias da plataforma STORM.

A figura 2a mostra o gráfico de ciclos por instrução e por número de processadores para a aplicação *mergesort*. Este gráfico mostra que o número de CPI tende a aumentar, conforme o número de processadores no sistema aumente. Isto se deve à utilização de um único módulo de memória, assim, todos os acessos à memória foram atendidos por um único diretório. Dessa maneira, quanto mais processadores, maior a quantidade de requisições e maior o tempo médio de resposta de cada uma.

A figura 2b mostra o número de CPI para as instâncias com mais de um diretório. Pode-se observar que quanto maior o número de diretórios, menor o número de CPI.

Os dados relativos ao tempo médio de resposta das operações de leitura e escrita para a simulação JPEG são ilustrados nas figuras 2c e 2d, respectivamente. A partir destes dados, pode-se concluir que o tempo médio de resposta diminui à medida que o tamanho da cache aumenta, uma vez que quanto menor a necessidade de buscar o dado na memória, menor a espera por ele.

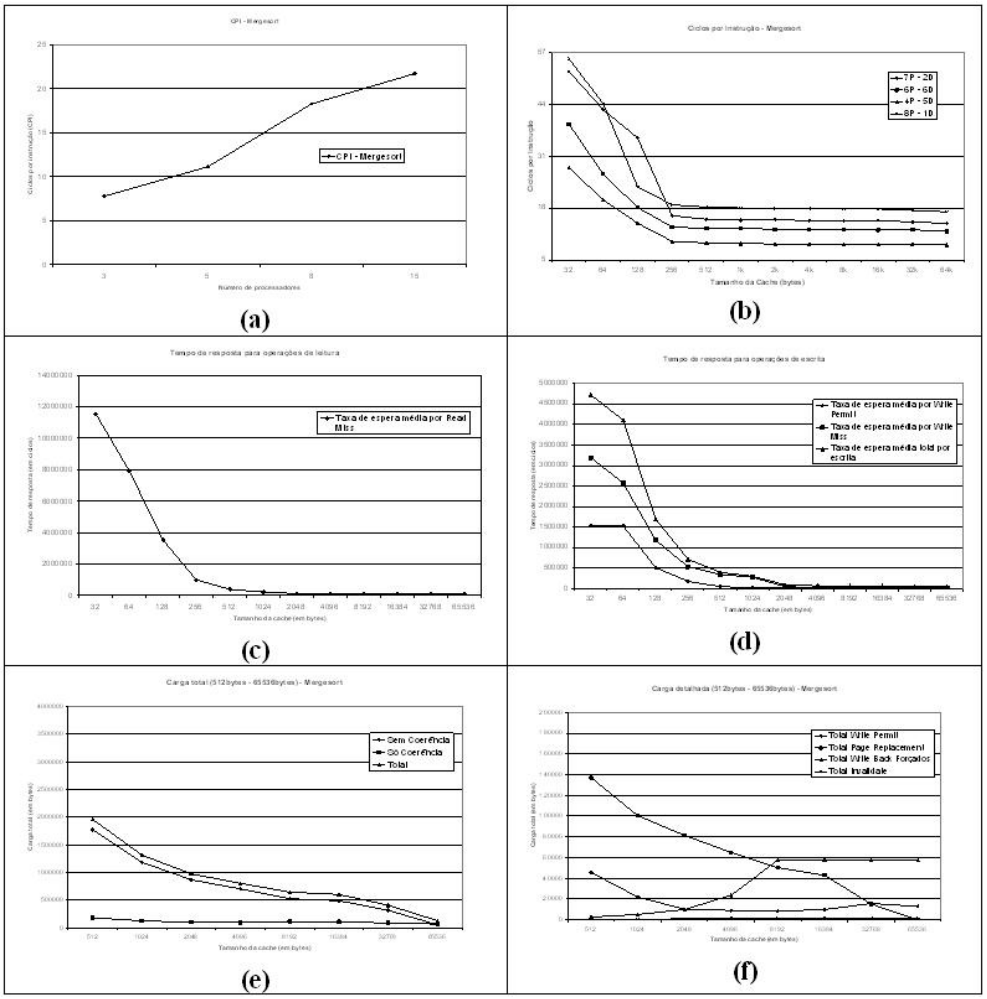

Figura 2 – Resultados de CPI, tempo de resposta e carga inserida na rede.

Com relação à carga que trafega na NoC, pode-se perceber, a partir da figura 2e, que a carga relativa à manutenção da coerência de cache é muito menor do que a carga relativa às outras operações e por isso conclui-se que a solução de coerência de cache adotado não causa um grande *overhead* nesse sentido. Ainda sobre a carga injetada na NoC, pode-se observar, a partir da figura 2f, que quando o tamanho da cache é muito baixo, os pacotes de *page replacement* são os maiores responsáveis pela carga injetada, enquanto que quando o tamanho da cache aumenta o número de pacotes de *write-back* é responsável pela quantidade de *bytes* que trafegam na NoC.

Conclusão

Uma plataforma MP-SoC é uma solução que atende as necessidades de aplicações que demandam um maior poder computacional, por se utilizar do conceito de paralelismo em *chip* único. Além disso, a partir dos resultados, é possível concluir que a solução de diretório é viável em MP-SoCs que utilizam NoC e que não causam um grande *overhead* em termos de tempo de resposta e carga inserida na rede.

Referências

BENINI, L.; DE MICHELI, G. Networks on chip: a new SoC paradigm. **IEEE Computer**, p. 70-78, Jan. 2002.

BERTOZZI, D. et al. NoC synthesis flow for customized domain specific multiprocessor systems-on-chip. **Parallel and Distributed Systems**, v.16, n.2, p.113 – 129, Feb. 2005.

BERTOZZI. S. et al. Supporting Task Migration in Multi-Processor Systems-on-Chip:A Feasibility Study. In: DESIGN, AUTOMATION AND TEST IN EUROPE CONFERENCE AND EXHIBITION, 2006. **Proceedings…** Los Alamitos: IEEE Computer Society, 2006.

JERRAYA, A. A.; WOLF, W. The what, why, and how of MPSoCs. In:_____ (Ed.). **Multiprocessor System-on-Chip.** San Francisco: Morgan Kauffman, 2004. p. 1-18.

LOGHI. M. et al. **Analyzing On-Chip Communication in a MPSoC Environment**. In: DESIGN, AUTOMATION AND TEST IN EUROPE CONFERENCE AND EXHIBITION, 2004. **Proceedings…** Washington: IEEE Computer Society, 2004.

SANGIOVANNI-VINCENTELLI, A.; MARTIN, G. Plataform-based design and software design methodology for embedded systems. **Design & Test of Computers**, v. 18, n. 6, p. 23-33, Nov. / Dec. 2001.

_____ et al. Platform-based system design: benefits and challenges for platform-based design. In: ANNUAL CONFERENCE ON DESIGN AUTOMATION, 41. 2004. **Proceedings…** San Diego: ACM Press, 2004.

TANENBAUM, A.S. **Structured Computer Organization.** 4 ed. Englewood Cliffs: Prentice Hall, 1999.

HWANG, K. **Advanced Computer Architecture**: Parallelismo, Scalability, Programmability. New York: McGraw-Hill, 1993.

ZEFERINO, C. A. et al. A study on communication issues for systems-on-chip. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN, **Proceedings…** p. 121–126, Sep. 2002.

Gustavo Girão
Endereço eletrônico: girao@natalnet.br
Base de Pesquisa: Concepção de Sistemas de Computação (Consiste)
Endereço postal: Departamento de Informática e Matemática Aplicada, Universidade Federal do Rio Grande do Norte, Campus Universitário, Natal/RN 59078-970 – Brasil