Arquitetura PRAM: Estudo de Viabilidade

Helena Lai
{lai@dimap.ufrn.br}

Ivan Saraiva Silva
{ivan@dimap.ufrn.br}

Universidade Federal do Rio Grande do Norte (UFRN)
Centro de Ciências Exatas (CCE)
Departamento de Informática e Matemática Aplicada (DIMAp)

Resumo

A arquitetura paralela surgiu com o intuito de se acabar com o gargalo de von Neumman, permitindo a realização de várias instruções em um mesmo ciclo. Modelos teóricos de tais arquiteturas são muitas vezes utilizados antes de sua construção, pois elimina preocupações com restrições físicas e detalhes de implementação.

Trabalhou-se aqui com o modelo CRCW-PRAM, tratando-se os acessos simultâneos à sua memória compartilhada, de forma que não existam problemas de coerência.

Abstract

Parallel architecture appears with intention to end von Neumann's neck, allowing realization of several instructions at same cycle. Theoretical models of these architectures are often used before their construction because they eliminate worries about physical constraints and implementation details.

We work with CRCW-PRAM model, treating simultaneous access to shared memory so that exist no coherence problems.

1. Introdução

Modelos teóricos de computadores paralelos são abstraídos dos modelos físicos e provêem uma base para o desenvolvimento de algoritmos paralelos, sendo bastante utilizados por construtores de algoritmos, desenvolvedores de chips VLSI e na análise de escalabilidade e programabilidade.

2. O modelo PRAM

O PRAM (Parallel Random-Access Machine) é um modelo teórico que possui custo zero de sincronização e *overhead* de acesso à memória [1]. Neste modelo n blocos de memória são compartilhados entre n processadores, que operam em ciclos sincronizados de leitura, execução e escrita (figura 1). Isto é, todos os processadores lêem, executam e escrevem na memória no mesmo ciclo. Com isso, o modelo deve especificar como leituras e escritas são manipuladas, existindo quatro opções de atualização da memória:

- Leitura exclusiva (ER) - Apenas um processador lê a memória em cada ciclo;
- Escrita exclusiva (EW) - Só um processador escreve na memória por vez ;
- Leitura concorrente (CR) - Leituras simultâneas na mesma região de memória.
- Escrita concorrente (CW) - Escritas simultâneas na mesma região de memória.

Combinações dessas opções geram variações do modelo PRAM, tais como, EREW-PRAM, CREW-PRAM, ERCW-PRAM e CRCW-PRAM.

3. O modelo CRCW

Neste modelo leituras e escritas são concorrentes. Os processadores elementares estão autorizados a solicitar, da memória, operações simultâneas de leitura ou escrita, independentemente do módulo ou posição acessada. Como os módulos de memória são compartilhados entre todos os processadores (figura 1), conflitos podem surgir quando mais de um processador elementar ($\mathbf{PE_i}$) desejar acessar o mesmo módulo de memória (M_i) no mesmo ciclo.

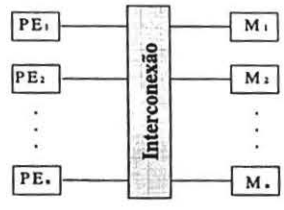

Figura 1 - Modelo teórico

A solução aqui apresentada baseia-se na utilização de um sistema de interconexão capaz de executar, sem conflito, n transferências simultâneas de dados. Uma vez que o

modelo PRAM trabalha em ciclos sincronizados, estudamos a possibilidade de se estender os ciclos conflitantes (leitura e escrita) até que os conflitos sejam resolvidos.

Para o estudo realizado, considerou-se uma arquitetura com 4 processadores elementares, 4 módulos de memória e barramentos de dados e endereço de 8 bits cada. Entretanto a solução proposta obedece ao critério da escalabilidade, o que torna-a independente do grau de paralelismo utilizado (figura 2).

Para cada módulo de memória $\mathbf{M_j}$ ($1 \leq j \leq n$) optou-se pelo uso de um árbitro $\mathbf{A_j}$ [2], que será responsável pelo controle de acesso a este módulo. Além dos barramentos de dados e endereço os PEs foram dotados de três *tokens* **T**, **H** e **C**, cuja funcionalidade será explicada adiante.

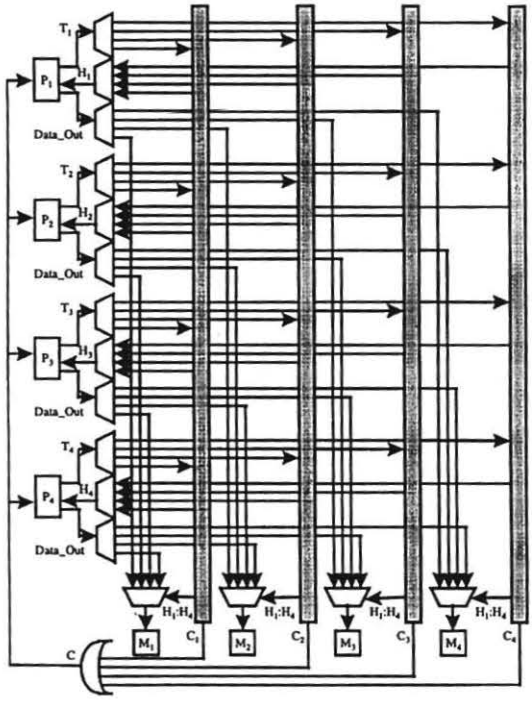

Figura 2 - Solução proposta

Quando um PE_i ($1 \leq i \leq n$) deseja acessar um modulo $\mathbf{M_j}$, o token de estado $\mathbf{T_i}$ é setado e enviado para o árbitro $\mathbf{A_j}$. O árbitro, executando uma função de arbitragem [2], retorna *tokens* de habilitação $\mathbf{H_i}$, indicando quais dos PEs concorrendo ao acesso tem prioridade naquele instante. Observa-se que apenas um token de habilitação será posicionado. Isso faz com que apenas o PE que receber **H=1** acesse aquela memória no ciclo corrente. Caso exista conflito, o token **T** do PE que teve sua solicitação atendida deve ser resetado e uma nova

arbitragem será iniciada, até que todos os processadores tenham executado seus acessos à memória.

Para a manutenção do sincronismo entre os processadores, propomos a extensão dos ciclos conflitantes. Para isto cada árbitro gera um sinal (C_j) indicando os conflitos. O token C acrescentado aos PEs é posicionado sempre que um dos árbitros indica a existência de conflito. Tal token desabilita a passagem dos PEs ao ciclo de operação seguinte.

O fluxo de dados e tokens são conduzidos, aos módulos de memória e árbitros, através de barramentos *tree-state* e multiplexadores (figura 2). A habilitação destes barramentos e multiplexadores é inteiramente feita por bits do barramento de endereço dos processadores e pelo vetor de tokens H, gerado pelos árbitros. Para a configuração estudada, com barramento de endereço de 8 bits e quatro módulos de memória, os dois últimos bits (E_7 e E_6) do endereço a ser acessado serão utilizados para a habilitação.

4. Conclusão

As operações descritas anteriormente são realizadas paralela e independentemente, ou seja, diferentes módulos de memória podem ser manipulados em um mesmo ciclo sem que isso cause interferências no funcionamento geral da arquitetura proposta.

Os acessos à memória do modelo CRCW-PRAM foram solucionados de forma simples. Para isso foram utilizados, basicamente, árbitros associados a cada um dos módulo de memória e, apesar de ter-se trabalhado com 4 processadores e 4 módulos de memória, a arquitetura foi projetada de forma escalonável.

Nosso estudo propõe também um modelo simplificado de processador elementar, de modo que arquitetura proposta adapte-se a um ambiente de ensino de temas em arquitetura de computadores, software básico e processamento paralelo [3]. A apresentação do processador, bem como do ambiente de ensino fogem do escopo deste artigo.

5. Referências

[1] Hwang, K. "*Advanced Computer Architecture: Parallelism, Scalability, Programmability*", McGraw-Hill, 1993

[2] Archambaud, D. e Faudeway, P. "*An Arbitration Tree Adapted to Object Oriented Associative Memories*", ICCD Cambridge (MA) Octobre, 1994.

[3] Silva, I. S. "Arquitetura Reconfigurável Flexível de Apoio ao Ensino e à Formação Técnica em Informática" Projeto de Pesquisa, MCT(RHAE)/CNPq (Proc. N° 610.038/96-6).