AMBIENTE DE SIMULAÇÃO DE MULTIPROCESSADORES COM ARQUITETURA PARALELA

J. Homero F. Cavalcanti
Joberto S.B. Martins
Gurdip Singh Deep
Departamento de Engenharia Elétrica
Centro de Ciências e Tecnologia
Universidade Federal da Paraíba
Av. Aprígio Veloso s/n, Bodocongó
Campina Grande - PB CEP 58100

RESUMO

Apresentam-se algumas considerações sobre topologias, programação e aplicações de máquinas paralelas. Descrevem-se também alguns detalhes do ambiente de simulação no IBM PC para máquinas paralelas com ênfase na topologia hiper-cubo e os resultados obtidos na simulação dessa máquina.

ABSTRACT

A few considerations about the topologies, programming aspects and applications of parallel architecture of computer systems are presented. Some details of simulation environment for parallel machines, in a IBM PC, with emphasis on hyper-cube topology and some simulation results of these machines are also described.

1. INTRODUÇÃO

Os progressos obtidos no desenvolvimento tecnológico de circuitos integrados em larga escala possibilitam a implementação de multi-computadores ou de máquinas de arquitetura paralela compostas de um conjunto de processadores se comunicando numa rede via interconexões de alta velocidade. Frequentemente, duas organizações de processadores para essas máquinas paralelas são propostas: a organização processador-memória (figura 1a), que corresponde aos sistemas que se comunicam via memória compartilhada; e a organização processador-processador (figura 1b), que corresponde aos sistemas que se comunicam usando a técnica de mensagem passante ("message-passing"). Essas arquiteturas utilizam um grande número de processadores que possibilitam a execução rápida de algorítmos complexos anteriormente executados em mono-processadores com grande dificuldade [1].

São possíveis diversos modos de operação dos processadores em máquinas paralelas. Os processadores MIMD ("Multiple Instruction stream-Multiple Data stream"), geralmente são constituídos de N processadores e de N memórias. Processadores SIMD ("Single Instruction stream-Multiple Data stream"), consistem tipicamente de N processadores, de N memórias, de uma rede de interconexão, e de uma unidade de controle [2] Neste trabalho, a ênfase é dirigida a máquinas paralelas tipo SIMD, baseadas em microprocessadores, para aplicações em tempo real [3][4].

Um aspecto importante a ser considerado no projeto de máquinas paralelas, tipo processador/processador, é a performance da rede de interconexão dos processadores [5][6]. A referência 6 classifica os processadores paralelos em três tipos: **a)** processadores tipo bite serial caracterizados pela grande quantidade de processadores com baixa capacidade de processamento; **b)** configurações de alto desempenho, caracterizadas pelo baixo número de processadores e tecnologia avançada, representado pelo super computador CRAY X-MP; **c)** representando a alternativa intermediária, tem-se as máquinas baseadas em microprocessadores, onde cada nó da rede é um microprocessador que possui a sua memória particular juntamente com um controlador de comunicação capaz de enviar e receber mensagens sem atrasar o processador. Esse tipo de máquina paralela, caracterizado pelo iPSC da Intel, foi desenvolvido para a execução de programas baseados em algoritmos que necessitam de tarefas concorrentes que devam ser executadas assincronamente nos diferentes nós de um hipercubo e se comuniquem via mensagens passantes [6].

Têm sido sugeridas diversas técnicas para a avaliação da performance de uma rede de computadores [6], e mais especificamente da sua topologia. Entre essas formas, por exemplo, a distância inter-nó ou diâmetro da rede é usada como referência para a caracterização da performance de uma máquina paralela [7]. Em [7], Finkel descreveu critérios para a escolha ou adequação de topologias para interconexão de muitos processadores do mesmo tipo, citando como exemplo o hipercubo e a sua relação com esses critérios.

O surgimento de diferentes estruturas de máquinas paralelas tornou a atividade de programação mais difícil e mais complexa. Programadores necessitam de ferramentas de software para expressar o paralelismo necessária a uma programação eficiente [8]. Máquinas paralelas com memória compartilhada necessitam de recursos de software básico (tipo "FORK" e "JOIN" [9]) que permitam a sincronização dos processadores no acesso à memória compartilhada. No caso de

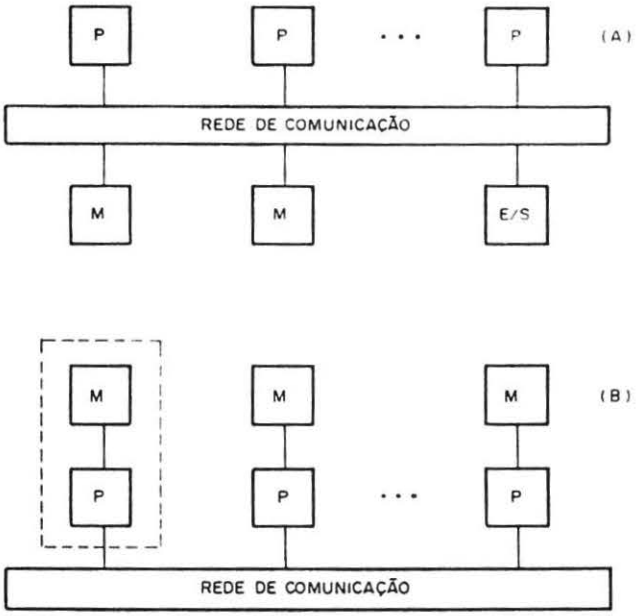

Figura 1. Organização processador/memória e processador/processador

máquinas paralelas com mensagem passante, o mecanismo de software necessário à sincronização da comunicação dos processadores é geralmente do tipo "SEND" e "RECEIVE" [8].

Programas para máquinas paralelas exibem muitos níveis de paralelismo [10][8], comumente chamado de granularidade [11][12]. A granularidade de uma aplicação geralmente é determinada pelo grau de interação do problema e também pelo sistema de processamento paralelo utilizado. O estilo de programação paralela varia, dependendo do tipo de arquitetura utilizada na máquina paralela, isto é, memória compartilhada e mensagem passante. Devido ao alto custo das máquinas paralelas e também à necessidade de se estudarem novas arquiteturas e topologias dessas máquinas, diversas instituições desenvolveram simuladores, baseados em modelos abstratos, para essas máquinas. Vários modelos foram desenvolvidos para analisar a capacidade e performance de máquinas paralelas. Esses modelos abstratos usam ferramentas de software com mecanismos do tipo semáforo ou "FORK" e "JOIN" [13]. Um exemplo de um sistema desenvolvido como ambiente para a programação de máquinas paralelas, do tipo processador/processador, é o POKER [14].

O desenvolvimento de máquinas paralelas, baseadas no conceito de tarefas, necessita do escalonamento dessas tarefas para os diversos processadores. Vários estudos foram feitos para alocação de tarefas [15][16][17] aos processadores das máquinas paralelas. Esses estudos incluem sistemas desenvolvidos para aplicações em tempo real, por exemplo, POOLPO [3], e sistemas paralelos reconfiguráveis [18].

Para aplicações em tempo real, as máquinas paralelas com arquitetura hiper-cubo, apresentam-se como uma das opções mais promissoras devido ao seu reduzido número de ligações entre processadores e a sua adaptabilidade à implementação de diferentes topologias como, por exemplo, anel, árvore, etc, [19][10]. A figura 2 mostra o esquema simplificado de uma máquina paralela com topologia hiper-cubo binária quadrimensional [20]. Nesse esquema é apresentado um computador mestre, geralmente usado para gerenciamento e preparação dos programas paralelos e para a comunicação com o operador ou outro computador [19]. Diversas propostas têm sido feitas para a construção de hipercubos com grande número de processadores [20]. A partir de 1983, algumas empresas do ramo têm desenvolvido hipercubos comerciais, como por exemplo, a Intel, com o seu "Intel Personal Supercomputer", iPSC, que usa microprocessadores de 16 bits (8Ø286) nos nós do hipercubo, com até 128 nós. Diversas configurações de máquinas paralelas, usando topologias hiper-cubo modificadas, têm sido propostas [21] que combinam as vantagens do hiper-cubo com as vantagens de outras topologias.

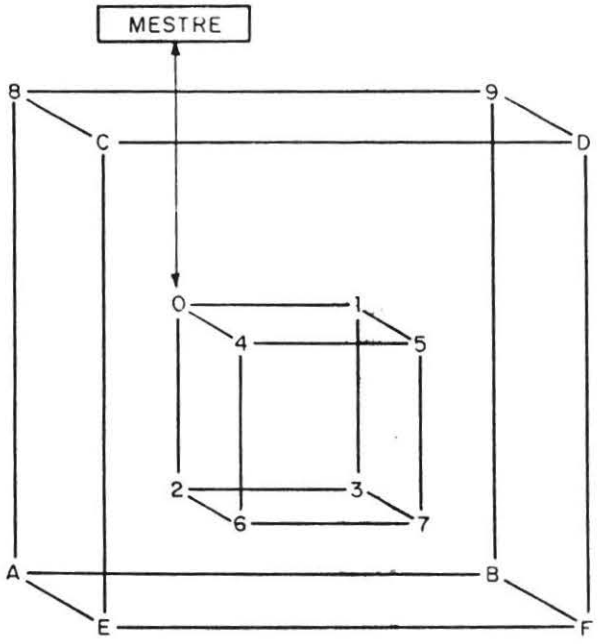

Figura 2. Hiper-cubo quadri-dimensional

A simulação de máquinas paralelas em um sistema hospedeiro, geralmente é feita com linguagens de programação procedurais do tipo C, FORTRAN, PASCAL, etc (por exemplo, o simulador de máquinas paralelas EUCLID, foi implementado em PASCAL [13]). Neste trabalho, apresenta-se o projeto e implementação, num microcomputador compatível com IBM PC, de um ambiente para simulação de máquinas paralelas (SHIC- Simulador de máquinas paralelas com topologia Hiper-Cubo). Usa-se a técnica "descrição executável do modelo" [22], baseado em programação lógica, para a implementação com a linguagem PROLOG do SHIC. Alguns trabalhos têm sido feitos para o

desenvolvimento de uma extensão de PROLOG que permita o uso de primitivas de sincronização do tipo "FORK" e "JOIN" [23]. Para o SHIC, desenvolvemos uma extensão de PROLOG, denominada de ACIOTA [24], que permite a programação paralela dos processadores e se baseia no mecanismo de sincronização "SEMÁFORO" [9]. A seguir, apresentam-se alguns detalhes da simulação do hiper-cubo, resultados obtidos na simulação de uma máquina paralela com topologia hiper-cubo e conclusões e comentários sobre o simulador desenvolvido.

2. SIMULAÇÃO DO HIPER-CUBO

A máquina paralela simulada compõe-se de um computador mestre (M) e de 16 processadores (H) conectados numa topologia hiper-cubo. No SHIC, atribui-se a cada processador da máquina paralela um executivo em tempo real que permite a gerência de tarefas sem preempção e com algoritmo estático para escalonamento dessas tarefas. No SHIC, existem seis tarefas (M1,M2, ...,M6) associadas ao processador mestre e cinco tarefas associadas a cada um dos processadores do hiper-cubo (exceto o processador HØ, a que são associadas sete tarefas).

O funcionamento da máquina paralela simulada é o seguinte: Programa-se o mestre para a geração do programa paralelo. O mestre envia informações (dados, programas, etc) a todos os processadores do hiper-cubo. Para essas funções básicas necessita-se de um protocolo de comunicação que permita a transmissão de dados e programas, em tempo real, entre todos os processadores. Nesta simulação, protocolam-se as mensagens com um cabeçalho contendo o endereço do processador (fonte e destino), o tipo do dado, o seu comprimento, e os dados. A figura 3 mostra os campos do protocolo das mensagens no SHIC. O apêndice A apresenta uma descrição sucinta de algumas características de roteamento de mensagens numa máquina paralela com topologia hiper-cubo.

FONTE	DESTINO	TIPO	COMPRIMENTO	DADOS

Figura 3. Protocolo de comunicação no SHIC.

A figura 4 mostra o esquema das tarefas do mestre juntamente com as tarefas de interface com o processador HØ do hipercubo. Cada tarefa do hiper-cubo foi simulada com um identificador numérico que facilita a determinação de proximidade (vizinhança entre processadores) e de roteamento no hiper-cubo. As tarefas do mestre foram identificadas por M1, M2, M3, M4, M5 e M6. As tarefas do hiper-cubo foram identificadas por T=n*1ØØ+k (figura 5), onde **n** indica o processador no hiper-cubo (n=1 para processador H1) e **k** indica o canal de comunicação entre processadores no hiper-cubo (k=3 para comunicação entre processadores H1 e M3).

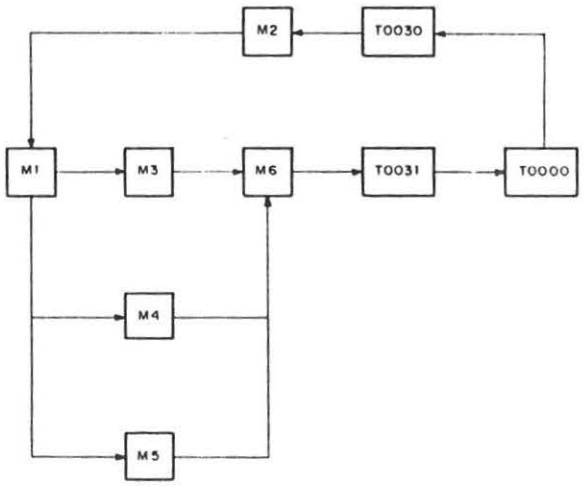

Figura 4. Tarefas do mestre e a ligação com o HØ

A seguir apresenta-se uma descrição sucinta das tarefas do mestre e do HØ.

M1 - Tarefa principal do mestre, compõe-se dos programas de geração dos programas paralelos e dados a serem distribuídos no hiper-cubo.

M6 - Tarefa para a transmissão dos dados a um dos processadores do hiper-cubo (no caso HØ). O processador HØ se encarrega da distribuição desses dados entre todos os demais processadores do hiper-cubo.

M2 - Recepção dos dados do hiper-cubo. Recebe do processador HØ os dados obtidos no processamento requerido pelo mestre.

M3 - Tarefa encarregada da preparação e envio dos programas e dados para o hiper-cubo (TIPO = Ø no protocolo da mensagem). Recebe o programa e dados da tarefa M1 e os codifica para envio dos 16 programas mais dados ao processador HØ.

M4 - Tarefa encarregada da preparação e envio dos programas para o hiper-cubo (TIPO = 1). Recebe programas da tarefa M1 e os codifica para envio ao processador HØ. Permite a execução de diferentes programas nos nós do hiper-cubo.

M5 - Tarefa encarregada da preparação e envio dos dados para o hiper-cubo (TIPO = 2). Recebe os dados da tarefa M1 e os codifica para envio ao processador HØ.

TØØ3Ø - Tarefa do HØ para recepção (SEND) do processador HØ ao mestre.

TØØ31 - Tarefa do HØ para recepção de mensagens (RECEIVE) enviadas pelo mestre.

No SHIC, a cada um dos processadores do hipercubo são associadas 5 tarefas. Uma das tarefas é a principal e as demais são usadas para comunicação com os processadores vizinhos. A figura 5 mostra o esquema de comunicação entre o processador H1 e os seus processadores vizinhos (H3, HØ, H7 e H9). Para o processador H1 foram associadas as tarefas TØ1Ø1, TØ1ØØ, TØ1Ø3, TØ1Ø7 e TØ1Ø9. Nesse esquema só são apresentadas as tarefas dos processadores vizinhos que participam diretamente da comunicação com o processador H1.

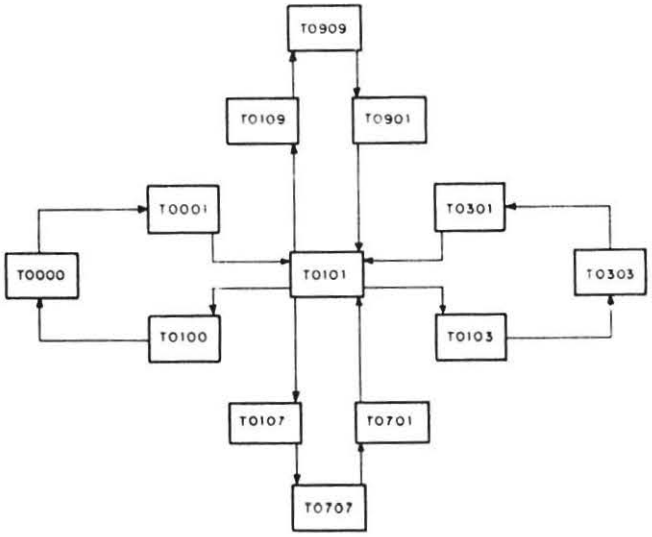

Figura 5. Comunicação entre processadores no hiper-cubo

No exemplo da figura 5 a tarefa principal do processador H1 do hiper-cubo é a TØ1Ø1. Para cada processador Hi no hiper-cubo, o identificador da tarefa principal (TP) é obtido pela fórmula TP = 1ØØ*i+i. O identificador das tarefas de comunicação (TC) com os processadores vizinhos (V) é determinado pela fórmula TPV = 1ØØ*i+V. A seguir apresenta-se uma descrição sucinta das tarefas da figura 5.

TØ1Ø1 - Tarefa principal, encarregada do recebimento do programa e dados do mestre. Recebe do mestre o programa paralelo, executa-o e prepara a comunicação (roteamento de mensagens) entre os processadores do hiper-cubo. Excepcionalmente, o processador HØ se encarrega da comunicação com o mestre via tarefas TØØ3Ø e TØØ31.

TØ1ØØ - Tarefa do processador H1, encarregada da comunicação com o processador HØ. A tarefa principal determina a qual processador deve enviar a mensagem (roteamento), e para cada processador vizinho ao nó é acionada uma tarefa que receberá a mensagem e a enviará ao processador vizinho. Se a tarefa principal do processador vizinho estiver ocupada (usa-se o conceito de SEND), a tarefa de comunicação ficará tentando o envio da mensagem. Quando a tarefa principal do processador vizinho estiver livre a mensagem será enviada (RECEIVE). As tarefas TØ1Ø3, TØ1Ø7 e TØ1Ø9 se comunicam com os processadores H3, H7 e H9, respectivamente.

As tarefas TØØØ1 (de HØ), TØ3Ø1 (de H3), TØ7Ø1 (de H7) e TØ9Ø1 se comunicam com o processador H1.

As tarefas TØØØØ, TØ3Ø3, TØ7Ø7 e TØ9Ø9, são tarefas principais dos processadores HØ, H3, H7 e H9, respectivamente.

3. RESULTADOS OBTIDOS NA SIMULAÇÃO

A figura 6 mostra a tela de vídeo padrão do SHIC quando simulado no IBM PC. Nessa tela pode-se observar duas "janelas". Na janela principal, que circunda toda a tela, no canto superior direito, é apresentado o valor atual do "pseudo relógio". No centro da tela, na sua parte superior, é mostrado o valor atual dos descritores das tarefas do SHIC. No canto superior esquerdo da tela é mostrada a janela do "MENU PRINCIPAL". Nesta janela são apresentadas as opções disponíveis no SHIC. **1.** SIMULAÇÃO: simula a execução da máquina paralela. **2.** EXEMPLO: apresenta um exemplo (geralmente executado em único processador) do sistema a ser simulado. **3.** TEXTO, apresenta o texto do programa SHIC. **4.** FINAL: termina a simulação.

A figura 7 mostra a tela obtida na simulação da execução de uma máquina paralela com topologia hiper-cubo. Usou-se como teste, a simulação da execução de um programa paralelo para cálculo do valor de PI. Na janela principal, na parte superior central é mostrado o descritor da tarefa corrente em execução, e no canto superior direito, é mostrado o valor atual do pseudo-relógio. A janela do lado esquerdo apresenta os valores obtidos na simulação. A janela do lado direito apresenta o esquema da máquina paralela com topologia hiper-cubo quadridimensional, juntamente com o seu processador mestre.

Durante a simulação, o operador pode acompanhar a execução dos processadores no hiper-cubo desenhado na tela de vídeo. Cada processador acionado é indicado pela mudança de seu código de impressão de "normal" para "reverso". Por exemplo, se o processador H5 do hiper-cubo é acionado (p.e. a tarefa TØ5Ø6 do processador H5), o código de impressão "5" no desenho do hiper-cubo ficará no modo "reverso" enquanto todos os demais códigos dos processadores do referido desenho, permanecerão no modo de apresentação "normal".

Os campos da janela do lado esquerdo da figura 7 representam os resultados obtidos da simula-

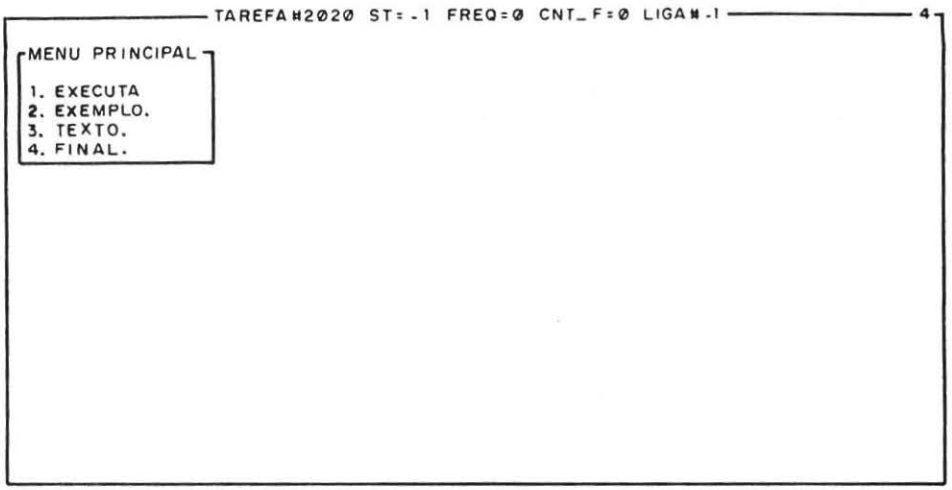

Figura 6. Tela de vídeo padrão do SHIC

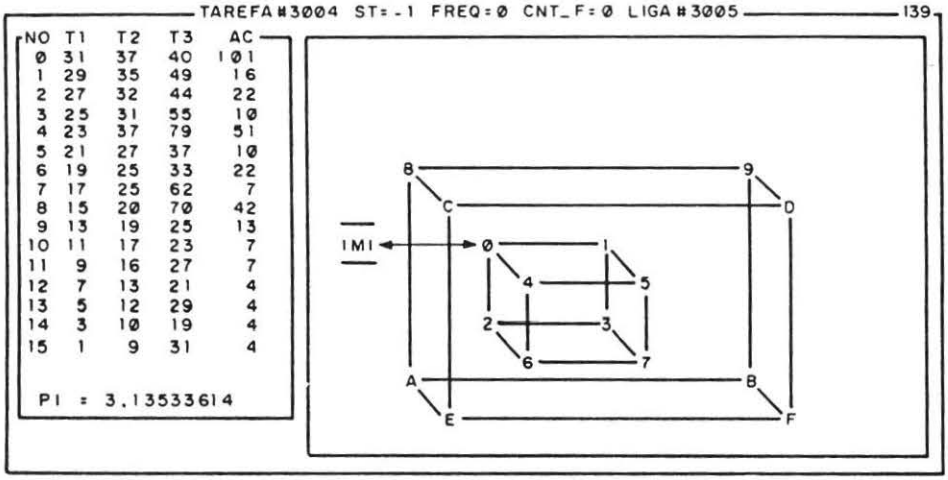

Figura 7. Tela de vídeo com resultados da simulação

ção até o instante indicado pelo pseudo-relógio. A seguir apresenta-se uma descrição sucinta do significado desses campos.

NO' - Mostra o número do processador do hiper-cubo (numerados de Ø a 15).

T1 - Indica o ciclo do pseudo-relógio (p.e., na figura 4, tarefa M3) em que foi gerado o programa a ser executado pelo processador H do hiper-cubo. Observe-se que a cada 2 ciclos do pseudo-relógio é gerado o programa para cada nó do hiper-cubo. Inicialmente é gerado o programa para o nó H15, em seguida para o nó H14, e assim sucessivamente até o nó HØ.

T2 - Indica o ciclo do pseudo-relógio em que cada nó do processador do hiper-cubo recebe a mensagem com o seu programa. Cada mensagem no hiper-cubo deve seguir uma rota para chegar ao seu destino. Por exemplo, a mensagem para o processador H15 segue a seguinte rota: M -> HØ -> H8 -> H9 -> H11 -> H15. Na simulação, para a mensagem percorrer essa rota são acionadas as seguintes tarefas: M3 -> M6 -> TØØ3Ø -> TØØØØ -> TØØØ8 -> TØ8Ø8 - TØ8Ø9 -> TØ9Ø9 -> TØ911 -> T1111 -> T1115 -> T1515.

T3 - Indica o ciclo do pseudo-relógio em que a mensagem com os resultados obtidos pela simulação da execução do processador especificado no campo destino do protocolo das mensagens, chega ao processador mestre. Por exemplo, para o cálculo do valor de PI, a mensagem com o resultado obtido pela execução do processador H15 chega ao processador mestre no ciclo 61 do pseudo-relógio. O roteamento dessa mensagem é semelhante ao do exemplo mostrado no campo T2.

AC - Este campo indica quantas vezes as tarefas do processador H do hiper-cubo foram acionadas. Por exemplo, no ciclo completo da simulação da máquina paralela mostrada na figura 7 foram acionadas 4 tarefas do processador H15.

4. CONCLUSÃO

Apresentou-se o desenvolvimento e implementação de um simulador para testes, avaliação preliminar, e estudo de máquinas paralelas. O uso de programação em lógica, ARITY/PROLOG, para a especificação e construção da ferramenta proporcionou uma visualização global de máquinas paralelas como também permitiu o estudo da comunicação entre microcomputadores usando "mensagens passantes".

O uso de PROLOG permitiu a simulação de algumas das características reais de máquinas paralelas. Por exemplo, o apêndice A apresenta dois algoritmos para roteamento de mensagens no hiper-cubo. O algoritmo 1 determina caminho único para roteamento de mensagens no hiper-cubo. O algoritmo 2, desenvolvido a partir do algoritmo 1, determina caminhos múltiplos para roteamento de mensagens no hiper-cubo. A implementação do roteamento de mensagens com o algoritmo 2 permitiu a escolha de caminhos livres entre os múltiplos caminhos determinados para o roteamento de mensagens. A implementação do algoritmo 2 com a linguagem PROLOG denominamos de "algoritmo de roteamento de mensagens inteligente".

Os resultados obtidos na simulação de uma máquina paralela com 16 processadores, mostrados na figura 7, referem-se a tempo unitário de execução de processadores, isto é, uma unidade de tempo para envio de mensagens, uma unidade de tempo para a execução do programa paralelo, etc. O SHIC permite diferentes unidades de tempo para simulação da execução das máquinas paralelas. Por exemplo, se um sistema a ser simulado usa granularidade grossa, o atraso resultante no processamento pode ser inserido no descritor da tarefa que simula esse atraso.

A arquitetura do SHIC é adaptável para o estudo, testes e avaliação de diversos tipos de topologias de multi-microcomputadores tipo hiper cubo, borboleta, anel, malha, árvore, etc.

REFERÊNCIAS

[1] Haynes, L.S. & Lau, R.L. & Siewiorek, D.P. & Mizell, D.W. "A Survey of Highly Parallel Computing", IEEE Computer, vol. 15, Nº 1, janeiro de 1982, pp.9-24.

[2] Siegel, H.J. & Siegel, L.J. & Kemmerer, F. C. & Mueller, P.T. Jr. & Smalley, H.E. Jr. & Smith, S.D. "PASM: A Partiotionable SIMD/MIMD System for Image Processing and Pattern Recognition", IEEE Transactions On Computers, vol. C-30, Nº 12, dezembro de 1981, pp.934-947.

[3] Kirrmann, H.D. & Kaufmann, F. "POOLPO - A pool of processors for process control applications", Transactions On Computers, vol. C-33, Nº 10, outubro de 1984, pp.869-878.

[4] Manner, R. & Shoemaker, R.L. & Bartels P. H., "The Heidelberg Polyp System", IEEE MICRO, vol. 7, Nº 1, fevereiro de 1987, pp 5-13.

[5] Feng, T. "A survey of interconnection Networks", IEEE Computer, vol. 14, Nº 12, dezembro de 1981, pp.12-27.

[6] Reed, D.A. & Dirk, C.G. "The Performance of Multicomputer Interconnection Networks" IEEE-COMPUTER, vol. 21. Nº 6, junho de 1987, pp.63-73.

[7] Finkel, R.A. & Solomon, M.H. "The Lens Interconnection Strategy", IEEE Transactions On Computers, vol. C-30, Nº 12, dezembro de 1981, pp.960-965.

[8] Karp, A.H. "Programming for Parallelism", IEEE-Computer, maio de 1987, vol. 20, Nº 5 pp.43-57.

[9] Hansen, P.B. "Operating Systems Principles", Prentice-Hall, Englewood Cliffs, N. J.m 1973.

[10] Howe, C.D. & Moxon, B. "How to program parallel processors", Tutorial da revista IEEE SPECTRUM, setembro de 1987, p.36.

[11] Cvetanovic, Z. "The effects of problem partitioning, allocation, and granularity on the performance of multiple-processor systems", IEEE Transactions On Computers, vol. C-36, Nº 4, abril de 1987, pp.421-432

[12] Kruatrachue, B. & Lewis, T. "Grain Size Determination for Parallel Processing", IEEE-SOFTWARE, janeiro de 1988, pp.23-32.

[13] Butler, J.M. & Oryc, A.Y. "A Facility for Simulating Multiprocessors", IEEE MICRO, outubro de 1986, pp.32-44.

[14] Snyder, L. "Parallel programming and the poker programming environment", IEEE COMPUTER magazine, vol. 17. Nº 7, julho de 1984, pp.27-43.

[15] Stone, H.S. & Bokhari S.H. "Control of distributed process", Computer, Julho de 1978, pp.97-196 (republicado no Tutorial of Distributed Control, 2ª edição, IEEE Computer Sociedade, pp.129-134).

[16] Bianchini, R.P. & Shen, J.P. "Interprocessor traffic scheduling algorithm for multiple-processor networks", IEEE Transactions On Computers, vol. C-36. Nº 4,abril de 1987, pp.396-409.

[17] Peng, D. & Shin, K.G. "Modeling of concurrent task execution in a distributed system for real-time control", IEEE Transactions On Computers, vol. C-36. Nº 4,abril de 1987, pp.500-516.

[18] Tuomenoksa, D.L. & Siegel, H.J. "Task pre loading schemes for reconfigurable paral lel processing systems", Transactions On Computers, vol. C-33, Nº 10, outubro de 1984, pp.895-905.

[19] Wiley, P. "A parallel architecture comes of age at last", Tutorial da revista IEEE SPECTRUM, junho de 1987, p.46.

[20] Siegel, H.J. "The Multstage Cube: A Versa tile Interconnection Network", IEEE Compu ter, vol. 14. Nº 12, dezembro de 1981, pp 65-76.

[21] Goodman, J.R. & Sequin, C.H. "Hypertree : A multiprocessor Interconnection Topolo gy", IEEE Transactions On Computers, vol. C-30, Nº 12, dezembro de 1981, pp.923-933

[22] Sidhu, D.P. & Crall, C.S. "Executable Lo gic Specifications for protocol service interfaces", IEEE Transactions on Soft ware Engineering, vol. 14. Nº 1, janeiro de 1988, pp.98-121.

[23] Carlton, M. & Roy, P.V. "A Distributed Prolog System with And Parallelism", IEEE SOFTWARE, janeiro de 1988, p.43-51.

[24] Cavalcanti, J.H.F. & Deep. G.S. "Protóti po de um sistema distribuído para contro le de processos em tempo real", trabalho aceito para apresentação no 7º Congresso Brasileiro de Automática", agosto de 1988 São José dos Campos, SP.

[25] Katseff, H.P. "Incomplete hypercubes", IEEE Transactions On Computers, Vol. 37, Nº 5, maio de 1988, pp.6040-608.

APÊNDICE A

A seguir apresentam-se definições de termos e algumas características e técnicas relativas ao roteamento de mensagens em máquinas parale las com topologia hiper-cubo |2| e algoritmos desenvolvidos a partir dessas propriedades |25|.

Define-se "nó" como o processador num sistema multiprocessador e a "ligação" conectando pro cessadores como "enlace". A lista ordenada de processadores visitada pela mensagem é defini da como "caminho". O comprimento de um cami nho é definido como o número de "enlaces" no caminho. Define-se endereço relativo ("end-rel") de dois nós "a" e "b" como sendo o ou-ex clusivo entre todos os bites do número do seu nó. A distância entre dois nós é definida como sendo o número de "i" (uns) nos seus endereços relativos. Um "enlace" é definido como "enla ce" número i ("enlace i") quando ele conecta dois nós cujos números diferem somente na i-ésima posição.

No hipercubo, em cada nó existe um processador com sua memória local, dispondo de caminhos de comunicação direto com n outros processadores (seus vizinhos). Esses caminhos correspondem aos arcos do cubo. Num hiper-cubo binário com dimensão n existem 2^n endereços binários que podem ser atribuídos aos 2^n possíveis processa dores. Esses endereços binários podem ser re presentados por |Bn-1, Bn-2, ..., B2, B1, B0| e o endereço de cada processador difere do seu vizinho de um único bit (distância de Hamming) Por exemplo, para a construção de um hiper-cu bo de dimensão n deve-se usar 2 hiper-cubos de dimensão (n-1), como é geralmente feito para a elaboração de mapas de Karnaugh |1|.

Katseff |25| descreve um algoritmo de roteamen to de mensagens num hiper-cubo desenvolvido por Sullivan e Bashkow. Esse algoritmo é um procedimento executado a partir do nó fonte e por cada nó no caminho ao nó destino. A seguir apresenta-se o algoritmo de Sullivan e Bash kow.

ALGORITMO 1

```
if (fonte = =destino)
    Envia a mensagem ao processador local
else
    Compute endrel <- - fonte ⊕ destino
    Inicie com o bite mais significante de en drel
    Seja i o número do bit do primeiro i no en drel
    Envie a mensagem ao enlace i
```

Na figura 8 é apresentado um hiper-cubo tri-di mensional com o caminho de uma mensagem, a par tir do nó fonte 011 ao nó destino 100 ou vi ce-versa, obtida pela execução do algoritmo 1.

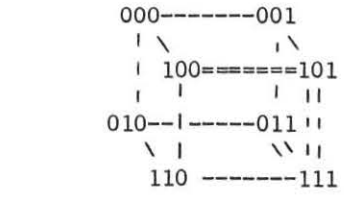

Figura 8. Roteamento de mensagem do nó 011 ao nó 100.

O caminho para roteamento de mensagens num hi per-cubo obtido pela utilização do algoritmo 1 é único. Na implementação proposta neste traba lho, usa-se uma variação desse algoritmo aqui denominado de algoritmo 2, que permite o rotea mento de mensagens por mais de um caminho. Pa ra o algoritmo 2, além das definições e propri edades do hiper-cubo usadas no algoritmo 1, de finimos o seguinte: dois nós estão no mesmo cu bo básico do hiper-cubo quando os bites i's (i>2) mais significativos dos seus endereços são iguais; denomina-se conjunto de nós comuns do cubo básico a intersecção do conjunto de nós vizinhos do nó fonte com o conjunto de nós vizinhos ao nó destino.

ALGORITMO 2

```
if (fonte = = destino)
    Envia a mensagem ao processador local
```

```
else
    Compute endrel <- fonte ⊕ destino
    if (cubo básico do fonte <> cubo básico do
        destino)
        Inicie com o bite mais significante do
        endrel
        Seja i o número do bite do primeiro  i
        no endrel
        Envie a mensagem ao enlace i

else
    Compute distância <-- número de "1" do en
    drel
    if (distância = =1)
    Envie a mensagem ao destino

    else
    if (distância = = 2)
    Compute conjunto de nós comuns
    Envie a mensagem ao nó  comum

    else
    if (distância = = 3)
    Envie a mensagem ao nó vizinho.
```

A figura 9 apresente diferentes caminhos obtidos com o algoritmo 2. Na figura 9.a apresentam-se os dois caminhos possíveis para o roteamento de mensagens entre os nós 000 e 101 (condição distância = = 2 no algoritmo 2). Na figura 9.b apresentam-se os 3 caminhos iniciais possíveis para o roteamento de mensagens entre os nós 000 e 111. Para hiper-cubos com dimensões superiores a 3, o algoritmo 2 usa o algoritmo 1 para determinação do caminho.

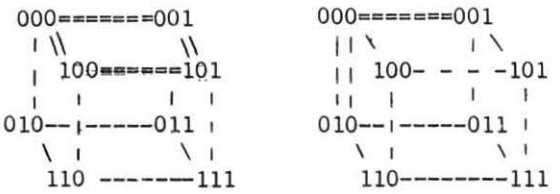

Figura 9. Roteamento de mensagens usando o algoritmo 2.