REQUISITOS DE HARDWARE PARA PROCESSAMENTO A FLUXO DE DADOS DISTRIBUÍDO *

Eduardo Marques[†]
Rosana C.M.G. Gonçalves[‡]
Claudio Kirner[§]

RESUMO

Este trabalho apresenta uma discussão geral de processamento a fluxo de dados num ambiente distribuído, proporcionado pelo Computador Paralelo Estruturado Recursivamente (CPER). Apresenta-se também os requisitos básicos de hardware envolvendo pesquisa em memória, comunicação, processamento e tolerância a falhas.

ABSTRACT

This paper shows a general discussion of aspects related to dataflow processing in a distributed environment provided by a Recursively Structured Parallel Computer (CPER). The hardware basic requirements for searching in memory, communication, processing and fault-tolerance are also presented.

*Projeto Centauro de Computação Paralela - UFSCar.

[†]Prof. Assistente ICMSC/USP; Bel. Ciência Computação (UFSCar,1985); Mestre Ciência Computação (ICMSC/USP,1988); Doutorando Eng. Elétrica (EP/USP). Arquiteturas de Computadores e Sistemas Distribuídos. ICMSC/USP, Cx. Postal 668, 13560 - São Carlos - SP. Fone: (0162) 72-6222/ r. 3808.

[‡]Bel. Ciência Computação (UNICAMP, 1987); Mestranda Ciência Computação (UFSCar). Processamento Paralelo/Arquitetura e Computação Gráfica. PPG-CC/UFSCar. Cx. Postal 384, 13560 - São Carlos - SP. Fone: (0162) 71-1100/ r. 143.

[§]Prof. Adjunto DC/UFSCar; Eng. Eletricista (EESC/USP, 1973); Mestre Eng. Eletrônica (ITA, 1978); Doutor Eng. de Sistemas e Computação (COPPE/UFRJ, 1986). Processamento Paralelo/Arquitetura e Sistemas Operacionais. DC/UFSCar, Cx. Postal 384, 13560 - São Carlos - SP. Fone: (0162) 71-1100/ r. 143.

REQUISITOS DE HARDWARE PARA PROCESSAMENTO A FLUXO DE DADOS DISTRIBUÍDO

1 Introdução

A Ciência da Computação dos anos 90 certamente será marcada pela proliferação dos supercomputadores. Tais máquinas estão sendo crescentemente desejadas nas mais diversas aplicações possíveis, desde a previsão do tempo ao cálculo estrutural de partes de um avião. Quanto a construção de supercomputadores, as perguntas são muitas: será de propósitos gerais ou de propósitos especiais?; processador vetorial ou fluxo de dados?; etc. [46 e 58]. Tratando-se de computação paralela, sua implementação tanto a nível de hardware quanto a nível de software é bastante complexa, mas já está bem equacionada na literatura [3,4,15,25 e 57].

Uma solução interessante para a concepção de uma máquina altamente paralela é o modelo de fluxo de dados distribuído. O modelo de fluxo de dados é atraente pelo fato do próprio sistema extrair o paralelismo automaticamente durante a execução, desde que haja as devidas ferramentas de conversão de um programa para um grafo fluxo de dados.A característica de distribuição, além de facilitar a tolerância a falhas no modelo, permite uma maior flexibilidade quanto ao crescimento incremental do sistema, atuando de acordo com as necessidades do usuário.

Vários esforços internacionais e nacionais têm sido feitos para a construção de uma máquina a fluxo de dados, tanto a nível de sistemas [1,6,8,10,16,21,27,42,49 e 50], quanto a nível de VLSI [28,35,39 e 40]. Dentro desse contexto, este artigo procura mostrar uma contribuição para a área, abordando o projeto de uma máquina com possibilidades de processamento a fluxo de dados distribuído [31]. Desta meneira, discute-se, neste trabalho, os requisitos de hardware necessários para sua implementação, envolvendo os aspectos de pesquisa de alta eficiência em memória; comunicação de alta velocidade; processamento de alto desempenho e tolerância a falhas.

2 Execução a Fluxo de Dados em Ambiente Distribuído

O modelo de fluxo de dados já se tornou um modelo conhecido de computação e está muito bem discutido em [22,55,59 e 60]. Portanto, a discussão a seguir estará restrita à máquina em questão e a maneira de como um programa é executado no ambiente de fluxo de dados distribuído.

2.1 Computador Paralelo Estruturado Recursivamente - CPER

A arquitetura do Computador Paralelo Estruturado Recursivamente (CPER) é apresentada na figura 1 [32]. Ela é composta por um processador hospedeiro (de alta capacidade de processamento); um subsistema de comunicação estruturado hierarquicamente (composto por barramentos paralelos, redes seriais em anel e "gateways"); e elementos de processamento (EP).

A recursividade na arquitetura surge na medida em que é utilizada uma estrutura básica, constituindo um grupo ("cluster") formado por um subsistema de comunicação, interligando N elementos de processamento. Este grupo, contendo processadores de nível 1, pode ser visto como um novo elemento de processamento (nível 2), que interligando-se com outros elementos iguais formam um novo grupo de nível mais elevado. O uso recursivo das mesmas regras de agrupamento permite a expansão do sistema ao nível desejado.

Os elementos de processamento e o processador hospedeiro encontram-se interligados de duas maneiras: através de redes em barramento e através de redes em anel. Maiores detalhes sobre esta interconexão é dado no item 3.2.1. Uma descrição da arquitetura de cada elemento de processamento é apresentada no item 3.3.

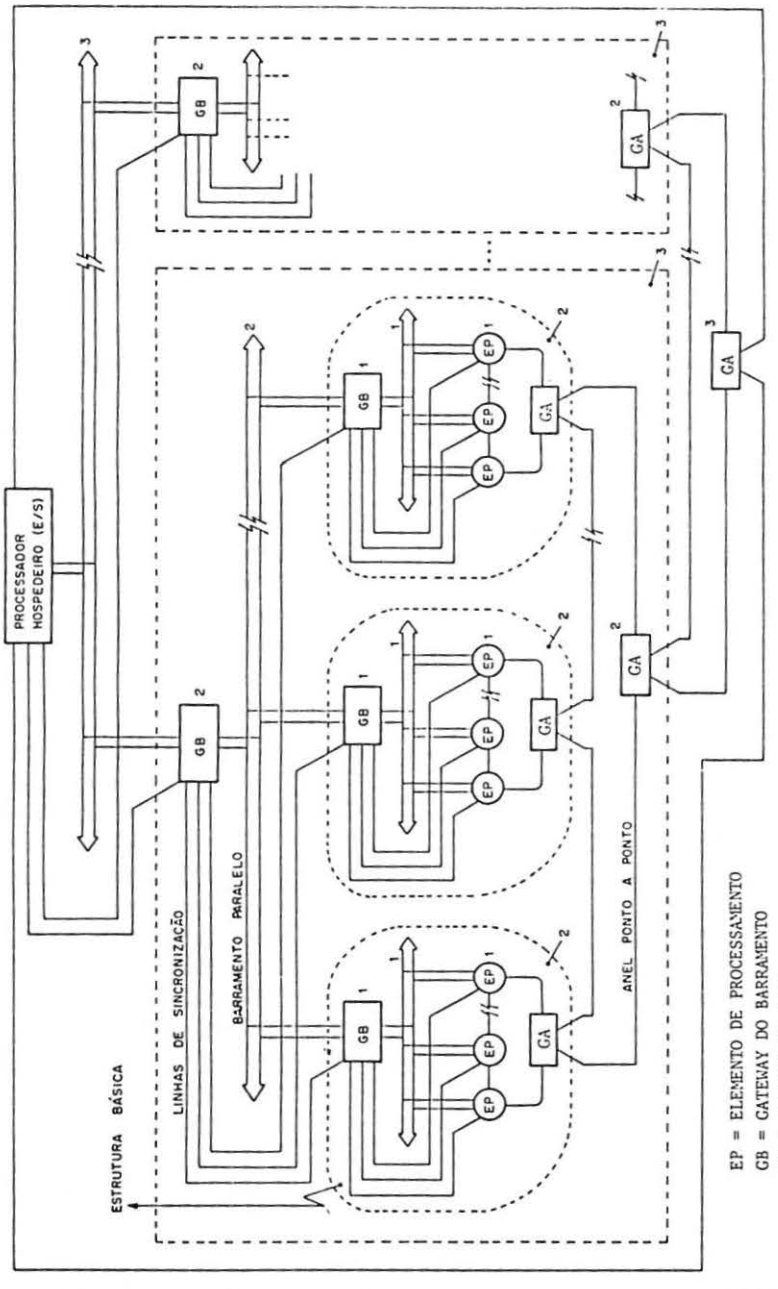

Figura 1: Arquitetura do Computador Paralelo Estruturado Recursivamente (CPER)

O CPER explora o paralelismo executando operações simultâneas por seus EP's e realizando comunicações simultâneas nos seus grupos em diversos níveis.

2.2 Execução de Programa a Fluxo de Dados

A execução de programas a fluxo de dados no CPER requer alocação de suas operações nos EP's antes de começar a execução. Subsequentemente, com a entrada dos dados, os EP's executarão as operações habilitadas e transferirão dados entre si até que o resultado final do programa seja obtido [13 e 52]. O processador hospedeiro também atuará como um outro EP especializado em entrada/saída , além de ser o encarregado de realizar as funções inicias do sistema (compilação, geração de grafos a fluxo de dados, otimização, escalonamento e carga do código nos EP's). Neste contexto, a execução de programas a fluxo de dados não depende de elementos centralizadores (memória, processador) e o fluxo de dados ocorre em um ambiente distribuído [12].

Optou-se incialmente pela utilização de um subconjunto da linguagen Pascal, como linguagem de programação [60]. Posteriormente, o subconjunto será ampliado e serão utilizadas outras linguagens.

Com um software básico apropriado, o CPER funciona como uma máquina a fluxo de dados dinâmica, utilizando-se "tagged-token" [6]. Cada dado contém um "tagged-token" associado a ele. Assim, as operações de programas reentrantes são disparadas por dados que tenham o mesmo "tagged-token" associado. Para cada execução de um programa, função, ou procedimento, é gerado um "tag", representando uma ativação. No caso de existirem "loops", implementados com operações iterativas ("While","Repeat","For"), a entrada em cada uma dessas operações gerará um novo "tag" que será associado aos parâmetros de entrada com os "tags" antigos. Para cada ciclo do "loop", o novo "tag" deverá ser ajustado para indicar a iteração. Como as operações iterativas podem ser aninhadas, é útil que o "tag" também contenha o nível de aninhamento. Ao término de cada operação iterativa, função, procedimento ou programa, a parcela do "tag" correpondente é destruída.

Depois do final da última ativação, o processo completo é concluído. Maiores detalhes sobre a execução do programa, formato dos "tags" e gabarito dos operadores podem ser encontrados em [12 e 53].

2.3 Granularidade

A granularidade no modelo a fluxo de dados pode ser classificada em fina, média e grossa [41]. A granularidade fina está num extremo onde as instruções de um grafo fluxo de dados são mapeadas praticamente uma a uma na máquina, ou seja, para cada instrução do grafo haverá uma instrução alocada a um determinado EP. Isto possibilita um alto paralelismo, mas tem a grande desvantagem de exigir muita comunicação entre os EP's, o que acaba sobrecarregando o subsistema de comunicação. Na maioria das vezes a comunicação torna-se um gargalo do sistema. No outro extremo encontra-se a granularidade grossa, onde são mapeados trechos de programa em cada EP. Esta alternativa favorece o subsistema de comunicação (pelo fato de haver pouca necessidade de comunicação entre os EP's), mas prejudica o paralelismo, uma vez que as operações concentradas num determinado EP poderiam estar também distribuídas entre os outros EP's disponíveis no sistema. No CPER utilizou-se uma solução de granularidade fina que consiste em construir grafos de fluxos de dados com um repertório de instruções de alto nível e poderosas. No entanto, uma instrução de alto nível é decomposta em uma série de instruções de máquina (instruções de baixo nível), sendo executadas no estilo von Neumann em um EP. Do ponto de vista macroscópico esta solução pode também ser classificada de granularidade fina, enquanto que do ponto de vista microscópico (a nível de execução final de hardware) é uma granularidade média.Esta alternativa compatibiliza velocidades altíssimas de processamento nos

EP's, com uma comunicação menos rápida no subsistema de comunicação. Como a tecnologia dos microprocessadores avança muito mais rápido do que a tecnologia de construção de barramentos, esta solução torna-se bastante atrativa.

3 Suporte de Hardware para Processamento a Fluxo de Dados Distribuído

Para que a máquina CPER possa apresentar o comportamento esperado, uma série de questões devem ser analisadas e respondidas.

3.1 Pesquisa de Alta Eficiência em Memória

Um dos maiores problemas no projeto de um computador fluxo de dados é a detecção da disponibilidade do dado, principalmente quando vários contextos de ativações coexistem na execução das instruções. Desta forma, é necessário a utilização de técnicas altamente eficientes de pesquisa em estruturas de armazenamento de dados (listas encadeadas e tabelas).

3.1.1 Estrutura e Funcionamento dos Operadores

No computador CPER um operador genérico com 3 entradas e 2 saídas foi definido para ser usado nos grafos a fluxo de dados (figura 2 a). Este procedimento facilita o tratamento e a manipulação dos operadores, tanto no processador hospedeiro (na fase de carga de código), quanto nos elementos de processamento (na fase de armazenamento e recuperação dos dados). Pelo fato do CPER utilizar execução a fluxo de dados dinâmica, houve a necessidade de se definir o gabarito ("template") de memória (do operador genérico) apresentado na figura 2b. Este gabarito contém os seguintes campos: um campo para a especificação da operação; três campos para os dados de entrada; dois campos para o endereço de saída; um campo indicando os campos de dados e de endereços válidos para a operação (máscara), pois existem vários tipos de operadores; e por último, um campo indicando a presença ou ausência dos dados ou endereços válidos ("status"). Como o sistema permite o uso de "tagged tokens", poderão existir várias ativações e várias iterações de cada operação. Neste caso, o gabarito de memória assume o formato de uma lista encadeada, contendo uma cabeça principal que aponta a lista de ativações sucessivas. Cada ativação, por sua vez, serve como cabeça secundária para apontar a lista de iterações sucessivas. A cabeça principal contém informações fixas, como a operação, endereços destino, e máscara de dados e endereços. A cabeça secundária contém a parcela do "tag" relativa a ativação recente e os outros "tags" anteriores. Cada iteração contém a parcela do "tag" relativa àquela iteração, o "status" do dado (ausente ou presente), e os campos de dados.

Uma operação simples, quando recebe um dado de uma única ativação, gera uma cabeça secundária e a iteração inicial, ficando como o gabarito de memória igual ao da figura 2b. Uma operação dentro de um "loop" iterativo vai adicionando novos elementos na lista encadeada apontada pela cabeça secundária, conforme vão chegando dados com novos "tags" alterados pelo campo de iteração.

A chegada de todos os dados de uma operação provoca a execução da operação e a destruição do respectivo elemento da lista encadeada; se não restar nenhum elemento ligado numa cabeça secundária, ela será destruída.

A localização de um dos vários conjuntos de dados (elementos de lista encadeada) é realizada pela pesquisa nas ativações, seguida pela pesquisa nas iterações. Quando o conjunto de dados não é encontrado, ele é criado e o dado inserido. A obtenção de um bom desempenho na execução das operações depende muito da existência de um mecanismo de busca eficiente.

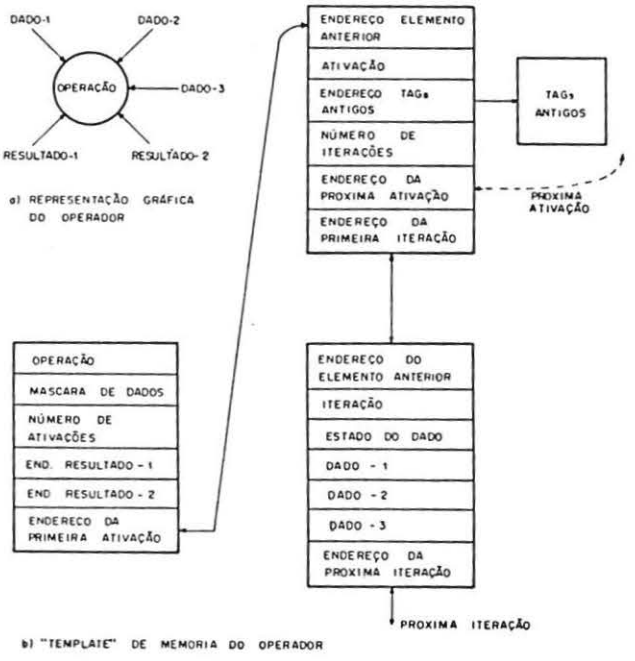

Figura 2: Estrutura Genérica do Operador

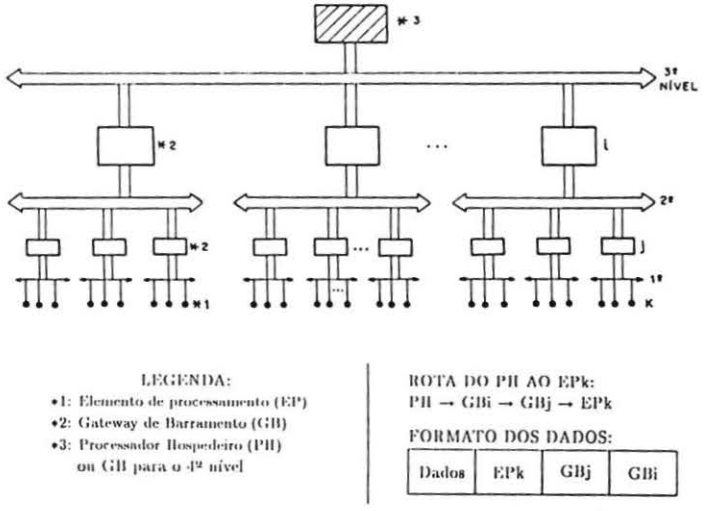

Figura 3: Estrutura Recursiva de Redes em Barramento (Modo Principal de Acesso aos EP's; comunicação Paralela)

3.1.2 Possíveis Soluções

Existem três maneiras para se fazer a busca da informação de modo eficiente. A primeira é por software, através de algoritmos de "hashing"; uma técnica de busca com superioridade e eficiência já comprovadas [36]. A segunda é por hardware, através da implementação de memória associativa com o uso de CAM's ("Content - Addressable Memories") [38]. A terceira é através da exploração de arquiteturas paralelas, construindo-se um elemento com alta capacidade de processamento para a realização da busca em paralelo numa estrutura de armazenamento distribuída.

A primeira alternativa, a qual é também considerada uma técnica de armazenamento associativo [38], é viável economicamente, porém lenta demais para os objetivos aos quais o CPER se propõe.

A terceira alternativa, apesar da facilidade de ser implementada por hardware através de uma matriz de "tranputers" [26], torna-se dispendiosa, além de exigir um gerenciamento à parte para o controle da informação distribuída nas unidades de busca, constantes em cada elemento de processamento.

A segunda alternativa, a implementação de memória associativa com o uso de CAM's, pode também ser vista como uma técnica de "hashing" implementada por hardware [38], utilizando-se circuitos integrados dedicados (as CAM's). Esta alternativa é viável em vários aspectos, porém restrita do ponto de vista de implementação. As memórias CAM's comerciais possuem alto custo (aproximadamente dez vezes mais caras que as RAM's estáticas), baixa densidade e inúmeras funções dispensáveis para o modelo a fluxo de dados (do tipo A > B ,A < B, A XOR B e etc.; onde só interessa A = B), uma vez que são construídas para propósitos gerais [11 e 38]. Além disso, estas memórias são difíceis de serem encontradas no mercado. A solução para tal impasse é a construção de uma memória associativa com o uso de memórias RAM's estáticas, acrescidas de uma lógica adicional para torná-las semelhantes a uma CAM. Deste modo, é possível implementar "hashing" por hardware, o qual tem a vantagem imediata de execução mais rápida; podendo-se inclusive associar a ele capacidade de acesso em paralelo aos bancos de memória, de tal forma a implementar eficientes mecanismos de busca.

Uma desvantagem do "hashing" vem do fato dos geradores de endereço gerarem dois endereços iguais para chaves diferentes. Esta geração de sinônimos é conhecida como colisão. Um bom gerador é complexo e toma tempo e espaço de armazenamento na tabela de "hash".

As maneiras para se resolverem as colisões não são complexas, mas novamente gasta-se tempo na busca e em memória. O gerador de endereços deve ser muito bem elaborado, pois tem influência direta no desempenho final do "hashing". Este desempenho é proporcional: ao número de colisões geradas; ao tempo ocupado na geração de endereços; e ao tempo necessário na solução dos sinônimos gerados. A literatura selecionada [17,36,38 e 54] apresenta vários métodos para tratar a colisão e suas implementações no sistema.

Recentemente, novas técnicas de busca tem sido criadas, geralmente técnicas híbridas, como a proposta por LITWIN[43], onde um método é apresentado sendo a combinação das técnicas "trie memory" e "hash coding" [38]. O inconveniente e o proibitivo deste método está na necessidade do local, onde estiver ocorrendo a busca (arquivo, memória), estar ordenado.

A nível internacional já existem diversas implementações de "hashing" por hardware, e algumas delas, específicas para o modelo de fluxo de dados [7,8,14,20,23,24 e 48]. As mais atrativas parecem ser as propostas de HIRAKI et al [23 e 24], GOTO et al [20] e DA SILVA [14]. Porém, como o gabarito do operador genérico é bastante completo, uma solução particular deverá ser eleborada (como tem mostrado as referências já citadas) e isto ainda requer mais estudos. Entretanto, alguns requisitos que o sistema de "hashing" deverá ter já é possível prever: 1)deverá existir um método rápido de difusão do argumento procurado para todas as locações de memória; 2)deverá ser feita uma comparação individual do argumento procurado com cada palavra-chave armazenada ou com um conjunto de atributos; 3)todos os dados, e outras informações associadas a ele, constantes na tabela de "hash" deverão ser manipulados (inserção /deleção) como elemen-

tos de listas encadeadas; 4)acesso em paralelo aos bancos de memória, visando reduzir o tempo de busca; 5)e por último, um bom gerador de endereços e um procedimento para manipular as colisões.

O requisto número três está relacionado ao fato de que o sistema de "hashing", além de fazer a busca da informação, cuida da inserção de novos dados na lista encadeada (caso a busca tenha uma resultado negativo) e da retirada de um elemento ou destruição da lista encadeada (caso os dados da respectiva ativação tenham sido completados). Estas ações implicam na criação/destrição dinâmica de elementos de lista encadeada e, consequentemente, seu gerenciamento. Uma maneira simples de facilitar esse gerenciamento é trabalhar com uma lista encadeada denominada "free-space". Esta lista fornece elementos livres para as outras listas, visando a inserção de novos dados, e recebe elementos desativados das mesmas, após a destruição de uma ativação ou iteração. O gerenciamento deverá cuidar de questões referentes as listas encadeadas do tipo "underflow", "overflow", "deadlock" e, eventualmente, funções de "garbage collection".

Uma consequência natural da sofisticação do sistema de "hashing" por hardware é a sua evolução para um processador associativo ou "content-address processor" [38]; vários processadores associativos já foram construídos [2,38,47,56 e 62].

3.2 Comunicação de Alta Velocidade

Como discutido no item "2.3 - Granularidade", a existência de um subsistema de comunicação de alta velocidade vem contribuir muito no desempenho da execução das operações. E para alcançar tal objetivo é necessário que várias partes do subsistema de comunicação estejam projetadas de forma harmoniosa; dentre elas: as vias de comunicação, o mecanismo de acesso às vias, o protocolo de comunicação, etc.

O subsistema de comunicação do CPER (figura 1) apresenta três elementos distintos: barramento paralelo, anel serial duplo e linhas de sincronização. Os dois primeiros elementos serão tratados separadamente no item 3.2.1. As linhas de sincronização servem para a sinalização bidirecional entre os elementos de processamento e o "gateway" de barramento de mesmo nível. Isto agiliza a comunicação hierárquica e alivia os outros elementos de comunicação em diversas situações. O subsistema de comunicação está sendo dimensionado para suportar : 32 elementos de processamento por "cluster" (31 unidades de processamento e mais um "gateway"); comunicação no barramento paralelo a uma taxa de 10M palavras/s (320 Mbits/s); e 20M bits/s no anel serial duplo. A seguir, são discutidas as partes do subsistema de comunicação citadas anteriormente.

3.2.1 Vias de Comunicação

Baseando-se na figura 1 é possível extrair duas estruturas de comunicação que coexistem no CPER (figuras 3 e 4). A primeira é uma estrutura recursiva composta por várias redes em barramento paralelo, interligadas por seus respectivos "gateways". A segunda é análoga à primeira, com a diferença de ser composta por redes em anel serial duplo.

3.2.1.1 Estrutura Recursiva de Redes em Barramento

A unidade básica que compõe a estrutura recursiva da figura 3 é uma rede em barramento paralelo curto e centralizado (primeiro nível na estrutura), cuja função é interligar os elementos de processamento entre si e ao nível superior, que eventualmente poderá ser o processador hospedeiro. A primeira versão já foi desenvolvida e obteve-se resultados encorajadores no protótipo construído [34 e 45]. Desta forma, em busca de velocidades mais altas, já está em andamento uma nova proposta [19], visando um maior paralelismo no protocolo de comunicação e atualizando o hardware com novas tecnologias de circuito integrado.

As caracteristícas do barramento (curto, paralelo e centralizado) conduzem a uma série de vantagens (alta velocidade, confiabilidade, fácil manutenção, aumento de segurança do sistema, crescimento incremental, etc.), as quais, apresentadas em [45], são essenciais para o bom funcionamento do CPER. O barramento possui cerca de 70 linhas, sendo 32 para transferência de dados, e as restantes para endereços de destino e origem, cabeçalho, operações especiais, controle, e reserva. As linhas de reserva fazem parte dos mecanismos de tolerância a falhas (item 3.4).

O projeto do "gateway" do barramento (GB) pode ser implementado basicamente por lógica de transferência de pacotes e alguns circuitos para detecção de mensagem. Sua função é detectar que existe mensagem para ele no barramento; retirar dos pacotes de dados os cabeçalhos correspondentes a seu endereçamento; e inserir a mensagem novamente no outro barramento.Sua construção é semelhante ao da interface de comunicação paralela (figura 6).

3.2.1.2 Estrutura Recursiva de Redes em Anel

A unidade básica que compõe a estrutura recursiva da figura 4 (primeiro nível na estrutura) é uma rede em anel formada por um meio de comunicação serial e bidirecional ("full-duplex"), o qual é duplicado para o aumento da confiabilidade (figura 5a), cuja função é interligar os elementos de processamento entre si e ao processador hospedeiro. A implementação desta rede será feita através da inserção de um "transputer" em cada elemento de processamento, visando a construção de uma interface de comunicação serial (figura 6), baseada no protocolo de comunicação da INMOS[26].

O projeto do "gateway" do anel (GA) pode ser implementado pela junção de dois "transputers", conforme a figura 5a, resultando num elemento com oito enlaces disponíveis. Este número de enlaces é necessário pelo fato do GA estar interligado por quatro pontos (figura 1) e as ligações serem duplicadas. Os "transputers" estão ligados de tal modo que a perda de um não isole toda uma rede conectada àquele "gateway", possibilitando o acesso normalmente aos elementos integrantes daquela rede (figura 5b).

3.2.1.3 Consequências da Sobreposição das Redes em Barramento e Anel

A sobreposição das redes em barramento e anel cria dois caminhos independentes e redundantes na comunicação processador hospedeiro/elemento de processamento e entre elementos de processamento. O que muda são apenas o meio de comunicação e a taxa de transmissão. A disponibilidade desta alternativa de comunicação dá uma flexibilidade ao sistema, pela razão de permitir: 1)aumento da velocidade no sistema, uma vez que as duas vias (serial e paralela) podem ser utilizadas simultaneamente por qualquer elemento do sistema para comunicação em paralelo; 2) uma segunda opção de comunicação para contornar situações de falhas críticas na rede em barramento; 3) monitoração do sistema através de um meio auxiliar; 4) aumento da confiabilidade da comunicação através do envio simultâneo de mensagens duplicadas a qualquer elemento do sistema. Alem disso, as semelhanças estruturais entre as duas redes favorecem a implementação das mesmas.

3.2.2 Mecanismo de Acesso às Vias de Comunicação

Considerando-se que as vias de comunicação são de uso comum entre os elementos de processamento, deve haver algum árbitro para controlar o acesso nestas vias, no caso de vários elementos de processamento desejarem utilizá-las ao mesmo tempo. Na rede em anel não há especificamente a necessidade de um árbitro, pois as ligações entre os "transputers" são do tipo ponto-a-ponto, e já existe um "handshake" definido pela INMOS. Quanto a rede em barramento, na nova versão [19], o acesso à via é proposto utilizando-se um árbitro distribuído com prioridade rotativa que decide, no caso de duas ou mais requisições, qual elemento ativo terá o direito ao acesso. Neste árbitro, os ciclos de disputa são sobrepostos aos ciclos de transmissão nos

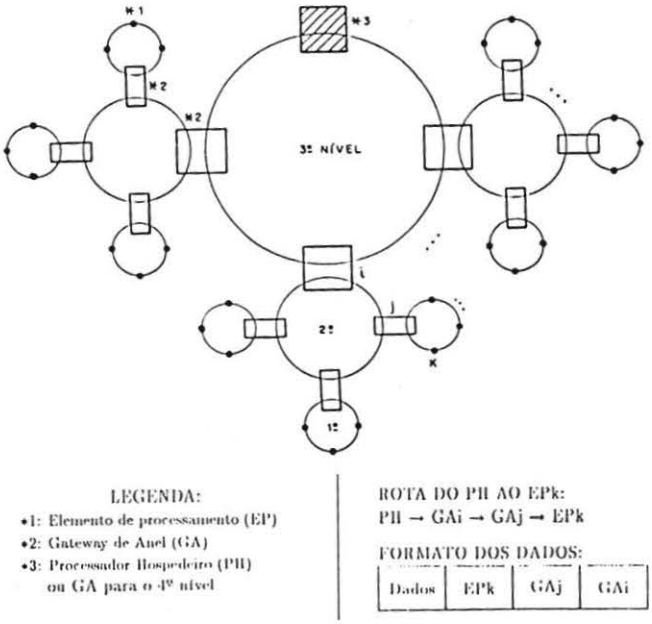

Figura 4: Estrutura Recursiva de Redes de Anel (Modo Alternativo de Acesso aos EP's; Comunicação Serial)

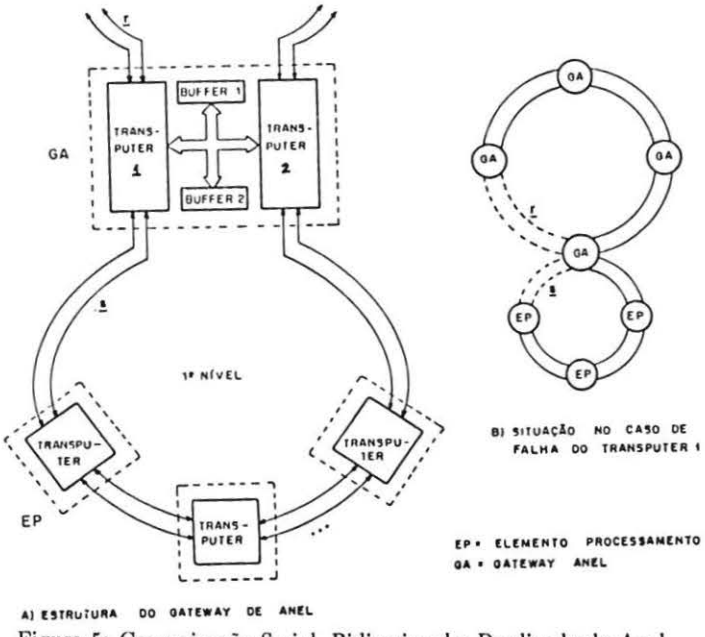

Figura 5: Comunicação Serial, Bidirecional e Duplicado do Anel

barramentos.

O árbitro opera da seguinte maneira: o dispositivo que tiver ganho o direito de acesso ao barramento, atribuirá a máxima prioridade ao seu vizinho físico e por decorrência disto, ficará com a menor prioridade. Se o vizinho não quiser transmitir, a prioridade passará ao dispositivo subsequente. A atribuição e passagem de prioridade é realizada através de uma linha denominada prioridade rotativa. Este mecanismo de acesso garante uma equidade na distribuição do barramento.

3.2.3 Particularidades da Comunicação

Para agilizar a interação do processador hospedeiro com os EP's e eventuais interações de um deles com os demais em conjunto, o subsistema de comunicação deverá possuir um mecanismo eficiente de difusão de mensagens. Isto deverá ser útil na fase de preparação do sistema e em situações que exijam a troca de mensagens de controle para superação de falhas e para outras atividades comuns a todos os EP's.

Devido a grande troca de informações entre os EP's , proporcionada pelo modelo a fluxo de dados, é conveniente a utilização de "buffers" na recepção das mensagens, de forma a não perder nenhuma mensagem enviada ao EP. Além disso, o "buffer" atua como um elemento regulador entre as diferentes taxas de chegada de mensagem e de leitura pelo EP. O "buffer" de recepção pode ser implementado rusticamente através da associação de registradores e memórias de uso comum, ou por circuitos integrados dedicados que implementam sofisticadas versões de FIFO's [5].

Por existir a possibilidade de recepção de diferentes tipos de mensagens, torna-se atrativo dispor de maneiras rápidas para extraí-las seletivamente dos "buffers", considerando-se, inclusive, a prioridade da mensagem. Isto pode ser feito de várias maneiras, mas a mais adequada parece ser a proposta por KING em [29]. Ele fornece sugestões para se construir uma FIFO com funções de "hashing", ou seja, converter uma FIFO em memória associativa. Esta proposta une duas características importantes num "buffer" e tem a vantagem de poder ser englobada como uma das funções adicionais do processador associativo discutido no item 3.1.2.

3.3 Processamento de Alto Desempenho

Conforme mencionado anteriormente, o elemento de processamento deverá ser dotado de um grande potencial em executar instruções. Sem este requisito, o CPER torna-se inviável computacionalmente. Entretanto, prover o EP com esta característica não é uma tarefa trivial, como por exemplo restringir-se apenas a quantidade de MIPS/MFLOPS executadas por um determinado microprocessador. A dificuldade começa na definição de sua eventual arquitetura. Somente com alguns estudos em simulação é possível estimar a demanda de processamento exigida por um EP. Estes estudos já iniciaram-se, porém ainda não concluídos [18]. Entretanto, algumas necessidades são claras, e baseando-se nelas, um esboço de arquitetura para o EP foi traçado (figura 6). Das partes apresentadas na arquitetura, apenas duas ainda não foram tratadas: interface de comunicação paralela e processador de trabalho.

A interface de comunicação paralela é encarregada de receber e enviar informações através da rede em barramento, e para isso necessita de circuito de controle de acesso, circuito de reconhecimento de mensagens, lógica de tolerância a falhas, lógica de transmissão, lógica de recepção, e etc. [19 e 45].

O processador de trabalho é o elemento encarregado de executar, de fato, as instruções provindas da transformação de um grafo de fluxo de dados em instruções de máquina. Escolher este processador é uma tarefa difícil e de grande responsabilidade para o futuro do projeto, como ressaltado por WILSON em [61]. Na época da família de 8 bits havia muita semelhança entre os microprocessadores; mas com a vinda das famílias de 32 e 64 bits, as diferenças aumentaram

e a complexidade também. Adicionando-se a isso, inúmeros microprocessadores são lançados no mercado mensalmente. A microeletrônica avança de tal forma que o atual de hoje pode se tornar obsoleto dentro de um ou dois anos. Dentro deste contexto, por enquanto, só existe expeculações e os dois microprocessadores mais viáveis para implementar o processador de trabalho seriam o "transputer" [26] e o i860[37]. Existe a possibilidade ainda de o processador de trabalho ser o próprio "transputer" encarregado de fazer a comunicação serial. Também não está descartada a hipótese da inclusão de um co-processador aritmético externo [9].

3.4 Tolerância a Falhas

O projeto do CPER permite que a cada nível de hierarquia introduzida na máquina, a quantidade de elementos de processamento seja multiplicada por até 31 vezes (a capacidade do barramento está dimensionada para 32 elementos, sendo um reservado para o "gateway"). Isto conduz a um projeto com possibilidades de centenas a milhares de elementos de processamento, o que resulta numa probabilidade considerável de falhas.

Por uma exigência do modelo de fluxo de dados é fácil notar que a comunicação entre os elementos de processamento será intensa, o que também resulta numa probabilidade considerável de perdas da informação.

Diante destas circunstâncias, torna-se essencial que o CPER tenha comportamento confiável e que seja tolerante a falhas. Os modos para introduzir essa tolerância a falhas dependem do ambiente (a nível de hardware ou software) e dos diversos níveis do sistema. Neste trabalho é discutido tolerância a falhas somente a nível de hardware. A tolerância a falhas a nível de software [33 e 51] complementa as decisões tomadas a nível de hardware, através de um protocolo tolerante a falhas com "backup" de dados.

3.4.1 Soluções a Nível de Hardware

A tolerância a falhas proposta no CPER atua nas vias de comunicação (rede em barramento e rede em anel) e nos elementos de processamento.

Na rede em barramento a tolerância a falhas é obtida de várias maneiras: pelo uso de linhas redundantes no barramento paralelo; pelo uso de circuitos dedicados para detecção/recuperação de falhas; pela alteração do modo de trasferência de informações (palavras,bytes); pelo uso de circuitos redundantes [44].

Na rede em anel a tolerância a falhas é obtida pela duplicação das linhas de comunicação e por um gerenciamento especial no fluxo de mensagens. O comportamento da rede já foi testado por simulação, obtendo-se bons resultados na reversão automática de rota, e na isolação e reintegração automática dos elementos [30].

Quanto ao comportamento da rede, qualquer rompimento de um enlace entre dois "gateways" ou entre dois elementos de processamento nunca os isolam, por causa da bidirecionalidade dos enlaces restantes, o que possibilita a criação de uma rota alternativa para a interligação dos mesmos novamente. Além disso, o enlace é formado por duas linhas e seu rompimento exige uma pane simultânea em ambas. Esta situação é mais comum quando o "transputer" responsável pelos enlaces falha de modo permanente, comprometendo dois enlaces. Mas neste caso, a mesma consideração apresentada anteriormente continua válida, uma vez que irá existir uma nova rota unindo as partes que o "transputer" com defeito separou. A duplicidade das linhas seriais aumenta a confiabilidade do sistema, pelo fato de possibilitar o envio de mensagens duplicadas, e permite uma segunda opção de comunicação no caso de uma falhar.

A sobreposição das redes em barramento e anel no CPER, com seus respectivos mecanismos de tolerância a falhas, propiciam um subsistema de comunicação altamente confiável.

A tolerância a falhas no elemento de processamento está mais a nível externo, cuja maior preocupação é a sua isolação (do barramento) e a sua substituição (lógica), por um outro elemento

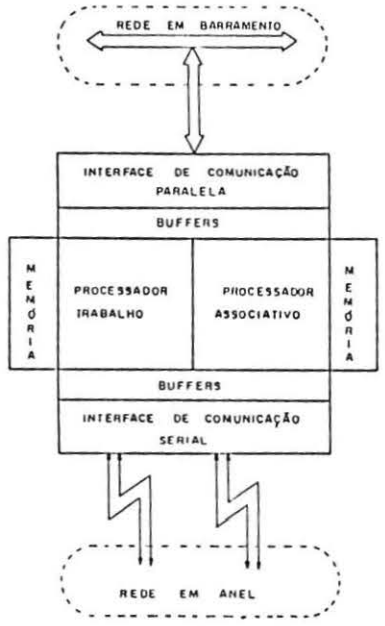

Figura 6: Estrutura de um Elemento de Processamento (EP)

capaz de assumir suas tarefas, em caso de falhas. Os mecanismos de testes, isolação e desconexão de um elemento de processamento do barramento são descritos em [44]. Suas implicações no software do sistema são, entretanto, os maiores problemas; e um tratamento adequado para isso é descrito em [51].

4 Conclusões

O trabalho fez uma discussão geral de processamento a fluxo de dados num ambiente de processamento distribuído, proporcionado pelo Computador Paralelo Estruturado Recursivamente (CPER). Nesse sentido, procurou-se ressaltar os requisitos fundamentais de hardware para suportar esse modelo de processamento.

Como pontos básicos foram tratadas as seguintes questões: pesquisa em memória através de mecanismos de hardware; compatibilização entre processamento paralelo de alto desempenho e velocidade de comunicação; e aspectos de tolerância a falhas.

O uso de técnicas de "hashing"na elaboração de um processador associativo resolve a questão da pesquisa em memória. A conversão de programas escritos com linguagem de alto nível em grafos proporciona uma solução que corresponde a granularidade fina em fluxo de dados e granularidade média a nível de execução final de hardware; esta solução permite o uso de processadores com velocidades altíssimas sem a necessidade de grandes velocidades de comunicação. A existência de redundância nos meios de comunicação e no conjunto de processadores resolve os problemas de tolerância a falhas no sistema.

Desta forma, foram destacados e discutidos os requisitos principais para que o CPER possa ser implementado com vistas a ser usado como uma máquina eficiente para execução de programas a fluxo de dados.

Referências

[1] ABRAMSON, D.A. & EGAN, G.K. - An Overview of the RMIT/CSIRO Parallel Systems Architecture Project. *The Australian Computer Journal, 20*(3): 113-121. 1988.

[2] AHUJA, S.R. & ROBERTS, C.S. - An Associative/Parallel Processor for Partial Match Retrieval Using Superimposed Codes. *Annual Symposium on Computer Architecture, 7.* USA. p. 218-227.

[3] ALMASI, G.S. & GOTTLIEB, A. *Highly Parallel Computing.* Redwood City, The Benjamin/Cummings, 1989. 520 p.

[4] AMORIM, C.L. & BARBOSA, V.C. & FERNADES, E.S.T. *Uma Introdução à Computação Paralela e Distribuída.* Campinas , IV Escola de Computação, 1988. 258 p.

[5] ANDREWS, W. - ASIC Memories: Bigger, Faster, and Customized. *Computer Design, 27*(18):44-62. 1988.

[6] ARVIND & NIKHIL, R.S. Executing a Program on the MIT Tagged-Token Dataflow Architecture. *Lecture Notes in Computer Science, 259*: 1-29. 1987.

[7] BURKOWSKI, F.J. - A Hardware Hashing Scheme in the Design of a Multiterm String Comparator. *IEEE Transactions on Computers, c-31*(9): 825-834. 1982.

[8] BURKOWSKI, F.J. - A Multi-user Data Flow Architecture. *Annual Symposium on Computer Architecture, 8.* Minneapolis, USA, 1981. p. 327-340.

[9] BURSKY, D. - Floating - Point Math Chip Delivers 200 MFLOPS Peak. *Electronic Design, 38*(4):51-55. 1990.

[10] CALUWAERTS,L.J. & DEBACKER, J. & PEPERSTRAETE, J.A. - A Data Flow Architecture with a Paged Memory System. *Annual International Symposium on Computer Architecture, 9.* USA, 1982. p. 120-127.

[11] COLOMB, R.M. - Table Searching Using a Content-Addressable Memory. *The Australian Computer Journal, 20* (3):105-112. 1988.

[12] D'AREZZO, V.M. & KIRNER, C. - Dataflow in Distributed Environment. *SCCC International Conference on Computer Science, 10.* Santiago, Chile, 1990. 10 p.

[13] D'AREZZO, V.M. & KIRNER, C. - Mapeamento de Operações em uma Máquina a Fluxo de Dados Dinâmica. *Congresso Nacional de Informática, 22.* São Paulo, SP, 1989. p. 186-194.

[14] DA SILVA, J.G.D. - *The Matching Unit of the Manchester Data-Flow Computer: a Pseudo-Associative Store with Hardware Hashing.* Manchester, University of Manchester, 1982. Ph.D. Thesis. 183 p.

[15] DECEGAMA, A.L. - *The Technology of Parallel Processing: Parallel Processing Architectures and VLSI Hardware.* Engl. Cliffs, Prentice-Hall, Vol 1, 1989. 478 p.

[16] DENNIS, J.B. & MISUNAS D.P. - A Preliminary Architecture for a Basic Data-Flow Processor. *Annual Symposium on Computer Architecture, 2.* Houston, USA, 1974. p. 126-132.

[17] ENBODY, R.J. & DU, H.C. - Dynamic Hashing Schemes. *ACM Computing Surveys, 20* (2):85-113. 1988.

[18] GONÇALVES,R.C.M.G. & KIRNER, C. - Simulação na Construção de Protótipos de Sistemas Distribuídos. *Congresso Nacional de Informática, 22.* São Paulo, SP, 1989. p.706-713.

[19] GONÇALVES, R.C.M.G. - CPER-1: *Uma Arquitetura para Exploração de Paralelismo Híbrido.* São Carlos, Universidade Federal de São Carlos, 1989. Proposta de Dissertação de Mestrado. 96 p.

[20] GOTO, E. & IDA, T. & GUNJI, T. - Parallel Hashing Algorithms. *Information Processing Letters, 6* (1):8-13. 1977.

[21] GURD, J.R. & KIRKHAM, C.C. & WATSON, I. The Manchester Prototype Dataflow Computer. *Communications of ACM, 28* (1): 34-52. 1985.

[22] GURD, J.R. et al - Fine-Grain Parallel Computing: The Dataflow Approach. *Lecture Notes in Computer Science, 272:* 82-152. 1987.

[23] HIRAKI, K. & NISHIDA, K. & SHIMADA, T. - Evaluation of Associative Memory Using Parallel Chained Hashing. *IEEE Transactions on Computers, c-33*(9):851-855. 1984.

[24] HIRAKI, K. & SHIMADA, T. & NISHIDA, K. - A Hardware Design of the Sigma-1. A Data Flow Computer for Scientific Computations. *International Conference on Parallel Processing, 1984.* Michigan, USA, 1984. p. 524-531.

[25] HWANG, K. & BRIGGS, F.A. *Computer Architecture and Parallel Processing.* New York, McGraw-Hill, 1984. 846 p.

[26] INMOS Corp. - *The Transputer Databook.* 72TRN20301. 2nd. ed., 1989. 582 p.

[27] ITO, N. et al - The Architecture and Preliminary Evaluation Results of The Experimental Parallel Inference Machine PIM-D. *Annual International Symposium on Computer Architecture, 13.* Tokyo, Japan, 1986. p. 149-156.

[28] JEFFERY, T. - The uPD7281 Processor. *Byte, 10* (12):237-246. 1985.

[29] KING,W.K. - Design of an Associative Memory. *IEEE Transactions on Computers, c-20*(6):671-674. 1971.

[30] KIRNER, C. & BAENA, W.C. - Projeto de um Ambiente Tolerante a Falhas para Implementação de Sistemas Operacionais Distribuídos. *Congresso Nacional de Informática,21.* Rio de Janeiro, RJ, 1988. p.686-695.

[31] KIRNER, C. - Ambiente para Desenvolvimento de Computadores Paralelos. *JAIIO, 19, CLAIO, 5,* Buenos Aires, Argentina, 1990. 13 p.

[32] KIRNER, C. - Design of a Recursively Structured Parallel Computer. *Annual Computer Science Conference, 17.* Louisville, USA, 1989.

[33] KIRNER, C. - Projeto Centauro de Computação Paralela: Uma Alternativa para Processamento de Alta Velocidade. Submetido ao III SBAC-PP, Rio de Janeiro, RJ, PUC/RJ, nov. 1990.

[34] KIRNER, C. & MARQUES, E. - Design of a Distributed System Support Based on a Centralized Parallel Bus. *ACM Computer Architecture News, 14* (4): 15-26. 1986.

[35] KISHI, M. & YASUHARA, H. & KAWAMURA, Y. - DDDP: A Distributed Data Driven Processor. *Annual International Symposium on Computer Architecture, 10.* Stockholm, Sweden, 1983. p. 236-242.

[36] KNUTH, D.E. - *The Art of Computer Programming.* Reading, Addison-Wesley, Vol. 3, 1973. 722 p.

[37] KOHN,L. & MARGULIS, N. - Introducing the Intel i860 64-bit Microprocessor. *IEEE Micro, 9*(4): 15-30. 1989.

[38] KOHONEN, T. *Content Addressable Memories.* Berlin, Springer-Verlag, 2 nd. Ed., 1987. 388 p.

[39] KOMORI, S. et al - The Data-Driven Microprocessor. *IEEE Micro, 9*(3):45-59. 1989.

[40] KRONLÖF, K. - Execution Control and Memory Management of a Data Flow Signal Processor. *Annual International Symposium on Computer Architecture, 10.* Stockholm, Sweden, 1983. p. 230-235.

[41] KRVATRACHUE, B. & LEWIS, T. - Grain Size Determination for Parallel Processing. *IEEE Software, 5*(1):23-32. 1988.

[42] LEMOS, F.E. & RUGGIERO, W.V. - Um Processador de Propósito Geral Dirigido pelo Fluxo de Dados. *Congresso da Sociedade Brasileira de Computação, 6.* Recife, PE, 1986. p. 385-396.

[43] LITWIN, W. & SAGIV, Y. & VIDYASANKAR, K. - Concurrency and Trie Hashing. *Acta Informática, 26* (7):597-614. 1989.

[44] MARQUES, E. & KIRNER, C. & OBAC RODA, V. - Mecanismos de Tolerância a Falhas num Subsistema de Comunicação Baseado em Barramento Paralelo Centralizado. *Simpósio em Sistemas de Computadores Tolerantes a Falhas, 2.* Campinas, SP, UNICAMP, 1987. p. 131-149.

[45] MARQUES, E. - *Projeto de uma Rede Local de Computadores de Alta Velocidade.* São Carlos, Universidade de São Paulo, ICMSC, 1988. Dissertação de Mestrado. 111 p.

[46] NORRIE, C. - Supercomputers for Superproblems: An Architectural Introduction. *Computer 17*,(3): 62-74. 1984.

[47] PARHAMI, B. - Associative Memories and Processors: An Overview and Selected Bibliography. *Proceedings of the IEEE, 61*(6):722-730. 1973.

[48] RAMAMOHANARAO, K. & SACKS-DAVIS, R. - Hardware Address Translation for Machines with a Large Virtual Memory. *Information Processing Letters, 13*(1):23-29. 1981.

[49] SHIPPEN, G.B. & ARCHIBALD, J.K. - A Tagged Token Dataflow Machine For Computing Small, Iterative Algorithms. *Computer Architecture News, 15*(6):9-18. 1987.

[50] SHIMADA, T. et al - Evaluation of a Prototype Data Flow Processor of the Sigma-1 for Scientific Computations. *Annual International Symposium on Computer Architecture, 13.* Tokyo, Japan, 1986. p. 226-234.

[51] SILVA, J.L. & KIRNER,C. - Fault-Tolerant Dataflow Processing in a Parallel Computer. SCCC Internatinal Conference on Computer Science, *10.* Santiago, Chile, 1990. 10 p.

[52] SILVA, J.L. & KIRNER, C. - Suporte para Programação "Dataflow" em um Computador Paralelo. *Congresso Nacional de Informática, 22.* São Paulo, SP, 1989. p. 135-143.

[53] SILVA, J.L. & KIRNER, C. - Development of the Basic Software of a Tagged-Token Dataflow Machine. *SCCC International Conference on Computer Science, 9.* Santiago, Chile, 1989. p. 217-229.

[54] SNADER, J.C. - Look it up Faster with Hashing. *Byte, 12* (1): 129-144. 1987.

[55] SRINI, V.P. An Architectural Comparison of Dataflow Systems. *Computer,19* (3): 68-88. 1986.

[56] THURBER, K.J. & WALD, L.D. - Associative and Parallel Processors. ACM Computing Surveys, 7(4):215-225. 1975.

[57] TRELEAVEN, P.C. - Parallel Architecture Overview. *Parallel Computing, 8* (1-3): 59-70. 1988.

[58] TRELEAVEN, P.C. & LIMA, I.G. - Future Computers: Logic, Data Flow,..., Control Flow? *Computer,17* (3):47-58. 1984.

[59] TRELEAVEN, P.C. & BROWNBRIDGE, D.R. & HOPKINS, R.P. - Data-Driven and Demand-Driven Computer Architecture. *ACM Computing Surveys, 15* (1): 93-143. 1982.

[60] VEEN, A.H. - Dataflow Machine Architecture. *ACM Computing Surveys, 18* (4): 365-396. 1986.

[61] WILSON, R. - Choosing a Microprocessor: Designers Take Many Paths to the Best Solution. *Computer Design, 27*(4):59-73. 1988.

[62] YAU, S.S. & FUNG, H.S. - Associative Processor Architecture - A Survey. *ACM Computing Surveys, 9*(1): 3-27. 1977.