RELATóRIO TÉCNICO

UM ESTUDO SOBRE REDES DE INTERCONEXÃO E A SUA UTILIZAÇÃO NO PROJETO MULTIPLUS

Adriano J. O. Cruz
Gerson Bronstein

NCE-12/90
Abril/90

Universidade Federal do Rio de Janeiro
Núcleo de Computação Eletrônica
Caixa Postal 2324
20001 - Rio de Janeiro - RJ
BRASIL

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
NÚCLEO DE COMPUTAÇÃO ELETRÔNICA

UM ESTUDO SOBRE REDES DE INTERCONEXÃO E A SUA UTILIZAÇÃO NO PROJETO MULTIPLUS

Adriano J. O. Cruz

Gérson Bronstein

NCE - 12/90
abril 1990

UM ESTUDO SOBRE REDES DE INTERCONEXÃO E A SUA UTILIZAÇÃO NO PROJETO MULTIPLUS

RESUMO

Este trabalho apresenta um estudo geral sobre redes de interconexão e algumas características particulares ligadas ao projeto Multiplus. Nos 2 primeiros capítulos é feita uma revisão (definições, classificações, etc.) do assunto. No capítulo 3, é vista com mais detalhes a estrutura interna de uma chave. No capítulo seguinte são discutidos alguns dos principais problemas relacionados a redes de interconexão, dentro do contexto global de uma máquina paralela. Em seguida, são apresentadas algumas arquiteturas paralelas existentes e suas respectivas redes e, por fim, são apresentadas e analisadas algumas características particulares do projeto Multiplus, relacionadas a redes de interconexão.

A STUDY ON INTERCONNECTION NETWORKS AND ITS UTILIZATION IN THE MULTIPLUS PROJECT

ABSTRACT

This paper presents a general study on interconnection networks and some particular aspects related to the Multiplus project. The first two chapters are devoted to a review of the subject related matters (definitions, classifications, etc.). In chapter 3, the internal structure of a switch is discussed in more details. In the next chapter some of the major problems related to interconnection networks are focused under the context of parallel machines. Following, a brief analysis of some existing parallel architectures and their respective interconnection networks is given. Finally, some specific characteristics of the Multiplus project related to interconnection networks are presented and analyzed.

1 - INTRODUÇÃO

A demanda por computadores de alto desempenho, ou supercomputadores, tem aumentado muito e, hoje em dia, estas máquinas encontram aplicação nas mais diversas áreas, como análise estrutural, meteorologia, prospecção de petróleo, etc. Sem a ajuda destes supercomputadores, muitos dos desafios atuais da humanidade não poderiam ser resolvidos num período razoável de tempo. Existem duas formas distintas de se aumentar a performance destas máquinas: aumentando-se a velocidade dos dispositivos e componentes ou implementando-se arquiteturas mais avançadas e eficientes. No entanto, os limites impostos pelas leis da física no desenvolvimento de dispositivos semicondutores indicam que os esforços para o aumento da performance dos computadores estarão concentrados no desenvolvimento de novas arquiteturas e na exploração do paralelismo das aplicações [Mena], identificando dentro de uma aplicação trechos que podem ser executados simultâneamente.

Para que se possa explorar o paralelismo de forma eficiente, é necessário que as diversas partes que estão sendo processadas possam se comunicar de forma eficiente. Em muitos problemas, a complexidade do algoritmo é dominada pelo tempo gasto em comunicação. Por exemplo, em algoritmos de ordenação, cada elemento a ser ordenado deve receber informações sobre todos os elementos da estrutura de dados original. A rede de interconexão é a responsável por toda a parte de comunicação entre processadores, sendo o desempenho global da máquina fortemente dependente do desempenho da rede. A partir daí, pode-se perceber a importância que as redes de interconexão tem dentro do contexto global de uma máquina paralela.

1.1 - CARACTERIZAÇÃO DE REDES DE INTERCONEXÃO

No momento da escolha da arquitetura de uma rede de interconexão, podem ser identificados 6 pontos principais. São eles: o modo de operação, a estratégia de controle, o modo de chaveamento, os parâmetros de desempenho, a topologia da rede e a capacidade de realizar as permutações desejadas.

1.1.1 - MODO DE OPERAÇÃO

Dois modos de operação podem ser identificados: síncrono e assíncrono. No modo síncrono, os pedidos de comunicação chegam sempre no mesmo instante nas portas de entrada e a rede opera sempre de forma síncrona. O modo síncrono é utilizado em todas as máquinas SIMD existentes. No modo assíncrono, os pedidos de comunicação chegam de forma aleatória. As máquinas MIMD usam, em geral, este modo de operação.

1.1.2 - ESTRATÉGIA DE CONTROLE

Uma rede de interconexão é constituida de elementos comutadores (chaves) e das ligações entre eles. As ligações origem/destino são realizadas através de configurações apropriadas das chaves. A decisão de qual configuração assumir pode ser tomada por um controlador central para todas as chaves (controle centralizado) ou por cada chave individualmente (controle distribuído). A maioria das máquinas SIMD adotam o controle centralizado. As redes não-bloqueáveis com reconfiguração (ítem 1.1.5.3) também adotam o controle centralizado.

1.1.3 - PARÂMETROS DE DESEMPENHO

Para que se possa avaliar e comparar o desempenho de redes de interconexão, são definidos alguns parâmetros:

a) Largura de Banda (BW). é definida como o número de pedidos aceitos pela rede por unidade de tempo.

b) Throughput (Th). é definido como o número de pacotes que atravessam a rede por unidade de tempo [Dias81][Bhuy89].

c) Atraso (D). é definido como o tempo necessário para um pacote atravessar a rede. Normalmente, se utiliza o atraso médio.

d) Custo de *Hardware* (O(x)). Fornece uma medida da quantidade de *hardware* utilizada como função da variável x.

1.1.4 - MODO DE CHAVEAMENTO

Os dois modos de chaveamento utilizados são o chaveamento por circuitos e o chaveamento por pacotes. No chaveamento por circuitos, um caminho físico (circuito) é estabelecido entre a fonte e o destino e este caminho só é desfeito quando a mensagem é toda transmitida. No chaveamento por pacotes, os dados são divididos em pacotes e roteados através da rede, sem que seja necessário se estabelecer um caminho físico. Em geral, o chaveamento por circuitos é utilizado quando se tem poucos acessos com grande quantidade de dados cada um. Já o chaveamento por pacotes é mais utilizado quando se tem um grande número de mensagens pequenas.

1.1.5 - TOPOLOGIA DA REDE

A topologia da rede indica de que forma as chaves estão interconectadas. A topologia pode ser estática ou dinâmica. Nas topologias estáticas não existem elementos comutadores e as conexões entre os elementos processadores são fixas e conhecidas *a priori* (fig. 1.1). Nas topologias dinâmicas, as conexões entre os elementos processadores podem ser reconfiguradas em tempo de execução, de acordo com o estado dos elementos comutadores (fig. 1.2). As topologias dinâmicas ainda podem ser divididas em 2 categorias: mono-estágio e multi-estágio. Nas topologias mono-estágio, as mensagens são roteadas várias vezes através do único estágio da rede, até alcançarem os seus destinos. Nas topologias multi-estágio, as mensagens são roteadas uma única vez através dos n estágios da rede. Aparentemente, as redes mono-estágio são equivalentes, em termos de atraso médio, às redes multi-estágio, porém com um custo de *hardware* menor ($O(n)$ contra $O(n \log_2 n)$). No entanto, a largura de banda é $\log_2 n$ vezes menor para as redes mono-estágio porque elas não suportam *pipelining*. Todas as $\log_2 n$ passagens de um grupo de mensagens devem ser completadas antes que outro grupo possa ser enviado [Alma89].

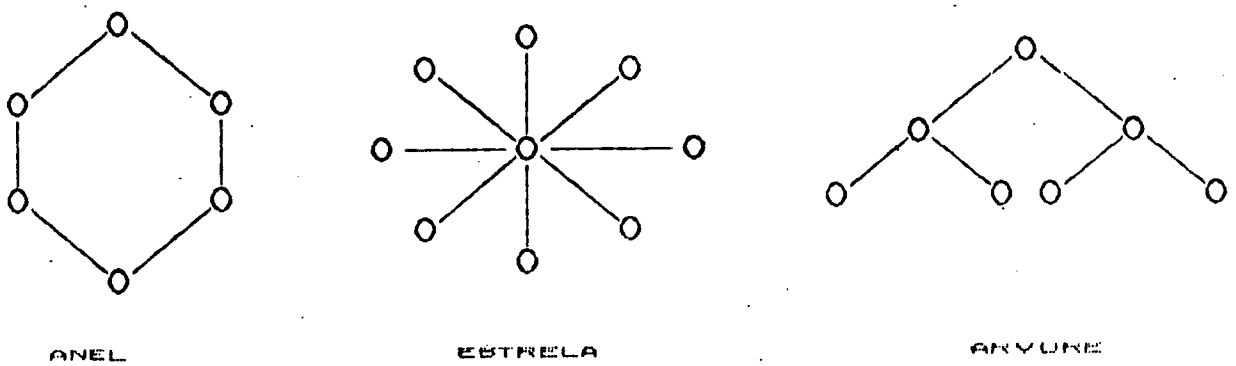

fig. 1.1 - Topologias Estáticas

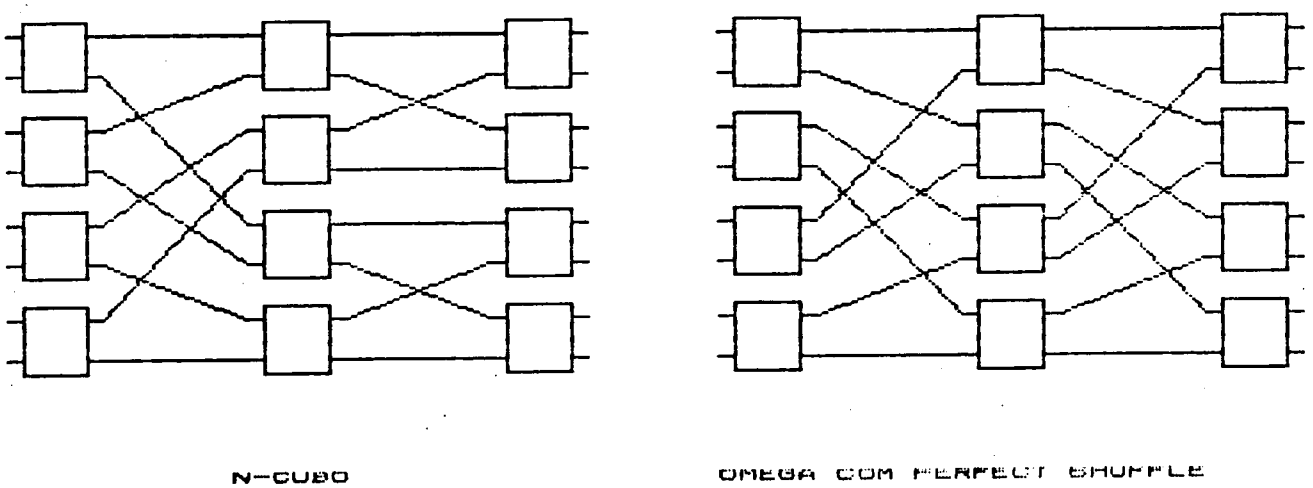

fig 1.2 - Topologias Dinâmicas Multi-estágio

1.1.6 - REALIZAÇÃO DE PERMUTAÇÕES

Outro fator importante para a caracterização das redes de interconexão, é a sua capacidade de realizar as permutações desejadas sem que haja nenhum conflito (*blocking*) nas chaves. Pode-se identificar então, 3 categorias de redes: bloqueáveis (*blocking*), não-bloqueáveis (*non-blocking*) e não-bloqueáveis com reconfiguração (*rearrangeable non-blocking*).

1.1.6.1 - BLOQUEÁVEIS

Estas redes permitem que apenas um subconjunto das permutações possíveis seja realizado sem bloqueio e possuem um único caminho para cada par origem/destino. O conflito se estabelece quando, em

uma determinada chave, ambas as entradas necessitam rotear as suas mensagens para a mesma saída. Em geral, estas redes possuem um custo $O(\frac{n \log n}{2})$. Ex.: redes omega, delta e baseline (fig. 1.3).

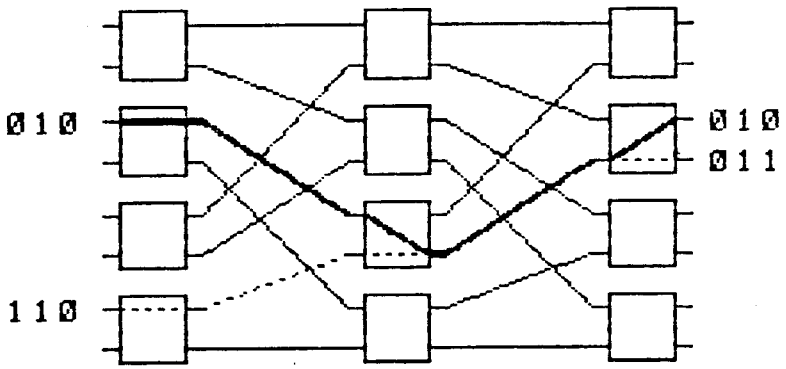

fig. 1.3 - Redes Bloqueáveis

1.1.6.2 - NÃO-BLOQUEÁVEIS

Permitem quaisquer permutações, desde que as saídas estejam livres, ou seja, não haja 2 mensagens ao mesmo tempo para o mesmo destino. Em geral, possuem um custo de *hardware* $O(n^2)$, o que torna a sua utilização proibitiva para n grande (p. ex., para n = 16 uma rede bloqueável apresenta um custo = 32 enquanto uma rede não-bloqueável apresenta um custo = 256). Ex.: crossbar (fig 1.4).

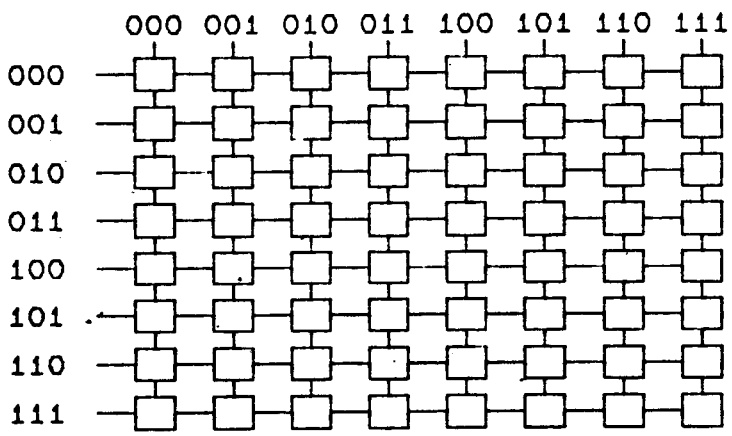

fig 1.4 - Redes Não-Bloqueáveis

1.1.6.3 - NAO-BLOQUEÁVEIS COM RECONFIGURAÇÃO

Permitem quaisquer permutações, pois possuem mais de um caminho

possível para cada par origem/destino. Estes caminhos redundantes são obtidos com a colocação de estágios extras na rede. Para que a alocação de caminhos não conflitantes seja possível, é necessário que todos os pares origem/destino sejam conhecidos antes das mensagens serem transmitidas através da rede. Portanto, é recomendável que este tipo de rede opere de modo síncrono. Ex.: rede de Benes (fig. 1.5).

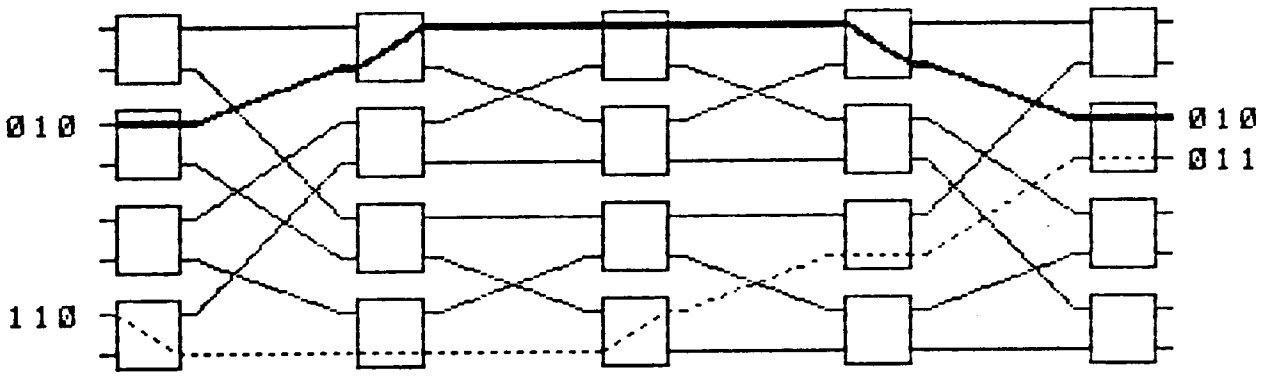

fig. 1.5 - Redes Não-Bloqueáveis com Reconfiguração

1.2 - ALGORITMOS DE ROTEAMENTO

Para que as mensagens sejam roteadas adequadamente através da rede, é necessário que se saiba quais os estados que as chaves devem assumir. A maneira mais comum de se fazer isto, seja o controle centralizado ou distribuido, é colocar junto com a mensagem uma identificação do destino (*tag*). Esta identificação deve possuir tantos *bits* quantos forem os estágios da rede. Cada *bit* controla um determinado estágio da rede. Em geral, o controle é feito da seguinte forma: *bit* = 0 ⟶ conecta a entrada à saída 0; *bit* = 1 ⟶ conecta a entrada à saída 1. Portanto, pode-se observar que haverá conflito em uma determinada chave se os 2 *bits* de controle (entradas 0 e 1) forem iguais. As redes não-bloqueáveis com reconfiguração são capazes de evitar conflitos desde que o conjunto de pares origem/destino seja conhecido *a priori*.

2 - CARACTERÍSTICAS TOPOLÓGICAS

2.1 - MODULARIDADE

Modularidade é uma característica apresentada por algumas topologias com relação à construção da rede. Nestas redes podem ser identificados módulos. Esta característica é bastante desejável quando se tem máquinas configuráveis. Quando se deseja expandir uma rede modular de 2^n portas para 2^{n+1} portas, é necessário apenas repetir a estrutura existente (módulo) e acrescentar mais um nível de chaves. Apresentam esta característica as redes n-cubo, n-cubo invertido, baseline, etc. A rede omega com *perfect shuffle* não é modular.

2.2 - SEPARABILIDADE

A separabilidade é a capacidade de uma rede poder ser dividida em sub-redes de tamanhos diferentes, cada sub-rede possuindo as mesmas características da rede original. A vantagem das redes separáveis é que o tráfego de uma sub-rede não interfere no tráfego das demais sub-redes. Com isso, pode-se alocar processos distintos à sub-redes distintas sem que haja o risco de conflito entre processos. A separação de redes é feita agrupando-se os processadores segundo os seus endereços. No exemplo a seguir (fig. 2.1) a rede original foi particionada em duas sub-redes das 3 formas possíveis: *bit* alto igual ({000 001 010 011} e {100 101 110 111}), *bit* do meio igual ({000 001 100 101} e {010 011 110 111}) e *bit* baixo igual ({000 010 100 110} e {001 011 101 111}). Observa-se que, em cada caso, no único estágio onde poderia haver conflito, a configuração da chave já está definida segundo o *bit* escolhido para o particionamento. As redes n-cubo, n-cubo invertido, baseline e omega com *perfect shuffle* são separáveis (fig. 2.1).

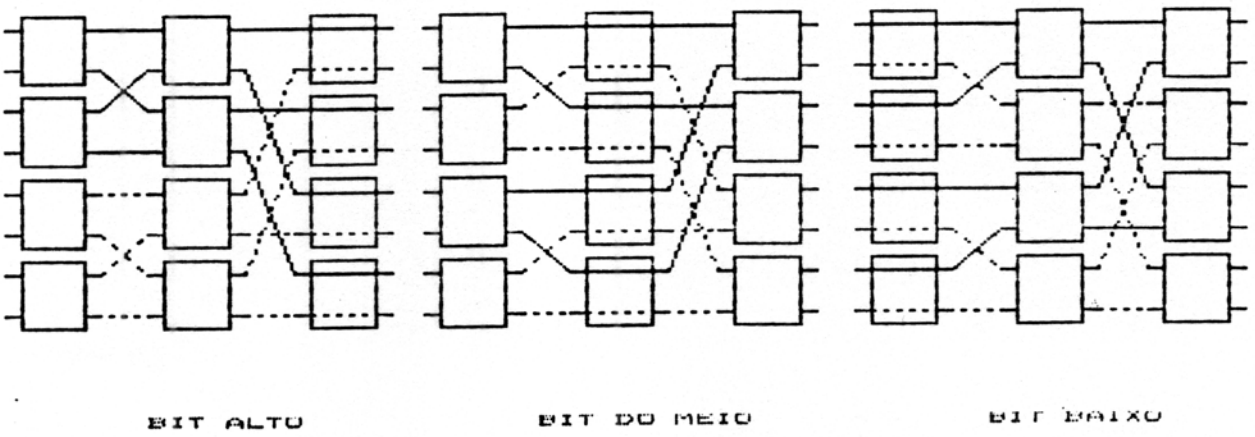

fig 2.1 - Separabilidade de Redes (N-Cubo Invertido)

A estrutura interna da chave desempenha um papel fundamental no desempenho final da rede. Dependendo do tipo de chaveamento escolhido, do tamanho da mensagem e das funções que a chave deve realizar, pode-se chegar a diversas estruturas O chaveamento por circuitos, por exemplo, é incompatível com *buffers* na chave, já que um caminho físico deve ser estabelecido entre a origem e o destino. Porém, a colocação de *buffers* aumenta o *throughput* da rede. No caso de chaveamento por pacotes, o tamanho do pacote é importante. Quanto maior a largura (em *bits*) do pacote, menor é o número de pacotes em que se precisa dividir a mensagem e, consequentemente, menor é o tempo médio de transmissão das mensagens através da rede. Por outro lado, quanto maior a largura do pacote, maior é o custo do *hardware* da chave (em termos de área de silício) pois os elementos internos da chave devem ser maiores (em *bits*) para suportar um pacote maior E ao se pensar em uma futura implementação em VLSI, o custo pode ser fundamental As redes devem ser, em geral, bidirecionais para que os pedidos sejam respondidos. Pode-se ter, então, uma rede fisicamente bidirecional (chave bidirecional) ou 2 redes unidirecionais: uma de pedidos e outra de respostas. A seguir, será dada uma descrição mais detalhada de cada um destes tópicos.

3.1 - CONTROLE DA COMUNICAÇÃO

Para a transmissão de mensagens entre as chaves, é necessário que haja um protocolo de comunicação. Este protocolo evita que mensagens sejam enviadas quando a chave destino não puder recebe-las. Uma possibilidade é a utilização de comunicação serial, semelhante à utilizada por microcomputadores. Neste caso, as informações referentes à sincronização são enviadas serialmente junto com os *bits* de dados (*start* e *stop bits*). Este esquema, apesar de simples e econômico (apenas 1 fio para dados e controle), apresenta desvantagens. O acréscimo de *bits* de controle na mensagem aumenta o seu tamanho e, consequentemente, o tempo de transmissão através da rede. E como a chave-origem não tem condições de saber, *a priori*, se a mensagem será aceita pela chave-destino, deve ser estabelecido algum esquema de confirmação de recebimento de

mensagem. Isto pode aumentar ainda mais o *overhead* da comunicação. Uma outra possibilidade é a utilização de um protocolo com sinalização independente. Pode-se implementar um esquema simples e eficiente utilizando 2 sinais para controle e sincronização, semelhante ao padrão CENTRONICS de comunicação paralela. Neste caso, a chave-destino informa, através do sinal BUSY, se pode ou não receber dados. E a chave-origem informa que está enviando dados através do sinal STROBE. Esta solução não possui o *overhead* da solução anterior pois a sincronização é feita em paralelo com a transmissão dos dados. Além disto, este protocolo permite que as chaves operem assincronamente.

3.2 - A ESTRUTURA INTERNA DA CHAVE

A estrutura interna da chave desempenha um papel importante no desempenho final de uma rede. Dentro deste tópico será dada uma atenção maior ao problema de alocação de *buffers* internos para o armazenamento temporário de mensagens.

A função da chave é receber as mensagens que chegam e roteá-las para as saídas correspondentes. Se apenas uma mensagem chega por vez (não há mensagens simultâneas), não haverá conflito, logo as mensagens serão roteadas com atraso mínimo. No entanto, quando 2 mensagens destinadas a mesma saída chegam ao mesmo tempo, uma delas tem que esperar. Portanto, as chaves devem possuir algum tipo de armazenamento interno. O *throughput* máximo que uma rede pode atingir depende diretamente de quão eficiente é a política de armazenamento das chaves. Uma chave ideal deve ter um espaço de armazenamento infinito, mas as mensagens só devem ficar armazenadas pelo tempo necessário a liberação da porta de saída correspondente. As chaves reais tem *buffers* finitos e uma largura de banda também finita. Isto pode resultar em conflitos causados por tentativas de acessos simultâneos a *buffers* compartilhados e por mensagens que não podem ser armazenadas por falta de espaço. Outro problema que deve ser evitado é que, dentro de uma mesma chave, mensagens destinadas a uma saída que esteja livre esperem atrás de mensagens destinadas a portas ocupadas (interferência entre as saídas). A seguir, serão vistas algumas estruturas de chaves e serão discutidas as suas principais vantagens e desvantagens.

3.2.1 - CHAVE SEM BUFFER

A opção mais imediata é a utilização de chaves sem *buffers*. A chave seria constituída apenas por uma chave *crossbar* e lógica de controle (fig. 3.1). Uma mensagem que tivesse o seu destino bloqueado, seria bloqueada ou descartada. A não utilização de *buffers* só é justificável quando se usa chaveamento por circuitos. No chaveamento por pacotes, a presença de *buffers* é indispensável para o aumento do *throughput* da rede. Além do mais, uma rede constituída por chaves sem *buffers* teria muito mais problemas de congestionamento, pois uma saída bloqueada refletiria imediatamente nos estágios anteriores. Diversos trabalhos analisam o desempenho de redes com *buffers* [Dias81][Tami88][Jump81] e a seguir serão mostradas algumas alternativas de alocação interna dos *buffers*.

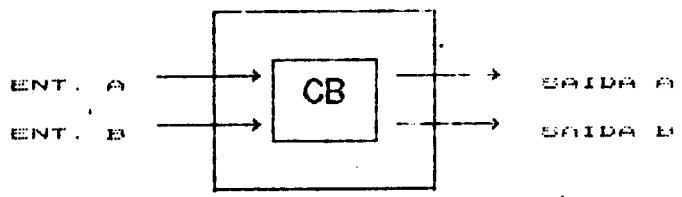

fig. 3.1 - Chave sem *Buffer*

3.2.2 - CHAVE COM BUFFER CENTRAL

Intuitivamente o compartilhamento completo do *buffer* parece ser mais eficiente do que dividi-lo entre as portas de entrada (fig. 3.2a). Porém, o *buffer* centralizado possui desvantagens tanto em desempenho, quanto em implementação. Estudos [Tami88] mostraram que, com o compartilhamento completo, uma única porta de saída congestionada pode bloquear todo o *buffer*, impedindo qualquer comunicação através da chave (fig. 3.2b). Este efeito pode se extender para os estágios anteriores, causando um colapso em toda a rede. A implementação de um *buffer* centralizado é complexa pois, para se atingir uma performance satisfatória, é necessário que ambas as portas de entrada da chave possam escrever no *buffer* ao mesmo tempo. Por causa disto, a largura de banda mínima necessária para o *buffer* deve ser igual à soma das larguras de banda das entradas. A utilização de registros multi-porta parece ser uma

solução, porém eles possuem um custo muito alto (em termos de área). Além do mais, se houver pacotes de tamanho variável, torna-se difícil a implementação de um controle que possa, rapidamente, tomar decisões com relação a alocação de espaço visando minimizar a fragmentação.

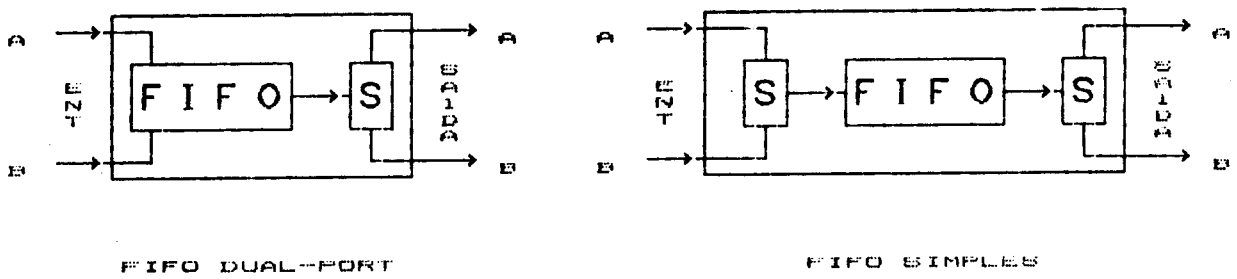

fig 3.2a - Chave com *Buffer* central

fig 3.2b - Congestionamento da Chave

3.2.3 - CHAVE COM BUFFER NA SAÍDA

A próxima opção seria colocar o *buffer* na saída (fig. 3.3). Segundo Tamir e Frazier [Tami88], o tamanho médio dos *buffers* colocados na saída é sempre menor que dos *buffers* colocados na entrada. A razão para isto é que não é necessário reservar espaço para mensagens destinadas à saídas livres, bloqueadas por mensagens destinadas à saídas ocupadas. O problema de se implementar *buffers* na saída é que a chave deve operar a uma velocidade mínima igual a soma das velocidades das portas de entrada. Ou então os *buffers* devem ser multi-portas para poder suportar escritas simultâneas. A implementação de *buffers* multi-portas aumenta o seu tamanho e diminui o seu desempenho. Esta implementação ainda possui o mesmo problema de alocação de espaço que no caso anterior.

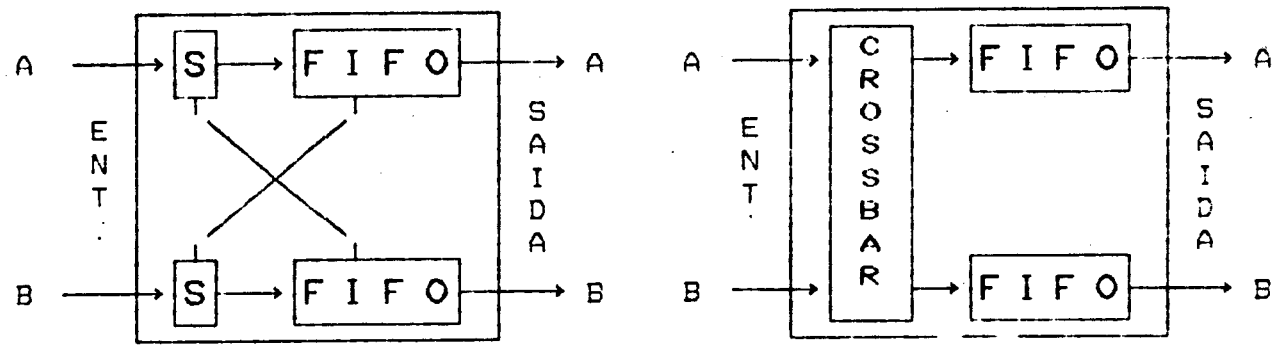

fig 3.3 - Chave com *Buffer* na Saída

3.2.4 - CHAVE COM BUFFER SIMPLES NA ENTRADA

A vantagem de se utilizar *buffers* na entrada é que apenas uma mensagem chega por vez em cada porta de entrada, portanto o *buffer* necessita de apenas uma porta de escrita (fig. 3.4a). Além do mais, se for utilizado um *buffer* do tipo FIFO, o tratamento de pacotes de tamanho variável se torna bastante simples. O problema das FIFO's é que uma única mensagem no topo do *buffer* destinada a uma porta bloqueada pode bloquear todas as outras mensagens, mesmo que seus destinos estejam livres (fig. 3.4b).

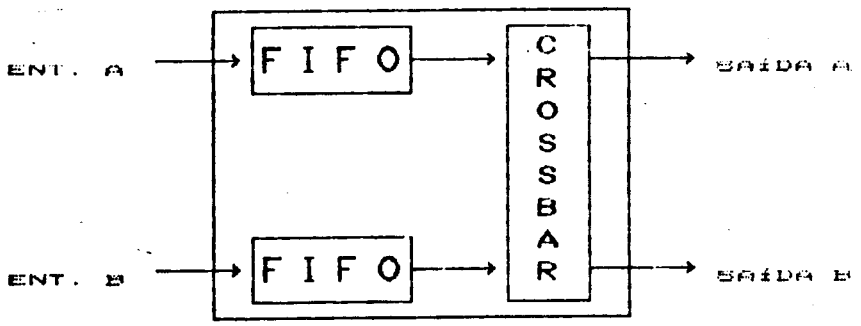

fig. 3.4a - Chave com *Buffer* Simples na Entrada

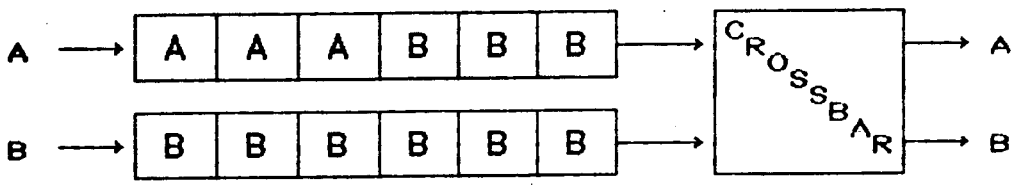

fig 3.4b - Interferência Entre as Saídas

3.2.5 - CHAVE COM BUFFER SEPARADO NA ENTRADA (CROSSBAR)

No ítem anterior, um problema sério era a possibilidade das portas de saída permanecerem inativas, mesmo havendo mensagens a elas destinadas. De forma a utilizar as portas de saída de um modo mais eficiente, as mensagens devem ser separadas de acordo com os seus destinos. Uma maneira de se fazer isto é implementar FIFO's separadas em cada entrada, cada FIFO destinada a uma porta de saída (fig. 3.5). Porém, esta implementação apresenta algumas desvantagens. A primeira delas,e mais evidente, é o aumento da fragmentação das FIFO's e o aumento do custo, em termos de área. Outra desvantagem é que as FIFO's de uma determinada entrada disputam entre si o acesso à *crossbar*, apesar de destinadas a saídas diferentes. Uma terceira desvantagem é que a sinalização de controle para o estágio anterior deve ser feita separadamente para cada *buffer*, permitindo que os *buffers* que ainda não estejam cheios continuem a receber dados. Isto implica em um aumento das linhas de controle. Apesar das desvantagens apresentadas, esta implementação possui a vantagem de desacoplar as saídas, isto é, uma saída bloqueada não interfere no tráfego das demais saídas.

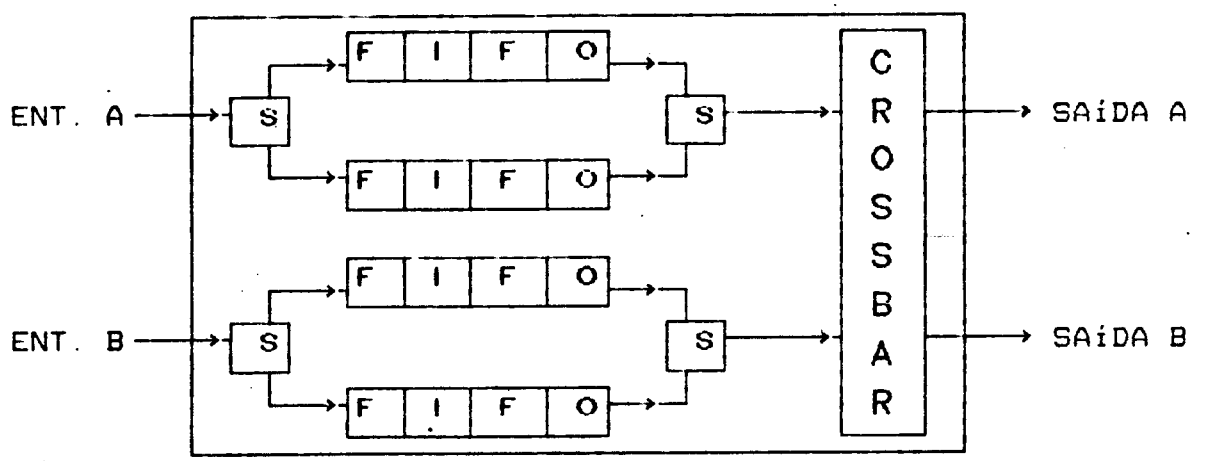

fig. 3.5 - Chave com *Buffer* Separado na entrada (CROSSBAR)

3.2.6 - CHAVE COM BUFFER SEPARADO NA ENTRADA (MULTIPLEX)

A partir do momento em que se tem *buffers* múltiplos para cada porta de entrada, uma chave crossbar 2x2 (supondo 2 entradas e 2 saídas) não suporta todas as possibilidades de interconexão *buffer*/saída. Para este caso (4 *buffers* e 2 saídas), seria necessário uma chave

crossbar 4x2. Uma outra alternativa seria agrupar os *buffers* destinados à mesma saída, e cada grupo independente dos demais (fig. 3.6). Esta solução elimina o problema de disputa pelo acesso à crossbar, sem aumentar a complexidade da chave. O controle é distribuído para as saídas, tornando o tráfego de cada saída independente das demais. Além disto, a implementação de multiplexadores é mais simples do que a implementação de uma chave crossbar 4x2. Porém, os problemas relativos ao aumento da fragmentação dos *buffers* e aumento das linhas de controle continuam existindo.

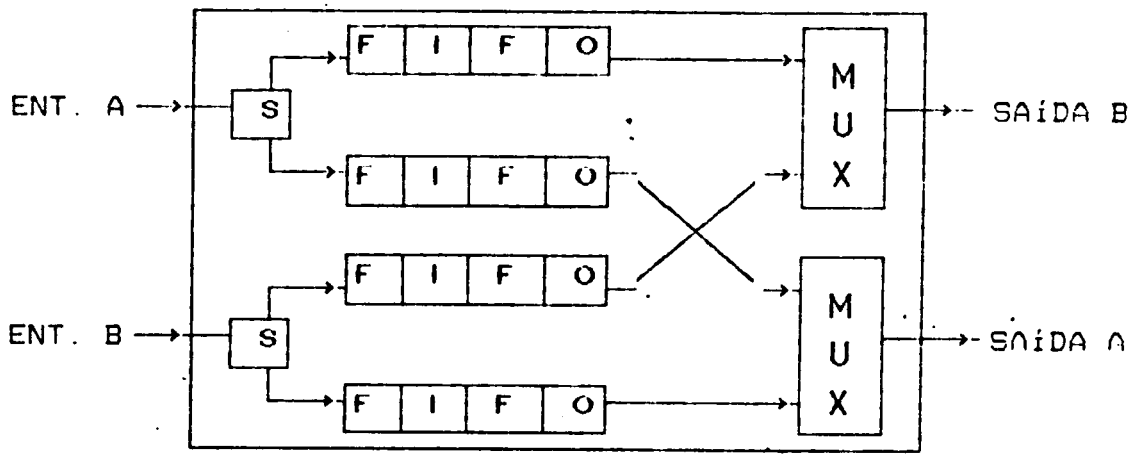

fig. 3.6 - Chave com *Buffer* Separado na Entrada (Multiplex)

4 - PROBLEMAS ENVOLVENDO A REDE

4.1 - RECOMBINAÇÃO

Em sistemas de processamento paralelo com dezenas ou até centenas de processadores, a contenção causada pelo acesso à posições de memória compartilhadas pode se tornar um problema sério. Uma tentativa de se reduzir este problema consiste em dotar as chaves com capacidade de combinar mensagens destinadas à uma mesma posição de memória. Com isto, um número grande de acessos concorrentes seria eliminado sendo processados pela própria chave.

O efeito causado pela contenção pode congestionar o tráfego de toda a rede, independente de sua topologia e seu modo de chaveamento [Pfis85a]. A recombinação é recomendada, portanto, quando as contenções na memória tendem a degradar demasiadamente o sistema. No entanto, existem 2 desvantagens associadas à recombinação. A primeira, e mais imediata, é que a complexidade e o custo da chave aumentam muito (de 6 a 32 vezes [Pfis85a]). Uma outra desvantagem é que, com o aumento da complexidade interna da chave, o tempo médio necessário para a transmissão de uma mensagem aumenta. Com isso, mesmo os acessos que não necessitam de recombinação são penalizados.

4.2 - *HOT SPOTS* E SATURAÇÃO EM ÁRVORE (*TREE SATURATION*)

Quando um número grande de processadores compartilha uma determinada posição de memória, chamada também de *hot spot*, não apenas a contenção causada pode degradar seriamente o desempenhodo sistema, mas também pode causar um fenômeno na rede chamado de saturação em árvore [Yew 87]. Este problema afeta não somente os acessos aos *hot spots*, mas também todos os outros acessos através da rede. Em resumo, um *hot spot* mesmo com uma percentagem pequena de acessos (menos de 10%) [Yew 87], pode congestionar o tráfego de toda a rede. Uma solução para o problema acima é a utilização de recombinação nas chaves, descrita no item anterior. Porém, esta solução possui um custo muito alto, podendo inviabilizar um projeto. Uma solução alternativa, chamada de *software combining tree* (árvore de combinação) [Yew 87], consiste em distribuir os *hot*

spots pela memória. Ao invés de se utilizar um único *hot spot* para *n* processadores, utiliza-se um número maior de *hot spots*, cada um sendo acessado por um número menor de processadores (fig. 4.1). A idéia é semelhante à da recombinação, porém é implementada em *software*. O *hot spot* original é dividido em vários *hot spots* e são associados à uma estrutura em árvore. Esta solução se mostra especialmente atraente pela sua simplicidade de implementação e custo zero.

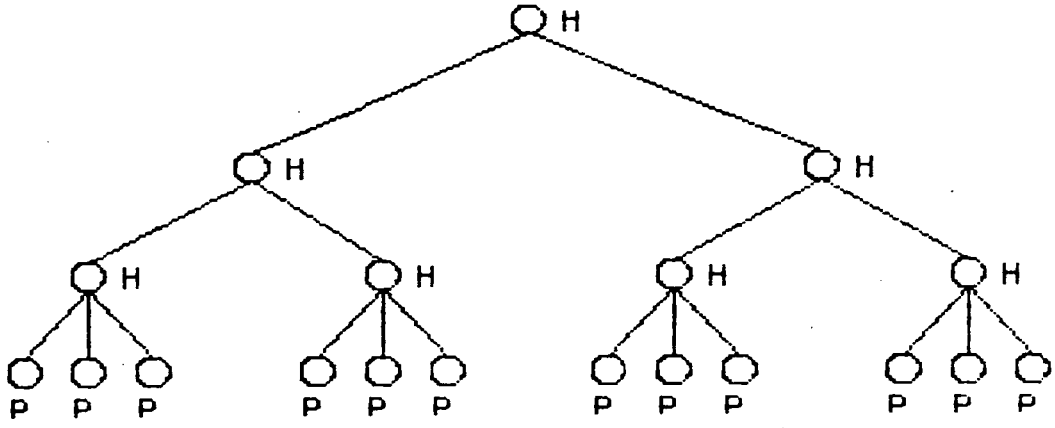

fig. 4.1 - *Software Combining Tree*

4.3 - BROADCASTING

Broadcasting é a capacidade que uma rede possui de um único processador poder enviar uma mensagem para *n* outros processadores simultaneamente. Isto é desejável, por exemplo, para sincronização de processos em *n* processadores. Para que a rede possua esta capacidade é necessário que as chaves possam assumir as seguintes configurações:

fig. 4.2 *Broadcast*

A presença de *broadcast* diminui o número de acessos à rede, porém só é justificável se a percentagem de *broadcast-messages* for grande em comparação com o total de mensagens. Além do mais, como a rede é, em geral, bloqueável, nada garante que as *n* mensagens chegarão aos seus destinos simultaneamente. Outro ponto negativo é a necessidade de se incluir na mensagem uma tag adicional para informar quais os estágios que devem realizar *broadcast*. Uma solução trivial e com custo zero, é a substituição de uma *broadcast-message* para n processadores por *n* mensagens normais.

5 - REVISÃO DAS ARQUITETURAS EXISTENTES

Neste capitulo é feita uma revisão das arquiteturas de algumas máquinas existentes. é apresentado um resumo da arquitetura global com uma ênfase maior na arquitetura da rede de interconexão de cada uma delas. São vistas 3 máquinas MIMD (BBN Butterfly, NYU Ultracomputer e IBM RP3) e 1 máquina SIMD (Illiac IV).

5.1 - BBN BUTTERFLY [Butt85][Jess88]

O BBN Butterfly é um multiprocessador homogêneo e escalonável, podendo ter de 1 a 256 elementos processadores (EP's). A memória é local a cada EP (não existem módulos de memória independentes), porém, cada EP pode acessar qualquer memória remota através da rede de interconexão. Cada EP é composto por uma CPU Motorola MC68000, pelo menos 1 Mbyte de memória, gerência de memória virtual e interface com a rede de interconexão.

A rede de interconexão utiliza chaveamento por pacotes e é topologicamente uma rede Banyan. Cada elemento comutador é uma chave 4 x 4 necessitando, portanto, de 2 *bits* por estágio para efetuar o roteamento das mensagens. Cada caminho existente através da rede suporta uma taxa de comunicação de até 32 Mbits/s e a largura de cada mensagem é de 1 *bit*. No caso de conflito nas chaves, uma das mensagens é transmitida e a outra é automaticamente retransmitida após um pequeno intervalo de tempo. Uma rede contendo caminhos alternativos entre cada par origem/destino pode ser construida adicionando-se chaves extras. Atualmente, sistemas com mais de 16 EP's são configurados para ter caminhos redundantes. As chaves não possuem recombinação e as operações atômicas são suportadas pelos EP's.

5.2 - NYU ULTRACOMPUTER [Gott83][Jess88]

O NYU Ultracomputer é um multiprocessador de uso geral. Sua memória é totalmente compartilhada e os processadores se comunicam com ela através de uma rede omega. Todas as operações de sincronização são realizadas através da instrução atômica Fetch&Add (F&A).

A principal característica desta máquina é a presença de recombinação nas chaves. Se todos os EP's comandarem um F&A para uma posição *x* de memória, o *hardware* da chave combina estes pedidos e ao final, apenas um F&A é realizado em *x*. As chaves se encarregam de responder aos F&A's originais. Portanto, a sequência de somas dos F&A's originais é mantida, mesmo que a ordem de chegada (que não é importante) não seja conhecida. A utilização de um *hardware* especial para recombinação é uma inovação importante que pode aumentar a velocidade de execução de muitos programas.

5.3 - IBM RP3 [Pfi185][Pfi285][Pfi385]

O RP3 é uma máquina MIMD de alto desempenho composta por 512 elementos processadores (EP's) de 32 *bits*, de 1 a 2 Gbytes de memória. Cada EP é composto por um microprocessador de 32 *bits*, 2 a 4 Mbytes de memória, 32 Kbytes de *cache* e suporte para I/O e operações de ponto flutuante. O RP3 possui um esquema de mapeamento de memória que permite seu particionamento dinâmico entre memória local e global.

A comunicação entre os processadores é feita por 2 redes distintas. Uma delas (topologia omega) possui recombinação, como no NYU Ultracomputer (utilizando F&A). A outra rede (topologia Banyan) possui latência menor que a anterior, pois não tem lógica de recombinação. Desta forma, apenas os acessos à posições compartilhadas de memória são penalizados com um atraso maior, através da rede com recombinação. Os acessos normais (posições não compartilhadas) são enviados pela outra rede.

5.4 - ILLIAC IV

O Illiac IV é uma máquina SIMD composta de 2 unidades básicas: a Unidade de Controle (UC) e os Elementos Processadores (EP). A UC é a responsável pelo controle da matriz de EP's (8 x 8) e também pela execução de instruções escalares (as instruções vetoriais são executadas na matriz de EP's). Cada EP é composto por uma sofisticada ALU e por 2048 x 64 *bits* de memória RAM. A memória de cada EP é estritamente local, não podendo ser acessada por outro EP.

A comunicação entre os EP's é feita através de uma rede estática do tipo *nearest-neighbour mesh* (vizinho mais próximo). Como os EP's estão organizados em uma matriz 8 x 8, cada PE_i pode se comunicar diretamente com PE_{i+1}, PE_{i-1}, PE_{i+8} e PE_{i-8}. O roteamento para qualquer outro PE é feito através de combinações de roteamentos de ±1 e ±8.

6 - CARACTERÍSTICAS PARTICULARES DO PROJETO MULTIPLUS

6.1 - A ARQUITETURA GLOBAL DA REDE

Não existe muita variação na forma em que os processadores são ligados às redes de interconexão. Normalmente, cada processador é ligado à uma porta de entrada da rede. No caso da memória estar distribuída entre os processadores, estes são conectados a ambos os lados da rede. Já no caso de não haver memória local aos processadores, um lado da rede é ligado aos processadores e o outro, as unidades de memória. A solução adotada pela IBM, no RP3, foi a utilização de 2 redes distintas: uma com lógica de recombinação e a outra sem recombinação, para não penalizar os acessos normais. Já o NYU Ultracomputer utiliza apenas uma rede com recombinação. No caso do projeto Multiplus, existe uma diferença fundamental com relação às outras máquinas. Na realidade, a cada porta de entrada da rede está ligado um barramento duplo que pode conter até 8 processadores. Com esta arquitetura proposta, pode-se obter as vantagens de ambas as implementações (rede de interconexão e barramento), e evitar as suas principais desvantagens. E a partir deste fato, surgem algumas outras alternativas de arquitetura da rede de interconexão, que serão descritas a seguir.

6.1.1 - INTERFACE ÚNICA E REDE ÚNICA

Nesta implementação, os 2 barramentos estão conectados apenas a uma interface de rede (IR) e esta interface, conectada apenas a uma rede de interconexão (RI) (fig. 5.1). Os comandos que chegam através dos 2 barramentos são enfileirados, pela ordem de chegada, na própria IR e são enviados através da rede única. O fato de, tanto a rede como a interface, serem únicas, implica na existência de apenas um caminho entre qualquer par origem/destino. Isto garante automaticamente a não violação do principio da serialização [Gott83]. Como desvantagem, pode-se citar o fato desta arquitetura não aproveitar a existência de 2 barramentos independentes. Embora possam chegar 2 comandos ao mesmo tempo, apenas 1 de cada vez é transmitido através da rede.

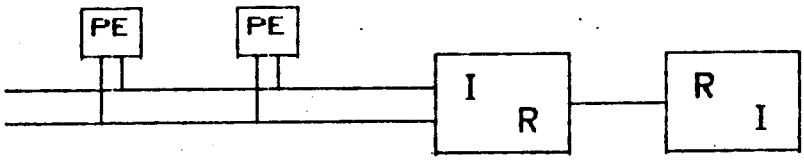

fig. 5.1 - Interface Única e Rede Única

6.1.2 - INTERFACE E REDE INDEPENDENTES PARA CADA BARRAMENTO

Nesta implementação, cada barramento possui uma IR e uma RI independentes (fig. 5.2). Portanto, os comandos que chegam através de cada barramento são armazenados independentemente nas suas respectivas IR's. Pelo fato de haver 2 redes distintas, pode-se ter o princípio da serialização violado. Suponha que 2 mensagens A e B cheguem, nesta ordem, uma em cada barramento (ambas as mensagens tem o mesmo endereço de origem e destino). Como existem 2 caminhos possíveis (cada rede possui 1 caminho) para cada par origem/destino, pode-se ter o caso em que cada mensagem é transmitida por uma das redes. Porém, nada pode ser dito com relação ao tempo que cada mensagem vai levar para ser transmitida (pode-se ter condições de tráfego completamente diferentes em cada uma das redes). Portanto, pode-se ter uma inversão da ordem das mensagens na chegada (a mensagem B chega antes da mensagem A). Este problema pode ser contornado especializando-se os barramentos, por exemplo, um dos barramentos só para dados e outro só para instruções. Isto garantiria o princípio da serialização mas, outra vez, não aproveitaria totalmente as potencialidades de 2 barramentos independentes.

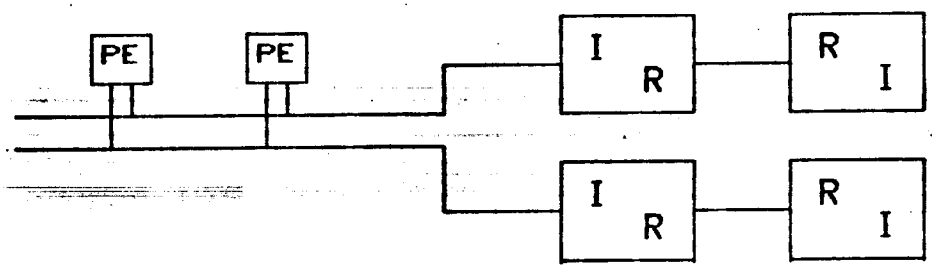

fig. 5.2 - Interface Dupla e Rede Dupla

6.1.3 - INTERFACE ÚNICA E REDE DUPLA

Uma terceira solução, que engloba as características positivas das 2 soluções anteriores é a utilização de uma IR única ligada a 2 redes independentes (fig. 5.3). O princípio da serialização pode ser mantido já que o controle de recebimento e envio das mensagens é centralizado. Também aproveita o fato de haver 2 barramentos . Neste caso, não há a necessidade de se especializar os acessos pois a IR fica encarregada de garantir o princípio da serialização. Uma característica desejável com relação as 2 redes independentes é que elas sejam topologicamente equivalentes do ponto de vista do particionamento. Ou seja, é desejável que as as 2 redes possam ser particionadas igualmente.

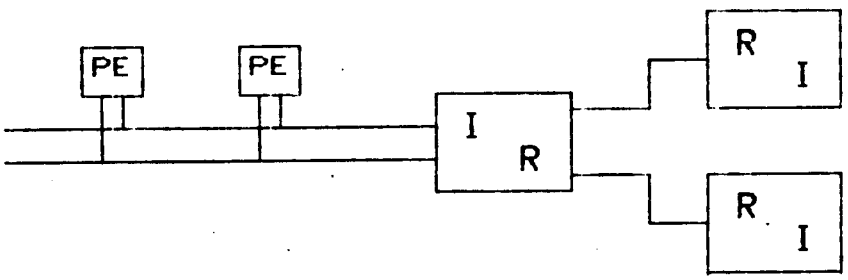

fig. 5.3 - Interface única e Rede Dupla

6.2 - O BARRAMENTO E OUTRAS CONSIDERAÇÕES

Como já foi dito anteriormente, a arquitetura mixta adotada no projeto Multiplus, com barramento e rede de interconexão, permite o aproveitamento das principais vantagens destas duas implementações, evitando as suas principais desvantagens. No caso do barramento, aproveita-se o fato da comunicação entre os diversos processadores ser bem mais rápida do que através da rede. Porém, a medida que o número de processadores cresce, o tráfego no barramento cresce também, até o momento em que a comunicação se torna inviável (congestionamento do barramento). Como o acesso através da rede é bem mais lento que o acesso através do barramento, seria desejável que o barramento ficasse livre enquanto a comunicação através da rede é completada. Para isso, é necessário que o barramento seja liberado logo após o pedido de comunicação e que, ao ser completado o acesso através da rede, o processador seja avisado. Durante o tempo entre o pedido e a resposta, o barramento pode ser utilizado

por outros processadores. Mas isto só acontece efetivamente se a interface de rede (IR) possuir um *buffer* de comandos, pois de nada adianta o barramento estar livre se a IR não puder receber comandos.

No caso de leitura através da rede, o processador deve aguardar a chegada do dado lido. No caso de escrita isto não é necessário, pois o processador não precisa aguardar que o dado seja escrito para continuar o processamento. Caso haja erro na escrita, o problema é bem mais sério, pois o processador só vai ser avisado do erro algum tempo depois. A gerência deste tipo de escrita (*write buffer*) é bem mais complexa e nem sempre é eficiente. Pode haver o caso em que o processo que gerou o erro não esteja mais sendo executado, e portanto o aviso de que houve erro perde o sentido.

6.3 - CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo principal a elaboração de um estudo inicial sobre redes de interconexão visando a definição de uma arquitetura de rede para o projeto Multiplus. Os dois primeiros capítulos apresentaram o assunto de forma conceitual, fornecendo algumas definições importantes. No capítulo 4 foram apresentados, de forma mais prática, alguns dos principais problemas envolvendo redes de interconexão. A utilização de recombinação nas chaves apareceu como solução para o problema de *hot-spots*, porém o aumento do custo associado a esta solução nos levou a considerar a utilização de *software combining tree* como uma solução mais atraente. No capítulo 3 foram apresentadas diversas estruturas de chaves, bem como as suas principais vantagens e desvantagens. Como continuação deste trabalho, sera feito um estudo de simulação das arquiteturas apresentadas visando definir qual delas apresenta melhor desempenho.

7 - REFERÊNCIAS

[Alma89] Almasi, G. S. e Gottlieb, A.
"Highly Parallel Computing"
Benjamin/Cummings Publishing, 1989

[Bhuy89] Bhuyan, L. N. et al
"Performance of Multiprocessor Interconnection Networks"
IEEE Computer, Fev. 1989

[Bouk72] Bouknight, W. J. et al
"The Illiac IV System"
Proceedings of the IEEE, vol. 60, Abr. 1972

[Butt85] -
"Butterfly Parallel Processor Overview"
BBN Laboratories Incorporated, 1985

[Dias81] Dias, D. M. e Jump, J. R.
"Packet Switching Interconnection Networks for Modular Systems"
IEEE Computer, Dez. 1981

[Feng81] Feng, T. Y.
"A Survey of Interconnection Networks"
IEEE Computer, Dez. 1981

[Gott83] Gottlieb, A. et al
"The NYU Ultracomputer - Designing a MIMD Shared Memory Parallel Computer"
IEEE Trans. Comp., vol. C-32, Fev. 1983

[Hwan85] Hwang, K. e Briggs, F. A.
"Computer Architecture and Parallel Processing"
McGraw-Hill, 1985

[Jess88] Jesshope, C. R. et al
"Parallel Computers vol. 2"
IOP Publishing Ltd., 1988

[Jump81] Dias, D. M. e Jump, J. R.
"Analysis and Simulation of Buffered Delta Networks"
IEEE Trans. Comp., vol. C-30, Abr. 1981

[Mena] Menascé, D. A. e Barroso, L. A.
"A Methodology for Performance Evaluation of Parallel Aplications on Multiprocessors"

[Pfis85a] Pfister, G. F. e Norton, V. A.
"Hot-Spot Contention and Combining in Multistage Interconnection Networks"
IBM Research Report, 1985

[Pfis85b] Pfister, G. F.
"The Architecture of the IBM RP3"
IBM Research Report, 1985

[Pfis85c] Pfister, G. F. et al
"An Introduction to the IBM RP3"
IBM Research Report, 1985

[Sieg79] Siegel, H. J.
"A Model of SIMD Machines and a Comparision of Various Interconnection Networks"
IEEE Trans. Comp. vol., C-28, Dez. 1979

[Tami88] Tamir, Y. e Frazier, G. L.
"High-Performance Multi-Queue Buffers for VLSI Communication Switches"
IEEE, 1988

[Yew 87] Yew, P. C. et al
"Distributing Hot-Spot Addressing in Large-Scale Multiprocessors"
IEEE Trans. Comp., vol. C-36, Abr. 1987.